# EcoStruxure™ Machine SCADA Expert Technical Reference Manual

07/2020

EIO0000002448_07

www.schneider-electric.com

Schneider Electric™

## Contents


# INTRODUCTION ..... 17

Conventions used in this documentation ..... 19
About this software ..... 21
Internal structure and data flow ..... 24
Executing and switching modules ..... 27
Executing and switching the Background Task ..... 31

# INSTALLATION GUIDE ..... 33

About the EcoStruxure Machine SCADA Expert 2020 software components ..... 34
Install the full EcoStruxure Machine SCADA Expert 2020 software ..... 35
    Install .NET Framework 3.5 in Windows Server 2012 R2 ..... 39
Install the Thin Client software ..... 41
    Install the Custom Widget Framework on a client station ..... 42

# LICENSING ..... 44

License Settings ..... 45
    Product Versions ..... 46
    Execution Modes ..... 46
About hardkey licenses ..... 48
    Install a new hardkey license ..... 48
    Upgrade an existing hardkey license ..... 48
About softkey licenses ..... 50
    Install or upgrade a softkey license for the full EcoStruxure Machine SCADA Expert 2020 software ..... 50

# THE DEVELOPMENT ENVIRONMENT ..... 52

Title bar ..... 53
Quick Access Toolbar ..... 54
File menu ..... 56
    New ..... 56
    Open Project ..... 58
    Open ..... 59
    Save ..... 59
    Save As ..... 59
    Save All ..... 59
    Save All as HTML ..... 59
    Save as HTML ..... 60
    Save Screen Group as HMTL ..... 60
    Close ..... 60
    Close All ..... 60
    Recent Projects ..... 60
    Print ..... 60
    Exit ..... 60

Ribbon....61
Home tab....61
View tab....61
Insert tab....62
Project tab....62
Draw tab....62
Format tab....63
Help tab....64
Project Explorer....65
Global tab....65
Graphics tab....66
Tasks tab....67
Comm tab....69
Screen/Worksheet Editor....70
Watch window....71
Output window....72
Status bar....73
Standard Interfaces....74
Object Properties dialog box....74
Color Interface....74
Fonts....78
ASCII Character Table....80
Performing Common Tasks....82
Accessing Projects and Files....82
Using Common Dialog Buttons....83
Convert your project's display resolution....83
Using Shortcut Menus....85
Using Select All....85
Cutting, Copying, Pasting Objects....85
Find text in the current document or entire project....85
Replace text in the current document....87
Finding System Information....88
Using the Tag Properties Toolbar....89
Replacing project tags in a document or screen object....92
Testing Displays....93
Verify the project....93
Running Projects....95
Restoring Defaults....95
Saving Your Work....95
Printing Screens and Worksheets....95
Focusing the Object Properties Window....95

**CREATING A NEW PROJECT....97**

Creating a new project....98
About target platforms, product types, and target systems....100
Changing the target system of an existing project....101
Configuring additional project settings....102
Information tab....103
Options tab....104
Viewer tab....117
Communication tab....122
Preferences tab....129
Configuring your project's default email settings....132
Configuring your project's default FTP settings....134
Runtime Tasks....136
Run a project as a Windows service....138

# TAGS AND THE PROJECT DATABASE.....143

About Tags and the Project Database.....144
Project Tags Folder.....145
About classes.....149
Shared Database folder.....151
System Tags Folder.....151
Designing a Tag.....154
Naming the Tag.....154
Choosing the Tag Type.....154
Choosing the Tag Data Type.....156
Choosing the Tag Scope.....156
Creating Database Tags.....157
Adding Tags to the Datasheet.....157
Creating Tags "On-the-Fly".....158
Editing Tags.....159
About classes.....161
Tag Properties.....164
Set tag properties using the Project Tags datasheet.....165
Set tag properties using the Properties command.....167
Reference a tag property instead of a project tag.....169
Using TagsDB functions to edit the tags database during run time.....169
Properties of Integer and Real tags.....171
Properties of Boolean tags.....176
Properties of String tags.....180
Complete list of tag properties.....181
Change how out-of-range tag values are handled.....187
Using Tags in Your Project.....189
Deleting a tag from the project database.....190
Sort or filter the rows in a worksheet.....191
Using the Tags tools.....195
Global Replace Tool.....195
Replacing project tags in a document or screen object.....196
Removing unused tags from the project database.....197
Reset Tags Database.....198
Tagname Text Box.....199
Object Finder Tool.....199
Cross Reference Tool.....200
Set tag properties using the Properties command.....200
Import Wizard.....203
Import tags and files from a EcoStruxure Machine SCADA Expert 2020 project database.....206
Importing from OPC Server Databases.....210
Import tags from a CSV database.....211
Importing from ODBC Databases.....215
Importing from PanelBuilder32 Databases.....216
Importing PanelMate programs.....217
Import from a FactoryTalk application.....219
Import a Studio XML Screen.....222
Tag Integration.....225
Using TagsDB functions to edit the tags database during run time.....226

# SCREENS AND GRAPHICS.....229

Graphics tab ... 230
**Screens** folder ... 231
Screen Group Folder ... 236
Lay out project screens in a simulation of the client's display ... 237
Screen Objects and Animations ... 239
Editing ... 239
Shapes ... 242
Active Objects ... 251
Libraries ... 277
Applying animations to screen objects ... 315
Use custom properties to set property values when screens are opened ... 330
Format tab ... 334
Change the properties of multiple screen objects ... 334
Set the tab order of screen objects ... 336
Bring to front / Send to back ... 337
Group and ungroup screen objects ... 338
Align, Center and Distribute Tools ... 340
Rotate Tool ... 342
Resize Tools ... 343
Fill Color Tool ... 343
Line Color Tool ... 344
Fonts Tool ... 344
Data Input ... 345
Data input in screens on Thin Clients ... 345
Data input in screens on Mobile Access ... 349
Multi-Touch ... 351
About the Multi-Touch settings for project screens ... 352
About the different types of multi-touch gestures ... 358
About Touch Events ... 364
Import a Studio XML Screen ... 373

# ALARMS, EVENTS, AND TRENDS ... 376

Alarm worksheet ... 377
Alarm Worksheet Header ... 378
Alarm Worksheet Body ... 382
Saving your alarm history / event log to an external database ... 387
Format of the alarm history ... 389
Events ... 394
Enable the event logger ... 394
Saving your alarm history / event log to an external database ... 396
Format of the event history ... 398
Alarm/Event Control object ... 401
Customize the audible alarm ... 411
Trend worksheet ... 413
Sort or filter the rows in a worksheet ... 416
Creating Batch History ... 419
Converting Trend History Files from Binary to Text ... 421
Converting Trend History Files from Text to Binary ... 422
Make trend history accessible through OPC HDA ... 423
Trend Control object ... 424
About the trend control runtime interface ... 424
*Object Properties: Trend Control* dialog ... 427
Using the Data Source Text File ... 449
Using the Data Source Database ... 452
Display text- and image-based trend annotations in a trend control ... 459

Grid object....463
Data dialog....464
Columns dialog....467
Advanced dialog....471

INDUSTRIAL GRAPHICS....475

Create a new Industrial Graphics screen....476
Create a new Industrial Graphics symbol....478
Create a new Industrial Graphics toolset....479
Import an Industrial Graphics symbol library....480
Embed an Industrial Graphics symbol in a screen....481
Using project tags in Industrial Graphics screens....483
Known limitations of Industrial Graphics....484

BACKGROUND TASKS....486

Alarm worksheet....487
Trend worksheet....489
Recipes....492
Report worksheet....494
Create a new Math worksheet....496
About the Built-in Language interface....497
Using the Goto…Label structure in a Math worksheet....500
Using the For…Next loop in a Math worksheet....502
Script worksheet....504
Startup Script worksheet....505
Scheduler worksheet....506
Database/ERP worksheet....508
Sort or filter the rows in a worksheet....513

COMMUNICATION....517

Configuring direct communication with a remote device....518
Main Driver Sheet....525
Standard Driver Sheets....527
Read/write status codes for direct communication drivers....530
Tag Integration....534
Integrate tags from TwinCAT....534
Integrate tags from CoDeSys....537
Integrate tags from RSLogix 5000 Family....545
Integrate tags from Allen-Bradley PLC5, SLC500....547
Integrate tags from AutomationDirect Do-more H2 Series....550
Integrate tags from AutomationDirect Koyo....552
Integrate tags from AutomationDirect P Series....555
Integrate tags from AutomationDirect PAC 3000....558
Add GE PACSystems RX3i as a tag integration source....561
Integrate tags from Schneider Unity Modbus....564
Integrate tags from Siemens S7-1200/S7-1500....567
Add an OPC DA server as a tag integration source....569
Add an OPC UA server as a tag integration source....571
Use the Object Finder to select integrated tags....573
How integrated tags may be renamed in your project....575

OPC Clients and Servers....576
OPC UA Client....576
OPC XML/DA Client....605
OPC DA 2.05 Client....616
Tag Expansion for OPC Clients....620
Array Distribution for OPC Clients....622
OPC UA Server....624
OPC DA 2.05 Server....635
Array Distribution for OPC Clients....635
Communicate with another project runtime server....637

**PROJECT SECURITY....639**

About security modes....640
About security access levels....641
Using the Security System Configuration Wizard....643
Configuring server settings for security modes....649
Extending the LDAP schema to allow saving of security rights....656
Group Account dialog....664
Creating and configuring users....671
Security System dialog....673
Backing up the security system configuration....675
Logging on/off....677
Blocking or unblocking a user....678
Password-protecting screens, symbols, and worksheets....679

**PROJECT LOCALIZATION....681**

Add a target language to the Translation Table....682
Configure fonts for a target language....684
Examples of font configuration....685
Set the project's language at startup....687
Set the project's language during run time....688
Disable translation of selected screen objects....690
Configure the advanced translation settings....691
Import a legacy translation file into the Translation Table....693
About the date format and how to change it....694

**DEBUGGING TOOLS....696**

Watch window....697
Using the Watch tool....697
Opening the *Watch* page for Mobile Access....699
Output window....703
Configure the log settings for the Output window....703
Save log messages from the Output window to a file....709
About the LogWin tool....712
Open the LogWin tool....712
Configure the log settings for the LogWin module....714
Save log messages from the LogWin tool to a file....718

**REMOTE MANAGEMENT....719**

Enable security in Remote Agent and add users....720
Customize Remote Agent's encryption key....721
Download your project to the target device....723
Run or stop your project on the target device....725
Configure Remote Agent to autorun a project....726

THIN CLIENTS AND MOBILE ACCESS....727

Thin Clients....729
The Underlying Technology....729
Examples of Client/Server Architecture....730
Configuring the Data Server....734
Configuring a web server to host your project pages....736
Install the Thin Client software....740
Configure and run Secure Viewer....742
Implementing Security for Web-based Applications....746
List of network ports used by this software....751
View or disconnect client sessions....753
Mobile Access....756
Supported features in Mobile Access....756
Tips for Mobile Access development and run time....766
Mobile Access web server add-on....768
Configuring the Mobile Access web interface....777
Navigating the Mobile Access web interface....791
Troubleshooting project screens in Mobile Access....803
View or disconnect client sessions....812

DATABASE INTERFACE....815

SQL Relational Databases....816
Studio Database Gateway....818
Manually install Studio Database Gateway....826
Manually running Studio Database Gateway....828
Database Configuration....831
Configuring a Default Database for All Task History....836
Support for AVEVA Insight and Historian....837
Connect to AVEVA Insight using AVEVA Insight Publisher....837
Connect to a Historian database located on-premises....841
Database Troubleshooting....844
Appendices....848
Using ODBC Databases....848
Using Microsoft SQL Server....849
Using Oracle Databases....852
Using Microsoft Access or Microsoft Excel....853
Using Sybase....855
Using MySQL....855

TROUBLESHOOTING....857

General Troubleshooting....858
Frequently Asked Questions....860
Proxy Settings....866
Configure the proxy settings on a Windows computer or device....866

Help tab.... 868
Help.... 868
Communication Drivers.... 869
License Agreement.... 869
Product Web Site.... 869
Release Notes.... 869
Support.... 869
About.... 870

TUTORIAL: BUILDING A SIMPLE PROJECT.... 871

Creating a new project.... 872
Specifying the startup screen.... 874
Creating tags.... 875
Creating the main screen.... 877
Drawing the main screen's title.... 879
Drawing a button to open another screen.... 881
Saving and closing the main screen.... 882
Creating the synoptic screen.... 883
Drawing the synoptic screen's title.... 883
Drawing "Date" and "Time" displays.... 883
Placing an "Exit" icon.... 885
Testing the project.... 886
Placing an animated tank.... 887
Placing a level slider.... 889
Drawing a tank selector.... 890
Testing the project.... 891
Configuring the communication driver.... 892
Monitoring device I/O during run time.... 895

APPENDIX: BUILT-IN LANGUAGE.... 896

Logic and arithmetic operators.... 897
String expressions.... 899
How to read function descriptions.... 902
List of available functions.... 904
ActiveX and .NET Control functions.... 912
XGet.... 912
XRun.... 912
XSet.... 913
Arithmetic functions.... 915
Abs.... 915
Div.... 915
Format.... 916
GetBit.... 919
Mod.... 920
Pow.... 920
ResetBit.... 921
Round.... 922
SetBit.... 922
Sqrt.... 923
Swap16.... 923
Swap32.... 924
Trunc.... 924

Database/ERP functions....926
DBCursorClose....926
DBCursorColumnCount....927
DBCursorColumnInfo....929
DBCursorCurrentRow....931
DBCursorGetValue....932
DBCursorMoveTo....933
DBCursorNext....935
DBCursorOpen....936
DBCursorOpenSQL....939
DBCursorPrevious....942
DBCursorRowCount....943
DBDelete....945
DBExecute....947
DBInsert....949
DBSelect....951
DBUpdate....954
SyncAlarm....955
SyncAlarmStatus....956
SyncEvent....957
SyncEventStatus....958
SyncTrend....958
SyncTrendStatus....959
Date & Time functions....961
ClockGetDate....961
ClockGetDayOfWeek....962
ClockGetTime....963
DateTime2Clock....964
DateTime2UTC....965
GetClock....967
GetTimeZone....967
GetTimeZoneCount....969
GetUTC....969
Hour2Clock....969
SetSystemDate....970
SetSystemTime....971
SetTimeZone....971
UTC2DateTime....972
Email functions....974
CnfEmail....974
GetStatusSendEmailExt....976
SendEmail....976
SendEmailExt....978
Event Logger functions....980
SendEvent....980

File functions....982
DeleteOlderFiles....982
DirCreate....983
DirDelete....983
DirLength....984
DirRename....985
FileCopy....986
FileDelete....987
FileLength....988
FileReadFields....988
FileReadMessage....990
FileRename....991
FileWrite....992
FileWriteFields....993
FileWriteMessage....994
FindFile....996
FindPath....997
GetFileAttributes....998
GetFileTime....999
GetHSTInfo....1000
GetLine....1001
HST2TXT....1003
HST2TXTIsRunning....1006
ImportXML....1006
LookupContains....1008
LookupGet....1009
LookupLoad....1009
PDFCreate....1010
Print....1011
RDFileN....1012
WebGetFile....1013
FTP functions....1015
CnfFTP....1015
FTPGet....1016
FTPPut....1017
FTPStatus....1018
Graphic functions....1021
AutoFormat....1021
GetScrInfo....1021
PrintSetup....1023
PrintWindow....1023
ResetDecimalPointsTable....1025
RGBColor....1025
RGBComponent....1026
SaveScreenShot....1026
SetDecimalPoints....1028
SetDisplayUnit....1029
SetTagDisplayUnit....1030
Log Message functions....1031
Trace....1031
Logarithmic functions....1032
Exp....1032
Log....1032
Log10....1033
Logical functions....1034
False....1034
If....1034
Toggle....1035
True....1036

Loop functions ... 1037
For…Next ... 1037
Module Activity functions ... 1039
AppActivate ... 1039
AppIsRunning ... 1040
AppPostMessage ... 1041
AppSendKeys ... 1042
CleanReadQueue ... 1042
CloseSplashWindow ... 1043
DisableMath ... 1043
EnableMath ... 1044
EndTask ... 1044
Exec ... 1045
ExecIsRunning ... 1047
ExitWindows ... 1048
IsScreenOpen ... 1048
IsTaskRunning ... 1049
IsViewerInFocus ... 1050
KeyPad ... 1051
LogOff ... 1053
LogOn ... 1053
Math ... 1054
PostKey ... 1054
Recipe ... 1056
Report ... 1057
RunGlobalProcedureAsync ... 1059
RunGlobalProcedureAsyncGetCurrent ... 1060
RunGlobalProcedureAsyncGetStatus ... 1061
RunGlobalProcedureOnFalse ... 1062
RunGlobalProcedureOnServer ... 1064
RunGlobalProcedureOnTrigger ... 1065
RunGlobalProcedureOnTrue ... 1066
RunVBScript ... 1067
SecureViewerReload ... 1068
SendKeyObject ... 1069
SetAppPath ... 1071
SetViewerInFocus ... 1071
SetViewerPos ... 1072
ShutDown ... 1073
StartTask ... 1073
TaskUpdateConfig ... 1074
ViewerPostMessage ... 1076
Multimedia functions ... 1077
Play ... 1077
Screen functions ... 1078
Close ... 1078
Open ... 1079
OpenPrevious ... 1082
ShowInplaceInput ... 1083
ShowMessageBox ... 1085

Security functions ... 1087
BlockUser ... 1087
CheckESign ... 1088
CheckSecurityLevel ... 1089
CreateUser ... 1090
ExportSecuritySystem ... 1091
GetLastESignUser ... 1092
GetSecuritySystemStatus ... 1093
GetUserFullName ... 1094
GetUserNames ... 1095
GetUserPwdAging ... 1096
GetUserState ... 1097
ImportSecuritySystem ... 1097
RemoveUser ... 1099
SetPassword ... 1100
SetUserGroup ... 1102
UnblockUser ... 1103
Statistical functions ... 1105
Avg ... 1105
Max ... 1106
Min ... 1106
Rand ... 1107
String functions ... 1108
Asc2Str ... 1108
CharToValue ... 1108
CharToValueW ... 1109
ClassMembersToStrVector ... 1110
DecryptData ... 1111
EncryptData ... 1112
NCopy ... 1113
Num ... 1113
Str ... 1114
Str2Asc ... 1115
StrCompare ... 1115
StrCompareNoCase ... 1116
StrFromInt ... 1116
StrFromReal ... 1117
StrFromTime ... 1118
StrGetElement ... 1119
StrLeft ... 1119
StrLen ... 1120
StrLower ... 1120
StrRChr ... 1121
StrRight ... 1121
StrSetElement ... 1122
StrStr ... 1123
StrStrPos ... 1123
StrTrim ... 1124
StrTrimAll ... 1125
StrUpper ... 1125
ValueToChar ... 1126
ValueWToChar ... 1126

System Info functions....1129
DBVersion....1129
GetAppHorizontalResolution....1129
GetAppPath....1130
GetAppVerticalResolution....1130
GetComputerIP....1130
GetComputerName....1131
GetCursorX....1131
GetCursorY....1132
GetDisplayHorizontalResolution....1132
GetDisplayVerticalResolution....1132
GetHardKeyModel....1133
GetHardKeySN....1133
GetIPAll....1134
GetNetMACID....1134
GetOS....1135
GetPerformanceMetric....1135
GetPrivateProfileString....1138
GetProductPath....1139
GetRegValue....1139
GetRegValueType....1140
GetServerHostName....1141
GetTickCount....1141
InfoAppAlrDir....1142
InfoAppHstDir....1142
InfoDiskFree....1142
InfoResources....1143
IsActiveXReg....1143
IsAppChangedOnServer....1144
NoInputTime....1145
ProductVersion....1146
ReloadAppFromServer....1146
SaveAlarmFile....1147
SetAppAlarmPath....1147
SetAppHstPath....1148
SetDateFormat....1148
SetKeyboardLanguage....1149
SetRegValue....1151
SNMPGet....1152
SNMPSet....1153
WritePrivateProfileString....1154

Tags Database functions....1155
ExecuteAlarmAck....1155
ForceTagChange....1156
GetAlarmCount....1156
GetAlarmInfo....1157
GetTagValue....1159
SetTagValue....1160
TagsDBAddClass....1161
TagsDBAddClassMember....1161
TagsDBAddTag....1162
TagsDBBeginEdit....1164
TagsDBEndEdit....1164
TagsDBGetAlarm....1165
TagsDBGetClassMember....1167
TagsDBGetClassMemberCount....1168
TagsDBGetFirstClass....1168
TagsDBGetFirstClassMember....1169
TagsDBGetFirstTag....1170
TagsDBGetLoadStatus....1171
TagsDBGetNextClass....1172
TagsDBGetNextClassMember....1173
TagsDBGetNextTag....1174
TagsDBGetPreloadCount....1174
TagsDBGetTagCount....1176
TagsDBGetTagProperty....1176
TagsDBGetTrend....1178
TagsDBPreload....1178
TagsDBPreloadWait....1180
TagsDBRemoveAlarm....1182
TagsDBRemoveClass....1183
TagsDBRemoveClassMember....1183
TagsDBRemoveTag....1184
TagsDBRemoveTrend....1185
TagsDBSetAlarm....1186
TagsDBSetTagProperty....1188
TagsDBSetTrend....1190
TagsDBSync....1191
Translation functions....1193
Ext....1193
SetLanguage....1193
TranslationLoad....1194
TranslationLookupClose....1195
TranslationLookupGet....1196
TranslationLookupLoad....1197
Trigonometric functions....1199
ACos....1199
ASin....1199
ATan....1200
Cos....1200
Cot....1200
Pi....1201
Sin....1202
Tan....1202

**OVERVIEW OF VBSCRIPT....1204**

VBScript Interfaces in the Software....1205
Global Procedures....1206
Graphic Module....1210
Background Task....1219
Language Reference....1222
Operators....1222
Constants....1223
Objects and Collections....1225
Properties....1226
Statements....1226
Methods....1228
Functions....1228
Keywords....1229
Errors....1229
Tips & Tricks....1232
VBScript Editor IntelliSense....1232
VBScript Compared to VBA....1234
Screen Events....1235
MsgBox and InputBox Functions....1236
VBScript Procedures....1236
Creating Constants....1237
Declaring Variables....1238
Scope and Lifetime of Variables....1238
How Boolean tags are handled in VBScript....1238
Writing Real Values to Integer Tags....1239
Precedence of VBScript Operators....1240
Logical Operator NOT....1240
Using Conditional Statements....1241
Looping Through Code....1243
Support for ActiveX Controls....1245
Debugging VBScript....1246
About the Debug tab....1246
Set break points in your VBScript code....1247
Run your project in Debug mode....1248
Observe the current state in the Watch window....1250
Step through your VBScript code....1252

# Introduction

This *User Guide and Technical Reference* was designed to help you get the best results from your EcoStruxure Machine SCADA Expert 2020 software. This document provides technical information and step-by-step instructions for all the tasks you need to create web-enabled HMI/SCADA programs.

## Who should read this

This *User Guide and Technical Reference* is a comprehensive document designed to provide useful information for both novice and advanced users of EcoStruxure Machine SCADA Expert.

- **New Users**: This publication uses a step-by-step, hands-on approach to the project development process. Be sure to read the introductory chapters describing the product's features and development environment.
- **Experienced Users**: This publication offers *advanced* instructions, tips, and troubleshooting information to help you get the most out of your projects.

> **Note:** We assume you are familiar with working in a Windows environment, and we do not attempt to explain Windows navigation, file management, and so forth. If you are unfamiliar with any of these procedures, we recommend using the Windows Help feature (**Start** > **Get Help**) or consulting your Microsoft Windows documentation.

## Contents

The information in this document is organized into the following sections:

**Introduction**

This section, which provides an overview of the features and architecture of EcoStruxure Machine SCADA Expert 2020.

**Installation**

Step-by-step instructions for installing and uninstalling the full EcoStruxure Machine SCADA Expert 2020 software, the thin client software, and the project runtime software for a variety of platforms.

**Licensing**

Describes the licensing scheme for EcoStruxure Machine SCADA Expert 2020, as well as how to install and upgrade licenses on different platforms.

**The Development Environment**

A tour of the EcoStruxure Machine SCADA Expert development environment. Also, some basic skills and techniques you should understand before you create a new project.

**Creating a New Project**

Provides step-by-step instructions for creating and configuring a new project.

**Tags and the Project Database**

Explains basic concepts about the product database, tag types (arrays, classes, and pointers), tag values and parameters. Following the concepts discussion, this chapter provides instructions for creating and editing tags for your projects.

**Screens and Graphics**

Explains how to use the different EcoStruxure Machine SCADA Expert development tools to create your project screens and graphics.

**Alarms, Events, and Trends**

Explains how to create and configure task worksheets and screen objects to save and display historical data.

**Background Tasks**

Explains how to create and configure task worksheets for the other major background tasks.

**Communication with Other Devices**

Describes how to configure your project to read from and write to a device's registers, using a variety of communication drivers and protocols.

**Project Security**

Explains how to set-up and manage a security system for your projects.

**Project Localization**

Explains how to use the Translation Tool to translate the text in your projects from one language to another.

**Testing and Debugging**

Discusses how to test and debug projects using tools such as the Watch and Output windows. This chapter includes a list of possible error messages and methods for correcting those errors.

**Remote Management**

Explains how to download, monitor, and debug projects from a remote runtime workstation.

**Thin Clients and Mobile Access**

Explains how configure and run your projects on the Web.

**Database Interface**

Explains how to connect EcoStruxure Machine SCADA Expert to compatible databases.

**Troubleshooting**

Provides instructions for verifying projects, describes some common development errors, and explains what to do if you need to contact a support representative.

**Scripting Languages**

Describes EcoStruxure Machine SCADA Expert's built-in scripting language, as well as the support for VBScript in EcoStruxure Machine SCADA Expert.

## Related documentation

You may want to review the following manuals in addition to this Technical Reference:

- *Quickstart Guide*: Provides basic information about EcoStruxure Machine SCADA Expert 2020, including a systematic tutorial that allows you to develop a single project and become familiar with the product in a short time.
- *Tutorial Manual*: Describes how to build a project, step-by-step, with the main product features. You can use this document as a self-training manual.
- *Drivers User Guides*: Explain how to configure individual direct communication drivers, according to their unique protocol characteristics. One customized user guide is included with each driver.

**Note:** All manuals are located in the *Documentation* folder on the EcoStruxure Machine SCADA Expert installation CD. EcoStruxure Machine SCADA Expert installs the Drivers User Guides in the`\Drv` folder in the program directory. You also can access technical information from the **Help** menu.

## Conventions used in this documentation

This documentation uses standardized formatting and terminology to make it easier for all users to understand.

### Text conventions

This documentation uses special text formatting to help you quickly identify certain items:

- Titles, labels, new terms, and messages are indicated using *italic* text (for example, *Object Properties*).
- File names, screen text, and text you must enter are indicated using **`monospace`** text (for example, **`D:\Setup.exe`** ).
- Buttons, menu options, and keyboard keys are indicated using a **bold** typeface (for example, **File** menu).

In addition, this documentation segregates some text into **Tip**, **Note**, and **Caution** boxes:

- **Tips** provide useful information to save development time or to improve the project performance.
- **Notes** provide extra information that may make it easier to understand the nearby text, usually the text just before the note.
- **Cautions** provide information necessary to prevent errors that can cause problems when running the project, and may result in damage.

### Mouse and selection conventions

Because most PCs used for project development run a version of Microsoft Windows with a mouse, this documentation assumes you are using a mouse. Generally, a PC mouse is configured for right-handed use, so that the left mouse button is the primary button and the right mouse button is the secondary button.

This documentation uses the following mouse and selection conventions:

- **Click** and **Select** both mean to click once on an item with the left mouse button. In general, you click buttons and you select from menus and lists.
- **Double-click** means to quickly click twice on an item with the left mouse button.
- **Right-click** means to click once on an item with the right mouse button.
- **Select** also means you should use your pointing device to highlight or specify an item on the computer screen. Selecting an item with a touchscreen is usually the same as selecting with a mouse, except that you use your finger to touch (select) a screen object or section. To select items with your keyboard, you typically use the Tab key to move around options, the Enter key to open menus, and the Alt key with a letter key to select an object that has an underlined letter.
- **Drag** means to press down the appropriate mouse button and move the mouse before releasing the button. Usually an outline of the item will move with the mouse cursor.

### Windows conventions

This documentation uses the following Windows conventions:

- **Dialogs** are windows that allow you to configure settings and enter information.
- **Text boxes** are areas in dialogs where you can type text.
- **Radio buttons** are white circles in which a black dot appears or disappears when you click on the button. Typically, the dot indicates the option is selected or enabled. No dot indicates the option is cleared or disabled.
- **Check boxes** are white squares in which a check (

) appears or disappears when you click on it with the cursor. Typically, a check ☑ indicates the option is selected or enabled. No check ☐ indicates the option is cleared or disabled.
- **Buttons** are icons in boxes appear "pressed" when you click on them.
- **Lists** are panes (white boxes) in windows or dialogs containing two or more selectable options.
- **Combo boxes** have arrows that, when clicked, show part or all of an otherwise concealed list.

- **Dockable windows** are windows that you can drag to an edge of the interface and merge with that edge.

## About this software

EcoStruxure Machine SCADA Expert 2020 is powerful software for developing HMI, SCADA, and OEE/ Dashboard projects that can be deployed anywhere.

Each project consists of:

- A project tags database to manage all run-time data, including both internal variables and I/O data;
- Configurable drivers to communicate in real-time with programmable logic controllers (PLCs), remote I/O devices, and other data-acquisition equipment;
- Animated human-machine interface (HMI) screens and overall equipment effectiveness (OEE) dashboards; and
- Optional modules such as alarms, events, trends, recipes, reports, scriptable logic, schedulers, a project security system, and a complete database interface.

After you develop your project, you can either run it locally on your development workstation or download it to a remote computer and run it there. The project runtime server processes I/O data from connected devices according to your project parameters and then reacts to, displays, and/or saves the data.

### Product features

**ActiveX and .NET**

Use third-party controls to enhance your project. This software is a container for ActiveX and .NET controls. Add functionality such as browsers, media players, charting, and other tools that support the ActiveX and .NET interface standards.

**Alarms**

In addition to all of the alarm functions you would expect, this software also sends alarms using multi-media formats like PDF. Use remote notification to have alarms sent right to your email inbox, a printer, or a smartphone! Alarms are real-time and historical, log data in binary format or to any database.

**Animation**

This software gives you great command over graphics. Paste images, and even rotate them dynamically. Fill bar graphs with color, or adjust the scale of objects with easy-to-use configuration. Other animations include "command" (for touch, keyboard and mouse interaction), hyperlink, text data link, color, resize (independent height and width), position, and rotation (with custom rotation point).

**Database**

Connect to SQL databases (MS SQL, MySQL, Sybase, Oracle), MS Access and Excel, and ERP/ MES systems (including SAP), even from Windows Embedded. Flexible enough to have a built-in interface without the need to know SQL (for trends, alarms/events, grid and other objects), or use any SQL statement you need anywhere you need it.

**Drivers**

This software includes over 240 built-in communication drivers for most PLCs, temperature controllers, motion controllers, barcode/RFID readers, and other devices. Use these built-in drivers without the need for OPC servers (but are an optional connection method).

**Email**

Send email via SMTP to any desktop or mobile device. Get real-time information on alarms, process values, and other events. This software supports SSL encryption allowing the use of third-party providers such as Gmail.

**Events**

This software offers traceability for operator initiated actions or internal system activity. Log events such as security system changes (user logon or off), screen open/close, recipe/report

operations, custom messages and system warnings. Also any tag value changes including custom messages.

**FDA Traceability**

Take advantage of built-in traceability and e-signature features to create projects that fully comply with U.S. Food and Drug Administration regulations (Title 21 CFR Part 11). These features are often used in food and pharmaceutical applications, but they can be used in any application where traceability is required.

**FTP**

Automatically upload or download files during run time to/from remote storage locations using FTP and flexible scripting functions. Configure FTP via scripting or the included configuration interface.

**Graphics and Design Tools**

Create powerful screens to meet any application need using the improved tools in our graphic interface. Combine built-in objects to create any functionality required. Store graphics in the symbol library for future use. Easily make projects across a product line share a consistent "look and feel".

**Historical Performance**

We have optimized the trend history module and designed it to load millions of values from SQL relational databases with high performance, with built-in data decimation in the Trend Control. Easy-to-use tools provide quick access to Statistical Process Control (SPC) values without any need for programming.

**Intellectual Property Protection**

Screens, documents, scripts and even math worksheets can be individually password protected. This prevents unauthorized viewing or editing of your corporate custom functionality. Protect the entire project with just a few mouse clicks.

**Multi-Language**

Develop your project in one of many development languages, including English, Portuguese, German, and French.

**OPC**

As an alternative to the built-in drivers for direct communication with PLCs, you can also use any of several different versions of OLE for Process Control (OPC) to manage your devices. This software includes support for "classic" OPC DA (client or server), OPC UA (client or server), OPC XML-DA (client only), and OPC HDA (server only).

**PDF Export**

Send Alarms, Reports, or any file (including .doc or .txt) to a production supervisor, quality manager, or maintenance staff using the included PDF writer.

**Recipes**

Save time and maintain consistency by automating part parameters or productions quantities with any triggering event.

**Redundancy**

For critical applications where data is vital, this software supports web server, database and overall system redundancy.

**Reports**

Create clear, concise reports in text format, graphical RTF, XML, PDF, HTML, and CSV, or integrate with Microsoft Office. Get the data you need, in the format you need it, to make informed decisions, fast.

**Scalable**

Develop once and deploy anywhere, on any currently supported version of Microsoft Windows.

**Scheduler**

Schedule custom tag changes on date/time, frequency, or any trigger. Use this for simulation, to trigger reports or other functionality at a particular time of day, or even to trigger driver worksheets to read/write at a scan rate you choose.

**Scripting**

Two powerful scripting languages are supported. Use built-in functions or use standard VBScript to take advantage of widely available resources. Both can be used simultaneously to give you the functionality you need.

**Security**

This software provides support for group and user accounts, e-signatures, and traceability, as well as support for Lightweight Directory Access Protocol (LDAP). Integrate your project with your Active Directory, including Active Directory Application Mode (ADAM).

**SSL Support for Emails**

Native support for Secure Socket Layer (SSL), which makes it easy and secure to send emails from this software using third-party tools such as Gmail!

**Standards**

Take advantage of common industry standards to develop projects that are compatible with any format. TCP/IP, ActiveX/.NET, OPC (client and server), COM/DCOM, OLE, XML, SOAP, and HTML are all supported.

**SNMP**

Easily configure managed networked devices on IP networks (such as switches and routers) using incorporated SNMP configuration commands and an easy-to-use configuration interface.

**Symbols**

An extensive library of pre-made symbols features push buttons, pilot lights, tanks, sliders, meters, motors, pipes, valves and other common objects. Use the included symbols in your project, modify existing symbols to suit your needs, or create your own from scratch. Plus support for third-party symbol libraries and graphic tools.

**Tags Database**

This software features an object-oriented tags database with boolean, integer, real, strings, arrays, classes (structures), indirect tags, and included system tags.

**Thin Clients**

Remotely view project screens on several different types of thin clients. Use the standalone Secure Viewer to acheive the greatest security on plant-floor stations. Or use the HTML5-enabled Mobile Access to access your project from almost any other browser or mobile device.

**Trends**

Real-time and Historical trends are supported. Log data in binary format or to any database locally and remotely. Color or fill trends with graphic elements to enhance clarity of data. Date/Time based or numeric (X/Y plot) trends give you the flexibility to display information that best suits your project.

**Troubleshooting**

Quickly debug and verify a project using local and remote tools for troubleshooting, including status fields, Watch and LogWin. Capture screen open and close times, see communications in real-time, and messages related to OPC, recipes/reports, security, database errors and even custom messages. Quickly get your project finished using these powerful tools.

## Internal structure and data flow

The EcoStruxure Machine SCADA Expert 2020 project runtime runs on a variety of platforms and consists of the following modules or threads (program elements that can execute independently of other program elements):

**Background Task**

A supervisory task that executes other internal tasks (EcoStruxure Machine SCADA Expert worksheets). For example, the Background task executes scripts configured in the Math and Scheduler worksheets and manages parameters configured in the Alarm, Recipe, Report, and Trend worksheets.

**Watch**

A debugging tool that…

- Executes functions and/or expressions for testing purposes
- Reads data (such as tag values) from the tags database
- Writes data (such as tag values) to the Tagsdatabase

**LogWin**

A debugging tool that traces messages generated from other modules/tasks.

**Mobile Access Runtime**

Manages communication between your project runtime server and the web server that hosts the Mobile Access web interface.

**Driver Runtime**

Manages the read/write commands configured in the Driver worksheets.

**OPC Client**

Manages OPC communication with an OPC Server (local or remote), according to parameters configured in the OPC Client worksheets.

**OPC Server**

Manages OPC communication with an OPC Client (local or remote).

**TCP/IP Client**

A "thick client" that manages TCP/IP communication with the TCP/IP Server module in another EcoStruxure Machine SCADA Expert project, according to parameters configured in the TCP/IP Client worksheets.

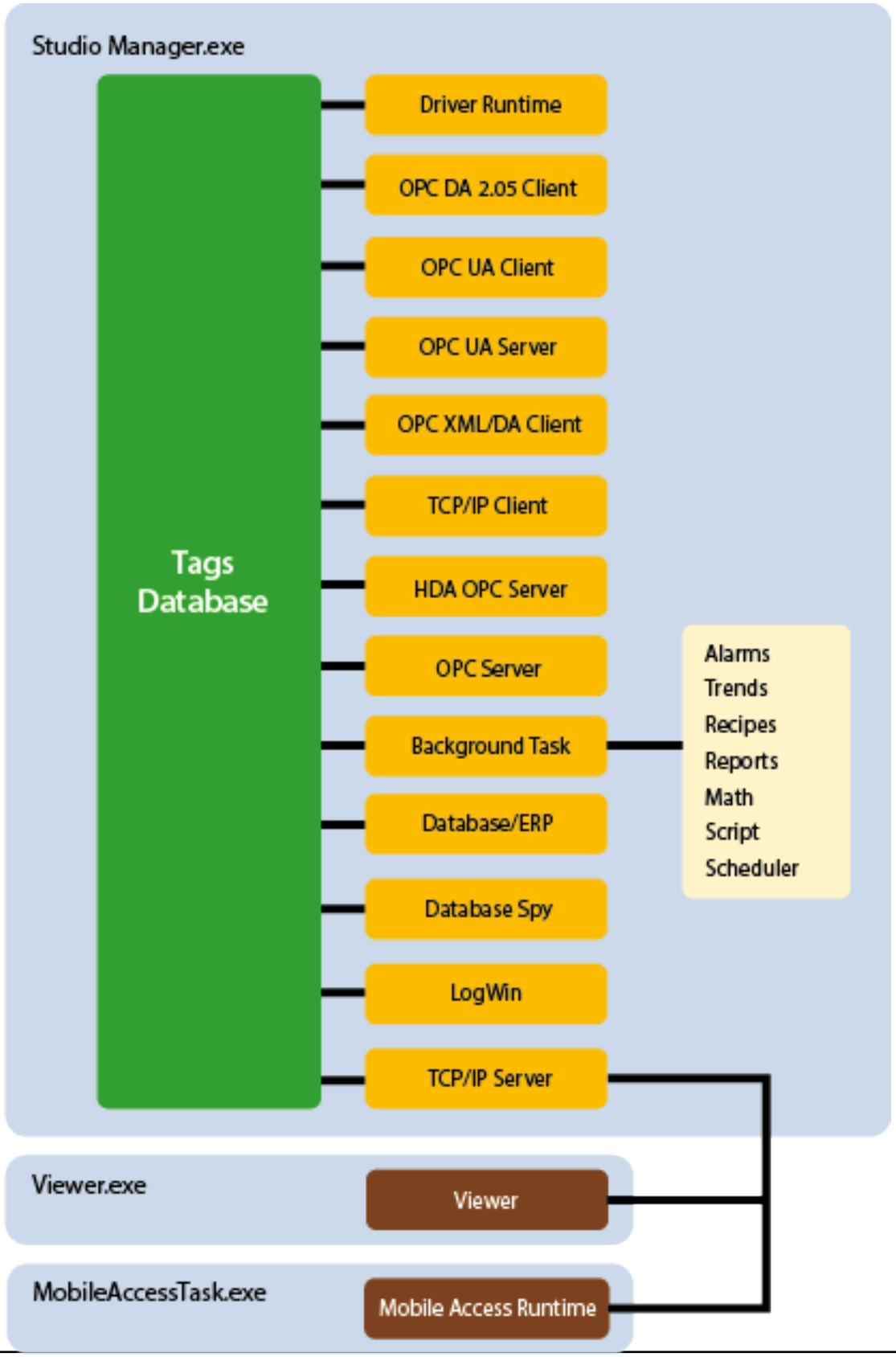
**TCP/IP Server**

Manages TCP/IP communication messages with both thick clients (i.e., the TCP/IP Client module in other EcoStruxure Machine SCADA Expert projects) and thin clients (i.e., the Viewer module).

**Viewer**

Executes all scripts (On Open, On While, On Close, Command, Hyperlink, and so forth) configured for project screens and updates the screen objects.

None of the preceding runtime modules exchange data directly with another module or task. Instead, runtime modules send data to and receive data from the tags database, which is the "heart" of EcoStruxure Machine SCADA Expert 2020.



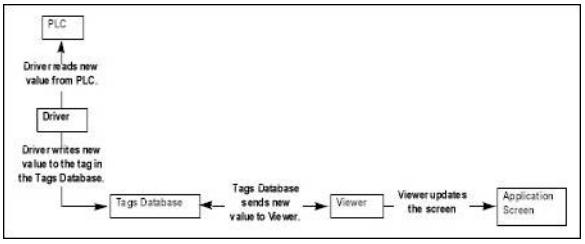

*Architecture of the project runtime on Windows*

The tags database manages the flow of data between modules. In addition, the tags database stores all tag values and the status of all properties associated with each tag (such as alarm conditioning, timestamp, quality, and so on).

Tags are variables (such as communication points in field equipment, calculation results, alarm points, and so on) that are used in screens and worksheets. For more information, see About Tags and the Project Database on page 144.

Each runtime module contains a virtual table of the tags that are relevant for that module at the current time. The tags database uses this table to determine which information must be updated in each module. For example, the Viewer contains a virtual table that lists all tags configured for all of the open project screens. If a tag value changes, the tags database sends a message to the Viewer, and then the Viewer updates the value in all objects where the tag is configured.

For example, if a driver reads a new value from the PLC, the driver updates the tag associated with this value in the tags database. Then, if this new information must display on the project screen, the tags database sends the new tag value to the Viewer task, and the Viewer updates the screen.




***An example of data flow***

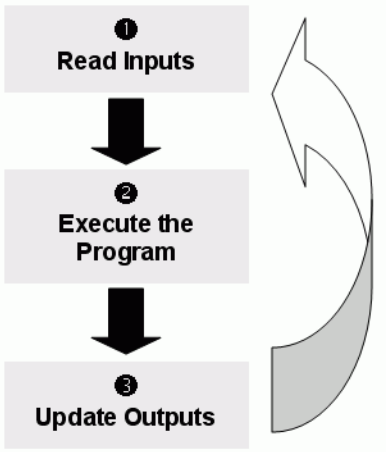
Note that the driver does not send new tag values directly to the Viewer. In addition, there is no pooling between tasks — the tags database receives the updated information and immediately forwards it to all runtime tasks requiring that information.

> **Note:** The Viewer module will update an object only when (at least) one of the object's tag values change.
>
> If you configure an object animation (e.g., Text Data Link) with a function that does not require a tag (e.g., `NoInputTime`), then the Viewer will not update the object because there is are no tags associated with that object.

The architecture of EcoStruxure Machine SCADA Expert 2020 significantly improves the internal data flow performance and makes it easy for you to add new internal tasks. Even though each task works independently, it can access information from any other task through the tags database.

## Executing and switching modules

EcoStruxure Machine SCADA Expert 2020 is a SCADA system composed of modules that must be executed simultaneously. Based on the multi-tasking concept, each runtime module (Viewer, Driver, and so forth) is a thread and the operating system switches from one thread to other automatically. It is a common misconception that you execute a SCADA system when you execute a PLC program. A PLC program contains a simple loop:




***PLC Program Loop***

However, in a SCADA system, there are several modules running simultaneously, and most of them can read and write data. Because a SCADA system modifies data (tag values) continuously during task execution, the preceding diagram is not applicable.

EcoStruxure Machine SCADA Expert 2020 only has one run-time process. On Windows and Windows Server, it is `Studio Manager.exe`. When you run a project, this process starts the tags database and all of the runtime modules configured for the project. You can specify which modules (such as Viewer and Driver) will start during run time.

Each process keeps a list of active threads for the operating system. Actually, each process activates and deactivates each thread during the runtime, according to the algorithm of each process. Also, when you create a thread you specify a priority value. The operating system continuously scans all currently active threads, and executes the threads according to their priority value — executing the higher-priority threads first. When threads with higher-priority values are active, the threads with lower-priority values are not executed at all. If there is more than one thread with the same priority value, and there are no other threads with higher-priority values, the operating system keeps switching between the threads with the same priority.

> **Note:** All EcoStruxure Machine SCADA Expert 2020 threads are set to priority 7, which is **`THREAD_PRIORITY_NORMAL`**. (Most programs contain this priority value.)
>
> Real-time program (such as SoftPLCs and Device Drivers) threads are assigned a higher-priority value (**`THREAD_PRIORITY_HIGHEST`**); however, these programs must provide a mechanism to keep them inactive for some period of time or the threads with normal priority would never be executed.
>
> EcoStruxure Machine SCADA Expert 2020 uses the `UNICOMM.DLL` library for serial drivers. This library creates a **`THREAD_PRIORITY_HIGHEST`** thread that "sleeps" (remains inactive) until data arrives in the serial channel. When EcoStruxure Machine SCADA Expert 2020 detects new data in the serial channel, the **`THREAD_PRIORITY_HIGHEST`** thread "wakes up" (becomes active) and transfers the data from the operating system buffer to the thread buffer, where it can be read by the Driver. This thread is the only highest-priority thread created by EcoStruxure Machine SCADA Expert 2020.

If you allowed threads to remain active all the time, the CPU usage would be 100% all the time, which must be avoided for performance reasons. Every program provides a mechanism to prevent threads from staying active all the time.

EcoStruxure Machine SCADA Expert 2020 uses the following parameters to prevent threads from staying active continuously:

- **TimeSlice** (from operating system): Causes the operating system to switch automatically between active threads with the same priority value.

  By default, the operating system executes each active thread for approximately 20ms and then switches to the next active thread. In other words, if there are multiple active threads with the same priority value waiting to be executed, the operating system will not execute any one active thread for more than 20ms.

- **TimeSlice** (from EcoStruxure Machine SCADA Expert 2020): Specifies how long each EcoStruxure Machine SCADA Expert thread can remain continuously active.

  You use this parameter in addition to the operating system's TimeSlice parameter. You configure a TimeSlice value for each EcoStruxure Machine SCADA Expert thread (except the Background Task) and specify how long each thread can remain continuously active. As long as a thread is active, the operating system can switch to that thread.

- **Period** (from EcoStruxure Machine SCADA Expert 2020): Specifies the maximum amount of time each EcoStruxure Machine SCADA Expert thread (except the Background Task) can remain inactive.

> **Note:** We strongly recommend that you do not change these default values unless it is absolutely necessary. Configuring these parameters incorrectly can cause the entire system to malfunction (for example, CPU usage will go to 100%) and/or cause some tasks to perform poorly.
>
> If you must change the parameter defaults, note the values before making your changes so if a malfunction occurs you can return to the original settings.

To change the EcoStruxure Machine SCADA Expert TimeSlice and Period parameter default values:

1. From the EcoStruxure Machine SCADA Expert 2020 installation directory (for example, **`C:\Program Files\`*`installation folder`*`\Bin`**), double-click `\BIN` to open the folder.
2. Double-click the **`Program Settings.INI`** file to open the file in Microsoft® Notepad.

   The following is a list of all parameters contained in this .ini file and their default values (in milliseconds).

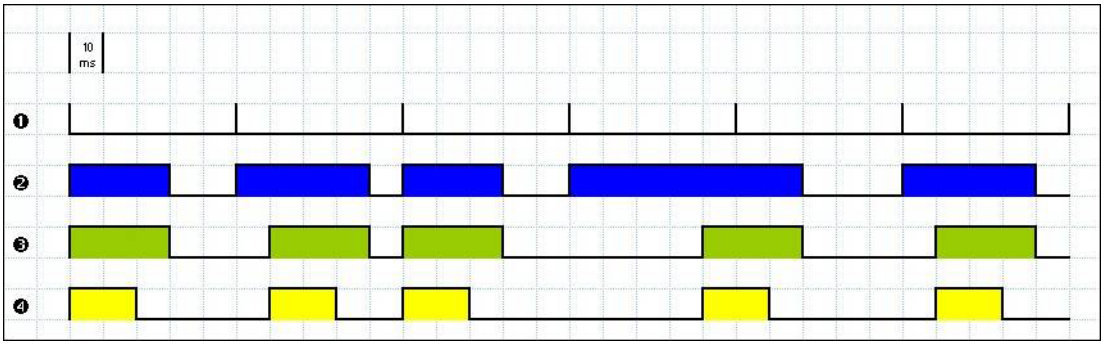
```
[Period]
DBSpy=1000
Driver=20
LogWin=100
OPCClient=20
OPCServer=20
TCPClient=100
TCPServer=100
Viewer=50

[TimeSlice]
Driver=10
OPCClient=10
OPCServer=10
TCPClient=200
TCPServer=200
Viewer=200
```

   > **Note:** You may not see all of these parameters listed when you open your Program Settings.INI file. However, even if a parameter is not visible in your list, EcoStruxure Machine SCADA Expert still uses that parameter and its default value.

   - To change the default value of a displayed parameter: In Notepad, delete the default value and type the new value in its place.
   - To change the default value of a parameter that is not displayed in your list: In Notepad, type the parameter name exactly as shown in the following list, the equal sign, and then the new value.

3. Save the file (**File** > **Save**) and close Notepad (**File** > **Exit**).

The following figure illustrates how EcoStruxure Machine SCADA Expert executes a generic thread (such as the Viewer).




***Executing a Generic Thread***

Where:

- Signal **1** is the Period time period (set to 50ms for this example).
- Signal **2** shows when the thread is active for the operating system.
- Signal **3** is the TimeSlice time period (set to 30ms for this example).
- Signal **4** shows the execution of the thread itself.

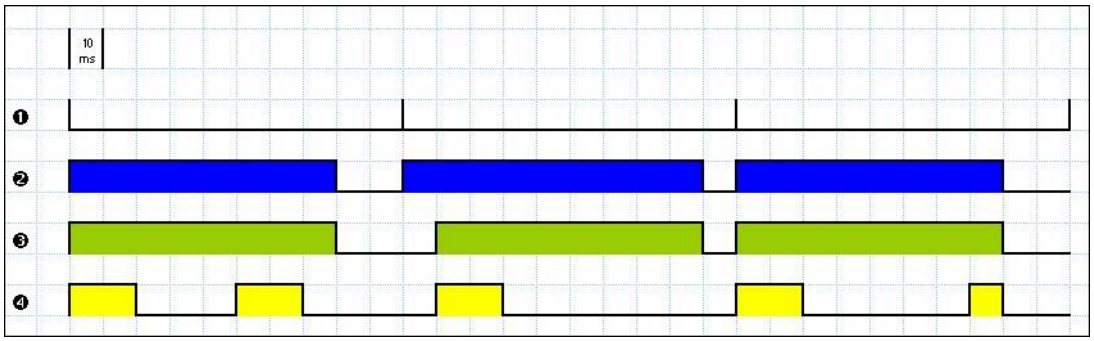
In this example, EcoStruxure Machine SCADA Expert generates a Period message every 50ms (signal **1**). When EcoStruxure Machine SCADA Expert generates this message, its thread becomes active and remains active until the specified TimeSlice time period (from EcoStruxure Machine SCADA Expert) expires. The thread then remains inactive until EcoStruxure Machine SCADA Expert generates the next Period message (signal **1**).

While the thread is active, the operating system is responsible for executing that thread. However, just because a thread is active does not mean the operating system will execute it immediately — the operating system may be executing other threads, for example.

When the operating system executes the thread, the TimeSlice timer starts counting and the thread is executed for 20ms (TimeSlice from the operating system). After the 20ms period, the operating system automatically switches to the next active thread (such as the Driver), and so on.

In the above example, the TimeSlice time was set to 30ms, which means the operating system is not supposed to execute the thread more than once in each TimeSlice of EcoStruxure Machine SCADA Expert. However, if you specify higher values for the EcoStruxure Machine SCADA Expert TimeSlice time period, it is likely that the operating system will execute the same thread more than once in the same TimeSlice time period.

In the next example, the Period and the TimeSlice values were changed as follows, but the default operating system TimeSlice period (20 ms) was not changed.




***Setting a Higher TimeSlice***

Where:

- Signal **1** is the Period time period (set to 100ms).
- Signal **2** shows when the thread is active for the operating system.
- Signal **3** is the EcoStruxure Machine SCADA Expert TimeSlice time period (set to 80ms).
- Signal **4** shows the execution of the thread itself.

Notice that the thread can be executed more than once in the same TimeSlice time period. When the EcoStruxure Machine SCADA Expert TimeSlice time period expires, the operating system interrupts the thread execution; however, even though the EcoStruxure Machine SCADA Expert Period and TimeSlice parameters are set to 100ms and 80ms respectively, the operating system will not execute this thread continuously for more than 20ms, because the operating system TimeSlice time period is set to 20ms.

When the operating system is not executing the Viewer thread, the CPU can execute any other thread or remain idle (if there are no other active threads to execute). Remember, the EcoStruxure Machine SCADA Expert Period and TimeSlice parameters were created to prevent all threads from being active at the same time to prevent 100% CPU usage.

During thread execution, the thread must handle its pending messages. For example, the Viewer module must update any related screen objects. If there are no messages pending, the thread deactivates itself and gives control back to the operating system. The operating system immediately switches to the next active thread. In other words, a thread can interrupt its own execution — even if the operating system TimeSlice time period has not yet expired (which occurs frequently in real-world applications).

> **Note:** The Watch and LogWin tasks do not have a TimeSlice parameter. Consequently, after each thread handles all of its pending messages, the threads become inactive until the next Period message for each one of the threads occurs.

The Background Task is the exception to the execution/switching process just discussed. The mechanism for executing/switching the Background Task is described in the next section.

## Executing and switching the Background Task

The Background Task executes scripts from the Math and Scheduler worksheets (for example, messages from Alarm and Trend worksheets). In addition, the Background Task executes all Recipe and Report commands when the `Recipe` or `Report` functions are executed during the runtime.

Although the Alarm, Math, Scheduler, and Trend tasks are not threads, you can specify or change their Period time in the `Program Settings.ini` file located in the EcoStruxure Machine SCADA Expert program directory.

The Period default values (in milliseconds) are as follows:

```
[Period]
Math=100
Sched=50
Alarm=100
Trend=1000
```

These values mean that every 100ms, EcoStruxure Machine SCADA Expert generates a Period message to the Math task. Every 50ms, EcoStruxure Machine SCADA Expert generates a Period message to the Scheduler task, and so on.

> **Note:** We strongly recommend that you do not modify the Background Task default values unless it is absolutely necessary. Configuring any of these parameters incorrectly can cause your entire system to malfunction (for example, CPU usage will go to 100%) and/or cause some tasks to perform poorly.
>
> If you must change the parameter defaults, note the values before making your changes so if a malfunction occurs you can return to the original settings.

Keep in mind that the Background Task thread has the same priority as other threads in EcoStruxure Machine SCADA Expert (Drivers, Viewer, and so forth), which means that the operating system will not execute this task continuously for more than 20ms.

The Background Task executes the Recipe and Report worksheets when the `Recipe` and `Report` functions are called, respectively. Because the `Recipe` and `Report` functions are synchronous, once the Background Task starts executing the functions, it will not switch to another task (Alarm, Math, Scheduler, or Trend) until it completely executes the functions. Executing a `Recipe` or `Report` function usually takes a few milliseconds.

The Background Task must switch between the Alarm, Math, Scheduler, and Trend tasks. When Background Task switches to the Scheduler task, it will not switch to another task (Alarm, Math, or Trend) until all Scheduler worksheets are executed. After executing all Scheduler worksheets, the Background Task will not execute the Scheduler again until it receives the next Period message for the Scheduler task.

The Background Task applies the same behavior when executing the Alarm and Trend tasks — when the Background Task switches to the Alarm or Trend task, it will not switch to another task until it handles all pending messages. So, the Background Task will not execute the Alarm or Trend task again, until EcoStruxure Machine SCADA Expert generates the next Period message for each of these tasks.

The Background Task typically executes the Alarm, Scheduler, and Trend tasks in a few milliseconds. However, it can take longer to execute the Math task because it usually contains loops and complex scripts. Consequently, the mechanism used to execute the Alarm, Scheduler, and Trend tasks cannot be applied to the Math task.

The Background Task executes the Math task for no more than 10ms continuously before switching to other task (such as the Scheduler). The Background Task cannot execute the Math task again for the next 50ms; however, the Background Task can execute other tasks (Alarm, Recipe, Report, Scheduler, or Trend) during this 50ms period. After the Background Task executes all of the Math worksheets, it will not begin a new scan of the Math worksheets until EcoStruxure Machine SCADA Expert generates a new Period message for the Math task.

It is important to re-emphasize that this process was created to prevent 100% CPU usage all the time.

> **Note:** We recommend caution when using the `Math` function in a Scheduler worksheet or for a screen object (such as the Command animation).

When the Scheduler task executes a `Math` function, no other task can be executed by the Background Task until the Scheduler executes the entire Math worksheet called by the `Math` function. This process can take several milliseconds or even seconds, depending on how you configured the script in the Math worksheet (especially for loops).

If you configure a `Math` function for a screen object, the Viewer stops updating the screen until the Viewer executes the entire Math worksheet called by the `Math` function.

If you must use the `Math` function for the Scheduler task or a screen object, we recommend using the following procedure to prevent process delays:

1. Specify one auxiliary tag with the value 1 and the Scheduler or Viewer task will send a message to the Tags database to update this tag value.
2. Configure the tag in the Execution field of the Math worksheet to be executed. When the Background Task scans the Math worksheet, EcoStruxure Machine SCADA Expert will execute the worksheet.
3. Reset the tag in the last line of the Math worksheet (write the value 0 to the auxiliary tag).

As a result, the Background Task will not execute the Math worksheet in the next scan unless the auxiliary tag is set to the value 1 again.

# Installation Guide

This is a brief guide to installing the full EcoStruxure Machine SCADA Expert 2020 software.

## About the EcoStruxure Machine SCADA Expert 2020 software components

The EcoStruxure Machine SCADA Expert 2020 software suite comprises several individual components that can be installed on different platforms to perform different functions. The architecture of your finished project depends on which components you install, where you install them, and how you connect them to each other.

The following table lists all of the available components.

| Component | Features | Platforms |
|---|---|---|
| EcoStruxure Machine SCADA Expert 2020 | • Project development environment<br>• Tag integration<br>• Remote management of project runtimes<br>• Project runtime<br>• Agent to allow remote management<br>• Project viewer | • Windows<br>• Windows Server |
| Studio Database Gateway (StADOSvr) | Enables communication between the project runtime and external databases, including Historian and most ADO.NET-compatible databases. | • Windows<br>• Windows Server |
| Mobile Access Runtime | Enables the project runtime to serve HTML5-enhanced project screens to web browsers and mobile devices. | • Internet Information Services (IIS) for Windows<br>• any CGI-enabled web server (e.g., Apache) |
| Secure Viewer | Project viewer, as a standalone program. (See note below.) | • Windows<br>• Windows Server |

It is important to distinguish between the project development environment and the project runtime. You can use the project development environment to design, develop, troubleshoot, deploy, and monitor projects. In contrast, the project runtime actually runs your project, communicates with external databases and devices, and serves screens to project viewers.

The full EcoStruxure Machine SCADA Expert 2020 software for Windows includes both the project development environment and the project runtime. Your software license determines which parts of the software you can use on any given computer or device. For more information, see Execution Modes on page 46.

In most cases, the first thing you should do is install the full EcoStruxure Machine SCADA Expert 2020 software on your primary workstation, because it not only sets up the project development environment for you, it also unpacks the rest of the components so that they can be installed on other computers and devices.

**Note:** We recommend that you use Mobile Access instead of our traditional Thin Client software whenever possible. Thin Client depends on legacy, Windows-only technologies, while Mobile Access allows you to use any HTML5-compatible browser running on any platform as a project viewer. Mobile Access does not yet support all of the features that Thin Client does, but we are continuing to improve Mobile Access with every new release.

## Install the full EcoStruxure Machine SCADA Expert 2020 software

Install the full EcoStruxure Machine SCADA Expert 2020 software on your Windows computer in order to develop projects, or to use the computer as a project runtime server and/or thin client.

To install and run the full EcoStruxure Machine SCADA Expert 2020 software, you must have the following:

- A Windows-compatible computer with a standard keyboard, a pointer input (i.e., a mouse, trackpad, or touchscreen), and an SVGA-minimum display;
- One of the following Windows operating systems:
  - Windows:
    - Windows 10, version 1803 or later (including LTSC/LTSB versions)
    - Windows 8.1
  - Windows Server:
    - Windows Server 2019
    - Windows Server 2016
    - Windows Server 2012 R2
- .NET Framework 3.5 and .NET Framework 4.7.2 (see note below);
- Internet Explorer 11 (not Microsoft Edge);
- Minimum 2 GB available storage (hard drive or non-volatile);
- Minimum 1 GB available memory (RAM); and
- An Ethernet or Wi-Fi network adapter.

We recommend the "Pro" and "Enterprise" editions of Windows, because they include Internet Information Services (IIS) as a pre-installed feature that can be turned on. You can use IIS to make your projects accessible to thin clients and mobile devices. We do not recommend the "Home" and "Education" editions of Windows, because they hide or disable many important features.

Only Windows 10, Windows Server 2016, and Windows Server 2019 are under what Microsoft calls "mainstream support", which means they are actively maintained and additional service packs might be released for them in the future. Windows 8.1 and Windows Server 2012 R2 are under what Microsoft calls "extended support", which means they are no longer actively maintained. For more information, go to: https://support.microsoft.com/en-us/help/13853/windows-lifecycle-fact-sheet

Regardless of which version or edition of Windows you are using, you should make sure it is fully updated before you install EcoStruxure Machine SCADA Expert 2020. Updating Windows ensures that it has all of the latest security fixes and system components.

The operating system, storage, and memory requirements will necessarily increase for larger projects; the minimum requirements listed above are only for projects of up to 4,000 tags. The following table shows the complete requirements:

| Project Size | Operating System | Storage | Memory |
|---|---|---|---|
| up to 4,000 tags | Windows, Windows Server | 2 GB available | 1 GB available |
| up to 64,000 tags | Windows, Windows Server | 4 GB available | 2 GB available |

Your computer needs to meet only the minimum requirements when you first install the software and begin to develop your project, but the requirements will increase as your project grows. Furthermore, every computer or device that you plan to use as a runtime station must meet the same requirements.

The following items are optional but recommended:

- A USB port or memory card slot, to be used for hardkey licensing of the software.

  This is optional because softkey licensing is also available.
- Serial COM ports and adapters, to be used for direct communication with PLCs and other devices.

This is optional because many newer device protocols use Ethernet communication (i.e., TCP/IP or UDP/IP) instead of serial communication.

- Internet Information Services (IIS) installed and turned on; for more information, see the description of the **Mobile Access Runtime** option below.

  This is optional because you may choose not to install the Mobile Access Runtime feature at this time, as part of the full EcoStruxure Machine SCADA Expert 2020 software. You can install it at a later time, for either IIS or CGI.

Finally, you must have Administrator privileges on the computer in order to install any software.

> **Note:**
>
> You must have both .NET Framework 3.5 and .NET Framework 4.7.2 (or later) installed and turned on in order to use all of the features in EcoStruxure Machine SCADA Expert 2020.
>
> If Windows is fully updated on your computer, the latest versions of .NET Framework should be installed already. They might not be turned on, however — in fact, in recent versions of Windows and Windows Server, .NET Framework 3.5 is turned off by default. Use the *Windows Features* control panel to make sure they are turned on.
>
> In some cases, it might not be possible to keep Windows fully updated through normal means. For example, if your computer is on a private network, it might not be able to contact the Windows Update service. The EcoStruxure Machine SCADA Expert 2020 software installer will check for both versions of .NET Framework, and if it does not find them, it will attempt to install them for you. Depending on your computer's security settings, however, the installation(s) might fail without notice. Therefore, if you experience issues later while using this software — in particular, while trying to communicate with external databases — use the **Windows Features** control panel to confirm that both versions of .NET Framework are installed and turned on.
>
> On Windows Server 2012 R2, .NET Framework 3.5 is not pre-installed and it cannot be installed by the EcoStruxure Machine SCADA Expert 2020 software installer. You must use the Server Manager tool to install it. For more information about how to do this, contact your software distributor and request the document *Install .NET Framework 3.5 on Windows Server 2012*.
>
> For more information about .NET Framework in general, go to: https://docs.microsoft.com/en-us/dotnet/framework/index

To install the full EcoStruxure Machine SCADA Expert 2020 software:

1. Close all other running programs, if possible.

   We recommend you do this because those programs can use a significant amount of system resources and therefore cause this installation to take longer to finish. Windows services (e.g., Windows Defender, Windows Update) can have the same effect, but we do not recommend you stop or disable those services.

2. Do one of the following:

   - Download the zipped installer to your computer, either from our website (www.se.com) or from another location on your network where you have previously saved it. Extract the files, open the resulting folder, and then locate and run the setup program (`setup.exe`).

   The installation wizard runs and asks you to select a language for the installation.

3. Select a language from the list, and then click **OK**.

   This selection determines the language of the user interface for both the installation wizard and the project development environment. You can change the language for the project development environment later, after the software has been installed.

   The wizard prepares for installation. During this step, it automatically installs SafeNet's Sentinel drivers (a part of the software licensing mechanism), .NET Framework 3.5, and .NET Framework 4.7.2, if necessary.

4. On the **Welcome** page of the wizard, click **Next** to proceed with the installation.

5. On the **License Agreement** page, click **Yes** to accept the agreement and proceed, or click **No** to refuse the agreement and exit the wizard.

6. On the **Customer Information** page, type your user name and company name, and then click **Next**.

7. On the **Choose Destination Location** page, select the folder where the software should be installed, and then click **Next**.

   By default, the software will be installed at the following location:

   ```
   C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\
   ```

8. On the **Select Features** page, select the specific features and components that you want to install, and then click **Next**.

| Feature | Description |
|---|---|
| **Program Files** | The main program files for the project development environment, the project runtime software for Windows and Windows Server, and the thin client software for viewing project screens.<br>You cannot deselect this feature. |
| **BDE for PanelMate™ Import Wizard** | Borland Database Engine (BDE), which is required to import a PanelMate Plus or PanelMate Power Pro program into a new project.<br>This feature is not selected by default, because it is used only in some cases. |
| **Custom Widget Framework** | Additional software that is required to develop HTML5-based widgets and then use them in project screens.<br>You cannot deselect this feature. |
| **Demo Projects** | Premade projects that demonstrate the capabilities of the EcoStruxure Machine SCADA Expert 2020 software. |
| **Hardkey Support** | Additional drivers that support the use of hardkey licenses. |
| **Industrial Graphics** | The Industrial Graphics editor and symbol library, which works as a companion to our native graphics tools. |
| **Mobile Access Runtime** | Add-on software for Internet Information Services (IIS) that lets you use any HTML5-compatible browser to view your project screens. If you select this feature, the installer will try to confirm that IIS is turned on in Windows, and if it is, the add-on software will be installed.<br>Regardless of whether you select this feature for installation, a separate Mobile Access Runtime software installer (`MobileAccessSetup.exe`) will be copied to your EcoStruxure Machine SCADA Expert 2020 program folder. You can run that installer at a later time. |
| **OPC Components** | Additional components that are required for communication with other OPC-compatible devices. This includes OPC DA (a.k.a. OPC Classic), OPC XML-DA, and OPC UA. |
| **PDF Printing** | Additional software that lets projects save run-time reports as PDF files. |
| **Security System Device Driver** | An additional keyboard driver that helps to enforce security during project run time. |
| **Symbol Library** | A large library of premade but customizable screen objects such as pushbuttons, toggle switches, gauges, dials, indicator lights, and so on. |
| **Historian** | Additional software that is required to save historical data (e.g., from Trend worksheets) to AVEVA Historian or AVEVA Insight.<br>If you want to use this feature in your project, you must have .NET Framework 4.7.2 (or later) installed and turned on. |
| **OPC Factory Server** | A standalone OPC server that your projects can use to communicate with the following Schneider Electric PLCs:<br>• M340<br>• M580<br>• Quantum<br>• TSX Compact<br>• TSX Micro<br>• TSX Momentum<br>• TSX Series 7 |

| Feature | Description |
|---|---|
| | • TSX S1000<br>• TSX/PCX Premium<br>• Unity Momentum<br>For more information, see the documentation that is installed with OPC Factory Server. |

9. On the **Ready To Install** page, click **Install**.

> **Note:** You might receive the following error message during installation: "Error 1628: Failed to complete script based install." For more information about this error and how to resolve it, go to: https://flexeracommunity.force.com/customer/articles/en_US/ERRDOC/Error-1628-Failed-To-Complete-Script-Based-Install

> **Note:** If you try to install an earlier version of this software on a computer that already has a later version installed, you might receive the following message during installation: "Version x.x.x.x of CodeMeter Development Kit is already installed. Downgrading to Version x.x.x.x is not possible, installation will be aborted." CodeMeter is supplemental software used by EcoStruxure Machine SCADA Expert 2020 to manage hardkey licenses. To resolve this issue, you must use Task Manager in Windows to stop CodeMeter Runtime Server (`CodeMeter.exe`) before you install the earlier version of the software.

The software is installed, and then when the installation is finished, the last page of the wizard is displayed.

10. Click **Finish** to close the installation wizard.

When you have finished the installation, you can find the software in your Windows Start menu at: **Start** > **EcoStruxure Machine SCADA Expert 2020**

The software includes the following "apps" (applications):

**EcoStruxure Machine SCADA Expert 2020 Studio**

The project development environment and/or the project runtime for Windows. Its capabilities are determined by your software license.

**EcoStruxure Machine SCADA Expert 2020 Help Manual**

A complete technical reference and user guide for all of the EcoStruxure Machine SCADA Expert 2020 software.

**EcoStruxure Machine SCADA Expert 2020 Quick Start Guide**

A brief guide to installing and using the project development environment, including a tutorial for developing a simple project.

**EcoStruxure Machine SCADA Expert 2020 Register**

A utility program that manages your EcoStruxure Machine SCADA Expert 2020 software license.

**EcoStruxure Machine SCADA Expert 2020 Release Notes**

A list of changes in the EcoStruxure Machine SCADA Expert 2020 software.

**EcoStruxure Machine SCADA Expert 2020 Remote Agent**

A utility program that allows other stations to remotely manage EcoStruxure Machine SCADA Expert 2020 as a project runtime.

**EcoStruxure Machine SCADA Expert 2020 StartUp**

A shortcut that automatically starts the project runtime and then runs the most recent project.

There should also be a shortcut icon on your desktop.

To run the software, do one of the following:

- Double-click the shortcut icon on your desktop; or
- Click **Start** > **EcoStruxure Machine SCADA Expert 2020** > **EcoStruxure Machine SCADA Expert 2020 Studio**.

If the installation failed for any reason, you can use System Restore to restore the computer to the restore point that was created at the beginning of the installation. For more information about System Restore, go to: https://support.microsoft.com/help/17127/windows-back-up-restore

### *Install .NET Framework 3.5 in Windows Server 2012 R2*

Use the Server Manager utility to install .NET Framework 3.5 in Windows Server 2012 R2.

Before you begin this task, if the computer on which you are installing EcoStruxure Machine SCADA Expert 2020 is not connected to the Internet, make sure you have a Windows Server 2012 R2 installation disc on hand. The files that you will need to install .NET Framework 3.5 are located on the disc.

Also, you must have Administrator privileges on the computer in order to install software.

.NET Framework 3.5 is part of the system requirements for the full EcoStruxure Machine SCADA Expert 2020 software; it must be installed and turned on before you can install EcoStruxure Machine SCADA Expert 2020. In most recent versions of Windows, .NET Framework 3.5 is pre-installed but not turned on; you can turn it on and then proceed with installing EcoStruxure Machine SCADA Expert 2020. In older versions of Windows, .NET Framework 3.5 is not pre-installed but can be installed automatically by the EcoStruxure Machine SCADA Expert 2020 software installer. For more information, see Install the full EcoStruxure Machine SCADA Expert 2020 software on page 35.

Windows Server 2012 R2 is a unique exception. In this version of Windows, .NET Framework 3.5 is not pre-installed and it cannot be installed automatically. Therefore, you must manually install it.

> **Note:** This task includes some additional steps to turn on other features that are not required to install EcoStruxure Machine SCADA Expert 2020, but that EcoStruxure Machine SCADA Expert 2020 might need later if you use it as a project runtime server (as opposed to using it only for project development). We recommend that you turn on all of these features now, at the same time, so that you do not need to repeat these steps later. For more information, see Turn on IIS for thin client access on page 768.

To install .NET Framework 3.5 in Windows Server 2012 R2:

1. In Windows Server, open **Server Manager**.
   On some Windows Server computers, Server Manager is configured to open automatically when a user with Administrator privileges logs on.
2. In the Server Manager dashboard, click **Add roles and features.**
   The first page of the *Add Roles and Features* wizard is displayed.
3. On the **Before You Begin** page, read the instructions, and then click **Next**.
   The next page of the wizard is displayed.
4. On the **Installation Type** page, click **Role-based or feature-based installation**, and then click **Next**.
   The next page of the wizard is displayed.
5. On the **Server Selection** page, click **Select a server from the server pool**, select your server in the **Server Pool** list (it is often the only choice), and then click **Next**.
   The next page of the wizard is displayed.
6. On the **Server Roles** page, in the **Roles** list, select the **Web Server (IIS)** check box if it is not already selected.
   A dialog box is displayed asking you to confirm that you want to add the features required for Web Server (IIS).
7. In the dialog box, click **Add Features**.
   The dialog box is closed, and the **Server Roles** page is displayed again with the **Web Server (IIS)** check box selected.
8. On the **Server Roles** page, click **Next**.
   The next page of the wizard is displayed.
9. On the **Features** page, in the **Features** list, select the **.NET Framework 3.5 Features** check box, and then click **Next**.
   The next page of the wizard is displayed.

10. On the **Web Server Role (IIS)** page, read the instructions, and then click **Next**.
    The next page of the wizard is displayed.
11. On the **Role Services** page, in the **Role services** list, expand **Web Server** > **Application Development**.
12. Under **Application Development**, select the **ASP** check box.
    A dialog box is displayed asking you to confirm that you want to add the features required for ASP.
13. In the dialog box, click **Add Features**.
    The dialog box is closed, and the **Roles Services** page is displayed again with the **ASP** and **ISAPI Extensions** check boxes selected.
14. Under **Application Development**, select the **ASP.NET 3.5** check box.
    A dialog box is displayed asking you to confirm that you want to add the features required for ASP.NET 3.5.
15. In the dialog box, click **Add Features**.
    The dialog box is closed, and the **Roles Services** page is displayed again with the **.NET Extensibility 3.5**, **ASP.NET 3.5**, and **ISAPI Filters** check boxes selected (in addition to the previously selected features).
16. Under **Application Development**, select the **ASP.NET 4.5** check box.
    A dialog box is displayed asking you to confirm that you want to add the features required for ASP.NET 4.5.
17. In the dialog box, click **Add Features**.
    The dialog box is closed, and the **Roles Services** page is displayed again with the **.NET Extensibility 4.5** and **ASP.NET 4.5** check boxes selected (in addition to the previously selected features).
18. On the **Roles Services** page, click **Next**.
    The next page of the wizard is displayed.
19. On the **Confirmation** page, review the list of selected features.
20. If the computer has an Internet connection, proceed to the next step. Otherwise, if the computer does not have an Internet connection but you do have a Windows Server 2012 R2 installation disc, do the following:
    a) Insert the installation disc in the computer's DVD-ROM drive.
    b) On the **Confirmation** page, click **Specify an alternate source path**.
    The *Specify Alternate Source Path* dialog box is displayed.
    c) In the *Specify Alternate Source Path* dialog box, in the **Path** box, type the file path for the installation files.
    Assuming the computer's DVD-ROM drive is the D drive, the file path should be the following: `D:\sources\sxs`
    d) Click **OK**.
    The dialog box is closed, and the **Confirmation** page is displayed again.
21. On the **Confirmation** page, click **Install**.
    The selected features, including .NET Framework 3.5, are installed and/or turned on. When the installation is finished, the next page of the wizard is displayed.
22. On the **Results** page, review the results of the installation, and then click **Close**.

You should now be able to proceed with the installation of EcoStruxure Machine SCADA Expert 2020.

## Install the Thin Client software

Install the Thin Client software on a client station in order to let users view your project.

> **Note:** We recommend that you use Mobile Access instead of our traditional Thin Client software whenever possible. Thin Client depends on legacy, Windows-only technologies, while Mobile Access allows you to use any HTML5-compatible browser running on any platform as a project viewer. Mobile Access does not yet support all of the features that Thin Client does, but we are continuing to improve Mobile Access with every new release.

If you have already installed either the full EcoStruxure Machine SCADA Expert 2020 software or one of the runtime editions on the computer or device that you want to use as a client station, you may skip this procedure because you do not need to install the Thin Client software on the same computer or device. The full software and the runtime editions include the same components as the Thin Client software, except that they are preconfigured to view the project that is running locally.

Before you begin this procedure, you should install the full EcoStruxure Machine SCADA Expert 2020 software on at least one Windows computer — typically, on your project development workstation — because doing so also unpacks the Thin Client software installer.

To run the Thin Client software installer, you must have a computer or device with a network connection and one of the following operating systems:

- Windows:
  - Windows 10, version 1803 or later (including LTSC/LTSB versions)
  - Windows 8.1
- Windows Server:
  - Windows Server 2019
  - Windows Server 2016
  - Windows Server 2012 R2

You must also have Administrator privileges on the computer or device in order to install any software.

The Thin Client software is based on ISSymbol, which is an ActiveX control that we developed to open project screens and exchange data (e.g., tag values) with the project runtime. It acts as a control layer between the client and the server, similar to the Java Virtual Machine for Java-based applications, and it provides a high level of security because it does not allow the project to access the operating system on the client station.

To install the Thin Client software:

1. Locate the Thin Client software installer in your EcoStruxure Machine SCADA Expert 2020 program folder.

   If EcoStruxure Machine SCADA Expert 2020 was installed at its default location on your computer, the Thin Client software installer should be located at:

   ```
   C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
    2020\Redist\WebAddOn\ThinClient\ThinClientSetup.exe
   ```

2. Copy the Thin Client software installer to the computer or device on which you want to install the software. Assuming the computer or device has a network connection — which it should, if you plan to use it as a project viewer — you can simply copy the installer across the network. Otherwise, copy the installer to removable media (e.g., a USB flash drive) and then carry it to the computer or device.
3. On that computer or device, run the Thin Client software installer (`ThinClientSetup.exe`).
   The first page of the installation wizard is displayed.
4. Click **Next**.
   The next page of the wizard is displayed.
5. On the **Customer Information** page, type your name and your company name, and then click **Next**.
   The next page of the wizard is displayed.

6. On the **Choose Destination Location** page, select the folder where the software should be installed, and then click **Next**.

   By default, the software will be installed at:

   **C:\Program Files (x86)\Schneider Electric\Thin Client*<ID string>***

   The next page of the wizard is displayed.

7. On the **Select Features** page, select the specific features and components that you want to install, and then click **Next**.

| Feature | Description |
|---|---|
| **Program Files** | The main program files for the thin clients. This feature cannot be deselected. |
| **Secure Viewer** | Creates shortcuts in the Start menu and on the desktop. If you deselect this feature, the program files will still be installed but the shortcuts will not be created. You will need to locate the Secure Viewer program (`Viewer.exe`) and then manually run it. |
| **PDF Printing** | Additional software that allows the project to save run-time reports as PDF files. |
| **Security System Device Driver** | An additional keyboard driver that enforces project security during run time by controlling user input. |

   The next page of the wizard is displayed.

8. On the **Ready to Install the Program** page, click **Install**.
   The software is installed, and then when the installation is finished, the last page of the wizard is displayed.

9. Click **Finish** to close the installation wizard.

Once the Thin Client software is installed, you may choose which type of thin client to use:

- If you choose to use Secure Viewer as a standalone program, you must configure it before you can run it. For more information, see Configure and run Secure Viewer on page 742.

The Thin Client software itself does not need to be licensed on any computer or device. The license for the project runtime determines the number of thin clients that are allowed to connect to it at the same time. For more information, see License Settings on page 45.

### *Install the Custom Widget Framework on a client station*

If your project screens include custom widgets, you might need to install Custom Widget Framework on some client stations to enable them to properly display the widgets.

This task applies only to stations on which you have already installed either Thin Client or Embedded HMI — in other words, stations that are using the Thin Client software to view your project.

Stations that are viewing your project through Mobile Access do not need to have Custom Widget Framework installed, because custom widgets are HTML5-based screen objects that can be displayed normally in the web browser.

Before you begin this task, you must have installed the full SCADA software on at least one Windows computer — typically, on your project development workstation — because doing so also unpacks the Custom Widget Framework installer. (Custom Widget Framework is not included in the Thin Client installer because it would greatly increase the file size of that installer, for a feature that not all projects use.)

You must have Administrator privileges on a computer or device in order to install any software.

To install the Custom Widget Framework on a client station:

1. Locate the Custom Widget Framework installer (`CustomWidgetFrameworkSetup.exe`) in your SCADA program folder.

If SCADA was installed in its default location, the Custom Widget Framework installer should be located at:

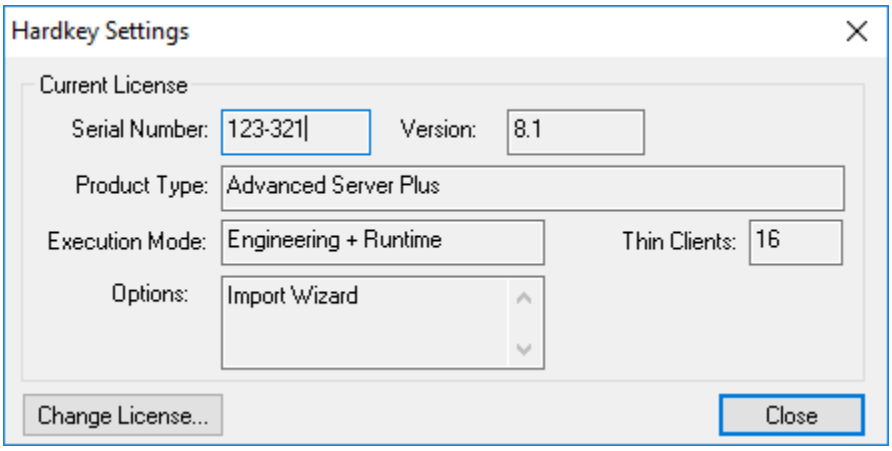
```
C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
 2020\Redist\CustomWidgetFramework\CustomWidgetFrameworkSetup.exe
```

2. Copy the installer to the client station, either over the network or on a portable hard drive.
3. Run the installer. You might need to do this as a user with Administrator privileges: right-click the installer, and then on the shortcut menu, click **Run as Administrator**.
4. Follow the installer's instructions. On the *Choose Destination Location* page of the installer, make sure the Bin folder of either Thin Client or Embedded HMI is selected. If it is not, click **Browse** and then use the file browser to locate and select the folder.

# Licensing

## License Settings

Every EcoStruxure Machine SCADA Expert 2020 software license has the following settings:




***Example of license settings***

**Serial Number**

The unique serial number of the hardkey, if you are using hardkey licensing. For more information, see About hardkey licenses on page 48.

**Version**

The version of the software for which the license is valid. Please note this is the major version only; updates, patches, and hot fixes are minor versions included in the major version. For more information, see Product Versions on page 46.

**Product Type**

The maximum product type that can be run by the project runtime, if the execution mode includes Runtime. The product type determines the maximum number of project tags that can be used in a single project. For more information, see About target platforms, product types, and target systems on page 100.

**Execution Mode**

Specifies one of the following options:

**Engineering Only**

You can develop a project and then run it for a limited period, for testing purposes only.

**Runtime Only**

You can run a project for an unlimited period, but you cannot develop or modify the project.

For more information, see Execution Modes on page 46.

**Thin Clients**

The maximum number of thin clients that can connect simultaneously to the project runtime. For more information, see Thin Clients and Mobile Access on page 727. One connection is included with every license; contact your software distributor to purchase additional connections.

**Options**

Additional options and features, including a list of the third-party programs that can be imported by the Import Wizard.

## Product Versions

All editions of the project development and runtime software should have the same version number, which uses the **X.Y+SPWW** syntax (e.g., EcoStruxure Machine SCADA Expert 2020 v20.0), where:

- **X** represents the **Family version**. The Family version changes only when major enhancements are added to the product technologies and concepts.
- **Y** represents the **Sub-version**: The Sub-version changes when minor enhancements and/or new features are added to the product.
- **WW** represents the **Service Pack**. The Service Pack version changes when you must install add-on packages to accomplish the following:
  - Upgrade files for the version previously installed
  - Fix bugs in the product (showstoppers and no-workarounds)
  - Provide minor enhancements before releasing the next version of the product

  Each Service Pack release supersedes the previous Service Pack release. For example, SP2 includes all the contents of SP1 and all newly upgraded files, bug fixes, and enhancements. SP3 includes all the contents of SP2 and all new upgraded files, bug fixes, enhancements and so on.

Later versions of this software can run projects that were developed in earlier versions, but earlier versions cannot run projects that were developed or modified in later versions. Opening and modifying a project in a later version will upgrade the project to that version.

> **Note:** We issue each license for a specific *Family* version and *Sub-version* (X.Y), and the license is only valid for that version (including Service Packs). The license is not valid for a newer *Family* version or *Sub-version* of the product. Therefore, if you install a new version, you must also upgrade your license to the version being installed. If you install a Service Pack only, you do not need to upgrade your license.

## Execution Modes

SCADA support the following execution modes:

| Execution Mode | SCADA |
|---|---|
| Evaluation Mode | Supported |
| Demo Mode | Supported |
| Licensed for Engineering Only | Supported |
| Licensed for Runtime Only | Supported |

**Evaluation Mode**

Enables all of the product's engineering and runtime features.

The first time you install EcoStruxure Machine SCADA Expert 2020 on a computer, the product runs for forty (40) hours in Evaluation Mode. This evaluation period includes any time you run a product module (engineering or runtime). You can use this evaluation period continuously or not; for example, 10 hours a day for 4 days, or 5 hours a day for 8 days, or 10 hours a day for 3 days plus 5 hours a day for 2 days, and so on.

After running for 40 hours in the Evaluation Mode, the evaluation period ends and the program automatically converts to Demo Mode until you apply a valid license. You cannot reactivate Evaluation Mode, even if you reinstall the software on your computer.

> **Note:** Each version of EcoStruxure Machine SCADA Expert 2020 has an evaluation period that is independent of every other version. For example, if an earlier version is running in Demo Mode because its evaluation period has expired, and then you install the latest version on the same computer, the latest version will begin its own 40-hour evaluation period and the earlier version will continue to run in Demo Mode.

**Demo Mode**

Allows you to download projects to remote stations and to run projects for testing or demonstration purposes. You can execute runtime tasks and use the debugging tools (*Watch* and *LogWin*), but they shut down automatically after running for two hours continuously. You can restart the Demo Mode again and run for another two hours, and so on.

You cannot create or modify screens, worksheets, or project settings in Demo Mode.

**Licensed for Engineering Only**

Enables all development options for an unlimited time.

This mode also allows you to continuously run the runtime tasks and debugging tools (*Watch* window, *Output* window, and *LogWin* module) for 72 hours. After that period, these tasks shut down, but you can restart them and run for another 72 hours, and so on. You can use this license for development and testing only.

**Licensed for Runtime Only**

Enables all runtime tasks and debugging tools (*Watch* window, *Output* window, and *LogWin* module) for unlimited time, but you cannot create or modify screens and/or worksheets.

The menu options available in Runtime Only mode are the same as the options listed for Demo Mode (see previous table).

**Note:** The Remote Management tool is always available, regardless of the execution mode, so that you can upload files from or download files to remote stations.

To see which execution mode you are currently running, click **About** on the Help tab of the ribbon; the *About* dialog shows the execution mode, including the time remaining if you are in Evaluation Mode.

## About hardkey licenses

An encapsulated chip that must be physically connected to the computer's parallel port (LPT1) or USB interface.

The software license resides in the hardkey, and you cannot share this license simultaneously with more than one other copy of software in the network. If you connect the hardkey to another computer, then you effectively transfer the license to that computer.

Using the parallel port hardkey does not prevent you from connecting another device — such as a printer — to the port. The hardkey should be electronically transparent to other devices connected to the parallel port. You simply connect the hardkey to the computer and then connect the printer cable to the hardkey. However, you may encounter problems if you install more than one hardkey (for different products) on the same parallel port.

> **Note:** Be careful when installing or removing a hardkey from the computer's parallel port. We strongly recommend that you turn off the computer and disconnect it from the power supply before installing or removing a hardkey.

On the other hand, while using the USB hardkey, the USB port cannot be shared with any other device.

### *Install a new hardkey license*

Install a new hardkey license for the full EcoStruxure Machine SCADA Expert 2020 software or the project runtime software.

Before you begin this task, make sure the appropriate software has been installed on your computer or device. For more information, see Installation Guide on page 33.

To install a new hardkey license:

1. On the computer or device where you have installed the software, connect the hardkey to the appropriate port (e.g., USB, SD or microSD).
2. Run the software.

If the software recognizes the hardkey, it will run normally without any alert messages.

If the software does not recognize the hardkey, try the following:

- For the full EcoStruxure Machine SCADA Expert 2020 software, use the Protection Manager utility program (a.k.a. Register) to make sure the software is set to check for a hardkey.

### *Upgrade an existing hardkey license*

Upgrade an existing hardkey license for the full EcoStruxure Machine SCADA Expert 2020 software or the project runtime software.

Before you begin this task, make sure you have access to the full EcoStruxure Machine SCADA Expert 2020 software installed on either your project development workstation or one of your project runtime stations. You need to use the Protection Manager utility program (a.k.a. Register) included with that software in order to perform this task.

Also, you should have already puchased a valid license or license upgrade. You may purchase it when you send in the site code (see below), but doing so might increase your downtime. For more information, contact your EcoStruxure Machine SCADA Expert 2020 software distributor. You can update any license setting (e.g., product type, options), or you can upgrade the software to a new version. The cost of the update/upgrade depends on the difference between the current and new license settings.

Finally, you must have Administrator privileges on the computer in order to run Protection Manager.

To upgrade an existing hardkey license:

1. Exit EcoStruxure Machine SCADA Expert 2020 if it is running.

   EcoStruxure Machine SCADA Expert 2020 and Protection Manager cannot run at the same time.

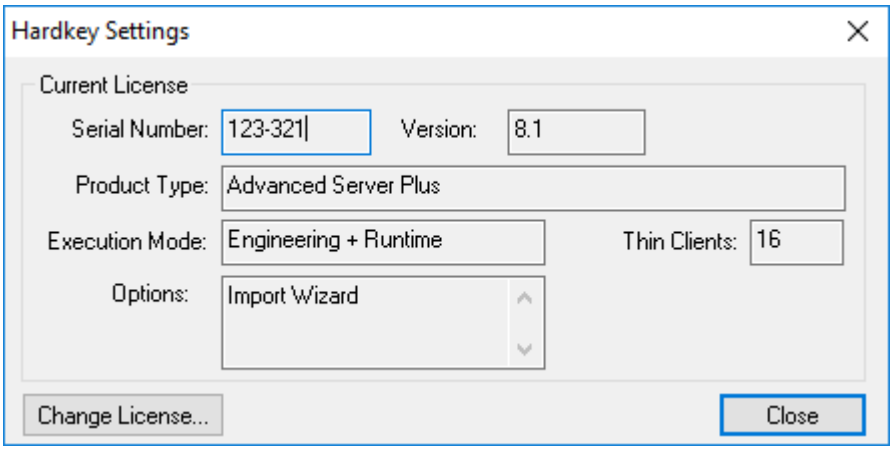
If you have a project running in EcoStruxure Machine SCADA Expert 2020, stop it before you exit the program.

2. Run Protection Manager: click **Start** > **EcoStruxure Machine SCADA Expert 2020** > **EcoStruxure Machine SCADA Expert 2020 Register**.

   The *Protection Manager* program window is displayed.

3. Select **Hardkey** if it is not already selected, and then click **Check**.

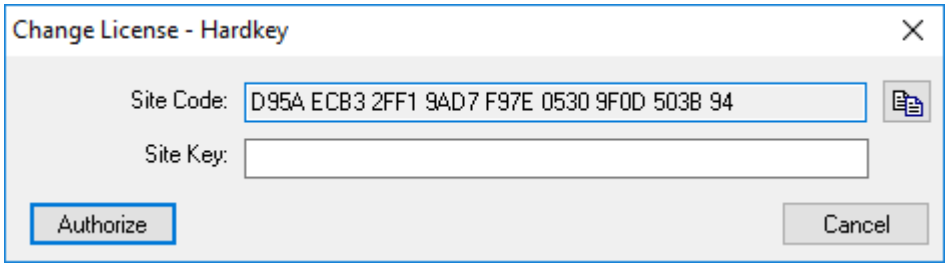
   If you have a valid hardkey license installed — that is, if a valid hardkey is connected to the computer — the *Hardkey Settings* dialog box is displayed with the settings on that hardkey.




   Otherwise, if you do not have a valid hardkey license installed, an alert message is displayed.

4. Click **Change License**.

   The *Change License* dialog box is displayed with a unique site code that is generated from the hardkey itself.




5. Send the site code to your software distributor.

   Typically, you will copy the site code to your clipboard and then paste it into an email to your software distributor. To copy it to your clipboard, click the Copy button to the right of the **Site Code** box.

6. When you receive the corresponding site key from your software distributor, type or paste it in the **Site Key** box, and then click **Authorize**. (You will be prompted to confirm.)
   The new license settings are written to the hardkey, and then a confirmation message is displayed.

If the new site key is not validated, an error message is displayed. If that happens, confirm that you entered the site key correctly. If you entered it correctly and still get an error message, contact your software distributor for further assistance.

## About softkey licenses

When you install the project development or runtime software, the program generates a unique site code. You can send this site code to your software distributor, who will then generate a site key to match your site code. You can then use the site key to install the license on your computer or target device, as opposed to having the license stored on a hardkey.

> **Note:** When you use a softkey, the license is recorded in the computer or device's permanent memory. If the computer is damaged or lost, you will lose the license.

### *Install or upgrade a softkey license for the full EcoStruxure Machine SCADA Expert 2020 software*

Install or upgrade a softkey license for the full EcoStruxure Machine SCADA Expert 2020 software running on a Windows computer.

Before you begin this task, make sure the full EcoStruxure Machine SCADA Expert 2020 software has been installed on the computer. For more information, see #unique_32.

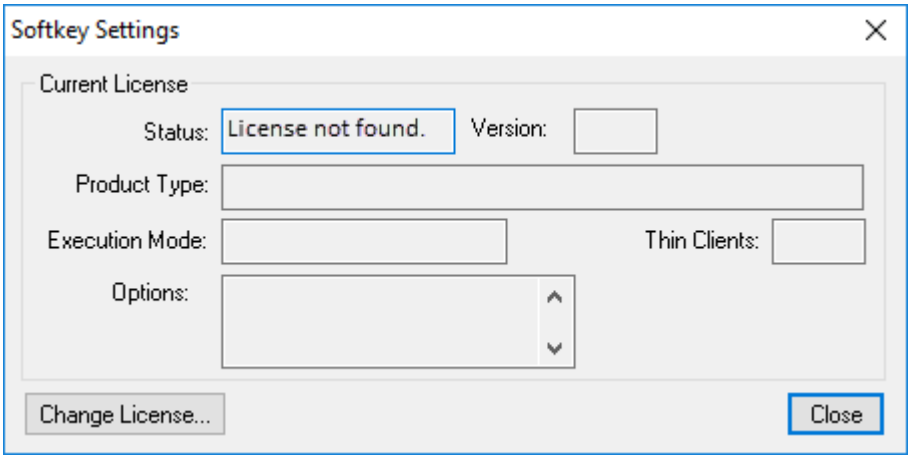
Also, you should have already puchased a valid license or license upgrade. You may purchase it when you send in the site code (see below), but doing so might increase your downtime. For more information, contact your EcoStruxure Machine SCADA Expert 2020 software distributor. You can update any license setting (e.g., product type, number of thin clients), or you can upgrade the software to a new version. The cost of the update/upgrade depends on the difference between the current and new license settings.

Finally, you must have administrator privileges on the computer in order to run the Protection Manager utility.

These instructions apply both to installing a new softkey license and to upgrading an existing softkey license; whatever license you apply will overwrite the existing settings.

To install or upgrade a softkey license for EcoStruxure Machine SCADA Expert 2020:

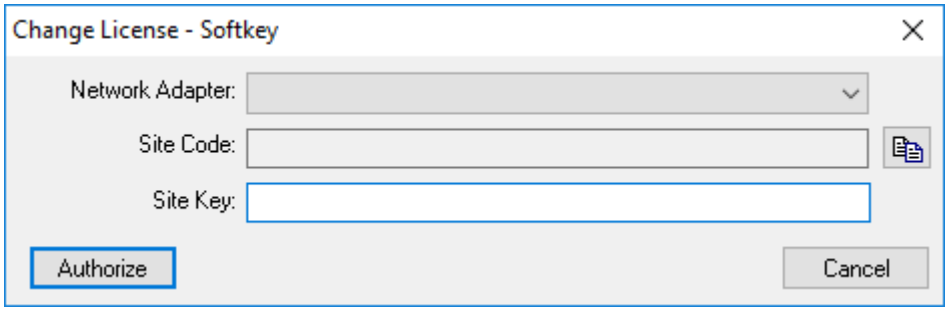
1. In EcoStruxure Machine SCADA Expert 2020, stop the project if it is running, and then exit the program.
2. Run the Protection Manager utility program: in Windows, click **Start** > **EcoStruxure Machine SCADA Expert 2020** > **EcoStruxure Machine SCADA Expert 2020 Register**.
   The program window is displayed.
3. Select **Softkey** if it is not already selected, and then click **Check**.
   The *Softkey Settings* dialog box is displayed. If you have a valid softkey license installed, the dialog box will show those settings. Otherwise, if you do not have a softkey, the status will be "License not found."




***Checking the softkey settings***

4. Click **Change License**.

The *Change License* dialog box is displayed.




***Changing the license***

5. In the **Network Adapter** list, select the network adapter (a.k.a. NIC) that Protection Manager should use to generate the unique site code.

   This option is provided because the site code is generated from the network adapter's MAC address, and in some cases — for example, if the computer is running in a virtual machine or if it is connected to a VPN — the computer might have two or more network adapters that it can use. You should select the network adapter that the computer will use under normal operating conditions. If you select another network adapter and then it becomes unavailable for any reason, your softkey license will become invalid. For more information about the listed network adapters, consult the documentation for the computer itself and the other software installed. Of course, if only one network adapter is listed, you should select that one.

   When the network adapter is selected, the site code is generated and displayed.

6. Send the site code to your software distributor.

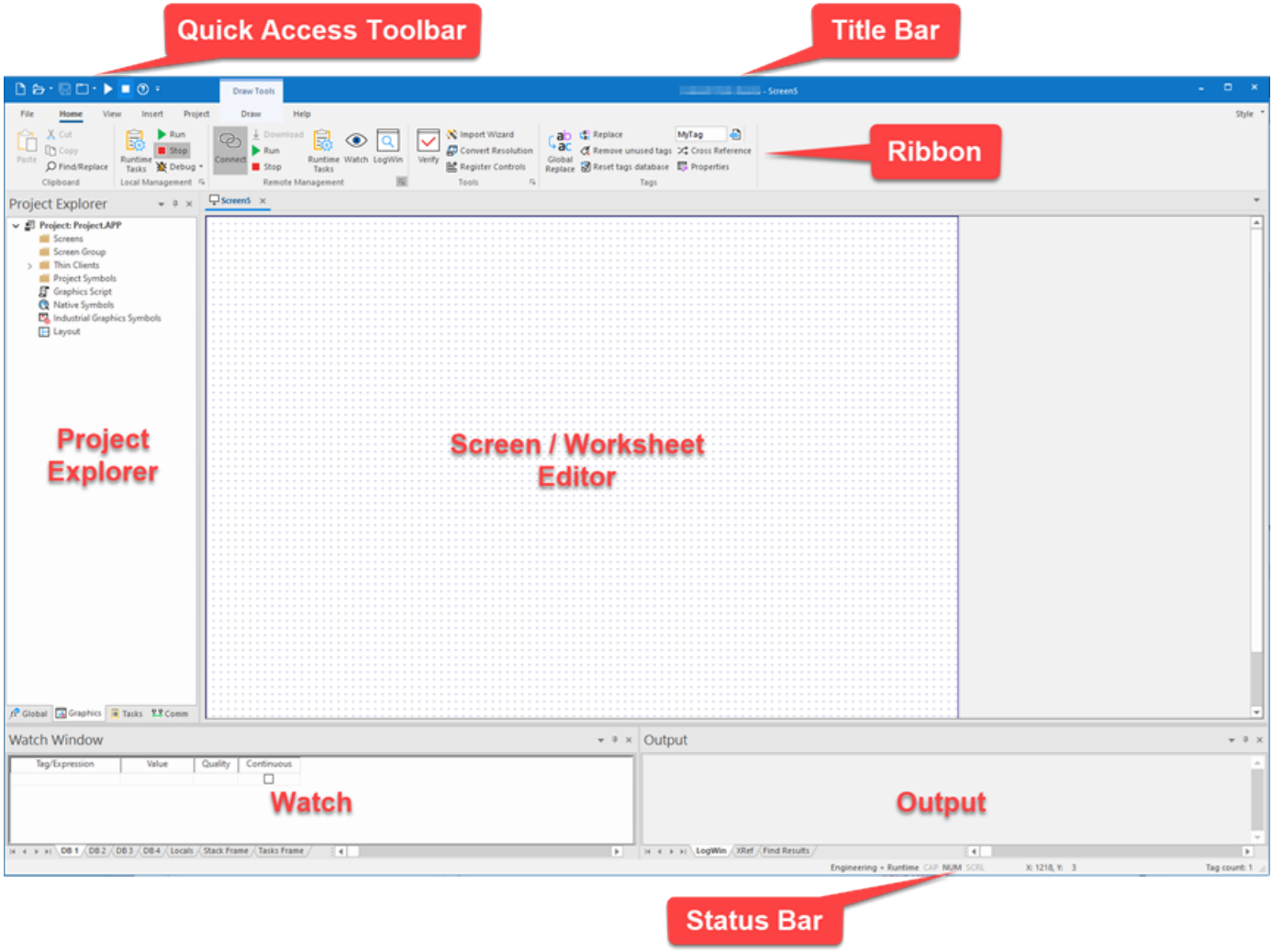
   Typically, you will copy the site code to your clipboard and then paste it into an email to your software distributor. To copy it to your clipboard, click the Copy button to the right of the **Site Code** box.

7. When you receive the corresponding site key from your software distributor, type or paste it in the **Site Key** box, and then click **Authorize**. (You will be prompted to confirm.)
   The new license settings are saved on the computer, and then a confirmation message is displayed.

If the new site key is not validated, an error message is displayed. If that happens, confirm that you entered the site key correctly. If you entered it correctly and still get an error message, contact your software distributor for further assistance.

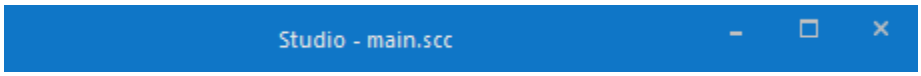
# The Development Environment

EcoStruxure Machine SCADA Expert 2020 incorporates a modern, Ribbon-based Windows interface to provide an integrated and user-friendly project development environment.



## Title bar

The Title Bar located along the top of the project development environment displays the full name of the Studio application (e.g., EcoStruxure Machine SCADA Expert 2020), followed by the name of the active screen or worksheet (if any).




***Example of Title Bar***

The Title Bar also provides the following buttons (from left to right):

- **Minimize** button: Click to minimize the development environment window to the Taskbar.
- **Restore Down** / **Maximize**: Click to toggle the development environment window between two sizes:
  - **Restore Down** button reduces the window to its original (default) size.
  - **Maximize** button enlarges the window to fill your computer screen.
- **Close** button: Click to save the database and then close the development environment. If you modified any screens or worksheets, the application prompts you to save your work. This button's function is similar to clicking **Exit Application** on the File menu.

> **Note:** Closing the project development environment does not close either the project runtime or the project viewer, if they are running.

## Quick Access Toolbar

The Quick Access Toolbar is a customizable toolbar that contains a set of commands that are independent of the ribbon tab that is currently displayed.

### Move the Quick Access Toolbar

The Quick Access Toolbar can be located in one of two places:

- Upper-left corner, above the menu bar (default location); or
- Below the ribbon, where it can run the full length of the application window.

If you don't want the Quick Access Toolbar to be displayed in its current location, you can move it to the other location:

1. Click **Customize Quick Access Toolbar** .
2. In the list, click **Show Below Ribbon** or **Show Above Ribbon**.

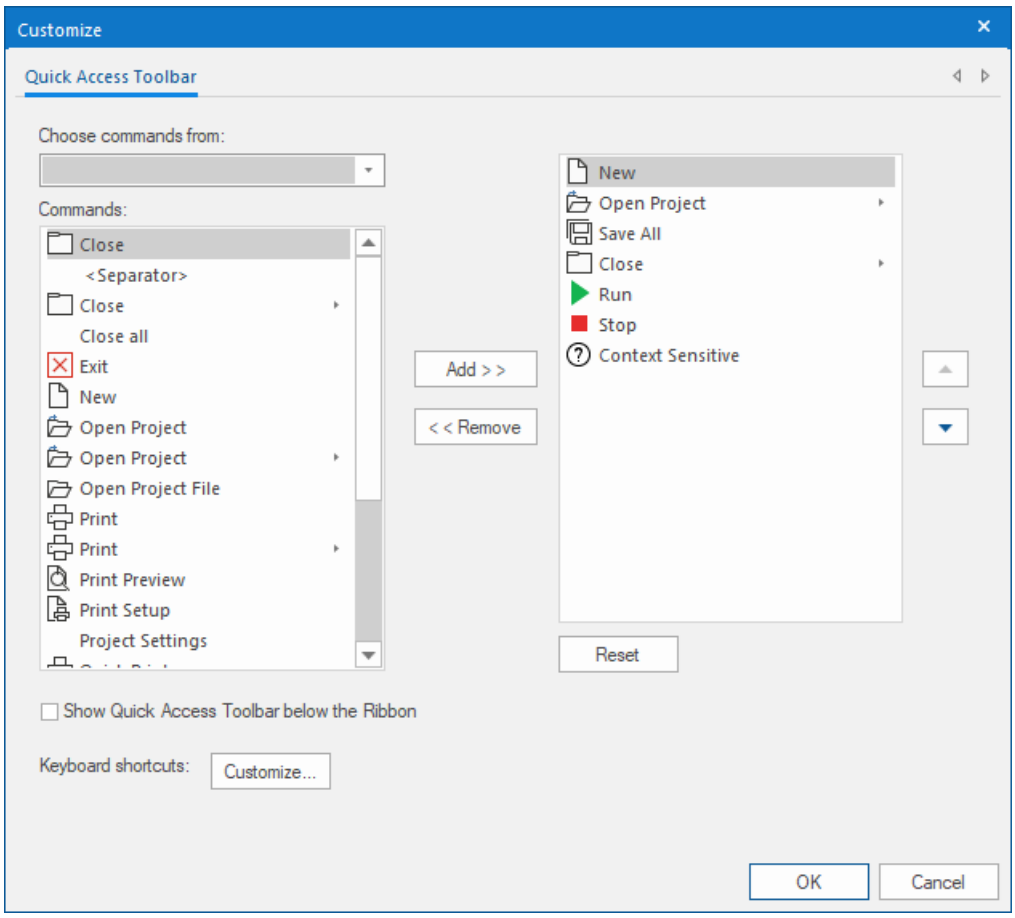
### Add a command to the Quick Access Toolbar

You can add a command to the Quick Access Toolbar directly from commands that are displayed on the ribbon:

1. On the ribbon, click the appropriate tab or group to display the command that you want to add to the Quick Access Toolbar.
2. Right-click the command, and then click **Add to Quick Access Toolbar** on the shortcut menu.

You can also add and remove commands — as well as reset the toolbar to its default — using the *Customize* dialog:

1. Click **Customize Quick Access Toolbar** .
2. In the list, click **More Commands**. The *Customize* dialog is displayed.




3. In the **Choose commands from** menu, select the appropriate Ribbon tab. The commands from that tab are displayed in the **Commands** list.
4. In the **Commands** list, select the command that you want to add to the Quick Access Toolbar.
5. Click **Add**.

Only commands can be added to the Quick Access Toolbar. The contents of most lists, such as indent and spacing values and individual styles, which also appear on the ribbon, cannot be added to the Quick Access Toolbar.

## File menu

When you click the **File** tab of the ribbon, it opens a menu of standard Windows application commands like New, Open, Save, Print, and Close.

### *New*

The **New** command on the File menu is used to create a new worksheet file or project.

The *New* dialog (see the following figures) contains two tabs:

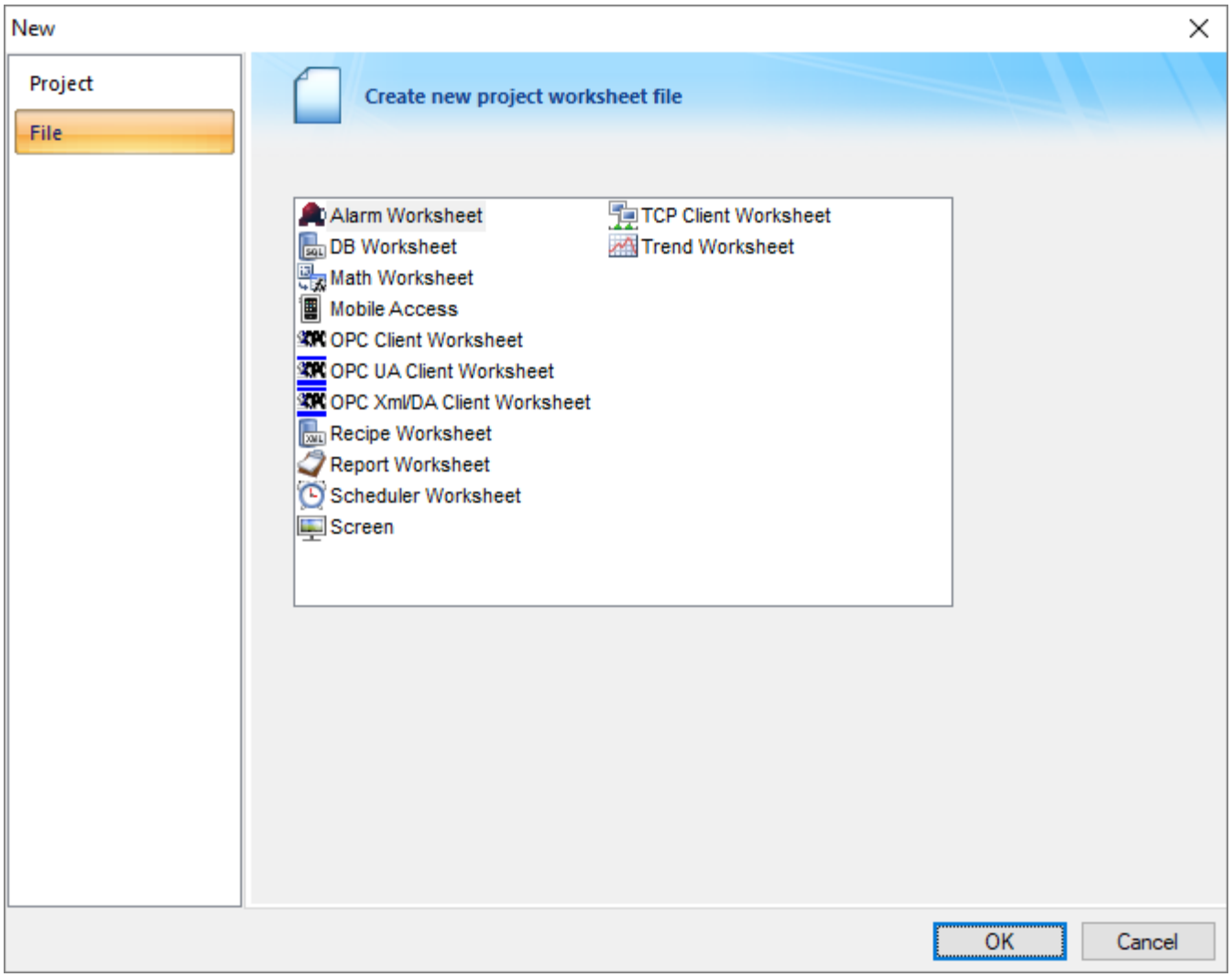
- **File** tab: Select this tab to create new worksheets or screens for an open project.
- **Project** tab: Select this tab to create a new project.

Instructions for creating new files and projects follow.

#### Creating a New File

To create a new worksheet or screen:

1. Click the **File** tab.




***New File tab***

2. Select **Display** or a **Worksheet** type from the list.
3. Click **OK**.

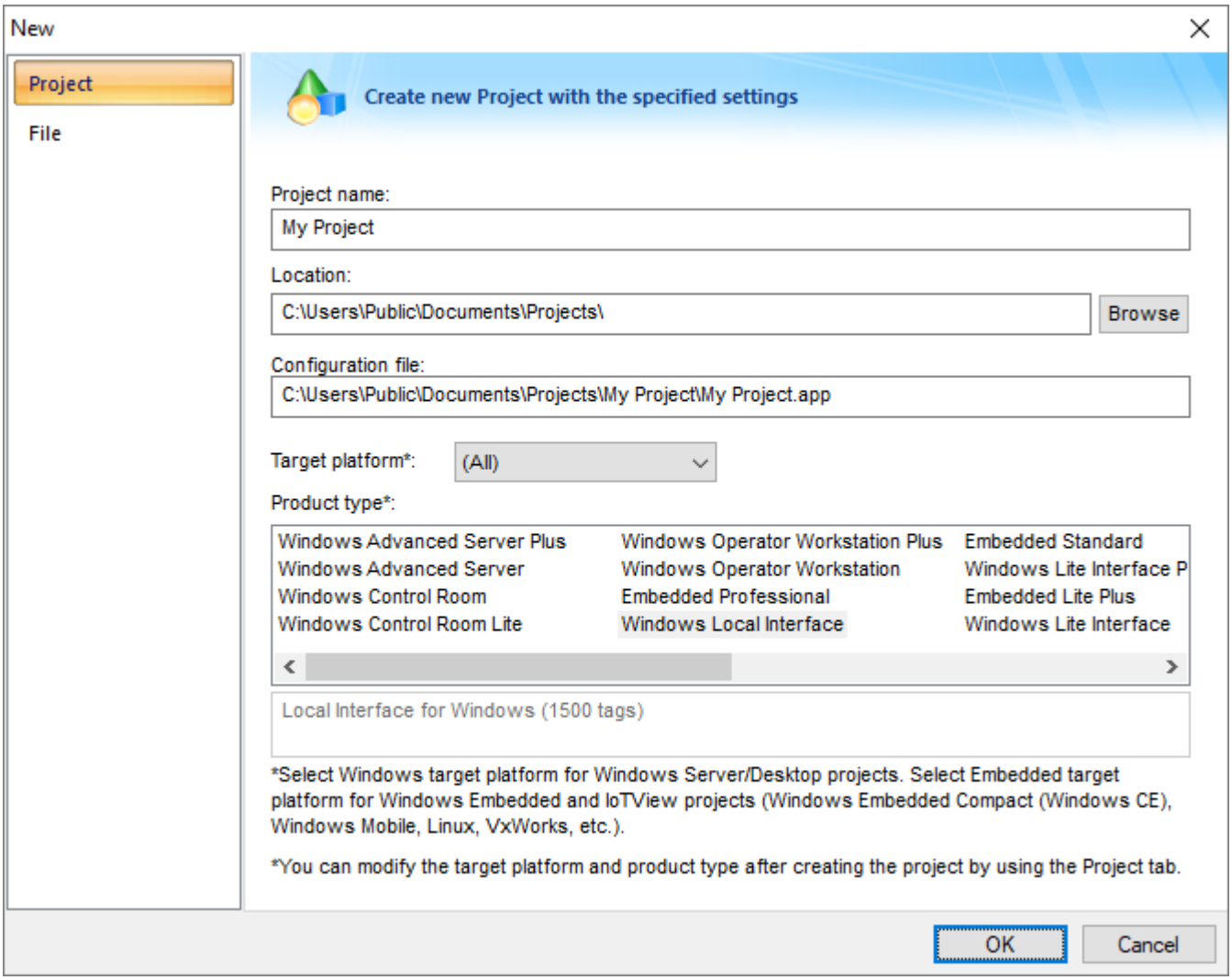
   The *New* dialog is closed and your selection is opened in the worksheet editor.

> **Note:** When you add an I/O driver to the project, an associated option allows you to open a new driver worksheet. You also can create new screens or worksheets by right-clicking on the folder in the *Project Explorer* and selecting the **Insert** option from the shortcut menu.

### Creating a New Project

To create a new project:

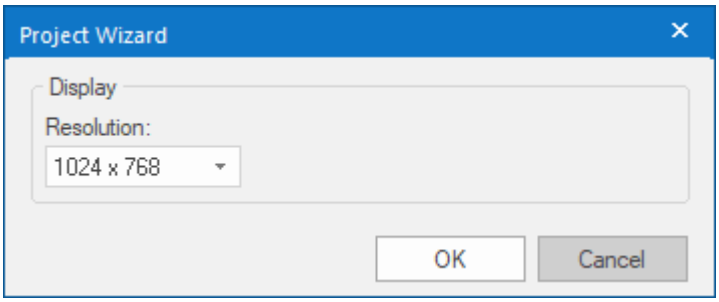
1. Click the **Project** tab.




***New Project tab***

2. In the **Project name** box, type a name for your project.
3. By default, EcoStruxure Machine SCADA Expert stores all projects in the location specified by the Default Project Path preference (**Preferences** on the Project tab of the ribbon), so that path will be automatically displayed in the **Location** box. To save your project in another location, click **Browse** and then select a folder.
4. Select a **Target platform**.

5. Click **OK** to continue to the *Project Wizard* dialog.




***Project Wizard***

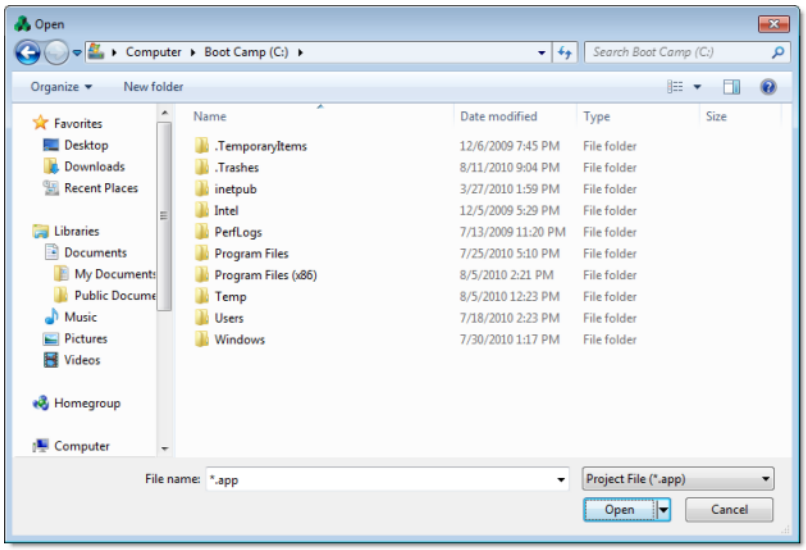
6. In the **Resolution** box, select a screen resolution. If you select Custom, then also type the width and height in pixels.
7. To share tags with another PC-based control application, select the application type from the list and click the **Configure** button. (Each type has its own configuration options; please consult the application vendor.) Otherwise, leave it set to **<None>**.
8. Click **OK** when you're done.

For a more detailed walkthrough, see Creating a new project.

## *Open Project*

The **Open Project** command on the File menu is used to open a saved project.

Selecting the command opens a standard Windows *Open* dialog, which you can use to locate and open the project file (`*.app`).

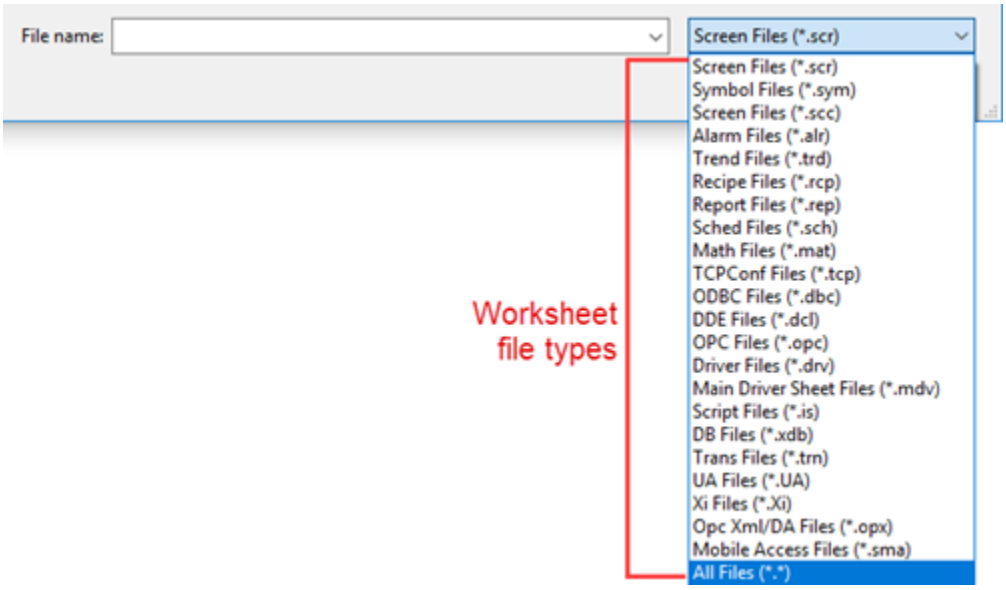



***Open dialog***

### Open

The **Open** command on the File menu is used to open a saved worksheet file.

Selecting the command opens a standard Windows *Open* dialog, which you can use to locate and open the worksheet file. The application can open many different file types, so use the **File type** combo-box to filter the files.




***Available worksheet file types in the Open dialog***

### Save

The **Save** command on the File menu is used to save the active screen or worksheet.

The command becomes available only after you modify the worksheet in some way.

### Save As

The **Save As** command on the File menu is used to open a save the active screen or worksheet at another location.

### Save All

The **Save All** command on the File menu is used to save all open worksheet files.

The command becomes available only after you modify the a worksheet in some way.

### Save All as HTML

The **Save All as HTML** command on the File menu is used to save all of your project's screens and screen groups in HTML format.

After saving, the files can be found in the **Web** folder in the Project Explorer. For more information, see Thin Clients and Mobile Access on page 727.

> **Note:** You must close all worksheets before you execute this command.

### Save as HTML

The **Save as HTML** command on the File menu is used to save the active screen in HTML format.

After saving, the file can be found in the **Web** folder in the Project Explorer. For more information, see Thin Clients and Mobile Access on page 727.

### Save Screen Group as HMTL

The **Save Screen Group as HTML** command on the File menu is used to save a selected screen group in HTML format.

After saving, the files can be found in the **Web** folder in the Project Explorer. For more information, see Thin Clients and Mobile Access on page 727.

### Close

The **Close** command on the File menu is used to close the active screen or worksheet.

When you select this command, you will be prompted to save your changes before closing.

### Close All

The **Close All** command on the File menu is used to close all open screens and worksheets.

When you select this command, you will be prompted to save your changes before closing.

### Recent Projects

The Recent Projects area of the File menu lists the most recently opened projects.

To open one of the listed projects, simply click it.

### Print

The **Print** command on the File menu is used to print the active screen or worksheet.

Selecting the command opens a standard Windows *Print* dialog, which you can use to adjust the print range and the number of copies.

### Exit

The **Exit** command on the File menu is used to close all open screens and worksheets, save the project database, and then exit the application.

When you select this command, you will be prompted to save your changes before closing.

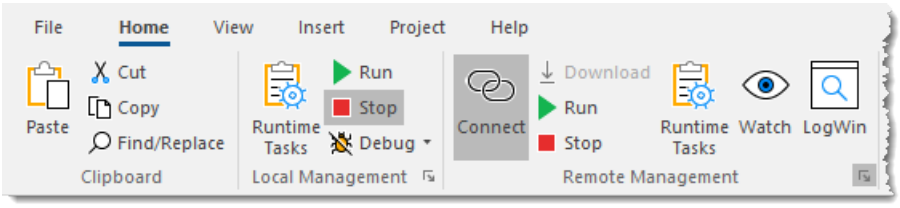
> **Note:** Selecting this command is the same as clicking the Close button on the title bar.

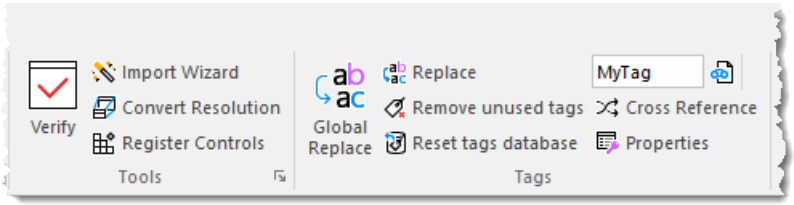
## Ribbon

The new ribbon combines the numerous menus and toolbars from the previous versions of this software into a single, user-friendly interface. Almost all application commands are now on the ribbon, organized into tabs and groups according to general usage.

### *Home tab*

The **Home** tab of the ribbon is used to manage your project within the development environment.



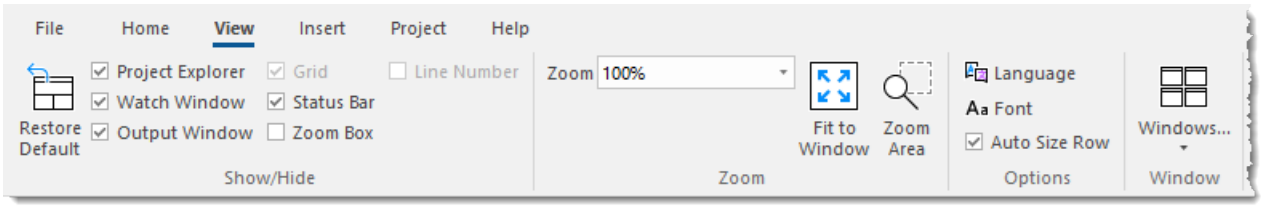




The tools are organized into the following groups:

- **Clipboard**: Cut, copy, paste, and find items in project screens and task worksheets.
- **Local Management**: Run and stop the project on the local station (i.e., where the development application is installed), as well as manage the execution tasks. You can also run a project in Debug mode, for debugging VBScript.
- **Remote Management**: Connect to a remote station so that you can download the project to it, and then run, stop, and troubleshoot the project on that station.
- **Tools**: Miscellaneous tools to verify the project, import tags from other projects, convert screen resolutions, and register ActiveX and .NET controls.
- **Tags**: Manipulate tags and tag properties in the project database.

### *View tab*

The **View** tab of the ribbon is used to customize the look of the development environment itself.



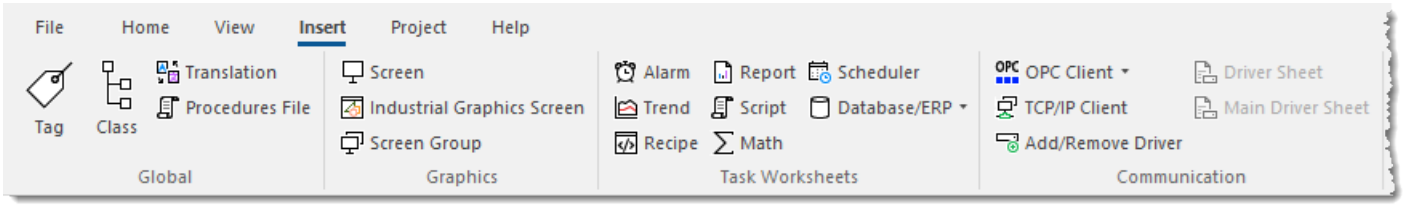

The tools are organized into the following groups:

- **Show/Hide**: Show and hide the different parts of the development environment, as well as restore the default layout.

- **Zoom**: Zoom in and out of the screen editor.
- **Options**: Change the language and font used in the development environment.
- **Window**: Arrange the windows in the development environment.

### *Insert tab*

The **Insert** tab of the ribbon is used to insert new tags, screens, worksheets, and other components into your project.

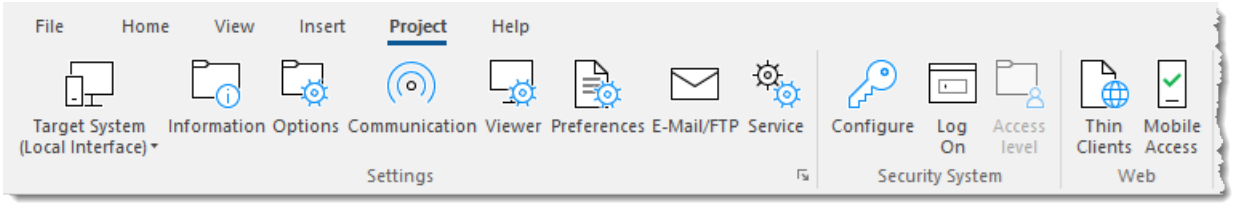



The tools are organized into the following groups:

- **Global**: Insert tags, classes, translations, and procedures into the Global tab of the Project Explorer.
- **Graphics**: Insert screens and screen groups into the Graphics tab of the Project Explorer.
- **Task Worksheets**: Insert task worksheets into the Tasks tab of the Project Explorer.
- **Communication**: Insert server configurations and communication worksheets into the Comm tab of the Project Explorer.

### *Project tab*

The **Project** tab of the ribbon is used to configure your project settings.

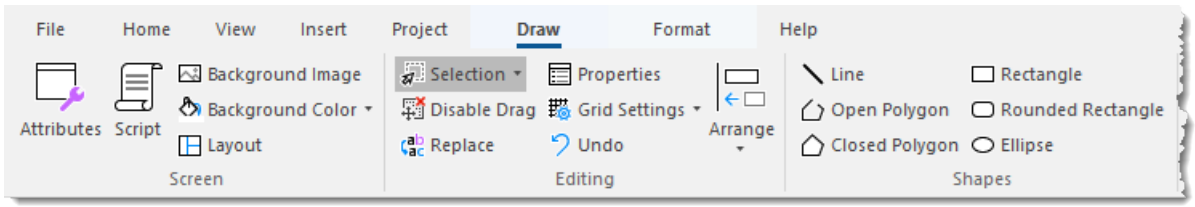


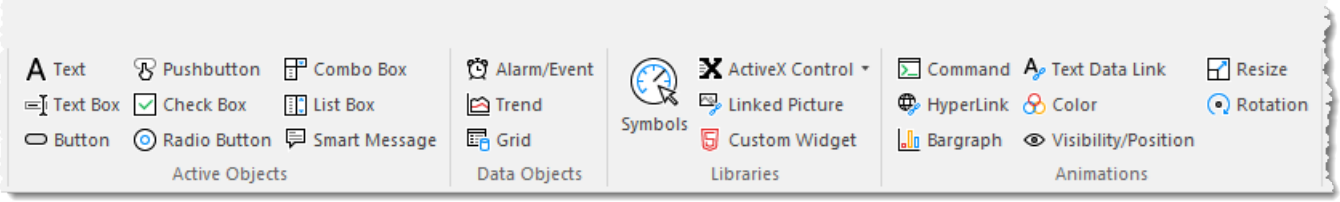

The tools are organized into the following groups:

- **Settings**: Configure the general project settings or set the project to run as a Windows service.
- **Security System**: Enable and configure the project security system.
- **Web**: Configure the project to accept connections from a variety of thin clients.

### *Draw tab*

The **Draw** tab of the ribbon is used to draw objects in project screens.





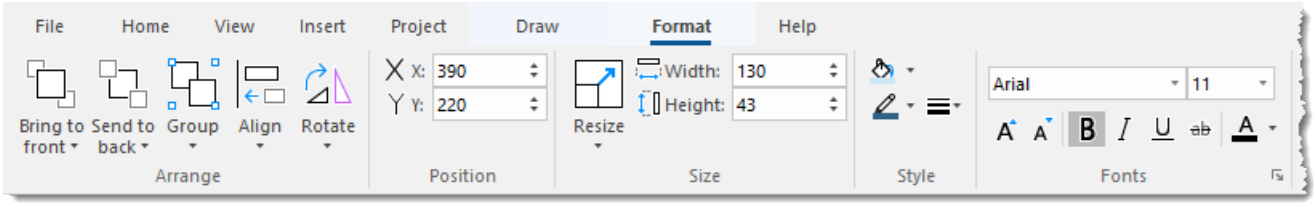

> **Note:** This tab is available only when you have a project screen open for editing.

The tools are organized into the following groups:

- **Screen**: Configure settings for the project screen itself, such as its attributes, script, and background color or image.
- **Editing**: Select and edit objects in the project screen.
- **Shapes**: Draw static lines and shapes.
- **Active Objects**: Draw active objects, like buttons and check boxes.
- **Data Objects**: Draw objects that display historical data, like alarms, events, and trends.
- **Libraries**: Select from libraries of premade objects, such as symbols, ActiveX and .NET controls , external image files, and HTML5-based custom widgets.
- **Animations**: Apply animations to other screen objects.

### *Format tab*

The **Format** tab of the ribbon is used to format and arrange objects in a project screen.



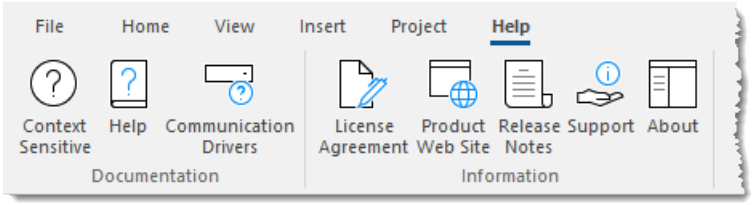

> **Note:** This tab is available only when you've selected one or more objects in a project screen.

The tools are organized into the following groups:

- **Arrange**: Arrange objects in a project screen, including bring to front and send to back, group, align, and rotate.
- **Position**: Precisely adjust the position of a screen object in a project screen.
- **Size**: Precisely adjust the size of a screen object.
- **Style**: Change the fill and line color of a screen object.
- **Fonts**: Change the caption font of a screen object.

### *Help tab*

The **Help** tab of the ribbon provides additional help with using the software.




The tools are organized into the following groups:

- **Documentation**: Access the documentation for the development application, including this help file / technical reference and notes for the individual communication drivers.
- **Information**: Access other information about EcoStruxure Machine SCADA Expert 2020, including the license agreement, product website, and release notes, as well as support details that make it easier for us to assist you.

## Project Explorer

The Project Explorer organizes all of the screens, worksheets, and other items that comprise your project and presents them in an expandable tree-view.

To open a folder and view its contents, either click the Expand icon ▷ to the left of the folder or double-click the folder itself.

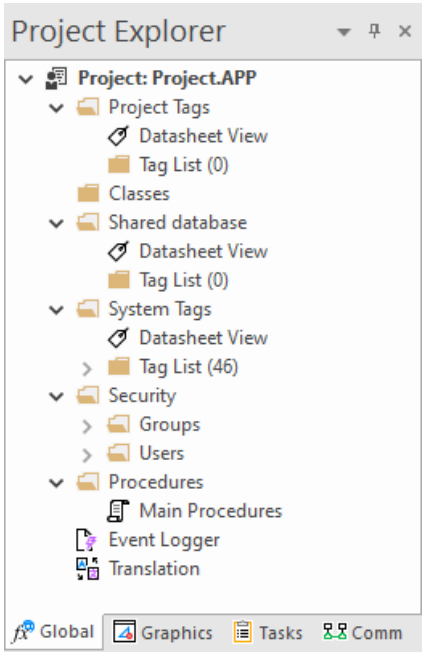
To close a folder, click the Collapse icon ◢ to the left of the folder.

If you right-click any item in the Project Explorer, then a shortcut menu will appear with contextual commands for that item.

There are four main sections, or tabs, in the Project Explorer: Global, Graphics, Tasks, and Comm.

### *Global tab*

The Global tab of the Project Explorer contains the project tags database, as well as other features that apply to the entire project such as the security system, VBScript procedures, and UI translation.




***Global tab of the Project Explorer***

The folders on the Global tab are described in the following sections:

**Project Tags**

The project tags database contains all of the data tags that you create during project development, such as screen tags (e.g., **`button1_state`**) or tags that read from / write to connected devices.

**Classes**

Classes are compound tags that you can create to associate a set of values, rather than a single value, with an object. For example, where you may normally create separate tags for a tank's pressure, its temperature, and its fill level, you can instead create a "tank" class that includes all three.

**Shared Database**

The shared database contains tags that were created in another program and then imported into or integrated with your project.

**System Tags**

System tags are predefined values such as the date, the time, the name of the current user, and so on. You can use these values to develop supervisory functions and housekeeping routines.

All system tags are read-only, which means you cannot add, edit, or remove these tags from the database.

**Security**

If you choose to enable it, you can use the project security system to control who may log on to your project and what they may do during runtime.

**Procedures**

Procedures are VBScript functions and sub-routines that can be called by any other script in your project.

**Event Logger**

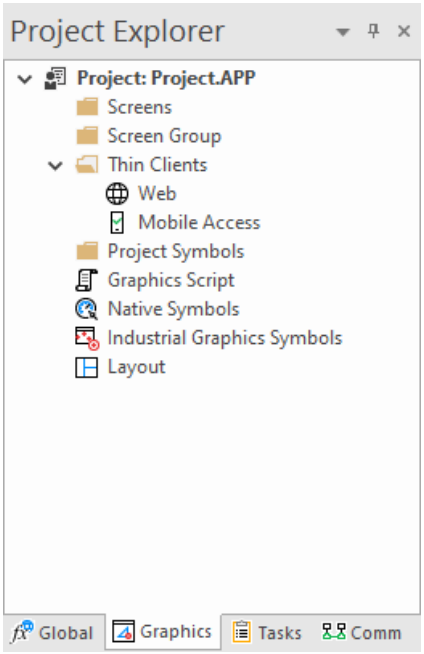
The event logger saves important runtime messages and task results to an external database.

**Translation**

You can use the translation table to develop a multilingual user interface (MUI) for your project.

### *Graphics tab*

The Graphics tab of the Project Explorer contains all of the screens, screen groups, and symbols in your project.




***Graphics tab of the Project Explorer***

The folders on the Graphics tab are described in the following sections:

**Screens**

You create screens to provide a graphical interface for your project. Each screen can contain many buttons, sliders, dials, indicators, graphs, and so on.

**Screen Groups**

You can combine individual screens into screen groups, so that they all open together at the same time.

**Thin Clients**

You can deploy your project as a web application to be accessed by thin clients such as desktop web browsers, tablets, and smartphones. You can even deploy different versions of your project with different levels of functionality for each type of client.

**Project Symbols**

This folder contains all of the custom symbols that you create for your project. A symbol is a group of interconnected screen objects that work together to perform a single function — for example, lines, rectangles, and text fragments that have been arranged to make a slider control.

**Graphics Script**

You can use this worksheet to define VBScript sub-routines that are called only when the graphics module starts (i.e., when a client station connects to the server and displays the graphical interface), while it is running, and when it ends.

**Native Symbols**

This folder is a library of the symbols that are created with the native graphics tools in Studio. It contains not only the custom symbols that you create (see Project Symbols above), but also a large selection of premade symbols that are installed with Studio.

**Industrial Graphics Symbols**

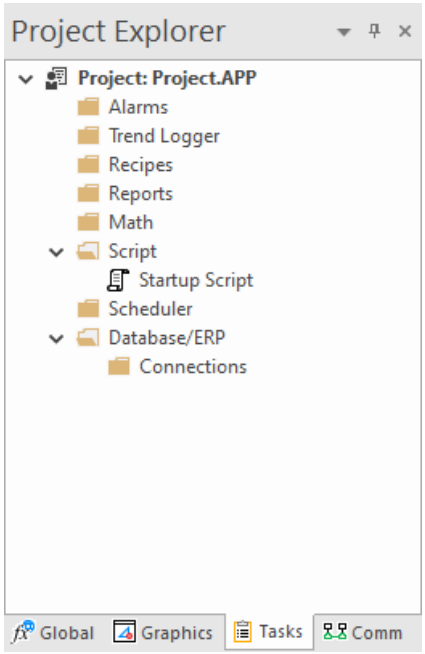
This folder is a library of the symbols that are created with the Industrial Graphics editor, which works as a companion to the native graphics tools in Studio.

**Layout**

The layout editor displays all of the screens the are currently open for editing. You can use it to visualize how the screens are arranged together and reuse screens in multiple layouts — for example, to create a common navigation bar across your entire project.

### *Tasks tab*

The Tasks tab of the Project Explorer organizes the worksheets that are processed as background tasks (i.e., server-based maintenance tasks that are not directly related to screen operations or device I/O) during project runtime.




***Tasks tab of the Project Explorer***

The folders on the Tasks tab are described in the following sections:

### Alarms

You can use Alarm worksheets to define when alarms are trigged, how they must be handled, and what messages they generate.

(You can then use the Alarm/Event Control screen object to display your alarms on screen, but that is a separate procedure.)

### Trend Logger

You can use Trend worksheets to select project tags that should be displayed as data trends and/or saved as historical data.

(You can then use the Trend Control screen object to actually display your trends on screen, but that is a separate procedure.)

### Recipes

You can use Recipe worksheets to select project tags that will load values from and/or save values to an external file. These worksheets are typically used to execute process recipes, but you can store any type of information such as passwords, operation logs, and so on.

(You can then call the `Recipe` function to actually run a configured Recipe worksheet, but that is a separate procedure.)

### Reports

You can use Report worksheets to design runtime reports that are either sent to a printer or saved to disk.

(You can then call the `Report` function to actually run a configured Report worksheet, but that is a separate procedure.)

### Math

You can use Math worksheets to develop complex runtime logic using the built-in scripting language.

### Script

You can use Script worksheets to develop complex runtime logic using VBScript.

### Scheduler

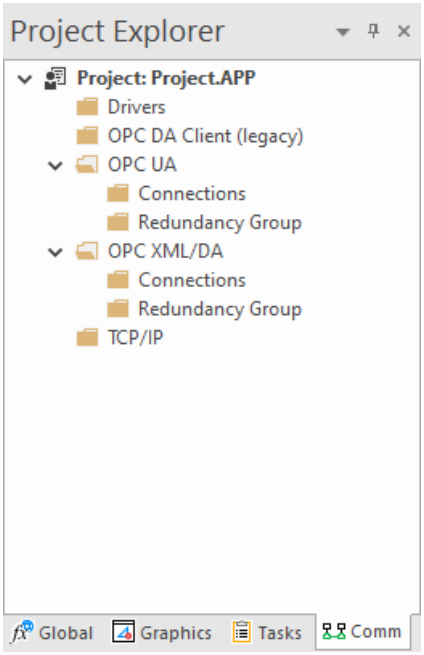
You can use Scheduler worksheets to run commands at specified times, dates, or trigger events.

### Database/ERP

You can use Database worksheets to set up connections and exchange data with external databases using the standard ADO.NET interface.

### *Comm tab*

The Comm tab of the Project Explorer organizes the worksheets that control communication with remote devices, using either direct communication drivers or other common protocols.




***Comm tab of the Project Explorer***

The folders on the Comm tab are described in the following sections:

**Drivers**

You can use Driver worksheets to communicate with PLCs and other hardware, using any of the hundreds of direct communication drivers that are installed with the development application.

**OPC DA 2.05**

You can use OPC worksheets to communicate with OPC servers via the OPC Classic protocol.

**OPC UA**

You can use OPC UA worksheets to communicate with OPC servers via the new OPC Unified Architecture protocol.

**OPC XML/DA**

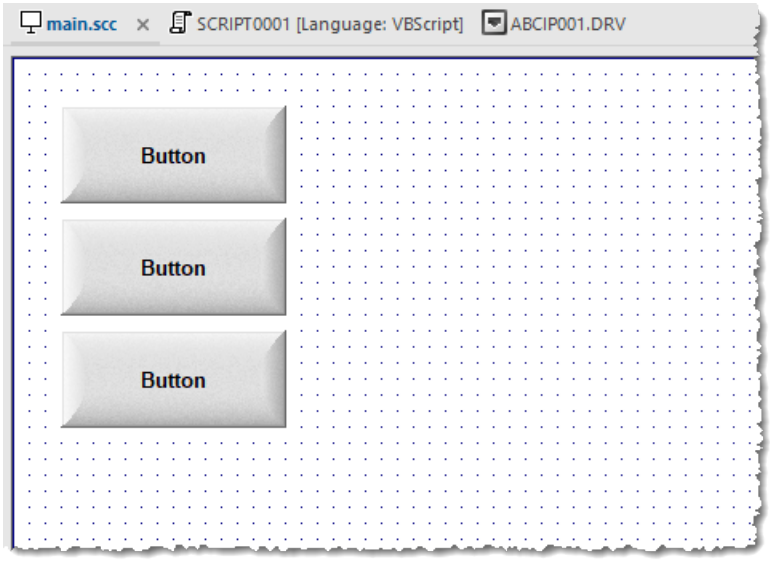
You can use OPC XML/DA worksheets to communicate with OPC servers via the new OPC XML-DA protocol.

**TCP/IP**

You can use TCP/IP worksheets to configure communication between your own project and other projects. The TCP/IP Client and TCP/IP Server modules enable two or more projects to keep their databases synchronized using the TCP/IP protocol.

## Screen/Worksheet Editor

Use the powerful, object-oriented screen editor to create and edit a variety of screens and worksheets for your projects. You can input information using your mouse and keyboard, output control data to your processes, and automatically update screens based on data input from your processes.



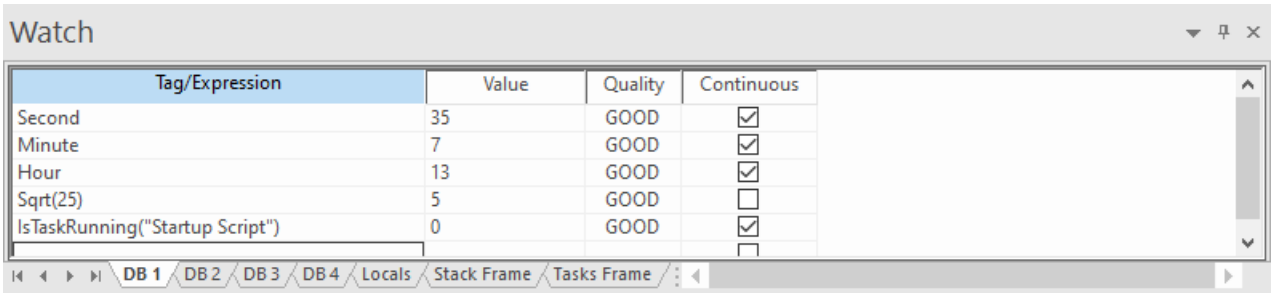

***Screen/Worksheet Editor***

Other screen editor features include:

- Simple point-and-click, drag-and-drop interface
- Grouping objects to preserve the construction steps of individual objects
- Editing objects without having to ungroup internal object components or groups
- Handling bitmap objects and background bitmaps
- Status line support in project windows and dialogs

## Watch window

The *Watch* window is a debugging tool that lets you: watch and force values to project tags; execute and test functions; and execute and test math expressions.




***Example of the Watch window***

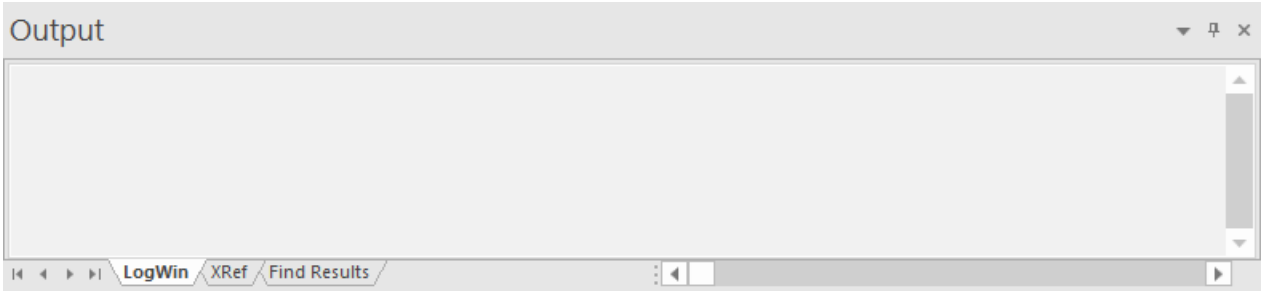
The *Watch* window contains the following elements:

- For each item that you want to watch during project run time:
  - **Tag/Expression**: Specify a project tag, system tag, or expression that you want to watch.
  - **Value**: Displays the value returned by the tag/expression.
  - **Quality**: Displays the quality (GOOD or BAD) of the value returned by the tag/expression.
  - **Continuous**: Select this option to have the project continuously evaluate the tag/expression.
- **DB** tabs: You can use these tabs to organize the items you are watching, so that you do not need to scroll through one long list of items.
- **Locals**, **Stack Frame**, and **Tasks Frame** tabs: These tabs are used to debug VBScript.
- Scroll bars: Use to view areas of the *Watch* window that are obscured from view because of the window size or the size of the current sheet.

> **Tip:** The *Watch* window is dockable, which means you can move it to another location in the project development environment. Click on the titlebar and drag it to a new location. Release the mouse button to attach or dock the window to its new location.

## Output window

Use the *Output* window to view additional information about your project. By default, the window is located in the bottom-right corner of the project development environment.




***Output window***

The *Output* window has three tabs:

- The **LogWin** tab displays the log messages that are generated by your project. You can select exactly which types of messages are displayed, but generally speaking, the log includes run-time messages from the tags database, the communication drivers, the background tasks, the project security system, and so on, as well as certain "housekeeping" messages generated by the project development environment itself. You can use these messages to test and debug your project.

  > **Note:** The *Output* window cannot display the log for a project running on a remote computer. It also cannot print or save log messages. If you want to do either of those things, use the **LogWin** command instead.

- The **XRef** tab displays the results of using the **Cross Reference** command to find where a specific tag is used in your project. The results include the file path and name of the worksheet in which the tag is used, as well as the column and row in the worksheet. So, if something changes in the tag and produces unexpected or unsuccessful results, you can locate all instances of the tag for debugging purposes.
- The **Find Results** tab displays the results of using the **Global Find** command.

The *Output* window is dockable, which means you can drag it to another location in the project development environment.

## Status bar

The Status Bar located along the bottom of the development environment provides information about the active screen (if any) and the state of the application.

***Example of Status Bar***

The Status Bar fields (from left to right) are described in the following table:

| Field | Description |
|---|---|
| Execution Mode | The current execution mode of the application. |
| CAP | Indicates whether the keyboard **Caps Lock** is on (black) or off (grey). |
| NUM | Indicates whether the keyboard **Num Lock** is on (black) or off (grey). |
| SCRL | Indicates whether the keyboard **Scroll Lock** is on (black) or off (grey). |
| Object ID | The ID number of a selected screen object. |
| Cursor Position | The location of the cursor on the active screen or worksheet. If it's a screen, then the position of the *mouse* cursor is given as X,Y coordinates, where X is the number of pixels from the left edge of the screen and Y is the number of pixels from the top edge of the screen. If it's a worksheet, then the position of the *text* cursor is given as Line and Column. |
| Object Size | The size (in pixels) of a selected screen object, where W is the width and H is the height. |
| No DRAG | Indicates whether dragging is disabled (**No DRAG**) or enabled (empty) in the active screen. |
| Tag Count | The total number of tags used so far in the project. |

# Standard Interfaces

## *Object Properties dialog box*

The *Object Properties* dialog box shows the configurable properties of a screen object or animation. Each type of object has its own object-specific properties, but all types have a few properties in common.

### Accessing the dialog box

To access the *Object Properties* dialog box for a screen object, do one of the following:

- Select the screen object, and then on the **Draw** tab of the ribbon, in the **Editing** group, click **Properties**;
- Right-click the screen object, and then click **Properties** on the shortcut menu; or
- Double-click the screen object.

### The dialog box in detail

All *Object Properties* dialog boxes contain the following elements:

**(Pin)**

Click this button to "pin" the dialog box, so that it remains open and active when you select other objects in the screen editor. For more information, see Focusing the Object Properties Window on page 95.

**Replace**

Click this button to open the *Replace* dialog box, which you can use to replace strings, tags, or properties in the selected object. For more information, see Replacing project tags in a document or screen object on page 92.

**Hint**

Type a hint or tooltip that will be displayed during run time, when the user hovers the mouse cursor over the object. This can be used to provide quick-help to the user.

The text in the **Hint** box is also temporarily written to the system tag **`Hint`**, so that you can trigger actions based on the value of this tag when the mouse cursor is moved over a specific object.

To show hints/tooltips during run time, the **Enable Tooltip** option must be selected in the project settings. You can enable/disable this feature separately for full project viewers (on the **Project** tab of the ribbon, in the **Settings** group, click **Viewer**) and for thin clients (on the **Project** tab of the ribbon, in the **Web** group, click **Web**).

**(Object Selector)**

Use this list at the top-right corner of the dialog box to select the specific object or animation in a group of objects that you want to configure. When you select another object, the dialog box immediately changes to show the properties of that object.

## *Color Interface*

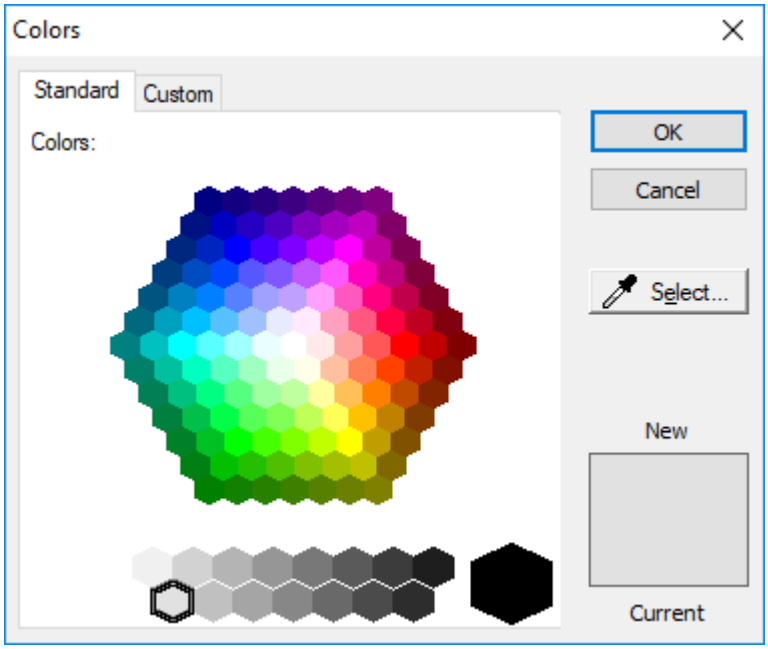
You can edit the color of a component with the Color interface.

1. Click the icon in the toolbar.

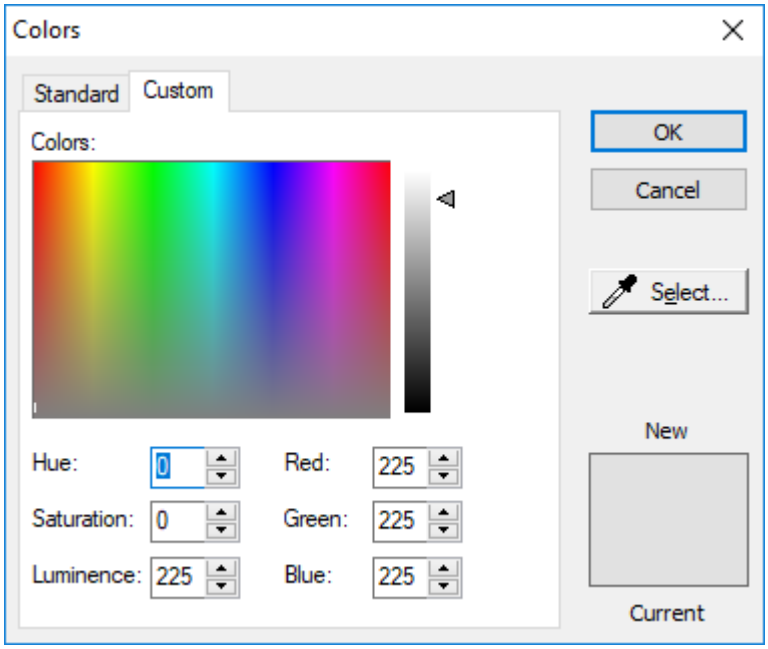
2. Click the desired color from the twenty that display when the pop-up box opens:

The selected color will be applied to the component that you are editing.

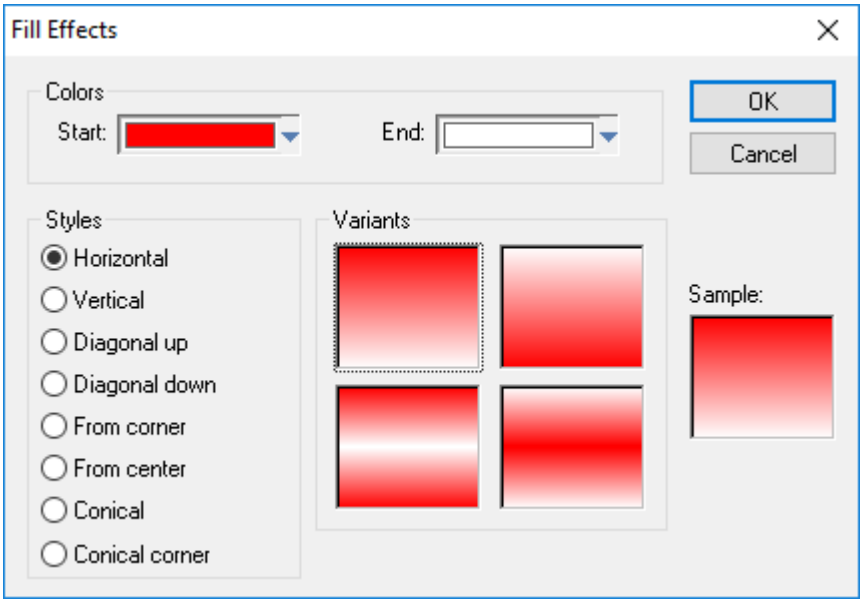
3. Click **More Colors**... if you want to apply a different color. The Colors dialog will open, displaying the 143 standard colors from your operating system.



4. Click the **Custom** tab to edit the HSL (Hue, Sat, Lum) or RGB (Red, Green, Blue) codes of any of the 143 standard colors, creating a custom color.




5. Click **OK** to apply the selected color to the component that is being edited.
6. Depending on the component that you are editing, the **Fill Effects** option is available from the pop-up interface (see step 2 above). Click this option to apply gradient colors with different styles and variants. The *Fill Effects* dialog will open.




7. Select two colors in the **Start** and **End** fields, select the *Style*, and click on the chosen *Variant*. Finally, click **OK** to apply the fill effect to the component which is being edited.

> **Note:** For projects on Windows Embedded target systems, the **Fill Effects** interface is available only for the **Rectangle** object.

> **Note:** Although **Fill Effects** is a useful tool for enhancing the look and feel of your screens, the operating system takes a longer time to fill an object with fill effects than with plain colors. You should develop criteria for using the feature without decreasing the performance of the system, especially for Windows Embedded target systems.

Using the Color animation, you can modify the color of a static object during runtime. When configuring this animation with Type = By Color, you can set the color that will be applied in the object during runtime, by the color code. The following table provides the code values as well as the RGB values for the most commonly used colors:

| Name | RGB Code | | | Code Value |
|---|---|---|---|---|
| | R (Red) | G (Green) | B (Blue) | |
| Black | 0 | 0 | 0 | 0 |
| Dark Red | 128 | 0 | 0 | 128 |
| Red | 255 | 0 | 0 | 255 |
| Pink | 255 | 0 | 255 | 16711935 |
| Rose | 255 | 153 | 204 | 13408767 |
| Brown | 153 | 51 | 0 | 13209 |
| Orange | 255 | 102 | 0 | 26367 |
| Light Orange | 255 | 153 | 0 | 39423 |
| Gold | 255 | 204 | 0 | 52479 |
| Tan | 255 | 204 | 153 | 10079487 |
| Olive Green | 51 | 51 | 0 | 13107 |
| Dark Yellow | 128 | 128 | 0 | 32896 |
| Lime | 153 | 204 | 0 | 52377 |
| Yellow | 255 | 255 | 0 | 65535 |
| Light Yellow | 255 | 255 | 153 | 10092543 |
| Dark Green | 0 | 51 | 0 | 13056 |
| Green | 0 | 128 | 0 | 32768 |
| Sea Green | 51 | 153 | 102 | 6723891 |
| Bright Green | 0 | 255 | 0 | 65280 |
| Light Green | 204 | 255 | 204 | 13434828 |
| Dark Teal | 0 | 51 | 102 | 7877376 |
| Teal | 0 | 128 | 128 | 8421376 |
| Aqua | 51 | 204 | 204 | 13421619 |
| Turquoise | 0 | 255 | 255 | 16776960 |
| Light Turquoise | 204 | 255 | 255 | 16777164 |
| Dark Blue | 0 | 0 | 128 | 8388608 |
| Blue | 0 | 0 | 255 | 16711680 |
| Light Blue | 51 | 102 | 255 | 16737843 |
| Sky Blue | 0 | 204 | 255 | 16737843 |
| Pale Blue | 153 | 204 | 255 | 16764057 |
| Indigo | 51 | 51 | 153 | 10040115 |

| Name | RGB Code | | | Code Value |
|---|---|---|---|---|
| | R (Red) | G (Green) | B (Blue) | |
| Blue-Gray | 102 | 102 | 153 | 10053222 |
| Violet | 128 | 0 | 128 | 8388736 |
| Plum | 153 | 51 | 102 | 6697881 |
| Lavender | 204 | 153 | 255 | 16751052 |
| Gray-80% | 51 | 51 | 51 | 3355443 |
| Gray-50% | 128 | 128 | 128 | 8421504 |
| Gray-40% | 150 | 150 | 150 | 9868950 |
| Gray-25% | 192 | 192 | 192 | 12632256 |
| White | 255 | 255 | 255 | 16777215 |

**Tip:** The `RGBColor` and `RGBComponent` functions can be used to manipulate color codes during runtime.

**Note:** The number of colors available when developing the project depends on the color settings configured on the operating system of the development station. The number of colors available when running the project depends on the color settings configured on the operating system of the runtime station.

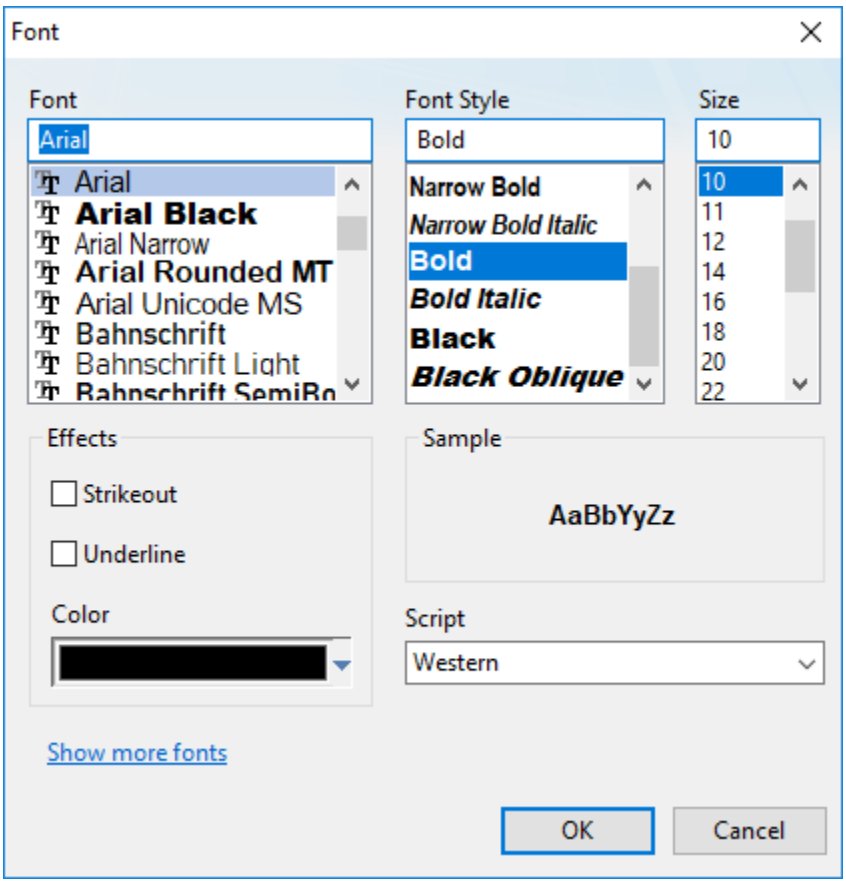
## *Fonts*

EcoStruxure Machine SCADA Expert supports any UNICODE font available on the operating system where EcoStruxure Machine SCADA Expert is running. Therefore, it is possible to configure interfaces using characters for languages that do not use the standard western characters, such as Japases, Chinese, Arabic, Cyrillic, etc.

The font used on the development environment (Worksheets, Dialogs, etc.) is the default font installed by the operating system and dependent on the language of the operating system. To select a different font for development, click **Font** on the View tab of the ribbon.

**Tip:** You can change the font style of several objects simultaneously by selecting them all (press the Shift key down as you click each one) and then using the **Fonts** tool on the Format tab of the ribbon.

When editing the objects that display text during runtime, you can set the font that will display the text by clicking on the Fonts button in the *Object Properties* window. The Font button launches the standard Fonts dialog:




You can set the font name, style, size, effects and script.

The icon displayed to the left of the font name indicates the font technology.

| Icon | Technology | Remarks | |
|---|---|---|---|
| | TrueType | Outline | TrueType and OpenType fonts are outline fonts that are rendered from line and curve commands. OpenType is an extension of TrueType. Both can be scaled and rotated. Both look good in all sizes and on all output devices supported by Windows.<br>Windows provides a selection of OpenType fonts, including Arial, Courier New, Lucida Console, Times New Roman, Symbol, and Wingdings.<br>Type 1, by Adobe Systems, Inc., is an outline font that is designed to work with PostScript printers. The outlines can be scaled and rotated. With OpenType technology, Windows fully supports Type 1 fonts. |
| | OpenType | | |
| N/A | Vector | | Vector fonts are supported because a number of programs still depend on them.<br>Vector fonts are rendered from a mathematical model. They are used primarily with plotters. Windows supports three vector fonts: Modern, Roman and Script. |
| N/A | Raster | | Raster fonts are supported because a number of programs still depend on them.<br>Raster fonts are stored in files as bitmap images and are composed of a series of dots whether they are displayed on the screen and on paper. |

It is strongly recommended that you use only TrueType or OpenType fonts. Fonts designed with other technologies (e.g., Courier) cannot be scaled properly and could cause issues during runtime.

> **Note:** When you design screens, the fonts you use are the ones available in the operating system of your development station. The fonts on the runtime station, however, may look different (e.g., different size in pixels), even if all settings are the same on both stations. This situation occurs more frequently when projects are run on Windows Embedded operating systems, where the fonts do not always match the proportions of the equivalent fonts on Windows PC operating systems. Therefore, it is important to test the graphic interfaces (screens) on the actual runtime platform during the development of the project. You should not wait until after the whole project has been developed, or it may become necessary to re-design the screens so the text objects display properly on the runtime platform.

## *ASCII Character Table*

| Character Set (0 - 127) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Code | Char | Code | Char | Code | Char | Code | Char |
| 0 | | 32 | [space] | 64 | @ | 96 | ` |
| 1 | | 33 | ! | 65 | A | 97 | a |
| 2 | | 34 | " | 66 | B | 98 | b |
| 3 | | 35 | # | 67 | C | 99 | c |
| 4 | | 36 | $ | 68 | D | 100 | d |
| 5 | | 37 | % | 69 | E | 101 | e |
| 6 | | 38 | & | 70 | F | 102 | f |
| 7 | | 39 | ' | 71 | G | 103 | g |
| 8 | ** | 40 | ( | 72 | H | 104 | h |
| 9 | ** | 41 | ) | 73 | I | 105 | i |
| 10 | ** | 42 | * | 74 | J | 106 | j |
| 11 | | 43 | + | 75 | K | 107 | k |
| 12 | | 44 | , | 76 | L | 108 | l |
| 13 | ** | 45 | - | 77 | M | 109 | m |
| 14 | | 46 | . | 78 | N | 110 | n |
| 15 | □ | 47 | / | 79 | O | 111 | o |
| 16 | □ | 48 | 0 | 80 | P | 112 | p |
| 17 | □ | 49 | 1 | 81 | Q | 113 | q |
| 18 | □ | 50 | 2 | 82 | R | 114 | r |
| 19 | | 51 | 3 | 83 | S | 115 | s |
| 20 | | 52 | 4 | 84 | T | 116 | t |
| 21 | | 53 | 5 | 85 | U | 117 | u |
| 22 | □ | 54 | 6 | 86 | V | 118 | v |
| 23 | □ | 55 | 7 | 87 | W | 119 | w |
| 24 | □ | 56 | 8 | 88 | X | 120 | x |
| 25 | □ | 57 | 9 | 89 | Y | 121 | y |
| 26 | □ | 58 | : | 90 | Z | 122 | z |
| 27 | | 59 | ; | 91 | [ | 123 | { |
| 28 | □ | 60 | < | 92 | \ | 124 | \| |
| 29 | □ | 61 | = | 93 | ] | 125 | } |
| 30 | - | 62 | > | 94 | ^ | 126 | ~ |
| 31 | | 63 | ? | 95 | _ | 127 | □ |

| Character Set (128 – 255) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Code | Char | Code | Char | Code | Char | Code | Char |
| 128 | € | 160 | [space] | 192 | À | 224 | à |
| 129 | □ | 161 | ¡ | 193 | Á | 225 | á |
| 130 | ‚ | 162 | ¢ | 194 | Â | 226 | â |
| 131 | ƒ | 163 | £ | 195 | Ã | 227 | ã |
| 132 | „ | 164 | ¤ | 196 | Ä | 228 | ä |
| 133 | … | 165 | ¥ | 197 | Å | 229 | å |
| 134 | † | 166 | ¦ | 198 | Æ | 230 | æ |
| 135 | ‡ | 167 | § | 199 | Ç | 231 | ç |
| 136 | ˆ | 168 | ¨ | 200 | È | 232 | è |
| 137 | ‰ | 169 | © | 201 | É | 233 | é |
| 138 | Š | 170 | ª | 202 | Ê | 234 | ê |
| 139 | ‹ | 171 | « | 203 | Ë | 235 | ë |
| 140 | Œ | 172 | ¬ | 204 | Ì | 236 | ì |
| 141 | □ | 173 | | 205 | Í | 237 | í |
| 142 | Ž | 174 | ® | 206 | Î | 238 | î |
| 143 | □ | 175 | ¯ | 207 | Ï | 239 | ï |
| 144 | □ | 176 | ° | 208 | Ð | 240 | ð |
| 145 | ' | 177 | ± | 209 | Ñ | 241 | ñ |
| 146 | ' | 178 | ² | 210 | Ò | 242 | ò |
| 147 | " | 179 | ³ | 211 | Ó | 243 | ó |
| 148 | " | 180 | ´ | 212 | Ô | 244 | ô |
| 149 | · | 181 | µ | 213 | Õ | 245 | õ |
| 150 | – | 182 | ¶ | 214 | Ö | 246 | ö |
| 151 | — | 183 | · | 215 | × | 247 | ÷ |
| 152 | ˜ | 184 | ¸ | 216 | Ø | 248 | ø |
| 153 | ™ | 185 | ¹ | 217 | Ù | 249 | ù |
| 154 | š | 186 | º | 218 | Ú | 250 | ú |
| 155 | › | 187 | » | 219 | Û | 251 | û |
| 156 | œ | 188 | ¼ | 220 | Ü | 252 | ü |
| 157 | □ | 189 | ½ | 221 | Ý | 253 | ý |
| 158 | ž | 190 | ¾ | 222 | Þ | 254 | þ |
| 159 | Ÿ | 191 | ¿ | 223 | ß | 255 | |

**Note:**

- Values 8, 9, 10, and 13 convert to backspace, tab, linefeed, and carriage return characters, respectively. They have no graphical representation, but depending on the project, they may affect the visual display of text.
- □ means it is not supported on the current platform.

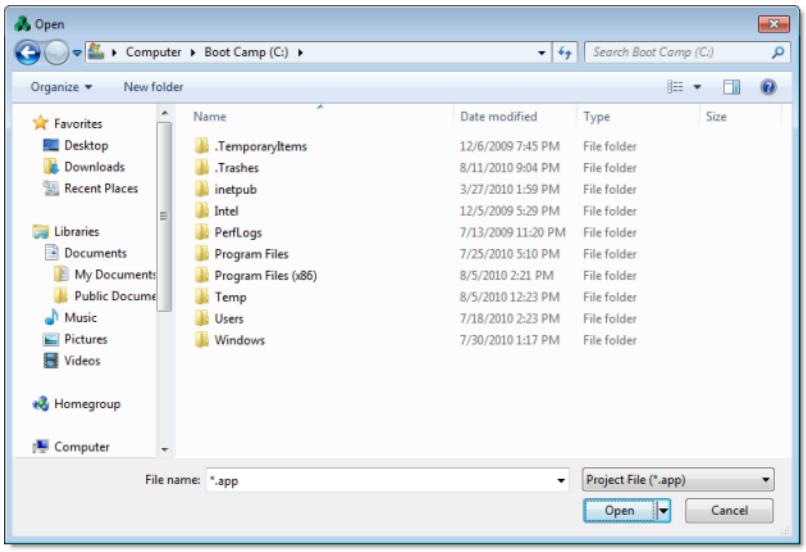
# Performing Common Tasks

## *Accessing Projects and Files*

These instructions assume you have already created a new project.

### Opening Projects

To open a project, go to **File** and then select **Open Project**. This displays the *Open* dialog, which lists all existing folders. You can use the *Open* dialog to locate and open a EcoStruxure Machine SCADA Expert project.




***Open dialog***

### Opening Files

To open a specific screen or worksheet file, go to **File** and then select **Open**. This displays the *Open* dialog (as shown in the preceding section), which lists all existing folders. To locate and open a screen or worksheet file from this dialog, click the **Files of type** combo-box button, and then click on a file name to select it from the list.

### Closing Projects

- From the *Standard* toolbar, choose **File** > **Save** to save any active screens or worksheets. The **Save** option becomes enabled (active) only after you modify the active file.

> **Note:** You can also use the **Save** button on the *Standard* toolbar or type Ctrl+S to save the open, active screen/worksheet.

- From the *Standard* toolbar, choose **File** > **Save As** to save active screens or worksheets, and to specify a (new) name and location for the file.
- Select the **Save As HTML** option to save the active display in HTML format.
- You can also click **Save All** on the File menu to save all open screens or worksheets. The **Save All** option becomes enabled (active) only after you modify the active file.

> **Note:** Using **File** > **Save All** is the same as using the **Save All** button on the *Standard* toolbar.

- Select the **File** > **Save All As HTML** option to save *all* project screens in HTML format. You have to close all documents before executing this command.

- Choose **File** > **Save Screen Group As HTML** to save the screen group in HTML format, making them available to the remote Thin Client through a Web browser.

### Closing Files

- On the File menu, click **Close** to close the active screen or worksheet. EcoStruxure Machine SCADA Expert prompts you to save all unsaved changes before it closes the screen/worksheet.

> **Note:** Using **File** > **Close** is the same as clicking the Close button on the title bar.

- You can also choose **File** > **Close All**. Selecting the **Close All** option closes all open screens or worksheets. EcoStruxure Machine SCADA Expert prompts you to save all unsaved changes before it closes the screens/worksheets.

## *Using Common Dialog Buttons*

The following table describes buttons that frequently appear in EcoStruxure Machine SCADA Expert dialogs and windows:

**Common Dialog Buttons**

| Button | Purpose |
|---|---|
| **OK** | Click this button to execute and save all changes, and close the dialog or window. |
| **Apply** | Click this button to execute and save all changes, but leave the dialog or window open. This button enables you to see the effects of your changes before closing the dialog/window. |
| **Cancel** | Click this button to close the dialog or window immediately (discarding any changes). |
| **Open** | Click this button to open a file. Generally, this button is associated with a combo-box or list pane. You use the combo-box or list pane to specify a file and then click the Open button to open the file. |
| **Close** | Click this button to close the open file, screen, dialog, and so forth. |
| **Browse** | Click this button to open a *Browse* dialog to search for a file or folder to open. |
| **Back** | Click this button to progress to the previous screen in a sequence of screens. |
| **Next** | Click this button to progress to the next screen in a sequence of screens. |
| **Replace** | Click to open a *Replace* dialog, which enables you to change tags or strings associated with a selected screen object. |
| **Remove** | Click to remove a selected (highlighted) object from a list or a screen. |

## *Convert your project's display resolution*

Use the **Convert Resolution** tool to convert your project's display resolution to a different size. All of the existing project screens will be resized proportionally.

Before you begin this task, you should manually back up your entire project. It is not absolutely necessary to do so, because the existing project screens will be automatically backed up before they are resized, but if you are not satisfied with the results of the conversion, it might be easier to start over from your backup than it would be to restore the screens.

You selected your project's display resolution when you created the project. The display resolution is used as the default size for new project screens. For more information, see Creating a new project on page 98.

You can also check your project's display resolution in the project settings. For more information, see Options tab on page 104.

> **Note:** If you want to change your project's display resolution but not resize the existing project screens, you should not use this feature. Instead, you should manually edit your project file in order to change the setting itself: close your project, exit Studio, open your project file (`<project name>.app`) in a text editor, and then edit the following property:
>
> ```
> [Info]
> ```

```
AppResolution=<width in pixels> <height in pixels>
```

When you convert your project's display resolution, all of the existing project screens will also be resized proportionally. For example, if you convert from 1280x800 to 640x480, all of the screens — not just those that are 1280x800 — will be resized to 50% of their original height and 60% of their original width.

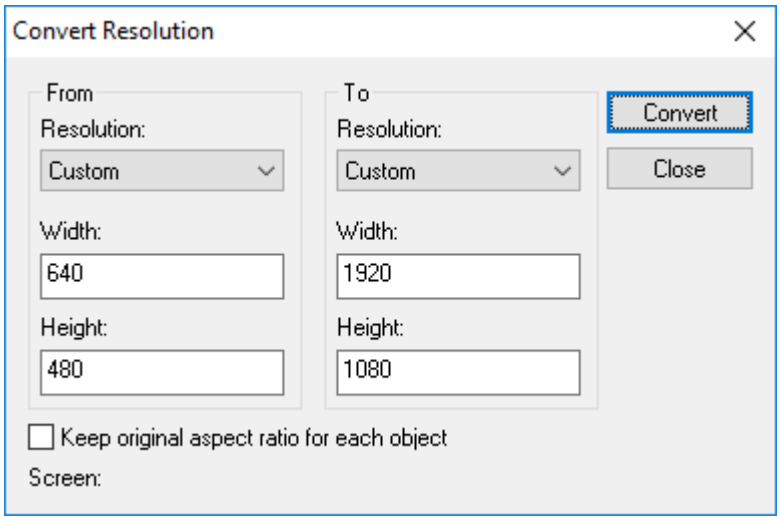
By default, the screen objects in those screens will also be resized and repositioned according to the same proportions, but that means the resized objects might be distorted somewhat. For example, squares might become rectangles and circles might become ellipses, depending on what the new proportions are. As such, you have the option to keep the original aspect ratio for each object; it will still be resized, but not necessarily by the same proportions as the entire screen.

The screen objects are repositioned according to their center points, rather than according to any of their corners. This can affect how the objects are distributed in the resized screen. For example, if you have several objects that are top-aligned (i.e., aligned so that their top edges all have the same Y position) but of different heights, they will be vertically distributed in the resized screen.

Keep in mind that you can group several screen objects together so that the entire group is resized/repositioned as a single object. Symbols are considered groups for this purpose.

To convert your project's display resolution and resize the existing project screens:

1. Close all open screens and worksheets.
2. On the **Home** tab of the ribbon, in the **Tools** group, click **Convert Resolution**.
   The *Convert Resolution* dialog box is displayed.




***Convert Resolution dialog box***

3. In the **From** area, specify the display resolution that you want to convert *from*: either select the resolution from the list, or type the width and height (in pixels).

   By default, the **From** resolution is your project's current display resolution, but you can specify another resolution if you want a different proportion for resizing screens.
4. In the **To** area, specify the display resolution that you want to convert *to*: either select the resolution from the list, or type the width and height (in pixels).

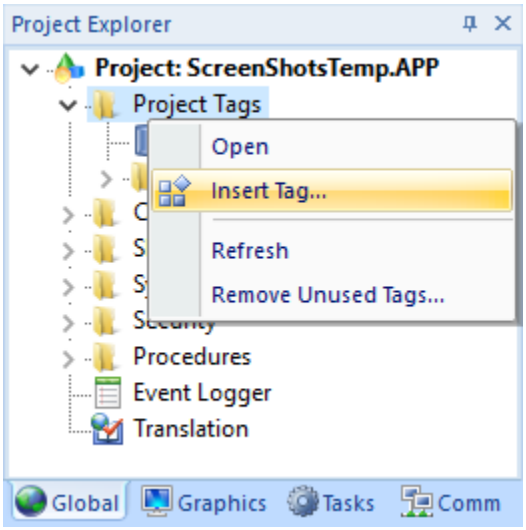
   By default, the **To** resolution is the same as the display resolution of the computer on which you are running Studio and editing your project.
5. If you do not want to resize the screen objects in the existing project screens, select **Keep original aspect ratio for each object**.
6. Click **Convert**.
   The display resolution is converted and the project screens are resized. The progress is displayed at the bottom of the dialog box.
7. Click **Close** to exit the dialog box.

Be aware that using this feature might disrupt the layout of some screens but not others, depending on how those screens are designed, so you should review all of the resized screens when you are done.

The existing project screens were automatically backed up before they were resized. You can find the backups in your project folder at `<project name>\Screen\Backup`. To restore them, exit Studio and then copy the backups to `<project name>\Screen`, so that they replace the resized screens.

### Using Shortcut Menus

If you right-click on any component in the Project Explorer, a menu displays with options related to that component. For example, the following shortcut menu enables you to **Open** the Project Tags database, **Insert** (create) a new tag, or **Refresh** the current view of the database:




*Right-Click to Open a Shortcut Menu*

### Using Select All

To select all objects on the active screen, or press **CTRL+A**.

### Cutting, Copying, Pasting Objects

To delete a selected item from a screen and store it on the Windows clipboard (replacing any previously selected objects stored on the clipboard), either click **Cut** on the Home tab of the ribbon or press **CTRL+X**.

To copy a selected item without deleting it and store it on the Windows clipboard, either click **Copy** on the Home tab of the ribbon or press **CTRL+C**.

To paste the contents of the Windows clipboard (cut or copied objects) onto the active screen, either click **Paste** on the Home tab of the ribbon or press **CTRL+V**. You can paste a cut or copied object multiple times.

To undo the last action performed (and up to 20 actions taken *prior* to the last action), either click the **Undo** tool on the Quick Access Toolbar or press **CTRL+Z**.

### Find text in the current document or entire project

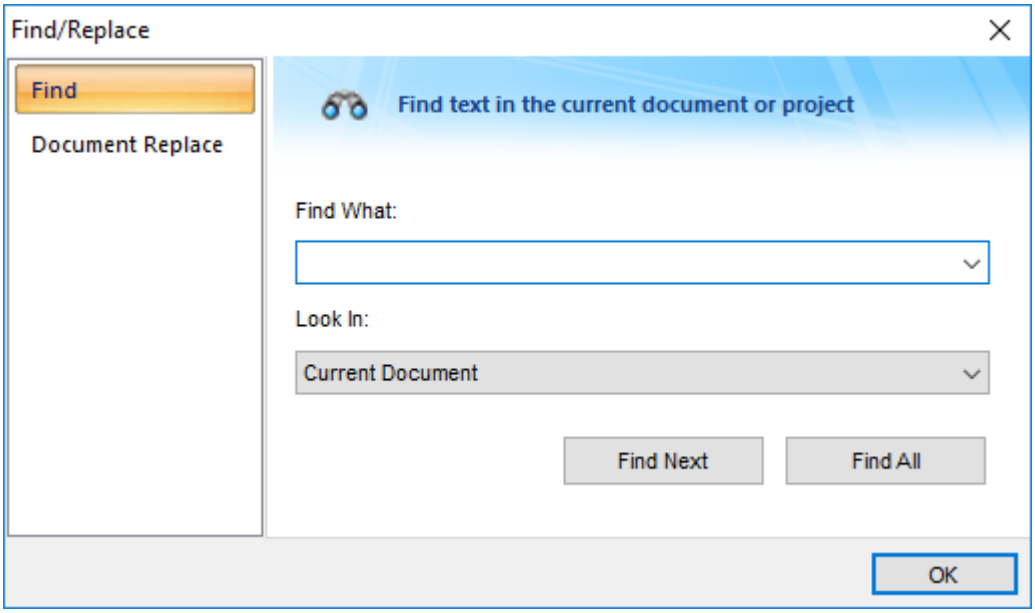
Use the **Find & Replace** command to find some specified text either in the current document or in your entire project.

For this task, a document is any one of the several screens and worksheets that make up your project. If you want to search the tags database, use the filter tools in the Project Tags Datasheet View. For more information, see Sort or filter the rows in a worksheet on page 191.

When you create a new project, a search database is also created for it. This database contains an index of every document in your project — that is, every project screen, script worksheet, task worksheet, tags datasheet, and so on — and it allows you to quickly find all occurrences of the specified text. Each time you edit and save a document, the search database is also updated.

To find text in documents:

1. If you want to find text in a single document, make sure that document is open in the Screen/Worksheet Editor. If more than one document is open, click the tab of the document you want in order to bring it to the front.
   It is now the "current document".
2. On the **Home** tab of the ribbon, in the **Clipboard** group, click **Find & Replace**.
   The *Find/Replace* dialog box is displayed.
3. In the *Find/Replace* dialog box, click the **Find** tab.
   The **Find** tab of the dialog box is displayed.




***The Find tab of the Find/Replace dialog box***

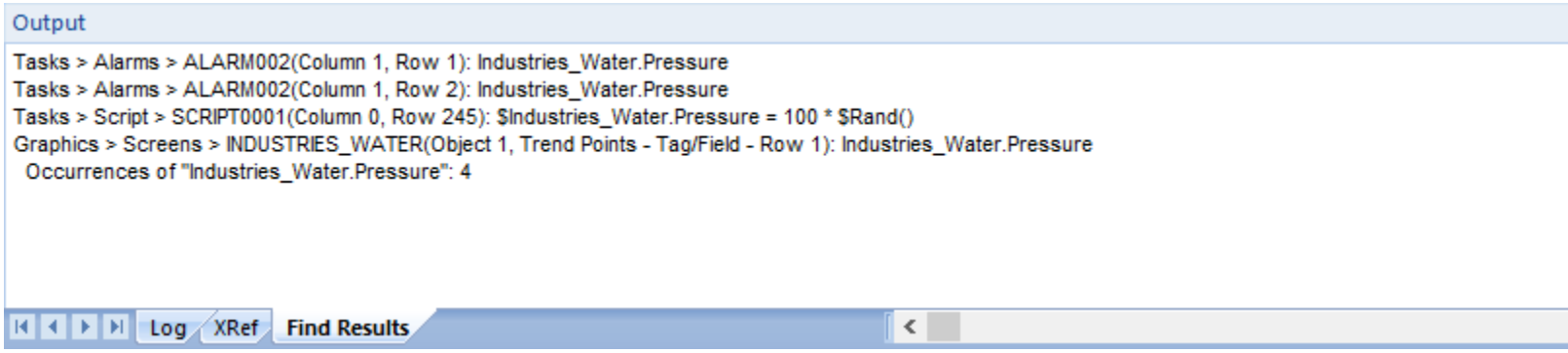
Whenever you use the **Find & Replace** command, it synchronizes the search database with the current state of the project. That synchronization is shown as a progress bar in the dialog box, immediately after the dialog box is displayed. The synchronization should be finished quickly, but in some cases — for example, if you are using the command for the first time in a large, existing project — it will take more time. You can proceed with finding text before the synchronization is finished, but if you do, the results might be inaccurate or incomplete.

4. In the **Find What** box, type or select the text that you want to find.
   Previous texts are saved in the list. To see the list, click the down arrow on the right.
5. In the **Look In** box, select either **Current Document** or **Entire Project**.
   If there is no current document (as determined in Step 1), **Entire Project** will be selected by default.
6. To find the first occurence of the specified text in the current document, click **Find Next**.
   (This works only for script and task worksheets, not for project screens.)

   If the text is found, it is highlighted in the document. Click **Find Next** again to find the next occurence, if there is one. When it reaches the end of the document, it starts over from the beginning.

   If the text is not found, a message is displayed to notify you of that.
7. To find all occurences of the specified text, either in the current document or in your entire project, click **Find All**.

If the text is found, the results are displayed in the **Find Results** tab of the *Output* window, in the development environment.




***Example of results for Find All***

Each line of the results represents a separate occurrence of the specified text, and it comprises the following information:

- The location of the document, in the Project Explorer, that contains the specified text;
- The location of the specified text in that document (e.g., the screen object and object property, the column and row in a worksheet); and
- The specified text itself, given in context (e.g., how the specified text is used as a variable in a line of VBScript).

> **Note:** If the specified text is found in a document that is password-protected, only the location of the document is displayed in the results. The other information is not available. The specified text can be found even in a password-protected document because the document was indexed when it was saved.

8. To go to an occurrence of the specified text, do one of the following:
   - Double-click that line in the results; or
   - Right-click that line in the results, and then click **Open file** on the shortcut menu.

   The corresponding document is opened in the Screen/Worksheet Editor, in the development environment, and the occurrence is highlighted.

## Replace text in the current document

Use the **Find & Replace** command to find and replace some specified text in the current document.

For this task, a document is any one of the several screens and worksheets that make up your project. If you want to search the tags database, use the filter tools in the Project Tags Datasheet View. For more information, see Sort or filter the rows in a worksheet on page 191.

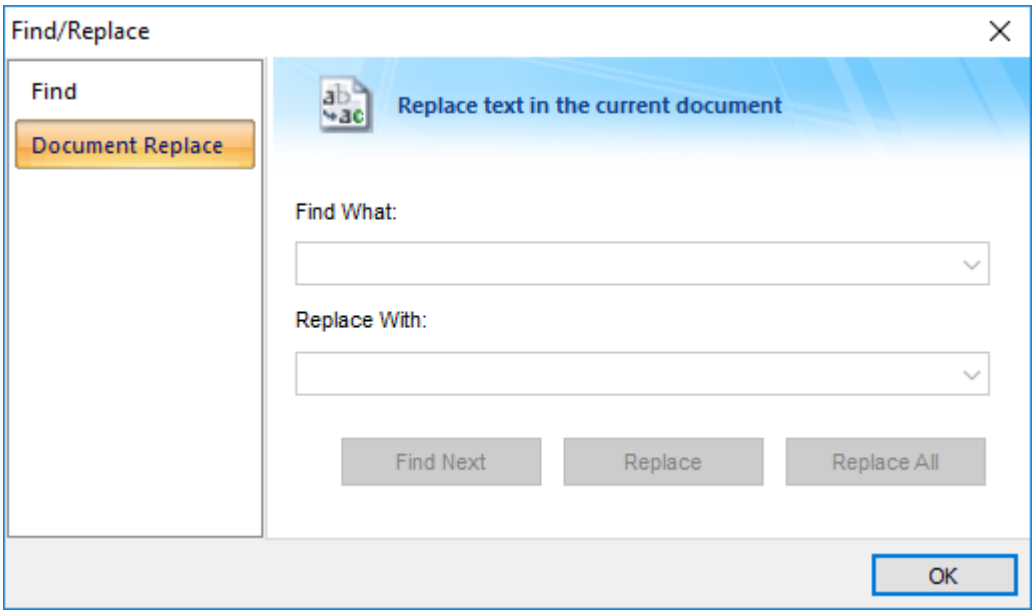
There is no option to replace all occurences of the specified text in your entire project, because it cannot be undone.

When you create a new project, a search database is also created for it. This database contains an index of every document in your project — that is, every project screen, script worksheet, task worksheet, tags datasheet, and so on — and it allows you to quickly find all occurrences of the specified text. Each time you edit and save a document, the search database is also updated.

To replace text in the current document:

1. Make sure the document that you want to search is open in the Screen/Worksheet Editor. If more than one document is open, click the tab of the document you want in order to bring it to the front. It is now the "current document".
2. On the **Home** tab of the ribbon, in the **Clipboard** group, click **Find & Replace**. The *Find/Replace* dialog box is displayed.

3. In the *Find/Replace* dialog box, click the **Document Replace** tab.
   The **Document Replace** tab of the dialog box is displayed.




***The Document Replace tab of the Find/Replace dialog box***

   Whenever you use the **Find & Replace** command, it synchronizes the search database with the current state of the project. That synchronization is shown as a progress bar in the dialog box, immediately after the dialog box is displayed. The synchronization should be finished quickly, but in some cases — for example, if you are using the command for the first time in a large, existing project — it will take more time. You can proceed with finding text before the synchronization is finished, but if you do, the results might be inaccurate or incomplete.

4. In the **Find What** box, type or select the text that you want to find.
   Previous texts are saved in the list. To see the list, click the down arrow on the right.
5. In the **Replace With** box, type or select the text that will replace the found text.
   Previous texts are saved in the list. To see the list, click the down arrow on the right.
6. To find and replace the specified text one occurence at a time, do the following.
   (This works only for script and task worksheets, not for project screens.)
   a) Click **Find Next**.
      If the text is found, it is highlighted in the document. If the text is not found, a message is displayed to notify you of that.
   b) To replace the highlighted text, click **Replace**.
      The text is replaced, and then the next occurence (if any) is highlighted.
   c) To skip the highlighted text and find the next, click **Find Next**.

   When it reaches the end of the document, it starts over from the beginning.
7. To find and replace all occurences of the specified text, click **Replace All**.
   If the text is found, it is replaced and then the results are displayed in the **Find Results** tab of the *Output* window, in the development environment. If the text is not found, a message is displayed to notify you of that.

To save your changes, you still need to save and close the document as you normally would.

## *Finding System Information*

To open the *System Information* window, click **System** on the Help tab of the ribbon. The *System Information* window provides the following:

- **System Information**: Displays details about your operating system.
- **Network Resources**: Displays details about your computer's network.
- **Applications**: Lists the applications that are running.
- **Processes**: Displays all Windows tasks that are running.
- **Services**: Lists the Windows NT/2000 services being used by EcoStruxure Machine SCADA Expert (Windows NT/2000 only).

***System Information window***

> **Note:** Although you open the *System Information* window from within the development application, this window only provides general information about the local station and the network. It does not provide specific information about the project.

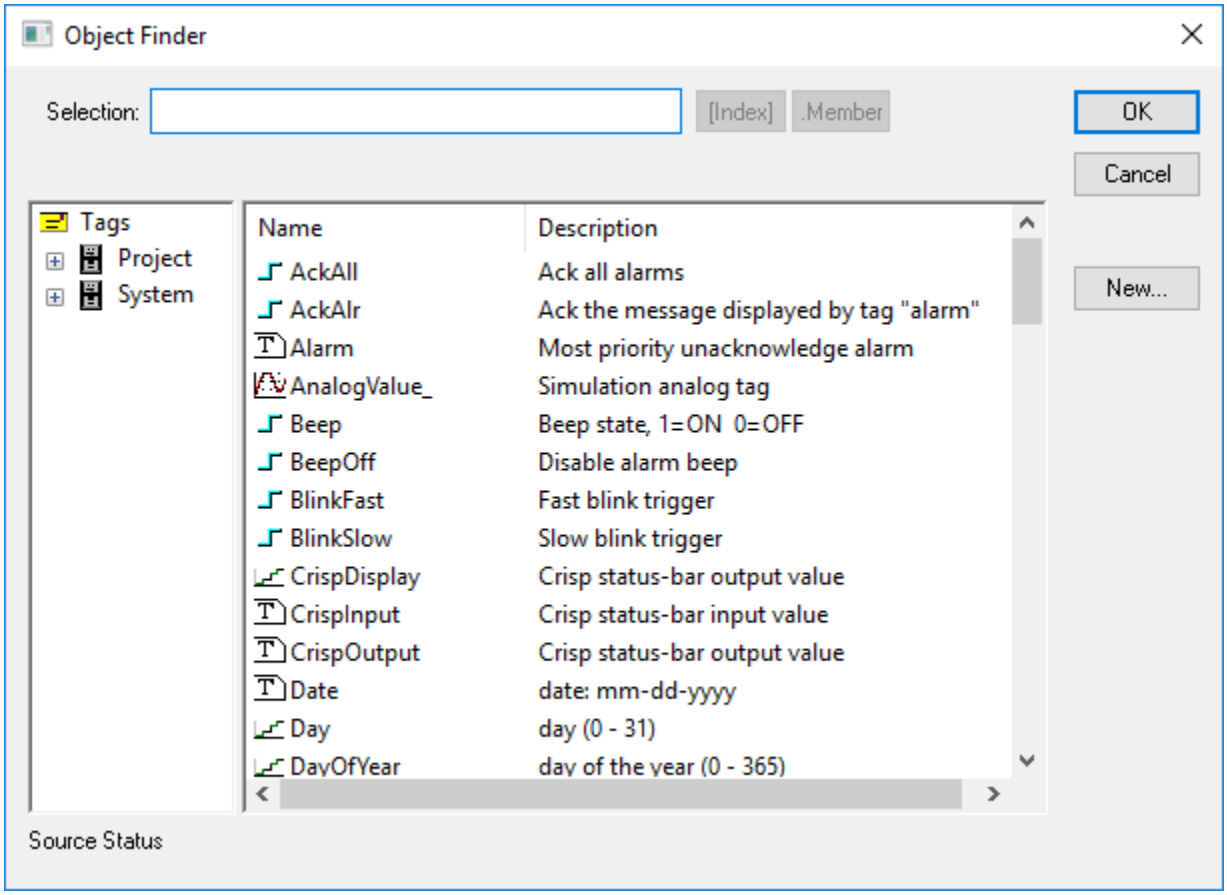
## Using the Tag Properties Toolbar

The *Tag Properties* toolbar provides a text box and several buttons (shortcuts) that enable you to create, locate, and access different tags, functions, and tag properties.

***Tag Properties Toolbar***

### Using the Object Finder

Click the **Object Finder** button to open the *Object Finder* dialog, which lists all Tags and Functions currently configured for the project.



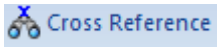

***Object Finder Dialog***

- To select an existing tag/function, double-click on the tag/function name, and then click **OK** to close the dialog. The selected name displays in the **Tagname** text box.
- To select a specific array index, click the **Index** button after specifying the array tag name.
- To select a specific member name, click the **Member** button after specifying the class tag name.
- To create a new tag, click the **New** button.

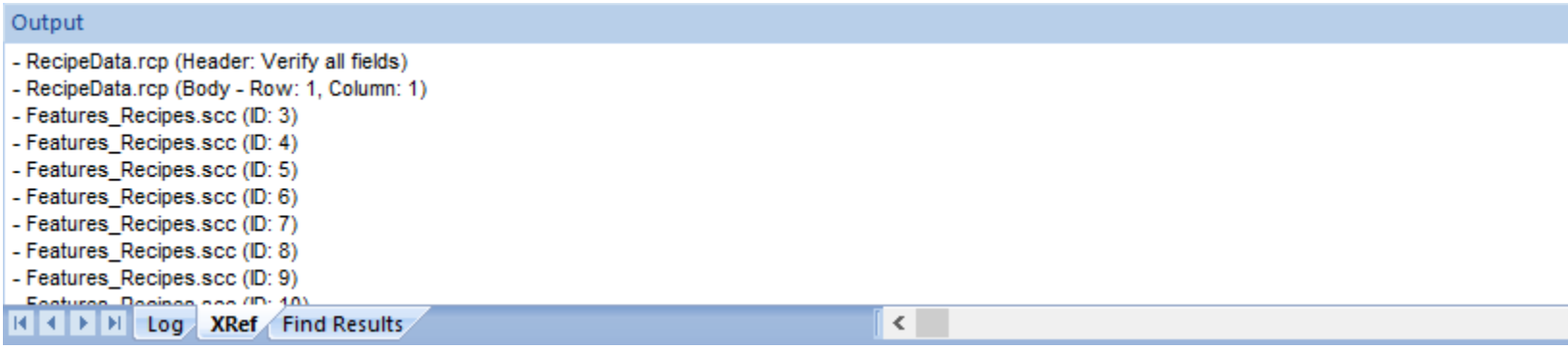
  When the *New Tag* dialog displays, enter the following information, then click **OK** to close the dialog:

  - **Name**
  - **Array Size**
  - **Type** (Boolean, Integer, Real, String, Class:Control, Class:msgonline, or Class:Alr)
  - **Description**
  - **Scope** (local or server)

### Using the X-ref Option

Click the **Cross Reference** button 

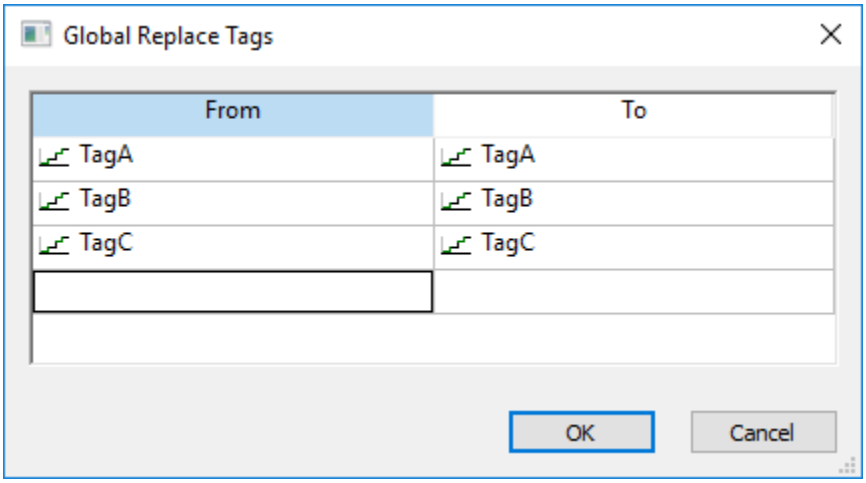
to search all project screens and worksheets for the tag noted in the **Tagname** text box. This function writes a log, detailing all the occurrences of the tag, to the **XRef** tab in the *Output* window. For example, the results of searching for a *BlinkFast* tag are as follows:




***XRef Results***

### Using the Global Tags Replace Option

When you select the **Global Tags Replace** button from the *Tag Properties* toolbar, the *Global Replace* dialog displays:




***Global Replace Dialog***

From the *Global Replace* dialog, you can replace any tag(s) from all documents (screens and worksheets) of the whole project. You can edit both the **From** and the **To** column.

When replacing composed tags (**`array size > 0`** and/or **`Type = Class`**), you can configure a specific array position (for example, **`TagA[1]`**) or class member (for example, **`TagB.MemberX`**) or both (for example, **`TagC[3].MemberY`**). If you configure only the *Main Tag Name* (for example, **`TagC`**) in the **From** column, all tags from this main tag will be modified for the tag configured in the **To** column.

If an invalid replacement is configured (for example, replace the *Main Tag* tag from a class type tag for a simple tag (not a class tag), the **OK** button will be disabled. When the **OK** button is pressed, the tags configured on the *Global Replace* dialog will be replaced in the order that they were configured on the dialog interface.

> **Note:** You must close all documents (screens and worksheets) before executing this command.

When changing the tag name on the *Tags Database* worksheet, EcoStruxure Machine SCADA Expert will ask you if you intend to replace this tag through the whole project.

The **Replace** option will be created in the **Edit** menu. By using this option, the *Global Replace* dialog is prompted, however, the changes are applied only to the current screen or worksheet in focus.

### *Replacing project tags in a document or screen object*

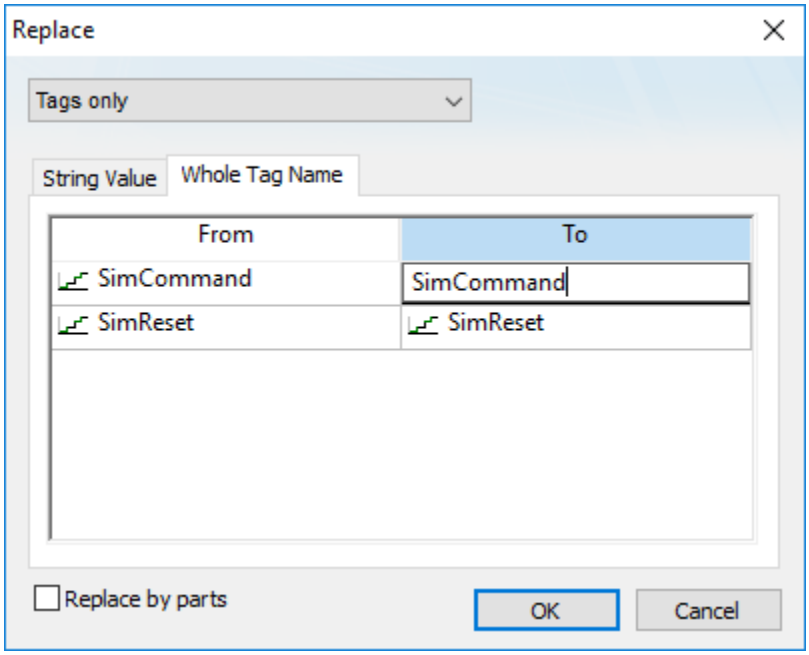
To replace all occurences of a tag in the current document, do one of the following:

- On the Home tab of the ribbon, in the Tags group, click **Replace**; or
- On the Graphics tab of the ribbon, in the Editing group, click **Replace**.

To replace all occurences of a tag in a screen object, double-click the object to open its *Object Properties* dialog and then click **Replace**.

All of these methods will open the *Replace* dialog, which is descibed below.

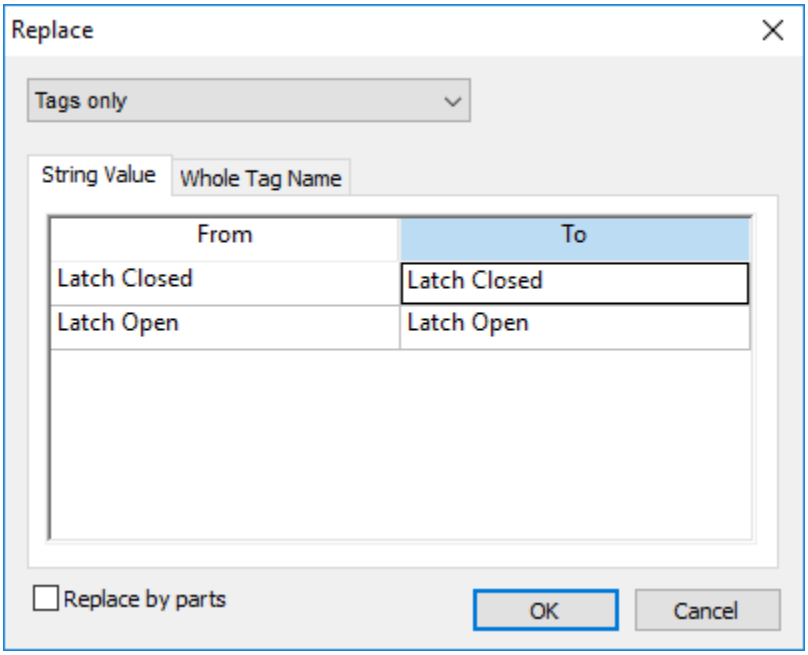
You can replace one or more tags by clicking the **Whole Tag Name** tab. Current tags used are displayed. The original tag names are shown in the **From** column on the left, and you can enter your new tag names in the **To** column on the right.




***Whole Tag Name tab***

Note that this does not *rename* or *delete* any tag — it only replaces the tags used in the object with other tags from the database.

You can also replace one or more strings (e.g., button captions, descriptive text) by clicking the **String Value** tab.




***String Value tab***

When you are done, click **OK**.

## *Testing Displays*

From the menu bar, select **Project** > **Test Display** to activate the test display mode, which allows you to configure the application while viewing graphical animations online in the development environment.

The *Test Display* mode does not enable you to use the Command or Text Data Link animations nor execute worksheets.

> **Note:** Using the **Test Display** menu option is the same as using the **Test Display** button on the *Execution Control* toolbar.

To stop the *Test Display* mode, select **Project** > **Stop display test**.

> **Note:** Using the **Stop display test** menu option is the same as using the **Stop display test** button on the *Execution Control* toolbar.

## *Verify the project*

Verify your project in order to perform various housekeeping tasks on the project database.

More specifically, verifying your project validates all screens and worksheets, recompiles expressions and scripts, checks for missing tags and broken tag references, updates screens that have been saved as HTML, and trashes unnecessary files.

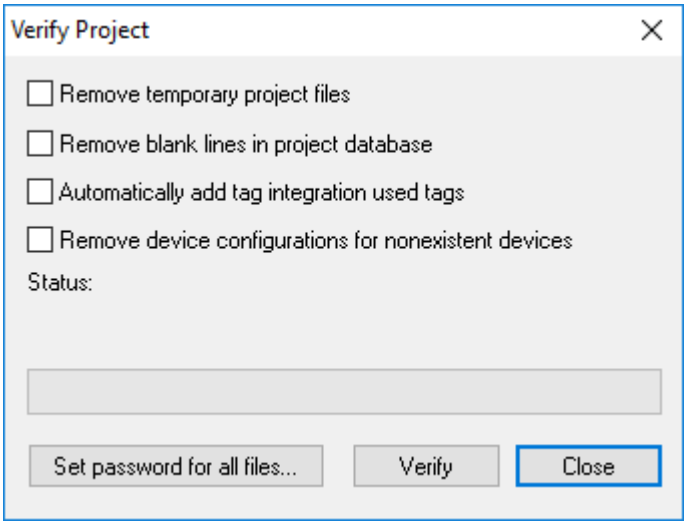
To avoid doing these tasks during project run time, which can decrease runtime performance and possibly even cause the project to freeze, we recommend that you verify your project whenever you have done one of the following:

- Update or upgrade a project from a previous version of Studio;
- Make major changes to the project, such as adding or removing several screens or worksheets; or
- Prepare to download the project to a target station.

> **Note:** Verifying a project does not also download it to the target station. To download the project, you must use the **Remote Management** commands.

To verify your project:

1. On the **Home** tab of the ribbon, in the **Tools** group, click **Verify**.
   The *Verify Project* dialog box is displayed.




***Verifying a project***

2. To trash any temporary files that were created during project development — for example, .txt, .mac, and .tag files — and decrease the size of your project folder, select **Remove temporary project files**.
3. To remove all blank lines (rows) that were manually inserted into worksheets, select **Remove blank lines in project database**.

   In previous versions of Studio, some project developers organized the contents of their worksheets by grouping related lines together and then inserting blank lines between the groups. It is not useful to do that anymore, however, since you can now sort and filter the rows of a worksheet.
4. To re-import integrated tags that are used screens and worksheets but have not been added to the Shared Database folder, select **Automatically add tag integration used tags**.

   This is sometimes necessary when you have copied screens from another project, or if refreshing the tag integration sources has broken some tag references. Make sure the prefixes on the used tags match the names of their respective sources.

   > **Note:** This option might not work if the used tags are multi-dimensional arrays or other complex data structures that had to be renamed to comply with Studio's tag syntax. If that is the case, you should manually add the tags.
5. To remove tag integration sources that can no longer be found, select **Remove device configurations for nonexistent devices**.

   Similar to **Automatically add tag integration used tags** above, this is sometimes necessary if refreshing the tag integration source(s) has caused a source to become lost.
6. To implement password protection on all of your project files at the same time, click **Set password for all files** and then type your desired password.

   For more information, see Password-protecting screens, symbols, and worksheets on page 679.
7. Click **Verify**.

The project is verified and the results are displayed in the *Output* window, in the development environment. If a tag is used in a screen or worksheet but is not defined in the tags database, that will be included in the verification results.

### Running Projects

To run your project -- specifically, to start the runtime modules specified as *Automatic* in the *Runtime Tasks* dialog -- click **Run** on the Home tab of the ribbon.

- When you start the *Viewer* module, it opens the screen(s) currently being edited.
- If you do not specify any *Automatic*tasks, EcoStruxure Machine SCADA Expert will launch the *Viewer* and *BGTask* tasks automatically when you click **Run**.
- If you are not currently editing screens in the development environment, the *Viewer* module opens the screen specified in the **Startup** screen field on the **Runtime Desktop** tab (*Project Settings* dialog).

To stop your project -- specifically, to stop all runtime modules -- click **Stop** on the Home tab of the ribbon.

> **Note:** Be sure you know which target system (local or remote) is configured before you run your project.

### Restoring Defaults

To restore the development environment to its default layout (after resizing or moving windows), click **Restore Default** on the View tab of the ribbon. You will need to close and reopen the development application for the changes to stick.

### Saving Your Work

Click the **Save** button to save any active screens or worksheets.

> **Note:**
>
> - Using the **Save** button is the same as selecting **File** > **Save** from the menu bar or typing the Ctrl+S key combination.
> - The **Save** function becomes available only when you modify the active file.

Click the **Save All** button to save all open screens or worksheets.

> **Note:**
>
> - Using the **Save All** button is the same as selecting **File** > **Save All** from the menu bar.
> - The **Save All** function becomes available only when you modify a screen or worksheet.

### Printing Screens and Worksheets

The screens and worksheets that make up your project are essentially documents like those you have worked with in other office applications, and they can be printed in the same way.

This task assumes that you have set up at least one printer for your computer, and that you know how to use the standard *Print* dialog in Windows.

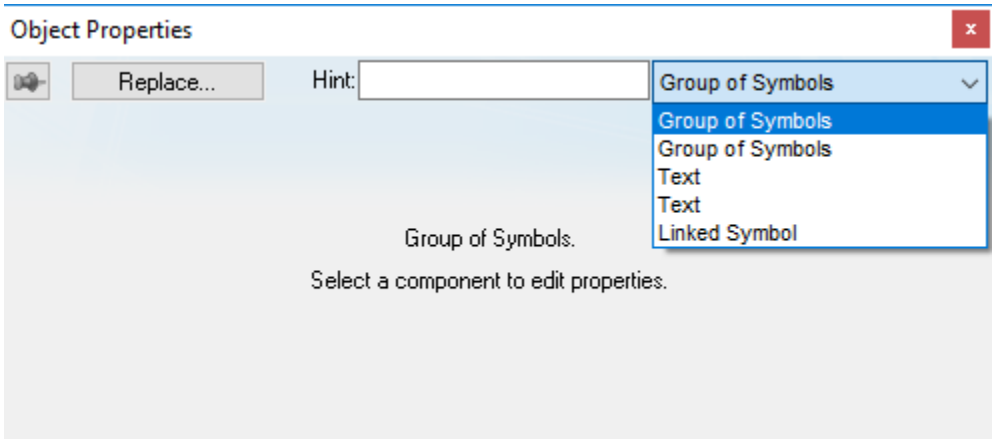
To print a screen or worksheet:

1. Open the screen or worksheet for editing, and then make sure it is the active tab in the Screen/Worksheet Editor.
2. Go to **File**, and the select **Print**.
3. Use the *Print* dialog to configure the settings and then print the document.

### Focusing the Object Properties Window

When you double-click any object (or group of objects) in the Screen Editor, the *Object Properties* window is launched, allowing you to configure the selected object's settings. The content of this dialog window varies

according to the specific object/animation that is being edited. However, there is always a pin button in the left upper corner of this dialog window:




***Object Properties Dialog***

The pin button looks like this, , when it is released, and like this, , when it is pressed.

When the pin button is released, the focus is passed to the object on the screen as soon as that object is selected. Therefore, we recommend you keep this button released when you want to manipulate (copy, paste, cut or delete) the objects. Although the *Object Properties* window is on the top, the keyboard commands (Ctrl +C, Ctrl+V, Ctrl+X or Del) are sent directly to the objects.

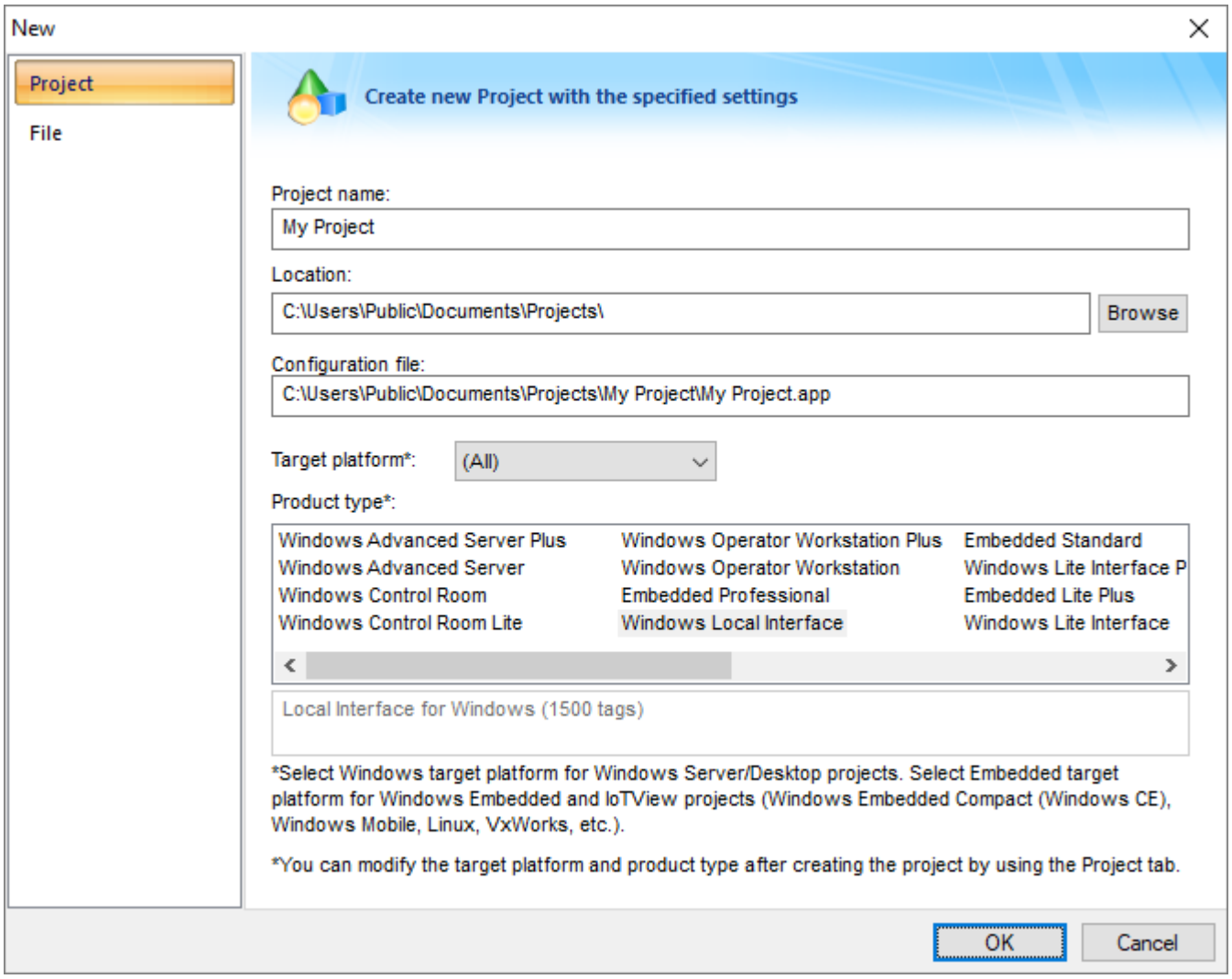
When the pin button is pressed the focus is kept on the *Object Properties* window, even when you click the objects on the screen. We recommend you keep this button pressed when you want to modify the settings of the objects. You can click an object and type the new property value directly in the *Object Properties* window (it is not necessary to click on the window to bring focus to it). Also, when the pin button is pressed, the *Object Properties* window does not automatically close when you click on the screen.

# Creating a New Project

## Creating a new project

This task describes how to create a new EcoStruxure Machine SCADA Expert project, including how to select the product type and default screen resolution.

1. Go to **File**, and then select **New**.
   The *New* dialog is displayed.
2. Click the **Project** tab if it is not already selected.




***Project tab of the New dialog***

3. In the **Project name** box, type the name of your project.

   Keep the following guidelines in mind:

   - You must follow the usual Windows naming conventions, particularly regarding the use of special characters; and
   - Do not use spaces in the name if you want to access your project from a Thin Client, because URLs cannot include spaces.

4. Click **Browse** to the right of the **Location** box, and then navigate to the folder where you want to save your project.

> **Tip:** By default, projects are saved in your Documents folder at `C:\Users\<user name>\Documents\EcoStruxure Machine SCADA Expert 2020 Projects\`. To change this default location for future projects, edit the **Default project path** setting in the project settings. For more information, see Preferences tab.

5. In the **Target platform** list, select the platform of the computer or embedded device that will host your project runtime.

   Selecting a platform will filter the list of available product types to show only those types that work on the selected platform. It is not absolutely necessary to select a target platform — if you leave the selection as **(All)**, then all product types are shown — but it helps you to make the decision.

6. In the **Product type** list, select the product type for your project.

   The product type determines the number of tags your project will support, among other things:

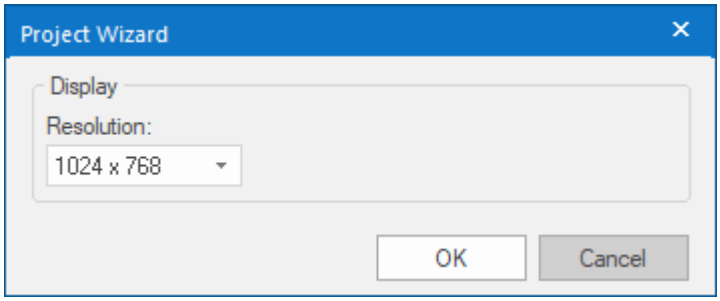
   **Target systems for EcoStruxure Machine SCADA Expert projects**

| Target Platform (Runtime Editions) | Product Type | Number of Tags |
|---|---|---|
| **Windows** (SCADA) | EcoStruxure Machine SCADA Expert Supervision | 64,000 |
| | EcoStruxure Machine SCADA Expert Line Management Plus | 32,000 |
| | EcoStruxure Machine SCADA Expert Line Management | 4,000 |
| | EcoStruxure Machine SCADA Expert Machine Control | 1,500 |

   For more information, see About target platforms, product types, and target systems on page 100.

> **Tip:** You can change the product type later, after you have created the project, by using the **Target System** command on the **Project** tab of the ribbon.

7. Click **OK**.
   The *New* dialog is closed, and the *Project Wizard* dialog is displayed.




***Project Wizard dialog***

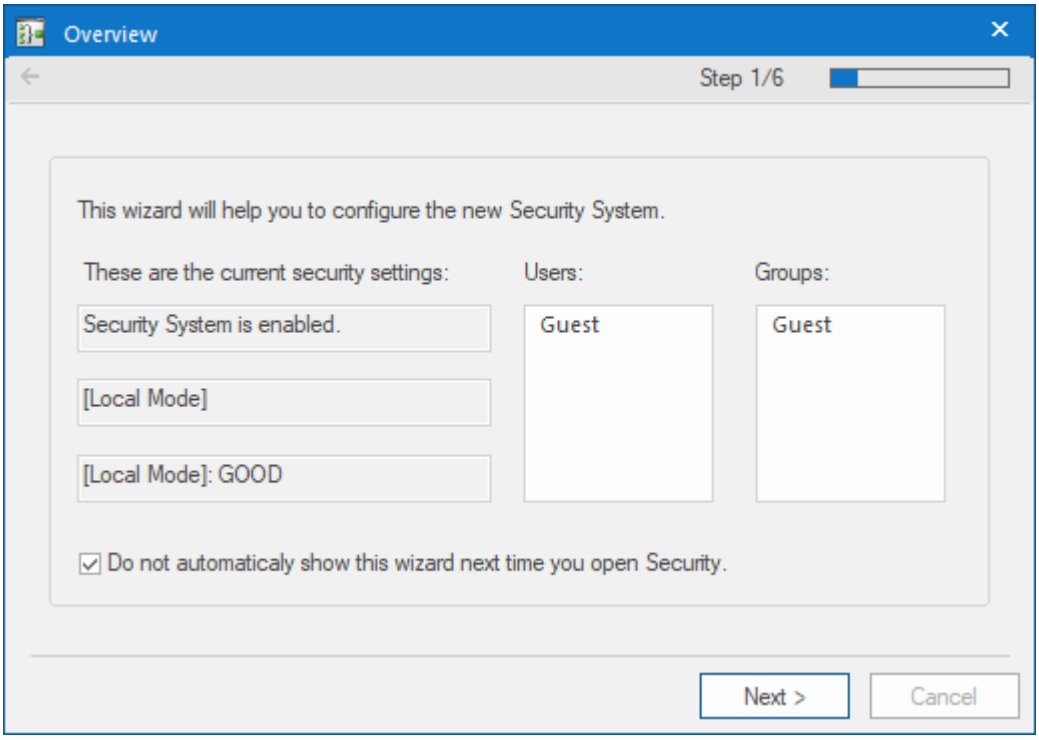
8. In the **Resolution** list, select the default resolution for your project screens. If you select **Custom**, then also specify the width and height (in pixels).

   In most cases, the default resolution should match the display size of the computer or device that will host the project runtime. Alternatively, if you plan to run your project on a "headless" station (i.e., a computer or device without an attached display) and then use thin clients to access the project screens, the default resolution should match the display size of those clients.

> **Tip:** You can change the resolution of individual project screens later, after you have created the project, by editing the screen attributes. You can also change the resolution of all project screens — effectively selecting a new default resolution — by using the **Convert Resolution** command on the **Home** tab of the ribbon.

9. Click **OK**.

The *Project Wizard* dialog is closed, the project is created in the development environment, and the *Security System Configuration Wizard* is displayed.




***Security System Configuration Wizard***

10. Use the *Security System Configuration Wizard* to set a Main Password for your project.

    The security system is enabled by default for all new projects. You can disable it later, but we recommend that you do not; even if you do not create any users or groups beyond the default Guest user, simply having it enabled and a Main Password set will prevent other people from making their own changes to it. For more information, see Using the Security System Configuration Wizard on page 643.

When you finish the *Security System Configuration Wizard*, your new project is ready for development.

## About target platforms, product types, and target systems

A project's target platform, product type, and target system determine important things about the project, such as how many tags the project will support and which features can be used during run time.

### Target platform

The target platform is the computer and operating system on which the project will run. It is generally either "Windows" or "Embedded":

**Windows**

A computer that runs one of the following operating systems:

- Windows:
  - Windows 10, version 1803 or later (including LTSC/LTSB versions)
  - Windows 8.1
- Windows Server:
  - Windows Server 2019
  - Windows Server 2016

- Windows Server 2012 R2

The computer must have the full EcoStruxure Machine SCADA Expert 2020 software installed, even if it will not be used to develop projects, because the full software includes the project runtime edition for Windows. The computer must also have an appropriate runtime license key.

All project features are available when **Windows** is selected as the target platform.

### Product type

The product type determines how many tags you can use in the project (including tags shared or imported from other systems). Given the available system resources, Windows computers can typically support far more tags than embedded devices.

The computer or device that will host your project runtime must have a license key that matches or exceeds the selected product type. To verify the license key, run Protection Manager (**Start** > **EcoStruxure Machine SCADA Expert 2020** > **EcoStruxure Machine SCADA Expert 2020 Register**) on that computer or device. Although you can change both the product type and license key later, we recommend that you verify the license key and then select the correct product type now so that you do not waste time developing a project that uses more tags than you are licensed for. For more information, see Licensing on page 44.

### Target system

The target platform and product type together determine the target system. You select the target system when you create a new project, and you can also change it later by using the **Target System** command (on the **Project** tab of the ribbon).

#### Target systems for EcoStruxure Machine SCADA Expert projects

| Target Platform (Runtime Editions) | Product Type | Number of Tags |
|---|---|---|
| **Windows** (SCADA) | EcoStruxure Machine SCADA Expert Supervision | 64,000 |
| | EcoStruxure Machine SCADA Expert Line Management Plus | 32,000 |
| | EcoStruxure Machine SCADA Expert Line Management | 4,000 |
| | EcoStruxure Machine SCADA Expert Machine Control | 1,500 |

Keep in mind that selecting a target system in the project development environment only serves as a guide during development. It automatically limits the number of project tags and hides unsupported features, so that you do not inadvertently develop a project you will not be able to run. The real limits are determined by the runtime license key that is installed on each computer or device.

Also, if you plan to run your project on multiple computers and devices with different license keys, we recommend that you develop for the lowest common target system.

## *Changing the target system of an existing project*

Use the **Target System** command to change the target system of an existing project.

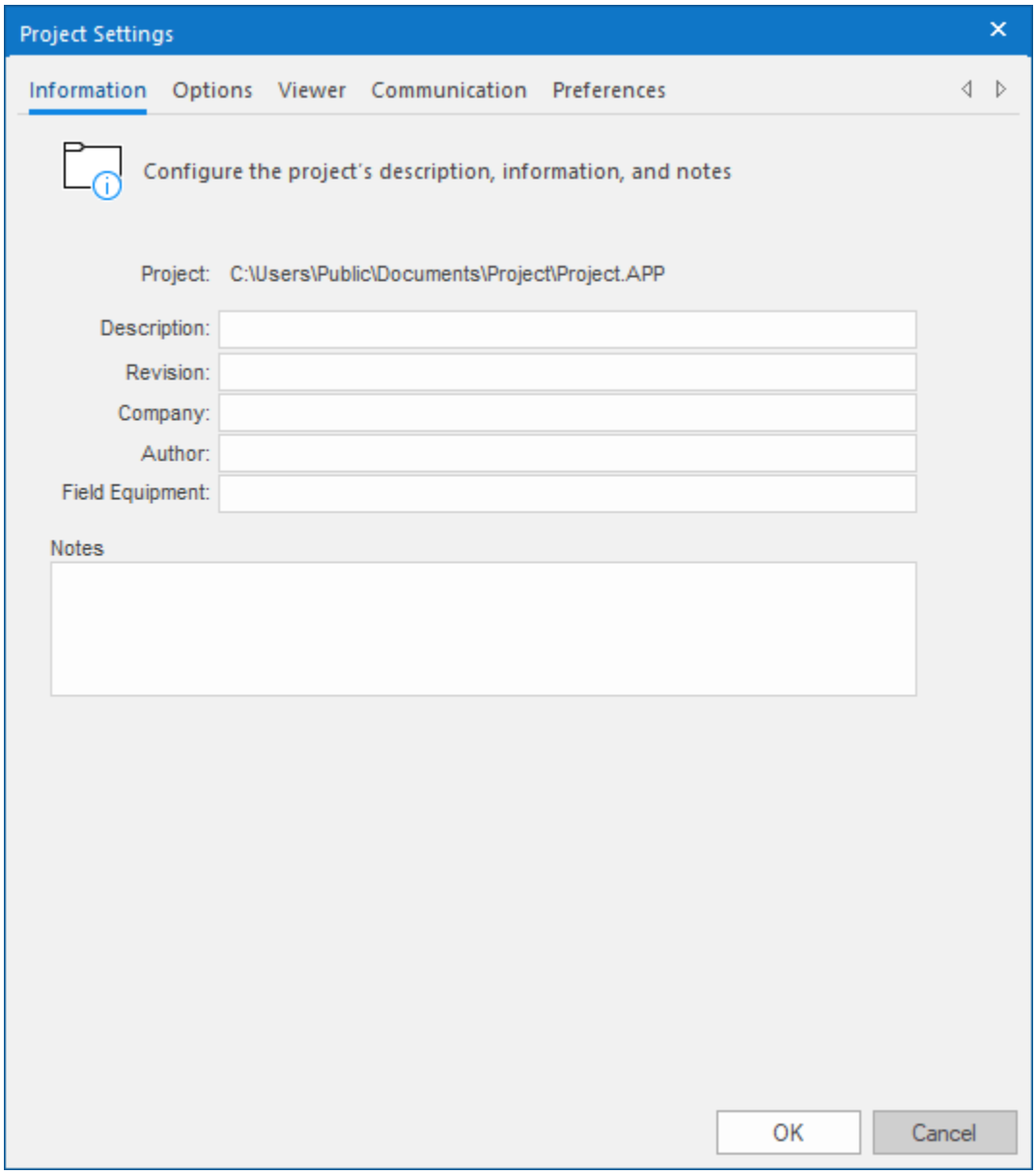
1. On the **Project** tab of the ribbon, in the **Settings** group, click **Target System**.
   A list of available target systems is displayed, grouped under the **Windows** and **Embedded** target platforms.
2. In the list, select the new target system for the project.
   The target system determines how many tags you can use in your project, among other things. For more information, see About target platforms, product types, and target systems on page 100.

Your project is converted to the selected target system.

If you changed from the **Windows** target platform to the **Embedded** target platform, some project features will no longer be available. Those features are automatically hidden in the project development environment in order to prevent you from developing a project that you cannot run on the selected target system.

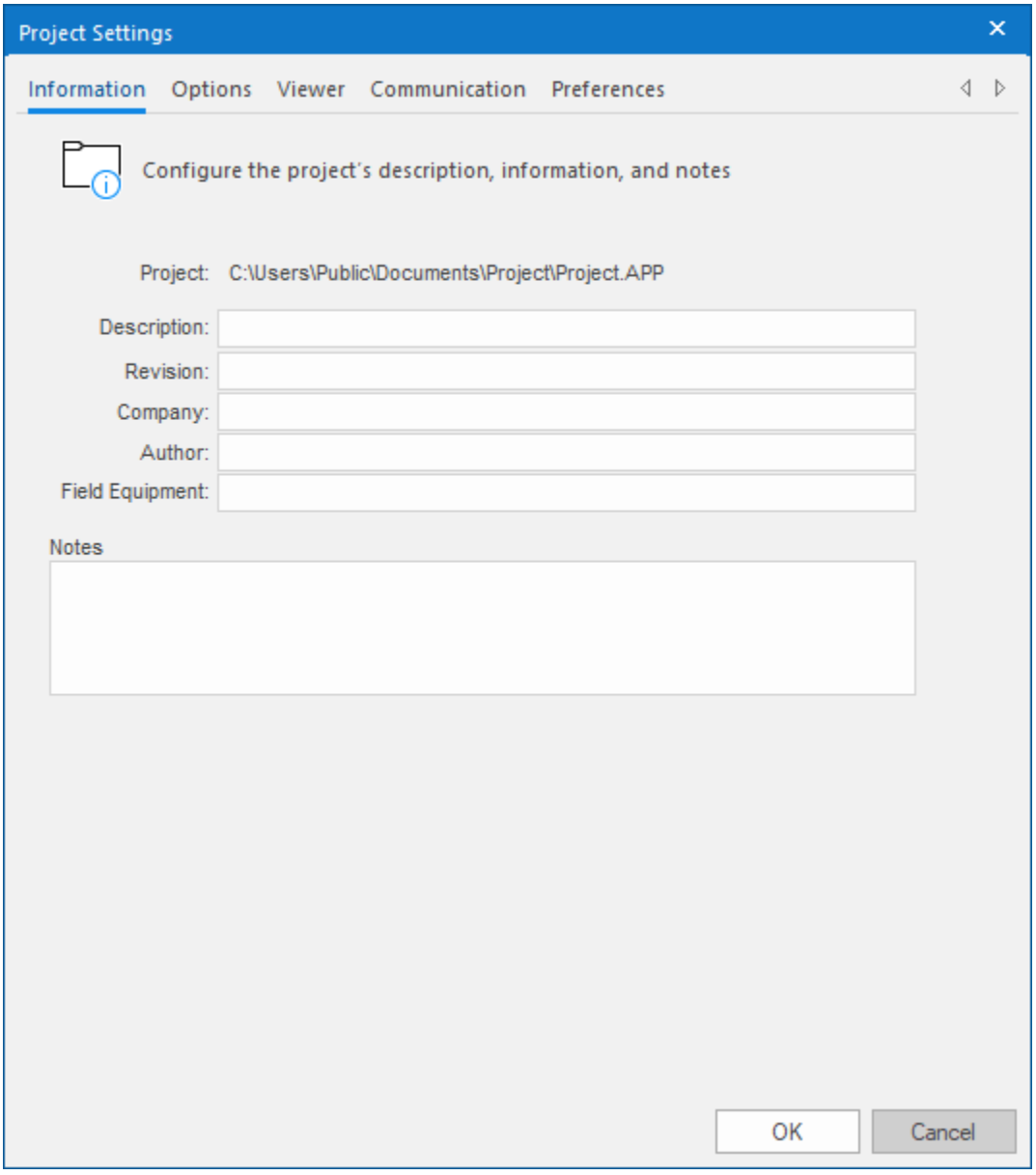
## Configuring additional project settings

Select **Project Settings** to open the *Project Settings* dialog, which controls settings that affect the overall project.




***Project Settings dialog***

### *Information tab*



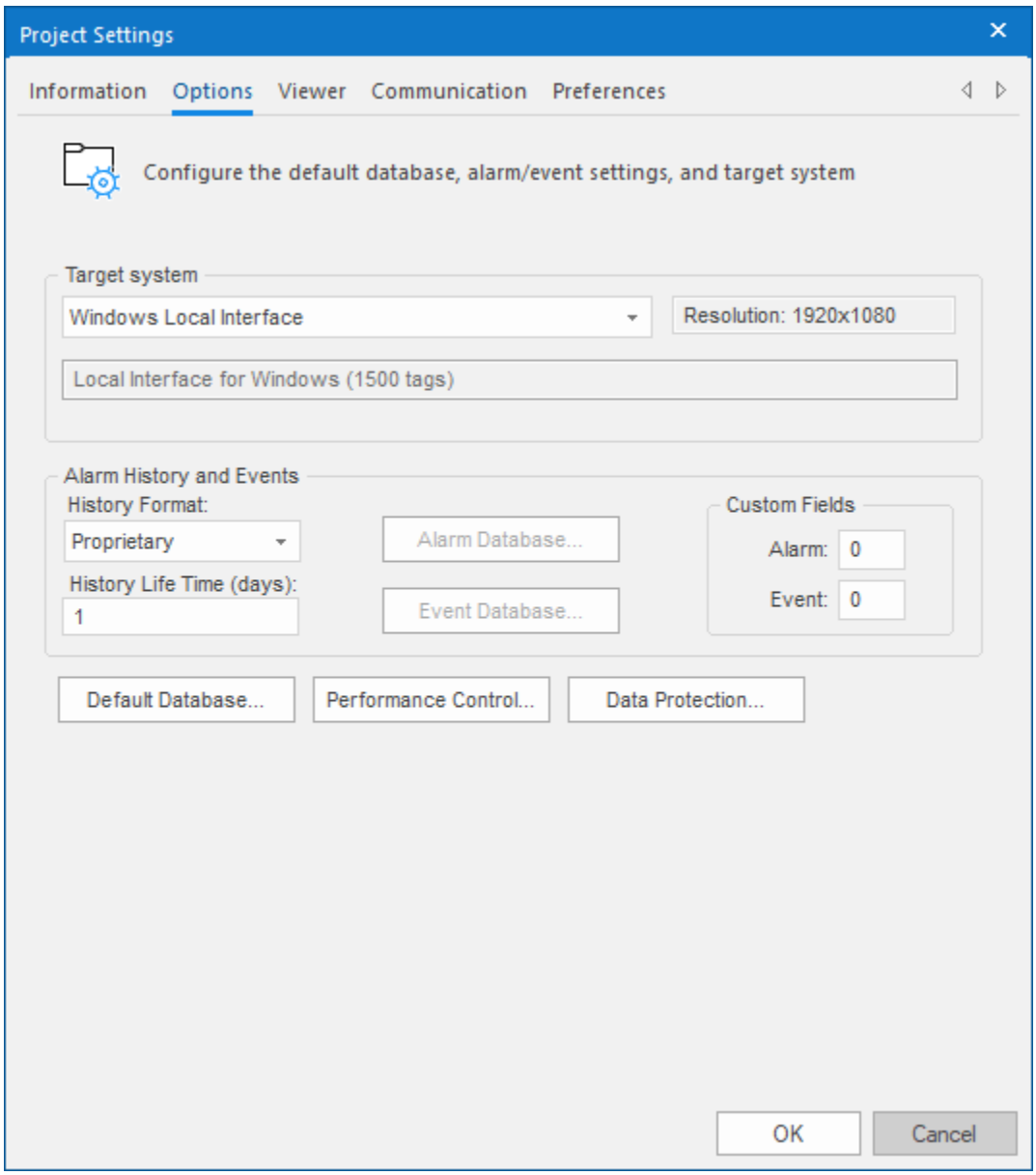

***Project Settings: Identification tab***

Use the following parameters to identify the project (for documentation purposes only).

- **Description**
- **Revision**
- **Company**
- **Author**
- **Field Equipment**
- **Notes**

### *Options tab*

Use this tab to specify parameters relating to your project in general.




***Project Settings: Options tab***

A description of these parameters follows:

- **Target system**: Use the combo box to specify the target system for the current project. The target system sets the project restrictions (such as number of tags supported) and must match your license. The description of the main license restrictions for each target system is displayed below the combo-box where you chose it.

> **Note:** If you specify a **Target System** level that does not match the actual license level on the target system, then your project might not run properly.

- **Resolution**: Displays your project's screen resolution.

- **Alarm History and Events**: Type a value into the History Life Time (days) field to specify how long to keep alarm and event history files. After the specified number of days, the project automatically deletes existing alarm/event history files that are older than the period specified. If you type zero in this field, the project does not delete any history files automatically. In such a case, you should create an external procedure to clean the old history files; otherwise, the free memory in the computer will eventually be depleted.
  - **History Format**: Select the format of the Alarm/History event, as follows:

| Format | Description |
|---|---|
| **Proprietary** | Saves the history data in the `Alarm` sub-folder of your project folder (by default) in text files using the proprietary format. |
| **Database** | Saves the history data in the SQL Relational Database specified by the user, using the built-in ADO interface. |
| **Binary** | Saves the history data in the `Alarm` sub-folder of your project folder (by default) in binary files using the proprietary format. |

  For more information, see Saving your alarm history / event log to an external database.
  - **Custom Fields**: Specify up to 10 custom fields for alarms and events, respectively, that will be saved in the history. For alarms, the custom fields are automatically added to the Alarm worksheet body. For events, the custom fields are available as parameters of the function `SendEvent`.

    If your project is running when you change these settings, you must stop the project and then run it again for the changes to take effect.

    > **Note:** This feature is not supported in projects running on Windows Embedded target systems.

- **Default Database**: Allows you to configure a Default Database, which can be shared by different tasks and objects. See Configuring a Default Database for All Task History for more information.
- **Performance Control**: Allows you to configure how memory is allocated for screen graphics during runtime. For more information, see Configure the performance control settings on page 112.

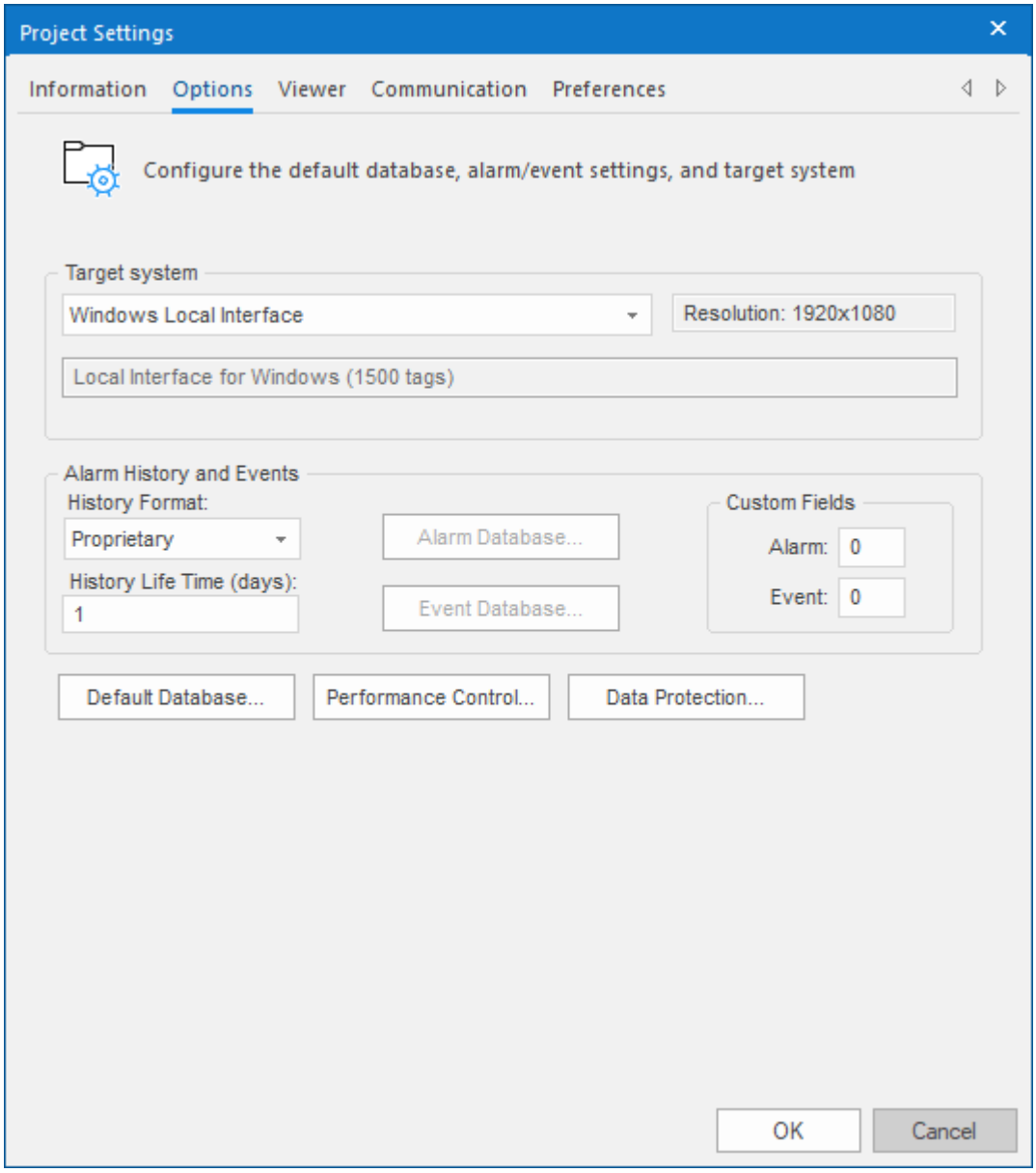
## SAVING YOUR ALARM HISTORY / EVENT LOG TO AN EXTERNAL DATABASE

By default, your project's alarm history and event log are saved to proprietary-format text files in your project's Alarms folder. However, you can change your project settings to save them to an external SQL database instead.

> **Note:** If your project was created with an earlier version of this software and then upgraded to the latest version, you should consider starting over with new database tables.
>
> In the latest version of this software, new database tables are automatically indexed by event time in order to improve run-time performance. Existing database tables cannot be indexed in this way, so if you can afford to discard that data, you should update your database configuration to create new tables.
>
> If you do this, you must also manually edit your project file (*<project name>*`.APP`) to correct the following setting:
>
> ```
> [Alarm]
> AddEventTimeColumn=1
> ```
>
> This setting exists in order to maintain backward compatibility, and it defaults to 0 for projects that were upgraded.

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**.

The *Project Settings* dialog is displayed.




***Project Settings: Options***

2. In the **Alarm History and Events** area, in the **History Life Time** box, type the number of days of history that you want to save.

   As the history exceeds the specified number of days, it will be automatically deleted in a first-in, first-out manner. If no number is specified — that is, if it is left blank or set to 0 — then history will never be deleted. There is no limit to how much history you can save, but the more you save, the more disk space it will take.

3. From the **History Format** list, select **Database**.

4. To configure a single, default database to be used for both the alarm history and the event log (as well as all other runtime tasks), in the **Default Database** area, click **Configure**.
   The *Default Database Configuration* dialog is displayed. Use the dialog to configure the database connection. For more information, see Configuring a default database for all task history.

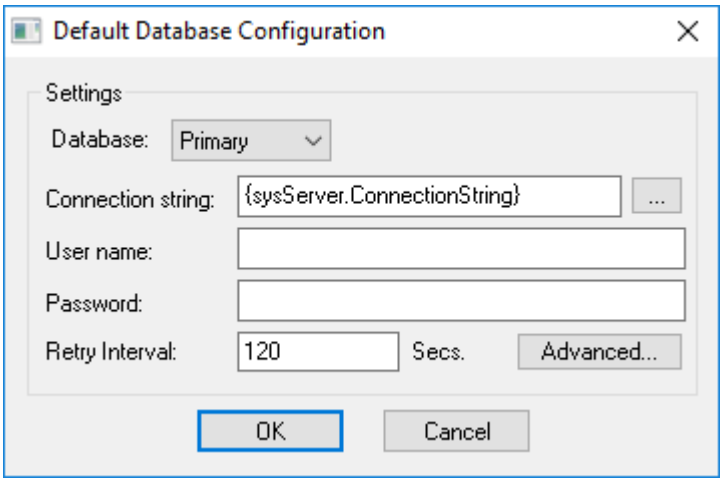
5. To configure a separate database for either your event log or your alarm history, click **Event Database** or **Alarm Database**, respectively.
   In either case, a *Database Configuration* dialog is displayed. Use the dialog to configure the database connection. For more information, see Database Configuration.
6. Click **OK**.

## CONFIGURING A DEFAULT DATABASE FOR ALL TASK HISTORY

You can configure a Default Database that will save the historical data from all Tasks in a project. After you do, when you create a new Task worksheet, you can choose either to use the Default Database or to configure a new database for that specific worksheet.

To configure the connection settings for the Default Database:

1. On the Project tab of the ribbon, in the Settings group, click **Options**. The *Project Settings* dialog is displayed.
2. Click **Configure**. The *Default Database Configuration* dialog is displayed.

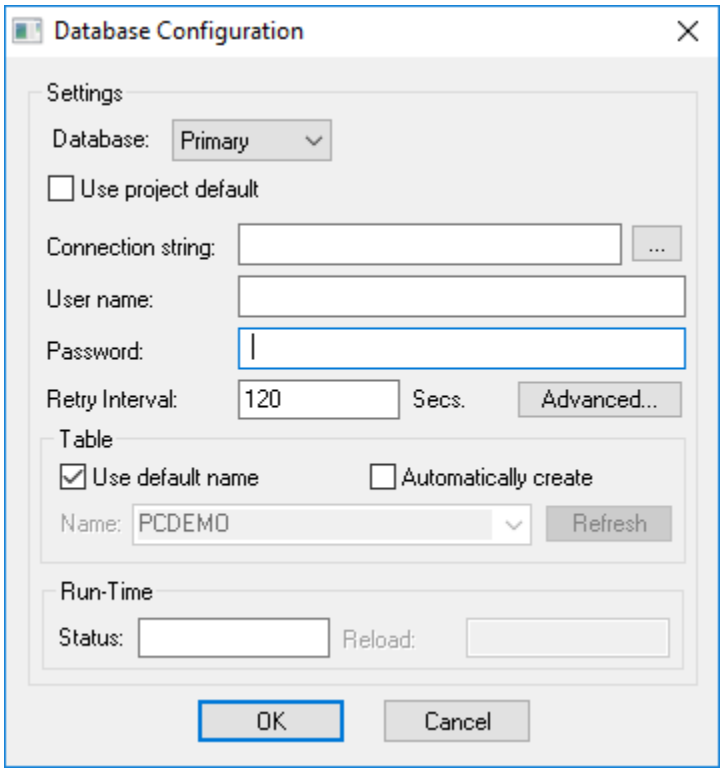



***Default Database Configuration dialog***

Please refer to *Database Configuration* dialog for help completing the fields in this window.

## DATABASE CONFIGURATION

The *Database Configuration* dialog allows you to configure the necessary settings to link EcoStruxure Machine SCADA Expert to an external database file.




***Database Configuration dialog***

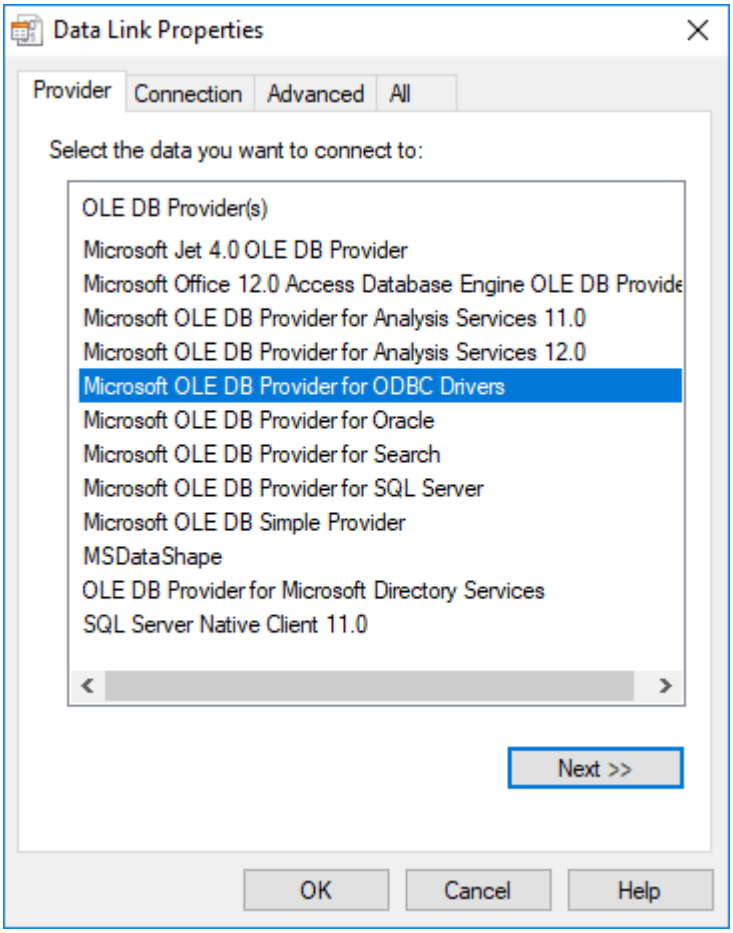
- **Database** combo-box: Allows you to select either Primary or Secondary. With Primary, all settings displayed in the Database Configuration window apply to the Primary Database interface. Otherwise, they apply to the Secondary Database interface. You can configure the Secondary database in the following modes:
  - **Disabled**: In this mode, EcoStruxure Machine SCADA Expert saves data in the Primary Database only. If the Primary Database is unavailable for any reason, the data is not saved anywhere else. This option may cause loss of data if the Primary Database is not available.
  - **Redundant**: In this mode, EcoStruxure Machine SCADA Expert saves data in both Primary and Secondary Databases. If one of these databases is unavailable, EcoStruxure Machine SCADA Expert keeps saving data only in the database that is available. When the database that was unavailable becomes available again, EcoStruxure Machine SCADA Expert synchronizes both databases automatically.
  - **Store and Forward**: In this mode, EcoStruxure Machine SCADA Expert saves data in the Primary Database only. If the Primary Database becomes unavailable, EcoStruxure Machine SCADA Expert saves the data in the Secondary Database. When the Primary Database becomes available again, EcoStruxure Machine SCADA Expert moves the data from the Secondary Database into the Primary Database.

> **Note:** The Primary and Secondary can be different types of databases. However, they must have the same fields.

  Using the Secondary Database, you can increase the reliability of the system and use the Secondary Database as a backup when the Primary Database is not available. This architecture is particularly useful when the Primary Database is located in the remote station. In this case, you can configure a Secondary Database in the local station to save data temporarily if the Primary Database is not available (during a network failure, for instance).
- **Use project default** checkbox: When this option is checked, EcoStruxure Machine SCADA Expert uses the settings configured in the Default Database for the task that is being configured (Connection string,

User name, Password, Retry Interval and Advanced Settings). When this option is not checked, you can configure these settings individually to the current task.

- **Connection string** field: This field defines the database where EcoStruxure Machine SCADA Expert will write and read values as well as the main parameters used when connecting to the database. Instead of writing the Connection string manually, you can press the browse button (...) and select the database type from the **Data Link Properties** window.




***Data Link Properties dialog***

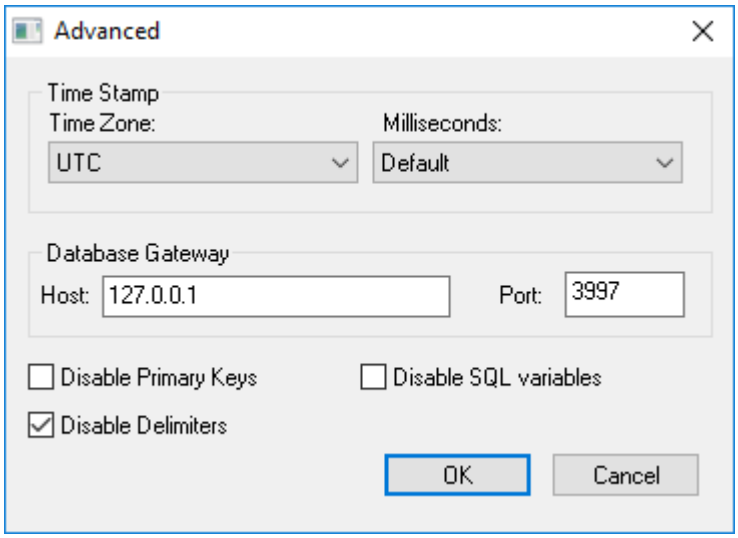
> **Note:** The list of Database Providers shown in the Data Link Properties window depends on the providers actually installed and available in the computer where you are running EcoStruxure Machine SCADA Expert. Consult the operating system documentation (or the database documentation) for further information regarding the settings of the Provider for the database that you are using.

- **User name** field: User name used to connect to the database. The user name configured in this field must match the user name configured in the database.
- **Password** field: Password used to connect to the database. The password configured in this field must match the password configured in the database.

> **Note:** In the **Connection string**, **User name**, and **Password** boxes, as in other boxes and fields that accept plain text, you can configure tag names in curly brackets (e.g., `{MyTag}`) in order to use the values of those tags. You can then change the tag values during run time and thereby change your database connection and credentials. You should be aware, however, that tag values are not encrypted when they are sent between the EcoStruxure Machine SCADA Expert project runtime server and connected thin clients. Therefore, to ensure that your database credentials cannot be

intercepted or compromised, you can configure only server tags — that is, tags that have **Scope** set to **Server**; for more information, see Choosing the Tag Scope on page 156 — in these boxes. The tags will be evaluated on the server only, and no tag values will be sent between the server and client.

- **Retry Interval** field: If EcoStruxure Machine SCADA Expert is unable to connect to the database for any reason, it retries automatically to connect to the database after the number of seconds configured in this field have passed.
- **Advanced** button: After pressing this button, you have access to customize some settings. For most projects, the default value of these settings do not need to be modified and should be kept.




***Database Configuration: Advanced dialog***

- **Time Zone** combo box:
  - **Local Time + Time Difference**: Save the local time on the computer, plus the difference (bias) between the local time zone and Coordinated Universal Time (UTC).
  - **Local Time**: Save the local time only with no bias. This is not recommended.
  - **UTC**: Save the UTC time only. This is the default, and it is strongly recommended for most situations.
- **Milliseconds** combo box: You can configure how the milliseconds will be saved when saving the date in the database. Each database saves the date in different formats; for example, some databases do not support milliseconds in a **Date** field. The following options are available:
  - **Default**: Uses the format pre-defined for the current database. The databases previously tested by Schneider Electric are previously configured with the most suitable option. When selecting Default, EcoStruxure Machine SCADA Expert uses the setting pre-configured for the current database type. If you are using a database that has not been previously configured, the Default option attempts to save the milliseconds in a separate field.

    **Tip:** The default option for each database is configured in the `StADOSvr.ini` file, stored in the `\BIN` sub-folder of EcoStruxure Machine SCADA Expert. See Studio Database Gateway on page 818 for information about how to configure the `StADOSvr.ini` file.
  - **Disable**: Does not save the milliseconds at all when saving the date in the database.
  - **Enable**: Saves the milliseconds in the same field where the date is saved.
  - **Separate Column**: Saves the milliseconds in a separated column. In this case, the date is saved in one field (without the milliseconds precision) and the number of milliseconds is saved in a different column. This option is indicated where you want to save timestamps with the precision of milliseconds but the database that you are using does not support milliseconds for the **Date** fields.

- **Database Gateway**: Enter the Host Name/IP Address where the EcoStruxure Machine SCADA Expert Database Gateway will be running. The TCP Port number can also be specified, but if you are not using the default, you will have to configure the EcoStruxure Machine SCADA Expert Database Gateway with the same TCP Port. See Studio Database Gateway on page 818 for information about how to configure the advanced settings for the EcoStruxure Machine SCADA Expert ADO Gateway.
- **Disable Primary Keys**: For some modules, EcoStruxure Machine SCADA Expert will try to define a primary key to the table in order to speed up the queries. If you are using a database that does not support primary keys (e.g., Microsoft Excel), then you should select (check) this option.
- **Disable Delimiters**: Select this troubleshooting option to disable the delimiters that are used to format communications with the database. Delimiters can cause problems when a Trend Control or Grid builds a query that includes aggregates such as Min and Max.
- **Disable SQL variables**: Select this troubleshooting option to disable SQL variables, such as **`@Value1`** and **`?`**, that are often used in SQL statements and queries. Some specific database providers do not support these variables.

> **Note:** If you select this option, you might need to specify the language or culture that should be used to format values in SQL statements. For more information, see "Advanced Settings" in Studio Database Gateway on page 818.

### Table Pane

This area allows you to configure the settings of the Table where the data will be saved. All tasks can share the same database. However, each task (Alarm, Events, Trend worksheets) must be linked to its own Table. EcoStruxure Machine SCADA Expert does not check for invalid configurations on this field, therefore you should make sure that the configuration is suitable for the database that you are using.

- **Use default name** checkbox: When this option is checked (default), EcoStruxure Machine SCADA Expert saves and/or retrieves the data in the Table with the default name written in the **Name** field.
- **Automatically create** checkbox: When this option is checked (default), EcoStruxure Machine SCADA Expert creates a table with the name written in the **Name** field automatically. If this option is not checked, EcoStruxure Machine SCADA Expert does not create the table automatically. Therefore, it will not be able to save data in the database, unless you have configured a table with the name configured in the **Name** field manually in the database.
- **Name**: Specifies the name of the Table from the database where the history data will be saved.

> **Tip:** To specify a sheet in a Microsoft Excel spreadsheet file, use the following syntax:
>
> **`[`*`sheetname`*`$]`**

- **Refresh** button: If the database configured is currently available, you can press the **Refresh** button to populate the **Name** combo-box with the name of the tables currently available in the database. In this way, you can select the table where the history data should be saved instead of writing the Table name manually in the **Name** field.

### Run-Time Pane

This area allows you set runtime values. The following fields are available:

- **Status** (output) checkbox: The tag in this field will receive one of the following values:

| Value | Description |
|---|---|
| 0 | Disconnected from the database. The database is not available; your configuration is incorrect or it is an illegal operation. |
| 1 | The database is connected successfully. |
| 2 | The database is being synchronized. |

- **Reload** (output): Specify a reload tag if you are using curly brackets in any of the configuration fields. When you want to reconnect to the database using the updated values on your tags, set the tag on this field to 1.

EcoStruxure Machine SCADA Expert will update the configuration when trying to perform an action in the database, setting the tag back to 0 when it is finished.

**See also**:

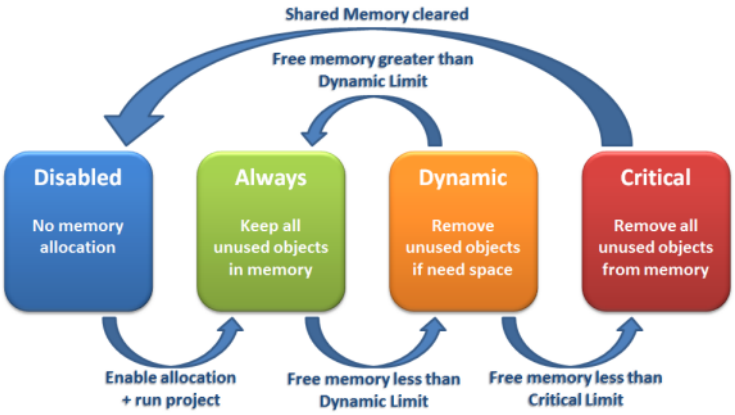
Configuring a Default Database for All Task History.

## CONFIGURE THE PERFORMANCE CONTROL SETTINGS

Configure the performance control settings to determine how memory is allocated for screen graphics during run time.

The project runtime client software has been improved to keep screen graphics (e.g., objects, images, fonts) in memory rather than load them from the hard drive each time a screen is opened. This makes opening and switching screens much faster, which in turn improves the overall run-time performance.

Devices that run the client software often have limited memory, however, so it is necessary to change the method of memory allocation as the memory becomes full.




***Memory allocation states***

When memory allocation is enabled (which it is by default in new projects) and your project is run, all unused objects are kept in memory so that project screens can be reopened or redrawn quickly. This method or state of memory allocation is called Always.

As the memory fills with objects, however, the amount of free memory decreases and may eventually reach the Dynamic Limit (i.e., the value configured in the **Before starting dynamic allocation** setting). When this happens, unused objects may be kept in memory but are removed if the space is needed for other objects that are actually being used. This method or state of memory allocation is called Dynamic.

As the memory continues to fill with objects — typically because the client has many project screens open and therefore many objects being used — the amount of free memory decreases until it finally reaches the Critical Limit (i.e., the value configured in the **Before disabling allocation** setting). When this happens, all unused objects are removed from memory and memory allocation is disabled until you restart the project. This method or state of memory allocation is called Critical.

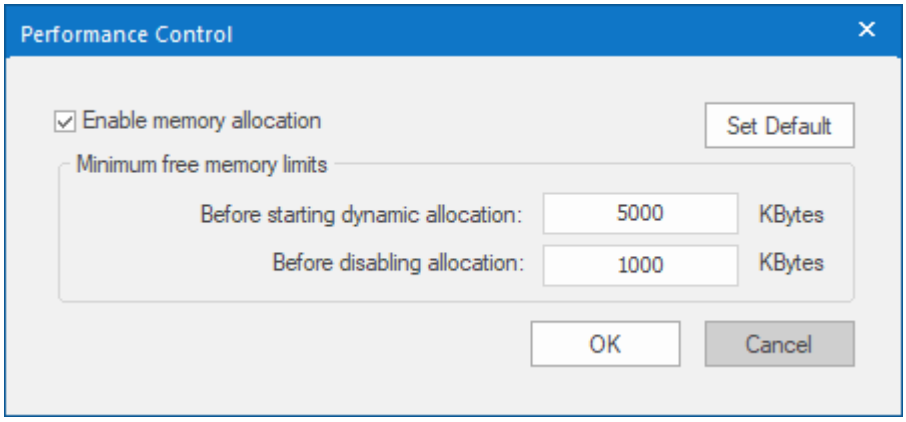
> **Tip:** Icon files (`*.ICO`) cannot be kept in memory. If you use many icon files in your project — particularly in Button objects — consider replacing them with new files in another format such as GIF, JPG, or PNG.

Performance control has default settings that should work for most projects running on most clients, but if you have problems, you can adjust the settings for your project or even disable memory allocation altogether.

To configure the performance control settings:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**.
   The *Project Settings* dialog box is displayed, with the **Options** tab selected.
2. In the **Options** tab, in the **Performance Control** area, click **Configure**.

The *Performance Control* dialog box is displayed.




***Performance Control dialog box***

3. To completely disable memory allocation, clear **Enable memory allocation**.
4. To adjust the limit at which memory allocation will change from Always to Dynamic, type a new value in the **Before starting dynamic allocation** box.
5. To adjust the limit at which memory allocation will change from Dynamic to Critical, type a new value in the **Before disabling allocation** box.
6. To restore the default settings, click **Set Default**.

You can monitor memory allocation during run time by calling the function `GetPerformanceMetric`.

## ENABLE DATA PROTECTION TO ENCRYPT SENSITIVE INFORMATION

Enable Data Protection in order to encrypt sensitive information such as the user names and passwords that you use to connect to databases and remote servers.

Before you begin this task, you should note the user names, passwords, and database connection strings that you have already configured in your project. You should also consider backing up your entire project, but that is not absolutely necessary.

There are many places in your project where you configure settings that can be encrypted by Data Protection. These places include but are not limited to:

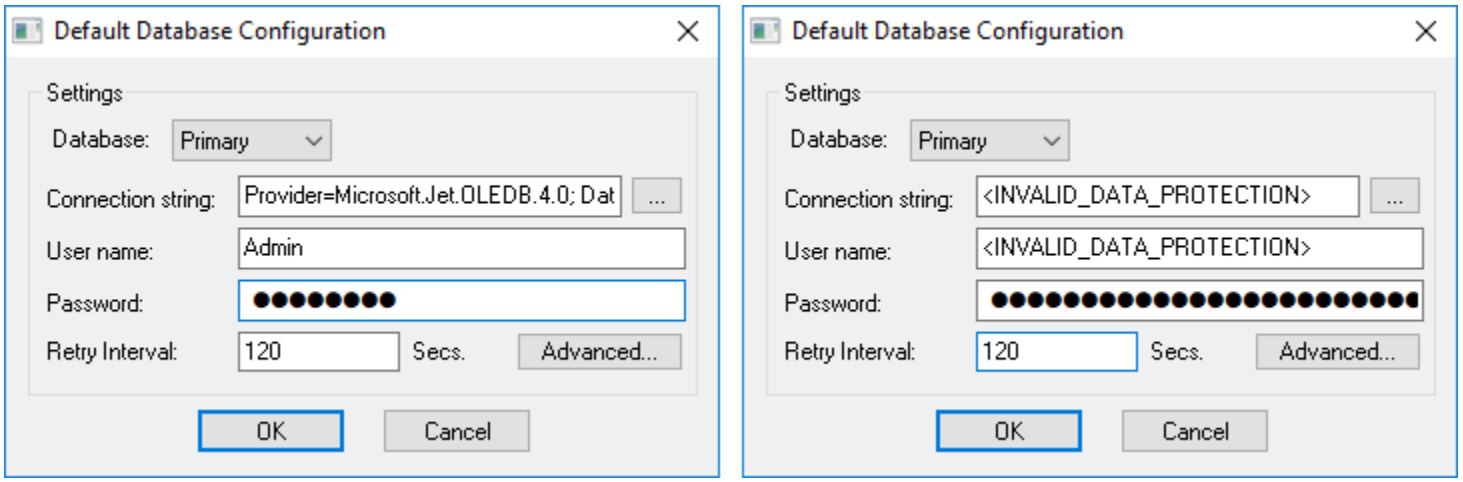
- In the Email and FTP settings;
- In the project security system, to connect to a LDAP server;
- In a Trend worksheet, to connect to a Historian database;
- In an OPC UA Client worksheet, to connect to an OPC UA server;
- In an OPC XML/DA Client worksheet, to connect to an OPC XML/DA server;
- In the settings to run your project as a Windows service; and
- In many other worksheets and screen objects that use the Studio Database Gateway to connect to external databases. For more information, see Database Configuration on page 108.

By default, these settings always have some basic protection that is hardcoded into EcoStruxure Machine SCADA Expert 2020, so they cannot be easily intercepted over the network. The settings are not fully encrypted, however, so they might be vulnerable to serious attacks.

You can enable Data Protection in your project settings in order to fully encrypt these settings. The encryption is done using Microsoft's Data Protection API (DPAPI), which is built into Windows, and it is keyed to the computer where Data Protection is enabled. The settings will still be displayed in plain text whenever you edit your project on the same computer, but they will be encrypted for all other development and run-time purposes. For a complete description of DPAPI, go to: msdn.microsoft.com/library/ms995355

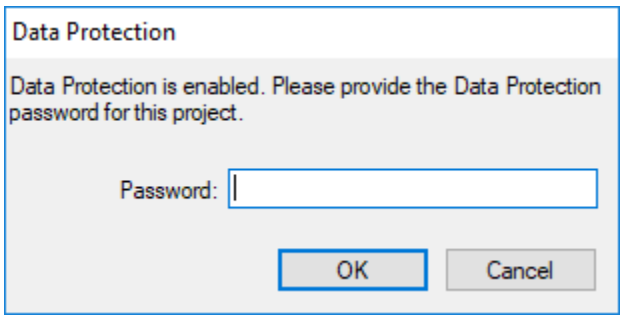
It is a simple procedure to enable Data Protection, but actually doing so will deeply affect your project and change how it can be used on other computers. First, all user names, passwords, and database connection

strings that you have previously configured will become invalid when you enable Data Protection. You will need to configure these settings again after Data Protection is enabled.




***Previously configured settings before (left) and after (right) Data Protection is enabled***

Second, because the encryption is keyed to the computer on which Data Protection is enabled, you will be prompted for your project's Data Protection password when you open the project on another computer for the first time. (This includes when you use the Remote Management tool to download the project to a target device. For more information, see Download your project to the target device on page 723.)



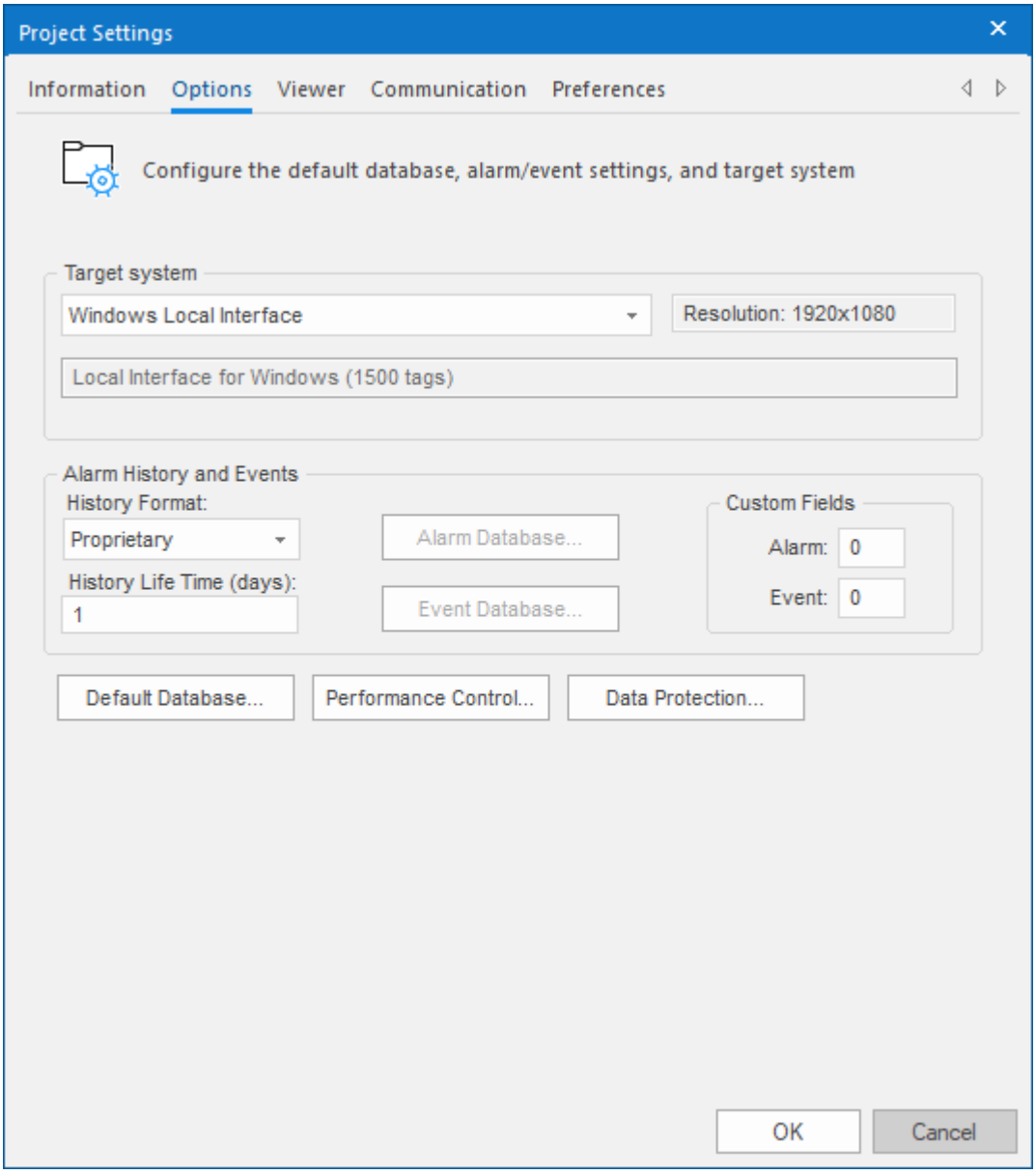

***Prompted for your password on another computer***

Third, you must also enable Data Protection in the Studio Database Gateway that will manage your project's database connections, so that the gateway can communicate securely with the project runtime server during run time. For more information about how to do that, see Studio Database Gateway.

To enable Data Protection in your project settings:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**.

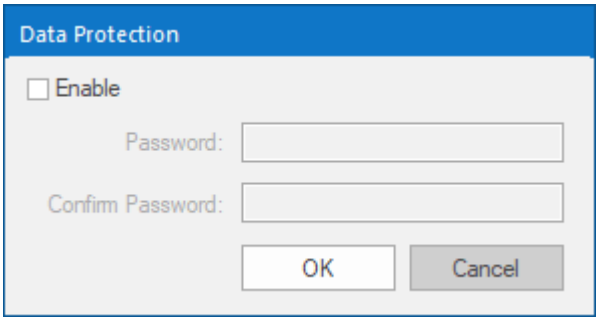
The *Project Settings* dialog box is displayed with the **Options** tab selected.




***Project Settings: Options dialog box***

2. Click **Data Protection**.

The *Data Protection* dialog box is displayed.




***Data Protection dialog box***

3. Select the **Enable** check box.
   The **Password** and **Confirm Password** boxes become active.
4. In the **Password** box, type your desired Data Protection password, and then in the **Confirm Password** box, type it again.

   A strong password should be at least 8 characters long and include at least 1 special character, 2 alphabetic characters, and 2 numeric characters. If your password does not meet these criteria, it will be accepted but a warning message will be displayed.
5. Click **OK**.

Data Protection is now enabled. You can disable it at any time simply by clearing the **Enable** check box; the Data Protection password is not required to do so. However, doing so will have different results depending on when you do it.

When you enable Data Protection, all previously configured user names, passwords, and connection strings become invalid. If you immediately disable Data Protection without making any other changes, the previously configured settings will be restored. Otherwise, you must configure the settings with new values.

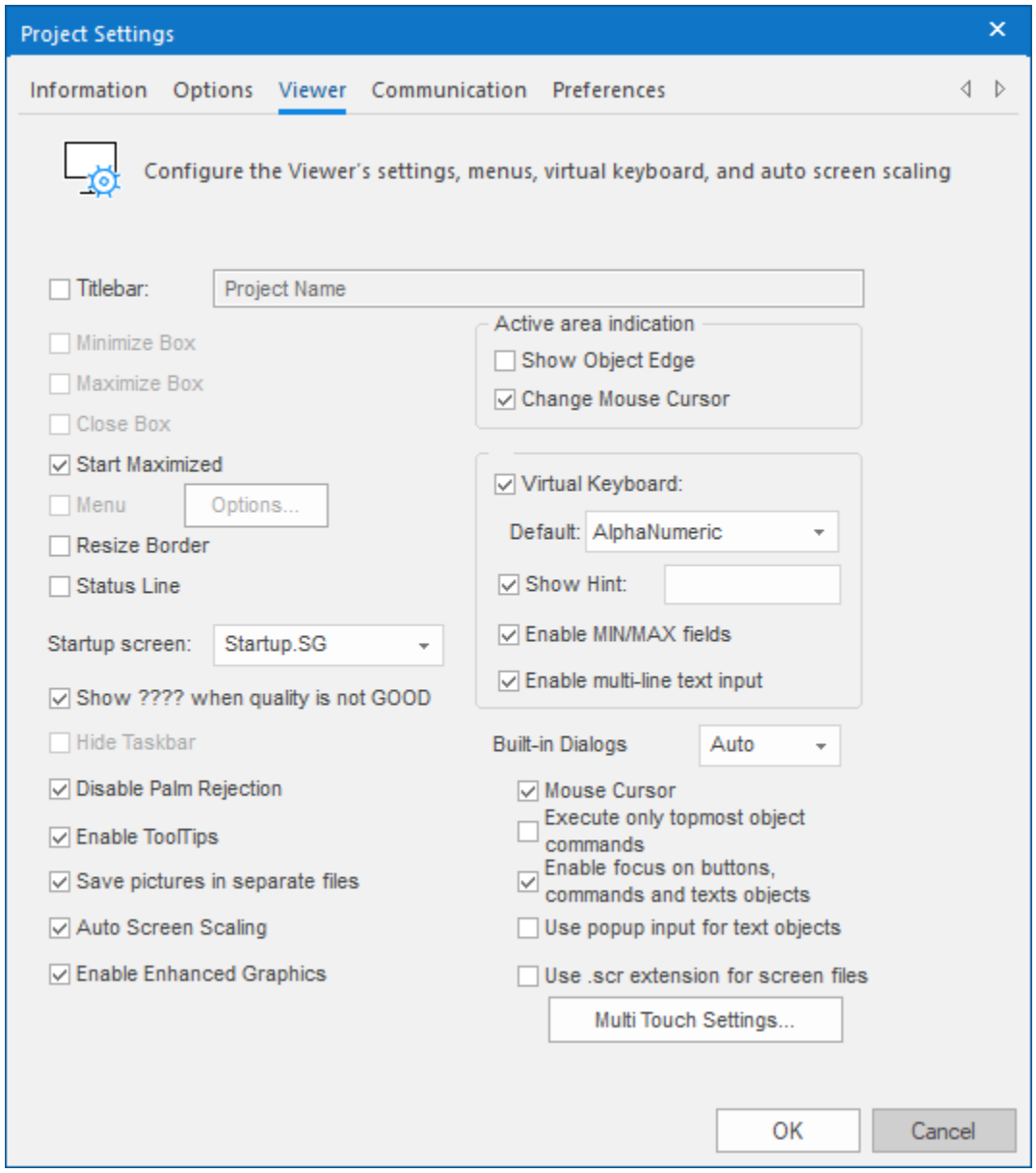
In contrast, if you disable Data Protection after you have configured the settings with new values, the settings will remain encrypted and therefore unusable during run time. You will need to either configure the settings again with new values OR re-enable Data Protection with the same password, before you try to run your project.

If you ever change the Data Protection password in your project settings, you must also update the password on the target device(s) to which you download the project, as well as in the Studio Database Gateway.

> **Note:** Information that has been encrypted with DPAPI cannot be decrypted without your Data Protection password. If you lose your password, neither Microsoft nor our own Technical Support representatives can recover the information for you. You will need to reconfigure all of the affected project settings.

### *Viewer tab*

Use the **Viewer** tab of the project settings to configure the project viewer and change certain run-time behaviors.




***Project Settings: Viewer tab***

**Titlebar**

Select this option and type a new name into the field provided to specify or change the default titlebar text for the Viewer window.

**Minimize Box, Maximize Box, Close Box**

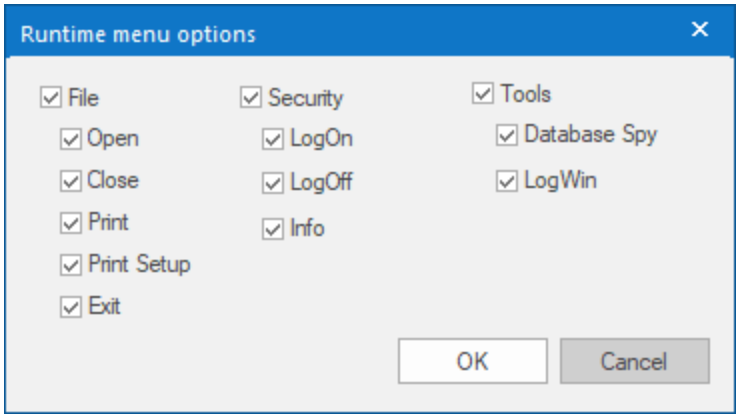
Select or clear these options to show (enable) or hide (disable) these buttons on the Viewer window.

**Start Maximized**

Select this option to maximize the Viewer window automatically when you run your project.

**Menu**

Select this option and then click **Options** to specify which menu options are available at run time. When the *Runtime menu options* dialog displays (as follows), click the checkboxes to show (enable) or hide (disable) these menu options.




**Resize Border**

Select this option to allow the user to resize the Viewer window during run time.

**Status Line**

Select this option to display the Status Line in the runtime project.

**Startup screen**

Click the combo box and select the screen (.scc or .scr file) or screen group (.sg file) that you want to display automatically when a thin client accesses your project.

> **Note:** Another way to specify a screen or screen group as the startup screen is to right-click on it in the *Project Explorer* and then choose **Set as startup** from the shortcut menu.

**Show ???? when quality is not GOOD**

Some screen objects (e.g., Text Box) can be configured to directly display tag values. Select this option to have an object display question marks (????) instead of the tag value when its quality is not GOOD (i.e., BAD).

**Hide Taskbar**

Select this option to hide the Windows taskbar by default.

**Disable Palm Rejection**

Select this option to disable Palm Rejection during project run time. Palm Rejection is a feature on Windows touchscreen devices that detects and rejects accidental touches from the operator's palm. However, it can somewhat slow the touchscreen's responsiveness, so disabling it can improve the performance of the project runtime.

Palm Rejection is not available on anything other than Windows touchscreen devices, so selecting this option will have no effect in projects running on other target platforms.

**Enable ToolTips**

Select this option to see Windows ToolTips when running your project. You can configure tooltips in the **Hint** field of the *Object Properties* dialog of each object.

**Save pictures in separate files**

When this option is selected, images that you paste into a screen are automatically converted to Linked Picture objects and then saved as separate bitmap (.bmp) image files in your project folder at `<project name>\Web\Resources`. You can then reuse the images without increasing

the size of your project, because every instance of the image will link to the same file. That should improve run-time performance.

When this option is cleared, images that you paste into a screen are kept as Bitmap objects and embedded in the screen file. Each instance of an image will be embedded, so if you use the same image more than once, it may greatly increase the size of the screen file.

> **Note:** The effect of this option is permanent: once a pasted image is converted to a Linked Picture object, clearing the option will not convert the image back to a Bitmap object.

The Resources sub-folder is automatically maintained by the development application, and it is used only for the bitmap images that are pasted and converted. You should not put other image files in it, because they may be moved or deleted. Instead, put other images that you want to link to in the Web sub-folder.

#### Auto Screen Scaling

Select this option to automatically scale project screens to fit the displays on client stations, even dynamically when a user resizes the viewer window. This is done by calculating the ratio between the project's full display resolution and the client station's actual display size and then multiplying both the screen size and the screen position by that ratio. Screens are only downscaled to fit smaller displays; they are not upscaled to fit larger displays.

This option applies only to the local Viewer module and the remote Secure Viewer. It does not apply to the other types of project clients, which have their own **Auto Screen Scaling** options in their respective settings.

This option is cleared by default because downscaling screens can sometimes make them unusable (e.g., buttons too small to be pressed, gauges too small to be read). If the screens are kept at full scale, a user can at least scroll and pan to see an entire screen.

#### Enable Enhanced Graphics

When this option is selected, your project screens will have enhanced graphics such as anti-aliasing and opacity.

Anti-aliasing smooths the lines, edges, captions, and gradient fills on Shapes (i.e., Line objects et al.), Text objects, and Button objects (Standard and OS styles). It is not supported for conical gradient fills.

Opacity (0% to 100%) allows you to make a screen object translucent and show other objects behind it. You can then arrange the objects to produce certain visual effects. To set opacity on an object, apply the Visibility/Position animation.

Enabling enhanced graphics might decrease run-time performance on some clients, so you should test it thoroughly before you deploy your project.

#### Active area indication

Click (enable) the **Show Object Edge** and **Change Mouse Cursor** checkboxes in this area to modify the object edge and the mouse cursor when moving the cursor over any object where the Command animation has been applied.

#### Virtual Keyboard

When this option is selected, the Virtual Keyboard (VK) is enabled for your project. The Virtual Keyboard allows the user to enter data (text or numeric values) during run time using the client station's touchscreen instead of a physical keyboard or keypad. For example, a Text object with the Text Data Link animation applied and the **Input Enabled** option selected.

You can establish a default configuration for the Virtual Keyboard:

##### Default

Select the default keyboard type to be used in your project, when no keyboard type is specified by the calling object or function.

##### Show Hint

When this option is selected, a hint is displayed in the title bar of the Virtual Keyboard window. For a specific object, you can configure the hint in the object properties. Otherwise, type a string value in the **Show Hint** box to serve as a defaut hint for all keyboards in your project.

**Enable Min/Max Fields**

When this option is selected and the **Keypad** type of keyboard is used, the minimum and maximum values allowed for the associated project tag will be displayed at the bottom of the keyboard. For some screen objects, you can configure those values in the object properties. Otherwise, the **Min** and **Max** properties of the associated project tag are used by default.

> **Note:** The **Min** and **Max** fields are displayed only on the **Keypad** type of keyboard, and only when the associated project tag is of Integer or Real type. If Min is greater than Max, user input will be disabled. If Min/Max configured on the object properties differs from Min/Max configured in the tag properties, the project runtime will attempt to scale the user input accordingly.

**Enable multi-line text input**

When this option is selected, the **AlphaNumeric** type of keyboard can be used to enter multi-line text, with "new line" control characters (i.e., CR+LF) between lines, in screen objects that accept multi-line text input.

**Built-in Dialogs**

Select or type the scale at which the built-in dialogs (i.e., Logon, E-Sign, and Virtual Keyboard) should be displayed during run time. This will make it easier to see and use the dialogs on project runtime clients with small, high-resolution displays. You can specify a scale between `100%` (native resolution) and `400%`, or you can select `Auto` which means the client will automatically select the best scale for the display.

**Mouse Cursor**

Select this option to show the mouse cursor in the runtime project.

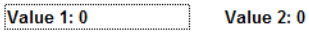
**Execute only topmost object commands**

This option controls how your project behaves when the user clicks in an area where two or more screen objects overlap. If this option is checked, only the commands on the topmost object will be executed. If this option is not checked, the commands on all of the overlapping objects will be executed.

> **Note:** The topmost object is the one with the highest ID number. (The ID number of an object is displayed in the status bar at the bottom of the development environment.) You can use the **Move to Back** / **Move to Front** and **Move Backward** / **Move Forward** tools to change the order in which objects are stacked.

**Enable focus on buttons, commands and text objects**

This option is selected by default. When it is selected, clicking on on a command button or text input during run time will put focus on that object, as shown in the illustration below:



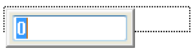

***Focus on text input (left), no focus on text input (right)***

After that, the end-user can press **Tab** to tab through all such objects in the screen or they can press **Enter** to activate the currently focused object (i.e., click on a command button, enter text in a text input), similar to a normal Windows application. This is useful if the client station has a physical keyboard and the end-user needs to quickly work through many such objects, because it saves time that would otherwise be spent repeatedly switching between the keyboard and the mouse or touchscreen.

To force the end-user to always use the mouse or touchscreen to activate screen objects, clear this option.

#### Use popup input for text objects

Select this option to display a small popup for text inputs, as shown in the illustration below:



This is an alternative to typing directly into the text object (which can seem like editing the screen itself and therefore be confusing to some end-users) and to displaying a Virtual Keyboard for input (which requires using the mouse or touchscreen).

> **Note:**
>
> If the **Virtual Keyboard** option (above) is selected, it will override this option.
>
> Also, if the **Enable focus on buttons, commands and text objects** option (above) is *cleared*, this option is automatically selected. This is to ensure there is some on-screen indication of which text input is currently active.

#### Use .scr extension for screen files

When this option is selected, screen files are saved with the .scr extension in your project folder. When this option is cleared, screen files are saved with the .scc extension.

The .scr extension has been deprecated because some anti-virus programs block it, and that can cause problems during project run time. The .scc extension supersedes the .scr extension, and it should be allowed by most if not all anti-virus programs.

This option was implemented in order to maintain backward compatibility with existing projects. In projects that are created with the latest version of this software, this option is cleared by default. In projects that were created with earlier versions of this software and then upgraded to the latest version, this option might be selected by default.

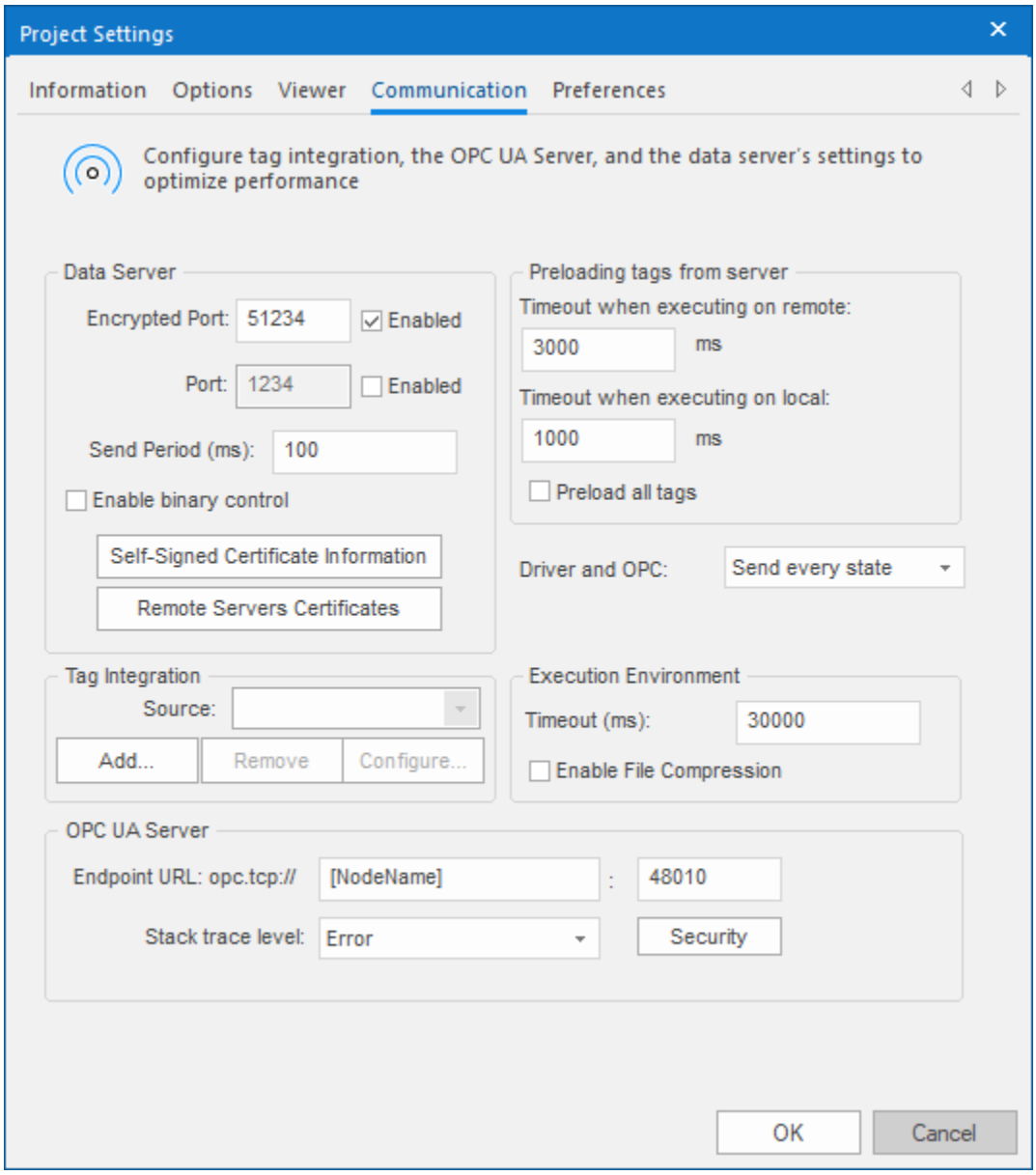
If you do not have issues during project run time, you do not need to do anything. If you do have issues, however, you can try selecting or clearing this option, as needed. When you do, all of the existing screen files in your project folder will be saved again with the preferred extension. Also, if you manually add a screen file to your project folder — for example, by copying it from another project folder — that file will be saved with the preferred extension the next time you either verify your project or open the screen for editing.

#### Multi-Touch Settings

Configure the default Multi-Touch settings for all screens in your project. For more information, see About the Multi-Touch settings for project screens on page 352.

### *Communication tab*

Use the **Communication** tab of the *Project Settings* dialog box to configure the communication settings for your project.




## Data Server

Configure the communication settings for your project's data server (i.e., the TCP/IP Server Runtime task), which exchanges data with thin clients and other projects during run time.

**Encrypted Port**

The port number for encrypted communication. (Communication is encrytped via TLS/SSL.) This port is enabled by default in order to make your project more secure. The default port number is 51234, but you can change it as long as you remember to update the corresponding settings in your thin clients and network firewalls.

**Port**

The port number for standard, unencrypted communication. This port is disabled by default in order to encourage you to use encrypted communication (see **Encrypted Port** above). You can enable both **Encrypted Port** and **Port** and then configure each client station to connect to either one or the other, depending on its particular needs. The default port number is 1234, but you can change it as long as you remember to update the corresponding settings in your thin clients and network firewalls.

#### Send Period (ms)

The period (in milliseconds) between two consecutive messages sent from the data server to client stations. The default period is 100 ms. A shorter period (i.e., a lower number) will increase run-time performance but also increase network traffic. Conversely, a longer period (i.e., a higher number) will decrease run-time performance but also decrease network traffic.

#### Enable binary control

Binary control encrypts communication between the data server and client stations by converting messages to binary data using a proprietary method, before the messages are sent to client stations. Enabling binary control can significantly decrease run-time performance.

> **Note:** Binary control is a legacy feature. It has been deprecated in favor of TLS/SSL (see **Encrypted Port** above), and it remains in the software only to support existing projects.

#### Self-Signed Certificate Information

Opens the Self-Signed Certificate Information tool, which you can use to edit the certificate your project presents to other clients and servers that are using the Encrytped Channel feature to communicate. For more information, see Edit your project's self-signed certificate on page 125.

#### Remote Servers Certificates

Opens the Certificate Store Management tool, which you can use to manage your project's certificate store. For more information, see Managing your project's certificate store on page 126.

### Preloading tags from server

To improve performance, the viewer preloads all of the Server tags (i.e., tags with Server scope) that are used in a project screen before it displays that screen. Configure the timeout settings for both remote and local viewers.

#### Timeout when executing on remote

Specifies the time (in milliseconds) that a Thin Client running on a remote station will wait to load the tags. If the timeout expires before the tags are loaded, the viewer will display the project screen even though it is not yet synchronized.

#### Timeout when executing on local

Specifies the time (in milliseconds) that the Viewer program running on the local station (i.e., on the computer that hosts the project runtime) will wait to load the tags. If the timeout expires before the tags are loaded, the viewer will display the project screen even though it is not yet synchronized.

#### Preload all tags

When this option is selected, the viewer will subscribe to and preload all of the Server tags that are defined for the entire project, not just the tags that are used in the project screen to be displayed. The same timeouts apply, so they might need to be adjusted to allow for the increased load.

### Driver and OPC

Select the method used by all communication drivers and OPC Client worksheets configured in the current project when writing values to the remote PLC/device.

**Send every state**

When the communication task is configured to write values upon a change of tag value, all changes in the tag value are buffered in a queue and sent to the device when the communication task (Driver or OPC) is executed.

> **Note:**
>
> There is a limit on the size of the buffer for tag value changes, to prevent accumulated changes from decreasing run-time performance. If the buffer size is exceeded, then only the most recent changes are kept until the next time the task is executed and the changes are sent. Also, a warning message is logged in the dump file.
>
> To adjust the buffer size, manually edit the following setting in your project file (*<project name>*.APP):
>
> ```
> [Options]
> DriverAndOpcBufferSize=5
> ```

**Send last state**

When the communication task is configured to write values upon a change of tag value, only the current (last) value of the tag is sent to the device when the communication task (Driver or OPC) is executed. When this method is selected, if the tag changed value more than once while the communication task was not being executed, the transient values of the tag are not sent to the device. This is the desired behavior for most projects.

### Tag Integration

Use these settings to integrate tags from remote devices into your project's tags database. For more information, see Tag Integration on page 225.

### Execution Environment

Configure the communication settings for the Remote Management tool, which sends your project files to a target system.

**Timeout (ms)**

Specifies the time (in milliseconds) that the project will wait to communicate with the target system.

**Enable File Compression**

Select this option to compress the system and project files before sending them to the target system. This may reduce the download time if you have a slow connection between your server and the target system. (If you have a fast connection, however, then selecting this option may actually *decrease* performance because each compressed file must be decompressed on the target system before the next file is sent. Select this option only if you have an extremely slow connection, such as dial-up.) File compression is disabled by default.

### OPC UA Server

Configure the communication settings for the OPC UA Server module, which can make your project tags database available to OPC UA clients on your network. For more information, see Configure the communication settings for OPC UA Server on page 624.

## EDIT YOUR PROJECT'S SELF-SIGNED CERTIFICATE

Use the Self-Signed Certificate Information tool to edit the certificate your project presents to other clients and servers that are using the Encrytped Channel feature to communicate.

> **Note:** This topic describes only the certificate for the Encrypted Channel feature. There is a similar but separate certificate for the OPC UA Server feature. For more information, see Configure the communication settings for OPC UA Server on page 624.

With regards to the Encrypted Channel feature, your project is considered a client if any of the following is true:

- Your project includes a TCP/IP Client worksheet configured to exchange data with another project, and the **Enable Encrypted Channel** option is selected in that worksheet's security settings;
- Your project's security system is set to **Distributed – Client** mode, and the **Encrypted Channel** option is selected in that mode's server settings; or
- You use the Watch and/or LogWin tools to monitor a project running on a remote computer, and you select the **Encrypted Channel** option when you make the connection.

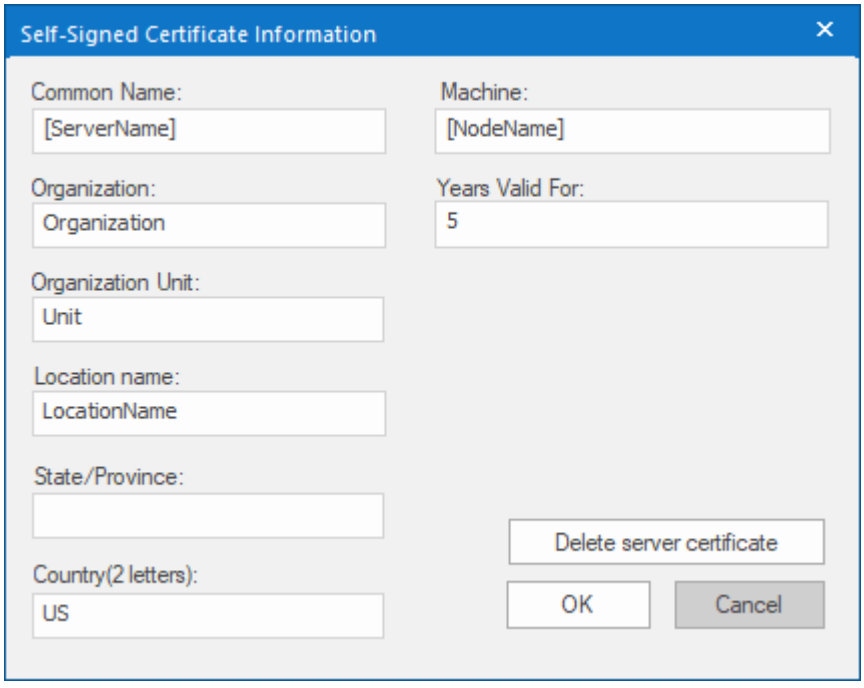
Your project is considered a server if the **Encrypted Port** option is selected in the Communication tab of the project settings.

A project can be both a client and a server, depending on how it is configured and how other projects connect to it, and it presents the same certificate for both types of connections. A self-signed certificate is signed by the software itself, as opposed to being signed by a Certificate Authority (CA).

To edit your project's self-signed certificate:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.
   The **Communication** tab of the *Project Settings* dialog box is displayed.
2. Under **Data Server**, click **Self-Signed Certificate Information**.

   The *Server Self-Signed Certificate Information* dialog box is displayed.




3. In the **Common Name** box, enter the common name of the data server itself.

The default is **[ServerName]**. This is a special string that includes **[NodeName]** (see **Machine** below), and it evaluates as the following:

```
StudioDataServer@[NodeName]
```

4. In the **Organization**, **Organization Unit**, **Location Name**, **State/Province**, and **Country** boxes, enter the appropriate information for your project.
5. In the **Machine** box, enter the name of the machine that hosts the data server.
   The default is **[NodeName]**. This is a special string that automatically gets the name of the computer or device that hosts the project runtime.
6. In the **Years Valid For** box, enter the number of years for which the certificate will be valid, starting from the date it is issued.
   The default number of years is 5. When a certificate expires, you must delete it and then issue a new one.
7. Click **Delete server certificate**.
   This is to make sure the existing certificate (if any) is deleted from the project files, so that a new certificate can be issued with the updated information.
8. Click **OK** to close the *Server Self-Signed Certificate Information* dialog box.

The project runtime automatically issues the data server certificate when the project is run and the TCP/IP Server Runtime task is started. At that time, the project runtime gets the name and address of the computer or device that hosts it, and then it uses that information to issue the certificate. As such, if you test your project on your local computer, a certificate will be issued using the name and address of that computer. You should delete the certificate — by clicking **Delete server certificate** — before you copy your project to another computer or download it to a project runtime.

When the certificate is issued, the certificate files are saved in your project's certificate store at:

```
<project name>\Config\certstore\own\studio_dataserver.der
<project name>\Config\certstore\own\studio_dataserver.pem
```

The .der file is the certificate itself, and the .pem file is the associated key. Both files must be present for the certificate to be valid.

You can double-click the .der file in Windows in order to examine the certificate information. You will see the **Subject** field of the certificate contains the value that you entered for **Common Name** above.

You will also see the **Subject Alternative Name** field of the certificate contains all of the valid names and addresses of the computer or device that hosts the project runtime. When a client tries to connect to the server, it compares the actual name or address of the server's host against the **Subject Alternative Name** field of the server's certificate. If the name or address cannot be validated, the connection fails. This is done to confirm that the server's certificate was issued on the server's host, rather than copied from another computer or device.

You can replace a self-signed certificate with a CA-signed certificate, as long as the certificate files have the correct file names and locations. If they do not, the project runtime will not be able to find and use them. Acquiring a CA-signed certificate and then using it to sign other certificates is beyond the scope of this documentation, however.

## MANAGING YOUR PROJECT'S CERTIFICATE STORE

Use the Certificate Store Management program to manage your project's certificate store. Certificates are exchanged between clients and servers that are using the Encrytped Channel feature to communicate.

> **Note:** This topic describes only the certificate store for the Encrypted Channel feature. There is a similar but separate certificate store for the OPC UA Server feature. For more information, see How to manage OPC UA Server during project run time on page 631.

With regards to the Encrypted Channel feature, your project is considered a client if any of the following is true:

- Your project includes a TCP/IP Client worksheet configured to exchange data with another project, and the **Enable Encrypted Channel** option is selected in that worksheet's security settings; or

- Your project's security system is set to **Distributed – Client** mode, and the **Encrypted Channel** option is selected in that mode's server settings.
- You use the Watch and/or LogWin tools to monitor a project running on a remote computer, and you select the **Encrypted Channel** option when you make the connection.

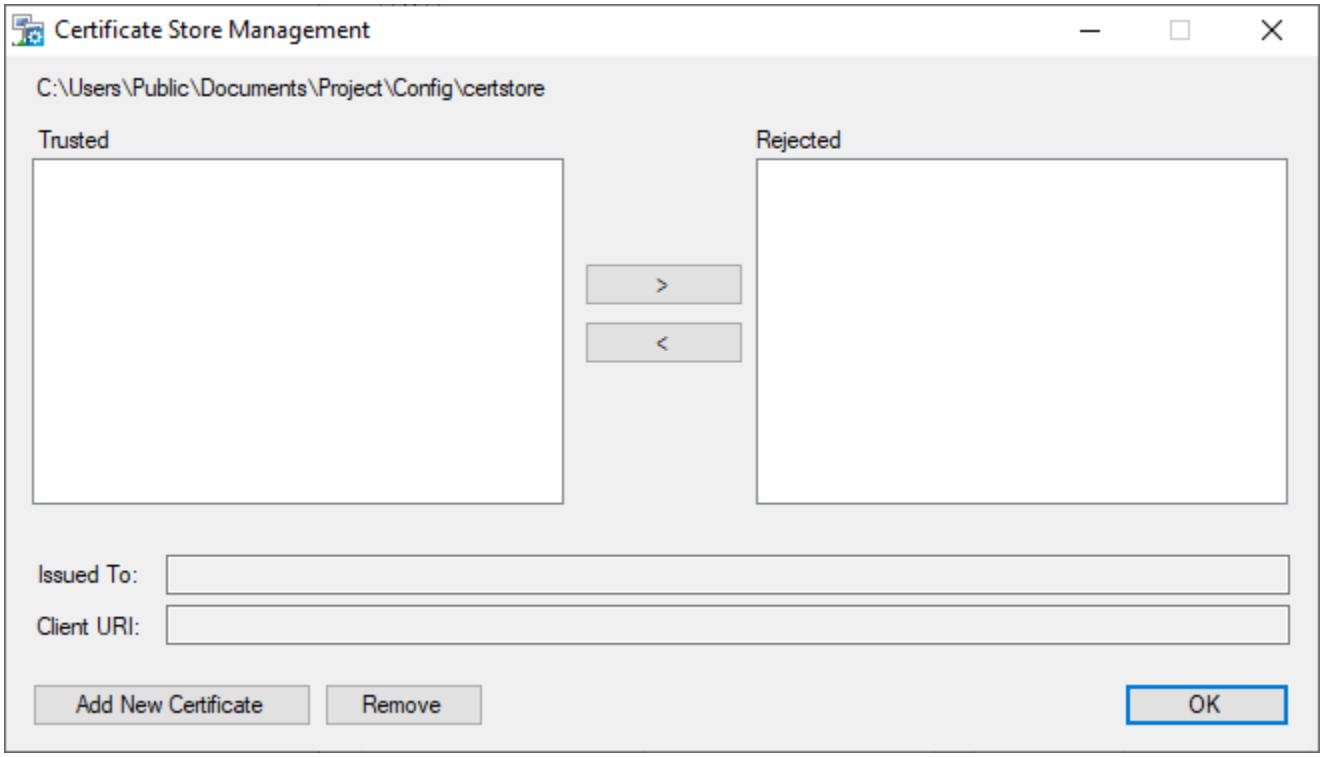
Your project is considered a server if the **Encrypted Port** option is selected in the Communication tab of the project settings.

A project can be both a client and a server, depending on how it is configured and how other projects connect to it, and it has a single certificate store for both types of certificates.

When a client attempts to connect to and communicate with a server, they will present their certificates to each other. Your project will automatically trust or reject a certificate depending on whether you selected the **Automatically trust server certificate** option for a given feature, and the certificate files are saved in the appropriate folders in your project's certificate store at:

```
<project name>\Config\certstore\rejected\
<project name>\Config\certstore\trusted\
```

This software includes a small utility program called Certificate Store Management (`CertStoreManager.exe`), which helps you to view certificates and move them between folders in the certificate store. If you have the full SCADA software installed, you can open the program by clicking **Remote Servers Certificates** in the Communication tab of your project settings. The program will automatically open the certificate store for the current project:




***Certificate Store Management program***

**Trusted, Rejected**

These lists display the respective contents of the `trusted` and `rejected` folders in your project's certificate store. To move a certificate from one list to the other, select it and then click **>** or **<** as needed.

**Issued To, Client URI**

Certificate information extracted from the selected certificate. This information cannot be edited.

**Add New Certificate**

Click to add a new certificate to the **Trusted** list. You will be asked to locate and open the certificate file. You should be familiar with certificate file names and extensions, as defined by the X.509 standard, so that you can locate the correct file. The file will be copied to the `trusted` folder in your project's certificate store.

**Remove**

Click to remove the selected certificate; the certificate file will be deleted from your project's certificate store, regardless of which list/folder it is in. Be aware that removing a certificate does not prevent a client or server from presenting the same certificate again in the future. If you want to reject the certificate, move it to the **Rejected** list.

> **Tip:** It is often useful to add certificates to your project's certificate store while you are still developing your project, so that they can be downloaded with the rest of the project files.

If the SCADA software is licensed for Runtime only — that is, if it is installed on another computer and running only as a project runtime server — you might not be able to access the project settings in order to open the Certificate Store Management program. In this case, you can manually run the program by locating it in the SCADA application folder and then double-clicking it. The program file should be located at:

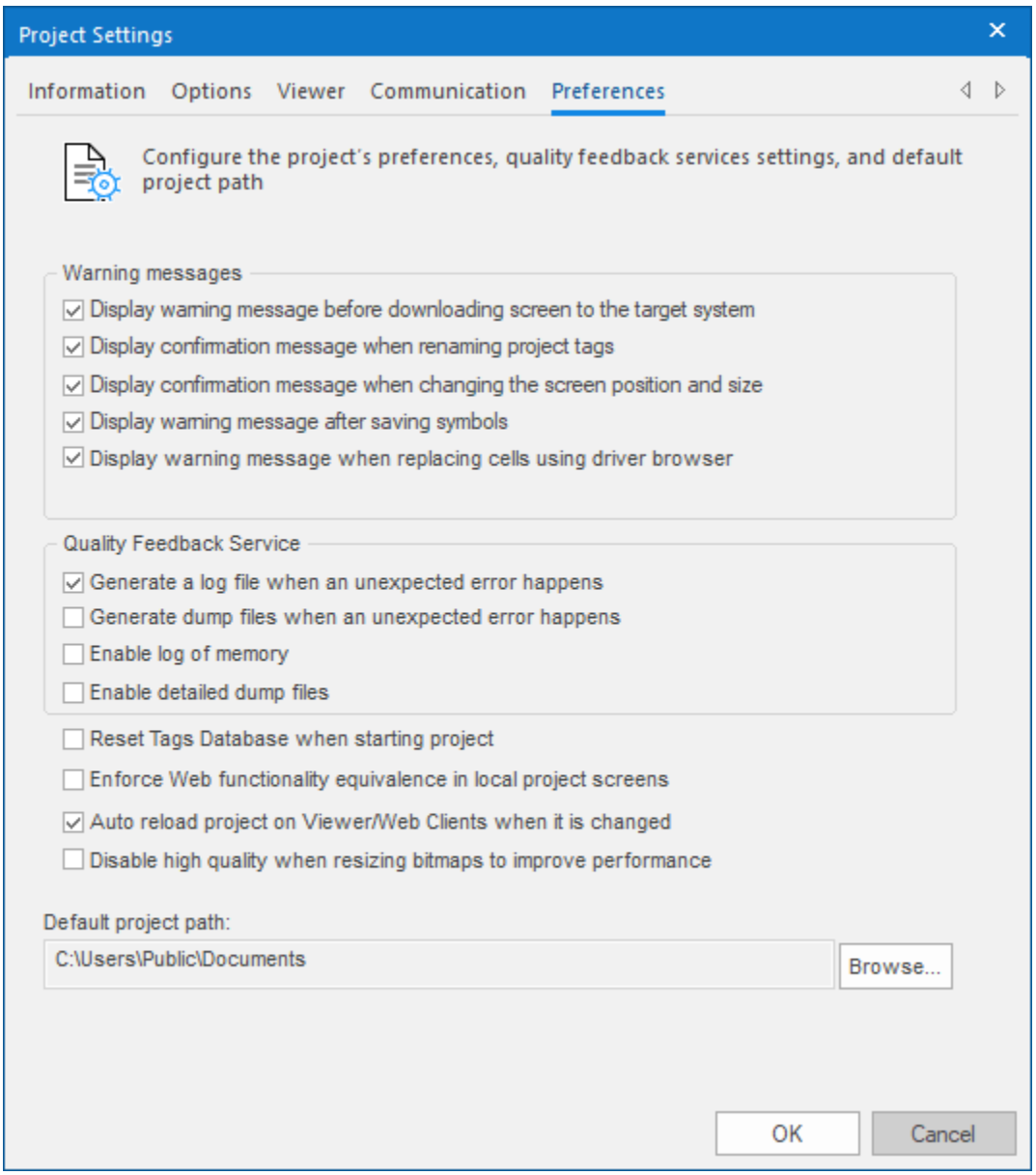
```
C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin
\CertStoreManager.exe
```

When you open the Certificate Store Management program in this way, it will first ask you to locate and open your project file (`<project name>.app`). Otherwise, the program behaves the same as if you opened it from the project settings.

Regardless of how you move the certificate files, you should restart the TCP/IP Server Runtime task and/or the TCP/IP Client Runtime task after you make any changes. This is to make sure rejected clients and servers that were previously trusted are properly disconnected. You do not need to restart the entire project, thanks to the project runtime server's task-based architecture.

### *Preferences tab*

Use this tab to configure your preferences for developing a project in the project development environment.




***Project Settings: Preferences tab***

## Warning Messages

**Display warning message before downloading screen to the target system**

When this option is selected, if you make changes to a screen while the project development environment is actively connected to a target system (via Remote Management), you will be prompted to confirm that you want to download the updated screen to the target system. If this option is cleared, the screen will be downloaded automatically.

**Display confirmation message when renaming project tags**

When this option is selected, if you rename a tag in the Project Tags datasheet, you will be prompted to replace the old tag name with the new tag name in all of the screens and worksheets in your project. It is similar to using the Global Replace tool.

#### Display confirmation message when changing the screen position and size

When this option is selected, if you use the Layout tool to change the position and size of a project screen (i.e., **Width**, **Height**, **Top**, **Left** in the screen attributes), you will be prompted to confirm those changes.

#### Display warning message after saving symbols

When this option is selected, a warning message is shown after saving symbols.

#### Display warning message when replacing cells using driver browser

When this option is selected and you have used tag integration to get tags from a remote device, a warning message will be displayed whenever you replace a local tag in a screen or worksheet with a tag selected from the remote device.

### Quality Feedback Service

This section allows you to configure your project to generate log files and/or dump files that can be used to diagnose hardware and software problems, such as memory leaks and unexpected errors. These files are saved in the Dump sub-folder of the running project at: `<project name>\Web\Dump`

#### Generate a log file when an unexpected error happens

When this option is selected, the runtime modules append the Log File (`\Web\Dump\Dump.txt`) whenever an internal exception (error) occurs. These exceptions may not necessarily crash the runtime modules, but they can affect the stability of the system and should be investigated.

The Log File is continually appended until it reaches its maximum size of 2MB. After it reaches its maximum size, the existing file is deleted and a new file is created.

#### Generate a dump file when an unexpected error happens

When this option is selected, the runtime modules generate a new Dump File (`*.dmp`) with useful information about the conditions of the error. This is a binary file that can only be read by your support representative.

Dump Files are named `WinXXX.dmp` — where **xxx** is an identifying number (in hexadecimal format) automatically generated by the system — in order to prevent an existing file from being overwritten when an new error occurs. Therefore, if more than one error occurs, you will find multiple Dump Files in the directory. The Log File indicates the name of the Dump File associated with each error.

#### Enable log of memory

When this option is selected, the runtime modules append the Log File every 15 minutes with information about the current memory allocation. (The first log entry is written out 15 minutes after the runtime module is started.) This information can be used to identify memory leaks.

#### Enable detailed dump files

When this option is selected, the generated Dump Files will contain more detailed information than they normally do. Please note that these files take much more hard drive space, so you should select this option only when you are working with your support representative to troubleshoot a specific issue.

Even if none of these Quality Feedback options are selected, a post-mortem Dump File (`WinDump.dmp`) will always be generated when the runtime module is terminated by a fatal error. However, for debugging purposes, it is strongly recommend that you enable all options in this section and then send the Log File and all Dump Files to your support representative.

### Other Preferences

#### Reset Tags Database when starting project

When this option is selected, the project tags are automatically reset to their startup values whenever you run your project. For more information, see Reset Tags Database.

#### Enforce Web functionality equivalence in local project screens

When this option is selected, the project development environment will warn you when you try to use features or functions that behave differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. This ensures that your project screens always behave the same regardless of where they are viewed. For example, when this option is selected, the `Open` function cannot be called in Global Procedures, Script worksheets, or other background tasks.

This option is cleared by default in order to maintain compatibility with previous versions of EcoStruxure Machine SCADA Expert 2020. If you do not need to maintain compatibility, you should select this option.

#### Auto reload project on Viewer/Web Clients when it is changed

When this option is selected, project viewers on client stations will periodically check the project runtime server to see if they have the latest version of the project. If they do not, they will automatically reload the project from the server.

This option is not supported in a project running as a Windows service, because that project runs in its own thread separate from the project development environment. After you make your changes to the project, you must first restart the Windows service to include the changes and then restart the project viewers to reload the project from the server.

#### Disable high quality when resizing bitmaps to improve performance

When this option is selected, bitmaps in project screens are resized at lower quality. Normally, when bitmaps are displayed at anything other than actual size, they must be resampled to maintain high quality, and if this resampling must be done frequently and/or for many bitmaps in the same screen, it might decrease run-time performance.

#### Default project path

This is the location where new projects are saved by default when you create them. When you install the full EcoStruxure Machine SCADA Expert 2020 software, the default project path is automatically set to:

**C:\Users\<*user name*>\Documents\EcoStruxure Machine SCADA Expert 2020 Projects**

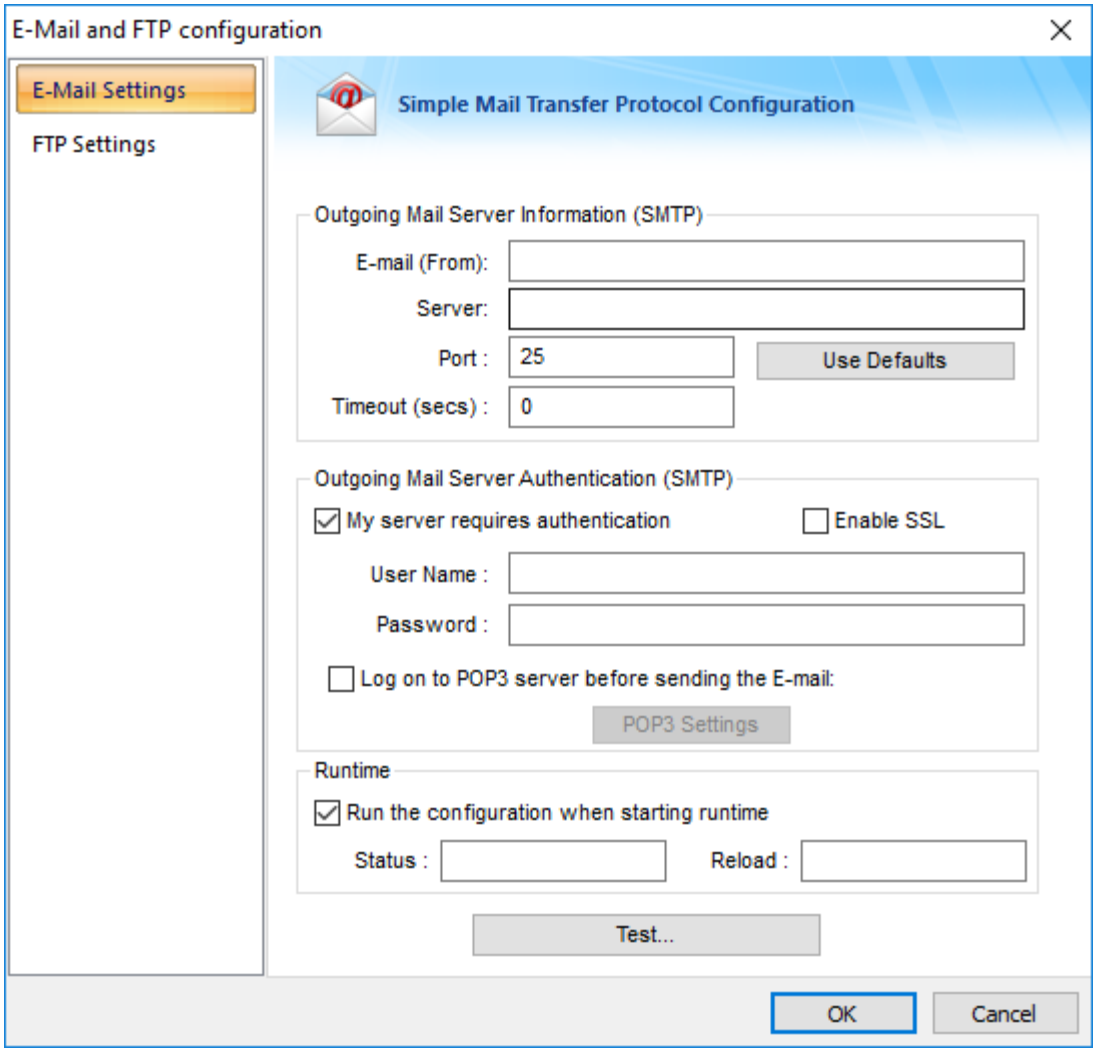
You can subsequently change the default project path to any location on your computer or network.

## Configuring your project's default email settings

Some features, such as alarms and certain functions, are able to send email to designated recipients. To use these features, you must configure your project's email settings.

The email settings can be configured at any point during runtime by calling the `CnfEmail` function. However, you can also configure default settings that are automatically used when the project is first run and then restored as needed during runtime, overwriting any changes made by calling the `CnfEmail` function.

1. On the **Project** tab of the ribbon, in the **Web** group, click **Email/FTP**.
   The *E-mail and FTP configuration* dialog is displayed.
2. Click the **E-Mail Settings** tab.




3. In the **E-mail (From)** box, type your email address.
4. In the **Server** and **Port** boxes, type the server address and port number for your outgoing mail server.
   The default port for SMTP is 25, but it depends on your server and network configuration. Please consult your email provider.
5. If your outgoing mail server requires authentication, select **My server requires authentication**. If authentication must also be encrypted, select **Enable SSL**. Then type your credentials in the **User Name** and **Password** boxes.
   Most outgoing mail servers do require authentication, to prevent spamming and other abuse from unknown users.

> **Note:** Encryption via SSL is not supported in projects running on Windows Embedded target systems.

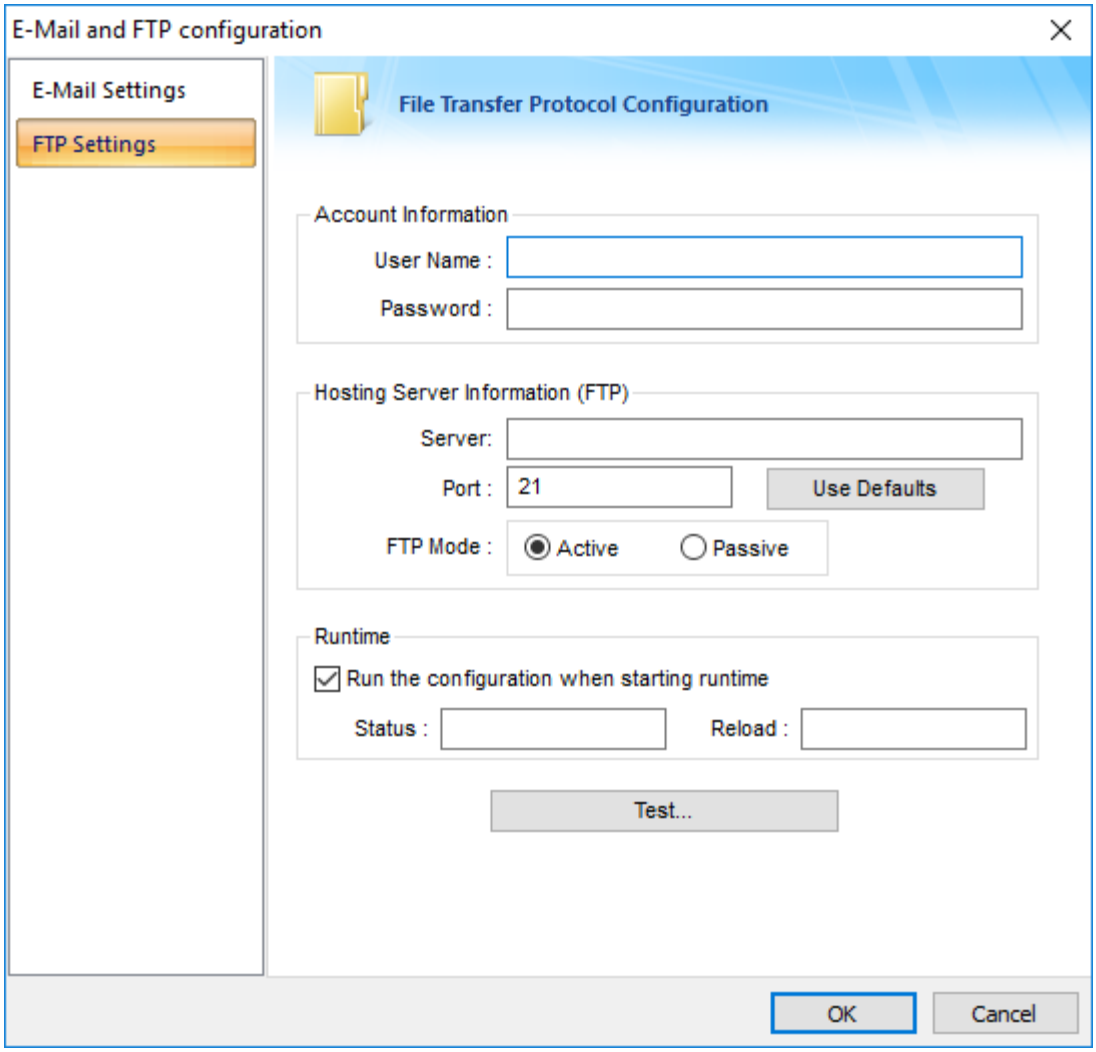
6. In the **Status** box, type the name of a tag (Integer type) that will receive status codes when the project sends email.
7. In the **Reload** box, type a tag/expression. When the value of this tag/expression changes, the project will reload these default email settings.
8. Click **OK** to save your configuration and close the dialog.

## Configuring your project's default FTP settings

Some features in EcoStruxure Machine SCADA Expert 2020, such as certain functions, are able to transfer files between computers using FTP. To use these features, you must configure your project's FTP settings.

The FTP settings can be configured at any point during runtime by calling the `CnfFTP` function. However, you can also configure default settings that are automatically used when the project is first run and then restored as needed during runtime, overwriting any changes made by calling the `CnfFTP` function.

1. On the Project tab of the ribbon, in the Web group, click **Email/FTP**.
   The *Email/FTP Configuration* dialog is displayed.
2. Click the **FTP** tab.



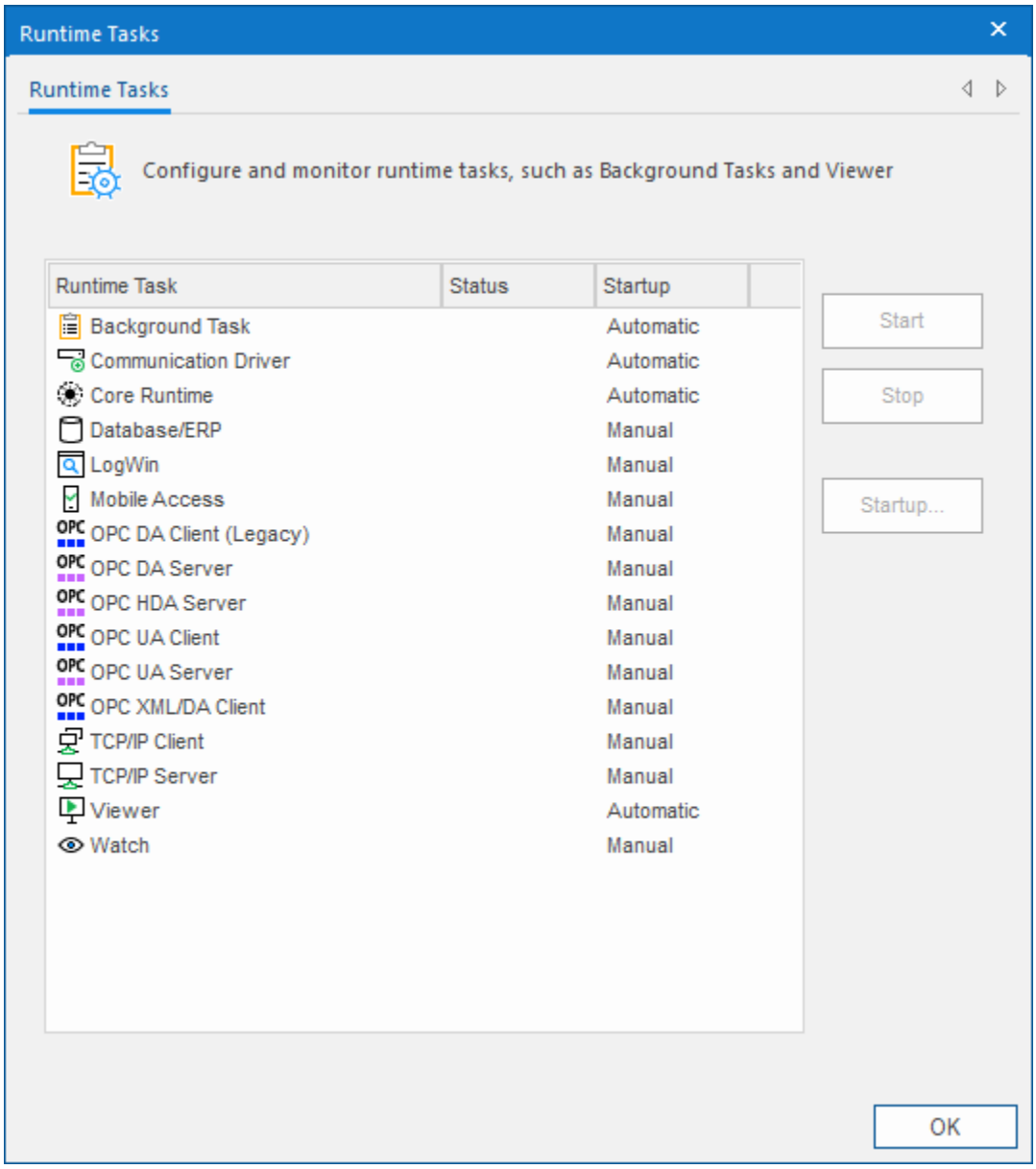

3. In the **User Name** and **Password** boxes, type your credentials for the FTP server.
4. In the **Server** and **Port** boxes, type the server address and port number.
   The default port for FTP is 21, but it depends on your server and network configuration. Please consult your server administrator.
5. Select **Active** or **Passive** mode, depending on the server's configuration.
   Passive FTP mode can be used to bypass some network firewalls. Again, please consult your server administrator.

6. In the **Status** box, type the name of a tag (Integer type) that will receive status codes when the project transfers a file.
7. In the **Reload** box, type a tag/expression. When the value of this tag/expression changes, the project will reload these default FTP settings.
8. Click **OK** to save your configuration and close the dialog.

## Runtime Tasks

Use the **Runtime Tasks** dialog box to configure which Runtime Tasks and runtime modules must be automatically started when the project is run, as well as to manually start and stop tasks during project run time.

The tab lists the available tasks for the current project. Each task's status and startup mode (**Manual** or **Automatic**) is also displayed.




***Runtime Tasks dialog box***

For more information about these tasks and how they interact, see Internal structure and data flow on page 24.

Tasks that are configured as **Manual** must be manually started and stopped. Tasks that are configured as **Automatic** are automatically started when the project is run.

To start or stop a specific task, select that task in the list and then click **Start** or **Stop** on the right. You can also use the `StartTask`, `EndTask`, and `IsTaskRunning` functions to programmatically start and stop tasks.

To change the startup mode of a specific task:

1. Select the task in the list, and then click **Startup** on the right. The *Startup* dialog box is displayed.
2. Select **Manual** or **Automatic** as needed.
3. Click **OK**.

The following table shows which tasks are available on each target platform:

| Task | Windows |
|---|---|
| Background Task | Yes |
| Communication Driver | Yes |
| Core Runtime | Yes |
| Database/ERP | Yes |
| LogWin | Yes |
| Mobile Access | Yes |
| OPC DA Client (Legacy) | Yes |
| OPC DA Server | Yes |
| OPC HDA Server | Yes |
| OPC UA Client | Yes |
| OPC UA Server | Yes |
| OPC XML/DA Client | Yes |
| TCP/IP Client | Yes |
| TCP/IP Server | Yes |
| Viewer | Yes |
| Watch | Yes |

For more information about runtime editions, see About the EcoStruxure Machine SCADA Expert 2020 software components on page 34. For more information about target platforms, see About target platforms, product types, and target systems on page 100.

## Run a project as a Windows service

Your EcoStruxure Machine SCADA Expert project can be configured to run under Windows services.

Microsoft Windows services, formerly known as NT services, allow you to create long-running programs that run in their own Windows sessions. These sessions can be automatically started when the computer starts up, can be paused and restarted, and do not show any user interface. These features make services ideal for use on a server or whenever you need long-running functionality that does not interfere with other users who are working on the same computer. You can also run services in the security context of a specific user account that is different from the logged-on user or the default computer account. For more information about services, please refer to the Microsoft Developer Network (MSDN) Library.

Why would you want to run your project under Windows services?

- To ensure that your project always runs with whatever system privileges it needs, regardless of the privileges of the user that is currently logged on to Windows;
- To prevent the user from interfering with your project while it is running; or
- To let your project keep running when there is no user logged on at all.

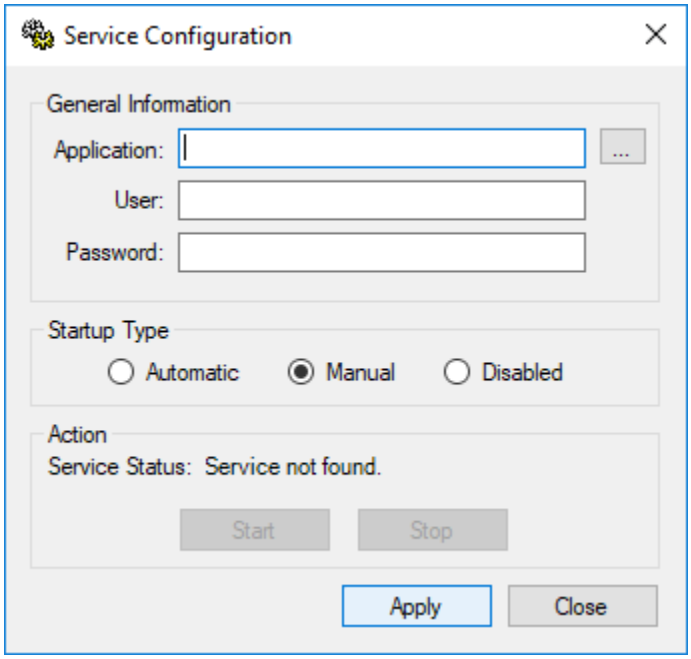
### Create and configure the Windows service

> **Note:** To perform these actions, you must be logged on as a user with Administrator privileges and you should know how to use the *Computer Management* console. (To access the console, right-click the **Computer** icon, and then click **Manage** on the shortcut menu.)

There are two ways to create and configure the Windows service for your project: you can use the Service Configuration tool in the EcoStruxure Machine SCADA Expert development environment, or you can use the command-line utility that is installed with the EcoStruxure Machine SCADA Expert 2020 software.

**Service Configuration Tool**

You can configure and run a new service from within the development environment by clicking **Service** on the Project tab of the ribbon. This opens the *Service Configuration* dialog box:




*Service Configuration dialog box*

**Application**

The location of the project file (*<project name>*.app) that the service will load and run when it is started. This must be a complete file path. Use the browse button (...) to find and select the project file on your computer.

**User**

The Windows user account under which the service will run. This is an optional setting; if it is not used, then the service will run under Local System.

> **Note:** Try to avoid running the service under Local System. That account has too much privilege to the file system and too little privilege to run the OPC Client and Server modules properly. The best alternative is to create a user solely to run EcoStruxure Machine SCADA Expert and configure its privileges to fit the needs of your project. For more about this, see "Configuring User Privileges" below.

**Password**

The password for the specified user account. This is an optional setting; it is not needed if no user is specified or if the specified user does not have a password.

**Startup Type**

How the Windows service will start. The following options are available:

- **Automatic**: The service will start automatically when the computer starts up.
- **Manual**: The service can be started manually in the *Computer Management* console or by clicking **Start**, as described below.
- **Disabled**: The service will be created and then disabled. It cannot run until a user with Administrator privileges enables it in the *Computer Management* console.

**Action pane**

Start or stop the service. Please note that these buttons are not enabled until the service is actually created.

**Creating a New Service**

To create a new service:

1. Next to the **Project** box, click ... to open a standard Windows file browser. Use the browser to find and select your project file.
2. In the **User** and **Password** boxes, type the username and password (if any) for the Windows user account under which the service will run.
3. Select a **Startup Type**.
4. Click **Apply**. The service is created with the specified settings.

After the service has been created, it will appear in the *Services* console (**This PC** > **Control Panel** > **System and Security** > **Administrative Tools** > **Services**) under the name "EcoStruxure Machine SCADA Expert 2020". You can use that console to quickly stop and restart the service, if you do not want to run the EcoStruxure Machine SCADA Expert development application.

**Command-line Utility**

You can also configure the service by using the command-line utility, **StdSvcInst.exe**. It offers a few more options than the Service Configuration tool described above — such as specifying a name and description for the service — and it can be used without running the EcoStruxure Machine SCADA Expert development application. The utility is located in the `Bin` folder of your EcoStruxure Machine SCADA Expert program directory. To run the utility, open a command prompt, navigate to the `Bin` folder (`C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin`), and then enter the command with the desired parameters.

The utility has the following command syntax:

**StdSvcInst { -create -app *filepath* -startup { auto | manual | disabled } -user *username* -password *password* -name *displayname* -descr *description* | -start | -stop | -delete }**

**-create**

Creates the Windows service.

**-app *filepath***

Specifies which project file (*<project name>*.app) the service will load and run when it is started. (This is the same as the **Project** box in the *Service Configuration* dialog.) You must include the complete file path, and it must be enclosed in quotes.

This parameter is required when you create a new service.

**-startup** { **auto | manual | disabled** }

Specifies how the service will start. (This is the same as the **Startup Type** in the *Service Configuration* box.) This parameter is optional; if it is not used, then the default behavior for a new service is manual.

**-user *username***

Specifies the Window user account under which the service will run. (This is the same as the **User** box in the *Service Configuration* dialog.) This parameter is optional; if it is not used, then the service will run under Local System.

**-password *password***

Specifies the password for the given user account. (This is the same as the **Password** box in the *Service Configuration* dialog described above.) This parameter is optional; it is not needed if no user is specified or if the specified user does not have a password.

**-name *displayname***

Defines the service name that is displayed in the *Computer Management* console. The name must be enclosed in quotes. This parameter is optional; the default name is "Studio".

**-descr *description***

Defines the service description that is displayed in the *Computer Management* console. The description must be enclosed in quotes. This parameter is optional.

**-start**

Starts the service. This is the same as starting the service using the *Computer Management* console or by clicking **Start** in the *Service Configuration* dialog.

**-stop**

Stops the service. This is the same as stopping the service using the *Computer Management* console or by clicking **Stop** in the *Service Configuration* dialog.

**-delete**

Deletes the service.

**Example: Creating the Service**

In this example, we want to create a new Windows service with the following options:

| | |
|---|---|
| EcoStruxure Machine SCADA Expert Project File | C:\Users*<user name>*\Documents\EcoStruxure Machine SCADA Expert 2020 Projects*<project name>**<project name>*.app |
| Startup Mode | Automatic |
| User | EcoStruxure Machine SCADA Expert |
| Password | EcoStruxure Machine SCADA Expert |

| | |
|---|---|
| Service Name | "EcoStruxure Machine SCADA Expert 2020" |
| Service Description | "Starts EcoStruxure Machine SCADA Expert project" |

Note that the system must already have a user account named "EcoStruxure Machine SCADA Expert" with password "EcoStruxure Machine SCADA Expert".

So, to create the service with the desired options:

1. Make sure you are logged on as a user with Administrator privileges.
2. Open a command prompt (**Start** > **Windows System** > **Command Prompt**).
3. Navigate to the `Bin` folder:

```
cd "C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
 2020\Bin"
```

4. Enter the command:

```
StdSvcInst –create –app "C:\Users\<user name>\Documents\EcoStruxure Machine
 SCADA Expert 2020 Projects\<project name>\<project name>.app" -startup auto –
user EcoStruxure Machine SCADA Expert -password EcoStruxure Machine SCADA Expert –
name "EcoStruxure Machine SCADA Expert 2020" –descr "Starts EcoStruxure Machine SCADA
 Expert project"
```

If the procedure is successful, then the system will display the message `Service created`. Otherwise, it will display an error message.

**Example: Changing the Project File**

After you create the service, you may want to change the EcoStruxure Machine SCADA Expert project file that it runs. You can do this by using the -app parameter:

1. Make sure you are logged on as a user with Administrator privileges.
2. Stop the service if it is running.
3. Open a command prompt.
4. Navigate to the `Bin` folder.
5. Enter the command — for example, to set MyProject as the project file:

```
StdSvcInst –app "C:\Users\<user name>\Documents\EcoStruxure Machine SCADA Expert 2020
 Projects\MyProject\MyProject.app"
```

**Example: Deleting the Service**

To delete the service:

1. Make sure you are logged on as a user with Administrator privileges.
2. Stop the service if it is running.
3. Open a command prompt.
4. Navigate to the `Bin` folder.
5. Enter the command:

```
StdSvcInst –delete
```

### Configure user privileges

The service will run under the privileges of the user account specified in the **User** field of the *Service Configuration* tool (or by the -user parameter of the command-line utility). If EcoStruxure Machine SCADA Expert needs some system resource to which that account does not have privileges, it will fail. Therefore, you must configure the account to have the necessary privileges.

> **Note:** The following actions can be performed only by a user with Administrator privileges.

**Enabling the User Account to Log On as a Service**

Before anything else, the specified user account must be enabled to log on to the computer as a service. To enable the account:

1. Open the *Local Security Settings* console (**This PC** > **Control Panel** > **System and Security** > **Administrative Tools** > **Local Security Policy**).
2. In the console window, select the **User Rights Assignment** folder (**Security Settings** > **Local Policies** > **User Rights Assignment**).
3. In the list of policies, double-click **Log on as a service**.

   The *Log on as a service* dialog box is displayed.
4. Click **Add User or Group**.

   The *Select Users or Groups* dialog is displayed.
5. Type the name of the user account under which you want the service to run.
6. Click **OK**.

**Giving the User Account Full Control Over the Project Folder**

For your EcoStruxure Machine SCADA Expert project to run properly, the specified user account must have full control over the project folder and all of the files in it. To give the account those privileges:

1. In *Windows Explorer*, locate your EcoStruxure Machine SCADA Expert project folder (i.e., the folder that contains the file `<project name>.APP`).
2. Right-click the folder, and then click **Properties** on the shortcut menu.
3. In the properties sheet, click the **Security** tab, and then click **Edit**.
4. In the *Permissions* dialog box, click **Add**, and then add the user account that you specified when you created the service.
5. Select the user that you added, and then in the list of permissions, set **Full Control** to **Allow**.
6. Click **OK** to apply your changes and close the dialog, and then click **OK** again to close the properties sheet.

**Allowing the User Account to Run the OPC Client/Server Module**

As mentioned previously, normal users have too few privileges to properly run the OPC Client/Server module. Therefore, you must configure the user account to have those privileges:

1. Open the *Component Services* console (**This PC** > **Control Panel** > **System and Security** > **Administrative Tools** > **Component Services**).
2. In the console window, select the **DCOM Config** folder (**Console Root** > **Component Services** > **Computers** > **My Computer** > **DCOM Config**).
3. In the *DCOM Config* pane, right-click **Studio Scada OPC Server**, and then click **Properties** on the shortcut menu.
4. In the properties sheet, click the **Identity** tab.
5. Select **This user** and then complete the fields with the same username and password that you specified when you created the service.
6. Click **OK** to apply your changes and close the properties sheet.
7. Close the *Component Services* console.

## Troubleshooting

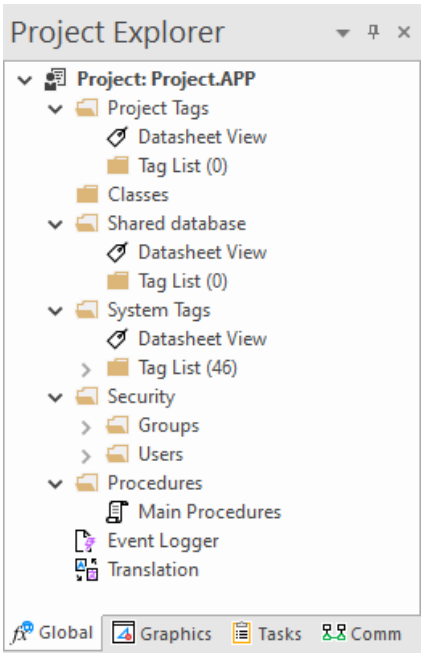
When you run your EcoStruxure Machine SCADA Expert project as a Windows service, it has no user interface. Therefore, if an error occurs, it will only be logged as a Windows application event. You can check the messages by using the *Event Viewer* console (**This PC** > **Control Panel** > **System and Security** > **Administrative Tools** > **Event Viewer**).

# Tags and the Project Database

## About Tags and the Project Database

Tags are a core component of any project. Simply put, project tags are variables used to receive and store data obtained from communication with plant floor devices, from the results of calculations and functions, and from user input. In turn, tags can be used to display information on screens (and Web pages), to manipulate screen objects, and to control runtime tasks.

But tags are more than simple variables. The project runtime includes a real-time database manager that provides a number of sophisticated functions such as time-stamping of any value change, checking tag values against runtime minimum and maximum values, comparing tag values to alarming limits, and so on. A project tag has both a value and various properties that can be accessed, some at development and others only at runtime.




All tags are organized into one of the following categories, which are represented by folders on the **Global** tab of the *Project Explorer*:

- **Project Tags** are tags that you create during project development. Places where project tags are used include:
  - Screen tags
  - Tags that read from/write to field equipment
  - Control tags
  - Auxiliary tags used to perform mathematical calculations
- **Shared Database** tags are created in a PC-based control program and then imported into the tags database.

  For example you might create tags in SteepleChase and import them into your project so that it can read/write data from a SteepleChase PC-based control product.

  You cannot modify shared tags within your project — you must modify the tags in the original PC-based control program, and then re-import them into the tags database.
- **System Tags** are predefined tags with predetermined functions that are used for supervisory tasks during project run time. For example,
  - Date tags hold the current date in string format

- Time tags hold the current time in string format

Most system tags are *read-only*, which means you cannot add, edit, or remove these tags from the database.

To see a list of the system tags, select the **Global** tab in the *Project Explorer*, open the **System Tags** folder, and open the **Tag List** subfolder.

After creating a tag, you can use it anywhere within the project, and you can use the same tag for more than one object or attribute.

## Project Tags Folder

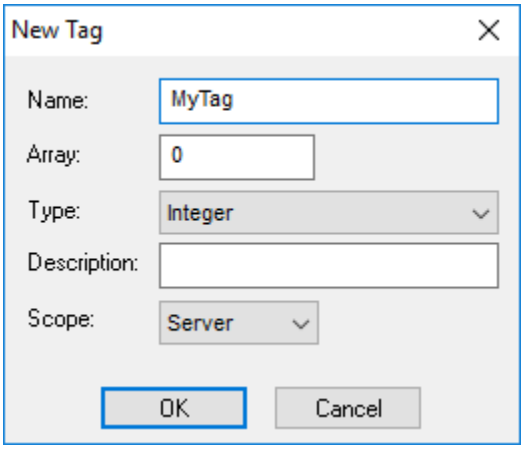
The *Project Tags* folder contains all tags created and customized by the user. You can create project tags for displays, to read from and write to field equipment, for control, to perform mathematical calculations, and so forth.

To update a list of project tags, right-click on the *Project Tags* folder (or **Datasheet View** icon ) and select the **Refresh** option.

> **Important:** Before deleting a tag, we strongly recommend using the **Object Finder** tool (on the Home tab of the ribbon) to verify that you are not using the tag in another part of the project (screens, math sheets, so forth). If you delete a tag from the project database that is being used in another part of the project, you will cause a compiling error and the project will function poorly.

To create a new tag, right-click on the *Project Tags* folder, the *Tag List* sub-folder, or **Datasheet View** icon and select **Insert Tag** from the shortcut menu. You also can click **Tag** on the Insert tab of the ribbon.

The *New Tag* dialog displays, as shown in the following figure:




***New Tag dialog***

Use this dialog to specify the following parameters:

- **Name** field: Type a name for the new tag. The first character must be a letter and you can use up to 255 characters in the name.
- **Array Size** field: Type a value to specify the size of the tag. Any size greater than 0 implies that the tag is an array.
- **Type** combo-box: Select a standard tag type from the list (**Boolean**, **Integer**, **Real**, or **String**). You also can define new types as structures formed by the classes.
- **Description** text box: Type a tag description for documentation purposes.
- **Scope** combo-box: Click to select one of the following options:

- **Server** (default): The tag is maintained on the project server, and it is shared by all connected thin clients. A change to the tag value affects the entire project.
- **Local**: A virtual copy of the tag is maintained separately on each local station (server + clients), and a change to the tag value affects only the station on which the change was made.

These options have no affect on projects that do not have Web capabilities. If you select a **Scope** option for a project with Web capabilities, then any object property using the Local tag will not work properly over the Web.

> **Note:** You must create unique tag names. You cannot create a tag that uses the name of an existing tag.

You can view or edit the properties of a tag from either of the following dialoges:

- *Tag Property* dialog: Click **Properties** on the Home tab of the ribbon when the tag name displays in the **Tag name** field or double-click on the tag name in the *Tag List* subfolder located in the *Project Tags* folder.
- *Project Tags* dialog: Click the **Datasheet View** icon in the *Project Tags* folder.

The *Project Tags* datasheet includes columns for many of the tag properties.

Project Tags

| | Name | ray Si: | Type | Description | Scope | UA External Availability |
|---|---|---|---|---|---|---|
| | Filter text | F... | (All) | Filter text | (All) | (All) |
| 1 | startup | 0 | Boolean | | Server | Disabled |
| 2 | SysOS | 0 | String | | Local | Disabled |
| 3 | SysProdVer | 0 | String | | Local | Disabled |
| 4 | SysCompName | 0 | String | | Local | Disabled |
| 5 | SysIPAddress | 0 | String | | Local | Disabled |
| 6 | TestABC | 0 | Integer | | Server | Disabled |
| 7 | TagA | 5 | Integer | | Server | Disabled |
| 8 | TagMin | 0 | Integer | | Server | Disabled |
| 9 | TagMax | 0 | Integer | | Server | Disabled |
| 10 | SimulAnalog | 0 | Real | | Server | Disabled |
| * | | | Integer | | Server | Disabled |
| * | | | Integer | | Server | Disabled |
| * | | | Integer | | Server | Disabled |
| * | | | Integer | | Server | Disabled |
| * | | | Integer | | Server | Disabled |

***Project Tags datasheet***

Use this dialog to create, modify, or delete tags or tag properties. You can right-click on a tag property and use standard Windows commands to cut (Ctrl+X), copy (Ctrl+C), or paste (Ctrl+V), any tag and its properties. You can also undo (Ctrl+Z) the last modification to a field.

> **Tip:** You can sort the data in the *Project Tags* sheet and/or insert/remove additional columns to/from the sheet by right-clicking on it and choosing the applicable option from the shortcut menu.

## SET TAG PROPERTIES USING THE PROJECT TAGS DATASHEET

Use the *Project Tags* datasheet to set the properties of project tags.

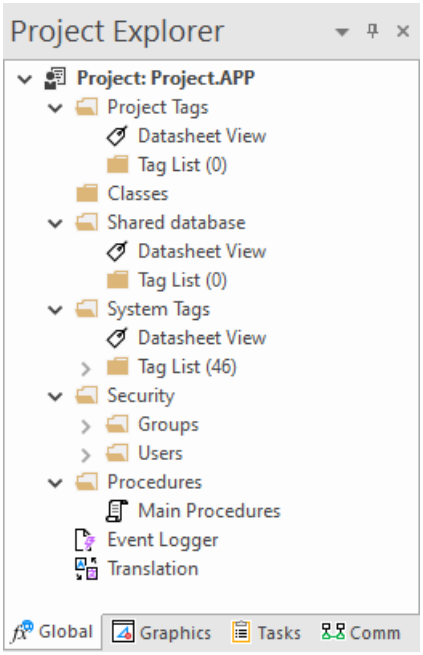
The datasheet is essentially a spreadsheet that lists all of the tags (not including shared and system tags) that are in your project database, as well as certain properties of those tags. You can change which tag properties are included in the spreadsheet by showing or hiding additional columns, and you can set the properties of specific tags by entering new values in the appropriate cells.

> **Note:** You cannot use the *Project Tags* datasheet during run time. If you need to edit the tags database during run time, use the Tags Database functions.

To use the *Project Tags* datasheet to set tag properties:

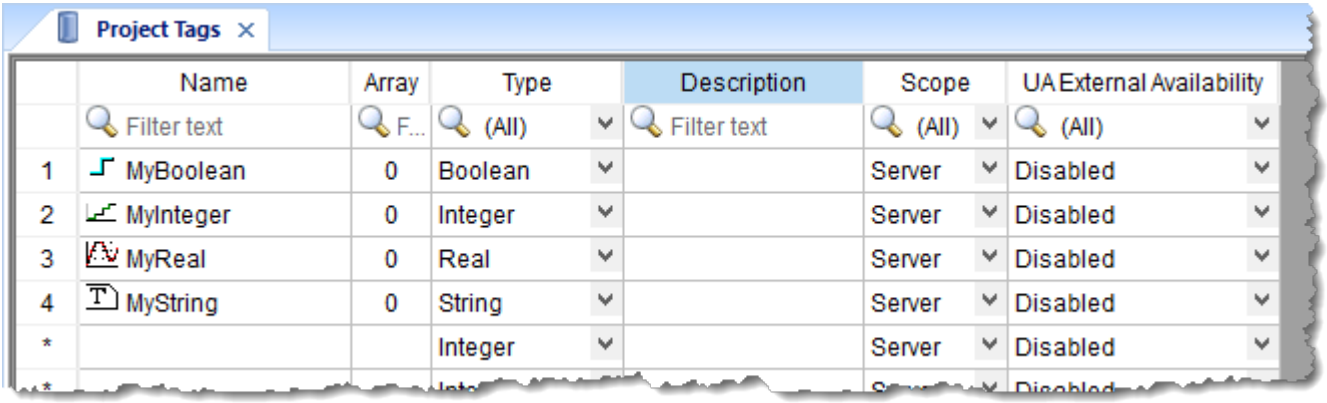
1. At the bottom of the *Project Explorer* window, click **Global**.

   The **Global** tab is displayed.




2. Expand the **Project Tags** folder, and then in the folder, double-click **Datasheet View**.

   The *Project Tags* datasheet is opened in the screen/worksheet editor.



3. To show/hide additional columns for other tag properties, right-click anywhere in the datasheet, and then on the shortcut menu, click the desired properties:
   - **Name** (cannot be hidden)
   - **Size** (shown by default)
   - **Type** (shown by default)
   - **Description** (shown by default)
   - **Scope** (shown by default)
   - **More Columns** > **Startup**
   - **More Columns** > **Min**
   - **More Columns** > **Max**
   - **More Columns** > **Unit**
   - **More Columns** > **Retentive Value**
   - **More Columns** > **Retentive Parameters**
   - **More Columns** > **Dead Band**
   - **More Columns** > **Smoothing**
   - **More Columns** > **UA External Availability** (shown by default)

   Each row of the datasheet represents a project tag, and each column of the datasheet represents a property of that tag.
4. To set a tag property, enter the new value in the appropriate cell. Repeat as needed.

   Some properties do not apply to all data types, so for more information about the applicable properties, see:
   - Properties of Integer and Real tags on page 171
   - Properties of Boolean tags on page 176
   - Properties of String tags on page 180
5. When you are done, save and close the datasheet.

## EXTENDING THE PROJECT TAGS DATASHEET

The Project Tags worksheet can be extended up to 65,488 rows, if necessary.

The datasheet is normally limited to a maximum of 32,721 rows. (This is separate from the maximum size of the project database as a whole, as well as the runtime limit that is set when you select a target platform for a new project.)

To extend the worksheet, edit your project file (*<project name>*.app) to include the following entry:

```
[Options]
EnableExtendedTagCount=1
```

Doing so, however, brings the following restrictions:

- Project tags in rows 32,722 through 65,488 of the worksheet cannot be used as array indices in expressions. That is, in an expression like **Abs(numArray[indexTag])**, **indexTag** cannot be in that range of rows. (This restriction does not apply to the VBScript interface.)
- In a Class worksheet, only the first 32 class members can have alarms. For all class members after the first 32, alarms will not work.

Generally speaking, extending the Project Tags datasheet stretches the capabilities of this software and should be done only when it is absolutely necessary. It is better to design your project to conserve tags.

## *About classes*

*Class* tags are compound tags that permit a high-degree of encapsulation within the Tags database. Where basic tags receive a single value, *classes* are designed to receive multiple values.

You can create a class-type tag by grouping basic or array tags, which then become the class *members*. The maximum number of members for any class depends on the product specification.

You specify class-type tags in one of two formats:

- For a simple class tag the syntax is ***TagName.ClassMemberName***. (Where the period is used as a separator.)

  For example, if you wanted to monitor several different conditions (such as *temperature*, *level* and *pressure*) in a tank, you might create a class tag as follows:

  - **Tank.Temperature**
  - **Tank.Level**
  - **Tank.Pressure**

- For creating a complex class tag (using an array tag) the syntax is ***ArrayTagName*[*ArrayIndex*].*ClassMemberName***. (Where again, the period is used as a separator.)

  If you wanted to monitor the *temperature*, *level*, and *pressure* conditions in multiple tanks, you might create a class tag as follows:

  - **Tank[tk].Temperature**
  - **Tank[tk].Level**
  - **Tank[tk].Pressure**

  Where **tk** is an array index, representing the tank number.

### Classes Folder

The *Classes* folder contains all of the project classes and their respective members. Classes are compound tags consisting of user-defined data-type structures or tag types (*Integer*, *Real*, *Boolean*, and *String*). Classes allow for high-level encapsulation in the project database. A class-type tag provides a set of values for its members.

To define a class you must define the *members* and their types. Class members are variables that hold values for an object with particular characteristics. Thus, the defining a class can be very useful for projects with a repeating group of variables.

> **Note:** When you create a class folder, a **Class** icon displays in the *Tag List* subfolder located in the *Project Tags* folder.

To access the members of a class, use the following syntax with a period ( . ) as the separator: ***TagName.MemberName***. For example: **tk.LEV** or **tk.TMP**.

If the **Tank** tag is an array, you use the following syntax:

***ArrayTagName*[*ArrayIndex*].*MemberName***

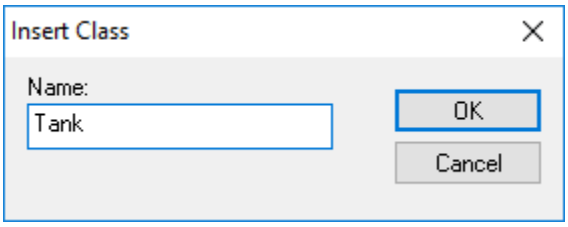
For example: **Tank[1].Level** or **Tank[n].Temperature**

A class-type tag contains a set of values (rather than a single value) associated with the class. You create class-type tags by grouping simple tags, which become the members. The maximum number of members for any class depends on the product specification. Class members can hold standard *Integer*, *Real*, *Boolean*, and *String* values, as mentioned previously.

To create a new class, use one of the following methods to open the *Insert Class* dialog:

- On the Insert tab of the ribbon, in the Global group, click **Class**;
- Right-click on the *Classes* folder, the *Members List* sub-folder, or the **Datasheet View** icon in the *Classes* folder; or
- Create a new class tag in the *Project Tags* folder.

When the *Insert Class* dialog displays, enter a class name in the **Name** field, and then click **OK** to close the dialog.

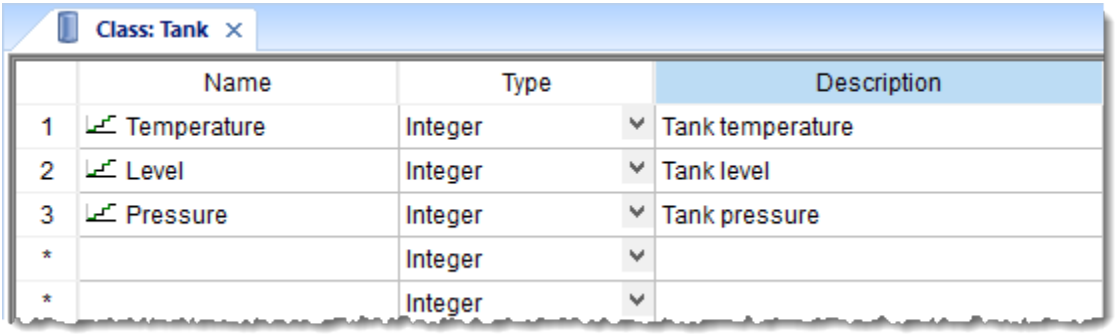



***Insert Class dialog***

> **Note:** You must type a unique class name. You cannot create two classes with the same name. In addition, you cannot configure shared tags and system tags as classes.

EcoStruxure Machine SCADA Expert saves class folders in the *Tag List* subfolder (located in the *Project Tags* folder). You can edit the classes in this folder.

When the *Class* datasheet displays, you can use it to create, modify, or delete any class members and their viewable properties. (You cannot edit classes from the *Tag Property* dialog.)




***Class datasheet***

> **Note:** The Classes folder can contain up to 16,384 classes and up to 4,096 members per class. This is a technical limitation of the tags database, not a licensing restriction; unused classes do not count against the total number of tags used.
>
> When a new tag is created with a class type, however, each class member counts as a tag used because each member holds a value. (For example, if you create a class with 5 members and then create 5 tags with that class type, then you have a total of 25 tags used.) The total number of tags used cannot exceed the number of tags supported by the project's target system / runtime license.

To edit a class member or property, you can right-click on the item and use standard Windows commands to cut (Ctrl+X), copy (Ctrl+C), or paste (Ctrl+V). You can also undo (Ctrl+Z) the last modification to a field.

You also edit member properties as follows:

- **Name** field: Type a name for the member or member property. The first character must be a letter and you can use up to 255 characters in the name.
- **Type** combo-box: Select a member type (*Boolean*, *Integer*, *Real*, or *String*).
- **Description** field: Type a description of the member property for documentation purposes.

> **Note:** Members of a class cannot be of another class type.

Also, you must create a unique class name. You cannot reuse the name of an existing class. However, you can create members with the same name in different classes.

To delete a class and all its members, right-click on a class folder and select delete. EcoStruxure Machine SCADA Expert disables the delete option if you are running any runtime tasks. In addition, you cannot delete a class if it is associated with any tag.

### *Shared Database folder*

The **Shared Database** folder shows the tags that you have added to your project through tag integration.

The folder is located on the **Global** tab of the *Project Explorer*. It provides both a Datasheet View and a Tags List similar to the **Project Tags** folder, but you cannot use them to edit the properties of integrated tags. You can only read and write actual tag values during run time. If you want to edit the properties of integrated tags, you must use the appropriate programming software to do it on the source device(s).

Integrated tags are automatically and continuously updated as long as the project runtime server remains connected to the source device(s). You can use these tags the same as you would use normal project tags that you created in EcoStruxure Machine SCADA Expert 2020; it is not necessary to configure an OPC or Driver worksheet that associates project tags with device registers.

The **Shared Database** folder only shows the integrated tags that you have already added to your project. If you have not set up any tag integration sources and then used the Object Finder to select specific tags, the folder will be empty.

For more information, see Tag Integration on page 225.

### *System Tags Folder*

The *System Tags* folder contains predefined tags that have specific functions (time, date, acknowledge alarms, storage of the logged user, and so forth). You cannot edit or delete these tags; but you can access their values from any EcoStruxure Machine SCADA Expert task, copy them, and use them elsewhere.

**Note:** To update EcoStruxure Machine SCADA Expert's shared database with the system tags files, right-click on the *System Tags* folder or **Datasheet View** icon, and then click the **Refresh** option.

For a list of system tags, including their properties and descriptions, see List of System Tags.

You can view the properties of a system tag using the *System Tags* datasheet, which contains four columns (*Name*, *Size*, *Type*, and *Description*).

**Important:** Most system tags are read-only. To change the time, for example, you must use the proper math function and set the system time rather than writing to the system time tag.

## LIST OF SYSTEM TAGS

This is a list of the system tags that are available in all projects.

### List of system tags

| Tag Name | Data Type | Description | Scope |
|---|---|---|---|
| AckAll | Boolean | Toggle to **1** to acknowledge all alarms. [1] | Server |
| AckAlr | Boolean | Toggle to **1** to acknowledge the highest priority unacknowledged alarm; see **Alarm** below. [1] | Server |
| Alarm | String | The name of highest priority unacknowledged alarm. [1] | Server |
| AnalogValue_ | Real | A simulated analog value that steadily increases from **0** to **96** at the rate of 1.6/sec. When it reaches **96**, it starts over at **0**. | Local |
| Beep | Boolean | The current state of the audible alarm (i.e., the "beep"): **0** = OFF, **1** = ON. [1] | Local |
| BeepOff | Boolean | Toggle to **1** to mute the audible alarm (i.e., the "beep"). [1]<br>This does not acknowledge or disable the alarm itself, and this does not toggle the value of the **Beep** system tag. This only suppresses the sound of | Local |

| Tag Name | Data Type | Description | Scope |
|---|---|---|---|
| | | the alarm on the local station. If you want to disable the audible alarm on all stations, see Customize the audible alarm on page 411. | |
| BlinkFast | Boolean | Toggles every 200 milliseconds by default.<br>To adjust this period, edit the program settings file (`Program Settings.INI`) in order to change the **`BlinkFast`** setting. | Local |
| BlinkSlow | Boolean | Toggles every 600 milliseconds by default.<br>To adjust this period, edit the program settings file (`Program Settings.INI`) in order to change the **`BlinkSlow`** setting. | Local |
| CrispDisplay | Integer | Legacy tags used by the CrispView custom interface. They are kept in EcoStruxure Machine SCADA Expert only for backward compatibility and should not be used in new projects. For more information, please contact Technical Support. | Local |
| CrispInput | String | | Local |
| CrispOutput | String | | Local |
| Date | String | The current date on the local station, formatted according to the station's current date format (e.g., `05/15/2008`). For more information about the date format, see About the date format and how to change it on page 694.<br>If this tag is referenced on a project thin client, its value might be different from the current date on the project runtime server. See **`ServerDate_`** below. | Local |
| Day | Integer | The current day of the month, between **`1`** and **`31`**. | Local |
| DayOfYear | Integer | The current day of the year, between **`0`** and **`365`**. | Local |
| DigitalValue_ | Boolean | A simulated digital value that alternates between **`0`** and **`1`** each second. | Local |
| Goto | String | The start point of a Goto…Label structure. For more information, see Using the Goto…Label structure in a Math worksheet on page 500. | Local |
| GroupCNFHiLevel | Integer | The end of current security level range to CNF. [3] | Local |
| GroupCNFLoLevel | Integer | The start of current security level range to CNF. [3] | Local |
| GroupHiLevel | Integer | The maximum security level of the current user's group. [2 3] | Local |
| GroupLoLevel | Integer | The minimum security level of the current user's group. [2 3] | Local |
| GroupName | String | The name of the Group to which the current User belongs. If the user belongs to more than one group, this tag will have all the group names separated by comma. [2] | Local |
| Hint | String | The hint text for the current object. For more information, see Object Properties. | Local |
| Hour | Integer | The current hour of the day, between **`0`** and **`23`**. | Local |
| InputMaxRange | Real | The maximum value (Max) for the current tag or object; see **`InputOutOfRange`** below. | Local |
| InputMinRange | Real | The minimum value (Min) for the current tag or object; see **`InputOutOfRange`** below. | Local |
| InputOutOfRange | Boolean | This tag toggles to **`1`** when the user's input is outside the Max/Min range of the current tag or object. | Local |
| Label | String | The end point of a Goto…Label structure. For more information, see Using the Goto…Label structure in a Math worksheet on page 500. | Local |
| LastCodeChar_ | Integer | Last code char in the Viewer (???) | Local |
| LptOff | Boolean | Toggle to **`1`** to suppress the printing of alarms to the default printer. [1] | Local |
| Minute | Integer | The current minute of the hour, between **`0`** and **`59`**. | Local |
| Month | Integer | The current month of the year, between **`1`** and **`12`**. | Local |
| Next | Integer | The end point of a For…Next loop. For more information, see Using the For…Next loop in a Math worksheet on page 502. | Local |
| Reserved___1 | Boolean | Reserved for future use. | Local |

| Tag Name | Data Type | Description | Scope |
|---|---|---|---|
| Reserved___2 | Boolean | Reserved for future use. | Local |
| Reserved___8 | Boolean | Reserved for future use. | Local |
| Reserved___9 | Boolean | Reserved for future use. | Local |
| Reserved___10 | Boolean | Reserved for future use. | Local |
| Second | Integer | The current second of the minute, between **0** and **59**. | Local |
| ServerDate_ | String | The current date on the project runtime server, formatted according to the server's current date format (e.g., `05/15/2008`). For more information about the date format, see About the date format and how to change it on page 694. | Server |
| ServerTime_ | String | The current time on the project runtime server, presented as a string in HH:MM:SS format. For example, `23:45:03`. | Server |
| Time | String | The current time on the local station, presented as a string in HH:MM:SS format. For example, `23:45:03`. (This may be different from the current time on the project runtime server; see **`ServerTime_`** above.) | Local |
| Tomorrow | Integer | The next day of the month, between **1** and **31**. | Local |
| UserName | String | The name of the User that is currently logged in. [2] | Local |
| Weekday | Integer | The current day of the week on the local station, presented as an integer: **0** = Sunday, **1** = Monday, **2** = Tuesday, **3** = Wednesday, **4** = Thursday, **5** = Friday, **6** = Saturday. | Local |
| Year | Integer | The current, four-digit year (e.g., **2008**). | Local |
| Yesterday | Integer | The previous day of the month, between **1** and **31**. | Local |

[1] For more information about configuring alarms, see *Alarms* folder.

[2] For more information about configuring Users, Groups, and Security Levels, see Project Security on page 639.

[3] This system tag has been deprecated and is available only for backward compatibility. If the user belongs to more than one group, this tag has a value of `-1`. Use the function `CheckSecurityLevel` instead.

# Designing a Tag

## *Naming the Tag*

Observe the following guidelines when you name a tag or class member:

- Each name must be unique — you cannot specify the same name as another user-created tag or class member, an imported tag, a system tag, or a built-in function. If you enter an existing name, the project development environment will recognize that name and it will not prompt you to create a new tag.
- The name can be composed of uppercase and lowercase letters (**`A–Z`**, **`a–z`**), the accented forms of those letters (e.g., **`é`**, **`ü`**, **`ç`**), standard numerals (**`0–9`**), and the underscore character (_). All other punctuation, special characters, mathematical symbols, and non-Latin alphabets are not allowed.
- The name must begin with a letter.
- The name can be up to 255 characters long.
- Even though the name can be composed of both uppercase and lowercase letters, it is not actually case-sensitive. It will be recognized as long as it is spelled correctly. Therefore, you can use uppercase and lowercase letters to make the name more readable to you. For example, **`TankLevel`** and **`tanklevel`** both refer to the same tag.

> 👉 **Tip:** To indicate a tag will be used as an indirect tag, insert the "at" sign (**`@`**) at the beginning of the tag name.

## *Choosing the Tag Type*

EcoStruxure Machine SCADA Expert allows you to create the following types of tags:

- **Basic tags** hold a single value.
- **Array tags** are a set of tags that use the same name with unique indexes.
- **Class tags** are a set of compound tags that consist of user-defined data types (Boolean, Integer, Real or String) or data-type structures.
- **Indirect tags** are pointers that provide indirect access to another tag type, including class tags.

A discussion of these tag types follows.

### Basic Tags

A *basic* tag receives a single value. Typically, most tags defined for a project are basic tags. Some examples of a basic tag include:

- **`TankID`** (to identify different tanks in your project)
- **`Temperature`** (to identify the current temperature of an object)
- **`Status`** (to identify whether an object is open or closed)

### Array Tags

An *array* tag consists of a set of tags that all have the same name, but use unique array indexes (a matrix of *n* lines and one column) to differentiate between each tag. An *array index*can be a fixed value, another tag or an expression. Maximum array sizes are determined by product specifications.

You can use array tags to:

- Simplify configurations
- Enable multiplexing in screens, recipes, and communication interfaces
- Save development time during tag declaration

You specify array tags in one of two formats:

- For a simple array tag, type:

  ***ArrayTagName*[*ArrayIndex*]**

- For a complex array tag (where the array index is an expression consisting of a tag and an arithmetic operation), type:

  ***ArrayTagName*[*ArrayIndex*+*c*]**

  Where:

  - ***ArrayTagName*** is the tag name;
  - **[*ArrayIndex*]** is the unique index (fixed value or another tag);
  - **+** is an arithmetic operation; and
  - ***c*** is a numerical constant.

> **Note:**
>
> - You must specify a maximum index for each array tag by typing a value (*n*) in the Array Size column of an *Project Tags* datasheet or in the Array Size field on a *New Tag* dialog. (See "Creating project database Tags").
>
>  When you create an *n*-position array tag, EcoStruxure Machine SCADA Expert actually creates ***n*+1** positions (from 0 to *n*). For example, if you specify **ArrayTag[15]**, the array will have 16 elements, where 0 is the start position and 15 is the end position.
>
> - You must not use spaces in an array tag.
>
>  When EcoStruxure Machine SCADA Expert reads a tag it begins with the first character and continues until it finds the first space or null character. Consequently, the system does not recognize any characters following the space as part of the array tag.
>
>  For example, if you type **a[second + 1]** EcoStruxure Machine SCADA Expert regards **a[second** as the tag and considers it invalid because EcoStruxure Machine SCADA Expert does not find (recognize) the closing bracket. However, if you type **a[second+1]**, this is a valid array tag.

You can specify an array tag wherever you would use a variable name. Also, because array tags greatly simplify configuration tasks and can save development time, we suggest using them whenever possible.

For example, suppose you want to monitor the temperature of four tanks. The conventional configuration method is:

- **temperature1** — high temperature on tank 1
- **temperature2** — high temperature on tank 2
- **temperature3** — high temperature on tank 3
- **temperature4** — high temperature on tank 4

You can use array tags to simplify this task as follows (where [*n*] represents the tank number):

- **temperature[*n*]** — high temperature on tank *n*

The following table contains some additional examples of an array tag:

### Array Tag Examples

| Array Tag Example | Description |
|---|---|
| **Tank[1]**, **Tank[2]**, **Tank[500]** | Simple arrays, where the array indexes (**1**, **2**, and **500**) are numerical constants. For example, tank numbers. |
| **Tank[tk]** | A simple array, where the array index (**tk**) is a tag. For example, a tag representing the tank number. |
| **Tank[tk+1]** | A complex array, where the array index (**tk+1**) is an expression. For example, the value of **tk** (tank number) plus 1. |

> **Note:** When using another tag to reference the index of an array, if the value of the tag is outside the size of the array, then the following results are given:
>
> - If ***IndexTag*** is greater than the size of the array, then **MyArray[*IndexTag*]** will point to the end position of the array; and
> - If ***IndexTag*** is less than 0, then **MyArray[*IndexTag*]** will point to the start position of the array (i.e., **MyArray[0]**).

### Indirect Tags

Indirect tags "point" to other database tags (including class-type tags). Using indirect tags can save development time because they keep you from having to create duplicate tags (and the logic built into them).

You create an indirect tag from any string-type tag simply by typing the **@** symbol in front of the tag name **@*TagName***.

- To reference a simple tag, assume the **strX** tag (a string tag) holds the value **"Tank"**, which is the name of another tag, then reading from or writing to **@strX** provides access to the value of the **Tank** tag.
- To reference a class-type tag and member, you simply create a string tag that points to the class tag and the member. For example, if a tag **strX** (a string tag) holds the value **"Tank.Level"**, which is the name of the class tag, then reading from or writing to **@strX** provides access to the value of the **Tank.Level** member.
- You can also point directly to a class-type tag member; by identifying a class-type that points to a class member. For example: to access the **Tank.Level** member of the class, you must store the **"Tank"** value within the **strX** tag and use the syntax, **@strX.Level**.

## *Choosing the Tag Data Type*

Another consideration when designing a project tag is what type of data the tag will receive. The following data types are recognized:

- **Boolean** (*one bit*): Simple boolean with the possible values of 0 (false) and 1 (true). Equivalent to the "bool" data type in C++. Typically used for turning objects off and on or for closing and opening objects.
- **Integer** (*four bytes*): Integer number (positive, negative, or zero) internally stored as a signed 32-bit. Equivalent to the "signed long int" data type in C++. Typically used for counting whole numbers or setting whole number values. Examples: **0**, **5**, **–200**.
- **Real** (*floating point, eight bytes*): Real number that is stored internally as a signed 64-bit. Equivalent to the "double" data type in C++. Typically used for measurements or for decimal or fractional values.
- **String** (*alphanumeric data, up to 1024 characters*): Character string up to 1024 characters that holds letters, numbers, or special characters. Supports both ASCII and UNICODE characters. Examples: **Recipe product X123**, **01/01/90**, ***** On *****.

You can also assign a new tag to a class that you have previously created.

You can find these tag types (and their respective icons) in the **Global** tab of the *Project Explorer*.

## *Choosing the Tag Scope*

EcoStruxure Machine SCADA Expert allows you to decide whether a tag "lives" on the project server or on each local station:

- **Server** (default): The tag is maintained on the project server, and it is shared by all connected thin clients. A change to the tag value affects the entire project.
- **Local**: A virtual copy of the tag is maintained separately on each local station (server + clients), and a change to the tag value affects only the station on which the change was made.

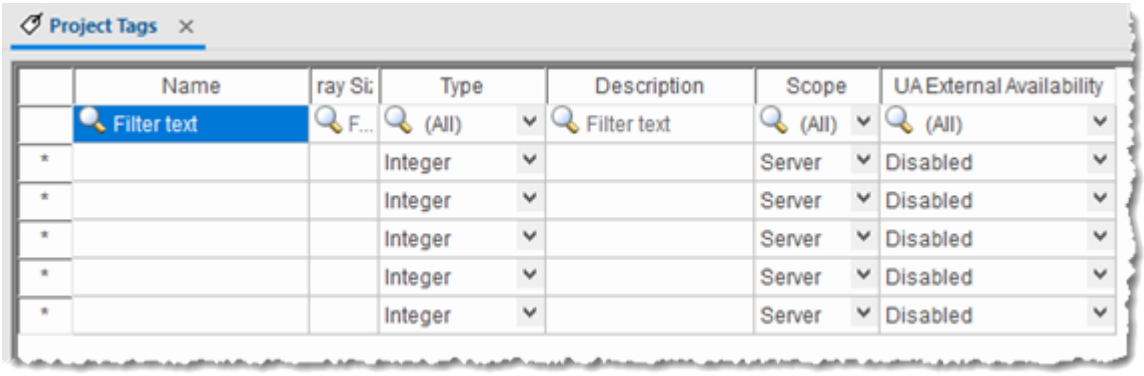
## Creating Database Tags

### *Adding Tags to the Datasheet*

Use the following steps to create tags from the *Project Tags* datasheet:

1. Select the **Global** tab in the Project Explorer, and then in that tab, expand the *Project Tags* folder.
2. Double-click **Datasheet View**.

   The *Project Tags* datasheet is opened for editing.




3. Locate an empty line in the datasheet and then configure the following fields. You can press the **Tab** key to move to the next field.

   **Name**

   Type a name using the proper syntax. For more information, see Naming the Tag on page 154.

   **Array Size**

   The default value is 0, which indicates it is a simple tag. To make the tag an array, type a value to specify the maximum index of the array.

   **Type**

   Click the arrow to select a data type — Boolean, Integer, Real, or String — of the tag. For more information, see Choosing the Tag Data Type on page 156.

   **Description**

   Type a description of the tag. This is for documentation purposes only, and therefore it is optional.

   **Scope**

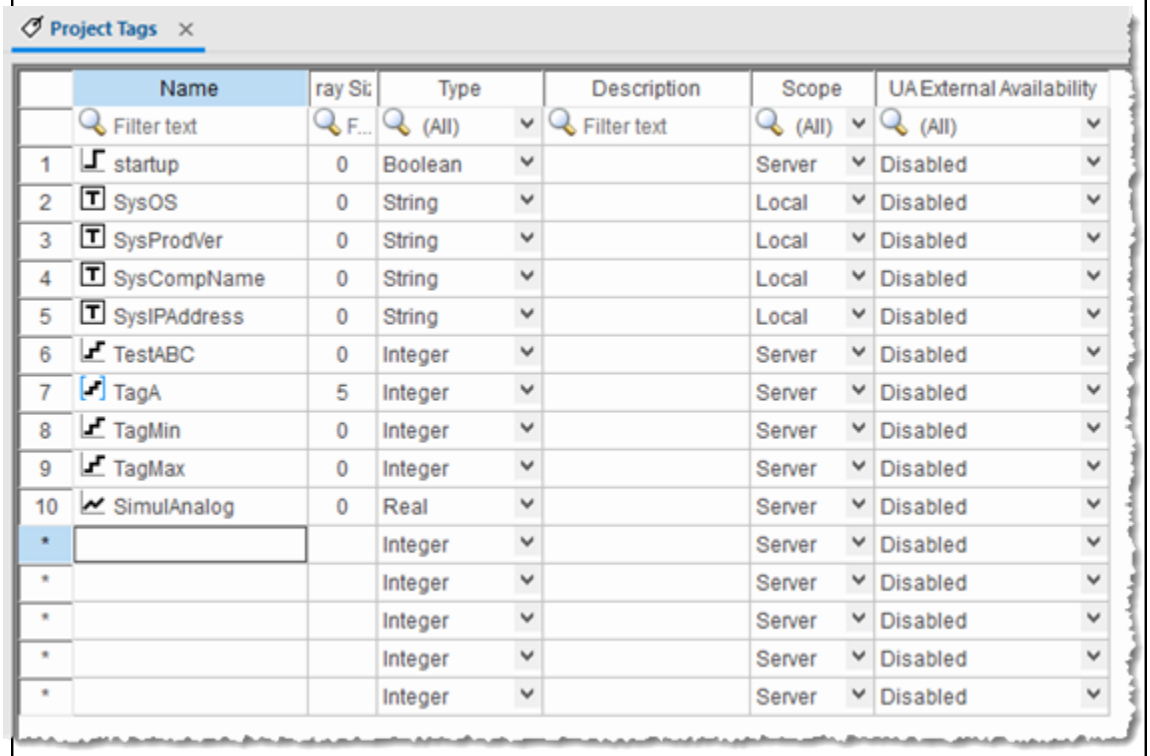
   Click the arrow to select the scope — Server or Local — of the tag. For more information, see Choosing the Tag Scope on page 156.

   **UA External Availability**

   Click the arrow to select the availability — Disabled, Read Only, or Read/Write — of the tag to OPC UA clients. For more information, see OPC UA Server on page 624.

4. Click in a new line to create another tag, or if you have no other tags to create, then save and close the *Project Tags* datasheet.

The following example shows a variety of tags configured in an *Project Tags* datasheet.



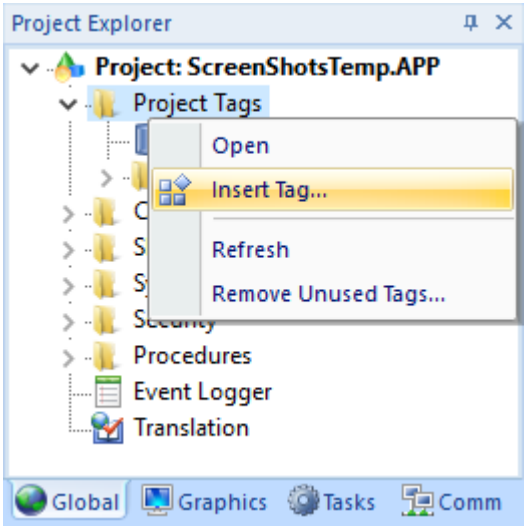

***An example of the Project Tags datasheet***

## *Creating Tags "On-the-Fly"*

Instead of opening the *Project Tags* datasheet every time you want to create a new tag, you can create individual tags "on-the-fly" by performing any of the following actions:

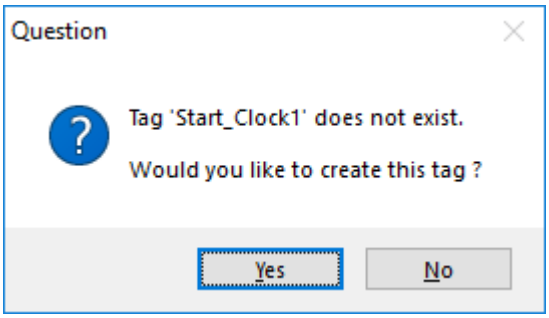
- On the Insert tab of the ribbon, in the Global group, click **Tag**;

- In the *Project Explorer*, right-click on the **Project Tags** folder, the **Datasheet View** icon, or the **Tag List** subfolder and then select **Insert Tag** from the shortcut menu; or




***Inserting a Tag***

- Type a new tag name into any **Tag/Expression** text field (available from *Object Properties* dialogs, worksheets, and so forth). When the *Question* dialog asks if you want to create a new tag, click **Yes**.




***Creating a New Tag***

Any of these actions causes a *New Tag* dialog to display, which you can then complete as needed. For more information, see "Configuring a New Tag".

## *Editing Tags*

You can change the properties of a tag at any time during development or runtime. This section describes two methods you can use to edit tags.

> **Note:** You can right-click on a tag property and use standard Windows commands to cut (Ctrl +X), copy (Ctrl+C), or paste (Ctrl+V) any tag and its properties. You can also Undo (Ctrl+Z) the last modification to a field.

### From the Project Tags Datasheet

Use the following steps to edit one or more tags in the *Project Tags* datasheet:

1. Select the **Global** tab, open the Project Tags folder, and double-click on the **Datasheet View** button.
2. When the *Project Tags* datasheet opens, locate your tag.
3. Double-click in the column containing the information to be changed, and then type the new information into the datasheet.

If you changed a tag name, the *Confirm Global Replace* dialog box is displayed. Click **Yes** to replace the tag throughout your project, so that all objects, animations, tasks, and scripts will keep using the same renamed tag. Click **No** to change the tag name only, but be aware that if the old tag name is still used anywhere in your project, it will cause compiler errors when you try to run your project. To find where the old tag name is used, verify your project.

When the *Confirm Global Replace* dialog box is displayed, you can also select the **Do not display this dialog box again** check box. If you select the check box, you will no longer be able to globally replace tags by editing the datasheet. It is equivalent to always clicking **No** in the *Confirm Global Replace* dialog box, as described above.

4. When you are finished editing, save your changes to the tags database.

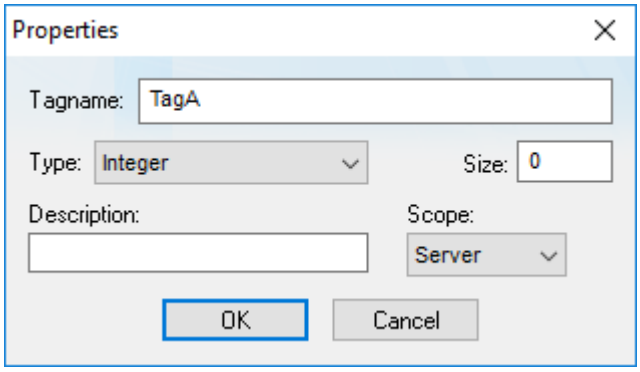
> **Tip:** You can sort the data in the *Project Tags* sheet and/or insert/remove additional columns to/from the sheet by right-clicking on it and choosing the applicable option from the shortcut menu.

### From the Tag List Folder

Use the following steps to edit one or more tags from the Tag List folder:

1. Select the **Global** tab, open the Project Tags folder, and double-click on the the Tag List folder to view a list of all your tags.
2. Locate your tag and double-click on the tag name to open a *Properties* dialog.

> **Note:** You also can right-click on the tag's icon and choose **Properties** from the shortcut menu.




***Properties dialog***

The *Properties* dialog contains fields and combo-boxes that correspond in name and function to the columns on the *Project Tags* datasheet.

3. Make your changes in the *Properties* dialog as follows:
   - To change the current **Type** or **Scope** properties, click the arrow button and select the new information from the list.
   - To change the **Size** or **Description**, highlight the existing text and type the new information into the text box.
4. Click **OK** to save your changes to the tags database and close the *Properties* dialog.

> **Tip:** You can sort the data in the *Project Tags* sheet and/or insert/remove additional columns to/from the sheet by right-clicking on it and choosing the applicable option from the shortcut menu.

## About classes

*Class* tags are compound tags that permit a high-degree of encapsulation within the Tags database. Where basic tags receive a single value, *classes* are designed to receive multiple values.

You can create a class-type tag by grouping basic or array tags, which then become the class *members*. The maximum number of members for any class depends on the product specification.

You specify class-type tags in one of two formats:

- For a simple class tag the syntax is ***TagName.ClassMemberName***. (Where the period is used as a separator.)

  For example, if you wanted to monitor several different conditions (such as *temperature*, *level* and *pressure*) in a tank, you might create a class tag as follows:

  - **Tank.Temperature**
  - **Tank.Level**
  - **Tank.Pressure**

- For creating a complex class tag (using an array tag) the syntax is ***ArrayTagName*[*ArrayIndex*].*ClassMemberName***. (Where again, the period is used as a separator.)

  If you wanted to monitor the *temperature*, *level*, and *pressure* conditions in multiple tanks, you might create a class tag as follows:

  - **Tank[tk].Temperature**
  - **Tank[tk].Level**
  - **Tank[tk].Pressure**

  Where **tk** is an array index, representing the tank number.

### Classes Folder

The *Classes* folder contains all of the project classes and their respective members. Classes are compound tags consisting of user-defined data-type structures or tag types (*Integer*, *Real*, *Boolean*, and *String*). Classes allow for high-level encapsulation in the project database. A class-type tag provides a set of values for its members.

To define a class you must define the *members* and their types. Class members are variables that hold values for an object with particular characteristics. Thus, the defining a class can be very useful for projects with a repeating group of variables.

> **Note:** When you create a class folder, a **Class** icon displays in the *Tag List* subfolder located in the *Project Tags* folder.

To access the members of a class, use the following syntax with a period ( **.** ) as the separator: ***TagName.MemberName***. For example: **tk.LEV** or **tk.TMP**.

If the **Tank** tag is an array, you use the following syntax:

***ArrayTagName*[*ArrayIndex*].*MemberName***

For example: **Tank[1].Level** or **Tank[n].Temperature**

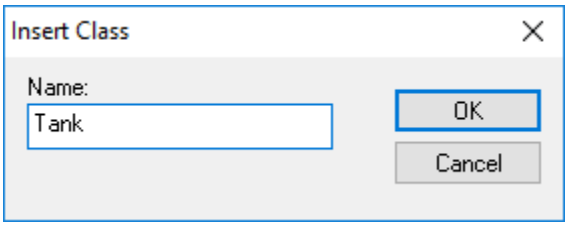
A class-type tag contains a set of values (rather than a single value) associated with the class. You create class-type tags by grouping simple tags, which become the members. The maximum number of members for any class depends on the product specification. Class members can hold standard *Integer*, *Real*, *Boolean*, and *String* values, as mentioned previously.

To create a new class, use one of the following methods to open the *Insert Class* dialog:

- On the Insert tab of the ribbon, in the Global group, click **Class**;
- Right-click on the *Classes* folder, the *Members List* sub-folder, or the **Datasheet View** icon in the *Classes* folder; or

- Create a new class tag in the *Project Tags* folder.

When the *Insert Class* dialog displays, enter a class name in the **Name** field, and then click **OK** to close the dialog.

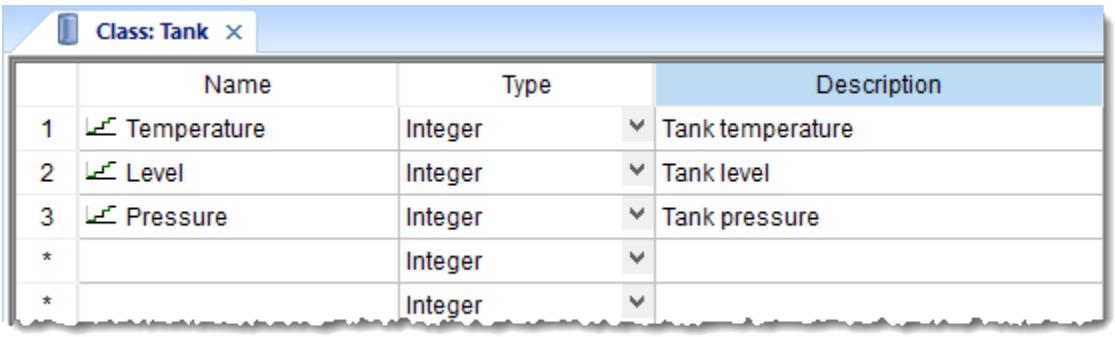



***Insert Class dialog***

> **Note:** You must type a unique class name. You cannot create two classes with the same name. In addition, you cannot configure shared tags and system tags as classes.

EcoStruxure Machine SCADA Expert saves class folders in the *Tag List* subfolder (located in the *Project Tags* folder). You can edit the classes in this folder.

When the *Class* datasheet displays, you can use it to create, modify, or delete any class members and their viewable properties. (You cannot edit classes from the *Tag Property* dialog.)




***Class datasheet***

> **Note:** The Classes folder can contain up to 16,384 classes and up to 4,096 members per class. This is a technical limitation of the tags database, not a licensing restriction; unused classes do not count against the total number of tags used.
>
> When a new tag is created with a class type, however, each class member counts as a tag used because each member holds a value. (For example, if you create a class with 5 members and then create 5 tags with that class type, then you have a total of 25 tags used.) The total number of tags used cannot exceed the number of tags supported by the project's target system / runtime license.

To edit a class member or property, you can right-click on the item and use standard Windows commands to cut (Ctrl+X), copy (Ctrl+C), or paste (Ctrl+V). You can also undo (Ctrl+Z) the last modification to a field.

You also edit member properties as follows:

- **Name** field: Type a name for the member or member property. The first character must be a letter and you can use up to 255 characters in the name.
- **Type** combo-box: Select a member type (*Boolean*, *Integer*, *Real*, or *String*).
- **Description** field: Type a description of the member property for documentation purposes.

> **Note:** Members of a class cannot be of another class type.

Also, you must create a unique class name. You cannot reuse the name of an existing class. However, you can create members with the same name in different classes.

To delete a class and all its members, right-click on a class folder and select delete. EcoStruxure Machine SCADA Expert disables the delete option if you are running any runtime tasks. In addition, you cannot delete a class if it is associated with any tag.

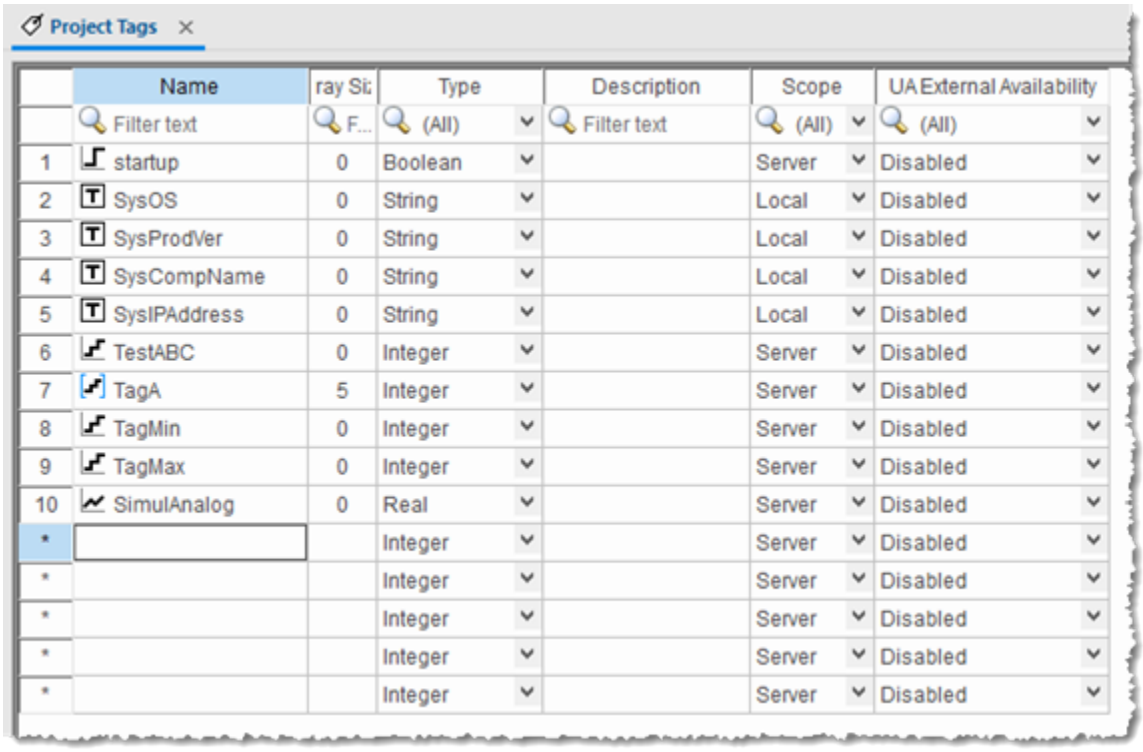
## Tag Properties

Tag properties are metadata associated with each project tag in the database.

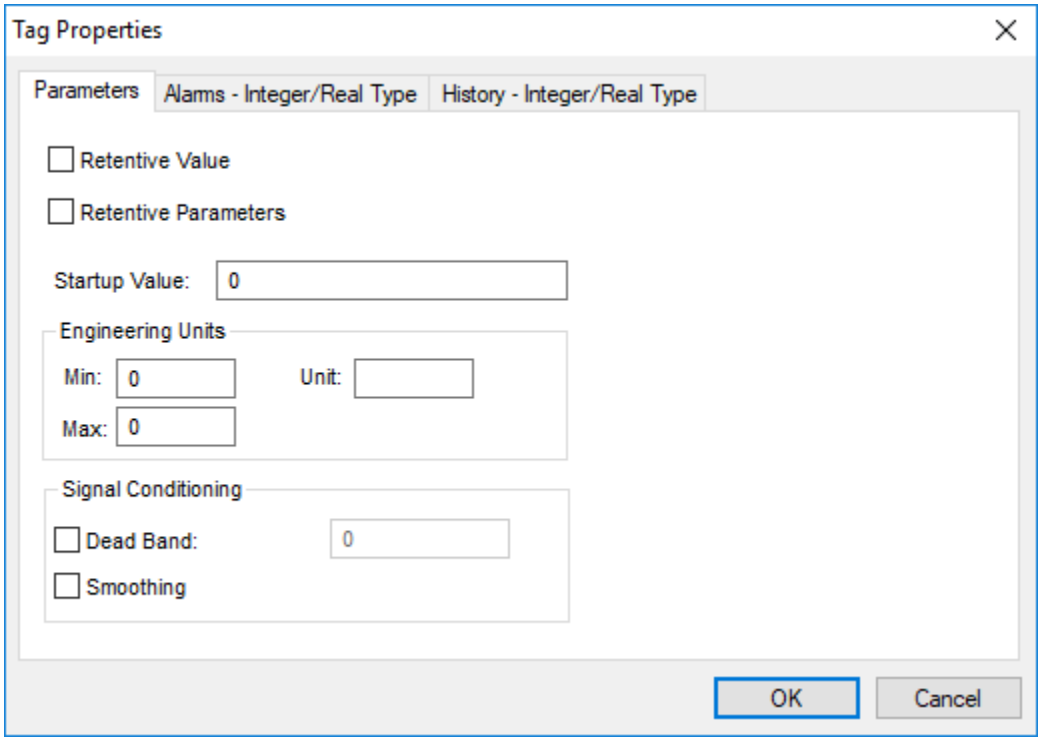
Most of the time, you may think of project tags as simple program variables that store values, because that is how you typically use them in your project. Each tag, however, is in fact a complex data structure that can be handled in different ways during project run time, depending on how the tag properties are configured.

In addition to handling standard metadata like array size, data type, description, and scope, the tag properties can be used to configure alarm conditions, get tag quality, convert between different units of measurement, access individual bits, retain values through project restarts, save historical data, and so on.

Many tag properties can be viewed and edited directly in the **Project Tags** datasheet. In fact, every column in the datasheet (including **Name**) is another tag property, and while the datasheet shows the most common properties by default, you can configure it to show other properties in additional columns.



You can also use the **Properties** command (on the **Home** tab of the ribbon, in the **Tags** group) to open the *Tag Properties* dialog box, which shows all of the properties for a selected tag.




Finally, you can get and set many tag properties during project run time, just as you would get and set the values of the project tags themselves.

### *Set tag properties using the Project Tags datasheet*

Use the *Project Tags* datasheet to set the properties of project tags.

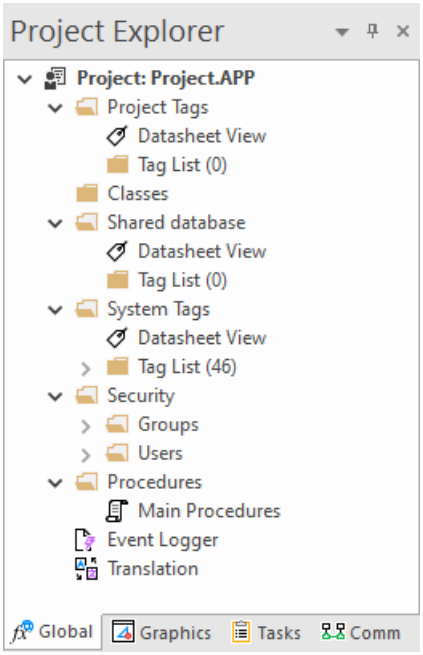
The datasheet is essentially a spreadsheet that lists all of the tags (not including shared and system tags) that are in your project database, as well as certain properties of those tags. You can change which tag properties are included in the spreadsheet by showing or hiding additional columns, and you can set the properties of specific tags by entering new values in the appropriate cells.

> **Note:** You cannot use the *Project Tags* datasheet during run time. If you need to edit the tags database during run time, use the Tags Database functions.

To use the *Project Tags* datasheet to set tag properties:

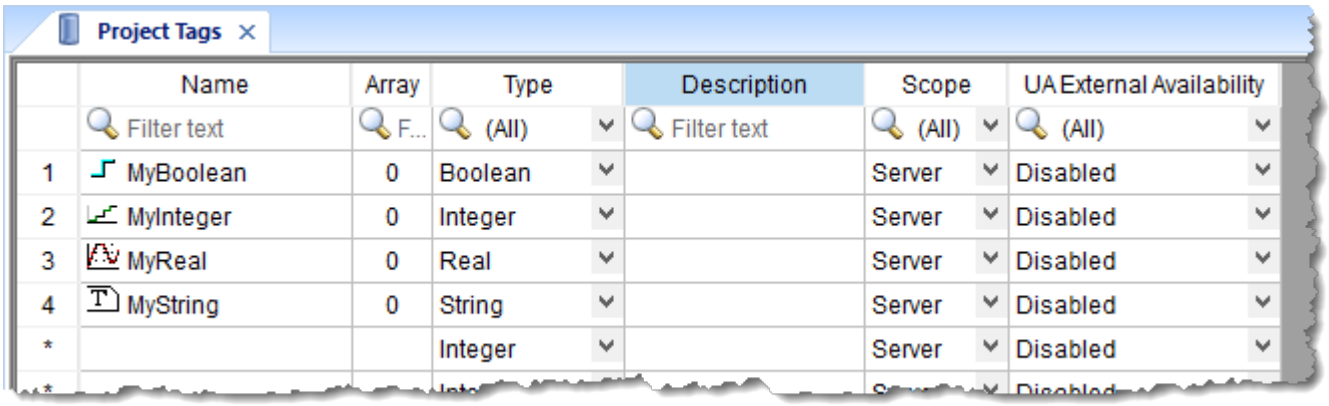
1. At the bottom of the *Project Explorer* window, click **Global**.

   The **Global** tab is displayed.




2. Expand the **Project Tags** folder, and then in the folder, double-click **Datasheet View**.

   The *Project Tags* datasheet is opened in the screen/worksheet editor.




3. To show/hide additional columns for other tag properties, right-click anywhere in the datasheet, and then on the shortcut menu, click the desired properties:
   - **Name** (cannot be hidden)
   - **Size** (shown by default)
   - **Type** (shown by default)
   - **Description** (shown by default)
   - **Scope** (shown by default)
   - **More Columns** > **Startup**
   - **More Columns** > **Min**
   - **More Columns** > **Max**
   - **More Columns** > **Unit**

- **More Columns** > **Retentive Value**
- **More Columns** > **Retentive Parameters**
- **More Columns** > **Dead Band**
- **More Columns** > **Smoothing**
- **More Columns** > **UA External Availability** (shown by default)

Each row of the datasheet represents a project tag, and each column of the datasheet represents a property of that tag.

4. To set a tag property, enter the new value in the appropriate cell. Repeat as needed.

   Some properties do not apply to all data types, so for more information about the applicable properties, see:

   - Properties of Integer and Real tags on page 171
   - Properties of Boolean tags on page 176
   - Properties of String tags on page 180

5. When you are done, save and close the datasheet.

### *Set tag properties using the Properties command*

Use the **Properties** command to set the properties of project tags.

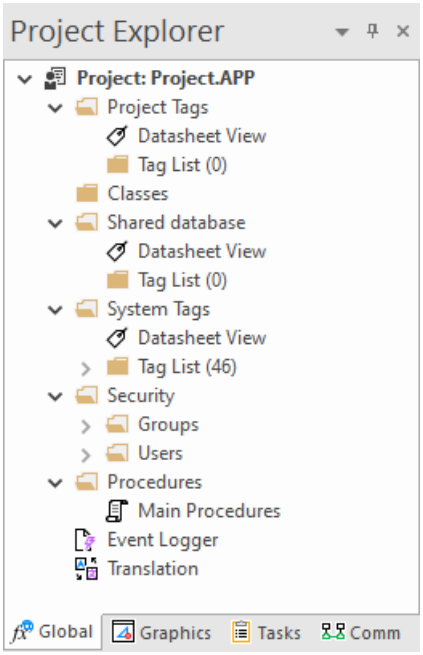
The **Properties** command opens the *Tag Properties* dialog box, which you can use to set any and all of the properties of a selected tag. This includes alarms and history properties, which cannot be set using the Project Tags datasheet. However, when you use the **Properties** command, you can set the properties of only one project tag at a time.

> **Note:** You cannot use the **Properties** command during run time. If you need to edit the tags database during run time, use the Tags Database functions.

To use the **Properties** command to set tag properties:

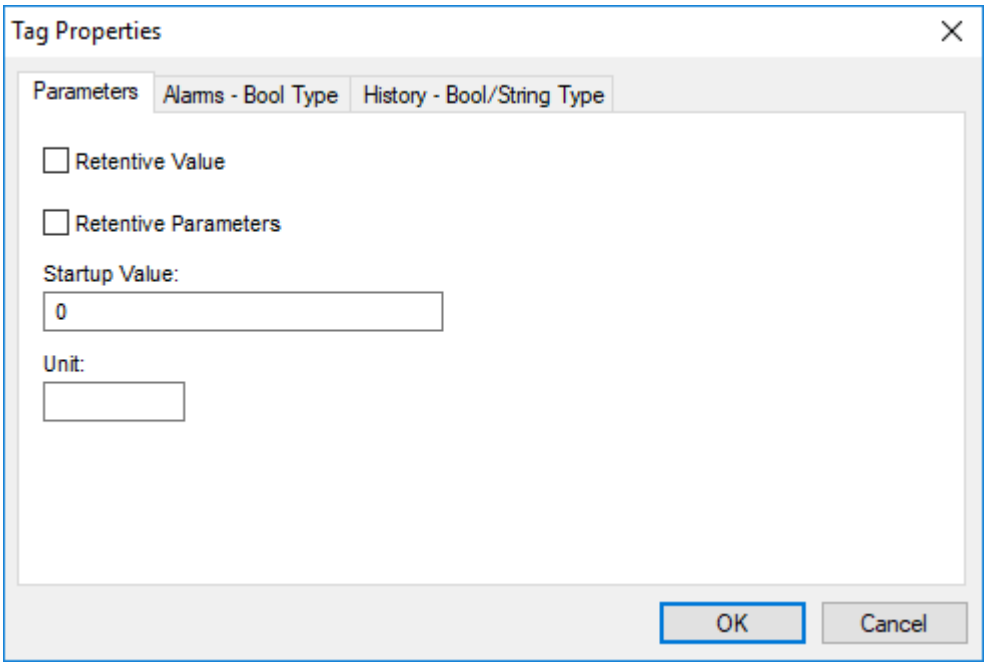
1. At the bottom of the *Project Explorer* window, click **Global**.

   The **Global** tab is displayed.




2. Expand the **Project Tags** folder, and then in that folder, expand the **Tag List** folder.
   A list of all of the project tags is displayed.
3. In the **Tag List** folder, select the project tag for which you want to set properties.
4. On the **Home** tab of the ribbon, in the **Tags** group, click **Properties**.

   The *Tag Properties* dialog box for the selected tag is displayed.




5. Use the dialog box to set the tag properties as needed, keeping in mind that the properties are distributed among multiple tabs in the dialog box.

Some properties do not apply to all data types, so for more information about the applicable properties, see:

- Properties of Integer and Real tags on page 171
- Properties of Boolean tags on page 176
- Properties of String tags on page 180

6. When you are done, click **OK** to save your changes and close the dialog box.

### Reference a tag property instead of a project tag

You can use special syntax to reference a tag property in the same way that you reference a project tag.

Anywhere in your project that you would normally reference a project tag — for example, when you configure a tag/expression to control the visibility of a screen object — you can reference a tag property instead, using the following syntax:

```
tagname->property
```

Examples:

```
MyInteger->Quality
```

```
MyReal->HiHi
```

```
MyClass.FirstMember->MemberName
```

```
MyArray[1]->Index
```

This works everywhere in EcoStruxure Machine SCADA Expert 2020, including in VBScript and in function parameters, and it can be used to both get and set the values of tag properties.

> **Tip:** When tag properties are referenced like this, they are also known as tag fields.

To get the correct spelling of property names, as well as some property-specific limitations, see Complete list of tag properties on page 181.

### Using TagsDB functions to edit the tags database during run time

Use the Tags Database (TagsDB) functions to add and remove tags, classes, and class members during project run time, as well as to set properties and alarm conditions on tags.

There are several important things to keep in mind when you use TagsDB functions, because the functions can do much more than set and get tag values. They actually change the structure of the tags database, which can cause serious problems for a running project and all connected thin clients if it is not done properly. As such, most TagsDB functions can be executed only under the following limitations.

First, you must use the SCADA runtime edition for Windows to run your project — in other words, you must install the full Studio software on a Windows computer and then license it for "Runtime only". (You can also license the software for "Engineering only", but your project will run for only 72 hours before it needs to be restarted.) The TagsDB functions use the project development environment's database editor in essentially the same way that you do when you manually edit your project during run time. Due to this limitation, the TagsDB functions cannot be used in projects that are developed for and run on embedded devices.

Second, the TagsDB functions can be called only by scripts that are executed on the project runtime server. The functions cannot be executed on project thin clients because a client cannot make structural changes to the tags database without interfering with other clients, decreasing run-time performance, and potentially corrupting the database. (In this case, "project thin clients" includes the Viewer module that runs on the same

computer as the project runtime server, because it runs as a separate process on that computer.) Therefore, generally speaking...

TagsDB functions **can** be used in:

- The Startup Script, which is executed when the project itself is run;
- Script Groups, which are periodically scanned by the Background Task; and
- Global Procedures that are called by the Startup Script or Script Groups.

TagsDB functions **cannot** be used in:

- The Graphics Script, which is executed by each project thin client client when it starts;
- Screen Scripts, which are attached to project screens and executed when those screens are opened; and
- Command animations.

To work around this limitation, create a Script Group to call the TagsDB functions and then configure a trigger to control the execution of that Script Group.

Third, in any script that calls TagsDB functions to make structural changes to the tags database, you must call the `TagsDBBeginEdit` function at the beginning of the script and the `TagsDBEndEdit` function at the end of the script. The `TagsDBBeginEdit` function locks the database for editing and prevents any other run-time changes. The `TagsDBEndEdit` function applies the changes made by TagsDB functions and then allows the database to resume normal run-time behavior. Both functions must be called in the same script, because that script (more specifically, the program thread running that script) owns the tags database while it is locked. You cannot call `TagsDBBeginEdit` in one script and then call `TagsDBEndEdit` in another script.

Normally, when a project is edited during run time, the project runtime server and all project thin clients must be updated with the changes as they are made. This is not a problem when you manually edit your project, because you make your changes slowly and one at a time. In contrast, the TagsDB functions allow you to make a large number of changes quickly, so updating the server and clients with all of those changes while the project is running can severely decrease run-time performance. Therefore, to maintain performance and protect the tags database, the server — including all background tasks such as alarms, trends, and other scripts — is effectively paused when the `TagsDBBeginEdit` function is executed, and then the changes are applied as a batch when the `TagsDBEndEdit` function is executed. Also, as part of this update process, project screens that were already open on clients will be reopened and their OnOpen screen scripts will be executed again.

> **Note:** The `TagsDBBeginEdit` function has a persistent effect, which means that if you call the function to lock the tags database during project run time and then stop the project, the database will remain locked and you will not be able to manually edit it.
>
> Restarting the project may or may not unlock the database, depending on how you developed your project and which function call locked the database in the first place. As such, while the project is stopped, you should use the *Watch* window to manually call the `TagsDBEndEdit` function. When it is successfully executed, you can safely restart the project.

### Examples

The following example shows how to use the TagsDB functions in VBScript to add a new class, then add a new class member to that class, then add a new tag of that class, then set an alarm and a trend on that tag.

```
If($TagsDBBeginEdit()=0) Then
  If($TagsDBAddClass("TempClass")=0) Then
    If($TagsDBAddClassMember("TempClass","TempMember","Real")=0) Then
      If($TagsDBAddTag("TempTag","TempClass",2,0)=0) Then
        If($TagsDBSetAlarm("TempTag[0].TempMember",1,0,3.5)<>0) Then
          $Msg = "Alarm not Set"
        End If
        If($TagsDBSetTrend("TempTag[0].TempMember",0,1)<>0) Then
          $Msg = "Trend not Set"
        End If
      Else
        $Msg = "Tag not created"
      End If
    Else
```

```
      $Msg = "Class Member not added"
    End If
  Else
    $Msg = "Class not created"
  End If
  $TagsDBEndEdit()
Else
  $Msg = "Tag functions not enabled"
End If
```

Please note how the script begins with the `TagsDBBeginEdit` function and then ends with the `TagsDBEndEdit` function. Also, see how the nested **If**…**Then**…**Else** structures ensure that each function is executed successfully (i.e., returns a value of 0) before the next one is attempted.

The following example shows how to remove the alarm, trend, tag, class member, and class, in reverse order from how they were added in the previous example.

```
If($TagsDBBeginEdit()=0) Then
  If($TagsDBRemoveAlarm("TempTag",1)<>0) Then
    $Msg = "Alarm not removed"
  End If
  If($TagsDBRemoveTrend("TempTag")<>0) Then
    $Msg = "Trend not removed"
  End If
  If($TagsDBRemoveTag("TempTag")=0) Then
    If($TagsDBRemoveClassMember("TempClass","TempMember")<>0) Then
      $Msg = "Class member not removed"
    End If
    If($TagsDBRemoveClass("TempClass")<>0) Then
      $Msg = "Class not removed"
    End If
  Else
    $Msg = "Tag not removed"
  End If
  $TagsDBEndEdit()
Else
  $Msg = "Tag functions not enabled"
End If
```

It is not absolutely necessary to remove the alarm and trend before removing the tag they are on, because they are discarded with the rest of the tag properties and other metadata when the tag itself is removed. They are included in the example above simply to be thorough. In contrast, the class member and class cannot be removed until every tag in that class is removed.

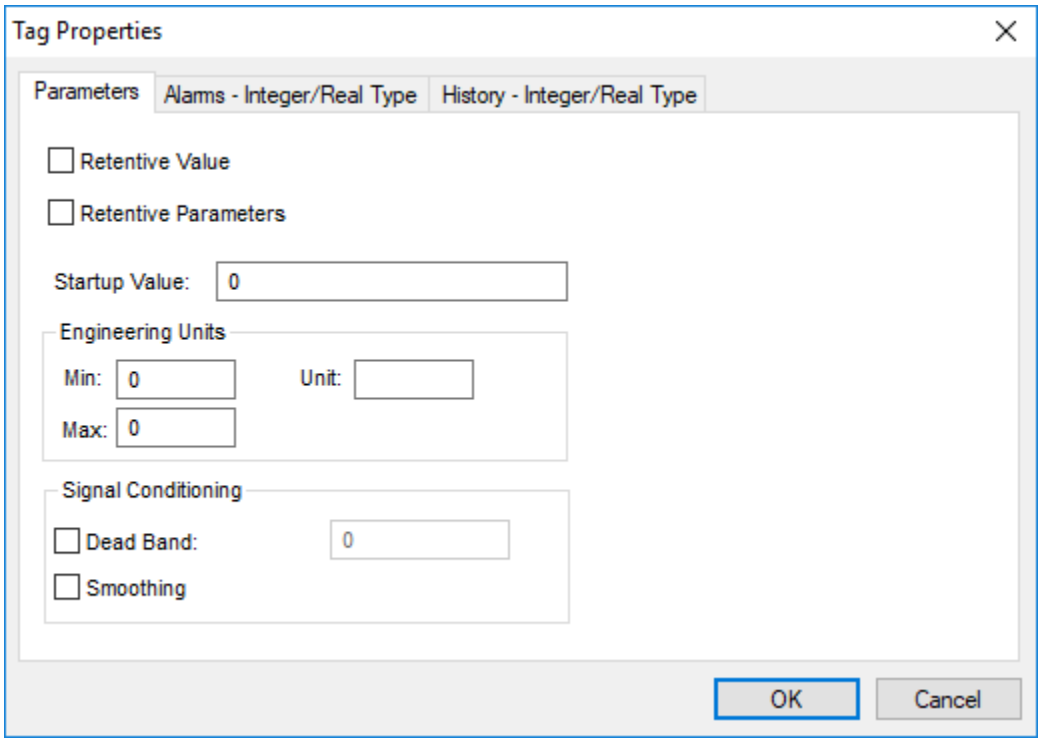
## *Properties of Integer and Real tags*

Each Integer and Real tag in the tags database has several properties (or metadata) in addition to its actual value. You can set these properties by using the *Project Tags* datasheet, the **Properties** command on the ribbon, or the Tags Database functions.

### Parameters

The following list describes the general parameters of Integer and Real tags. These parameters determine how the selected tag is used in your project.

> **Tip:** You can also use the Project Tags datasheet to set the parameters of a project tag. This dialog box reflects any changes you make in that datasheet, and vice versa. For more information, see Set tag properties using the Project Tags datasheet on page 147.




**Retentive Value**

Continuously save the actual tag value during project run time, in case the project stops unexpectedly. When the project is run again, the project tag will start with the last saved value.

Selecting this option will increase drive access during project run time and therefore can reduce performance.

**Retentive Parameters**

Continuously save the tag properties for the project tag during project run time, in case the project stops unexpectedly. When the project is run again, the project tag will start with the last saved properties.

Selecting this option will increase drive access during project run time and therefore can reduce performance.

**Startup Value**

The value with which the project tag will start when the project is run.

If the **Retentive Value** option is selected, the last saved value will be used instead of the startup value.

**Engineering Units**

**Min**

The minimum allowed value of the project tag. Any attempt to set a value less than this minimum will be ignored, and the project will generate a log message indicating that it tried to set a tag value outside of the defined range. For more information, see Change how out-of-range tag values are handled on page 187.

> **Note:** If both **Min** and **Max** are 0, there is no minimum value.

#### Max

The maximum allowed value of the project tag. Any attempt to set a value greater than this maximum will be ignored, and the project will generate a log message indicating that it tried to set a tag value outside of the defined range. For more information, see Change how out-of-range tag values are handled on page 187.

> **Note:** If both **Min** and **Max** are 0, there is no maximum value.

#### Unit

A brief description or reference (up to 7 characters) for the project tag.

This parameter is typically used to describe the engineering units (e.g., kg, BTU, PSI) in which the tag value is given. It is for reference only and does not affect the actual tag value.

### Signal Conditioning

#### Dead Band

The minimum amount by which the tag value must change in order for the new tag value to be saved. (By default, every change in the tag value is saved.)

To enable the dead band, click the **Dead Band** check box and then type the dead band value in the box to the right. Do not specify a percentage; the dead band value must be specified in the same units as the tag value.

#### Smoothing

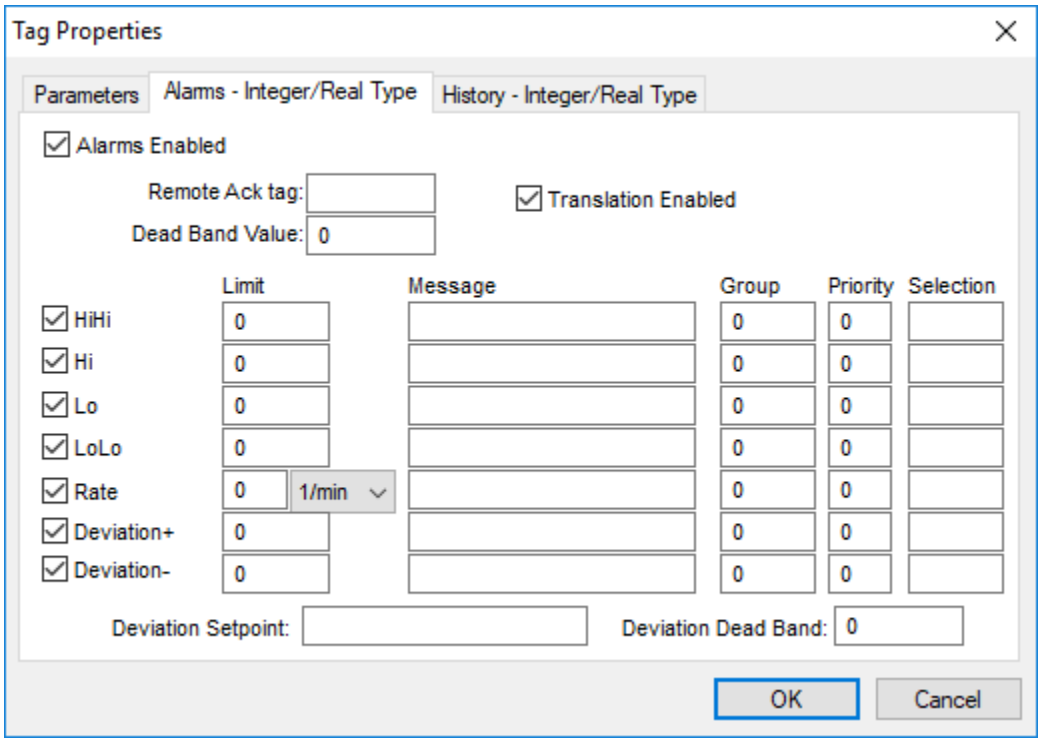
Average together successive changes in the tag value in order to reduce statistical noise.

> **Note:** This can change the actual tag value.

## Alarms

The following list describes the alarm properties of Integer and Real tags. These properties determine how the project checks for alarm conditions on the selected tag.

> **Tip:** You can also use an Alarm worksheet to set the alarm properties of a project tag. This dialog box reflects any changes you make in that worksheet, and vice versa. For more information, see Alarm worksheet on page 377.




**Alarms Enabled**

Enable checking for alarm conditions on this tag, as configured below.

**Remote Ack tag**

The name of another tag that can be used to acknowledge alarms on this tag. When the value of the specified tag changes, all of the unacknowledged alarms are acknowledged.

**Dead Band Value**

The minimum amount by which the tag value must come within its normal range in order for an active alarm to be normalized.

For example, a tag has a HiHi alarm limit of 90 and a dead band value of 5. When the tag value is greater than or equal to 90, the alarm becomes active. After that, the alarm becomes normalized only when the tag value is less than or equal to 85.

**Translation Enabled**

Enable translation of the alarm messages that are configured for this tag. For more information about translation, see Project Localization on page 681.

> **Note:** Only the original alarm messages are saved in the historical database. The translated alarm messages are saved in a separate file in your project folder at: `<project name>/Database/alarm.txt`

**Alarms**

The alarm conditions that are configured for this tag. To enable a specific type of alarm condition, click the corresponding check box to the left. When it is enabled, additional properties become available:

**Limit**

For **HiHi**, **Hi**, **Lo**, and **LoLo**, the limit is the actual value that the tag must exceed in order to activate the alarm.

For **Rate**, the limit is the instantaneous rate of change that the tag must exceed in order to activate the alarm. The instantaneous rate of change is calculated by averaging the changes in tag value over time. Please note that if you enable this alarm type and specify a limit for it, you must also select the frequency at which the rate will be checked. For example, if the alarm is configured with a limit of 10 and a frequency of 1/s, and the tag value changes from 50 to 65 within one second, the alarm will become active.

For **Deviation+** and **Deviation-**, the limit is value that is added to or subtracted from the deviation set point. Please note that if you enable either of these alarm types and specify limits for them, you must also specify a value for **Deviation Setpoint** below.

**Message**

The message that is displayed when the alarm becomes active. Messages can be displayed in an Alarm/Event Control object, emailed to personnel, saved in the historical database, and/or sent to the run-time log, depending on how your project is configured.

**Group**

The Alarm group/worksheet to which this alarm condition belongs.

**Priority**

The priority number associated with the alarm. When viewing alarms in an Alarm/Event Control object , the user can sort and/or filter the alarms by priority.

**Selection**

An alias (e.g., AreaA, AreaB) associated with the alarm. When viewing alarms in an Alarm/Event Control object, the user can sort and/or filter the alarms by their selection values.

**Deviation SetPoint**

The set point for the **Deviation+** and **Deviation-** alarm conditions. When the actual tag value deviates from this set point, the appropriate alarm becomes active.

If you want to be able to change the set point during project run time, type the name of another project tag (e.g., **`MyDevSetPoint`**). The value of that tag will be used as the set point.

**Deviation Dead Band**

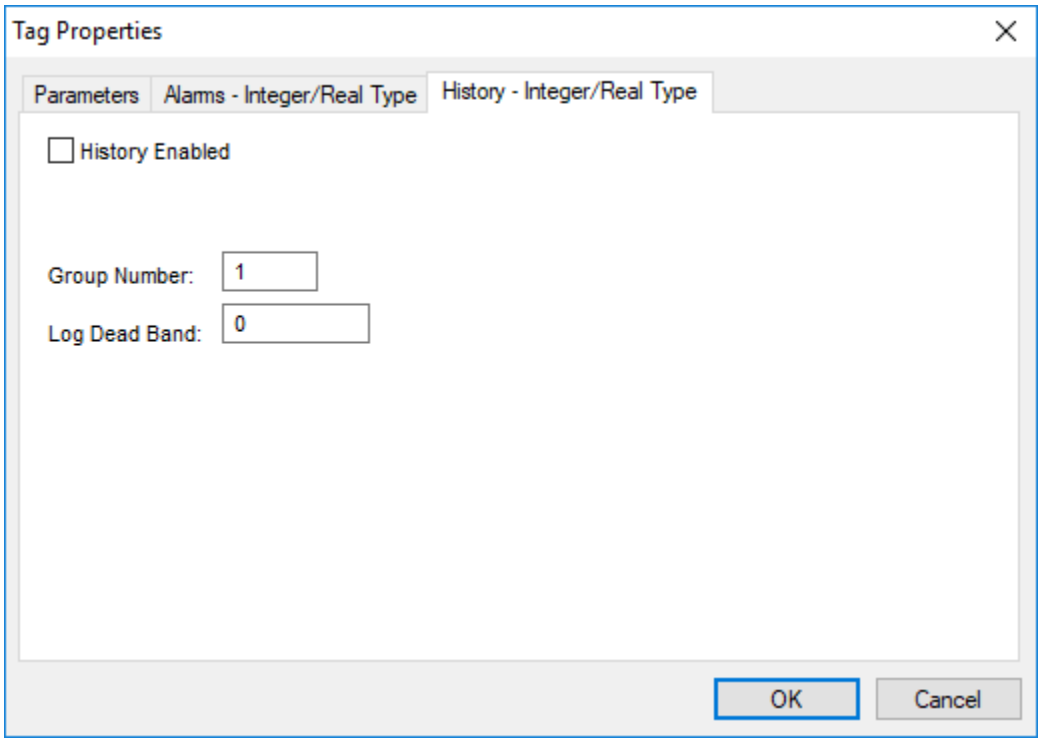
The dead band for the **Deviation+** and **Deviation-** alarm types, similar to **Dead Band Value** above for the other alarm types. This is the minimum amount by which the tag value must come within its normal range in order for an active alarm to be normalized.

For more information about the types of alarms and how they are used during project run time, see Alarm worksheet on page 377.

## History

The following list describes the history properties of Integer and Real tags. These properties determine how the project saves historical data for the selected tag.

> **Tip:** You can also use a Trend worksheet to set the history properties of a project tag. This dialog box reflects any changes you make in that worksheet, and vice versa. For more information, see Trend worksheet on page 413.




**History Enabled**

Enable the saving of historical data for this tag.

**Group Number**

The Trend group/worksheet to which this project tag is assigned.

**Log Dead Band**

The amount by which the actual tag value must change in order for the change to be saved in the historical database.

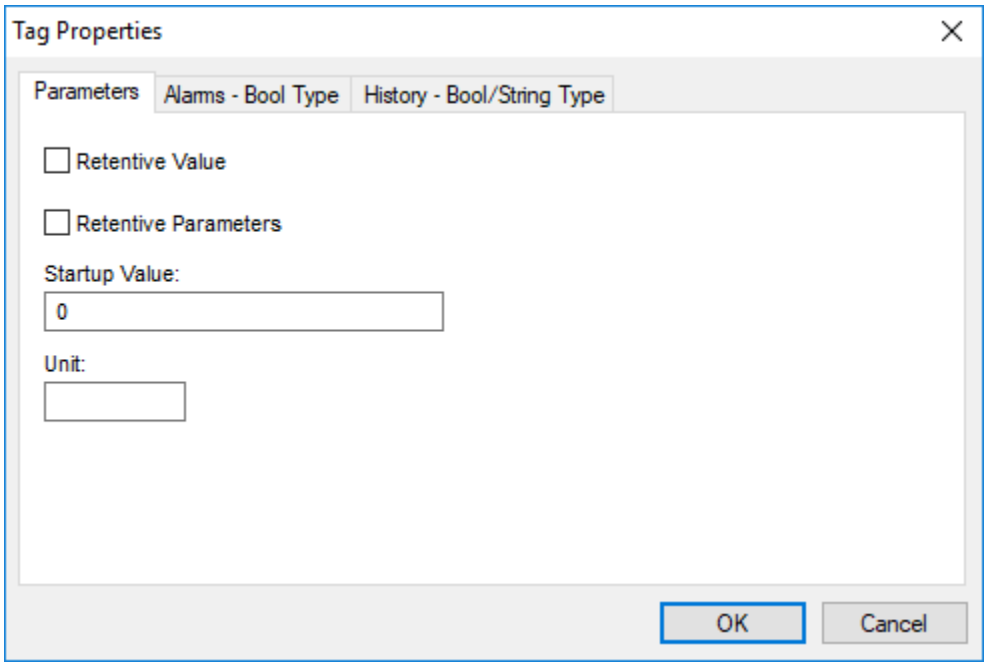
## *Properties of Boolean tags*

Each Boolean tag in the tags database has several properties (or metadata) in addition to its actual value. You can set these properties by using the *Project Tags* datasheet, the **Properties** command on the ribbon, or the Tags Database functions.

### Parameters

The following list describes the general parameters of Boolean tags. These parameters determine how the selected tag is used in your project.

> **Tip:** You can also use the Project Tags datasheet to set the parameters of a project tag. This dialog box reflects any changes you make in that datasheet, and vice versa. For more information, see Set tag properties using the Project Tags datasheet on page 147.




**Retentive Value**

Continuously save the actual tag value during project run time, in case the project stops unexpectedly. When the project is run again, the project tag will start with the last saved value.

Selecting this option will increase drive access during project run time and therefore can reduce performance.

**Retentive Parameters**

Continuously save the tag properties for the project tag during project run time, in case the project stops unexpectedly. When the project is run again, the project tag will start with the last saved properties.

Selecting this option will increase drive access during project run time and therefore can reduce performance.

**Startup Value**

The value with which the project tag will start when the project is run.

If the **Retentive Value** option is selected, the last saved value will be used instead of the startup value.

**Unit**

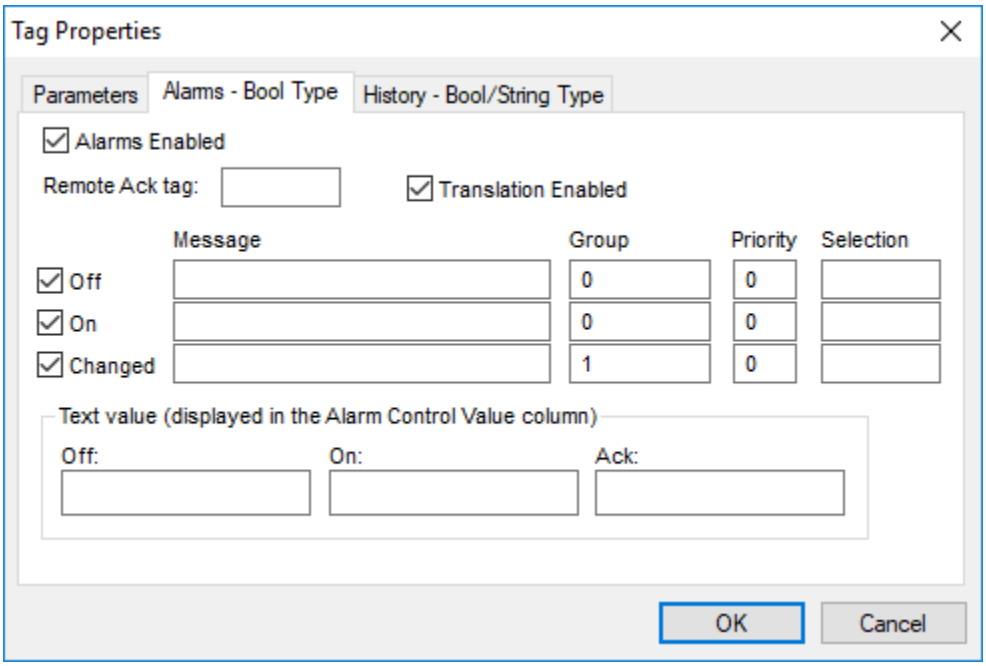
A brief description or reference (up to 7 characters) for the project tag.

This property is typically used to describe the engineering units in which the tag value is given. Boolean and String tags do not have engineering units, however, so this property can also be used to supplement the normal tag description.

## Alarms

The following list describes the alarm properties of Boolean tags. These properties determine how the project checks for alarm conditions on the selected tag.

> **Tip:** You can also use an Alarm worksheet to set the alarm properties of a project tag. This dialog box reflects any changes you make in that worksheet, and vice versa. For more information, see Alarm worksheet on page 377.




**Alarms Enabled**

Enable checking for alarm conditions on this tag, as configured below.

**Remote Ack tag**

The name of another tag that can be used to acknowledge alarms on this tag. When the value of the specified tag changes, all of the unacknowledged alarms are acknowledged.

**Translation Enabled**

Enable translation of the alarm messages that are configured for this tag. For more information about translation, see Project Localization on page 681.

> **Note:** Only the original alarm messages are saved in the historical database. The translated alarm messages are saved in a separate file in your project folder at: `<project name>/Database/alarm.txt`

**Alarms**

The alarm conditions that are configured for this tag. To enable a specific type of alarm condition, click the corresponding check box to the left. When it is enabled, additional properties become available:

**Message**

The message that is displayed when the alarm becomes active. Messages can be displayed in an Alarm/Event Control object, emailed to personnel, saved in the historical database, and/or sent to the run-time log, depending on how your project is configured.

**Group**

The Alarm group/worksheet to which this alarm condition belongs.

**Priority**

The priority number associated with the alarm. When viewing alarms in an Alarm/Event Control object , the user can sort and/or filter the alarms by priority.

**Selection**

An alias (e.g., AreaA, AreaB) associated with the alarm. When viewing alarms in an Alarm/Event Control object, the user can sort and/or filter the alarms by their selection values.

**Text values**

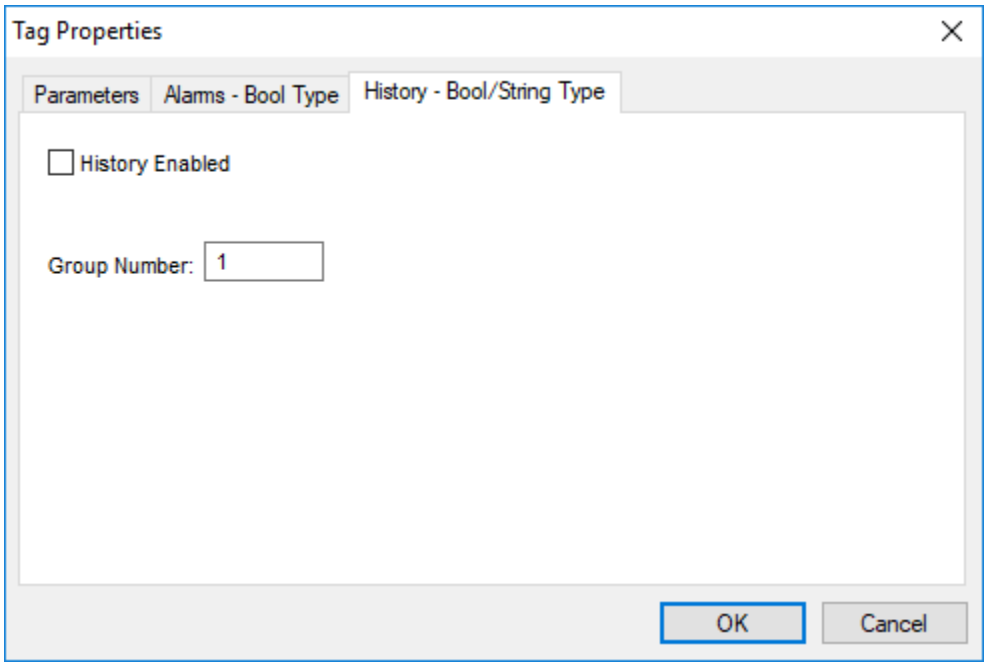
Text that is displayed in the Value column of an Alarm/Event Control object when each type of alarm becomes active. These are typically mnemonics that correspond to the states of the Boolean tag — for example, "Closed" when the Off alarm is active and "Open" when the On alarm is active. If you do not configure these mnemonics, the actual tag value (`0` or `1`) will be displayed instead.

For more information about the types of alarms and how they are used during project run time, see Alarm worksheet on page 377.

## History

The following list describes the history properties of Boolean tags. These properties determine how the project saves historical data for the selected tag.

> **Tip:** You can also use a Trend worksheet to set the history properties of a project tag. This dialog box reflects any changes you make in that worksheet, and vice versa. For more information, see Trend worksheet on page 413.




**History Enabled**

Enable the saving of historical data for this tag.

**Group Number**

The Trend group/worksheet to which this project tag is assigned.

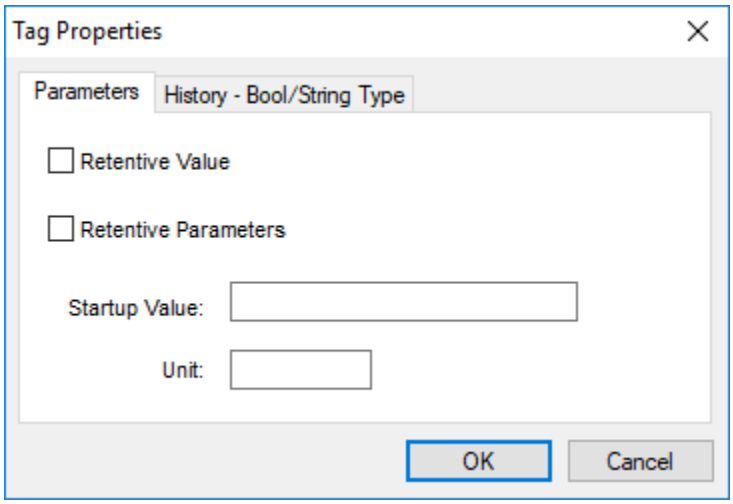
## *Properties of String tags*

Each String tag in the tags database has several properties (or metadata) in addition to its actual value. You can set these properties by using the *Project Tags* datasheet, the **Properties** command on the ribbon, or the Tags Database functions.

### Parameters

The following list describes the general parameters for String tags.

> **Tip:** You can also use the Project Tags datasheet to set the parameters of a project tag. This dialog box reflects any changes you make in that datasheet, and vice versa. For more information, see Set tag properties using the Project Tags datasheet on page 147.




**Retentive Value**

Continuously save the actual tag value during project run time, in case the project stops unexpectedly. When the project is run again, the project tag will start with the last saved value.

Selecting this option will increase drive access during project run time and therefore can reduce performance.

> **Note:** Due to a technical limitation in EcoStruxure Machine SCADA Expert 2020, only the first 100 characters of a string can be saved in this way.

**Retentive Parameters**

Continuously save the tag properties for the project tag during project run time, in case the project stops unexpectedly. When the project is run again, the project tag will start with the last saved properties.

Selecting this option will increase drive access during project run time and therefore can reduce performance.

**Startup Value**

The value with which the project tag will start when the project is run.

If the **Retentive Value** option is selected, the last saved value will be used instead of the startup value.

**Unit**

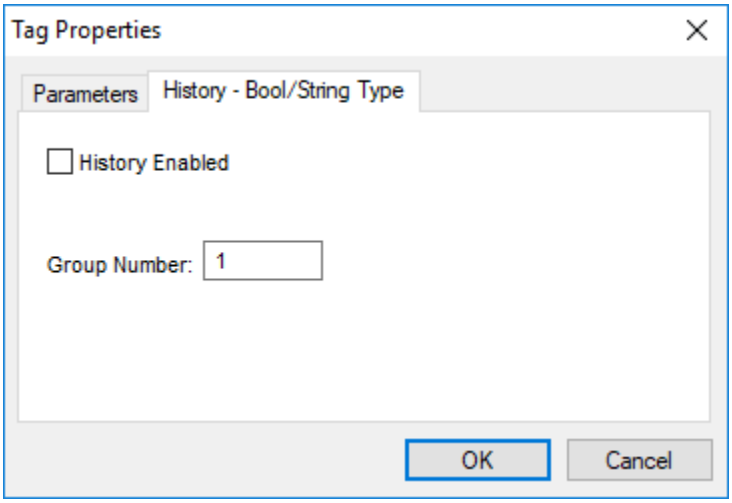
A brief description or reference (up to 7 characters) for the project tag.

This property is typically used to describe the engineering units in which the tag value is given. Boolean and String tags do not have engineering units, however, so this property can also be used to supplement the normal tag description.

### History

The following list describes the history properties of String tags. These properties determine how the project saves historical data for the selected tag.

> **Tip:** You can also use a Trend worksheet to set the history properties of a project tag. This dialog box reflects any changes you make in that worksheet, and vice versa. For more information, see Trend worksheet on page 413.




**History Enabled**

Enable the saving of historical data for this tag.

**Group Number**

The Trend group/worksheet to which this project tag is assigned.

## *Complete list of tag properties*

This is a complete list of all tag properties that are supported by the project tags database. Please note that some properties do not apply to all data types.

| Property Name | Description | R or R/W | Data Type of Property | Data Type of Tag… | | | | Retain |
|---|---|---|---|---|---|---|---|---|
| | | | | Bool | Int | Real | Str | |
| Name | The name of the tag, as configured in the Project Tags datasheet. | R | String, up to 255 chars | # | # | # | # | n/a |
| MemberName | The name of the class member in a properly configured class. The syntax must be:<br>***class.member->MemberName***<br>Example: **`Tank.Lvl->MemberName`** returns `Lvl`. | R | String, up to 255 chars | # | # | # | # | n/a |
| Size | Array size. (An array is any project tag of Size greater than 0.) If the tag is not an array, this returns `0`. | R | Integer | # | # | # | # | n/a |

| Property Name | Description | R or R/W | Data Type of Property | Data Type of Tag… | | | | Retain |
|---|---|---|---|---|---|---|---|---|
| | | | | Bool | Int | Real | Str | |
| Index | The index number of an element in an array. The syntax must be:<br>***tagname*[*index*]->Index**<br>Example: **Tag[1]->Index** returns 1. | R | Integer | # | # | # | # | n/a |
| Description | The description of the tag, as configured in the Project Tags datasheet. | R | String, up to 128 chars | # | # | # | # | # |
| Quality | Tag quality, which can be one of the following:<br>• BAD (0–63)<br>• UNCERTAIN (64–127)<br>• N/A (128–191)<br>• GOOD (192–255)<br>The project runtime updates this property whenever the tag receives a value returned by an expression or received from a communication task (such as driver or OPC).<br>If the expression is invalid (such as, division by zero) or if there is a reading communication error associated with the tag, then the project sets the quality to BAD. | R | Integer | # | # | # | # | |
| TimeStamp | Time and date when the value of the tag last changed. | R | String | # | # | # | # | |
| SendEveryState | This property can have one of the following possible values:<br>• 0: Only the most recent change in the tag value is sent from a project thin client to the project runtime server.<br>• 1: Every change in the tag value is sent from a project thin client to the project runtime server.<br>This property is automatically set to 1 if the tag is used in any Driver or OPC worksheet. It ensures all changes are communicated from clients through the server to connected devices.<br>Otherwise, when this property is set to 0, the tag value is synchronized between client and server at the normal synchronization rate, and only the most recent change before synchronization is sent from the client to the server.<br>**Note:** If you use a tag in a Driver or OPC worksheet, so that its SendEveryState property is set to 1, be careful about how you use the same tag in scripts that run on the client. A script that causes a large number of tag value changes in a short period of time — for example, a script that uses the tag as a loop counter — might fill the client/server communication buffer and affect run-time | R | Boolean | # | # | # | # | |

| Property Name | Description | R or R/W | Data Type of Property | Data Type of Tag... | | | | Retain |
|---|---|---|---|---|---|---|---|---|
| | | | | Bool | Int | Real | Str | |
| | performance, particularly if the client station is an embedded device with limited memory. | | | | | | | |
| Blocked | This property can have two values:<br>• 0: The tag is unblocked and all runtime tasks can access it normally.<br>• 1: The tag is blocked and all runtime tasks will ignore it. It is effectively removed from the project database.<br>This is useful when you want to dynamically disable all actions associated with a specific tag. Even when a tag is blocked, however, it still counts towards the total number of tags used for licensing purposes. | R/W | Boolean | # | # | # | # | |
| Unit | A brief description (up to 7 characters) of the engineering unit (i.e., the unit of measurement) for the tag value. For example, `Kg`, `BTU`, `psi`. | R/W | String, up to 7 chars | # | # | # | # | # |
| Max | The maximum value that can be stored in the tag during run time. | R/W | Real | | # | # | | # |
| Min | The minimum value that can be stored in the tag during run time | R/W | Real | | # | # | | # |
| B0 … B31 | Boolean value (0 or 1) of any of the 32 bits (b0, b1, b2, … b31) of an Integer tag. (B0: LSB, B31: MSB) | R/W | Boolean | | # | | | |
| DisplayValue | A converted tag value that is only displayed on-screen:<br>DisplayValue = (Value / UnitDiv) + UnitAdd<br>This is used when the actual tag values have one Engineering Unit (see Unit above) but need to be displayed on-screen in another Engineering Unit (see DisplayUnit below). For example, Celsius degrees and Farenheit degrees.<br>If user input changes DisplayValue during run time, then the conversion is reversed before the change is actually written to the tag:<br>Value = (DisplayValue − UnitAdd) * UnitDiv | R/W | Real | | # | # | | n/a |
| DisplayUnit | A brief description (up to 9 characters) of the Engineering Unit for DisplayValue.<br>**Note:** This property can only be set by using the functions `SetDisplayUnit` and `SetTagDisplayUnit`. | R | String, up to 9 chars | | # | # | | |
| UnitDiv | Number by which the tag value is divided to get DisplayValue. To perform no division, UnitDiv should be 1.<br>**Note:** This property can only be set by using the functions | R | Real | | # | # | | |

| Property Name | Description | R or R/W | Data Type of Property | Data Type of Tag... | | | | Retain |
|---|---|---|---|---|---|---|---|---|
| | | | | Bool | Int | Real | Str | |
| | SetDisplayUnit and SetTagDisplayUnit. | | | | | | | |
| UnitAdd | Number added to the tag value to get DisplayValue. To perform no addition, UnitAdd should be 0.<br>**Note:** This property can only be set by using the functions SetDisplayUnit and SetTagDisplayUnit. | R | Real | | # | # | | |
| DisplayMax | The maximum value that can be input to DisplayValue during run time:<br>DisplayMax = (Max / UnitDiv) + UnitAdd<br>If DisplayMax is changed during run time, then Max is also changed as follows:<br>Max = (DisplayMax − UnitAdd) * UnitDiv | R/W | Real | | # | # | | |
| DisplayMin | The minimum value that can be input to DisplayValue during run time:<br>DisplayMin = (Min / UnitDiv) + UnitAdd<br>If DisplayMin is changed during run time, then Min is also changed as follows:<br>Min = (DisplayMin − UnitAdd) * UnitDiv | R/W | Real | | # | # | | |
| HiHiLimit | Limit value for the HiHi alarm. | R/W | Real | | # | # | | # |
| HiLimit | Limit value for the Hi alarm. | R/W | Real | | # | # | | # |
| LoLimit | Limit value for the Lo alarm. | R/W | Real | | # | # | | # |
| LoLoLimit | Limit value for the LoLo alarm. | R/W | Real | | # | # | | # |
| RateLimit | Limit value for the Rate alarm. | R/W | Real | | # | # | | # |
| DevSetpoint | Set point for Deviation alarms.<br>**Note:** When referenced using the syntax *tagname->property*, this property is read-only and then only if it is a literal value. As an alternative, you can configure a project tag for the set point and then read/write the value of that tag. For more information, see Properties of Integer and Real tags on page 171. | R | Real | | # | # | | n/a |
| DevPLimit | Limit value for the Deviation+ alarm. | R/W | Real | | # | # | | # |
| DevMLimit | Limit value for the Deviation- alarm. | R/W | Real | | # | # | | # |
| HiHi | If 0, the HiHi alarm is not active. If 1, the HiHi alarm is active. | R | Boolean | # | # | # | | n/a |
| Hi | If 0, the Hi alarm is not active. If 1, the Hi alarm is active. | R | Boolean | # | # | # | | n/a |
| Lo | If 0, the Lo alarm is not active. If 1, the Lo alarm is active. | R | Boolean | # | # | # | | n/a |

| Property Name | Description | R or R/W | Data Type of Property | Data Type of Tag... | | | | Retain |
|---|---|---|---|---|---|---|---|---|
| | | | | Bool | Int | Real | Str | |
| LoLo | If 0, the LoLo alarm is not active. If 1, the LoLo alarm is active. | R | Boolean | # | # | # | | n/a |
| Rate | If 0, the Rate alarm is not active. If 1, the Rate alarm is active.<br><br>**Note:** When this alarm is enabled on a Boolean tag, it is also known as the Changed alarm. The alarm becomes active whenever the tag value changes (i.e., from 0 to 1, or from 1 to 0), but it is also immediately normalized because that is the expected behavior for Boolean tags. To make this alarm useful for Boolean tags, the **Ack Required** option should be selected in the appropriate Alarm worksheet. | R | Boolean | # | # | # | | n/a |
| DevP | If 0, the Deviation+ alarm is not active. If 1, the DevP alarm is active. | R | Boolean | | # | # | | n/a |
| DevM | If 0, the Deviation- alarm is not active. If 1, the DevM alarm is active. | R | Boolean | | # | # | | n/a |
| AlrStatus | Integer value with the status of the current active alarms associated to the tag. Each bit of this integer value indicates a specific status:<br>• Bit 0 (LSB): HiHi Alarm active<br>• Bit 1: Hi Alarm active<br>• Bit 2: Lo Alarm active<br>• Bit 3: LoLo Alarm active<br>• Bit 4: Rate Alarm active<br>• Bit 5: Deviation+ Alarm active<br>• Bit 6: Deviation- Alarm active<br>Examples: If `Tag->AlrStatus` returns 2, it means that the Hi alarm is active. If it returns 3, it means the HiHi and Hi alarms are active simultaneously.<br>If this property returns 0, it means that there are no active alarms associated with this tag.<br>For Boolean tags, only the values 1 (bit 1), 4 (bit 2) or 16 (bit 4) can be returned. | R | Integer | # | # | # | | |
| Ack | This property can have two values:<br>• 0: There are no alarms on this tag that require acknowledgment.<br>• 1: There is at least one alarm on this tag that requires acknowledgment.<br>This works as a global acknowledge for the tag and goes to 0 only when all alarms on the tag have been acknowledged. | R | Boolean | # | # | # | | |
| UnAck | This property can have two values: | R/W | Boolean | # | # | # | | |

| Property Name | Description | R or R/W | Data Type of Property | Data Type of Tag... | | | | Retain |
|---|---|---|---|---|---|---|---|---|
| | | | | Bool | Int | Real | Str | |
| | • 0: There is at least one alarm on this tag that requires acknowledgment.<br>• 1: There are no alarms on this tag that require acknowledgment.<br>If you manually set this value to 1, then the active alarms (if any) are acknowledged. The value of this property is always the opposite of the Ack property. | | | | | | | |
| AlrAckValue | Text associated with the Acknowledged state of a Boolean tag. This text is displayed in the Value column of an Alarm/Event Control.<br>You can also edit this text in the *Tag Properties* dialog box for the Boolean tag. For more information, see Properties of Boolean tags on page 176. | R/W | String, up to 32 chars | # | | | | # |
| AlrOffValue | Text associated with the Normalized state of a Boolean tag. This text is displayed in the Value column of an Alarm/Event Control.<br>You can also edit this text in the *Tag Properties* dialog box for the Boolean tag. For more information, see Properties of Boolean tags on page 176. | R/W | String, up to 32 chars | # | | | | # |
| AlrOnValue | Text associated with the Active state of a Boolean tag. This text is displayed in the Value column of an Alarm/Event Control.<br>You can also edit this text in the *Tag Properties* dialog box for the Boolean tag. For more information, see Properties of Boolean tags on page 176. | R/W | String, up to 32 chars | # | | | | # |
| AlrDisable | This property can have two values:<br>• 0: The alarms associated with this tag are enabled. This means that when an alarm condition occurs, the alarm will become active.<br>• 1: The alarms associated to this tag are disabled. This means that even if an alarm condition occurs, the alarm will not become active. | R/W | Boolean | # | # | # | | |

## Tag Integration

The following tag properties are available only for shared tags that you have imported into your project via Tag Integration.

Also, these tag properties are available only on the project runtime server, which means they can be used only in scripts and worksheets that are executed on the server. If these properties are referenced on a client, they have no value. This limitation is by design — these properties are kept in the server's memory only to facilitate communication between the server and the Tag Integration source, and it would require additional system and network resources to make these properties available on all connected clients. To work around this limitation: create new project tags that correspond to the properties you want to reference; make sure the scope of those tags is set to **Server**, so they can be used on both the server and the clients; and then set the values of those tags to be equal to the actual properties.

| Property Name | Description | R or R/W | Data Type of Property | Data Type of Tag... | | | | Retain |
|---|---|---|---|---|---|---|---|---|
| | | | | Bool | Int | Real | Str | |
| Source | The source name, as specified in the Tag Integration settings. The default is DEV. | R | String | # | # | # | # | n/a |
| Address | The I/O address of the shared tag. Example: `.step` | R | String | # | # | # | # | n/a |
| CommUrl | A uniform resource locator (URL) that describes the location of the shared tag. This URL includes the source name, the I/O address, Div, Add, Action (Read, Write, Read/Write), and Scan (Screen, Always, Auto). Examples:<br>`SOURCE:// DEV/.steps/1/0/ ReadWrite/Auto/.steps`<br>`SOURCE://DEV1/ PLC_PRG.OPCTEST.ArrayWordTest_4_/1/0/ ReadWrite/Auto/ PLC_PRG.OPCTEST` | R | String | # | # | # | # | n/a |
| OriginalTag | The actual name of the PLC register or variable. Example: for a project tag named…<br>`DEV1..ZZ_Array3Dim_0_1_[2]`<br>…the actual name is…<br>`.ZZ_Array3Dim[0,1,2]` | R | String | # | # | # | # | n/a |
| Used | This property is set to 1 if the shared tag is being used in your project in a screen, script, etc. Otherwise, it is set to 0.<br>Please note that each element in an array has its own Used property, so you can use this property to create virtual groups of array elements that are actually being used in your project. | R | Boolean | # | # | # | # | n/a |

### Notes

- If a property is marked "n/a" with regards to being retentive, it is because either the property is inherent in the tag definition (e.g., Name, Size) or the value of the property is continuously derived during run time (e.g., alarm activation, DisplayValue).
- To enable retention of a tag's properties, select the **Retentive Parameters** option for that tag.
- You cannot use tag properties (such as Bit fields) to configure *Alarm* or *Trend* worksheets.

## *Change how out-of-range tag values are handled*

A project tag will occasionally receive a value that is outside of its normal range. You can change how these out-of-range values are handled during run time.

> **Note:** This topic applies only to Integer and Real tags, because they are the only tags for which the Min and Max properties have meaning. Boolean tags can only have values of 0 and 1, and String tags do not have numeric values at all.

A project tag has a range of possible values, and that range is determined by the tag's Min and Max properties. By default, if the tag receives a new value that is outside of its range — that is, if the value is less than the minimum or greater than the maximum — then the received value is ignored, the existing value is retained, the tag quality is set to UNCERTAIN, and a warning message is sent to the *Output* window.

In some situations, however, it is useful to change how these out-of-range values are handled — in particular, you might need to emulate how out-of-range values are handled by another vendor's hardware or software.

To change this behavior, you must manually edit your project file (e.g., *<project name>*.app) to add or modify the following settings:

```
[Options]
WriteOutOfRange={FALSE|TRUE}
CapOutOfRange={FALSE|TRUE}
```

**WriteOutOfRange**

By default, this setting is FALSE. When this setting is TRUE, out-of-range tag values are accepted as good (i.e., the tag quality is not set to UNCERTAIN).

**CapOutOfRange**

By default, this setting is FALSE. When this setting is TRUE, out-of-range tag values are capped at the tag's minimum and maximum.

> **Note:** If a setting is not present in the project file, then its default value is used.

To change how out-of-range tag values are handled:

1. Locate your project file, which is typically at: EcoStruxure Machine SCADA Expert 2020 Projects \<project name>\<project name>.app
2. Open the project file with a standard text editor, such as Notepad, and then add or modify the settings **WriteOutOfRange** and **CapOutOfRange**.
   The following table shows how the two settings work together:

| Settings | Behavior |
|---|---|
| **[Options]**<br>**WriteOutOfRange=FALSE**<br>**CapOutOfRange=FALSE** | The received value is ignored, the existing value is retained, the tag quality is set to UNCERTAIN, and a warning message is sent to the *Output* window.<br>This is the default behavior. |
| **[Options]**<br>**WriteOutOfRange=FALSE**<br>**CapOutOfRange=TRUE** | If the received value is lower than Min, then the tag value is set to Min, and if the received value is greater than Max, then the tag value is set to Max. The tag quality is set to UNCERTAIN. |
| **[Options]**<br>**WriteOutOfRange=TRUE**<br>**CapOutOfRange=FALSE** | Min and Max are ignored when setting tag values; the tag value is set to whatever value it receives, and the tag quality is set to GOOD.<br>Min and Max are still applied to linear conversions in device communication. |
| **[Options]**<br>**WriteOutOfRange=TRUE**<br>**CapOutOfRange=TRUE** | If the received value is lower than Min, then the tag value is set to Min, and if the received value is greater than Max, then the tag value is set to Max. The tag quality is set to GOOD. |

3. Save and close the project file.

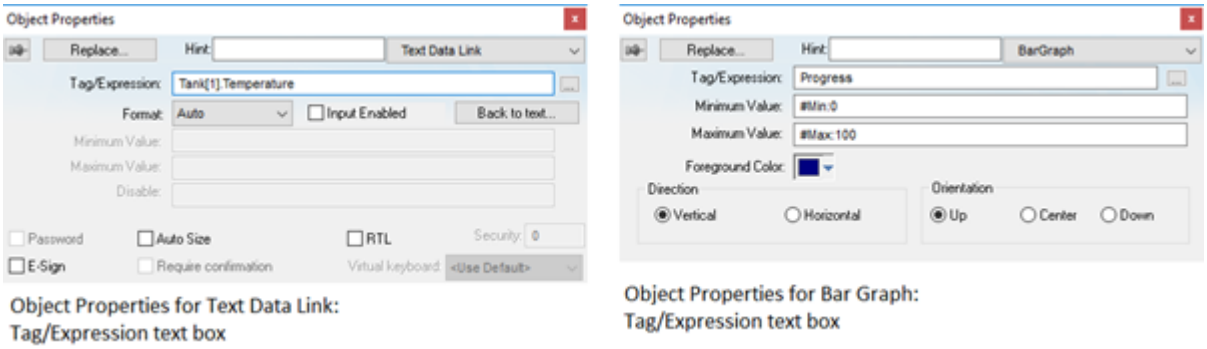
## Using Tags in Your Project

Once you have added a tag to the project database, you can use that tag in your project by associating it to objects on a screen.

The basic process for associating tag to screen objects consists of the following steps:

1. In the project screen, select the object to which you want to apply the tag.
2. Click one of the buttons in the Animations group to apply that animation to the object.
3. Double-click on the object to open its *Object Properties* dialog.
4. Locate the **Tag** text box for that property and type the tag name into the field.

   Tag text box names and locations will vary, depending on the type of property you are using. For example:




***Applying Tags to an Object***

Comprehensive instructions for applying tags to screen objects are provided throughout the documentation where appropriate.

## Deleting a tag from the project database

Delete a tag that is no longer in use by deleting its line in the Project Tags or Shared Database datasheet.

Before you delete a tag, we strongly recommend that you use the **Cross-Reference** tool to make sure the tag is not being used anywhere in your project. (If you delete a tag that is still being used, then you will not be able to verify and run your project.) Fix any screens or worksheets where the tag is being used before you proceed.

> **Note:** This task applies to both the Project Tags and Shared Database datasheets.

To delete a tag:

1. Stop the project if it is running.
2. Open the datasheet for editing.
3. In the datasheet, find the line for the tag you want to delete.
4. Right-click the line, and then select **Delete Line** from the shortcut menu.
   If the option is disabled, then you may need to clear any sorting or filtering that you previously applied to the datasheet.
   An alert dialog is displayed asking you to confirm the action.
5. Click **Yes**.
   The line is deleted from the datasheet.
6. Save and close the datasheet.

## Sort or filter the rows in a worksheet

Sort or filter the rows in a worksheet in order to make it easier to browse the rows or find a specific item.

Before you begin this task, you must have already inserted a worksheet and opened it for editing. You should also be familiar with how sorting and filtering is done in general-purpose spreadsheet applications.

Please note that you can sort or filter rows only in the following types of worksheets:

- The Project Tags, Shared Tags, and System Tags datasheets;
- The Translation Table worksheet;
- All task worksheets except Report and Script, which do not have rows; and
- All communication worksheets.

None of the other worksheets have rows to sort or filter.

Sorting is done alphanumerically, by the selected column, in either ascending (0–9, A–Z) or descending (Z–A, 9–0) order.

| | Tag Name | Type |
|---|---|---|
| | Filter text | ... |
| 1 | Tag1 | HiHi |
| 2 | Tag1 | Hi |
| 3 | Tag1 | Lo |
| 4 | Tag1 | LoLo |
| 5 | Tag2 | HiHi |
| 6 | Tag2 | Hi |
| 7 | Tag2 | Lo |
| 8 | Tag2 | LoLo |
| 9 | Tag3 | HiHi |
| 10 | Tag3 | Hi |
| 11 | Tag3 | Lo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

***Alarm worksheet rows in their original order***

| | Tag Name | Type |
|---|---|---|
| | Filter text | ... |
| 10 | Tag3 | Hi |
| 2 | Tag1 | Hi |
| 6 | Tag2 | Hi |
| 9 | Tag3 | HiHi |
| 5 | Tag2 | HiHi |
| 1 | Tag1 | HiHi |
| 3 | Tag1 | Lo |
| 11 | Tag3 | Lo |
| 7 | Tag2 | Lo |
| 8 | Tag2 | LoLo |
| 4 | Tag1 | LoLo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

***Alarm worksheet rows sorted by Type***

Filtering is done according to whatever string you enter in the selected column. Only the rows that match the string will be displayed.

| | Tag Name | Type |
|---|---|---|
| | Tag3 | ... |
| 9 | Tag3 | HiHi |
| 10 | Tag3 | Hi |
| 11 | Tag3 | Lo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

*Alarm worksheet rows filtered where Tag Name is "Tag3"*

| | Tag Name | Type |
|---|---|---|
| | Filter text | Lo |
| 3 | Tag1 | Lo |
| 7 | Tag2 | Lo |
| 11 | Tag3 | Lo |
| * | | HiHi |
| * | | HiHi |

*Alarm worksheet rows filtered where Type is "Lo"*

> **Tip:** You can still delete rows while they are sorted or filtered.

To sort or filter rows:

1. To sort the rows, click the header of the column by which you want to sort. Click once to sort in ascending order, and then click again to sort in descending order.

   The current order (i.e., the direction of the sort) is indicated by the arrow to the right of the column name.

   > **Note:** You cannot sort by multiple columns.

2. To undo the sorting and restore the rows to their original order, click the header of the first (numbered) column.
3. To filter the rows, type the string that you want to match in the top (zero) row of the worksheet and then press either **Tab** or **Return**.

   You may include `*` and `?` as wildcard characters in your string:

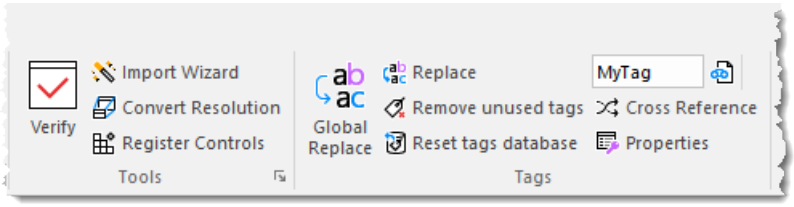
   - `*` matches any number of characters, including none. For example, `Tag*` would match **`Tag`**, **`Tag3`**, **`Tag34567`**, **`TagA`**, and **`Tag_TEMP`**.
   - `?` matches exactly one character. For example, `Tag?` matches **`Tag3`** and **`TagA`**, while `Tag?????` matches **`Tag34567`** and **`Tag_TEMP`**.

   Also, you may filter by multiple columns. Only the rows that match the filter strings in all columns will be displayed.
4. To undo the filtering and restore the rows to their original order, delete the string that you typed and then press either **Tab** or **Return**.

Please keep in mind that sorting or filtering the rows of a worksheet only helps you to edit that worksheet. It does not change how the worksheet is executed during run time. The rows will be executed in their original numbered order (i.e., the leftmost column) unless you actually move or delete a row.

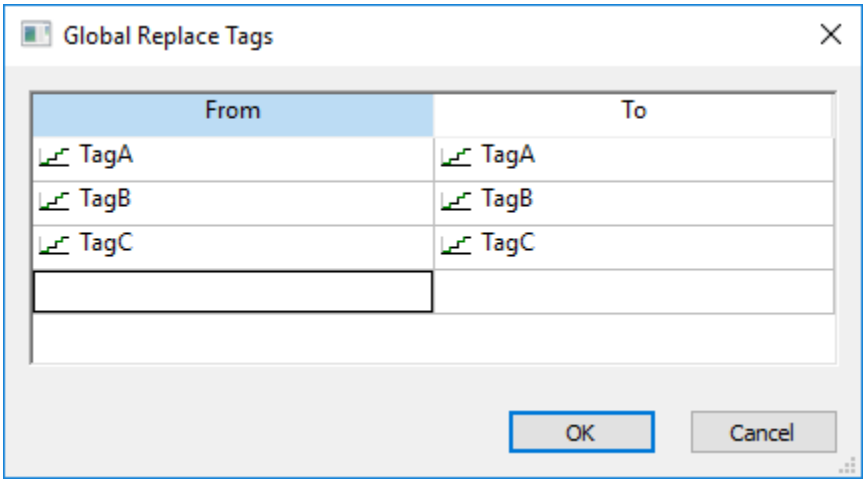
## Using the Tags tools

The **Home** tab of the ribbon provides several tools that you can use to find and manage project tags, tag properties, and functions.




### *Global Replace Tool*

When clicking on the **Global Replace** tool from the Tag Properties Toolbar, the following window displays:




***Global Replace dialog***

From the *Global Replace* dialog, you can replace any tag(s) from all documents (screens and worksheets) of the whole project. You can edit both the **From** and the **To** column.

When replacing composed tags (array size > 0 and/or Type = Class), you can configure a specific array position (for example, **`TagA[1]`** ) or class member (for example, **`TagB.MemberX`** ) or both (for example, **`TagC[3].MemberY`** ). If you configure only the Main Tag Name (for example, **`TagC`** ) in the **From** column, all tags from this main tag will be modified for the tag configured in the **To** column.

If an invalid replacement is configured (for example, replace the **Main Tag** tag from a class type tag for a simple tag (not a class tag), the **OK** button will be disabled. When the **OK** button is pressed, the tags configured on the *Global Replace* dialog will be replaced in the order that they were configured on the dialog interface.

> **Note:** You must close all documents (screens and worksheets) before executing this command.

When changing the tag name on the Project Tags database worksheet, EcoStruxure Machine SCADA Expert will ask you if you intend to replace this tag through the whole project.

The **Replace** option will be created in the **Edit** menu. By using this option, the *Global Replace* dialog is prompted, however, the changes are applied only the current screen or worksheet in focus.

### *Replacing project tags in a document or screen object*

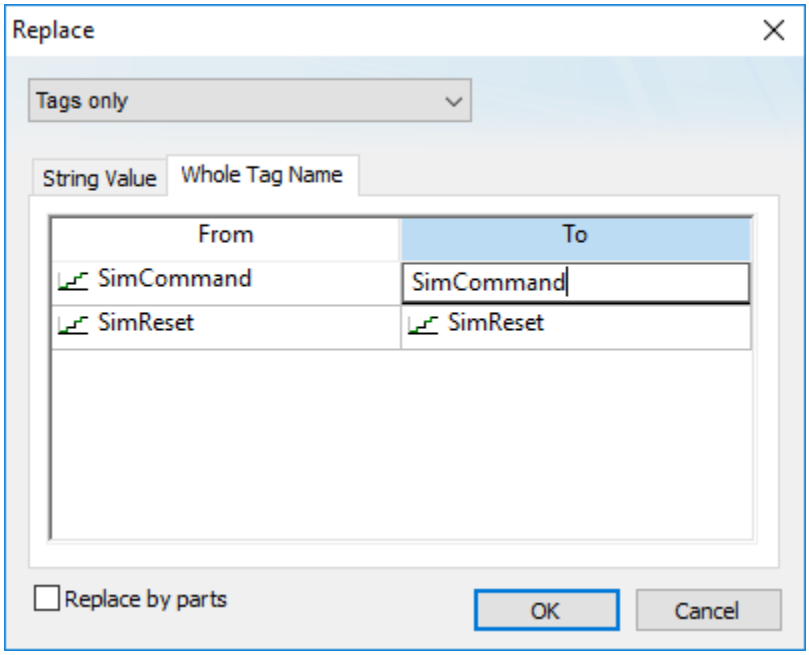
To replace all occurences of a tag in the current document, do one of the following:

- On the Home tab of the ribbon, in the Tags group, click **Replace**; or
- On the Graphics tab of the ribbon, in the Editing group, click **Replace**.

To replace all occurences of a tag in a screen object, double-click the object to open its *Object Properties* dialog and then click **Replace**.

All of these methods will open the *Replace* dialog, which is descibed below.

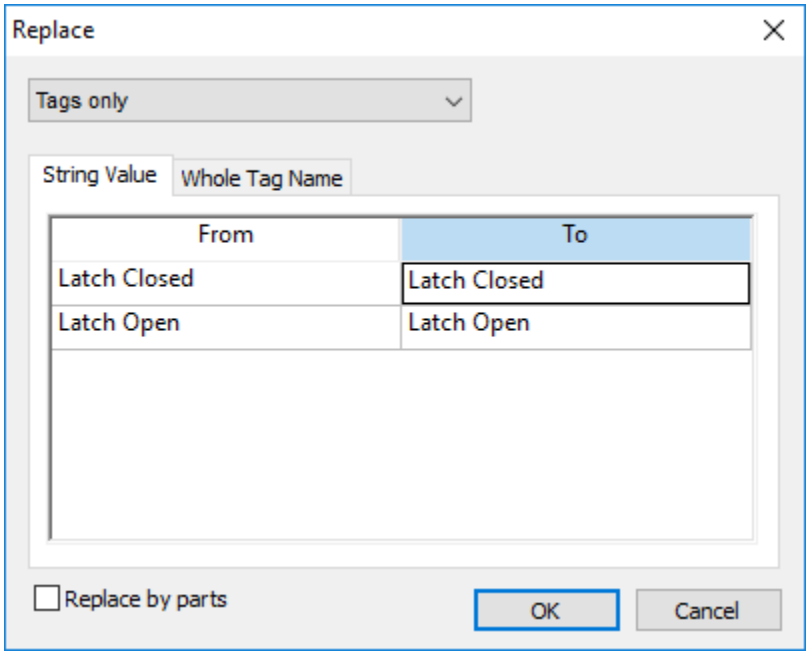
You can replace one or more tags by clicking the **Whole Tag Name** tab. Current tags used are displayed. The original tag names are shown in the **From** column on the left, and you can enter your new tag names in the **To** column on the right.




***Whole Tag Name tab***

Note that this does not *rename* or *delete* any tag — it only replaces the tags used in the object with other tags from the database.

You can also replace one or more strings (e.g., button captions, descriptive text) by clicking the **String Value** tab.




***String Value tab***

When you are done, click **OK**.

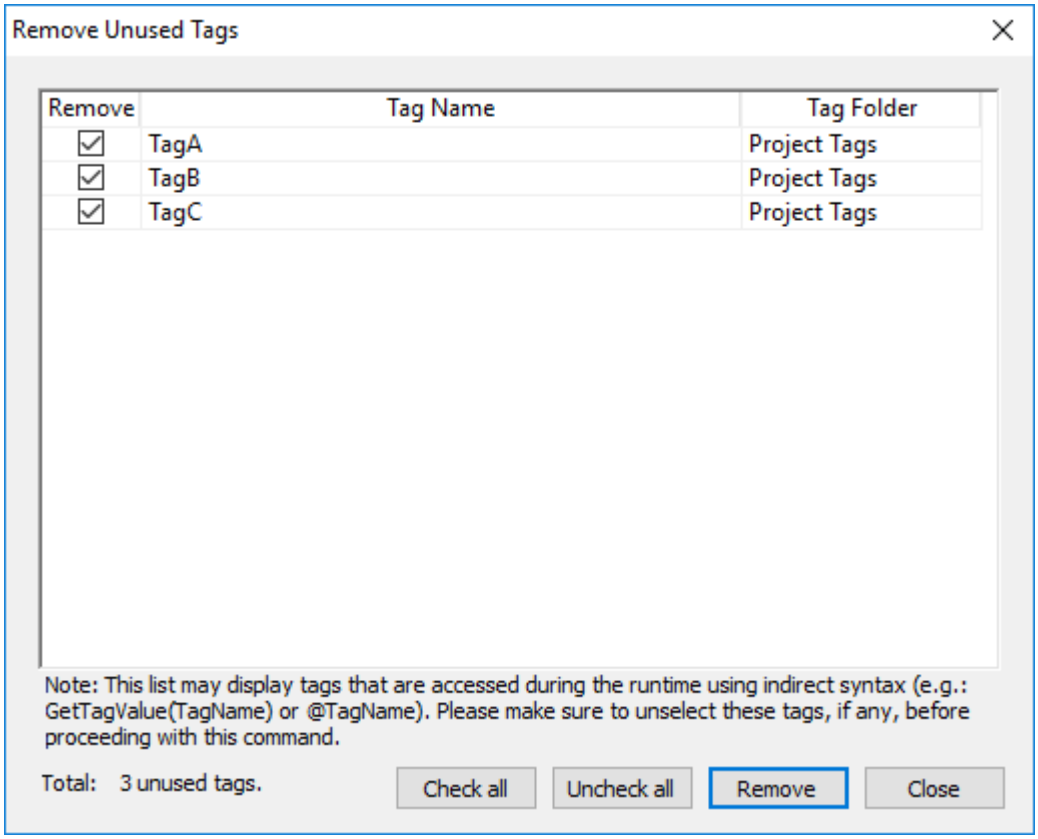
### *Removing unused tags from the project database*

The **Remove unused tags** tool is used to scan the project database for unused tags, which you can then select and remove.

"Unused tags" are tags that you have defined in the project database but have not used in any screen or task worksheet. Since your project has a limited number of available tags (as determined by your product/license type), you may want to remove some or all of these unused tags to decrease your project's tag count.

1. Save and close all open project screens and worksheets.
2. On the **Home** tab of the ribbon, in the **Tags** group, click **Remove unused tags**.
   The development application automatically verifies your project, and if it finds any unused tags, then it lists them in the *Remove Unused Tags* dialog.

> **Note:** The listed tags may include some that are accessed during runtime using indirect syntax (e.g., **`GetTagValue(TagName)`** or **`@TagName`**, where the value of **`TagName`** is the name of an unused tag).




***Unused tags listed in Remove Unused Tags dialog***

3. Determine which tags you want to remove, if any.
   - If you want to remove all of the listed tags, simply click **Remove**.
   - If you want to keep some of the listed tags, clear the **Remove** check boxes on the left for those tags, and then click **Remove**.
   - Click **Check all** or **Uncheck all** to select or clear, respectively, all of the **Remove** check boxes on the left.
   - If you do not want to remove any of the listed tags, click **Close**.

   The development application removes the selected tags and then asks if you want to verify the project again.
4. Click **Yes** to verify the project again.

## *Reset Tags Database*

Select **Reset Tags Database** to "reload" the tags database on the local station. This command affects all tags stored in the *Project Tags* folder. This option is useful for resetting the project tags and restoring the values they had when the project was loaded for the first time. When you stop the project but leave the development environment open, the tags are not reset by default when the project is run again. Therefore, you can execute this command to reset them before the project runs again.

When this command is executed, the **Startup Value** configured for each tag (*Tags Properties* dialog) is written to the respective tag. If you did not configure any **Startup Value** for a numeric tag (**Boolean**, **Integer** or **Real**), the value **0** (zero) is written to the tag. If you did not configure any **Startup Value** for a string tag, the empty value ("") is written to the tag.

This command is disabled (in gray) if there is at least one runtime task running on the local station. You must close all runtime tasks (**Stop** on the Home tab of the ribbon) before this command can be executed.

> **Note:** The tags stored in the *System Tags* folder and in the *Shared Tags* folder (if any) are not affected by this command.

> **Tip:** If you want to reset the project tags automatically whenever you run the project (**Run** on the Home tab of the ribbon), you can check the option **Reset Tags Database** when starting project on the *Preferences* tab of the *Project Settings* dialog.

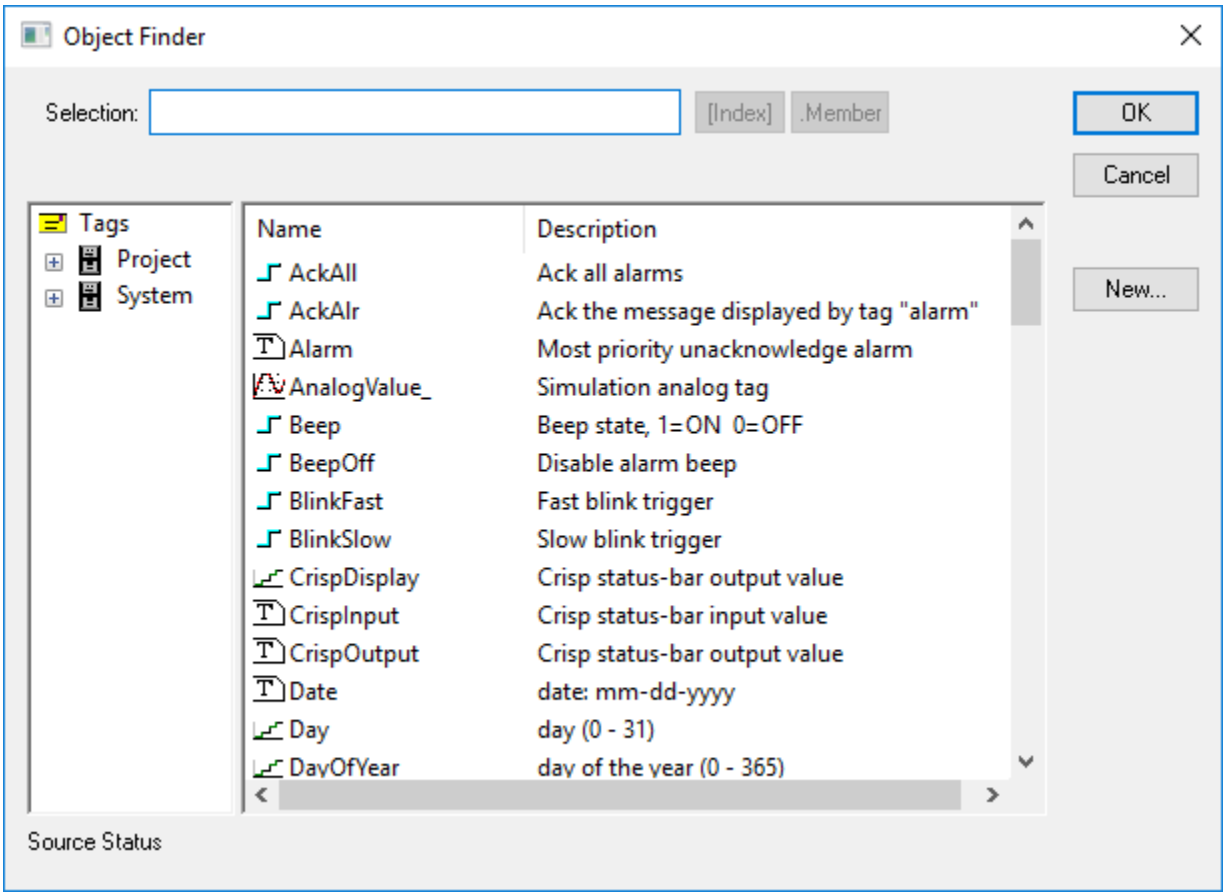
### *Tagname Text Box*

Type a name into the **Tagname** text box

to create a new tag for your project. The **Cross Reference** and **Tag Properties** tools will reference this tag name for their actions.

### *Object Finder Tool*

Click the **Object Finder** tool to open the *Object Finder* dialog, which lists all **Tags** and **Functions** currently configured for the project.



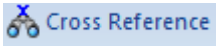

***Object Finder dialog***

To select an existing tag/function, double-click on the tag/function name, and then click **OK** to close the box. The selected name displays in the **Tagname** text box.

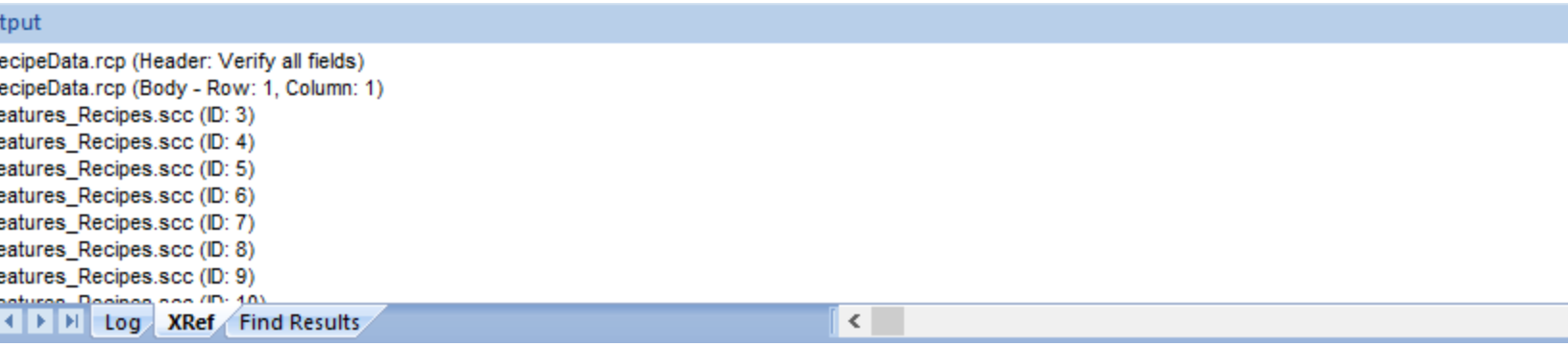
- To select a specific array index, click the **Index** button after specifying the array tag name.

- To select a specific member name, click the **Member** button after specifying the class tag name.
- To create a new tag, click the **New** button.
- When the *New Tag* dialog displays, enter the following information, then click **OK** to close the box:
  - **Name**
  - **Array Size**
  - **Type** (Boolean, Integer, Real, String, Class:Control, Class:msgonline, or Class:Alr)
  - **Description**
  - **Scope** (local or server)

### Cross Reference Tool

Click the **Cross Reference** tool 

to search all project screens and worksheets for the tag noted in the **Tagname** text box. This function writes a log, detailing all the occurrences of the tag, to the **XRef** tab in the Output window. For example, the results of searching for a **`BlinkFast`** tag are as follows:




***XRef Results***

### Set tag properties using the Properties command

Use the **Properties** command to set the properties of project tags.

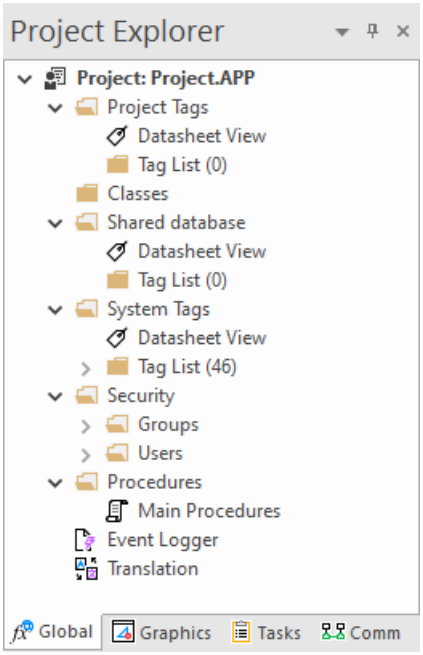
The **Properties** command opens the *Tag Properties* dialog box, which you can use to set any and all of the properties of a selected tag. This includes alarms and history properties, which cannot be set using the Project Tags datasheet. However, when you use the **Properties** command, you can set the properties of only one project tag at a time.

> **Note:** You cannot use the **Properties** command during run time. If you need to edit the tags database during run time, use the Tags Database functions.

To use the **Properties** command to set tag properties:

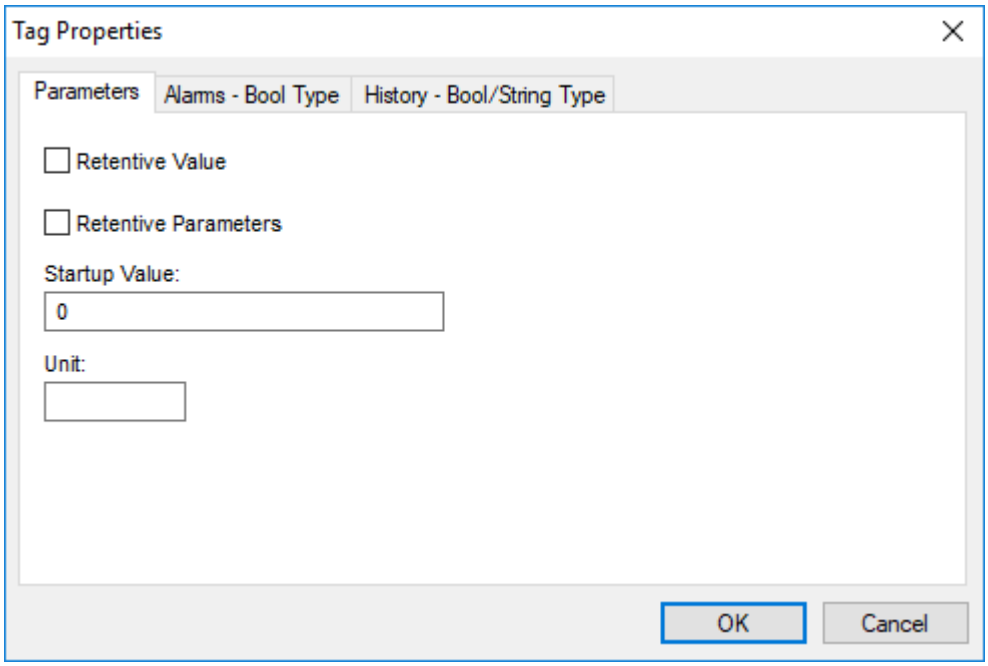
1. At the bottom of the *Project Explorer* window, click **Global**.

   The **Global** tab is displayed.




2. Expand the **Project Tags** folder, and then in that folder, expand the **Tag List** folder. A list of all of the project tags is displayed.
3. In the **Tag List** folder, select the project tag for which you want to set properties.
4. On the **Home** tab of the ribbon, in the **Tags** group, click **Properties**.

   The *Tag Properties* dialog box for the selected tag is displayed.




5. Use the dialog box to set the tag properties as needed, keeping in mind that the properties are distributed among multiple tabs in the dialog box.

Some properties do not apply to all data types, so for more information about the applicable properties, see:

- Properties of Integer and Real tags on page 171
- Properties of Boolean tags on page 176
- Properties of String tags on page 180

6. When you are done, click **OK** to save your changes and close the dialog box.

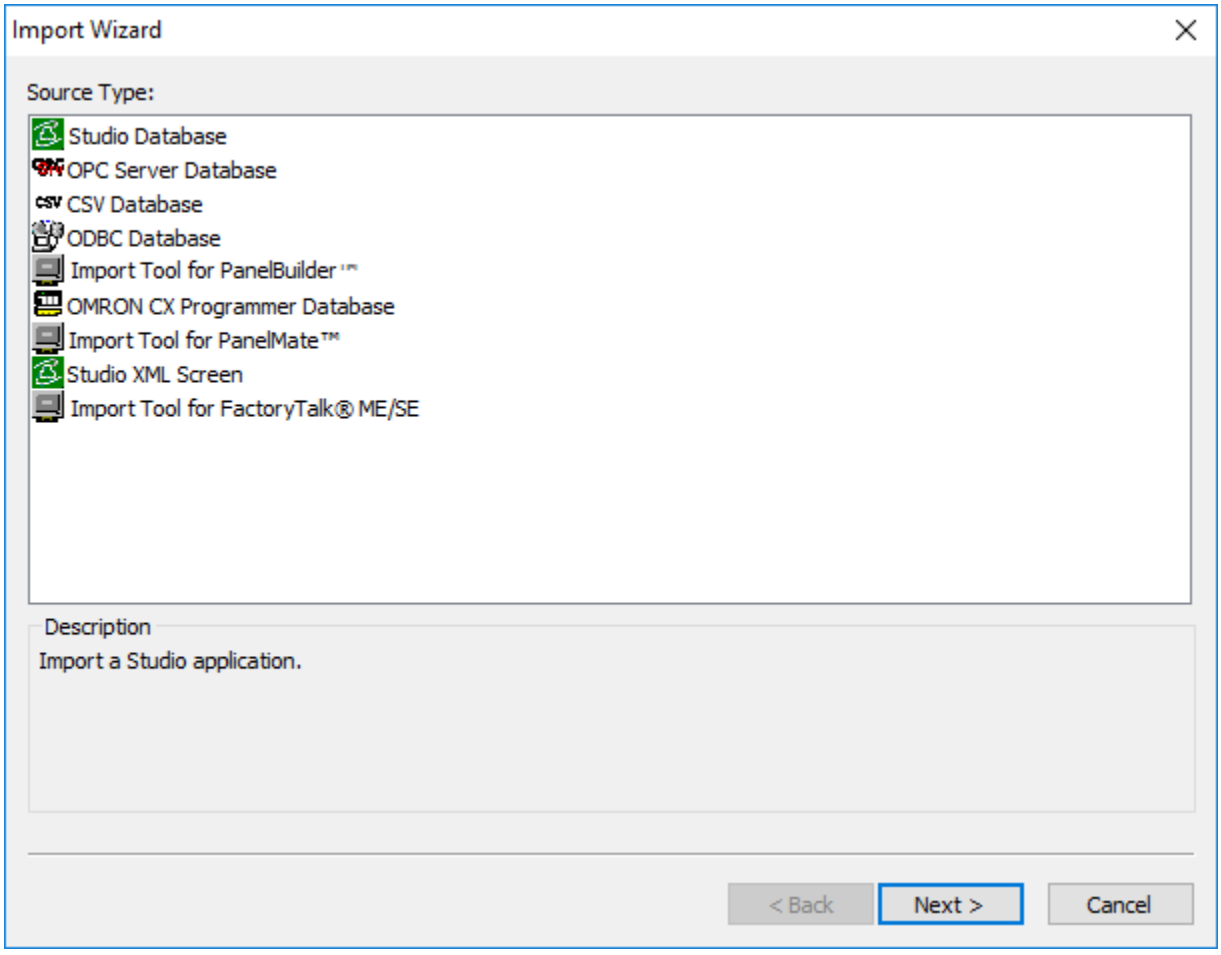
## Import Wizard

The Import Wizard is a powerful tool that reduces engineering time during project development. Using the Import Wizard, you can import tags from different data sources directly to the project tags database. Depending on the data source, you can import not only the tag names, but also the communication interface (the link between the tags and the PLC addresses).

When you click **Import Wizard** on the Home tab of the ribbon, an *Import Database Wizard* dialog displays to step you through the process of importing tags. There are three steps for importing tags from these data source types:

- EcoStruxure Machine SCADA Expert 2020 Project Database
- OPC Server Database
- CSV Database
- ODBC Database
- PanelBuilder32™ Database
- PanelMate Plus™ Database
- FactoryTalk™ Application
- Studio XML Screen

### Step1: Select the Source Type



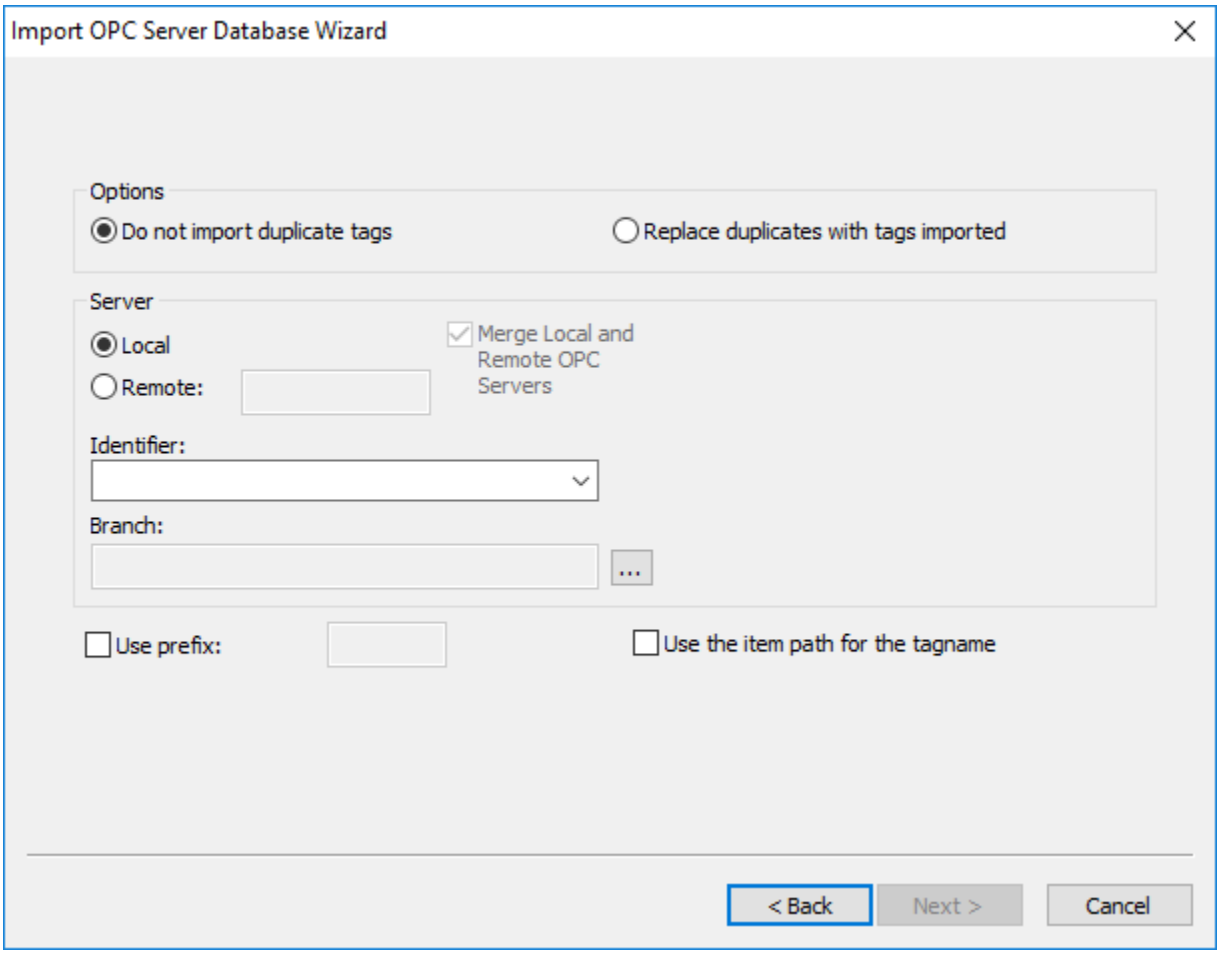

***Import Wizard dialog box***

Click the data Source Type, which is where the tags are being imported from. Click **Next**.

Continue to the appropriate section for the instructions you need to complete the import database procedure:

- Import tags and files from a EcoStruxure Machine SCADA Expert 2020 project database on page 206
- Importing from OPC Server Databases on page 210
- Import tags from a CSV database on page 211
- Importing from ODBC Databases on page 215
- Importing from PanelBuilder32 Databases on page 216
- Importing PanelMate programs on page 217
- Import from a FactoryTalk application on page 219

### Step 2: Configure the Source Type Settings




***Import OPC Server Database Wizard (1/2) dialog box***

Most of the settings in the second window depend on the data Source Type selected in the first step. The screenshot above is an example of one data Source Type (OPC Server Database). The settings that are common for any data Source Type are described below:

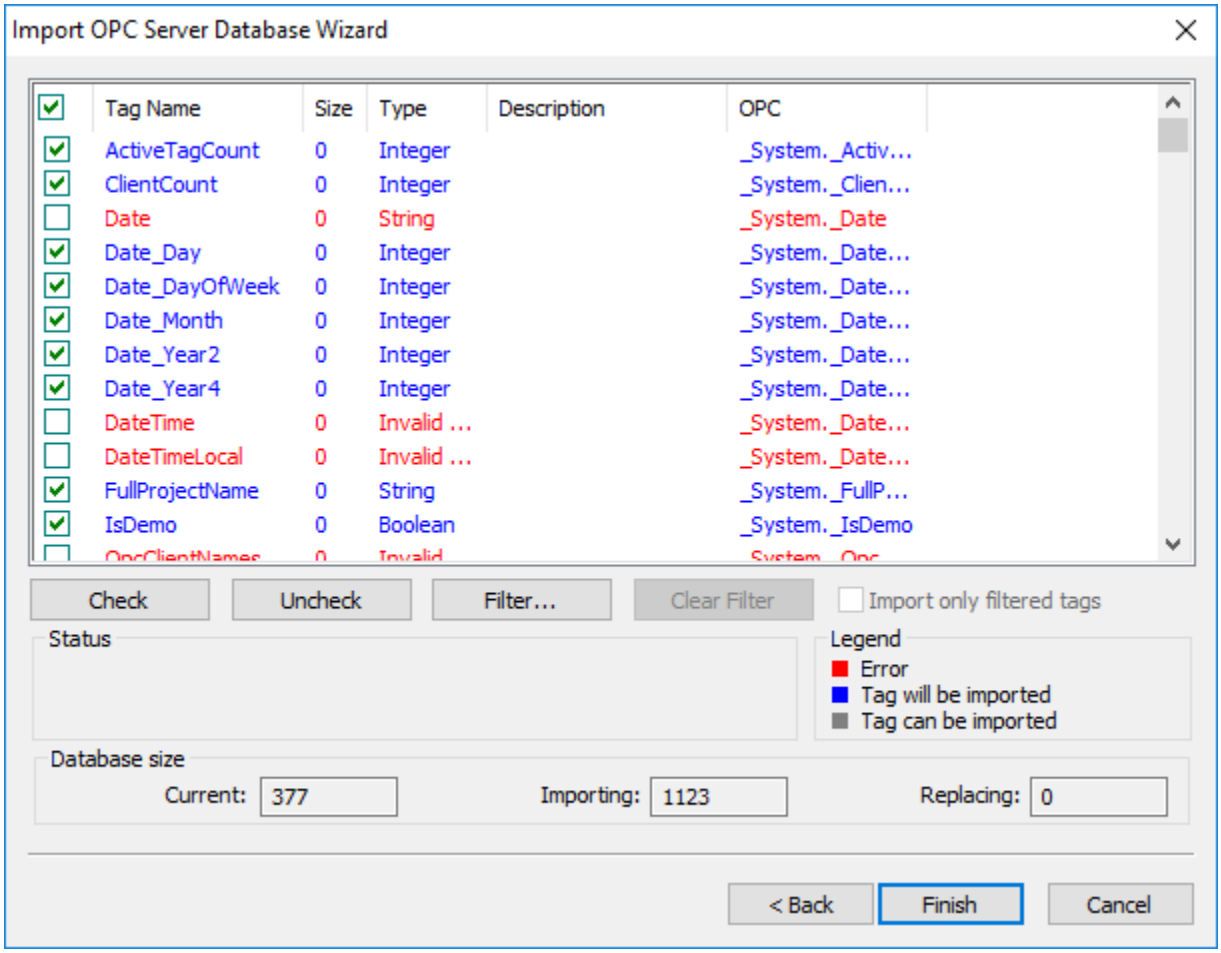
- **Options** box: Select **Do not import duplicated tags** if you do not want imported tags to overwrite tags with the same name that already exist in the Tags Database of the current project. Select **Replace duplicates with tags imported** to overwrite tags in the Tags Database with imported tags of the same name.

- **Use Prefix**: Check to specify a prefix (up to 4 characters) to be concatenated to the name of the imported tags. It is useful to use a prefix to differentiate the imported tags from the tags created manually.

> **Note:** The other settings vary according to the data source selected in the first step, and they are described in the specific sections for each data source type.

After configuring the settings in this dialog, click **Next**.

### Step 3: Filter the tags




***Import OPC Server Database Wizard (2/2) dialog box***

The screenshot above is an example of one data Source Type (OPC Server Database). The fields and settings that are common for all data Source Types include the following:

- **Grid**: Displays the list of tags found on the data source.
  - **checkbox**: Check to import the tag from the data source to the Tags Database of the current project.
  - **TagName**: Name of the tag
  - **Size**: Array size of the tag
  - **Type**: Data type of the tag (Boolean, Integer, Real, String or Class:<ClassName>)
  - **Description**: Description of the tag
- **Check** button: Click to select/import all tags in the grid

- **Uncheck** button: Click to uncheck all tags in the grid
- **Filter** button: Click to filter the tags. The Filter dialog will display, allowing you to specify a mask for each column in the grid. Wild cards (* and ?) can be used to filter data.
- **Clear Filter** button: Click to reset the filter.
- **Import Filtered Tags Only** checkbox: Check this option to import only the tags that are visible in the grid (filtered).
- **Status** box: Displays a message describing the status of the tag currently selected in the grid. This information is especially useful to indicate why a tag cannot be imported.
- **Legend** box: Describes the meaning of the colors that represent tag status:
  - (Red) **Error:**Tag cannot be imported because it is not supported by EcoStruxure Machine SCADA Expert. See the **Status** box for a detailed description of the error.
  - (Blue) **Tag will be imported:**Tag will be imported after you click the Finish button.
  - (Gray) **Tag can be imported:**Tag can be imported but it has not been checked.
- **Database size** box: Displays summary information regarding the current Import Wizard:
  - **Current**: Indicates the number of tags configured in the Project Tags database of the current project
  - **Importing**: Indicates the number of tags selected to be imported
  - **Replacing**: Indicates the number of tags configured in the Project Tags database of the current project that will be replaced by an imported tag with the same name.

After selecting the tags to import, click the Finish button, or click Cancel to abort the operation.

> **Note:** The other settings vary according to the data source selected in the first step, and they are described in the specific sections for each data Source Type (see below).

### *Import tags and files from a EcoStruxure Machine SCADA Expert 2020 project database*

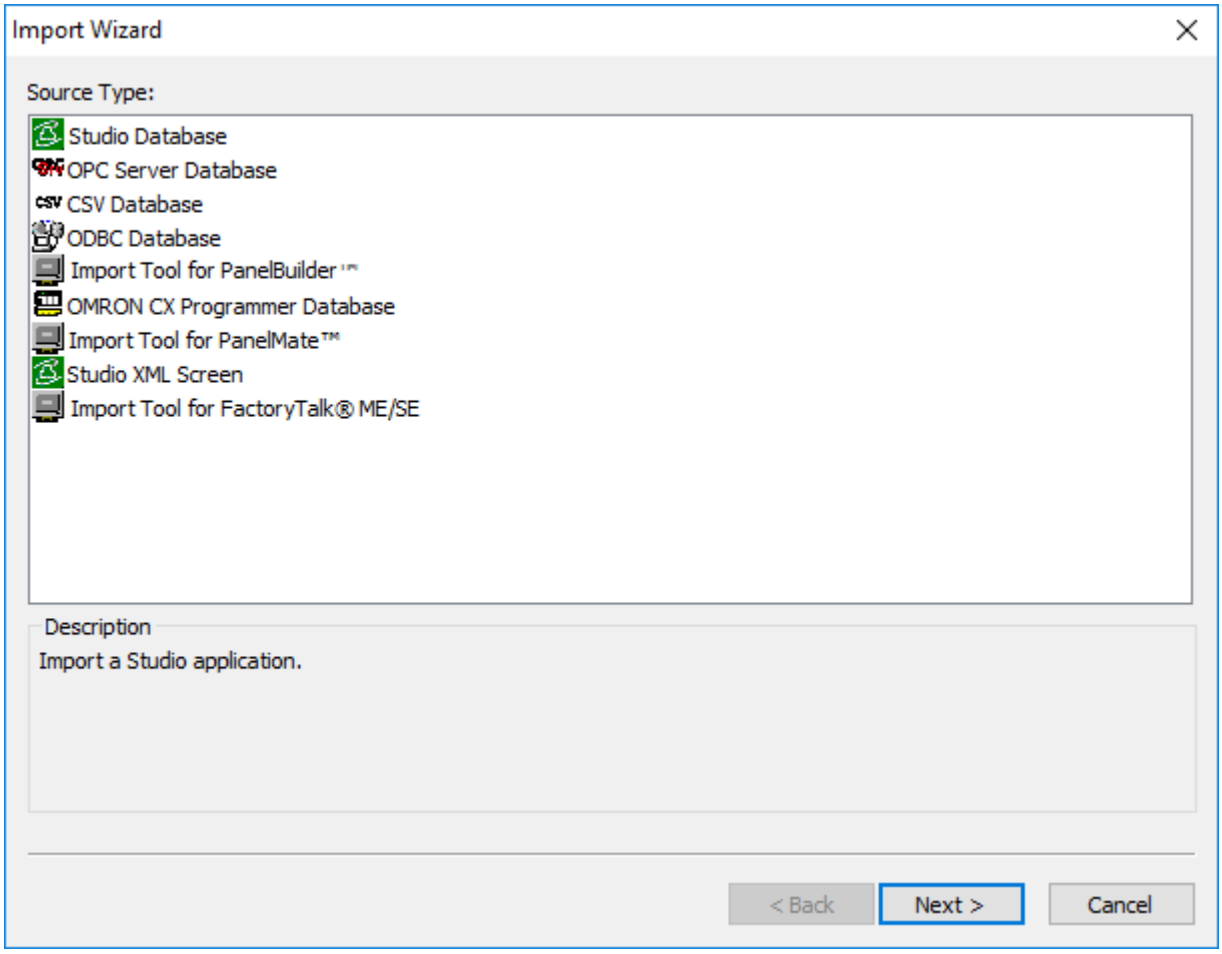
This tool allows you to import tags, screens, and worksheets from another EcoStruxure Machine SCADA Expert 2020 project database, which is useful for merging projects and importing project templates. It can also create a client-server connection between your current project and the imported project running on a remote station, which allows the two projects to share tag values during run time.

> **Note:** If you are running Studio on a Windows computer that has User Account Control (UAC) enabled, you might have issues while performing this task. If you do, try running Studio as an administrator (i.e., right-click the Studio program icon, and then click **Run as administrator** on the shortcut menu).

To import from another EcoStruxure Machine SCADA Expert 2020 project database into your current project:

1. On the **Home** tab of the ribbon, in the **Tools** group, click **Import Wizard**.

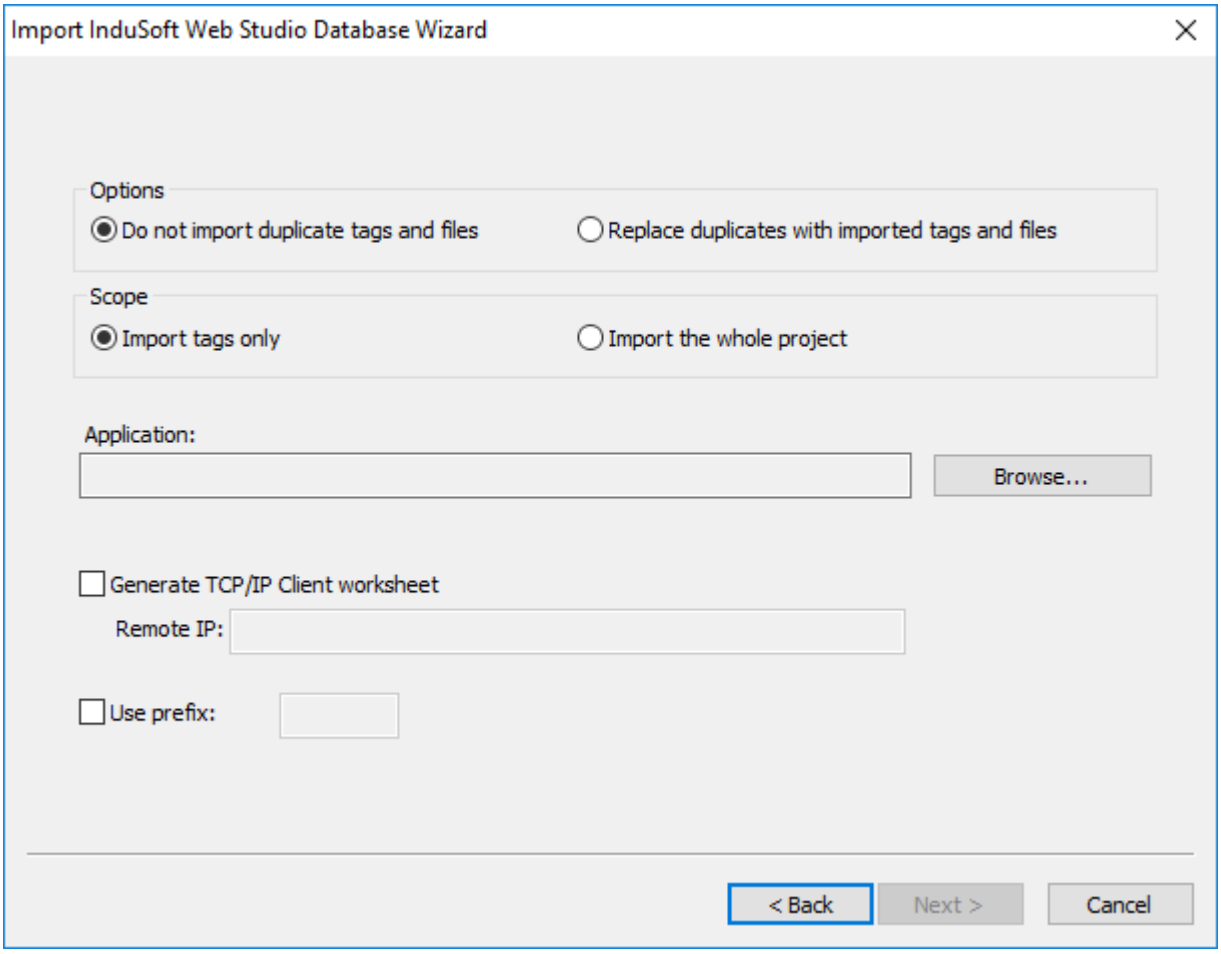
The first page of the import wizard is displayed.




***Import Wizard***

2. In the **Source Type** list, select **EcoStruxure Machine SCADA Expert 2020 Database**, and then click **Next**.

The next page of the import wizard is displayed.




*Import EcoStruxure Machine SCADA Expert 2020 Database Wizard*

3. In the **Options** area, select whether to import items that appear to be duplicates of existing items in your current project.

| Option | Description |
|---|---|
| **Do not import duplicate tags and files** | If an item has the same name as an existing item, do not import it.<br>The following items will not be imported at all:<br>• Main Procedures<br>• Startup Script |
| **Replace duplicates with imported tags and files** | Import all items. If an imported item has the same name as an existing item, replace the existing item. |

Some types of worksheets have incremented file numbers (e.g., SCRIPT0001, SCRIPT0002) instead of user-defined file names. These worksheets will always be imported and they will never replace existing worksheets, regardless of which option you select in this step. If there are existing worksheets with the same file numbers, the file numbers of the imported worksheets will be automatically increased in order to avoid replacing the existing worksheets. (Of course, this is relevant only if you also select the **Import the whole project** option in the next step.)

4. In the **Scope** area, select whether to import tags only or the whole project.

| Option | Description |
|---|---|
| **Import tags only** | Import project tags only. |
| **Import the whole project** | Import all items in the following folders:<br>• Project Tags<br>• Procedures<br>• Screens<br>• Screen Groups<br>• Web Pages<br>• Alarms<br>• Trends<br>• Recipes<br>• Reports<br>• Math<br>• Scripts<br>• Scheduler<br>• Drivers<br>• OPC (all types)<br>• TCP/IP |

5. In the **Application** area, do the following:
   a) Click **Browse**.
   A standard *Open* dialog box is displayed.
   b) Use the *Open* dialog box to locate and select the EcoStruxure Machine SCADA Expert 2020 project file (`<project name>.app`) that you want to import into your current project.
   c) Click **Open**.

   The *Open* dialog box is closed, and the file path and name for the project file you selected are displayed in the **Application** box.
6. If you also want to create a client-server connection between your current project and the imported project running on a remote station, do the following:
   a) Select the **Generate TCP/IP Client worksheet** option.
   b) In the **Remote IP** box, type the IP address of the computer or device that hosts — or will host — the imported project.

   The import wizard will use this information to create a new TCP/IP Client worksheet in the current project and then populate it with the imported tags. Keep in mind that once the worksheet has been created, you can change any of its settings including the IP address you entered here.
7. If you want to add a prefix to the names of the imported tags, in order to differentiate them from other tags in your current project, select the **Use prefix** option and then type the prefix (up to 4 characters) in the box.
8. Click **Next**.
   The last page of the import wizard is displayed.
9. Use the last page of the import wizard to select the tags that you actually want to import into your current project.
   The list shows all of the tags that the import wizard found in the imported database, and the check box to the left of each tag shows whether that tag is selected for import.
   - You can select or clear the check box for each tag in the list.
   - To select all of the check boxes, click **Check**. To clear all of the check boxes, click **Uncheck**.
   - To sort the list of tags, click the header of the column by which you want to sort.

- To filter the list of tags, click **Filter** and then configure filter strings for one or more columns. You can use wildcard characters (* and ?) in the filter strings.

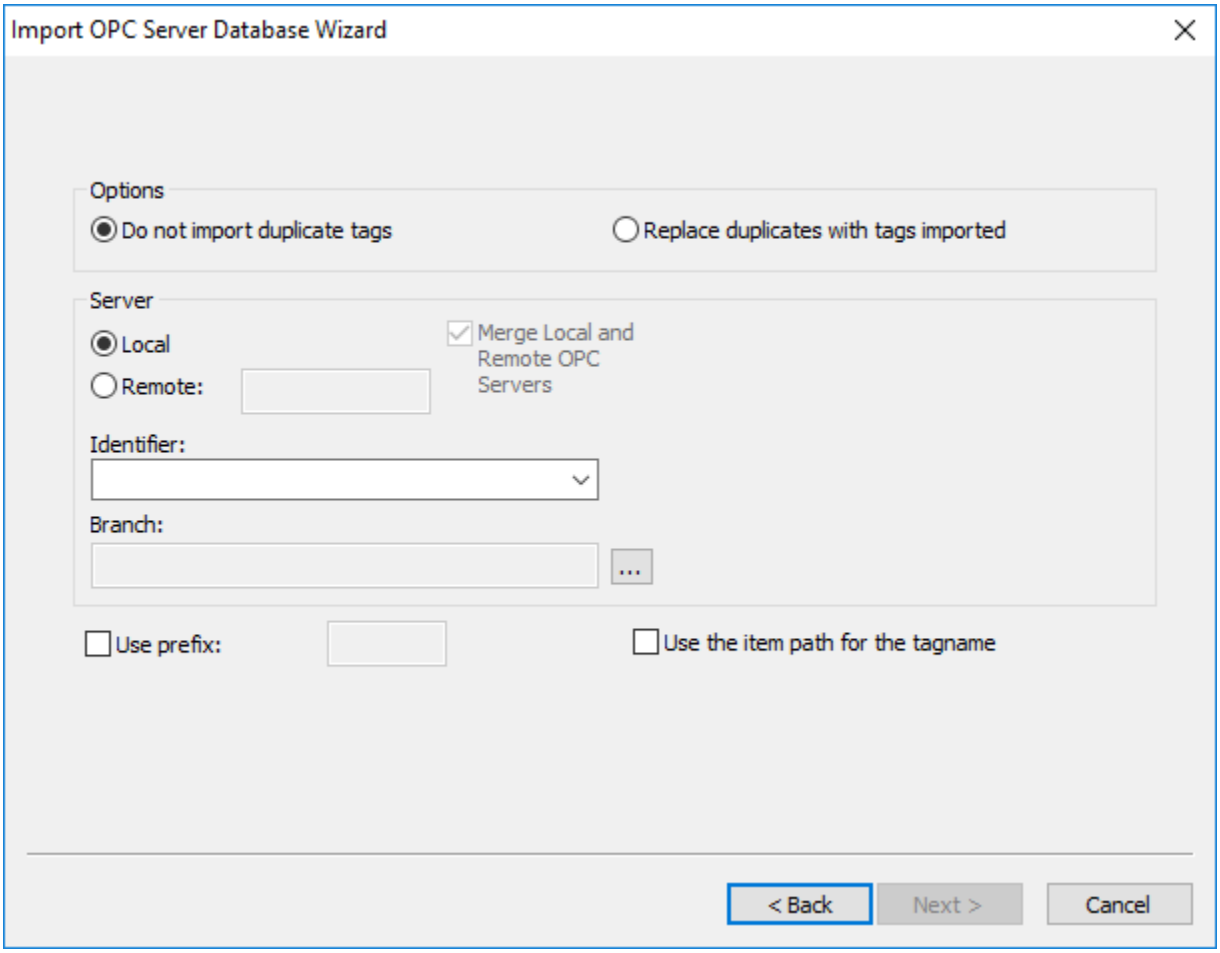
  Please note that filtering the list of tags does not select any tags for import. It only makes the list shorter and/or easier to navigate.

This is an important step because imported tags count against your project's tag limit. (For more information, see About target platforms, product types, and target systems on page 100.) The current number of tags in your project and the number of tags selected for import are shown in the **Database size** area of the page.

10. Click **Finish** to finish importing the database.

### *Importing from OPC Server Databases*

This wizard lets you import tags from an OPC Server running either locally or remotely. When you import tags from the OPC Server, an OPC Client worksheet is automatically created to link the tags, eliminating the need to manually configure the communication interface between your project and the external OPC Server.




***Import OPC Server Database Wizard***

- **Local/Remote**: Provide the following options:
  - **Local**: Select this option to import tags from an OPC Server installed in the local computer.
  - **Remote**: Select this option to import tags from an OPC Server installed in a remote computer. In the **Remote** field, type the IP Address (or the host name) of the remote computer that hosts the project runtime.

- **Merge Local and Remote OPC Servers** checkbox: If you selected a Remote server, check this option to display the list of OPC Servers installed in the local computer and also in the remote computer. Uncheck this checkbox to display only the list of OPC Servers installed in the remote computer.
- **Identifier** combo-box: Displays the list of available OPC Servers.
- **Branch**: Click on the Browse button (...) to select the branch of the OPC Server from which the tags (items) will be imported. Leave this field blank if you want to import tags from all branches configured in the OPC Server.
- **Use the item path for the tagname** checkbox: Check this option to concatenate the path name to the item name when importing tags from the OPC Server. Uncheck this option to use only the item names configured in the OPC Server.

In the grid displayed in Step 3 (**Import Wizard** on the Home tab of the ribbon) for this Data Source Type, there is an additional field with the label **OPC**, which displays the name of the items from the OPC Server.

> **Note:** See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.

### *Import tags from a CSV database*

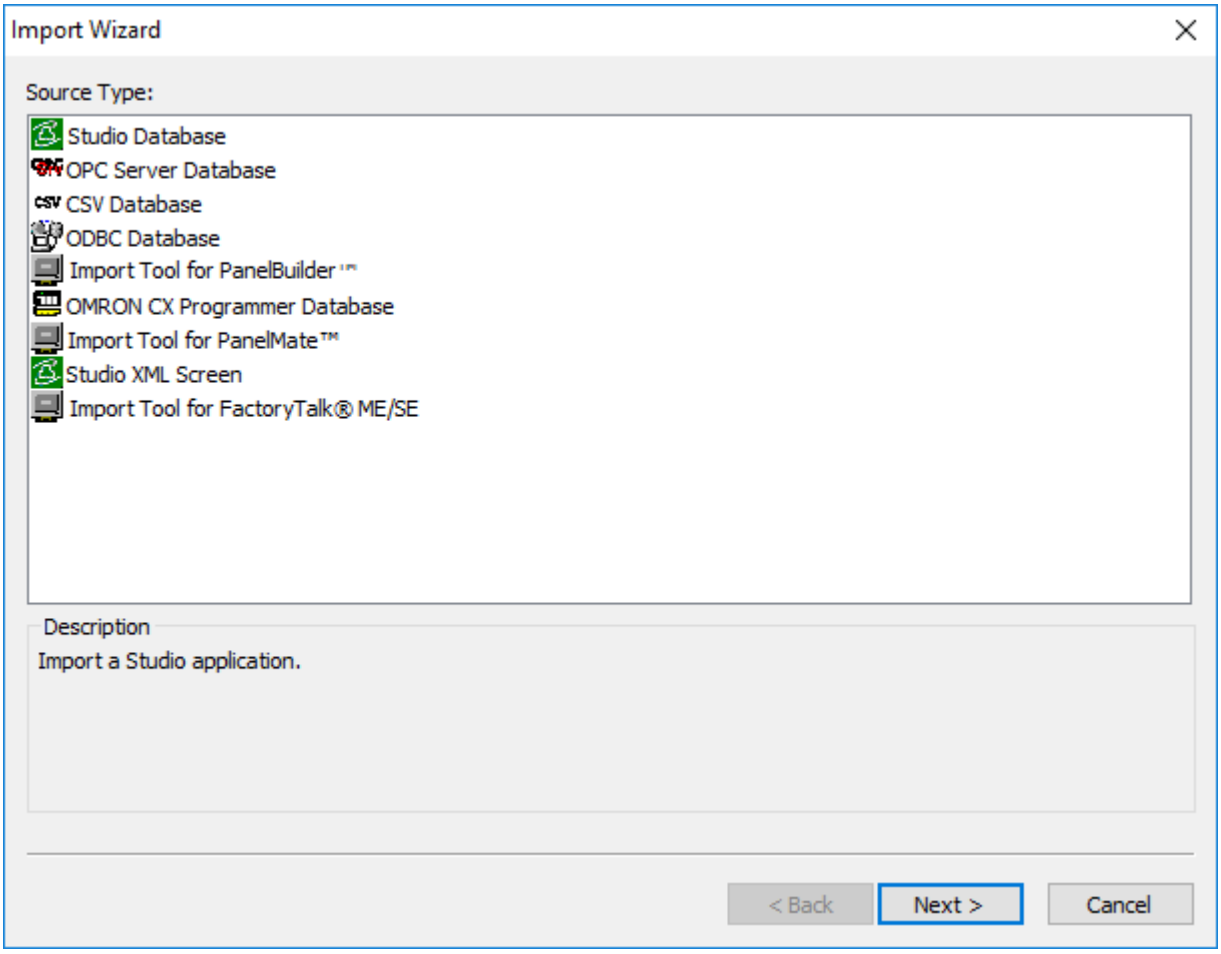
This tool allows you to import tags from a CSV database — that is, a flat database file that contains comma-separated values — or any similarly formatted text file.

> **Note:** If you are running Studio on a Windows computer that has User Account Control (UAC) enabled, you might have issues while performing this task. If you do, try running Studio as an administrator (i.e., right-click the Studio program icon, and then click **Run as administrator** on the shortcut menu).

To import from a CSV database file into your current project:

1. On the **Home** tab of the ribbon, in the **Tools** group, click **Import Wizard**.

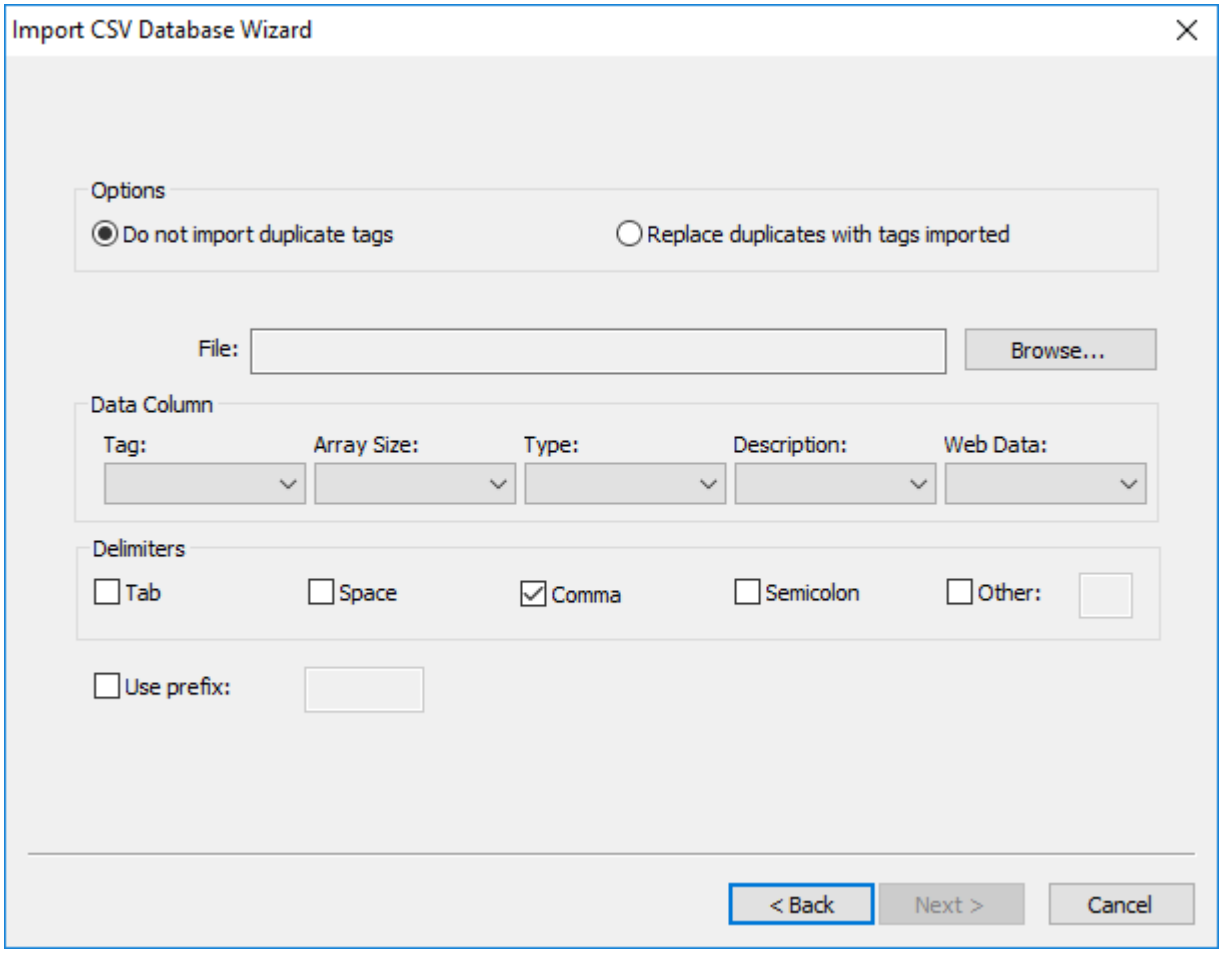
The first page of the import wizard is displayed.




***Import Wizard***

2. In the **Source Type** list, select **CSV Database**, and then click **Next**.

The next page of the import wizard is displayed.




***Import CSV Database Wizard***

3. In the **Options** area, select whether to import tags that appear to be duplicates of existing tags in your current project.

| Option | Description |
|---|---|
| **Do not import duplicate tags** | If a tag has the same name as an existing tag, do not import it. |
| **Replace duplicates with imported tags** | Import all tags. If an imported tag has the same name as an existing tag, replace the existing tag. |

4. Select the database file to be imported:
   a) To the right of the **File** box, click **Browse**.
   A standard *Open* dialog box is displayed.
   b) Use the *Open* dialog box to locate and select the file you want to import into your current project.

   Please note that the *Open* dialog box can only recognize files with the .csv extension, even though the import wizard can import files that do not strictly contain comma-separated values. As such, you might need to rename the file you want to import in order to have the dialog box recognize it. You will be able to specify a different separator in a subsequent step.
   c) Click **Open**.

The *Open* dialog box is closed, and the file path and name for the file you selected are displayed in the **File** box.

5. In the **Data Column** area, for each tag property, select the number of the column/field in the database file that contains the corresponding data.

   **Tag** (i.e., the tag name) is mandatory, but the other properties are optional — if a property is not included in the database file, select **Not used**.

   For example, if the tag name, array size and data type are listed in the second, third, and first columns of the database file, respectively, do the following: under **Tag**, select **2**; under **Array Size**, select **3**; under **Type**, select **1**; and under both **Description** and **Web Data**, select **Not used**.

   For tag properties that are not used, the import wizard will insert default values according to the following table:

| Tag Property | Default Value |
|---|---|
| Array Size | 0 |
| Type | Integer |
| Description | <blank> |
| Web Data (a.k.a. Scope) | Local |

6. In the **Delimiters** area, select the delimiters or separators for the database file to be imported. You can select more than one.

   **Comma** is the default for .csv files.

7. If you want to add a prefix to the names of the imported tags, in order to differentiate them from other tags in your current project, select the **Use prefix** option and then type the prefix (up to 4 characters) in the box.

8. Click **Next**.
   The last page of the import wizard is displayed.

9. Use the last page of the import wizard to select the tags that you actually want to import into your current project.

   The list shows all of the tags that the import wizard found in the imported database, and the check box to the left of each tag shows whether that tag is selected for import.

   - You can select or clear the check box for each tag in the list.
   - To select all of the check boxes, click **Check**. To clear all of the check boxes, click **Uncheck**.
   - To sort the list of tags, click the header of the column by which you want to sort.
   - To filter the list of tags, click **Filter** and then configure filter strings for one or more columns. You can use wildcard characters (* and ?) in the filter strings.

     Please note that filtering the list of tags does not select any tags for import. It only makes the list shorter and/or easier to navigate.

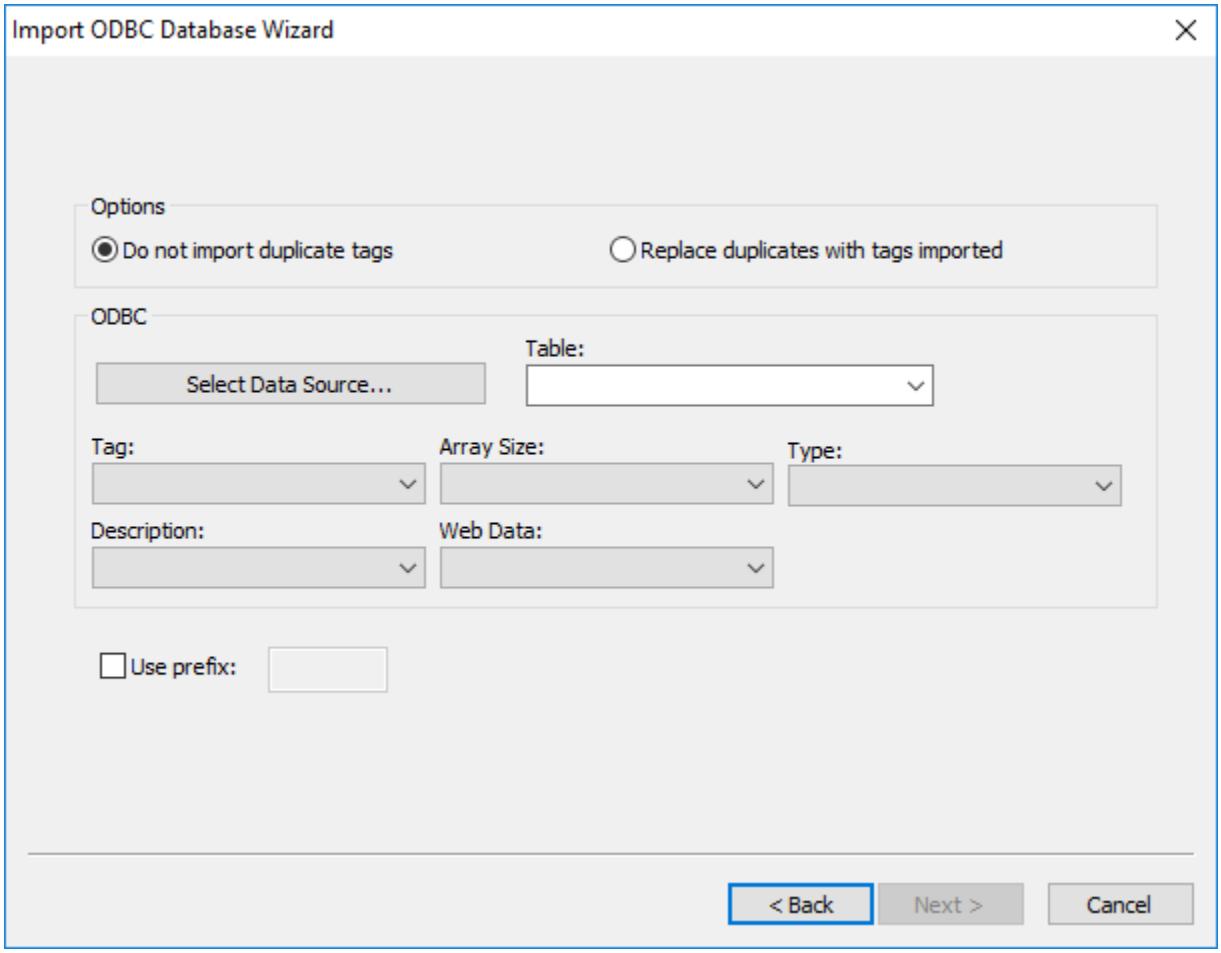
   This is an important step because imported tags count against your project's tag limit. (For more information, see About target platforms, product types, and target systems on page 100.) The current number of tags in your project and the number of tags selected for import are shown in the **Database size** area of the page.

10. Click **Finish** to finish importing the database.

> **Note:** This import wizard cannot import Class tags. If you need to import Class tags, open the database file in a spreadsheet application like Microsoft Excel and then copy-and-paste from the spreadsheet to the Project Tags datasheet. For more information, see Project Tags Folder on page 145.

### *Importing from ODBC Databases*

This wizard allows you to import tags from an external SQL Relational Database such as Microsoft Access, SQL Server, Oracle, My SQL, Sybase and others, through the ODBC interface.




***Import ODBC Database Wizard***

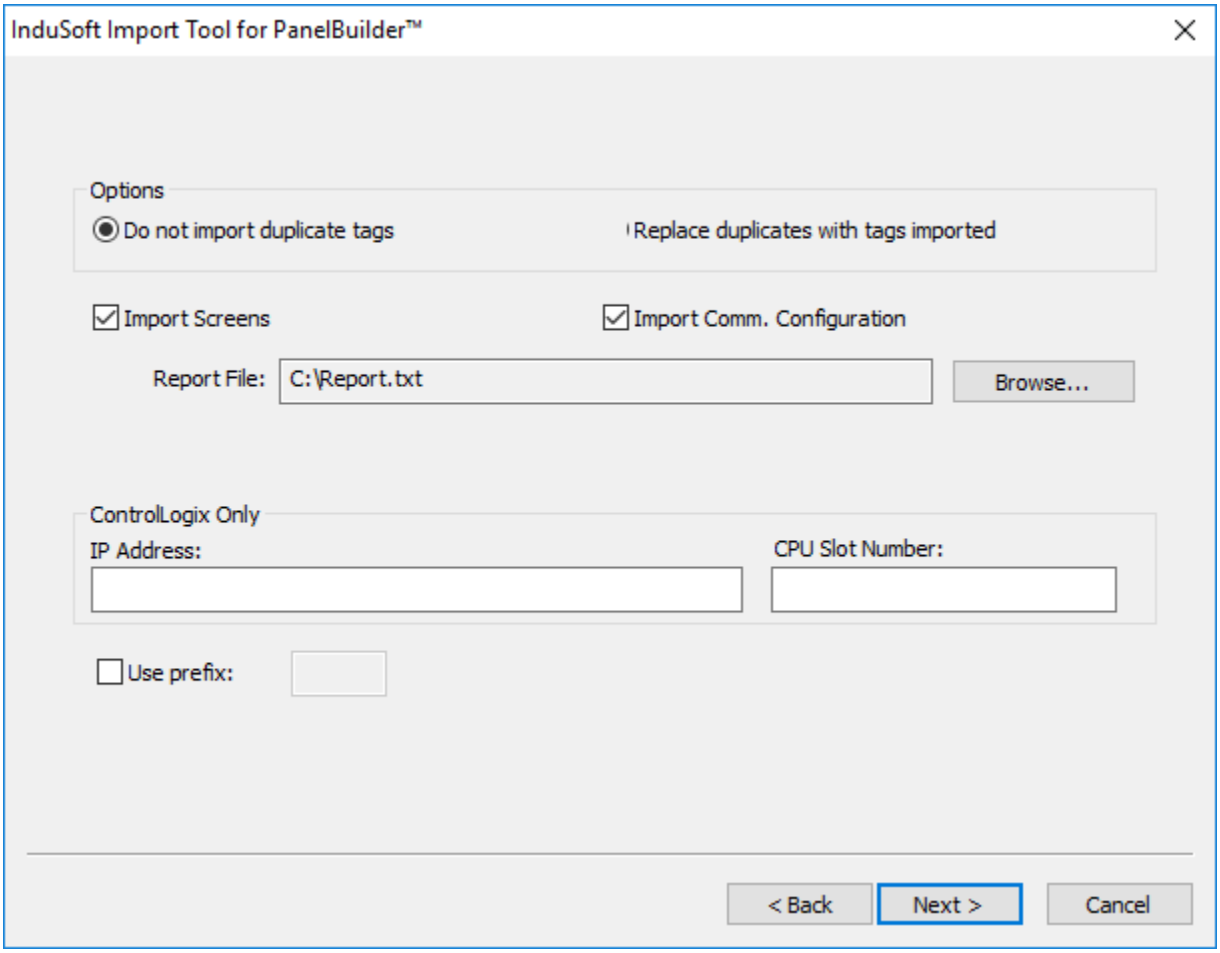
- **Select Data Source** button: Click to select the ODBC Data Source Name (DSN) linked to the database from which the tags will be imported. The DSN must have previously been created with the Data Sources (ODBC) window (**Control Panel** > **Administrative Tools** > **Data Sources [ODBC]**). After you select a DSN, the other fields in this window will be populated automatically with information from the selected database.
- **Table** combo-box: Select the table that holds the tags in the import database.
- **Tag** combo-box: Select the name of the column that holds the tags in the import database.
- **Array Size** combo-box: Select the name of the column that holds the array size for the tags in the import database.
- **Type** combo-box: Select the name of the column that holds the tag type in the import database.
- **Description** combo-box: Select the name of the column that holds the tag description in the import database.
- **Web Data** combo-box: Select the name of the column that holds the Web Data for the tags in the import database.

> **Note:** See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.

### *Importing from PanelBuilder32 Databases*

> **Note:** This wizard is sold as an add-on and requires a license to be enabled. Consult your software for further information.

This wizard lets you import not only the tags, but also the screens, alarm configuration and communication interface from a text file (report) exported by the PanelBuilder32™ software. Using this wizard, you can convert a PanelView™ program (developed with PanelBuilder32™) into the EcoStruxure Machine SCADA Expert 2020 project format and then run it on any platform supported by this software.




- **Import Screens**: Check this option to import the graphical screens (including their objects and animations).
- **Import Comm. Configuration**: Check this option to import the communication interface (tags linked to PLC addresses).
- **Report File**: Press the Browse button to select the name of the text file exported from PanelBuilder32™ (report printed to a text file).
- **ControlLogix Only**: When importing a program that was configured to exchange data with ControlLogix PLCs, this wizard can convert the communication interface to Ethernet/IP (ABCIP driver). To do so, type the IP Address of the PLC and its slot number. This information will be used to create the communication interface for the imported program. If the original program was already configured to use the Ethernet/IP interface, these fields can be left blank, because the IP Address and CPU Slot Number are retrieved from the program file itself.

In the grid displayed in Step 3 for this Data Source Type, there is an additional field with the label **Address**, which displays the tag addresses from the PanelBuilder project.

> **Tip:** Please consult the documentation for this import wizard for detailed information about how to export an program from the *.PBA format to the text (*.TXT) format, using PanelBuilder32™, and import it into this software.

> **Note:** See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.

> **Note:** This software does not support some special characters (e.g., `[ ] . –`) in tag names. When you import your PaneBuilder database, these special characters will be converted into underscores (_).

### *Importing PanelMate programs*

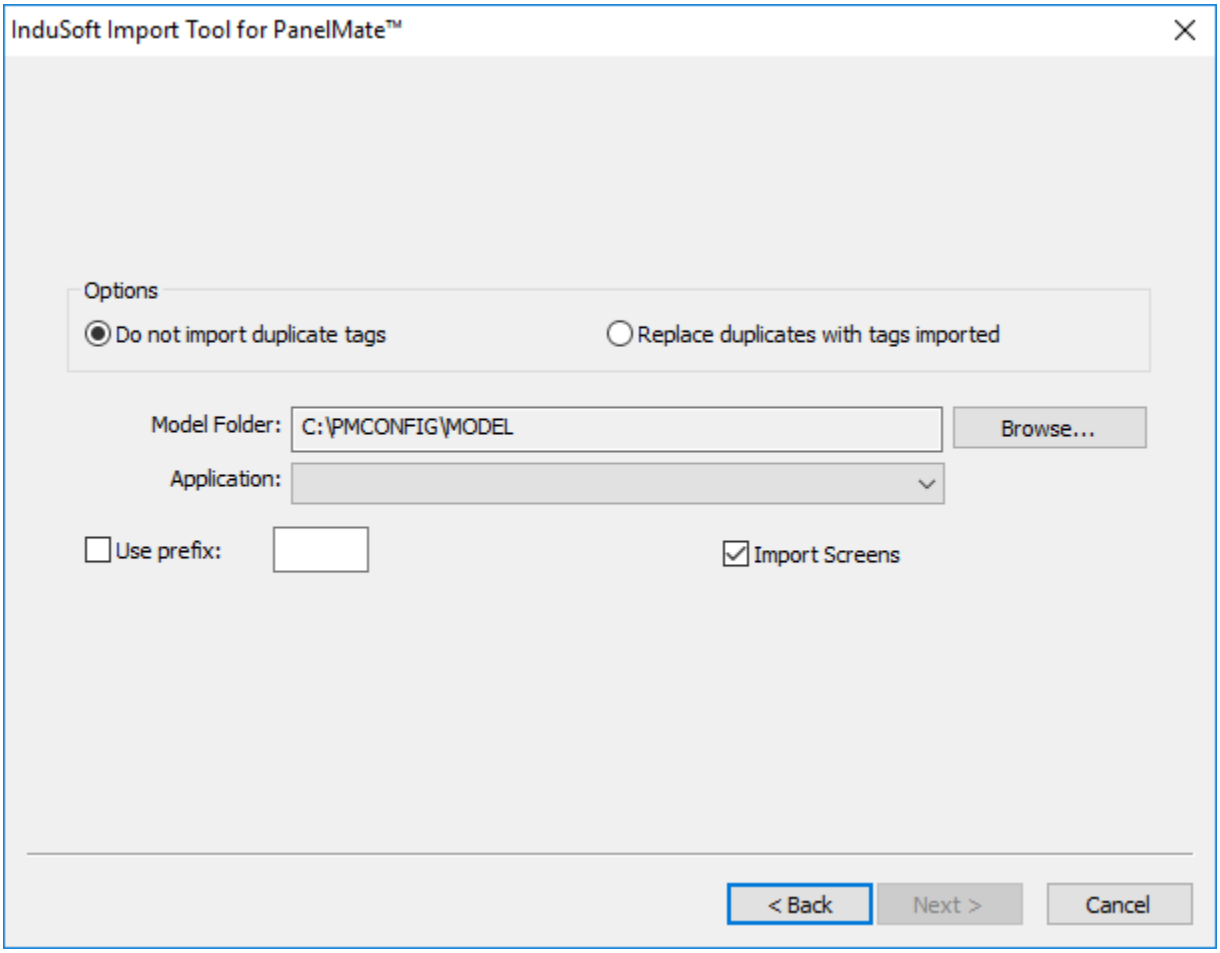
This wizard allows you to import not only the tags, but also the screens, alarm configuration, and communication settings from an operator interface program that was created with PanelMate™ software.

Using this wizard, you can convert a PanelMate program into the EcoStruxure Machine SCADA Expert 2020 project format and then run it on any platform supported by this software. The wizard can import programs from PanelMate Plus 4.01 (or earlier) and PanelMate Power Pro 2.21 (or earlier), and it supports the following communication drivers:

- Allen-Bradley Serial (ABKE)
- Modbus Serial, ASCII and RTU (MODBU)
- Modbus Plus (MODPL)
- Allen-Bradley Remote I/O (STRIO)

> **Note:** This import wizard is sold as an add-on for EcoStruxure Machine SCADA Expert 2020, and it requires a license to be enabled. For more information, consult your software vendor.
>
> Also, if you are running EcoStruxure Machine SCADA Expert 2020 on a Windows operating system that has User Account Control (UAC) enabled, then you may have problems using this import wizard. Close the application, and then run it again as an administrator (i.e., right-click the

EcoStruxure Machine SCADA Expert 2020 program icon, and then click **Run as administrator** on the shortcut menu).




***Import Tool for PanelMate***

**Import Screens**

Check this option to import the graphical screens (including their objects and animations) to EcoStruxure Machine SCADA Expert 2020.

**PanelMate Model**

Click **Browse** to select the directory where the database files of the PanelMate program that you intend to import are stored.

**Application**

After specifying the correct file path in the **PanelMate Model** box, the programs that are available in that directory will be available in this combo-box. Select the program that you want to import, and then click **Next**.

See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common to all Source Types.

### *Import from a FactoryTalk application*

Use the Import Tool for FactoryTalk to import tags, screens, alarms, and device configurations from an application that was previously created with FactoryTalk View.

This tool is sold as an add-on for Studio, which means your software license must include the appropriate upgrade in order for the tool to be enabled in the project development environment. For more information, contact your EcoStruxure Machine SCADA Expert 2020 software distributor.

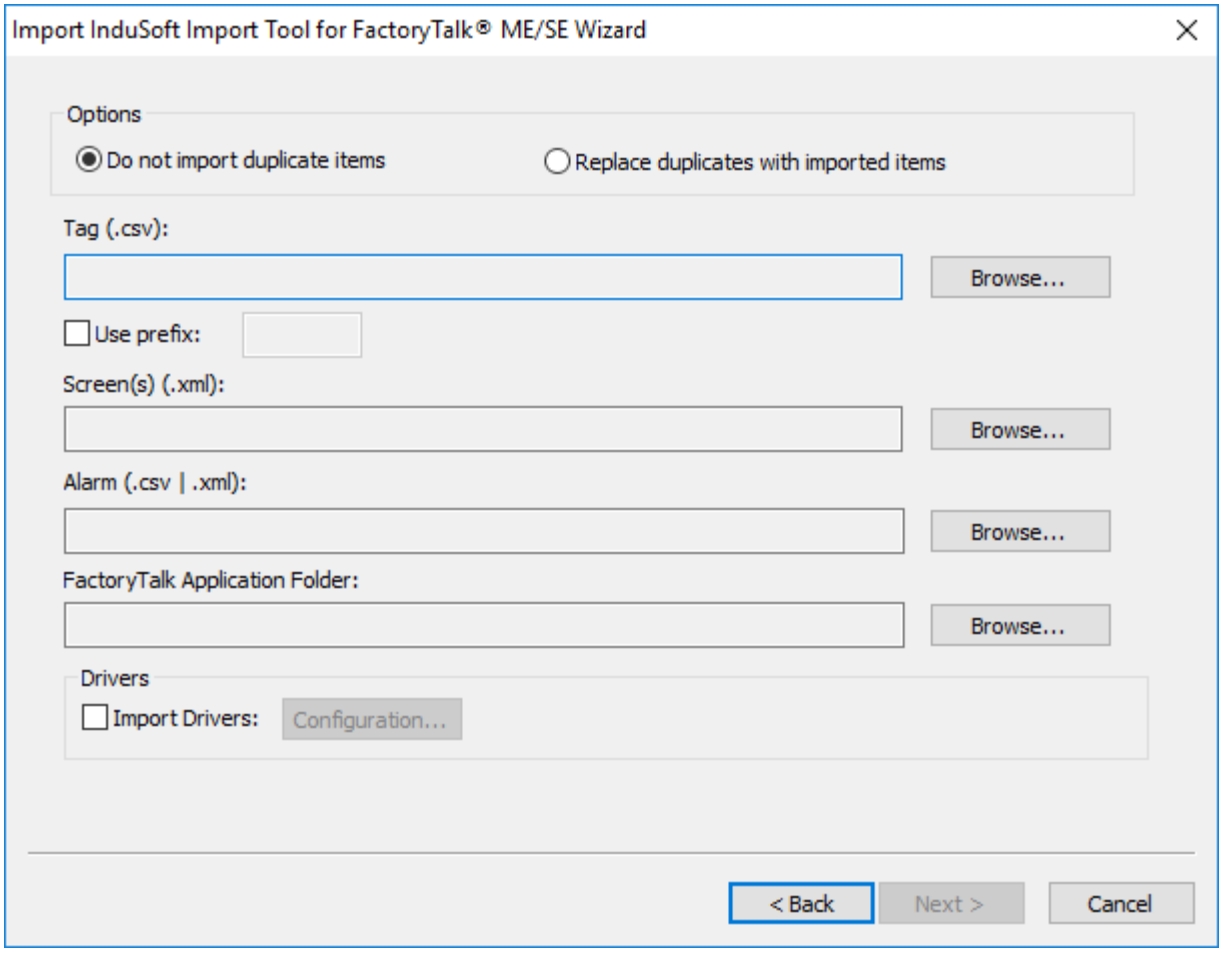
If you are running Studio on a Windows computer that has User Account Control (UAC) enabled, you might experience issues while using this tool. To avoid these issues, run Studio as an administrator (i.e., right-click the Studio program icon, and then click **Run as administrator** on the shortcut menu).

Depending on which items (e.g., tags, screens, alarms) you actually want to import from the FactoryTalk application, you might need to use FactoryTalk View to export those items from the application to external files. You should have those files ready before you begin this task. For more information, consult the documentation for FactoryTalk View.

The Import Tool for FactoryTalk can import both Site Edition (SE) and Machine Edition (ME) applications, and it supports the following communication drivers: ABCIP, ABENI, ABKE, and ABTCP.

To import from a FactoryTalk application:

1. On the **Home** tab of the ribbon, in the **Tools** group, click **Import Wizard**.
   The *Import Wizard* dialog box is displayed.
2. In the **Source Type** list, select **EcoStruxure Machine SCADA Expert 2020 Import Tool for FactoryTalk**, and then click **Next** to proceed to the next page of the import wizard.
   The *Import Tool for FactoryTalk* page is displayed.




***Import Tool for FactoryTalk dialog box***

3. In the **Options** area, select whether to import items that appear to be duplicates of existing items in your EcoStruxure Machine SCADA Expert project:

| Option | Description |
|---|---|
| **Do not import duplicate items** | When this option is selected, the following items will not be imported in case there are already equivalents in the current project:<br>• Tags Database (i.e., tags with the same name will not be imported)<br>• Screens (i.e., screens with the same name will not be imported) |
| **Replace duplicates with imported items** | When this option is selected, existing items in the current project will be replaced by items of the same name that are imported from the FactoryTalk application. |

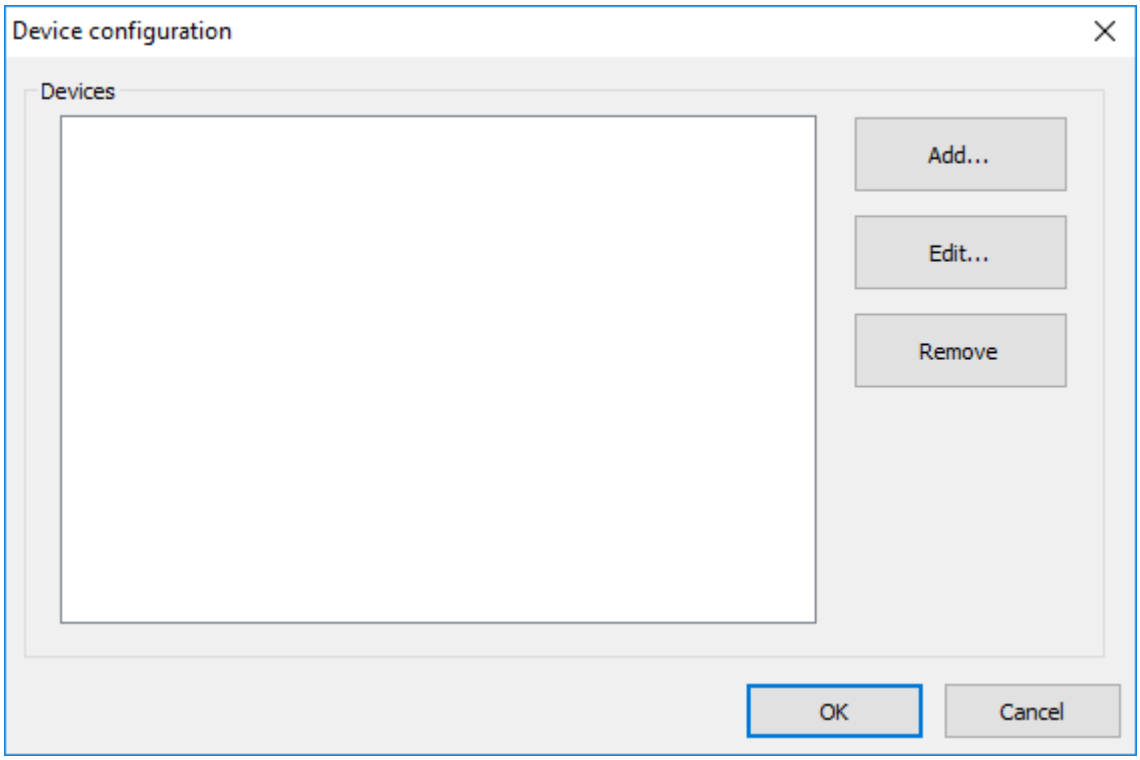
4. If you want to import tags from the FactoryTalk application:
   a) To the right of the **Tag** box, click **Browse**.
   A standard Windows file browser is displayed.
   b) Use the file browser to locate and select the .csv file that contains the tag information you exported from the FactoryTalk application, and then click **Open**.
   The location of the file is displayed in the **Tag** box.
   c) If you want to add a prefix to the names of the imported tags, in order to differentiate them from other tags in your EcoStruxure Machine SCADA Expert project, select **Use prefix** and then type the prefix in the box.
5. If you want to import screens — including objects and animations — from the FactoryTalk application:
   a) To the right of the **Screen(s)** box, click **Browse**.
   A standard Windows file browser is displayed.
   b) Use the file browser to locate and select the .xml file that contains the screen information you exported from the FactoryTalk application, and then click **Open**.
   The location of the file is displayed in the **Screen(s)** box.

   For more information about which objects and animations are supported, see *Import Tool for FactoryTalk User Manual*.
6. If you want to import alarms from the FactoryTalk application:
   a) To the right of the **Alarm** box, click **Browse**.
   A standard Windows file browser is displayed.
   b) Use the file browser to locate and select the .csv file that contains the alarm information you exported from the FactoryTalk application, and then click **Open**.

   > **Note:** At this time, the Import Tool for FactoryTalk can import alarm configurations only from FactoryTalk Site Edition (SE) applications.

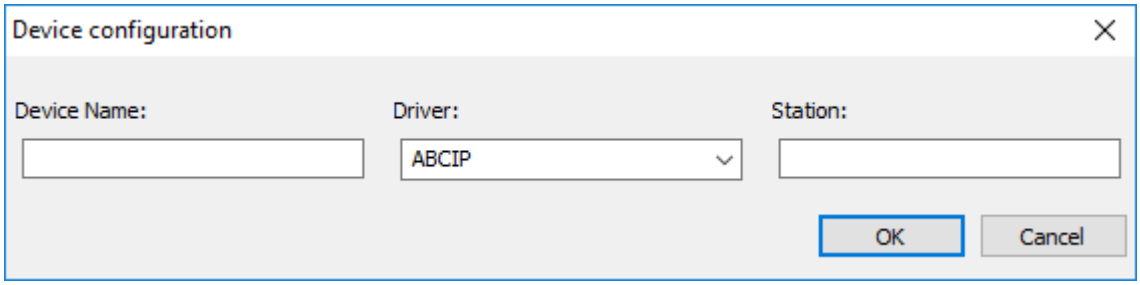
   The location of the file is displayed in the **Alarm** box.
7. To import all other information including images and parameters from the FactoryTalk application:
   a) To the right of the **FactoryTalk Application Folder** box, click **Browse**.
   A standard Windows file browser is displayed.
   b) Use the file browser to locate and select the FactoryTalk application folder, and then click **OK**.
   The location of the folder is displayed in the **FactoryTalk Application Folder** box.
8. If you want to import one or more device configurations from the FactoryTalk application:
   a) In the **Drivers** area, select **Import Drivers**, and then click **Configuration**.

The *Device Configuration* dialog box is displayed.




***Device Configuration dialog box***

b) Click **Add**.
The *Device* dialog box is displayed.




***Device dialog box***

c) In the **Device Name** box, type the name of the device exactly as it is in the FactoryTalk application.

d) In the **Driver** list, select a communication driver for the specified device.

| Option | Description |
|---|---|
| **ABCIP** | Driver for Ethernet communication with Allen-Bradley devices using the CIP protocol |
| **ABENI** | Driver for Ethernet communication Allen-Bradley devices using the AB-1761-NET-ENI Gateway Interface |
| **ABKE** | Driver for serial communication with Allen-Bradley devices using the DF1 protocol |
| **ABTCP** | Driver for Ethernet communication with Allen-Bradley devices using the DF1 protocol |

e) In the **Station** box, type the station ID of the specified device.

f) Click **OK**.
The specified device is added to the **Devices** list in the *Device Configuration* dialog box.

g) Repeat these steps for each device configuration that you want to import.

h) When you are done, click **OK** to close the *Device Configuration* dialog box.

9. Click **Next** to proceed to next page of the import wizard, which is common to all source types.
For more information, see Import Wizard on page 203.

### *Import a Studio XML Screen*

Use the Import Wizard to import a Studio XML Screen, which is an external text file created with EcoStruxure Machine SCADA Expert 2020's custom XML schema.

Before you begin this task, you must have a properly formatted Studio XML Screen file that you can import.

A Studio XML Screen file contains the same information as a regular screen file. It is simply formatted as human-readable XML instead of binary data, which makes it more flexible and portable.

XML files can be manually created in any text editor, of course, but our intent is to enable you to quickly and programmatically create large numbers of screens in Microsoft Visual Studio using our custom XML schema and its associated C# API. This often requires additional training and support, so the schema and API are not included in the standard installation of EcoStruxure Machine SCADA Expert 2020. Instead, the schema, API, documentation, sample files, and support contract can be purchased as a separate toolkit. For more information, please contact your software vendor.

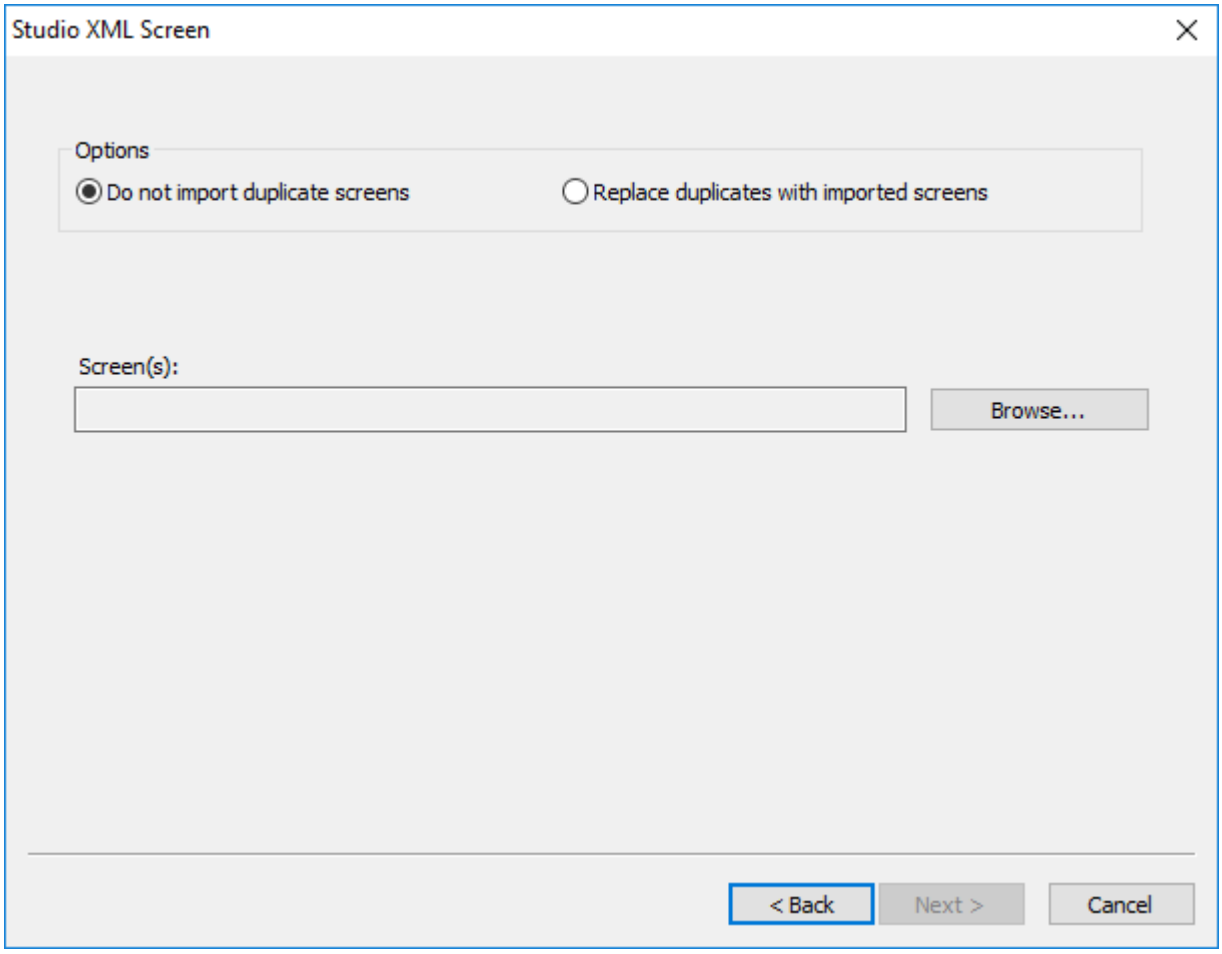
Once you have created your screens, you can use the Import Wizard to batch import them into your EcoStruxure Machine SCADA Expert project.

> **Tip:** You can also use the function `ImportXML` to import Studio XML Screen files during run time.

To import one or more Studio XML Screens:

1. On the **Home** tab of the ribbon, in the **Tools** group, click **Import Wizard**.
The *Import Wizard* dialog box is displayed.
2. In the **Source Type** list, click **Studio XML Screen**, and then click **Next**.

The next step of the import wizard is displayed.




***Selecting the screens to import***

3. Under **Options**, choose whether imported screens should automatically replace existing screens in your project.

   Screens are considered to be duplicates if they have the same file name. For example, `Objects.xml` and `Objects.scc` would be duplicates.

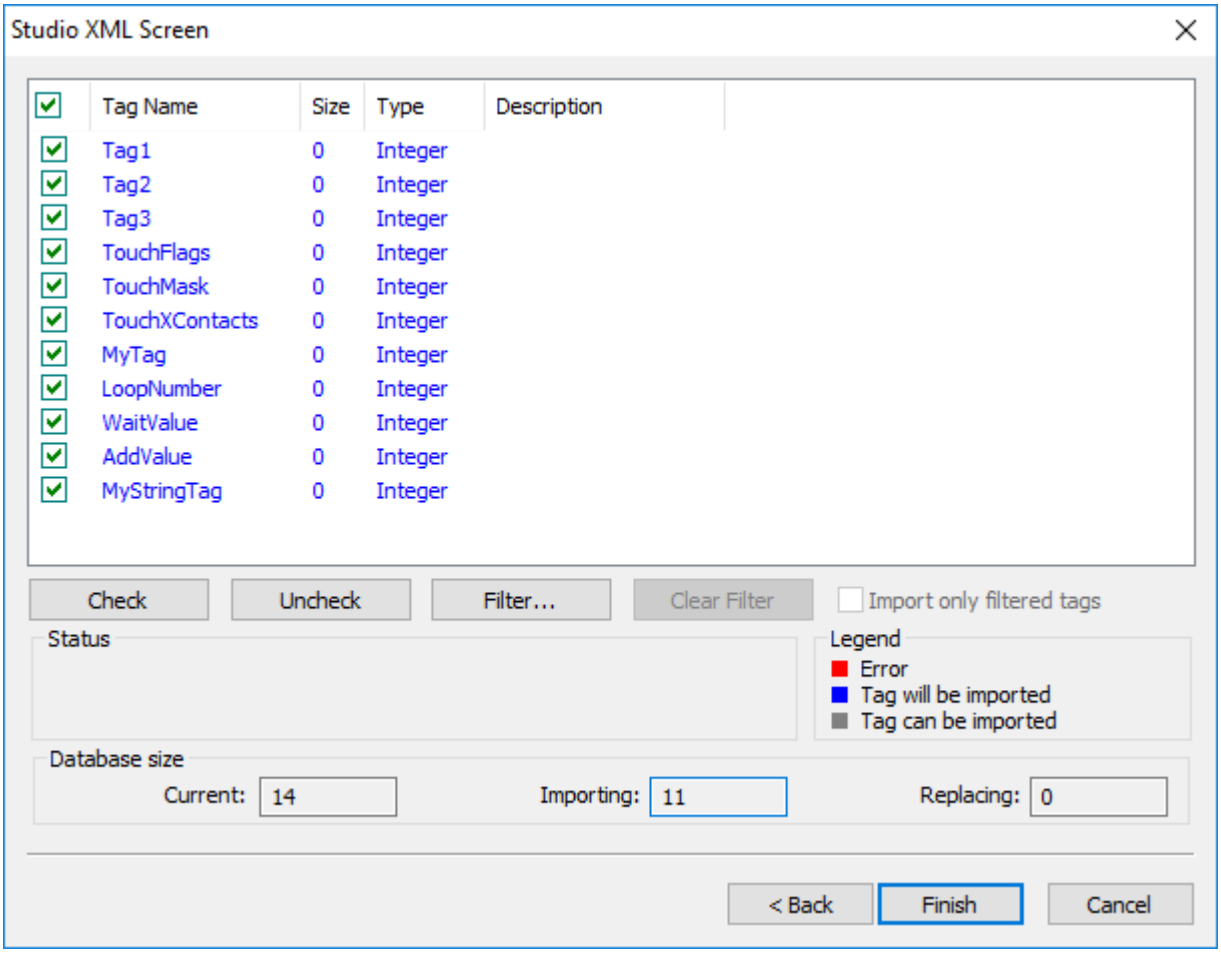
   - If you do not want the imported screens to replace existing screens in your project, select **Do not import duplicate screens**. A warning will be displayed for each duplicate that you try to import.
   - If you want the imported screens to automatically replace existing screens in your project, select **Replace duplicates with imported screens**.

4. Click **Browse**.
   A standard *Open* dialog box is displayed.
5. Use the file browser to locate and select the Studio XML Screen files that you want to import, and then click **Open**.

   You can Ctrl-click and Shift-click to select more than one file.

   The selected file(s) are displayed in the **Screen(s)** box.
6. Click **Next**.

The selected screens are processed, and then the next step of the import wizard is displayed showing the project tags that are included in the selected screens.




***Selecting the project tags to import***

7. In the list of project tags, select the tags that you want to import with the screens:
   - For each tag in the list, select or clear the check box to the left.
   - To select all of the check boxes, click **Check**. To clear all of the check boxes, click **Uncheck**.
   - To filter the list of tags, click **Filter** and then configure filter strings for one or more columns. You can use wildcard characters (`*` and `?`) in the filter strings.
8. Click **Finish**.

The screens and included tags are imported into your project. Also, the screens are automatically published for Thin Clients and Mobile Access — i.e., the corresponding `*.HTML`, `*.SCC`, and `*.SSMA` files are automatically generated and saved in your project's Web folder, so the imported screens should be immediately available for you to select.

## Tag Integration

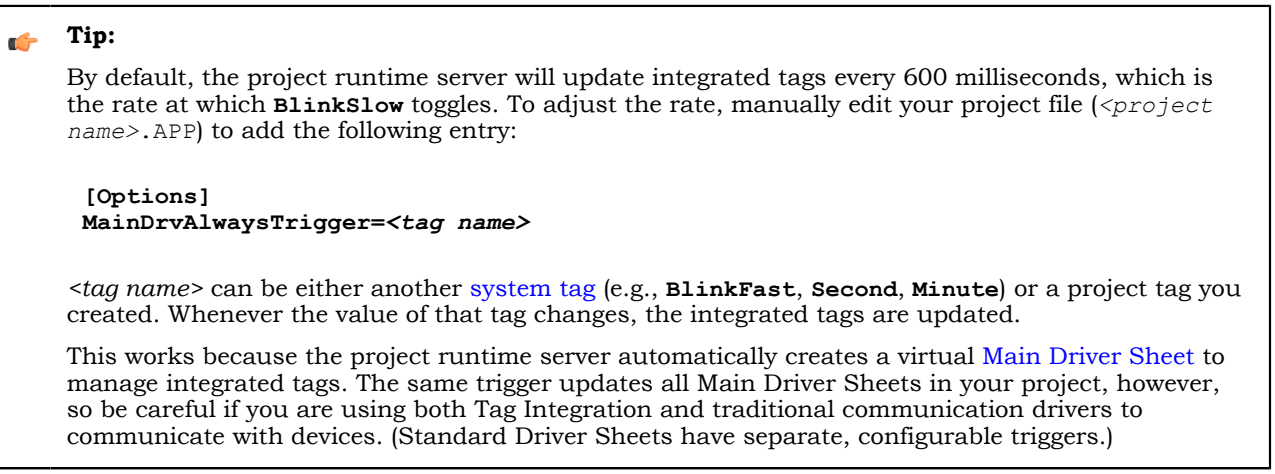
Tag Integration is an enhanced framework for communication with third-party applications and devices.

Tag Integration is built on the same communication drivers that are described in the Drivers section, but instead of manually configuring driver worksheets to associate project tags with device registers, you can use the Object Finder to browse a Tag Integration source and then import device registers directly into your project.

Device registers imported in this way appear as integrated tags in your project's Shared Database folder, and they count against your project's tag limit as determined by its target system. Integrated tags are "live", which means they are continuously and bilaterally updated during project run time as long as the Tag Integration source is also running and connected. In most cases, you can use integrated tags in the same ways that you would normally use project tags you created.

Tag Integration is available only for certain applications and devices, because additional work is required to upgrade a traditional communication driver to support this feature. Many of the drivers included with this software can be upgraded, however, so if the one you want is not listed in the Tag Integration settings, please contact your software distributor and ask about custom development.

Tag Integration is configured in the Communication tab of your project settings.



**Tip:**

By default, the project runtime server will update integrated tags every 600 milliseconds, which is the rate at which **BlinkSlow** toggles. To adjust the rate, manually edit your project file (*<project name>*.APP) to add the following entry:

```
[Options]
MainDrvAlwaysTrigger=<tag name>
```

*<tag name>* can be either another system tag (e.g., **BlinkFast**, **Second**, **Minute**) or a project tag you created. Whenever the value of that tag changes, the integrated tags are updated.

This works because the project runtime server automatically creates a virtual Main Driver Sheet to manage integrated tags. The same trigger updates all Main Driver Sheets in your project, however, so be careful if you are using both Tag Integration and traditional communication drivers to communicate with devices. (Standard Driver Sheets have separate, configurable triggers.)

## Using TagsDB functions to edit the tags database during run time

Use the Tags Database (TagsDB) functions to add and remove tags, classes, and class members during project run time, as well as to set properties and alarm conditions on tags.

There are several important things to keep in mind when you use TagsDB functions, because the functions can do much more than set and get tag values. They actually change the structure of the tags database, which can cause serious problems for a running project and all connected thin clients if it is not done properly. As such, most TagsDB functions can be executed only under the following limitations.

First, you must use the SCADA runtime edition for Windows to run your project — in other words, you must install the full Studio software on a Windows computer and then license it for "Runtime only". (You can also license the software for "Engineering only", but your project will run for only 72 hours before it needs to be restarted.) The TagsDB functions use the project development environment's database editor in essentially the same way that you do when you manually edit your project during run time. Due to this limitation, the TagsDB functions cannot be used in projects that are developed for and run on embedded devices.

Second, the TagsDB functions can be called only by scripts that are executed on the project runtime server. The functions cannot be executed on project thin clients because a client cannot make structural changes to the tags database without interfering with other clients, decreasing run-time performance, and potentially corrupting the database. (In this case, "project thin clients" includes the Viewer module that runs on the same computer as the project runtime server, because it runs as a separate process on that computer.) Therefore, generally speaking...

TagsDB functions **can** be used in:

- The Startup Script, which is executed when the project itself is run;
- Script Groups, which are periodically scanned by the Background Task; and
- Global Procedures that are called by the Startup Script or Script Groups.

TagsDB functions **cannot** be used in:

- The Graphics Script, which is executed by each project thin client client when it starts;
- Screen Scripts, which are attached to project screens and executed when those screens are opened; and
- Command animations.

To work around this limitation, create a Script Group to call the TagsDB functions and then configure a trigger to control the execution of that Script Group.

Third, in any script that calls TagsDB functions to make structural changes to the tags database, you must call the `TagsDBBeginEdit` function at the beginning of the script and the `TagsDBEndEdit` function at the end of the script. The `TagsDBBeginEdit` function locks the database for editing and prevents any other run-time changes. The `TagsDBEndEdit` function applies the changes made by TagsDB functions and then allows the database to resume normal run-time behavior. Both functions must be called in the same script, because that script (more specifically, the program thread running that script) owns the tags database while it is locked. You cannot call `TagsDBBeginEdit` in one script and then call `TagsDBEndEdit` in another script.

Normally, when a project is edited during run time, the project runtime server and all project thin clients must be updated with the changes as they are made. This is not a problem when you manually edit your project, because you make your changes slowly and one at a time. In contrast, the TagsDB functions allow you to make a large number of changes quickly, so updating the server and clients with all of those changes while the project is running can severely decrease run-time performance. Therefore, to maintain performance and protect the tags database, the server — including all background tasks such as alarms, trends, and other scripts — is effectively paused when the `TagsDBBeginEdit` function is executed, and then the changes are applied as a batch when the `TagsDBEndEdit` function is executed. Also, as part of this update process, project screens that were already open on clients will be reopened and their OnOpen screen scripts will be executed again.

> **Note:** The `TagsDBBeginEdit` function has a persistent effect, which means that if you call the function to lock the tags database during project run time and then stop the project, the database will remain locked and you will not be able to manually edit it.
>
> Restarting the project may or may not unlock the database, depending on how you developed your project and which function call locked the database in the first place. As such, while the project is

stopped, you should use the *Watch* window to manually call the `TagsDBEndEdit` function. When it is successfully executed, you can safely restart the project.

### Examples

The following example shows how to use the TagsDB functions in VBScript to add a new class, then add a new class member to that class, then add a new tag of that class, then set an alarm and a trend on that tag.

```
If($TagsDBBeginEdit()=0) Then
  If($TagsDBAddClass("TempClass")=0) Then
    If($TagsDBAddClassMember("TempClass","TempMember","Real")=0) Then
      If($TagsDBAddTag("TempTag","TempClass",2,0)=0) Then
        If($TagsDBSetAlarm("TempTag[0].TempMember",1,0,3.5)<>0) Then
          $Msg = "Alarm not Set"
        End If
        If($TagsDBSetTrend("TempTag[0].TempMember",0,1)<>0) Then
          $Msg = "Trend not Set"
        End If
      Else
        $Msg = "Tag not created"
      End If
    Else
      $Msg = "Class Member not added"
    End If
  Else
    $Msg = "Class not created"
  End If
  $TagsDBEndEdit()
Else
  $Msg = "Tag functions not enabled"
End If
```

Please note how the script begins with the `TagsDBBeginEdit` function and then ends with the `TagsDBEndEdit` function. Also, see how the nested **If**…**Then**…**Else** structures ensure that each function is executed successfully (i.e., returns a value of 0) before the next one is attempted.

The following example shows how to remove the alarm, trend, tag, class member, and class, in reverse order from how they were added in the previous example.

```
If($TagsDBBeginEdit()=0) Then
  If($TagsDBRemoveAlarm("TempTag",1)<>0) Then
    $Msg = "Alarm not removed"
  End If
  If($TagsDBRemoveTrend("TempTag")<>0) Then
    $Msg = "Trend not removed"
  End If
  If($TagsDBRemoveTag("TempTag")=0) Then
    If($TagsDBRemoveClassMember("TempClass","TempMember")<>0) Then
      $Msg = "Class member not removed"
    End If
    If($TagsDBRemoveClass("TempClass")<>0) Then
      $Msg = "Class not removed"
    End If
  Else
    $Msg = "Tag not removed"
  End If
  $TagsDBEndEdit()
Else
  $Msg = "Tag functions not enabled"
End If
```

It is not absolutely necessary to remove the alarm and trend before removing the tag they are on, because they are discarded with the rest of the tag properties and other metadata when the tag itself is removed. They

are included in the example above simply to be thorough. In contrast, the class member and class cannot be removed until every tag in that class is removed.

# Screens and Graphics

The most basic function performed by EcoStruxure Machine SCADA Expert is to provide a window into the process. The ability to display the status of the process by interacting with instrumentation (or computers), is described as the Human-Machine Interface (HMI).

EcoStruxure Machine SCADA Expert allows you to create projects that can monitor processes using high-resolution color screens.

The EcoStruxure Machine SCADA Expert graphic tools consist of two modules:

- The *Screen/ Worksheet Editor* in the EcoStruxure Machine SCADA Expert development environment (used to create or import graphics); and
- The runtime project *Viewer*.

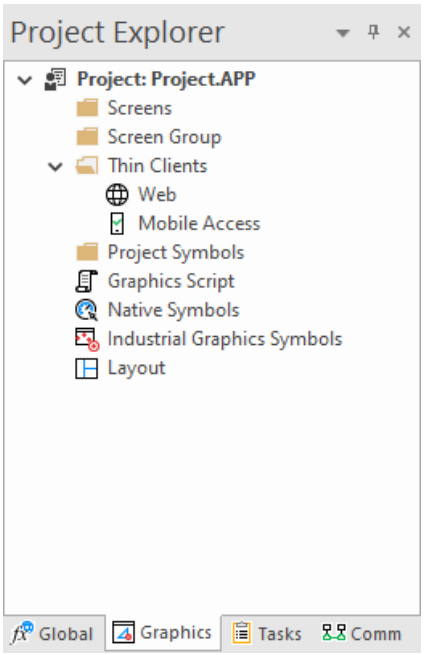
You can use *animations* to create dynamic graphic objects or symbols. Animations cause objects and symbols to change appearance to reflect changes in the value of a tag or an expression. Each screen is an association of static and animated objects.

Screens can have an optional bitmap that acts as a background in the object window. On the following screen for example, the static images can be part of a bitmap in the background object and objects with animation in the animation object layer can reflect the changes in the plant, giving the illusion that the screen is three#dimensional.

All EcoStruxure Machine SCADA Expert configuration tasks require a Windows-compatible pointing device, such as a mouse or touch pad. You can run a project in the Viewer without a pointing device if you configure keypad or keyboard keys for all commands.

## Graphics tab

The Graphics tab of the Project Explorer contains all of the screens, screen groups, and symbols in your project.




***Graphics tab of the Project Explorer***

The folders on the Graphics tab are described in the following sections:

**Screens**

You create screens to provide a graphical interface for your project. Each screen can contain many buttons, sliders, dials, indicators, graphs, and so on.

**Screen Groups**

You can combine individual screens into screen groups, so that they all open together at the same time.

**Thin Clients**

You can deploy your project as a web application to be accessed by thin clients such as desktop web browsers, tablets, and smartphones. You can even deploy different versions of your project with different levels of functionality for each type of client.

**Project Symbols**

This folder contains all of the custom symbols that you create for your project. A symbol is a group of interconnected screen objects that work together to perform a single function — for example, lines, rectangles, and text fragments that have been arranged to make a slider control.

**Graphics Script**

You can use this worksheet to define VBScript sub-routines that are called only when the graphics module starts (i.e., when a client station connects to the server and displays the graphical interface), while it is running, and when it ends.

**Native Symbols**

This folder is a library of the symbols that are created with the native graphics tools in Studio. It contains not only the custom symbols that you create (see Project Symbols above), but also a large selection of premade symbols that are installed with Studio.

**Industrial Graphics Symbols**

This folder is a library of the symbols that are created with the Industrial Graphics editor, which works as a companion to the native graphics tools in Studio.

**Layout**

The layout editor displays all of the screens the are currently open for editing. You can use it to visualize how the screens are arranged together and reuse screens in multiple layouts — for example, to create a common navigation bar across your entire project.

### *Screens folder*

The **Screens** folder is located in the **Graphics** tab of the *Project Explorer*. It contains all of your Screen worksheets, both completed and still in development.

To create a new Screen worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Graphics group, click **Screen**;
- Right-click the *Screens* folder in the Project Explorer, and the click **Insert** on the shortcut menu; or
- Go to **File**, select **New**, click the **File** tab in the *New* dialog, select **Screen** from the list of worksheet types, and then click **OK**.

When a Screen worksheet is opened for the first time, the *Screen Attributes* dialog for that worksheet is automatically displayed. For more information, see *Screen Attributes* dialog.

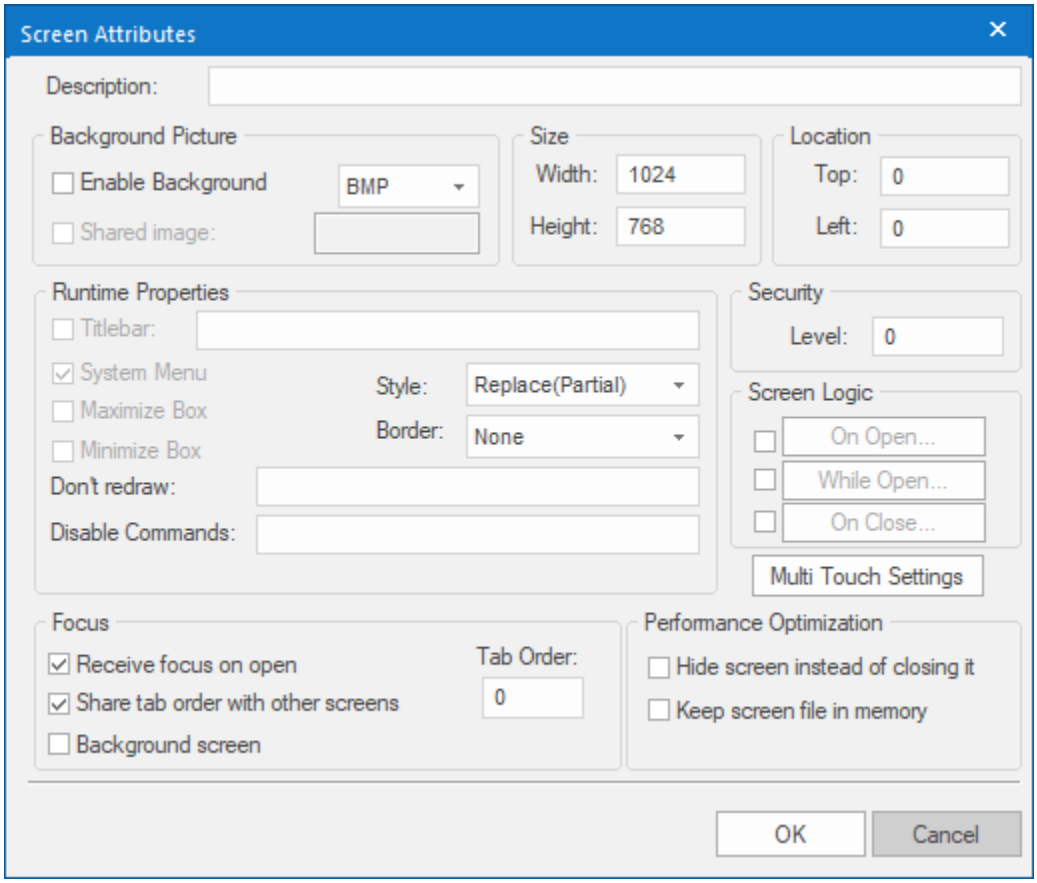
To open an existing Screen worksheet, expand the **Screens** folder and then double-click the worksheet.

## SCREEN ATTRIBUTES

Use the *Screen Attributes* dialog box to configure run-time settings — such as size, location, title bar, security level, and screen logic — for a specific project screen.

### Accessing the dialog box

The *Screen Attributes* dialog box is automatically displayed when you add a new Screen worksheet to your project.




You can also access the dialog box for an existing Screen worksheet (assuming the worksheet is open for editing) by doing one of the following:

- On the **Draw** tab of the ribbon, in the **Screen** group, click **Attributes**; or
- Right-click anywhere in the Screen worksheet, and then click **Screen Attributes** on the shortcut menu.

## Description

This is a brief description of the project screen, for your own convenience. It is not shown anywhere during run time.

## Background Picture

### Enable Background

Enables the background picture layer and specifies the image format. When this option is selected, a new image file with the same name as the screen is automatically created in the Screen sub-folder your project folder (e.g., `<project name>\Screen\<screen name>.BMP`). You can then edit the file that was created, replace it with another file that has the same name, or configure the **Shared Image** setting below to specify a file that has a different name.

### Shared Image

Specifies the name of an image file to use instead of the image file that has the same name as the screen. The file still must be located in the Screen sub-folder of your project folder. Do not include the file extension with the file name, because it is automatically determined by the image format. You can type a string expression for this setting (e.g., `{MyBackground}`).

For more information, see Modifying a screen's background color or image on page 235.

> **Note:** Only BMP, JPG, PNG, and PCX images are supported in projects running on Windows Embedded devices. However, each device's operating system must also be configured to support those image formats, and that configuration is typically done by the device manufacturer. BMP, JPG, and PNG are commonly used image formats, and they should be supported on most devices. PCX is less commonly used, and it might not be supported on some devices. For more information, see the manufacturer's documentation for your device.

## Size and Location

**Width**

The default width (in pixels) of the screen when it is opened.

**Height**

The default height (in pixels) of the screen when it is opened.

**Top**

The default distance (in pixels) between the top of the display and the top of the screen when the screen is opened. This is also known as the screen's Y-position.

**Left**

The default distance (in pixels) between the left side of the display and the left side of the screen when the screen is opened. This is also known as the screen's X-position.

If you use the Layout tool to lay out your project screens, these size and location settings will be automatically updated to reflect any changes you make in that tool.

If you configure the screen to have a border, the user will be able to change the size and location of the screen during run time. For more information, see **Border** below.

## Runtime Properties

**Style**

The general run-time behavior of the screen:

| Option | Description |
|---|---|
| **Overlapped** | Opens the screen without closing any other screens. |
| **Popup** | Forces the screen in front of all other screens but does not close them. |
| **Replace (Partial)** | Opens the screen and closes all other **Replace** screens that it partially covers. This is the default. |
| **Dialog** | Similar to **Popup**, except that the other screens are also disabled until the dialog is closed by the user. |
| **Replace (Complete)** | Similar to **Replace (Partial)**, except that it closes only other **Replace** screens that it completely covers. |

**Border**

The type of border around the screen:

| Option | Description |
|---|---|
| **None** | No border; the screen is a flat, immovable rectangle on the display. This is the default. |
| **Thin** | A thin border that makes the screen a movable window. Includes the title bar. |
| **Resizing** | A thick border that makes the screen a movable, resizable window. Includes the title bar. |

#### Titlebar

Shows the window's title bar with the specified window name. You can type a string expression for this setting (e.g., `{MyWindow}`). It is useful to specify a window name even if the title bar is not shown, because the window name is always included when the screen is printed.

#### System Menu

Shows a menu of basic window commands at the left end of the title bar.

#### Maximize Box

Shows the **Maximize** button at the right end of the title bar.

#### Minimize Box

Shows the **Minimize** button at the right end of the title bar.

#### Don't Redraw

While this tag/expression evaluates as TRUE (i.e., not 0), most animations (except Command animations; see **Disable Commands** below) in the screen are disabled. In other words, basic shapes will not be redrawn when their size, color, position, or visibility change. Active, data, and library objects (except symbols that include these animations) remain enabled, and all other screen graphics will continue to be updated.

#### Disable Commands

While this tag/expression evaluates as TRUE (i.e., not 0), Command animations in the screen are disabled. Typically, this means that clicking/tapping buttons in the screen will have no effect, although other types of objects can also have Command animations applied to them. Active, data, and library objects (except symbols that include Command animations) remain enabled, and all other screen graphics will continue to be updated.

### Security

#### Level

The minimum security level that a user must have in order to access this screen.

### Screen Logic

#### On Open

A list of expressions to be evaluated once when the screen is opened, similar to a Math worksheet.

#### While Open

A list of expressions to be continuously evaluated while the screen is open, similar to a Math worksheet. If you specify a trigger, however, then the list will be evaluated once each time it is triggered while the screen is open.

#### On Close

A list of expressions to be evaluated once when the screen is closed, similar to a Math worksheet.

### Multi-Touch Settings

Customizes the Multi-Touch settings for this screen. For more information, see About the Multi-Touch settings for project screens on page 352.

### Focus

#### Receive focus on open

When the screen is opened, the focus will automatically go to the first object in the screen (according to Object ID) that can receive focus, as if the user tabbed into the screen.

**Share tab order with other screens**

When the user tabs through the last object in the screen, the focus will go to the next open screen (according to **Tab Order** below) rather than back to the first object in the current screen.

**Tab Order**

Similar to Object ID for screen objects, this determines the tab order between screens when multiple screens are open. When the user tabs through the last object in a screen, the focus will go to the open screen with the next higher Tab Order number. Each screen should have a unique Tab Order number between 0 and 32767.

**Background screen**

When the user clicks on the screen, it remains in the background and is not brought in front of the other open screens. If more than one screen has this option selected, then the screens are arranged in tab order with the greatest Tab Order number being the furthest back.

### Performance Optimization

**Hide screen instead of closing it**

Closing the screen (by any user action or system process) in fact only makes it hidden, and reopening the screen makes it visible again. This makes the screen appear to open very quickly.

This option should be selected only for critical screens. If too many screens are kept open, regardless of whether they are visible or hidden, overall run-time performance will be affected.

**Keep screen file in memory**

When the screen is closed, the screen file is kept in memory so that it does not need to be reloaded from the hard drive when the screen is reopened. The screen still needs to be redrawn, however. This is not as fast as making a hidden screen visible again (see above), but it still makes the screen appear to open quickly.

This option should be selected only for important screens. If too many project files are kept in memory, overall run-time performance will be affected.

## MODIFYING A SCREEN'S BACKGROUND COLOR OR IMAGE

A project screen can have either a solid background color or an editable background image.

### Selecting a screen's background color

By default, a newly created project screen has a solid white background. To change this background color:

1. Make sure the screen file is open for editing.
2. On the **Draw** tab of the ribbon, in the **Screen** group, click **Background Color**. A standard color picker is displayed as a shortcut menu.
3. Use the color picker to select a color. The color is applied to the entire project screen.

> **Tip:** If you want to set a background color for only part of a screen, draw a shape object and then send it to the back.

### Enabling a screen's background image

To enable the background image for a screen and then edit it:

1. Make sure the screen file is open for editing.
2. On the Graphics tab of the ribbon, in the Screen group, click **Attributes**.The *Screen Attributes* dialog box is displayed.
3. Select **Enable Background**. A new BMP file with the same name as the screen is automatically saved in the Screen folder in your project folder (e.g., `[…]\<project name>\Screen\screenname.BMP`).
4. Click **OK** to close the *Screen Attributes* dialog box.

5. On the **Draw** tab of the ribbon, in the **Screen** group, click **Background Image**. Microsoft Paint is run automatically and the BMP file is opened for editing.
6. Use Microsoft Paint to edit the background image as needed.

> **Tip:** To use an image editor other than Microsoft Paint, manually edit the program settings file (`Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin\Program Settings.ini`) to add the following setting:
>
> ```
> [Options]
> ImageEditor=filepath
> ```

### Specifying an existing image file as the background

To select an existing image file, especially if it is in a format other than BMP:

1. Copy the image file that you want to use to the Screen folder in your project folder.
2. In the development environment, make sure the screen file is open for editing.
3. On the **Draw** tab of the ribbon, in the **Screen** group, click **Attributes**.The *Screen Attributes* dialog box is displayed.
4. Select **Enable Background**, and then in the list to the right of the option, select the image file format.

   > **Note:** Only BMP files are supported in projects developed for Windows Embedded target systems.

5. Select **Shared Image**, and then in the box to the right of the option, type the name of the image file. Do not include the file extension, because that is controlled by the list selection in the previous step.

   You can specify folders within the Screen folder. For example, if you type `MyBackgrounds\Background1`, the program will look for the image file at `[…]\<project name>\Screen\MyBackgrounds\Background1`.

   You can also specify a tag or expression in curly brackets (e.g., `{MyTag}`), in order to programmatically change the background image during run time.
6. Click **OK** to close the *Screen Attributes* dialog box. If the program can find and load the specified image file, the image will be displayed in the project screen. If not, then a warning message will be displayed.

## *Screen Group Folder*

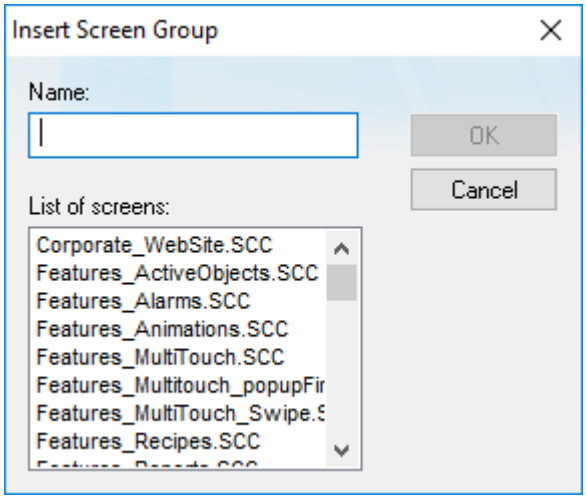
The *Screen Group* folder combines individual screens from the *Screens* folder into more manageable groups.

To open a specific screen group, open the *Screen Group* folder and right-click on the subfolder.

To remove a specific screen group, right-click on its subfolder and click the prompt screen to delete.

To create a new screen group:

1. On the Insert tab of the ribbon, in the Graphics group, click Screen Group to open the *Insert Screen Group* dialog:




***Insert a Screen Group dialog***

2. Type a name for the new folder into the **Name** field.
3. Create a group of screens for this folder by selecting screens from the **List of screens** list. To select multiple screens press the **Ctrl** key as you click on the screen names. Release the **Ctrl** key when you finish.

   This list contains only those screens currently located in *Screens* folder.
4. Click **OK** to close the Insert Screen Group dialog.

## *Lay out project screens in a simulation of the client's display*

Use the Layout tool to lay out — that is, to resize, reposition, and reorder — your project screens in a simulation of the client's display.

By default, when you create a new project screen, it is created at your project's full display resolution. (You selected the display resolution when you created the project itself, but you can also convert the resolution after the fact, if necessary.) Therefore, when you open two screens at the same time, the second screen will cover the first.

That works well enough for a simple project with only a few screens, but in a more complex project with many screens and sub-screens, you will want to arrange the screens so that they fit together and not overlap. You can use the Layout tool to do that.

When you make changes to a screen in the Layout tool, the screen's attributes are automatically updated to reflect the changes. For example, when you resize a screen, the screen's **Width** and **Height** attributes are updated. And when you reposition a screen, the screen's **Top** and **Left** attributes are updated. For more information, see Screen Attributes on page 231. The changes are not actually saved, however, until you close the screen file in the development environment.

To lay out your project screens:

1. In the **Graphics** tab of the Project Explorer, double-click **Layout**.
   The Layout tool is opened as a new tab in the Screen/Worksheet Editor area, and all currently open project screens are displayed in the tool. The order of the screens in the tool is determined by the order of screens' tabs in the editor area, so that the left-most tab is in the back and the right-most tab is in the front.
2. Make sure all of the project screens that you want to lay out are included in the Layout tool:
   a) To add a screen to the Layout tool, double-click it in the Project Explorer.
      The screen file is opened for editing, and it is also added to the Layout tool.
   b) To remove a screen from the Layout tool, click the Close icon (**×**) in the screen's tab.
      The screen file is saved and closed, and it is also removed from the Layout tool.

c) To reorder a screen in the Layout tool — that is, to bring it to the front or send it to the back — click and drag the screen's tab right or left in the editor area.

3. Configure the project settings to show or hide whichever elements of the Viewer program window — for example, the Title Bar, the Menu Bar, the window border, and so on — that you want have displayed to users during project run time.

   For more information, see Viewer tab on page 117.

   > **Note:** This applies only when the Windows-based Viewer program (i.e., either Secure Viewer or the local Viewer module) is used to view the project screens. It does not apply when Mobile Access is used, because then the browser is the program window.

   The simulated display in the Layout tool is updated to reflect the changes in the project settings.

4. Lay out your project screens as needed:
   a) To resize a screen, click and drag the edge of the screen.
   b) To reposition a screen, click and drag the middle of the screen.
   c) To arrange a screen, right-click in the screen, and then on the shortcut menu, click the appropriate command.
      For example, click **Top Left** to move the screen to the top-left corner of the full display.

   Each time you change a screen, a message is displayed asking you to confirm the change. You can choose to disable the messages and update the screen attributes without confirmation. You can also reenable the messages later, if necessary. For more information, see Preferences tab on page 129.

When you have finished laying out your project screens, you might want to save them as a screen group so that you can open them all together, at the same time, with a single command. For more information, see Screen Group Folder on page 236.

## Screen Objects and Animations

### *Editing*

The *Editing* group provides tools for general screen editing.

#### SELECTION

On the Graphics tab of the ribbon, in the Editing group, click **Selection** to display a mouse cursor that you can use to select and move objects on the screen.

#### DISABLING DRAG IN A SCREEN

You can disable the dragging of objects in the screen editor, to prevent accidental moves after you've layed out the screen exactly as you want it.

On the **Draw** tab of the ribbon, in the **Editing** group, click **Disable Drag**.

#### REPLACING PROJECT TAGS IN A DOCUMENT OR SCREEN OBJECT

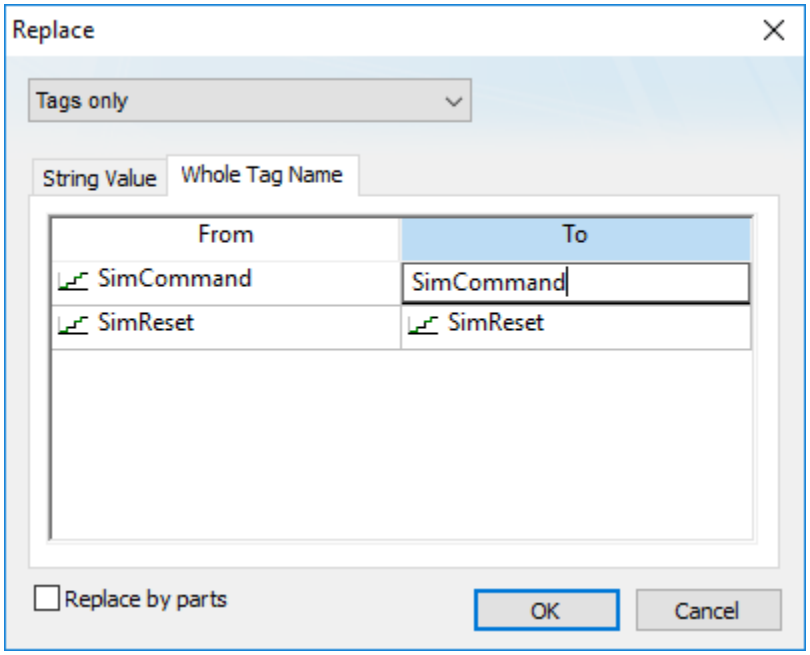
To replace all occurences of a tag in the current document, do one of the following:

- On the Home tab of the ribbon, in the Tags group, click **Replace**; or
- On the Graphics tab of the ribbon, in the Editing group, click **Replace**.

To replace all occurences of a tag in a screen object, double-click the object to open its *Object Properties* dialog and then click **Replace**.

All of these methods will open the *Replace* dialog, which is descibed below.

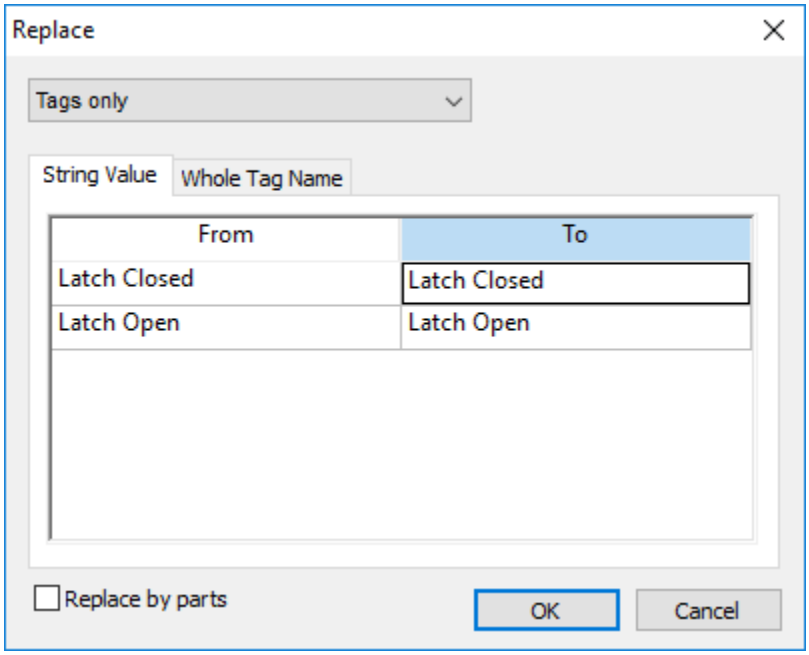
You can replace one or more tags by clicking the **Whole Tag Name** tab. Current tags used are displayed. The original tag names are shown in the **From** column on the left, and you can enter your new tag names in the **To** column on the right.




***Whole Tag Name tab***

Note that this does not *rename* or *delete* any tag — it only replaces the tags used in the object with other tags from the database.

You can also replace one or more strings (e.g., button captions, descriptive text) by clicking the **String Value** tab.




***String Value tab***

When you are done, click **OK**.

## OBJECT PROPERTIES DIALOG BOX

The *Object Properties* dialog box shows the configurable properties of a screen object or animation. Each type of object has its own object-specific properties, but all types have a few properties in common.

### Accessing the dialog box

To access the *Object Properties* dialog box for a screen object, do one of the following:

- Select the screen object, and then on the **Draw** tab of the ribbon, in the **Editing** group, click **Properties**;
- Right-click the screen object, and then click **Properties** on the shortcut menu; or
- Double-click the screen object.

### The dialog box in detail

All *Object Properties* dialog boxes contain the following elements:

**(Pin)**

Click this button to "pin" the dialog box, so that it remains open and active when you select other objects in the screen editor. For more information, see Focusing the Object Properties Window on page 95.

**Replace**

Click this button to open the *Replace* dialog box, which you can use to replace strings, tags, or properties in the selected object. For more information, see Replacing project tags in a document or screen object on page 92.

**Hint**

Type a hint or tooltip that will be displayed during run time, when the user hovers the mouse cursor over the object. This can be used to provide quick-help to the user.

The text in the **Hint** box is also temporarily written to the system tag **`Hint`**, so that you can trigger actions based on the value of this tag when the mouse cursor is moved over a specific object.

To show hints/tooltips during run time, the **Enable Tooltip** option must be selected in the project settings. You can enable/disable this feature separately for full project viewers (on the **Project** tab of the ribbon, in the **Settings** group, click **Viewer**) and for thin clients (on the **Project** tab of the ribbon, in the **Web** group, click **Web**).

**(Object Selector)**

Use this list at the top-right corner of the dialog box to select the specific object or animation in a group of objects that you want to configure. When you select another object, the dialog box immediately changes to show the properties of that object.

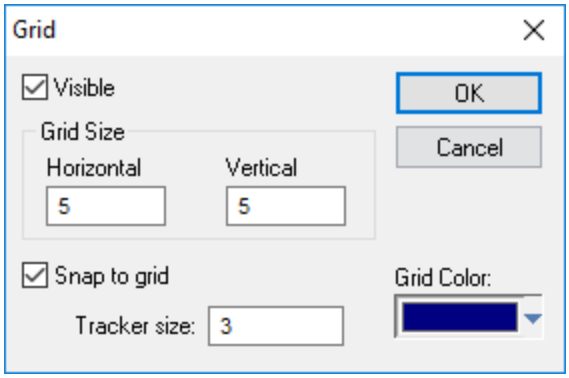
## GRID SETTINGS

To show/hide the grid in the screen editor, click **Grid Settings** on the Graphics tab of the ribbon and then click **View Gridlines** on the shortcut menu.

To edit the grid settings, do one of the following:

- Click **Grid Settings** on the Graphics tab of the ribbon and then click **Grid Settings** on the shortcut menu; or
- Right-click anywhere in the screen editor and then click **Grid Settings** on the shortcut menu.

Either method will open the *Grid Settings* dialog:




***Grid Settings dialog***

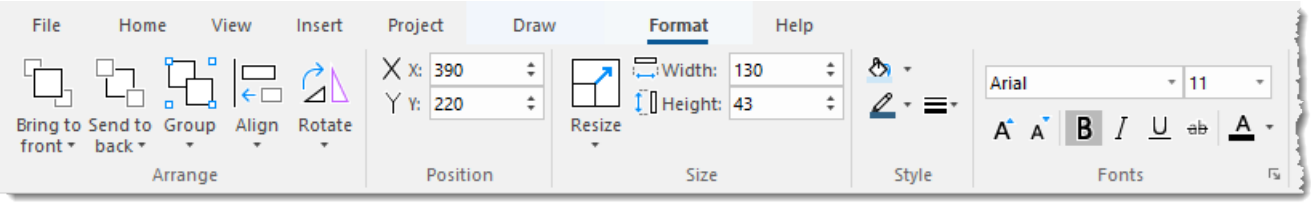
## UNDO

Select **Undo** to cancel the last action performed (and up to 20 actions taken prior to the last action) while working on a screen. (*Object Properties* actions do not increase **Undo** steps.)

> **Note:** Using the **Undo** menu option is the same as using **Undo** tool located on the *Standard* toolbar.

## FORMAT TAB

The **Format** tab of the ribbon is used to format and arrange objects in a project screen.




> **Note:** This tab is available only when you've selected one or more objects in a project screen.

The tools are organized into the following groups:

- **Arrange**: Arrange objects in a project screen, including bring to front and send to back, group, align, and rotate.
- **Position**: Precisely adjust the position of a screen object in a project screen.
- **Size**: Precisely adjust the size of a screen object.
- **Style**: Change the fill and line color of a screen object.
- **Fonts**: Change the caption font of a screen object.

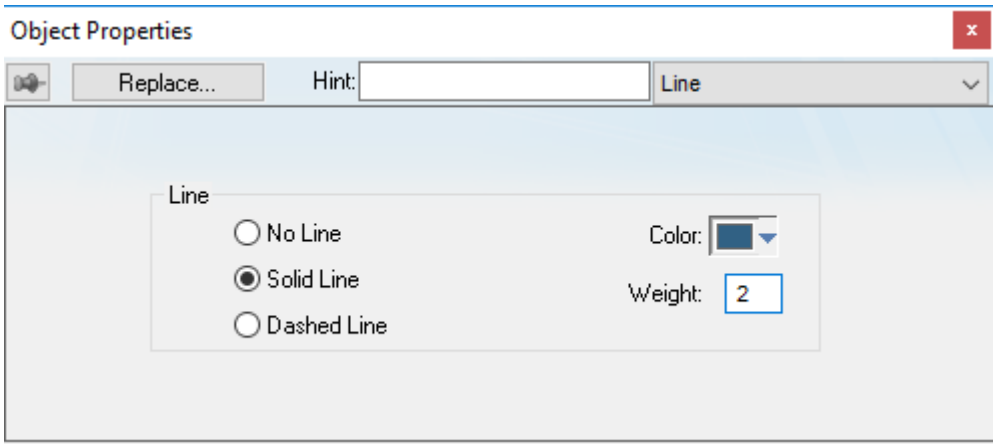
## *Shapes*

The *Shapes* group provides the following tools, which you can use to create polygons, rectangles, lines, and other objects for your screen.

### LINE OBJECT

On the **Graphics** tab, in the **Shapes** group, click **Line** to draw an orthogonal line in the drawing area, as follows:

1. Click the left mouse button to set the starting point of the line.
2. Drag the cursor to adjust the line size.
3. Click again to place the object.
4. To view the object properties, double-click on the object. The *Object Properties* dialog displays as follows.




***Object Properties: Line***

Use the *Object Properties* dialog to specify the following parameters for the orthogonal line:

- **Line**: Specify a line style by clicking the **No Line**, **Solid Line**, or **Dashed Line** button.
- **Color**: Specify a line color by clicking the **Color** button. When the *Color* dialog opens, click a color to select it and then close the dialog.
- **Weight**: Specify the line width (in pixels) by typing a number representing the line width into the text box.

> **Note:** Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

### OPEN POLYGON OBJECT

On the **Graphics** tab, in the **Shapes** group, click **Open Polygon** to draw an open polygon with a border in the specified foreground color.

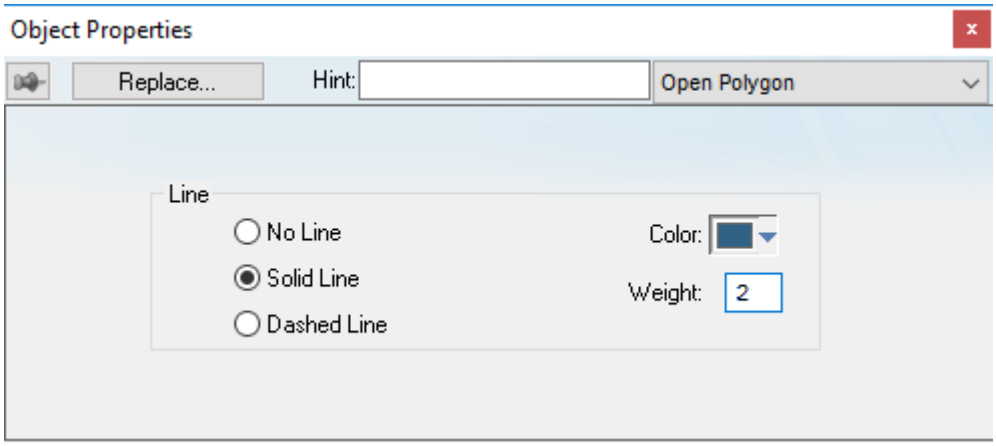
To draw an open polygon in the drawing area:

1. Click the left mouse button to set the starting point of the polygon.
2. Move the cursor to a new location and click again to place the second vertex.
3. Repeat this process until you create the desired polygon shape.
4. Double-click to stop drawing the polygon.

   To change the shape of a polygon after you've drawn it, select it and drag any of its points.

> **Tip:** If a polygon's individual points are not draggable, they may be grouped. To ungroup the points, right-click on the polygon and choose **Ungroup** from the shortcut menu.

To view the object properties, double-click on the polygon object and the *Object Properties* dialog is displays as follows.




***Object Properties: Open Polygon***

Use the *Object Properties* dialog to specify the following parameters for the polygon:

- **Line**: Specify a border line style by clicking the **No Line**, **Solid Line**, or **Dashed Line** button.
- **Color**: Specify a border line color by clicking the **Color** button. When the Color dialog opens, click on a color to select it and then close the dialog.
- **Weight**: Specify the borderline width (in pixels) by typing a number representing the line width into the text box.

> **Note:** Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

## CLOSED POLYGON OBJECT

On the **Graphics** tab, in the **Shapes** group, click **Closed Polygon** to draw a closed polygon, using a border in the specified foreground color.

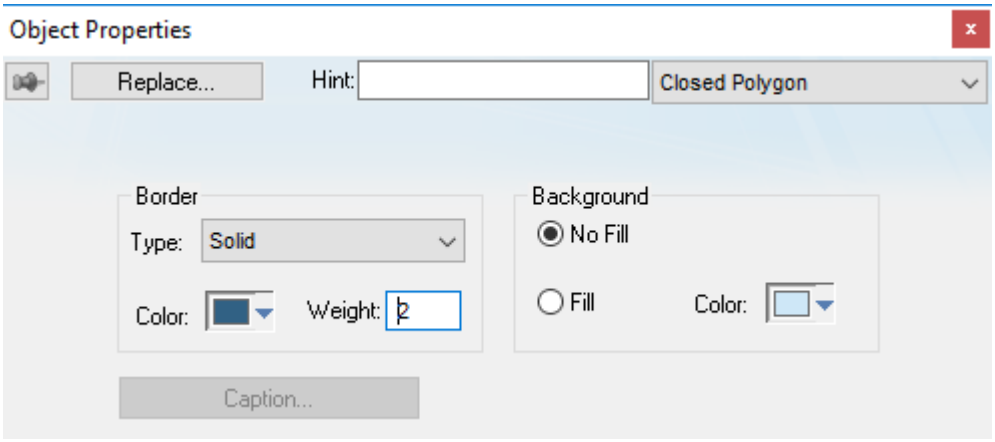
To draw a closed polygon in the drawing area:

1. Click the left mouse button to set the starting point of the polygon.
2. Move the cursor to a new location and click again to place the second point.
3. Repeat this process until you create the desired polygon shape.
4. Double-click or right-click to stop drawing the polygon.
5. To view the object properties, double-click on the polygon object.

   To change the shape of a polygon after you've drawn it, select it and drag any of its points.

> **Tip:** If a polygon's individual points are not draggable, they may be grouped. To ungroup the points, right-click on the polygon and choose **Ungroup** from the shortcut menu.

The *Object Properties* dialog is displays as follows.




***Object Properties Dialog: Closed Polygon***

Use the *Object Properties* dialog to specify the following parameters for the polygon:

- **Line**: Specify a border line style by clicking the **No Line**, **Solid Line**, or **Dashed Line** button.
- **Color**: Specify a border line color by clicking the **Color** button. When the *Color* dialog opens, click a color to select it and then close the dialog.
- **Weight**: Specify the borderline width (in pixels) by typing a number representing the line width into the text box.

> **Note:** Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

- **Fill**: To specify whether the polygon is filled, click **No Fill** or **Fill**.

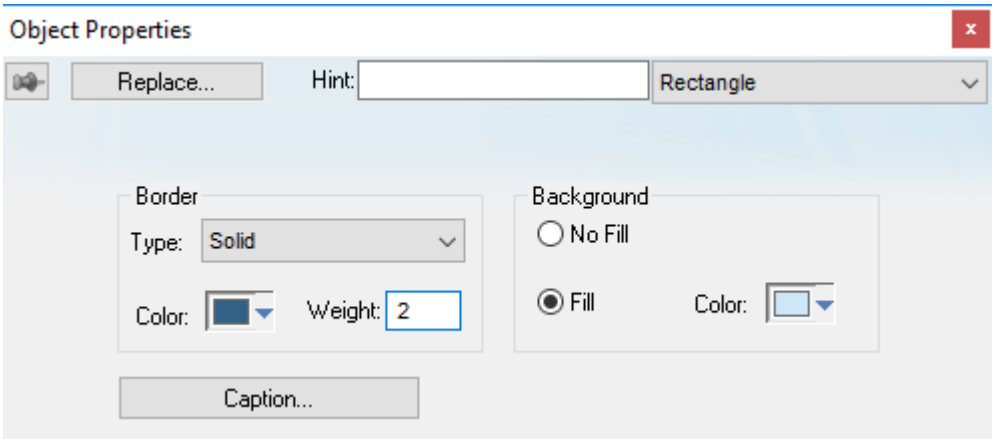
  If you enable the **Fill** option, you can specify a fill Color by clicking on the **Color** button. When the *Color* dialog displays, click a color to select it and close the dialog.

## RECTANGLE OBJECT

On the **Graphics** tab, in the **Shapes** group, click **Rectangle** to create rectangles, as follows:

1. Click in the drawing area and drag the mouse/cursor to draw the rectangle.
2. Release the mouse button when the rectangle is the size you want.

3. Double-click on the object to view the *Object Properties* dialog.




***Object Properties: Rectangle***

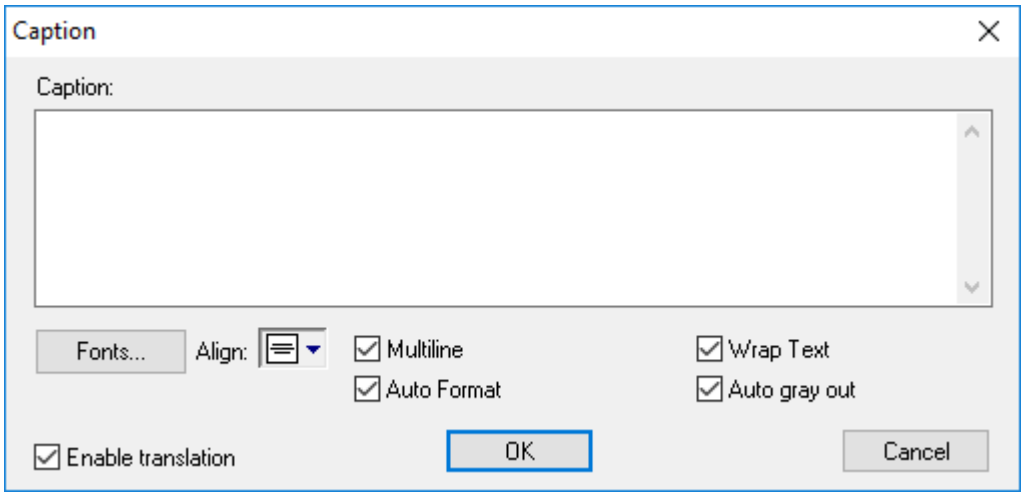
Use the *Object Properties* dialog to specify the following parameters for the orthogonal line:

- **Type**: Specify a border line style by clicking on **None**, **Solid**, **Dashed**, **Etched**, **Raised** or **Sunken**.
- **Color**: Specify a border line color by clicking the **Color** button to open the *Color* dialog. Click the color to select it, and then close the dialog.
- **Weight**: Specify a border line width by typing a number representing the line width (in pixels) into the text box provided.

> **Note:** Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

- **Fill**: Specify whether to fill the rectangle by clicking **No Fill** or **Fill**.

  If you select the **Fill** option, specify a fill color by clicking on the Color rectangle. When the Color dialog displays, click a color to select it and close the dialog.
- **Color**: Specify a fill color by clicking the **Color** button to open the *Color* dialog. Click a color to select it, then close the dialog.
- **Caption**: Press this button to open the *Caption* dialog where you can edit the text that can be written inside the rectangle object:



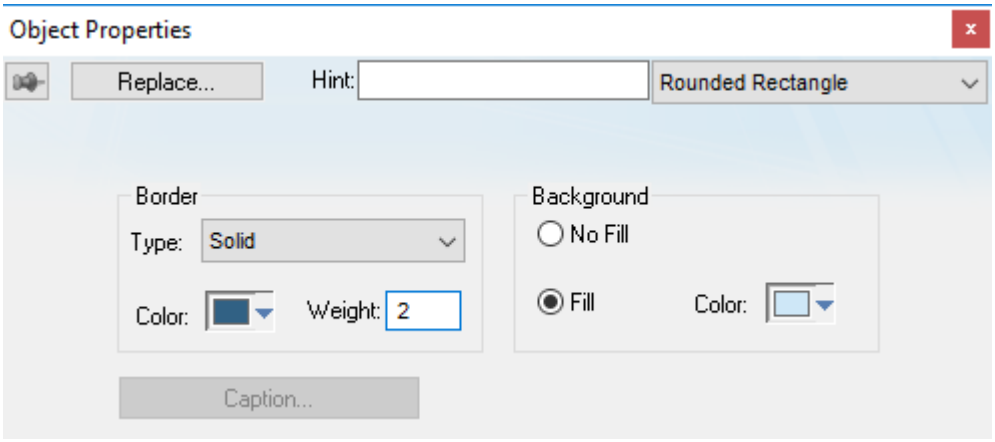

***Caption dialog***

- **Caption**: Enter the text that you want to display inside the rectangle object. You can include a tag by enclosing it in curly brackets (e.g., **{*tagname*}**).
- **Fonts**: Specify a font style for the caption by clicking the **Fonts** button.
- **Align**: Specify the alignment for the caption of the rectangle.
- **Multiline**: Allow the caption of the rectangle to be shown in more than one line, when checked.
- **Auto Format**: When checked, if the caption includes a decimal value enclosed by curly brackets (e.g., **{1.2345}**) or a tag of Real type (see **Caption** above), then the value will be formatted according to the virtual table created by the `SetDecimalPoints` function.
- **Wrap Text**: When checked, the object automatically wraps the text when necessary.
- **Auto gray out**: Turns the caption of the rectangle to gray when the Command animation applied to the rectangle is disabled by the **Disable** field or due to the Security System.
- **Enable translation**: Click (check) to enable translation during runtime using the Translation Tool.

## ROUNDED RECTANGLE OBJECT

On the **Graphics** tab, in the **Shapes** group, click **Rounded Rectangle** to draw rounded rectangles (empty or filled), as follows:

1. Click in the drawing area and drag the mouse/cursor to create the rectangle.
2. Release the mouse button to stop drawing the object.
3. Double-click on the object to view the *Object Properties* dialog.




***Object Properties: Rounded Rectangle***

> **Note:** You cannot use the rounded rectangle tool to create a bar graph for projects running on Windows Embedded target systems.

> **Tip:** A rounded rectangle has one extra handle in the bottom-right corner, which enables you to modify the arc angle.

Use the *Object Properties* dialog to specify the following parameters for the orthogonal line:

- **Line**: Specify a borderline style by clicking the **No Line**, **Solid Line**, or **Dashed Line** button.
- **Color**: Specify a borderline color by clicking the **Color** button to open the *Color* dialog. Click the color to select it and then close the dialog.
- **Weight**: Specify a borderline width by typing a number representing the line width (in pixels) into the text box provided.

> **Note:** Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

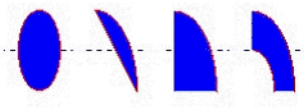
- **Fill**: Specify whether the rectangle is filled by clicking **No Fill** or **Fill**.

  If you select the **Fill** option, specify a fill color by clicking on the **Color** button. When the *Color* dialog displays, click a color to select it and close the dialog.

- **Color**: Specify a fill color by clicking the **Color** button to open the *Color* dialog. Click a color to select it, then close the dialog.
- **Caption**: This option is not enabled for this object.

## ELLIPSE OBJECT

On the **Graphics** tab, in the **Shapes** group, click **Ellipse** to draw ellipses, chords, arcs, and rings (see the following figures).

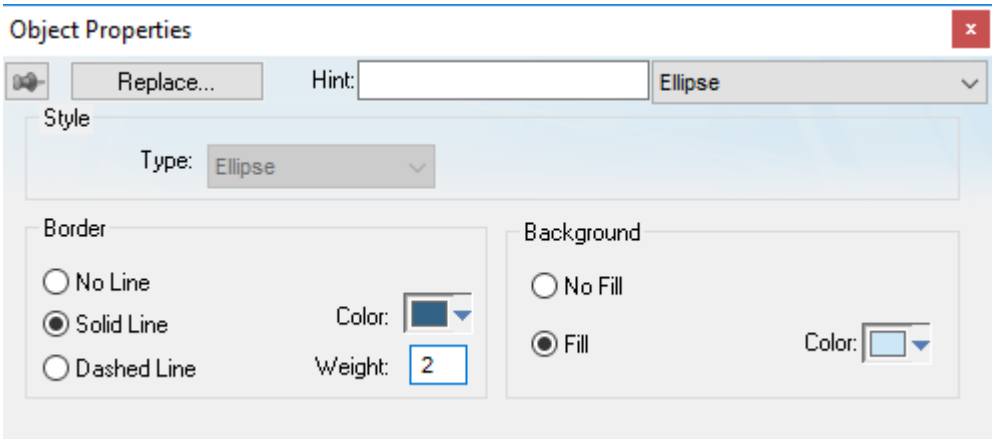


***Ellipse, Chord, Arc, and Ring***

> **Tip:** The *Ring* style is particularly useful when you are creating plumbing drawings.

To create an ellipse, use the following steps:

1. Click in the drawing area and drag the mouse/cursor to create an ellipse shape.
2. Release the mouse button to stop drawing the ellipse.
3. Use the *Object Properties* dialog to change the shape to a chord, arc, or ring.
4. Double-click on the object to view the *Object Properties* dialog.




***Object Properties: Ellipse***

Use the *Object Properties* dialog to specify the following parameters for the ellipse:

- **Style**: Specify the object style by selecting **Ellipse**, **Arc**, **Chord**, or **Ring** from the drop-down list. Next, select **Left-Bottom**, **Left-Top**, **Right-Bottom**, or **Right-Top** from the **Style** list to choose the quadrant into which the ellipse is drawn.

  For example to represent a half-circle pipe, create two **Ring** objects. Specify one as **Left-Bottom** and the other as **Right-Bottom** then join the two objects to create a half-pipe.

- **Line**: Specify a line style for the ellipse border by clicking the **No Line**, **Solid Line**, or **Dashed Line** button.
- **Color**: Specify the ellipse borderline color by clicking the **Color** button to open the *Color* dialog. Click the color to select it, then close the dialog.
- **Weight**: Specify a line width for the ellipse border by typing a number representing the line width (in pixels) into the text box provided.

> **Note:** Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

- **Fill**: To specify whether the ellipse is filled, click **No Fill** or **Fill**.

  If you select the **Fill** option, specify a fill color by clicking on the **Color** button. When the *Color* dialog displays, click on a color to select it and close the dialog.

## PASTE A BITMAP IMAGE INTO A SCREEN

To paste a bitmap image into a screen, copy it to the clipboard and then paste it directly into the Screen worksheet.

This task assumes that you have a Screen worksheet open for editing.

Please note that by default, using this method to add an image to a project screen will embed the image data in the screen file. This keeps everything in a single file, which is more convenient in some situations, but it increases the screen file size each time you reuse the image and it is also less flexible than linking to an external image file. Nevertheless, if this is what you want to do, then proceed with the task below.

If this is not what you want to do, however, then you have two other options.

First, if you want all bitmap images to be linked as external files, then you should first select the option **Save pictures in separate files**, in the project settings. When that option is selected and you paste a bitmap image into a screen, the image is automatically saved as a separate file in your project folder and then placed as a Linked Picture object in the project screen. For more information about the option **Save pictures in separate files**, see Project Settings: Viewer.

Second, if you want only a specific bitmap image to be linked as an external file, then you should not paste it into the screen. Instead, you should manually place it as a Linked Picture object. This leaves you the ability to paste and embed other bitmap images, according to the default settings.

The following table summarizes your options:

| Action | Save pictures in separate files... | |
|---|---|---|
| | ...is selected | ...is not selected |
| Paste image into screen | Image is placed as a Linked Picture object | Image is placed as a Bitmap object |
| Link to external image file | Image is placed as a Linked Picture object | Image is placed as a Linked Picture object |

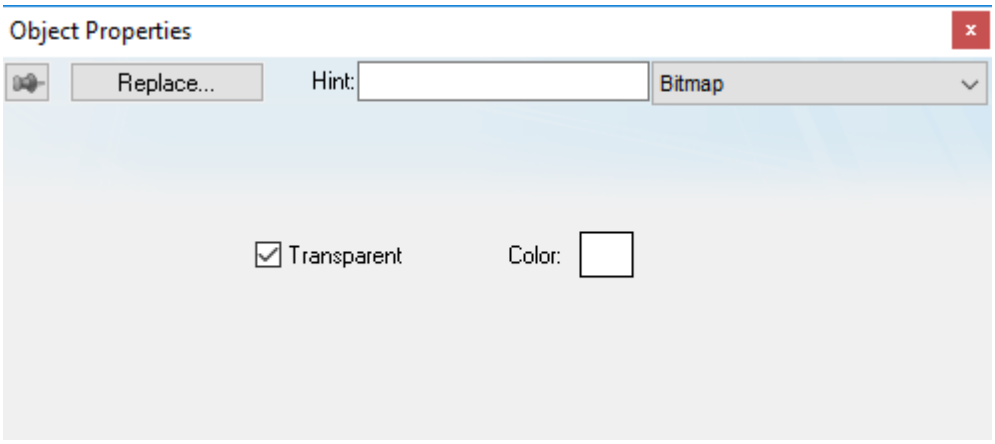
To paste a bitmap image into a screen:

1. Open the desired image in an appropriate image editor, such as Microsoft Paint.
2. Select part or all of the image, either by using the **Select** tool in Paint or by pressing **Ctrl+A**.
3. Copy the selection to the clipboard, either by using the **Copy** tool in Paint or by pressing **Ctrl+C**.
4. Switch to the development application, and then make sure that you have the correct Screen worksheet open for editing.
5. On the **Home** tab of the ribbon, in the **Clipboard** group, click **Paste**, or press **Ctrl+V**.
   The bitmap image is placed as a screen object.
6. Click and drag the object to where you want it to be positioned in the screen. Also, click and drag the object's handles to resize it, if necessary.
7. Double-click the screen object.

   If **Save pictures in separate files** is selected in the project settings, then the bitmap image was placed as a Linked Picture screen object. Skip the remaining steps of this task and instead proceed to Link to an external image file on page 313.

On the other hand, if **Save pictures in separate files** is not selected, then the bitmap image was placed as a Bitmap screen object. Proceed with the remaining steps below.

The *Object Properties: Bitmap* dialog is displayed.




***Object Properties: Bitmap***

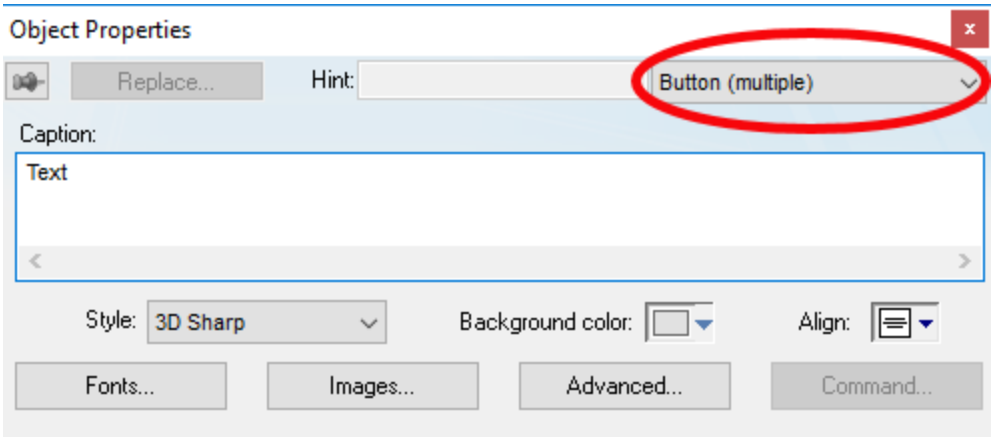
8. If you want some part of the picture to be transparent to the screen background and other objects, then select a transparent color:
   a) Select **Transparent**.
   b) Click and drag the tracker on the screen object until it is positioned over a sample of the desired transparent color.
      The tracker is an additional handle on the screen object that initially appears just inside the bottom-right corner of the object. Moving the tracker on the object does not move or resize the object itself.
9. Close the *Object Properties* dialog.

## CHANGE THE PROPERTIES OF MULTIPLE SCREEN OBJECTS

This task describes how to select two or more screen objects and then change the properties that are common to the selected objects.

Before you begin this task, you must have a project screen open in the screen editor.

Which properties you can change depends on whether you select multiple objects of the same type or of different types. If the objects are of the same type, you can change the properties that are specific to that type. For example, if you select multiple Button objects, then you can change the properties that are specific to Button objects.

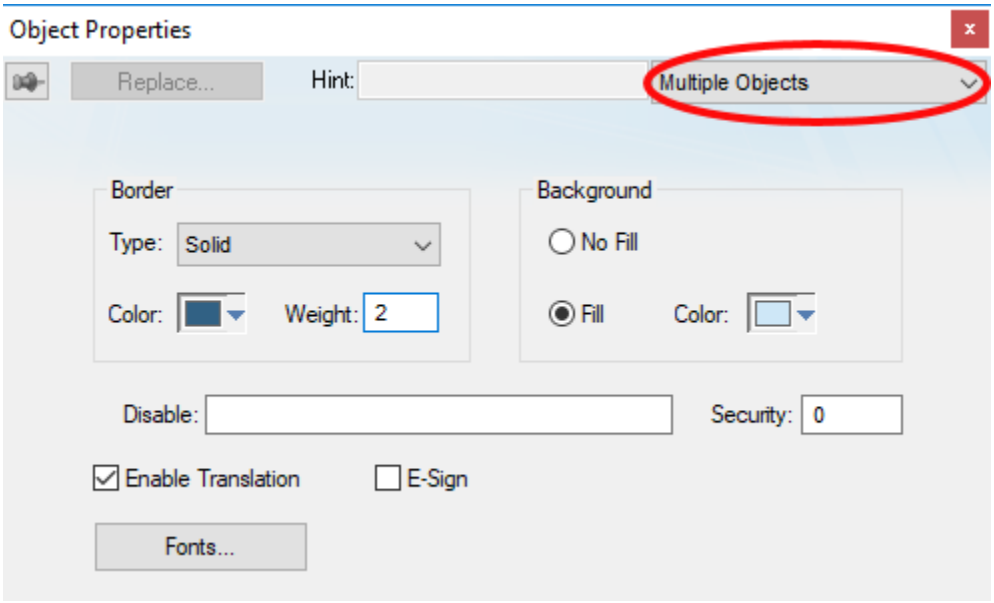



***Object properties of multiple selected Button objects***

For more information about the properties of a specific type of object, see the documentation for that object.

> **Note:** You can only use this method to change the properties of Shapes and Active Objects. You cannot use this method to change the properties of Data Objects, Animations, Library items, or objects in a group.

In contrast, if you select multiple objects of different types, you can change the properties that are common to all of the objects. This includes not only cosmetic properties like Border and Background, but also functional properties like Disable, Security, Enable Translation, and E-Sign. (Some properties may not apply to all objects. For example, Button objects do not have Border and Rectangle objects do not have Security.)




***Object properties for multiple objects of different types***

In both cases, the dialog box shows the current values of the properties of the last selected object.

It is only when you actually change the value of a property that the change is applied to the selected objects. All other properties are left unchanged, regardless the values shown in the dialog box.

To change the properties of multiple screen objects:

1. In the screen editor, do one of the following:
   - Press and hold either **Shift** or **Ctrl** on the keyboard, and then click each object that you want to change; or
   - Use the cursor to draw a selection box around all of the objects that you want to change.

   The objects are selected.
2. Do one of the following:
   - On the **Draw** tab of the ribbon, in the **Editing** group, click **Properties**;
   - Right-click the selected objects, and then on the shortcut menu, click **Properties**; or
   - Press **Alt+Enter** on the keyboard.

   > **Note:** You cannot double-click to open the *Object Properties* dialog box as you otherwise would, because clicking like that clears the selection.

   The *Object Properties* dialog box is displayed for the selected objects.
3. Change the property values that you want to change, and then close the dialog box.

The changes are applied to all of the selected objects.

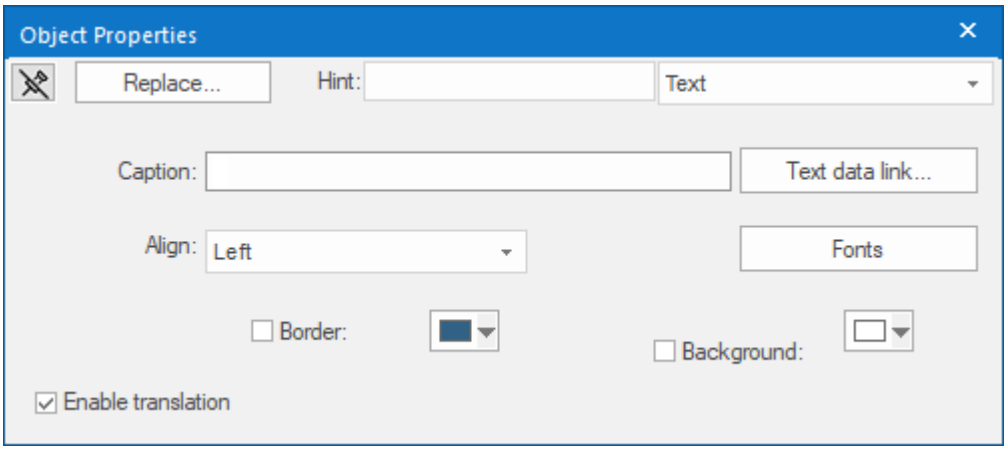
## *Active Objects*

The *Active Objects* toolbar provides the following tools, which you can use to create interactive objects. Active objects typically require more parameters than simple shapes.

### TEXT OBJECT

On the **Graphics** tab, in the **Active Objects** group, click **Text** to create text objects, as follows:

1. Click in the drawing area. When a cursor displays, you can type a line of text.
2. After entering the text string, double-click on the new text object to view the *Object Properties* dialog.




***Object Properties: Text***

Use the *Object Properties* dialog to specify the following properties:

- **Caption**: Specify a text string by typing a caption in the text box.
- **Text data link** button: Click to apply the Text Data Link animation to the Text object.

  If the caption doesn't include any placeholder characters (**###**) for the text-data link, then clicking this button also automatically appends those characters.
- **Align**: Align the text by selecting **Left**, **Center**, or **Right** from the combo-box.
- **Fonts**: Specify a font style for the text by clicking the **Fonts** button. When the *Fonts* dialog displays, you can specify the following parameters:
  - **Font** (typeface)
  - **Font style**
  - **Size**
  - **Effects**
  - **Color**
  - **Script**
- **Border**: Specify a text border by clicking the **Border** box.

  To select a border color, click the **Color** rectangle. When the *Color* dialog displays, click a color to select it, then close the dialog.
- **Background**: Specify a background color by clicking the **Color** button. When the *Color* dialog displays, click a color to select it, then close the dialog.
- **Enable translation** (optional): Specify an external translation file for the text by clicking (checking) this box.

## TEXT BOX OBJECT

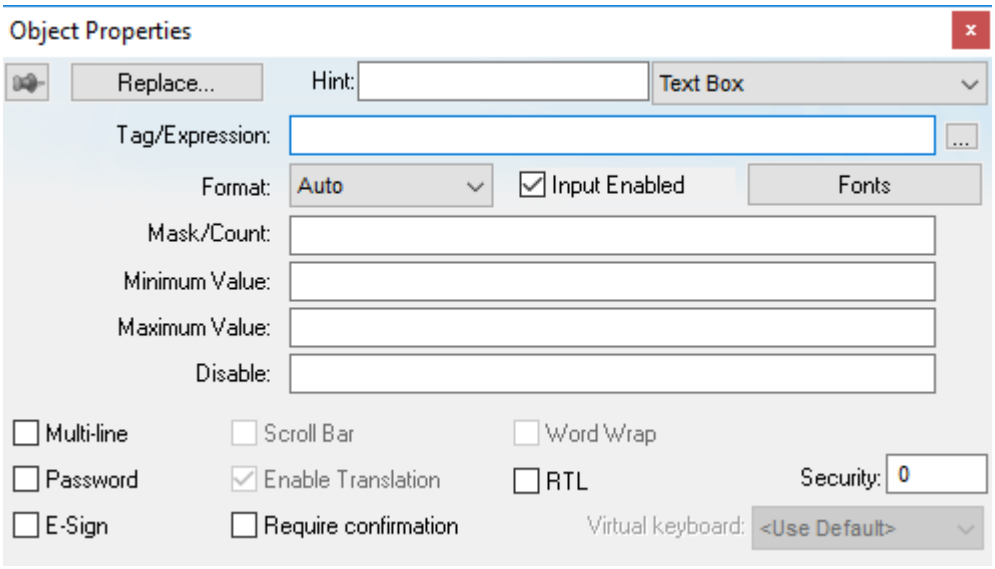
Draw a Text Box object to create a way to input and/or output text, including multiple lines.

This object is an OS-style text input/output box that can be configured to show multiple lines. When the object is associated with a String array, each line of the box corresponds to an array element: Line 1 is Array Index 0, Line 2 is Array Index 1, Line 3 is Array Index 2, and so on.

Otherwise, the Text Box object works much like the Text Data Link animation.

To draw and configure a Text Box object:

1. Open a screen for editing.
2. On the **Draw** tab of the ribbon, in the **Active Objects** group, click **Text Box**.
   The mouse cursor changes to a crosshair for drawing.
3. Draw the object where you want it on the screen, and then further move or resize it if necessary.
4. Double-click the object.
   The *Object Properties* dialog box is displayed.




***Text Box object properties***

5. In the **Hint** box, type a hint or tooltip that you want to have displayed during run time when the mouse hovers over the object.
6. In the **Tag/Expression** box, type the name of a project tag or an expression to associate it with the object.

> **Tip:** Click the browse button (...) on the right to use the Object Finder to form the tag/expression.

   A project tag can be used for either input or output. An expression can be used only for output.

   If you want to input or output multiple lines of text — that is, if you select **Multi-line**, which is described below — then type the name of a String array with an index of 0 (e.g., `MyStringArray[0]`). If you try to start at any other position, then you may see unexpected behavior during run time. For example, if you type `MyStringArray[4]`, then the first line of the object will display index 4 of the array but the second line will start over at index 0.

   You should not use any other type of array (i.e., Boolean, Integer, or Real) with the **Multi-line** option.

7. In the **Format** list, select how the numerical value (if any) of the specified tag/expression will be formatted and displayed on-screen. Available options include **Decimal**, **Hexadecimal**, **Binary** and **Auto**. If you select **Auto**, then the value will be formatted according to the virtual table created by the `SetDecimalPoints` function.

This option does not actually change the specified tag/expression in any way. For example, **Tag/Expression** is set to an Integer tag, **Input Enabled** is selected, and **Format** is set to **Hexadecimal**. You may input a new value in hexadecimal format, but it is saved in your project database as an integer.

8. By default, **Input Enabled** is selected. If you do not want to allow user input during run time, clear this check box.

> **Note:** If you clear the **Input Enabled** check box, the object's font color becomes black and its background color becomes gray. This is to visually indicate to the user that input is disabled, and it overrides the object's own font and color settings, including additional settings applied a Color animation.

9. In the **Mask/Count** box, type a value that will restrict the input:
   - To mask a numerical value so that it matches a specific format, type one or more # characters. Each # represents one character of input/output. You may include a decimal separator for decimal values (e.g., ###.##).

     It is important to remember that the project runtime will always display the most significant digits of a numerical value, regardless of the number or placement of # characters in the text. That means if you do not have sufficient # characters to display the value, then it will be transformed in some way depending on the format of the value (as set by the **Format** option described below):
     - In **Decimal** format, the number of decimal places is determined by the position of the decimal separator in the ### text. However, if you do not have enough # characters to the left of the decimal separator to display the whole value, then the whole value will overrun the fractional value. For example, if you try to display a value of 112.64 in #.##, you will see 112.
     - In **Hexadecimal** and **Binary** formats, if you have more # characters than you need to display the value, then the runtime project will fill in with leading zeroes. If you have less characters than you need, then the value will simply be truncated.
     - In **Auto** format, the runtime project will ignore the number of # characters and display the entire numeric or string value. Numeric values will be displayed in decimal format with their complete whole and fractional values, regardless of the placement of the decimal separator in the ### text. Given an exceptionally large value or long string, this may disrupt the layout of your screens.
   - To limit a string to specific character count, type a value between 0 and 1024. 1024 is the maximum limit because it is the maximum size of the String data type. If you enter more than 1024 characters, then the string will be truncated and the remaining characters will be discarded.

     Please note that this character count limit is per line, so if you configure the object to have multiple lines (i.e., if you select **Multi-line**) and associate it with a String array, then each line of text / array element may be up to the limit.

10. In the **Minimum Value** and **Maximum Value** fields, type the minimum and maximum numerical values (if any) that will be accepted from the user. This is optional.

11. In the **Disable** box, type the name of project tag or an expression. This is optional.
    When the value of the tag/expression is TRUE (1), the object is disabled.

12. To make the object accept/display multiple lines of text, select **Multi-line** .
    When **Multi-line** is selected, the **Scroll Bar** and **Word Wrap** options also become available and the **Password** option becomes unavailable.

13. To make the object obfuscate text input (e.g., ********), select **Password**.

> **Tip:** During run time, if you want to insert a line break in a multi-line text box, press **Shift+Return**, **Ctrl +Return**, or **Alt+Return**.

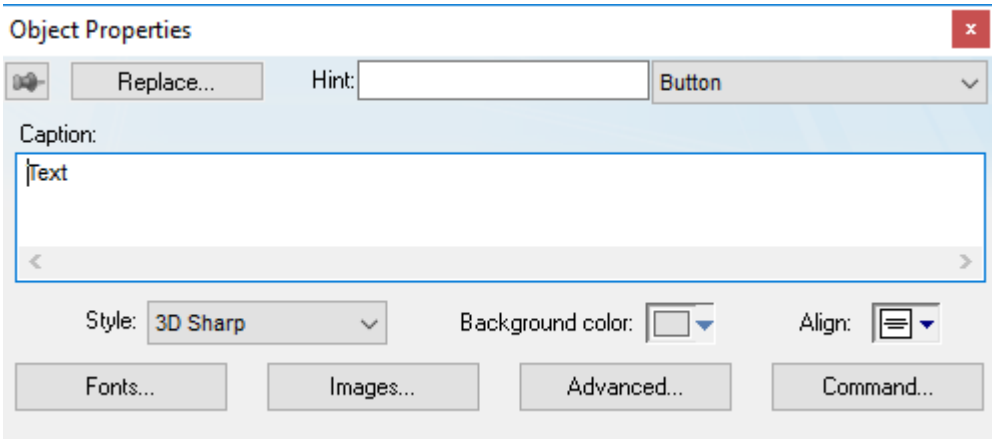
## BUTTON OBJECT

The Button object

On the **Graphics** tab, in the **Active Objects** group, click **Button** to create custom-sized buttons, as follows:

1. Click in the drawing area and drag the mouse/cursor to create the button shape.

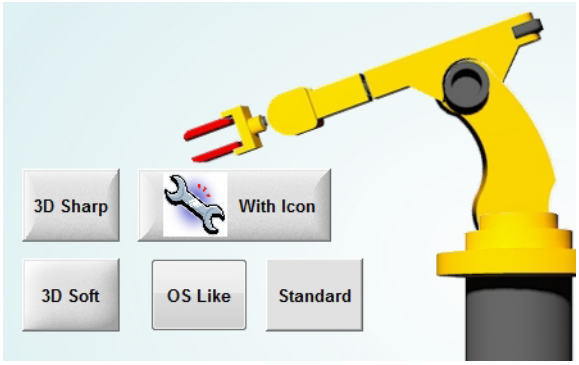
2. Release the mouse button when the button is the size you want.
3. Double-click on the object to view the *Object Properties* dialog.




***Object Properties: Button***

Use the *Object Properties* dialog to specify the following parameters for the button:

- **Caption**: Specify a caption by typing the text into the text box. You can include a tag by enclosing it in curly brackets (e.g., **{*tagname*}**).
- **Style**: Select a style for the button:
  - **3D Sharp**: A raised, rounded button with somewhat sharpened corners, suitable for touchscreen displays.
  - **3D Soft**: A raised, rounded button with softened corners, suitable for touchscreen displays.
  - **OS Like**: A button styled to match the operating system on which the project client is running, suitable for Windows desktops that have the Thin Client software installed.
  - **Standard**: The standard, flat button from the previous versions of EcoStruxure Machine SCADA Expert.




***Examples of button styles***

- **Background color**: Select a background color for the button.
- **Align**: Select the alignment for the caption of the button.
- **Fonts**: Specify a font style for the caption by clicking the **Fonts**.

  When the *Fonts* dialog is displayed, specify the following parameters:
  - **Font** (typeface)
  - **Font style**
  - **Size**
  - **Effects**

  - **Color**
  - **Script style**
- **Images**: Insert an image file into the button by clicking the **Images** button.

  When the *Images* dialog is displayed, specify the following parameters:

  - **File**: Type the file path to the image file. You can also click the browse button to the right of the box, to open a standard Windows file browser.

    > **Tip:** Icon files (`*.ICO`) cannot be kept in memory, which means they must be loaded from the hard drive every time the project screen is opened or redrawn, and that may decrease your run-time performance. If you use many icon files in your project, consider replacing them with new files in another format such as GIF, JPG, or PNG. For more information, see Configure the performance control settings on page 112.

  - By default, the image is displayed at its actual size. To change this, in the **Size** list, select **Custom**, and then configure the desired **Width** and **Height** (in pixels) of the image.
  - **Position**: Select where the image should be positioned in relation to the caption.
  - **Offset**: Specify the offset (in pixels).
  - **Transparent Color**: Select which color in the image should be transparent. The background color (see above) will show through these areas.
- **Advanced**: Specify advanced settings for the button by clicking the **Advanced** button.

  When the *Advanced* dialog is displayed, specify the following parameters:

  - **Enable translation** (*optional*): Specify an external translation file for the button label by clicking (checking) the box.
  - **Multiline**: Allow the caption of the button to be shown in more than one line, when checked.
  - **Wrap Text**: When checked, the object automatically wraps the text when necessary.
  - **Auto gray out**: Turns the caption of the button to gray when the Command animation applied to the button is disabled by the **Disable** field or due to the Security System.
  - **Auto Format**: When checked, if the caption includes a decimal value enclosed by curly brackets (e,g, `{1.2345}`) or a tag of Real type (see **Caption** above), then the value will be formatted according to the virtual table created by the **`SetDecimalPoints()`** function.
- **Command**: Click to automatically apply a Command animation to the button and then switch to the animation's properties.

## PUSHBUTTON OBJECT

On the **Graphics** tab, in the **Active Objects** group, click **Pushbutton** to create a Pushbutton object using the Command animation with an object or pre-configured pushbuttons.

EcoStruxure Machine SCADA Expert provides the following pre-configured button types, all of which mimic the standard panel buttons of the same name:

- **Momentary** (default): Changes state (**Open** or **Closed**) when you press the button and reverts to its initial state when you release the button. This button type always displays in its normal position when you open the screen.
- **Maintained**: Changes state (**Open** or **Closed**) when you press the button but does not revert to its initial state when you release the button. You must press the button again to change its present state. This button type maintains its state across screen changes.
- **Latched**: Changes state (**Open** or **Closed**) when you press the button and remains in this state until you release it by changing the Reset tag.

EcoStruxure Machine SCADA Expert also provides the following button styles:

- Rectangular with a faceplate and indicator light
- Rectangular without a faceplate or indicator light (default)

- Rectangular with a 3-D
- Rectangular with a floating appearance

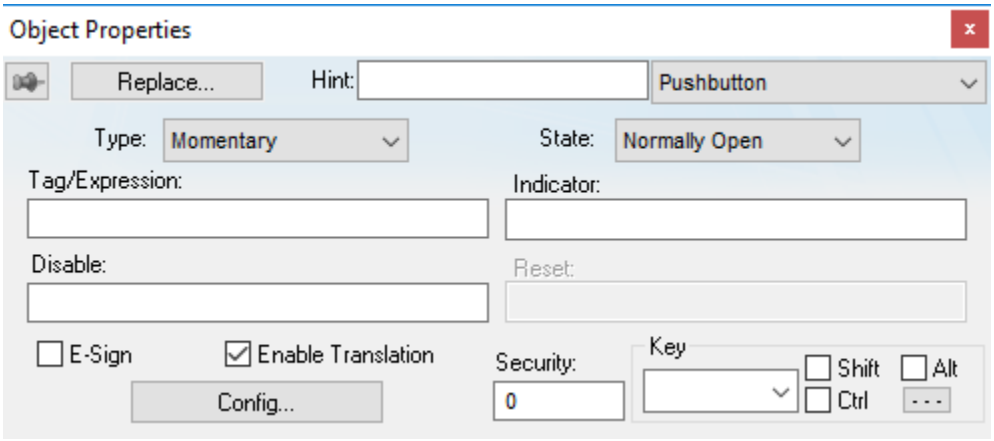
To add one or more pre-configured buttons to a screen:

1. Click the **Pushbutton** tool, and position the mouse (pointer) on the screen.
2. Click and drag to create/adjust the size of the rectangular button.

   The button size and text font characteristics determine how much text you can display and how much area you can touch on a touch screen. You can resize the button and change the font characteristics later to permit longer messages to be shown in a given space.
3. Double-click on the object to open the *Object Properties* dialog.

> **Tip:** Alternatively, you can right-click on the pushbutton object or highlight the object, press Alt +Enter, and select **Properties** from the resulting shortcut menu to open the *Object Properties* dialog.

### Object Properties: Pushbutton




***Object Properties: Pushbutton***

You can use this dialog to specify the following parameters:

- **Type** drop-down list: Click to select the pushbutton type (**Momentary** (default), **Maintained**, or **Latched**).
- **State** drop-down list: Click to specify a default state for the pushbutton (**Normally Open** (default) or **Normally Closed**).

  Click the button to toggle between its default and non-default state (according to its specified **Type**). For example, in the button's initial state, it may conform to characteristics specified in the **Open** area of the *Configuration* dialog (see below). Click the button again to toggle to the opposite state, which in this example is **Closed**, and conform to characteristics specified in the **Closed** area.
- **Tag/Exp** text box: Type a tag or an expression to accomplish the following:
  - Type in a tag to receive the **Write Value** from the appropriate state (**Open** or **Closed**) area in the *Configuration* dialog.
  - Type an expression to execute *On Down*, when you press the pushbutton down.

> **Note:** EcoStruxure Machine SCADA Expert *does not* write the result of any expression in the **Tag/Exp** field into a tag.

- **Indicator** text box: Type a tag to define an indicator that causes the button to change to a specified color when the tag value matches one of two specified values. You must define both the colors and tag values in the *Configuration* dialog. If you leave this field blank, the indicator changes color automatically when you press the button.

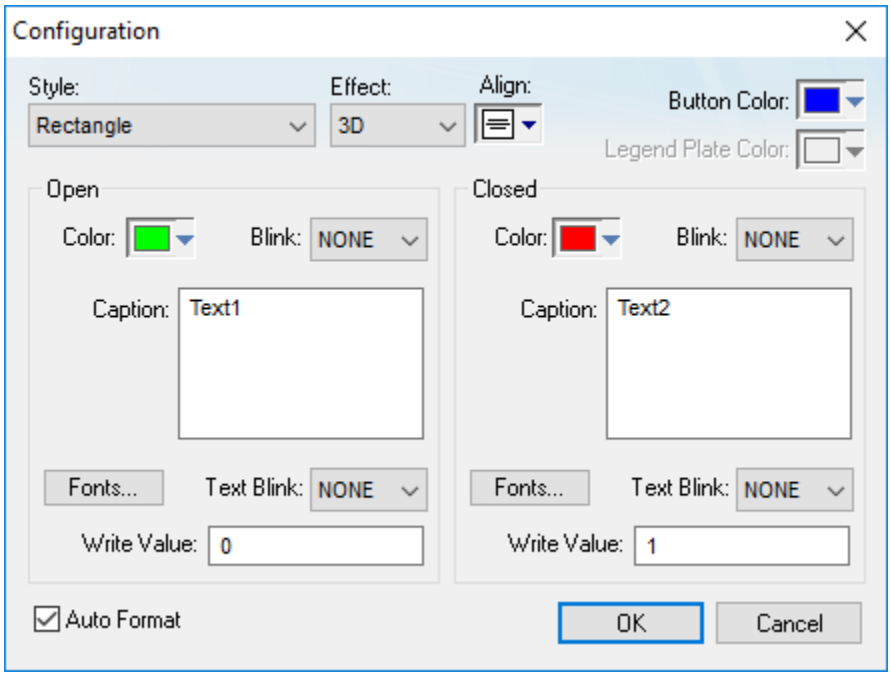
- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the animation.
- **Reset** text box (active for **Latched** pushbutton type only): Type a tag to control the button's latched state, as follows:
  - Type a zero and the button will remain in a latched state after you press it.
  - Type a nonzero value and a latched button will become unlatched after you press it. You must reset the tag value to zero before you can press the button again.
- **Key** area: Specify a keyboard key or create a key combination to toggle a pushbutton when you have no pointing device (mouse or touch screen) or if you want to create shortcut keys in addition to pushbuttons.
- **Key** drop-down list: Type a key in the text box or select a non-alphanumeric key from the drop-down list. Enter a single character or key only. Numbers are not valid entries for this field.

  Click (check) the **Shift**, **Ctrl**, or **Alt** box to create a combination key, meaning the Shift, Ctrl, or Alt key must be pressed with the key specified in the drop-down list.

  Click the browse button ... to open the *Key Modifier* dialog, which enables you to modify your combination keys. You can choose **Left**, **Right** or **Left or Right** to specify the position on the keyboard of the Shift, Ctrl or Alt key in the combination key. If you choose **Left or Right**, the command will be executed any time either of these keys is pressed in combination with the key specified in the drop-down list.

> **Tip:** If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects.

- **Disable** text box: Type a tag using a nonzero value to disable this pushbutton so that pressing the button has no effect. This box is empty by default, which also enables the Command animation.
- **Ext Trans.** checkbox: Click (check) to translate the text automatically using pre-configured translation worksheets. (See the Translation Tool for more information.)
- **Security** text box: Type a value to specify a security level (0 to 255) for this button. If the user does not have the specified security level, the button becomes inactive. If the user has the appropriate security level, or you leave this field blank, the button remains active.
- **Config** button: Click to open the *Configuration* dialog, which allows you to specify style and state parameters for the pushbutton:

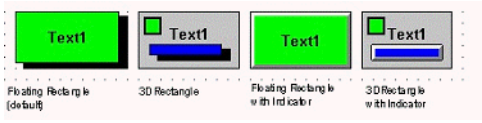



***Configuration dialog***

This dialog provides the following parameters:

- **Style** combo-box: Click the combo-box button to select a pushbutton style (**Rectangle** (default) or **Rectangle with Indicator**).
- **Effect** combo-box: Click to select a 3-D effect for the pushbutton.
  - **Floating** (default): Buttons resemble a flat object with a shadow
  - **3D**: Buttons have beveled edges and appear to "depress" into the screen when pressed.

  You can use the **Style** and **Effect** parameters in combination to create four different buttons, as shown in the following figures:




***Pushbutton Styles***

- **Align**: Specify the alignment for the caption of the pushbutton.
- **Button Color** box: Click to specify a default color for the button area of a pushbutton object that includes an indicator and a faceplate. When the *Color* dialog displays, click on a color to select it, and close the dialog.
- **Legend Plate Color** box: Click to specify or change a default color for the legend plate area of a pushbutton object that includes an indicator. When the *Color* dialog displays, click on a color to select it, and close the dialog.

  A legend plate encloses a button and indicator light. This field becomes inactive if the pushbutton Style does not include an indicator.
- **Open** and **Closed** areas: The following parameters are used to configure the appearance of a pushbutton object in its open and closed states.
  - **Color** box: Click to specify a default color for an indicator in each **State**. When the *Color* dialog displays, click on a color to select it, and close the dialog.

    If you selected a pushbutton style that does not include an indicator, you can use this field to specify a button color for each **State**.
  - **Blink** combo-box: Click to specify whether the color you specified in the **Color** box blinks and how fast it blinks for each state (**None** (no blinking, *default*), **Slow**, and **Fast**).

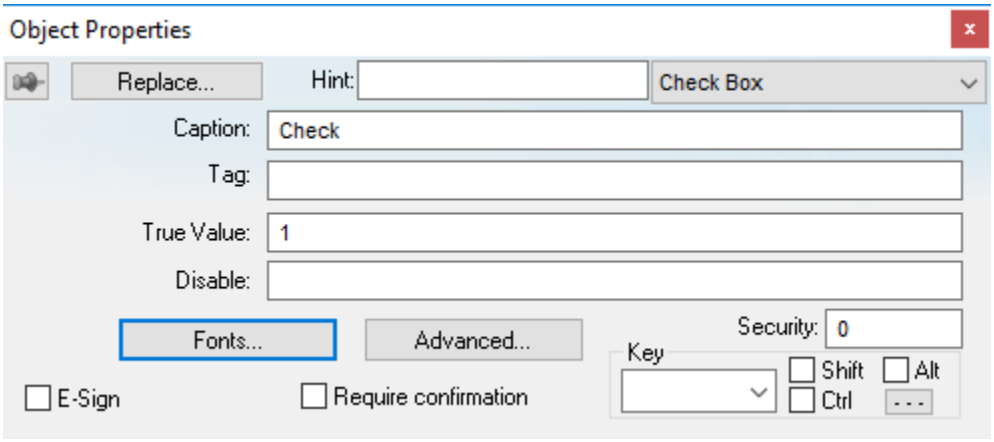
    If you set the color to blink, it alternates between the color specified in the **Color** box and the **Legend Plate Color** (if an indicator) or the **Button Color** (if a button).
  - **Caption** text box: Use this text box to enter the caption of the button. Alternatively, if the button style includes an indicator, the legend plate. You can include a tag by enclosing it in curly brackets (e.g., **`{tagname}`**).
  - **Fonts** button: Click to open the *Font* dialog, which you can use to specify or change the message font characteristics for each state.
  - **Text Blink** combo-box: Click to specify whether the text you specified blinks and how fast it blinks for each state (**None** (no blinking, *default*), **Slow**, and **Fast**). Unlike a blinking color, blinking text appears and disappears.
  - **Write Value** combo-box: Click to select a value in either field. When the pushbutton is in the appropriate state (**Open** or **Closed**), EcoStruxure Machine SCADA Expert writes this value to the tag specified in the **Tag/Exp** field (*Object Properties* dialog).
- **Auto Format**: When checked, if the caption includes a decimal value enclosed by curly brackets (e,g, **`{1.2345}`**) or a tag of Real type (see **Caption** above), then the value will be formatted according to the virtual table created by the **`SetDecimalPoints()`** function.

## CHECK BOX OBJECT

The Check Box object is useful to create interfaces where the users can enable/disable an option on the display.

On the **Graphics** tab, in the **Active Objects** group, click **Check Box** to create a Check Box object on your screen:

1. Click in the drawing area and drag the mouse/cursor to draw the check box and its label.
2. Release the mouse button when the object is the size you want.
3. Double-click on the object to view the *Object Properties* dialog.




***Object Properties: Check Box***

> **Tip:** To change the default size of the check box, edit your project file (`<project name>.app`) to add the following setting:
>
> ```
> [Objects]
> CheckBoxSize=height_in_pixels
> ```
>
> Doing this will change the size of all check boxes in your project.

Use the *Object Properties* dialog to specify the following parameters for the Check Box object:

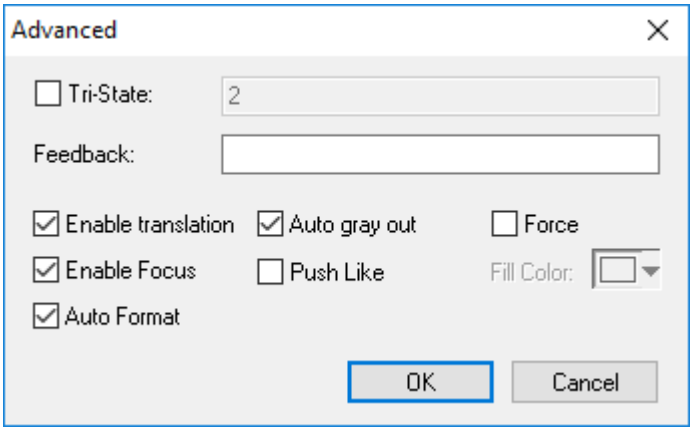
- **Caption**: Specify a caption by typing the text into the text box. You can include a tag by enclosing it in curly brackets (e.g., **{*tagname*}**).
- **Fonts**: Specify a font style for the caption by clicking the **Fonts** button.
- **E-Sign**: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the command.
- **Confirm** check box: Click (check) this box to ensure EcoStruxure Machine SCADA Expert prompts you to confirm the action at runtime.
- **Key** drop-down list: Select a key from the list to associate that keyboard key with the object or group of objects. You can then press this key to check/uncheck the check box.

  Click (check) the **Shift**, **Ctrl**, or **Alt** box to create a combination key, meaning the Shift, Ctrl, or Alt key must be pressed with the key specified in the drop-down list.

  Click the browse button ... to open the *Key Modifier* dialog, which enables you to modify your combination keys. You can choose **Left**, **Right** or **Left or Right** to specify the position on the keyboard of the Shift, Ctrl or Alt key in the combination key. If you choose **Left or Right**, the command will be executed any time either of these keys is pressed in combination with the key specified in the drop-down list.

  > **Tip:** If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects.

- **Disable** field: Type a tag or expression into this field to enable and disable the object. You disable the Check Box object when you enter a value different from 0.
- **Security** field: Type a value in this field to specify a security level for the object, as defined under Security. When a user logs on, and does not have the specified security level, EcoStruxure Machine SCADA Expert disables the object.
- **Tag** field: When the user clicks on the check box during runtime, the value of this tag is updated. If no **Feedback** was specified, the value of this tag is also used to indicate the current status of the object.
- **True Value** field: Specify a value that will be used to change the control to TRUE state and to indicate that the control is in TRUE state. For more information about states, please refer to the states table.
- **Advanced** button: Press this button to open the *Advanced* dialog:




***Advanced dialog***

- **Tri-State**: If enabled the control has a third state. The third state will be displayed when the tag configured in the **Feedback** field assumes the value specified in the **Tri-State** field. If the **Feedback** field is left in blank, the third state will be displayed when the tag configured in the **Tag** field assumes the value specified in the **Tri-State** field.

> **Note:** The **Tri-State** field must not be configured with the same value as the **True Value** field, nor with an empty string value.

- **Feedback**:Value that indicates the state of the object (FALSE, TRUE, or TRI-STATE). When the value of the tag configured in **Feedback** is equal to the value of the tag configured in **True Value**, the state is set to TRUE. When the value of the tag configured in **Feedback** is equal to the value of the tag configured in **Tri-State**, the state is set to TRI-STATE. When none of these conditions are satisfied, the state is set to FALSE. If the **Feedback** field is left in blank, then the tag configured in the **Tag** field will be used as the **Feedback** tag.
- **Ext Trans.**: When this option is checked, the caption of the object supports the translation.
- **Auto gray out**: Turns the caption of the object to gray when it is disabled by the **Disable** field or due to the Security System.
- **Force**: Click (check) this box to force the Tag Database to recognize a tag change when the user clicks on the object, even if the value of the tag in question does not change.
- **Enable Focus**: When this option is checked, the object can receive the focus during runtime by the navigation keys.
- **Push Like**: When this option is checked the control is displayed as a button, instead of the standard check box standard shape.
- **Fill Color**: Specify the fill color for the button. This option is enabled only when the **Push Like** option is checked.

- **Auto Format**: When checked, if the caption includes a decimal value enclosed by curly brackets (e,g, `{1.2345}`) or a tag of Real type (see **Caption** above), then the value will be formatted according to the virtual table created by the `SetDecimalPoints` function.

### Modes of Operation

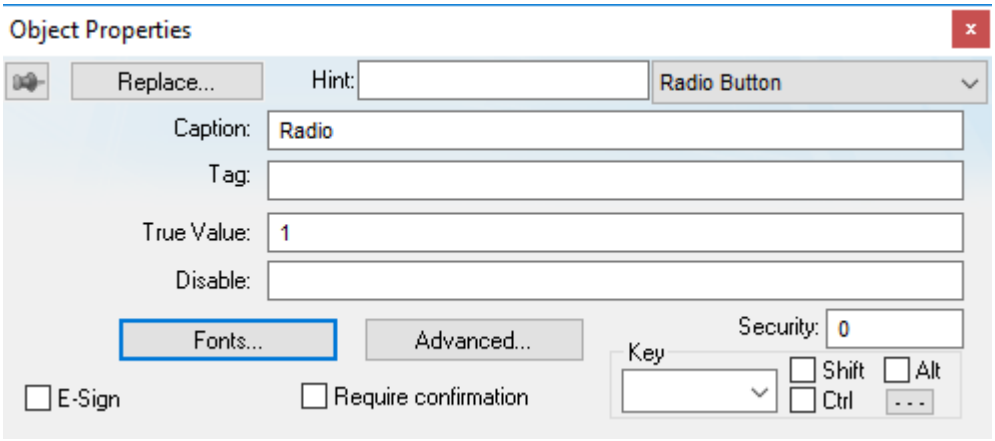
The Check Box object can operate in two different modes: Normal and Tri-State. For more information, see Modes of operation for Check Box and Radio Button objects.

## RADIO BUTTON OBJECT

The radio button object is useful to create interfaces where the users can chose one option from multiple options on the display.

On the **Graphics** tab, in the **Active Objects** group, click **Radio Button** to create a radio button object on your screen:

1. Click in the drawing area and drag the mouse/cursor to draw the radio button and its label.
2. Release the mouse button when the object is the size you want.
3. Double-click on the object to view the *Object Properties* dialog.




***Object Properties: Radio Button***

> **Tip:** To change the default size of the radio button, edit your project file (`<project name>.app`) to add the following setting:
>
> ```
> [Objects]
> RadioButtonSize=height_in_pixels
> ```
>
> Doing this will change the size of all radio buttons in your project.

Use the *Object Properties* dialog to specify the following parameters for the radio button object:

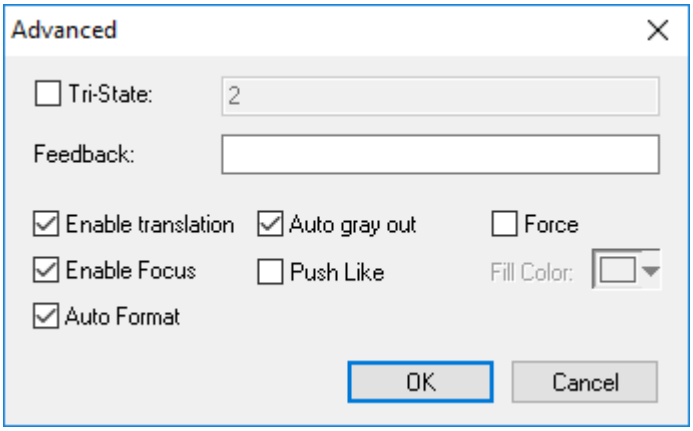
- **Caption**: Specify a caption by typing the text into the text box. You can include a tag by enclosing it in curly brackets (e.g., `{tagname}`).
- **Fonts**: Specify a font style for the caption by clicking the Fonts button.
- **E-Sign**: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the command.
- **Confirm** checkbox: Click (check) this box to ensure EcoStruxure Machine SCADA Expert prompts you to confirm the action at runtime.
- **Key** drop-down list: Select a key from the list to associate that keyboard key with the object or group of objects. You can then press this key to check/uncheck the radio button.

  Click (check) the **Shift**, **Ctrl**, and/or **Alt** boxes to create a combination key, meaning the Shift, Ctrl, and/or Alt key must be pressed with the key specified in the drop-down list.

Click the browse button ... to open the *Key Modifier* dialog, which enables you to further modify your combination keys. You can choose **Left**, **Right** or **Left or Right** to specify the position on the keyboard of the Shift, Ctrl, or Alt key in the combination key. If you choose **Left or Right**, the command will be executed any time either of these keys is pressed in combination with the key specified in the drop-down list.

> **Tip:** If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects.

- **Disable** field: Type a tag or expression into this field to enable and disable the object. You disable the radio button object when you enter a value different from 0.
- **Security** field: Type a value in this field to specify a security level for the object, as defined under Security. When a user logs on, and does not have the specified security level, EcoStruxure Machine SCADA Expert disables the object.
- **Tag** field: When the user clicks on the radio button during runtime, the value of this tag is updated. If no **Feedback** was specified, the value of this tag is also used to indicate the current status of the object.
- **True Value**: Specify a value that will be used to change the control to TRUE state and to indicate that the control is in TRUE state. For more information about states, please refer to the states table.
- **Advanced**: Press this button to open the *Advanced* dialog:




***Advanced dialog***

- **Tri-State**: If enabled the control has a third state. The third state will be displayed when the tag configured in the **Feedback** field assumes the value specified in the **Tri-State** field. If the **Feedback** field is left in blank, the third state will be displayed when the tag configured in the **Tag** field assumes the value specified in the **Tri-State** field.

  > **Note:** The **Tri-State** field must not be configured with the same value as the **True Value** field, nor with an empty string value.

- **Feedback**: Value that indicates the state of the object (FALSE, TRUE, or TRI-STATE). When the value of the tag configured in **Feedback** is equal to the value of the tag configured in **True Value**, the state is set to TRUE. When the value of the tag configured in **Feedback** is equal to the value of the tag configured in **Tri-State**, the state is set to TRI-STATE. When none of these conditions are satisfied, the state is set to FALSE. If the **Feedback** field is left in blank, then the tag configured in the **Tag** field will be used as the **Feedback** tag.
- **Ext Trans.**: When this option is checked, the caption of the object supports the translation.
- **Auto gray out**: Turns the caption of the object to gray when it is disabled by the **Disable** field or due to the Security System.
- **Force**: Click (check) this box to force the Tag Database to recognize a tag change when the user clicks on the object, even if the value of the tag in question does not change.

- **Enable Focus**: When this option is checked, the object can receive the focus during runtime by the navigation keys.
- **Push Like**: When this option is checked the control is displayed as a button, instead of the standard radio button standard shape.
- **Fill Color**: Specify the fill color for the button. This option is enabled only when the **Push Like** option is checked.
- **Auto Format**: When checked, if the caption includes a decimal value enclosed by curly brackets (e,g, `{1.2345}`) or a tag of Real type (see **Caption** above), then the value will be formatted according to the virtual table created by the `SetDecimalPoints()` function.

### Modes of Operation

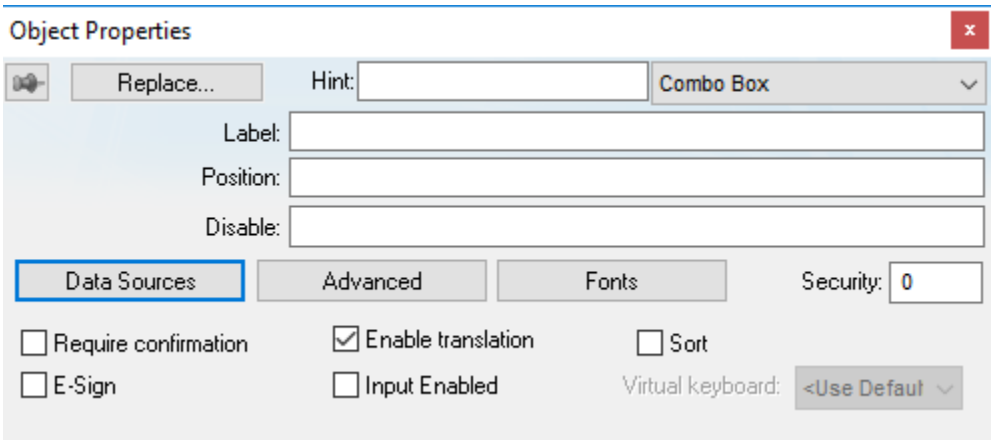
The Radio Button object can operate in two different modes: Normal and Tri-State. For more information, see Modes of operation for Check Box and Radio Button objects.

## COMBO BOX OBJECT

On the **Graphics** tab, in the **Active Objects** group, click **Combo Box** to select a single label from a combo-box list of labels.

If the list is longer that the space allotted, a scroll bar is enabled for the list. During runtime, if you select a label from the list, the combo-box hides itself and the selected label displays in the combo box.

Double-click on the combo-box object to open the *Object Properties* dialog box.

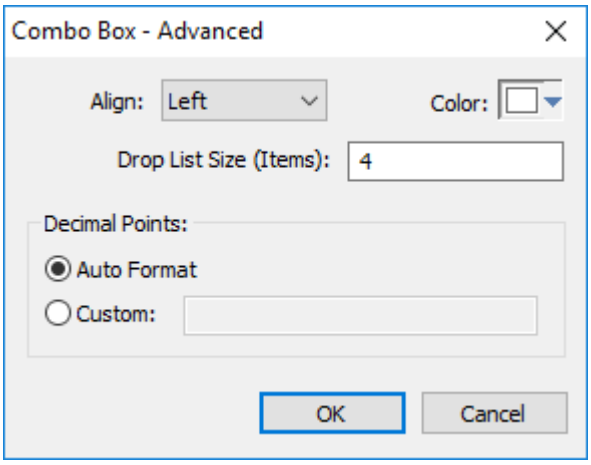



***Object Properties: Combo Box***

You can use this dialog box to set the following parameters:

- **Label** text box: Type a string tag to receive the value of the label currently displayed in the combo box.
- **Position** text box: Type an integer tag, which corresponds to the label currently displayed in the combo box. Changing this tag value changes the label being displayed.
- **Disable** text box: Type a tag with a nonzero value to disable this combo box. Type a zero, or leave the field blank (default) to enable the Command animation. If you disable the combo box, it appears grayed out during runtime.
- **Data Sources** button: Click to open the *Data Sources* dialog box (see below).

- **Advanced** button: Click to open the *Combo Box - Advanced* dialog box:




***Combo Box - Advanced***

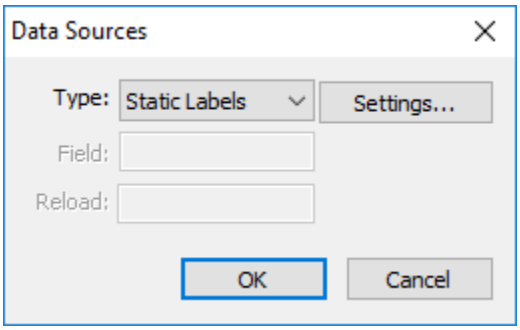
  - **Align** combo-box: Click to specify the label alignment (**Left**, **Center**, or **Right**) which affects the alignment in both the combo box and its list.
  - **Color** box: Click to specify a background color for the combo box. When the *Color* dialog box opens, click a color to select it, then click **OK** to close the dialog box.
  - **Drop List Size (Items)** field: Specify the number of items that should be displayed at one time when the user clicks on the combo box. The higher the number of items, the longer the drop list will appear.

    > **Note:** If this number is less than the *total* number of items in the list, then the drop list will also scroll.

  - **Decimal Points**: Select how decimal values will be displayed on-screen:
    - **Auto Format**: Decimal values will be formatted according to the virtual table created by the `SetDecimalPoints` function.
    - **Custom**: Enter the number of decimal places to display (e.g., 2) for all decimal values.
- **Fonts** button: Click to open a standard *Font* dialog box. Use this dialog box to change the characteristics of a message font.
- **Security** text box: Type a security level for the command (0 to 255). If an operator logs on and does not have the specified security level, the command becomes inactive. If an operator logs on and does have the specified security level, or you leave this field blank, the Command animation remains active.
- **Require confirmation** checkbox: Click (check) to prompt an operator to confirm a command during runtime.
- **Enable translation** checkbox: Click (check) to enable automatic translation of the combo box labels using the Translation Table.
- **Sort** checkbox: Click (check) to display the contents of your array of labels in alphabetical order. This parameter is available only when you select the **Array Tag** type.
- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the animation.
- **Input Enabled** checkbox: Click (check) to allow an operator to select a label by typing the contents of that label into a tag in the **Label** field.
- **Virtual keyboard**: Virtual Keyboard type used for this object. You need to select the Virtual Keyboard option in the *Viewer* settings (**Viewer** on the Project tab of the ribbon) before configuring the Virtual Keyboard for this interface.

### Data Sources

Use the *Data Sources* dialog box to configure the items/labels that will be displayed in the Combo Box object.

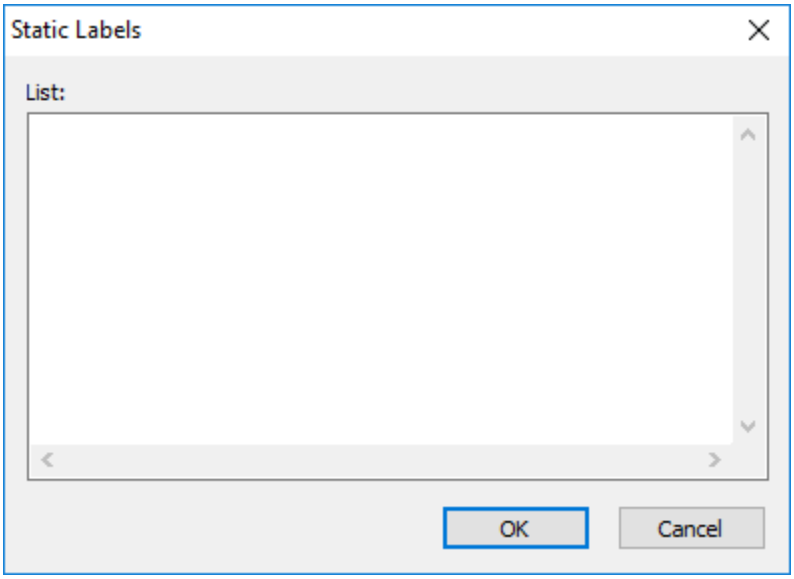



***Data Sources dialog box***

- **Type** combo box: Select the type of data source that you want to use, and the click **Settings** to configure the source. Each type of source is described in detail below.
- **Field** field (for Text File and Database only): Specify which field/column of the data source to read from.
- **Reload** field (for Text File and Database only): Enter a tag name. When the value of the specified tag changes, the combo box will reload the labels from the data source.

### Data Source Type: Static Labels

When **Type** is set to **Static Labels**, you can configure the following settings:




***Static Labels dialog box***

Enter your labels — with one label per line — just as if you were editing a plain text file.

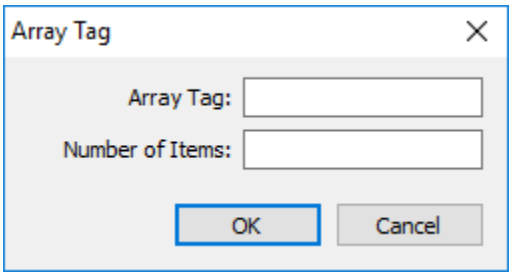
The labels are not sorted in any way, so be sure to put them in the order you want them displayed during runtime. The first line is position 0, the second line is position 1, and so on.

Click **OK** when you are done.

### Data Source Type: Array Tag

> **Note:** This type is not supported in project screens that are viewed through Mobile Access.

When the **Type** is set to **Array Tag**, you can configure the following settings:




***Array Tag dialog box***

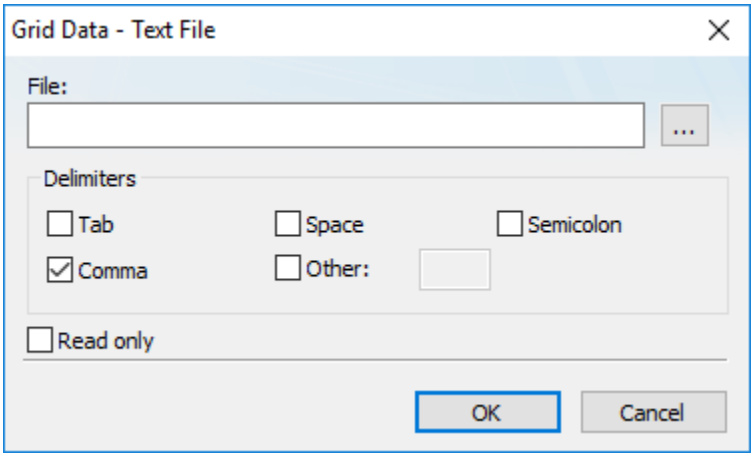
- **Array Tag**: Enter the name of a String array that contains the items for the combo box.
- **Number of Items**: Specify how much of the array should be displayed in the combo box. Keeping in mind that the combo box counts *array index 0* as the first item, if you enter a value of 4, then the combo box will display *array index 0* through *array index 3*.

Click **OK** when you are done.

### Data Source Type: Text File

> **Note:** This type is not supported in project screens that are viewed through Mobile Access.

When the **Type** is set to **Text File**, you can configure the following settings:




***Grid Data – Text File dialog box***

- **File**: Enter the name of the text file source. You can either type the file name and its path or click the ... button to browse for it. (If the file is stored in your project folder, then you can omit the path in the name.)

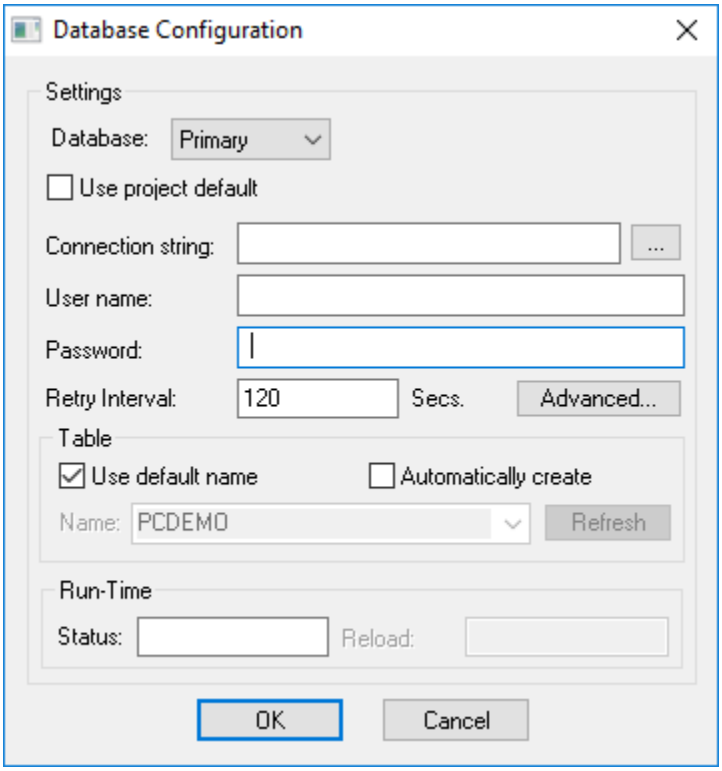
  > **Tip:** You can configure tag names between curly brackets (e.g., `{tagname}`) in the **File** field.

- **Delimiters**: Set the delimiter(s) used in the data source file. For instance, if the data will be read from a CSV (comma separated values) file, you would select the **Comma** option. You can even choose a custom delimiter by checking the **Other** option and typing the custom delimiter in the field beside it.

Click **OK** when you are done.

### Data Source Type: Database

When the **Type** is set to **Database**, you can configure the following settings:




***Database Configuration dialog box***

For more information, see Database Configuration on page 108.

## LIST BOX OBJECT

The List Box screen object displays a list of messages or menu items for the user to select from. When the user selects a message, its corresponding numerical value is written to a project tag.

If the list of messages is too long to fit within the viewable area of the List Box object, the object provides scroll bars.

The user can browse the list and make a selection by doing one of the following, depending on how you design your project interface:

- Use mouse or touchscreen input to click/tap the list's scroll bar and then select a message;
- Press the **Up**, **Down**, **Esc**, **Tab** and/or **Enter** keys on the keyboard or keypad; or
- Use on-screen controls (e.g., Button objects, linked symbols) that have been configured with the `PostKey` function to post the equivalent key codes.

Generally, when you run a project, the *active* List Box object displays a list of messages. On a screen containing only one List Box object and no text input boxes, the List Box object will be active automatically. On a screen containing multiple List Box objects and text input boxes, you can use a cursor (pointing device) or the Tab key to select and activate a List Box object.

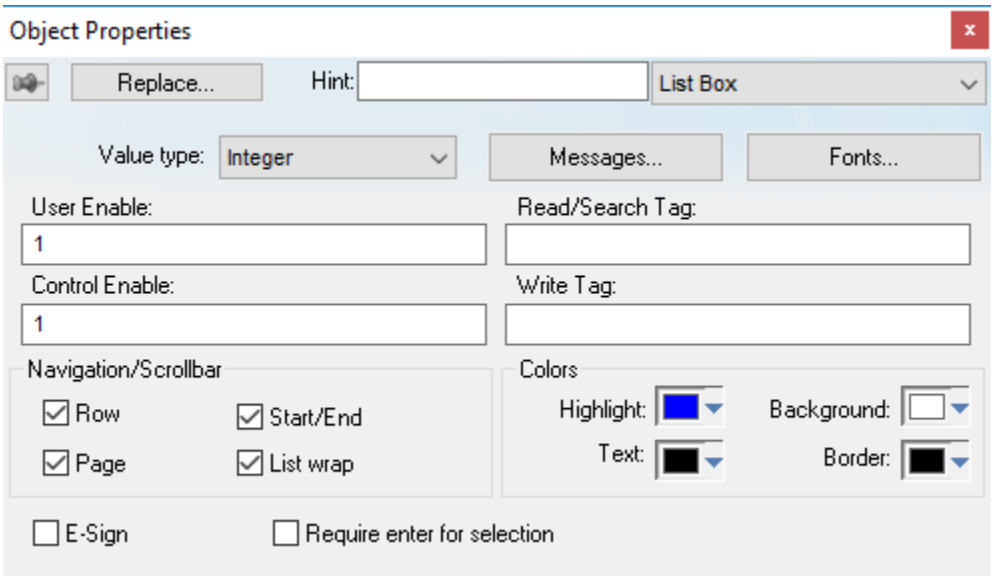
To add a List Box object to a screen:

1. On the **Draw** tab of the ribbon, in the **Active Objects** group, click **List Box**.
2. Draw the List Box object in the screen, and then drag object's handles to adjust its size.

The height of the object and the font size determine how many messages are visible. The width of the object determines how much of the message length is visible.

After you draw the object, you can adjust the size and font characteristics to allow more messages to display in the given space.

3. Double-click on the object to open the *Object Properties* dialog.




***Object Properties: List Box***

You can use this dialog to specify the following parameters:

- **Value** drop-down list (located below the **Replace** button): Click to select one of the following the tag values used to index the message list.
  - **Boolean**
  - **Integer** (default)
  - **LSB** (Least Significant Bit)

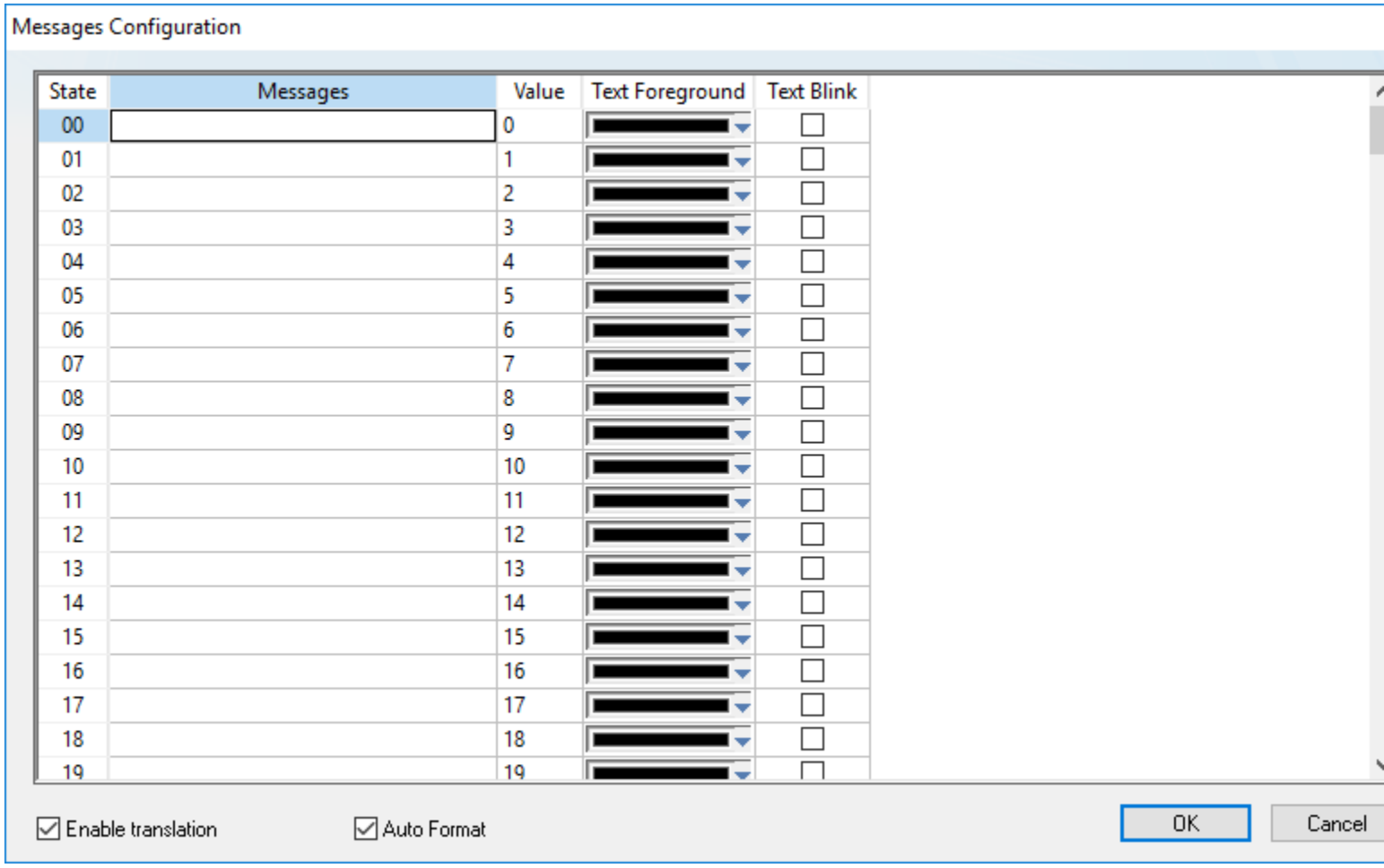
  > **Note:** For more information, see the discussion about the **State** field on the *Messages Configuration* dialog.

- **Messages** button: Click to open the *Messages Configuration* dialog (see below).
- **User Enable** text box: Type a tag, expression, or a (nonzero) number to select a message in the runtime project. The default is `1` (*true* or *enabled*).
- **Control Enable** text box: Type a tag, expression, or a (*nonzero*) number to select a message in the runtime project — depending on the current value of the **Read/Search Tag**. The default is `1` (*true* or *enabled*).

  EcoStruxure Machine SCADA Expert bases this parameter on the **Value** field (in the *Messages Configuration* dialog) that you associate with the selected message. Enabling this field allows tag changes triggered by the process to affect which messages you can select.
- **Read/Search Tag** text box: Type an integer or a Boolean tag to point to a selected message based on the message **Value** field (in the *Messages Configuration* dialog). You can use the **Control Enable** and **User Enable** fields to control whether the operator or a process can alter this tag.
- **Write Tag** text box (*optional*): Type a string tag to receive the **Message** value of the last-selected message. When you close and reopen the screen containing a List Box object, EcoStruxure Machine SCADA Expert uses this tag value to determine the last message selected in the list box.

- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature when using this object.
- **Row** checkbox: Click (check) to include set up and set down arrows in the List Box object scroll bar.
- **Page** checkbox: Click (check) to include page up and page down arrows in the List Box object scroll bar.
- **Start/End** checkbox: Click (check) to include home and end arrows in the List Box object scroll bar.
- **List wrap** checkbox: Click (check) to continue displaying and scrolling the message list (starting at the opposite end) after you scroll to the beginning or end of the list.
- **Require enter for selection** checkbox: Clicking (checking) this box requires the user to press **Enter** (or post the equivalent key code) to make a selection. If this option is not checked, then a selection is made whenever focus changes to another screen object (i.e., the user tabs out of the list or clicks/taps on another object).
- **Color** boxes: Click a color box to open the *Color* dialog or the 16-color *Color Selection* dialog. Either dialog allows you to specify or change colors for the List Box object. Click a color to select it and then click **OK** to close the dialog.
  - **Highlight Color** box: Specify a color for highlighting messages (default is blue).
  - **Text Color** box: Specify a color for highlighting message text (default is black).
  - **Win Color** box: Specify a color for the list box background (default is white).
  - **Border Color** box: Specify a color for the list box border (default is black).

### Messages Configuration Dialog




***Message Configuration dialog***

Use the parameters on this dialog as follows:

- **State** field (*read-only*): Use this field to view the indexed individual messages. EcoStruxure Machine SCADA Expert numbers this field based on the **Read/Search Tag** type you selected:
  - **Boolean**: Provides two valid states, labeled 0 and 1
  - **Integer**: Provides 256 valid states, labeled 0 to 255
  - **LSB**: Provides 32 valid states (i.e., the 32 bits in an integer value), labeled 0 to 31
- **Message** field: Enter the string to be displayed in the List Box object. You can include tags in a message by enclosing them in curly brackets (e.g., `{tagname}`).
- **Value** field: Type a message value matching the specified **Read/Search Tag** value. (Also, the same value written to the write tag.)

  If you specify **LSB** for the **Value** field, EcoStruxure Machine SCADA Expert uses the value specified in the **State** field for both the **Read/Search Tag** and the write tag.
- **Text Foreground** color field: Click to specify a color for the message text foreground. The color is displayed only when the message is not selected.
- **Text Blink** checkbox: Click (check) to cause a message to blink, once per second, when it is selected.
- **Fonts** button: Click to open the *Font* dialog, which allows you to change the characteristics (style, size, and so forth) of the message font.
- **Enable translation**: Click (check) to enable translation during runtime using the Translation Tool.
- **Auto Format**: When checked, if a message includes a decimal value enclosed by curly brackets (e,g, `{1.2345}`) or a project tag of Real type (see **Message** above), then the value will be formatted according to the virtual table created by the `SetDecimalPoints` function.

## SMART MESSAGE OBJECT

The Smart Message screen object displays messages and images that can be changed during run time by updating the value of the associated project tag.

The following smart message object types are available:

- **Message Display**: Enables you to display any one of multiple messages within a single screen object.
- **Multistate Indicator**: Enables you to display any one of multiple messages within a single screen object, and also has the ability to display bitmap images with the messages.
- **Multistate Pushbutton**: Enables you to display messages and bitmap images. This object also resembles a multi-position switch in that it allows you to increment through the messages by clicking on the object during run time.

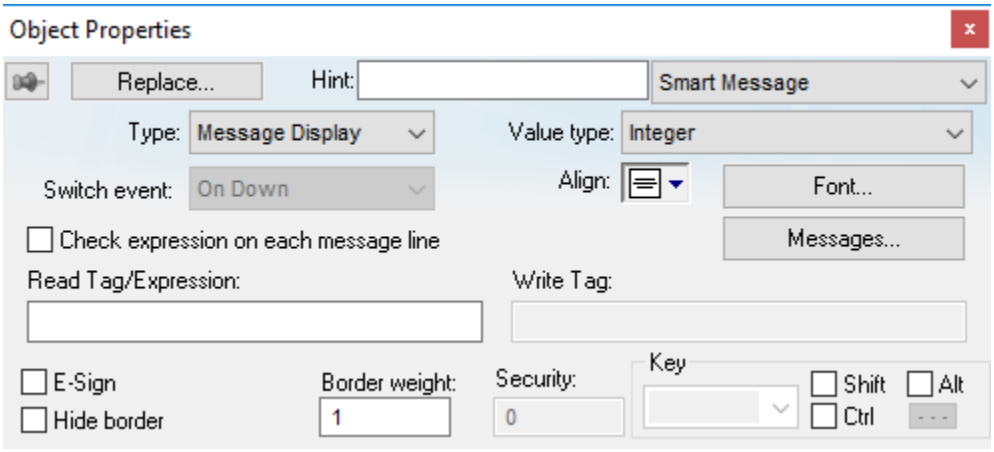
These smart message object types vary in their ability to display messages and graphics, write to a tag, and control how many messages and graphics display on the screen. However, all of the object types can receive process input (Read Tag value) to determine which message to display.

To add a smart message object to the screen:

1. On the **Draw** tab of the ribbon, in the **Active Objects** group, click **Smart Message**.
2. Draw the Smart Message object in the screen, and then drag object's handles to adjust its size.

   You use the object's height, width, and font size to determine how much text and how large a bitmap image you can display on the screen.

3. Double-click on the object to open the *Object Properties* dialog.




***Object Properties: Smart Message***

You can use this dialog to specify the following parameters:

- **Type** combo-box: Click to select the smart message object type. The object type sets the behavior of the object during run time and the features supported by it:
  - **Message Display** (default)
  - **Multistate Indicator**
  - **Multistate Pushbutton**
- **Value type** drop-down list: Select the type of values used to index the message list:
  - **Boolean**: Provides two valid states. Use this selection when you want to display either one of two different messages, based on a boolean value (0 or 1).
  - **Integer** (default): Provides 500 valid states. Use this selection when you want to display different messages based on specific values from an Integer tag.
  - **LSB** (least significant bit): Provides 32 valid states (i.e., 32 bits in an integer value). Use this selection when you want to display different messages based on which bit from an integer tag is set. If more than one bit from the Integer tag is set simultaneously, the message associated with the least significant bit that is set (value 1) will be displayed.

> **Note:** If **Multistate Pushbutton** is the **Smart Message** type, only 16 different messages can be associated with the object, even for **Integer** or **LSB** values.

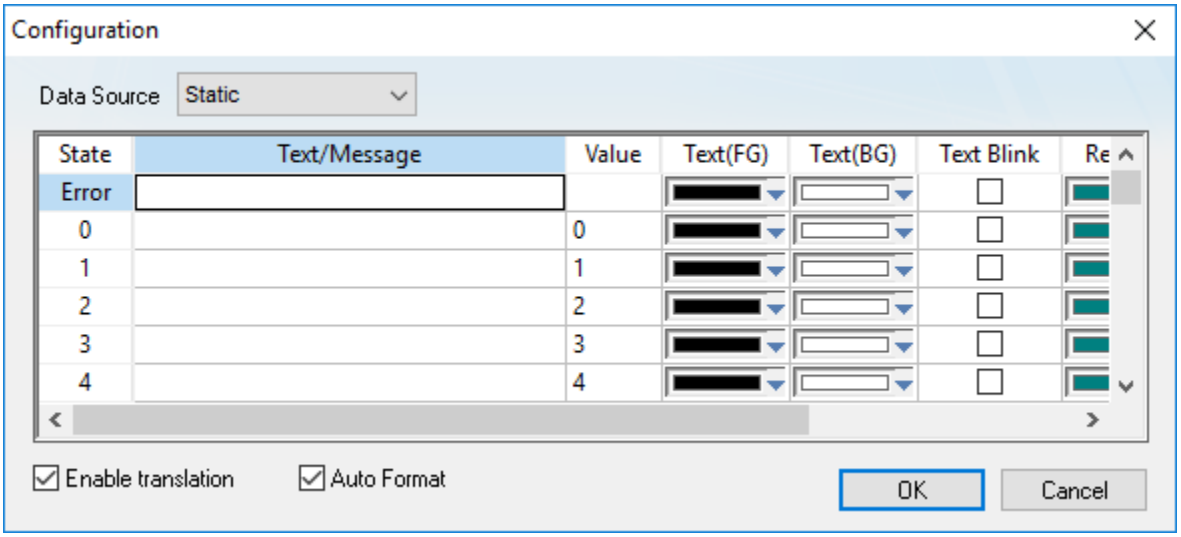
- **Switch event** drop-down list (available for **Multistate Pushbutton** only): Select one of the following options to specify when the message is changed:
  - **On Down**: Switch to the next message when you click on the object (default).
  - **While Down**: Switch to the next message continuously while you hold the mouse button down on the object.
  - **On Up**: Switch to the next message when you release the mouse button on the object.
- **Align**: Select the alignment of the text displayed by the Smart Message object.
- **Fonts**: Launches the Fonts dialog, where you can configure the font settings for the text displayed in the object.
- **Check expression on each message line** check box: When this option is cleared, only **Read Tag/Expression** is evaluated during run time and the resulting value determines which message is displayed. When this option is selected, each message has its own tag/expression
- **Read Tag/Expression** text box: Enter a project tag or expression. The value determines which message is displayed by the object during run time.

- **Messages**: Displays the *Configuration* dialog box, where you can configure the messages for the object. See "Messages Configuration" below.
- **Write Tag** text box (optional and available for **Multistate Pushbutton** only): Enter the name of a project tag. The value associated with the message currently displayed by the object is written to this tag.
- **E-Sign** (available for **Multistate Pushbutton** only): When this option is selected, the user will be prompted to enter the Electronic Signature before executing the animation.
- **Hide border**: When this option is selected, the line border of the object is not visible.
- **Border weight**: Defines the thickness (in pixels) of the line drawn around the object.
- **Security** (available for **Multistate Pushbutton** only): System Access Level required for the object/animation.
- **Key** area (optional and available for **Multistate Pushbutton** only): Shortcut used to go to the next message (step) using a keyboard when the Multistate Pushbutton type is selected. This option is especially useful when creating projects for runtime devices that do not provide a mouse or touch-screen interface, when the keyboard is the only physical interface available to interact with your project during run time.

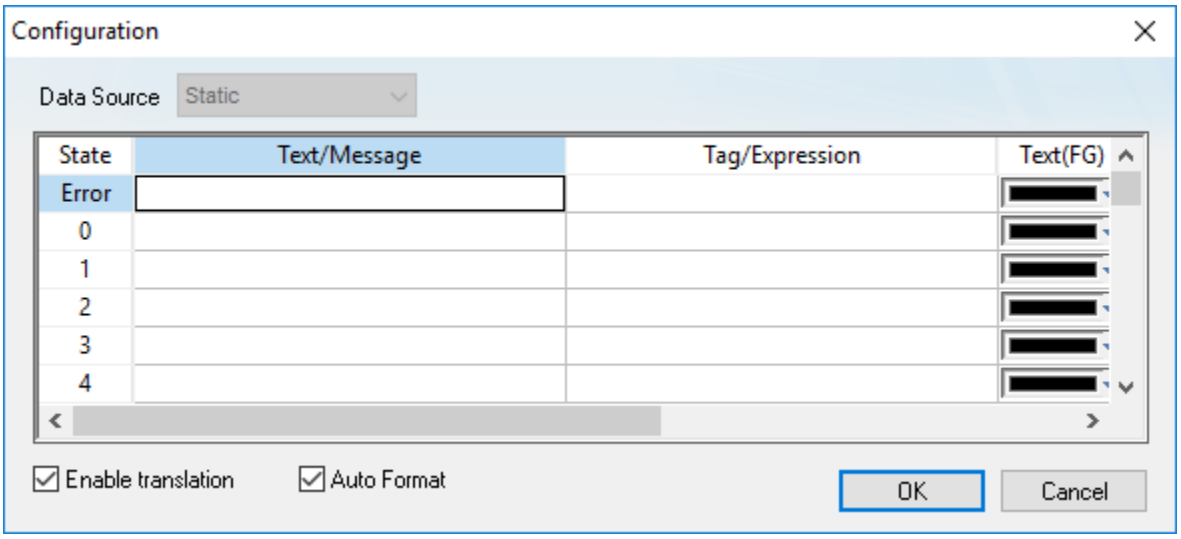
> **Tip:** If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects.

### Messages Configuration

The behavior of the object, and thus the layout of this dialog box, depend on whether **Check expression on each message line** is selected.




***Configuring messages for the resulting values of Read Tag/Expression***




***Configuring a different tag/expression for each message line***

**Data Source** list box: The source of the messages to be displayed by the object. Select one of the following:

- **Static**: The messages are configured and stored directly on the object. Configure a worksheet row for each possible state. For more information, see the table "Properties for Data Source: Static" below.
- **Text File**: The messages are loaded from an external source file. In the **Source File** box, type the path (relative to your project folder) and name of the file.

  > **Tip:** If you want to change the file path during run time, type the name of a string tag in curly brackets (e.g., `{MyTag}`) and then store the file path in that tag.

  For more information about how to create and format the file, see "Source File Format" below.

The following table describes the meaning of the properties associated with each message, regardless of the Data Source:

### Properties for Data Source: Static

| Property | Description |
|---|---|
| **Text/Message** | Message (text) that will be displayed when selected during run time. You can include tags in a message by enclosing them in curly brackets (e.g., {*tagname*}). |
| **Value** (when **Check expression on each message line** is cleared) | The unique value associated with each **Text/Message**.<br>**Read Tag/Expression** (in the Object Properties) is continuously evaluated during run time, and the resulting value determines which message is displayed on the object. If the resulting value is not in this list, the message configured in the first row (**Error**) is displayed.<br>When **Type** (in the Object Properties) is **Multistate Pushbutton**, the resulting value is also written to the tag configured in **Write Tag**. |
| **Tag/Expression** (when **Check expression on each message line** is selected) | The tag/expression associated with each **Text/Message**.<br>All of these are continuously evaluated during run time, and the first one in the list that evaluates as TRUE (non-zero) determines which message is displayed on the object. If none of these evaluate as TRUE (non-zero), the message configured in the first row (**Error**) is displayed. |
| **Text (FG)** | Foreground color for the messages displayed during run time. |
| **Text (BG)** | Background color for the messages displayed during run time. |
| **Text Blink** | If selected, the message text will blink during run time. |
| **Rec (FG)** | Line color (Border) for the rectangle behind the message. |
| **Rec (BG)** | Background (Fill) color for the rectangle behind the message. |
| **Rec Blink** | If selected, the rectangle behind the message will blink during run time. |
| **Graphic File** | Path and name of the bitmap file (*.BMP) (if any) that will be displayed when the message associated with it is selected during run time. If you do not specify the path, the bitmap file must be stored in your project folder. |
| **Transparent** | Select the color that will be transparent in the graphic file, if the **En. Transparent** checkbox is selected. |
| **En. Transparent** | If selected, the color selected in the **Transparent** field will be set as transparent in the graphic file. |

**Enable translation**: Click (check) to enable translation during run time using the Translation Tool.

**Auto Format**: When selected, if a message includes a decimal value enclosed by curly brackets (e,g, **{1.2345}**) or a project tag of Real type (see **Text/Message** below), then the value will be formatted according to the virtual table created by the function `SetDecimalPoints`.

> **Note:** The properties **Graphic File**, **Transparent** and **En. Transparent** are not available for the **Message Display** type.

> **Tip:** You can copy data from this dialog and paste it into an Excel worksheet, and vice versa.

## Source File Format

This section describes the format of the text file supported by the *Smart Message* object when the **Data Source** is **Text File**. The main advantage of using an external text file instead of static values is that it gives you the flexibility to change the messages during run time, by pointing to a different text file, or even by changing the content of the text file dynamically.

The text file must be created in the CSV format (comma separated values), where the comma character ( , ) is used to divide the columns (data) in each line (row) of the file. Therefore, you can use any CSV editor such as Microsoft Notepad and Microsoft Excel to create the CSV file with the messages and their properties for the *Smart Message* object.

The description of each property associated with the messages is provided in the Smart Message section. The order of the data in the CSV file is described in the following table:

| Column # | Property | Default Value |
|---|---|---|
| 1 | Text/Message | - |
| 2 | Value | - |
| 3 | Text (FG) | 0 |
| 4 | Text (BG) | 16777215 |
| 5 | Text Blink | 0 |
| 6 | Rec (FG) | 8421376 |
| 7 | Rec (BG) | 16777215 |
| 8 | Rec Blink | 0 |
| 9 | Graphic File | - |
| 10 | Transparent | 0 |
| 11 | En. Transparent | 0 |

When configuring text messages that have the comma character as part of the message, you must configure the whole message between quotes (e.g., **`"Warning, Turn the motor Off"`** ); otherwise, the comma will be interpreted as a data separator instead of as part of the message.

- The first line of this file is equivalent to the **State** = **Error**. In other words, if there is no message associated with the current value of the tag configured in the **Read Tag** field, the message configured in the first row (**State** = **Error**) is displayed during run time.
- The data configured in the **Value** column of the first row from this file is irrelevant. This row must always be configured, regardless of the object type (even for **Multistate Pushbutton**).
- Only the **Text/Message** and **Value** columns are mandatory. The other columns are optional, and the default values will be used if you do not specify any value for them (see table).
- The fields **Text(FG)**, **Text(BG)**, **Rec(FG)**, **Rec(BG)** and **Transparent** can be configured with the code of the color associated with it. The code can be entered directly in decimal format (e.g., 255) or in hexadecimal format using the syntax #value (e.g., #0000FF).
- The fields **Text Blink**, **Rec Blink** and **En. Transparent** can be configured with Boolean values **`0`** or **`1`** (**0** = **Unselected**; **1** = **selected**), or with the keywords **FALSE** or **TRUE** (**FALSE** = **Unselected**; **TRUE** = **selected**).

Example:

```
Error Message,,0,16777215,1,8421376,16777215,1,error.bmp,0,0
Message Zero,0,0,16777215,0,8421376,16777215,0,open.bmp,65280,1
Message Ten,10,0,16777215,0,8421376,16777215,0,closed.bmp,65280,1
Message Twenty,20,0,16777215,0,8421376,16777215,0,,0,0
Message Thirty,30,0,16777215,0,8421376,16777215,0,,0,0
```

> **Tip:** You can use the *Smart Message* editor (**Data Source** is **Static**) to configure the messages, values and colors. To do so, select the configuration, copy it and paste it into an Excel worksheet. Then, you can save the Excel worksheet as a CSV file (**File** > **Save As**). This procedure provides you with a user friendly interface for configuring the color codes.

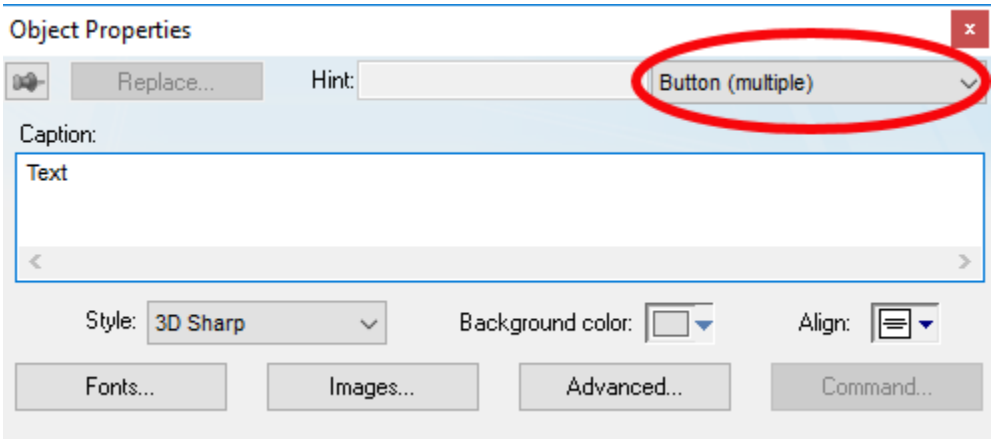
## CHANGE THE PROPERTIES OF MULTIPLE SCREEN OBJECTS

This task describes how to select two or more screen objects and then change the properties that are common to the selected objects.

Before you begin this task, you must have a project screen open in the screen editor.

Which properties you can change depends on whether you select multiple objects of the same type or of different types. If the objects are of the same type, you can change the properties that are specific to that type.

For example, if you select multiple Button objects, then you can change the properties that are specific to Button objects.

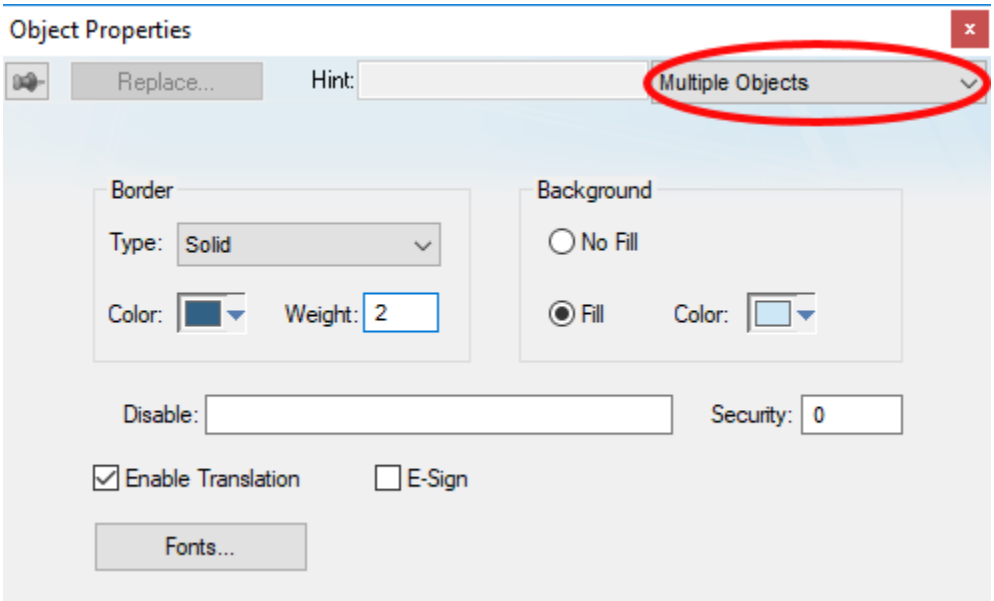



***Object properties of multiple selected Button objects***

For more information about the properties of a specific type of object, see the documentation for that object.

> **Note:** You can only use this method to change the properties of Shapes and Active Objects. You cannot use this method to change the properties of Data Objects, Animations, Library items, or objects in a group.

In contrast, if you select multiple objects of different types, you can change the properties that are common to all of the objects. This includes not only cosmetic properties like Border and Background, but also functional properties like Disable, Security, Enable Translation, and E-Sign. (Some properties may not apply to all objects. For example, Button objects do not have Border and Rectangle objects do not have Security.)




***Object properties for multiple objects of different types***

In both cases, the dialog box shows the current values of the properties of the last selected object.

It is only when you actually change the value of a property that the change is applied to the selected objects. All other properties are left unchanged, regardless the values shown in the dialog box.

To change the properties of multiple screen objects:

1. In the screen editor, do one of the following:
   - Press and hold either **Shift** or **Ctrl** on the keyboard, and then click each object that you want to change; or
   - Use the cursor to draw a selection box around all of the objects that you want to change.

   The objects are selected.
2. Do one of the following:
   - On the **Draw** tab of the ribbon, in the **Editing** group, click **Properties**;
   - Right-click the selected objects, and then on the shortcut menu, click **Properties**; or
   - Press **Alt+Enter** on the keyboard.

   > **Note:** You cannot double-click to open the *Object Properties* dialog box as you otherwise would, because clicking like that clears the selection.

   The *Object Properties* dialog box is displayed for the selected objects.
3. Change the property values that you want to change, and then close the dialog box.

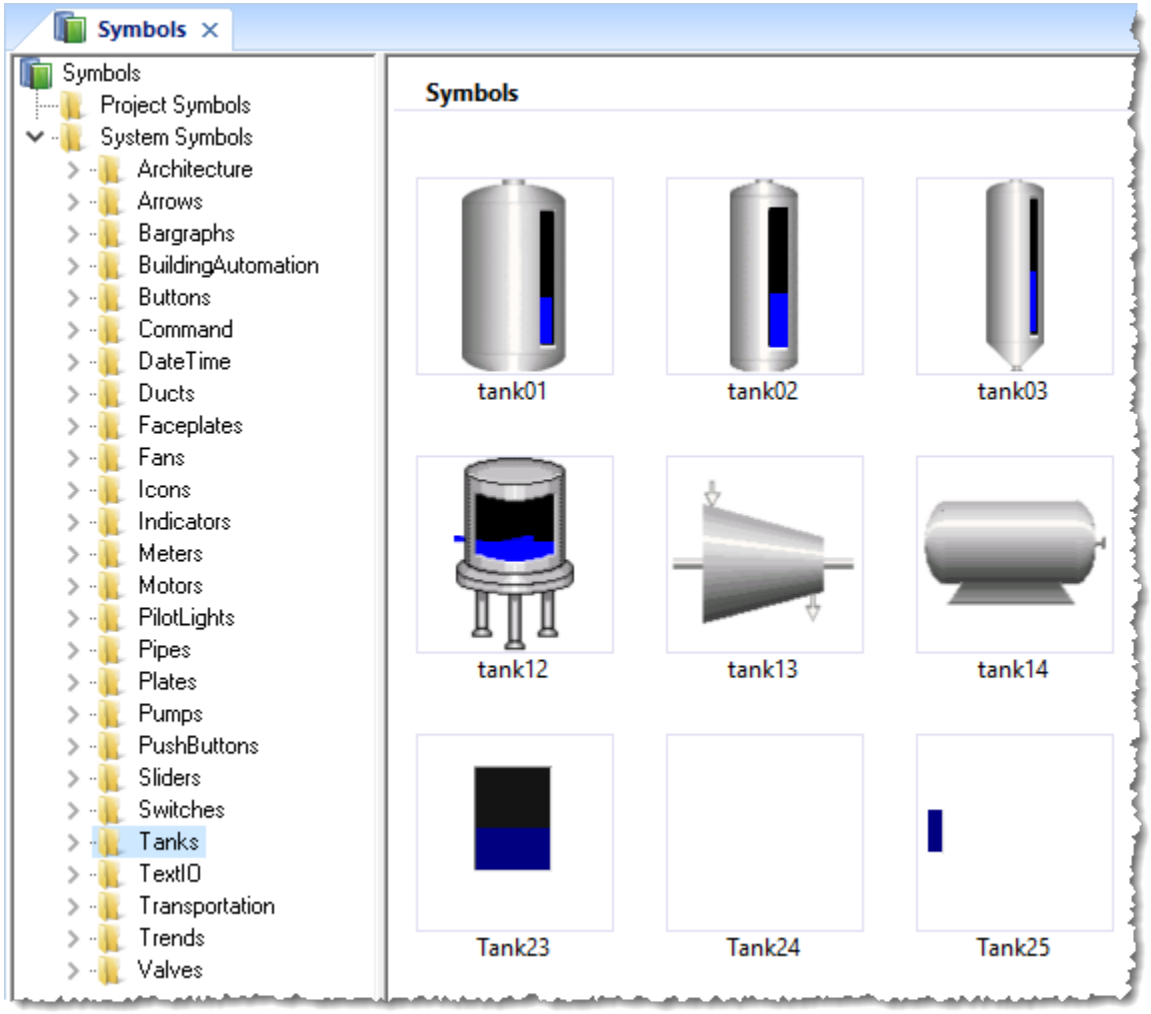
The changes are applied to all of the selected objects.

## *Libraries*

### SYMBOLS LIBRARY

The Symbols library is a visual browser for all of the symbols that are available to be used in your project screens. To open the library: on the **Draw** tab of the ribbon, in the **Libraries** group, click **Symbols**.

The Symbols library is displayed:




***The symbols library***

The library is divided into two main folders: the **Project Symbols** folder contains your user-made symbols for the current project, and the **System Symbols** folder contains all of the premade symbols (sorted by type) that are installed with the EcoStruxure Machine SCADA Expert 2020 software.

To select a symbol and place it in a project screen:

1. Find the symbol you want in the library, and then double-click it. The mouse cursor will change to indicate that you have a symbol waiting to be placed.
2. Return to the project screen where you want to place the symbol.
3. Click anywhere in the project screen to place the selected symbol.
4. Edit the symbol's object properties as needed, including any custom properties.

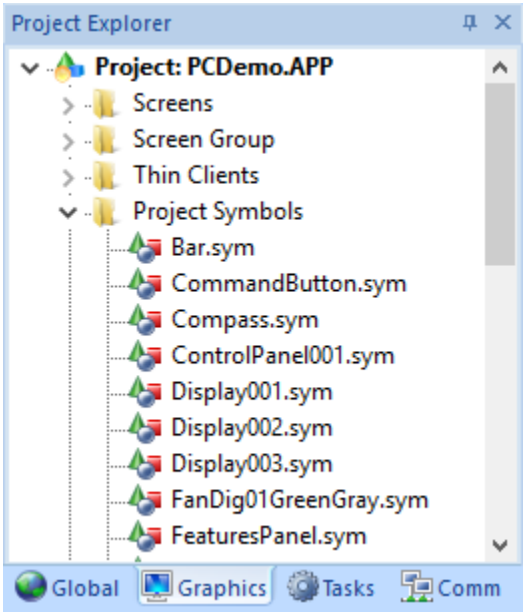
For more information, see Save your own project symbols on page 279.

### Save your own project symbols

A symbol is a set of one or more screen objects that is saved in the Symbols library, so that you can reuse it in your projects.

Every time you reuse a symbol, you actually make a copy that is linked to the master symbol in the Symbols library. (These linked copies are also called "instances" of the symbol.) Thereafter, if you make any changes to the master symbol, those changes automatically propagate to every linked copy in every project.

You can customize each linked copy of the master symbol by defining **Custom Properties**. For example, when you create a gauge that displays tank levels and then save that gauge as a master symbol, you can define custom properties on the symbol that will allow each linked copy to display the level of a different tank.

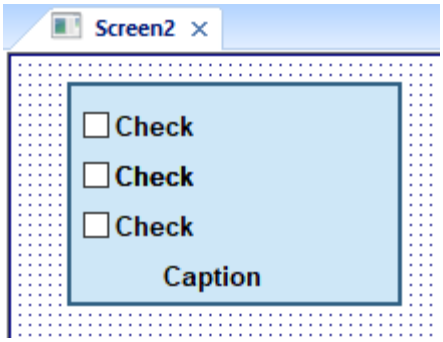



***Project Symbols folder in the Graphics tab***

### Create a master symbol

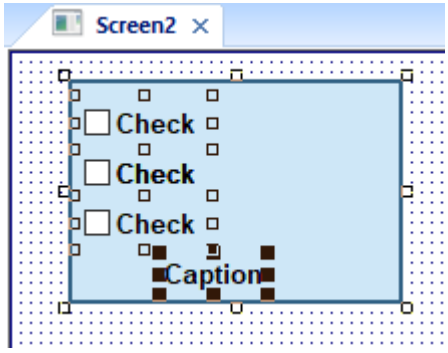
To create a master symbol and save it to the *Symbols* folder:

1. Design your symbol just as you would normally draw a project screen, using any combination of Static and Active Objects. For example, three check boxes in a rectangular pane:




***Drawing objects in a project screen***

2. Select the object(s) or Group that you want to save as a symbol.

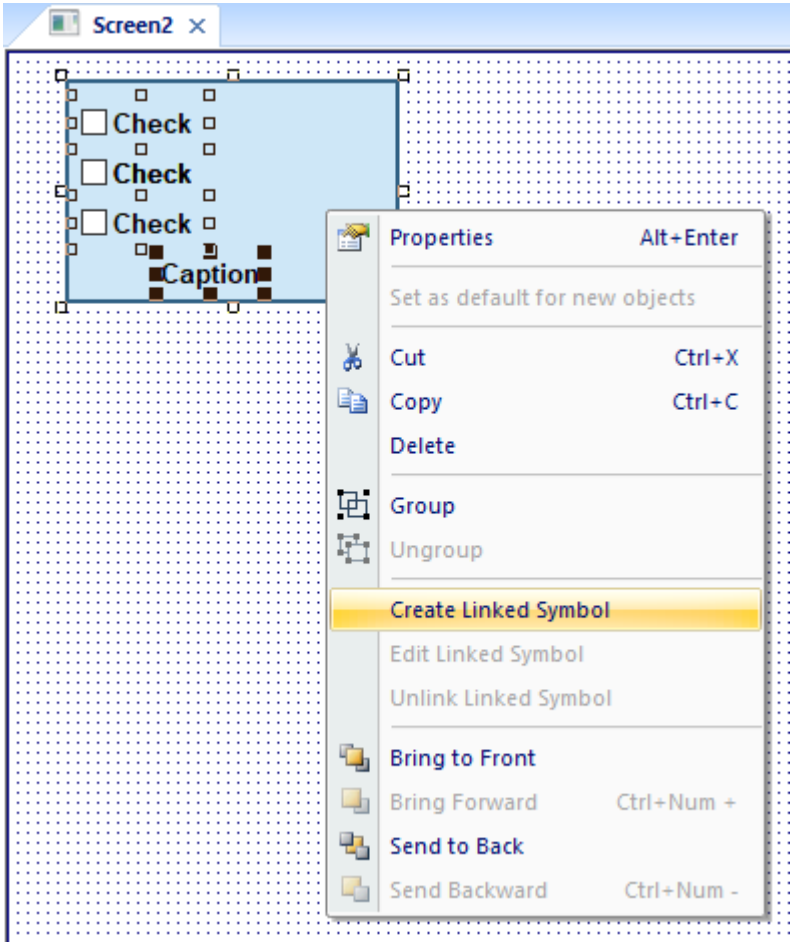



*Selecting the objects*

> **Note:** It is not necessary to make a Group out of two or more objects before saving them as a symbol. Saving the objects together as a symbol effectively groups them.

There is a situation, however, where you may want to group the objects first. A symbol can have only one hint. If more than one object has a hint configured on it (in the Object Properties), those hints are not shown when the objects are saved together as a symbol. To specify a hint for the symbol as a whole, you must first group the objects and then configure the hint on the Group. That hint will carry through when you save the Group as a symbol.

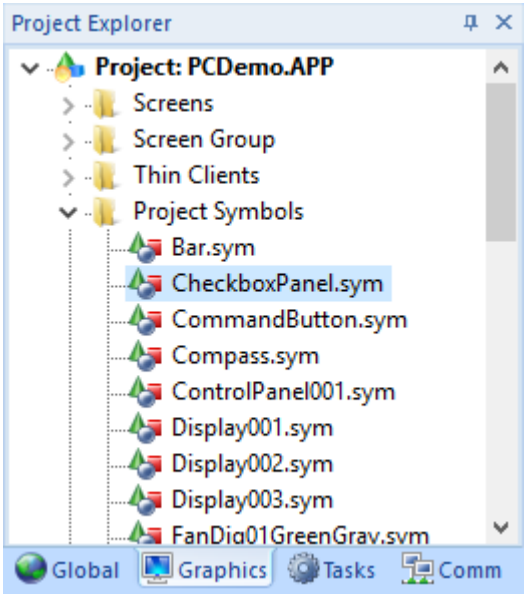
3. Right-click on the selection, and then click **Create Linked Symbol** on the shortcut menu.




***Creating a linked symbol***

4. A standard *Save As* dialog box is displayed, and you are prompted to give the new symbol a file name. Symbol files (**`*.sym`**) are saved in the **`\Symbol`** folder of your project.

5. Click **Save** to save the file. The symbol appears in the Project Symbols folder, in the *Graphics* tab of the Project Explorer.




***Symbol file in the Project Explorer***

The symbol also appears in the *Project Symbols* folder of the Library.

The symbol is now ready to be reused in your project, but the way it is currently saved, every copy will have identical properties. You must now define custom properties on the symbol — that is, the properties you want to be able to customize each time you reuse the symbol.

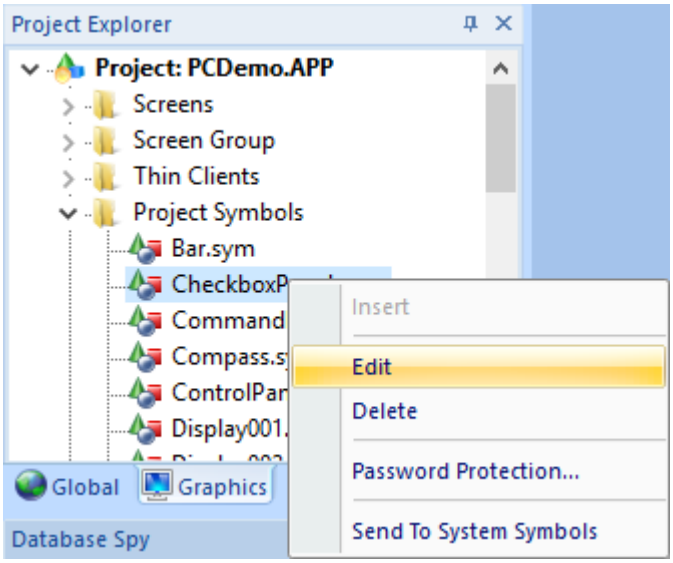
### Edit the master symbol

You can edit a master symbol after you've initially saved it, to add or delete objects in the symbol or to define custom properties on it. Remember that any changes you make to the master symbol will automatically propagate to every linked copy in every project.

> **Note:** There is one exception. If you change the hint on a symbol — as described in "Create a master symbol" — then the change will appear only on new instances of the symbol. Existing instances will be unchanged.

To edit a symbol:

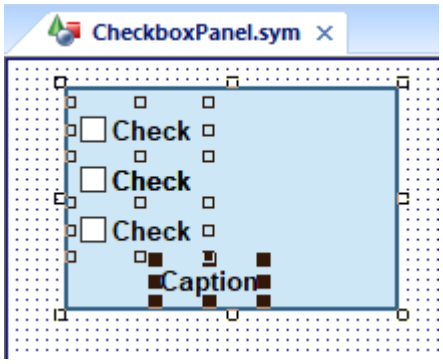
1. Right-click on the symbol file in the *Symbols* folder, and then choose **Edit** from the shortcut menu.




***Editing the symbol file***

> **Tip:** You can also right-click on any instance of the symbol and choose **Edit Linked Symbol** from the shortcut menu.

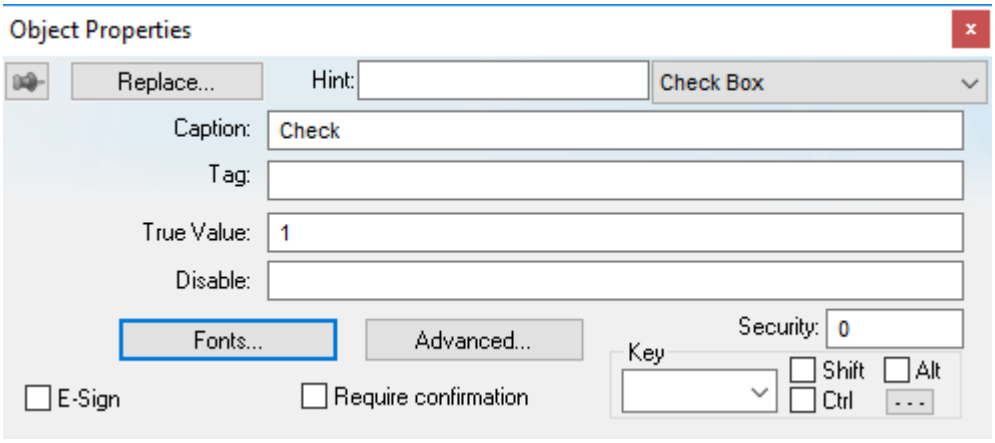
The symbol file is opened for editing in its own window. This symbol editor works in the same way as a regular screen editor, except that every object in the window is part of the symbol. If you add, move or delete objects in the symbol editor, you may change the size or shape of the symbol and disrupt the layout of any Screens where it is used.




***Symbol file opened for editing***

Besides adding, moving or deleting objects in the symbol, you can also edit the Object Properties as you normally would. You may want some properties to be the same in every instance of the symbol, but other properties need to be customized according to where and how the symbol is used. In this example, you probably want to customize the captions for the three check boxes, the tags with which the check boxes are associated, and the caption for the pane itself.

2. Select the first object in the symbol and open its Object Properties. For example, the first check box:




***Object Properties dialog box for the first Check Box object***

3. In any field where you would normally configure a tag, expression, or value, you can instead define a custom property using the sytax...

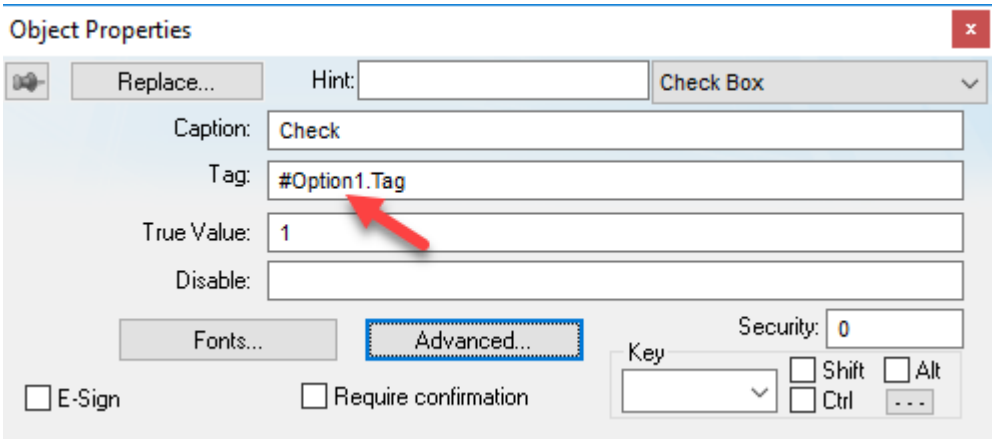
   **`#[Category.]Property:[Value]`**

   ...where:

   - ***`Category`*** is an optional name for a collection of related properties, such as all captions or all Check Box values. If you do not specify ***`Category`*** for a property, it will be automatically listed under the "Main" category.
   - ***`Property`*** is a label to identify the specific property. ***`Property`*** is required for each property, and it must always be followed by a colon (`:`).
   - ***`Value`*** is an *optional* default value for the property.

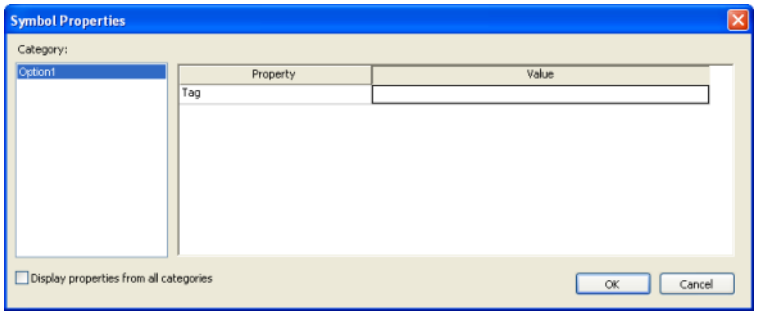
   > **Note:** All Tag/Expression syntax rules apply to *Value*, including tag names, pointers, arrays, strings, numerical and boolean values, and scripting functions.

   In the following example, we want to be able to customize which tag will be set when the Check Box is selected or cleared. So, in the **Tag** field, type **`#Option1.Tag:`** as shown.




***Defining a custom property for the Tag property***

When you go to complete the properties on an instance of the symbol, **`#Option1.Tag:`** will appear like this:

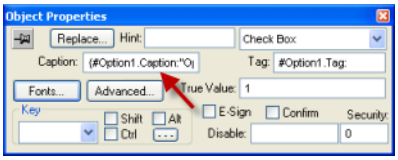



***Custom properties on a symbol***

But more about that later…

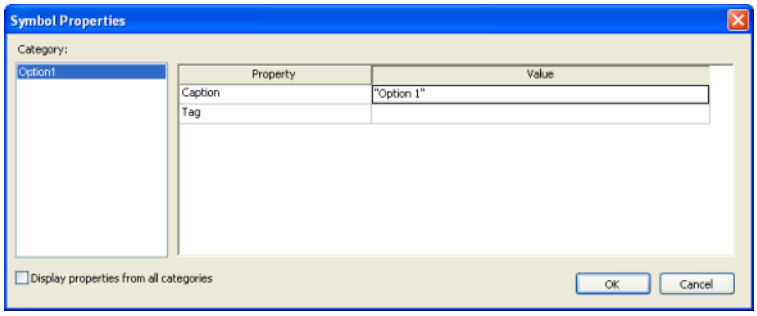
4. Depending on the context, some object properties require a specific type of value like a String, a Boolean or a numerical value. For these properties, you must enclose the custom property declaration in curly brackets (**`{}`**).

   In this example, the **Caption** field requires a String, so type **`{#Option1.Caption:"Option 1"}`** as shown.




***Defining a custom property for the Caption property***

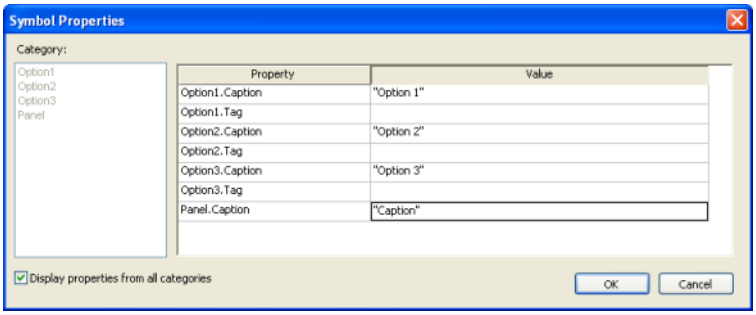
Again, when you go to complete the properties on an instance of the symbol, they will appear like this:




***Custom properties on a symbol***

5. Repeat steps 2 through 4 as needed, to define the rest of the custom properties on the symbol.

In this example, the finished symbol has all of the following properties:

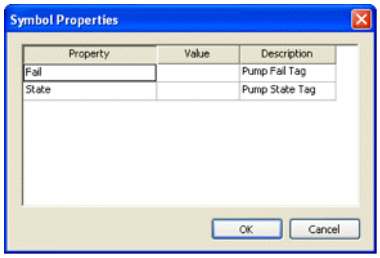



***Custom properties on a symbol***

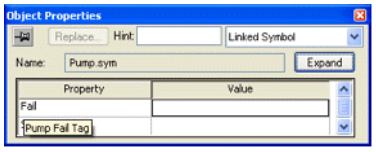
6. Save the symbol and close the symbol editor.
7. On the Home tab of the ribbon, in the Tools group, click **Verify**. This will update all existing instances of the symbol in your project.

### Add tooltips to custom properties

You can configure a description for each custom property available in the symbol. After creating a symbol, open it with the symbol editor, right-click in the symbol editor (not on the symbol itself) and choose **Edit Symbol Properties** from the shortcut menu.




When assigning values to the custom properties of the symbol on the screens, the user can read the description as Tooltips just by moving the mouse cursor on the property name, as illustrated on the following picture:




***Tooltip showing the description of the property***

### Password protect a symbol

You can put a password on any of your user-made symbols to prevent them from being edited or analyzed by other users. To protect a symbol:

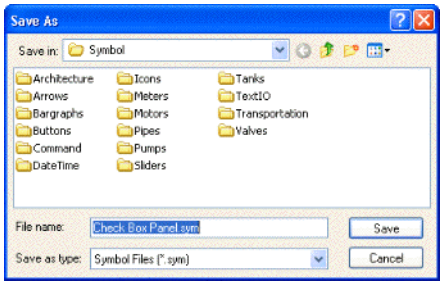
1. In the *Symbols* folder, right-click on the desired symbol file (**`.sym`**) and then choose **Password Protection** from the shortcut menu. A *Password Protection* dialog box is displayed.
2. In the **New Password** field, type your password.
3. In the **Confirm Password** field, type your password again.
4. Click **OK**.

Once this is done, you will be prompted for the password whenever you attempt to edit the symbol or unlink an instance of the symbol.

### Make a user-made symbol available to other projects

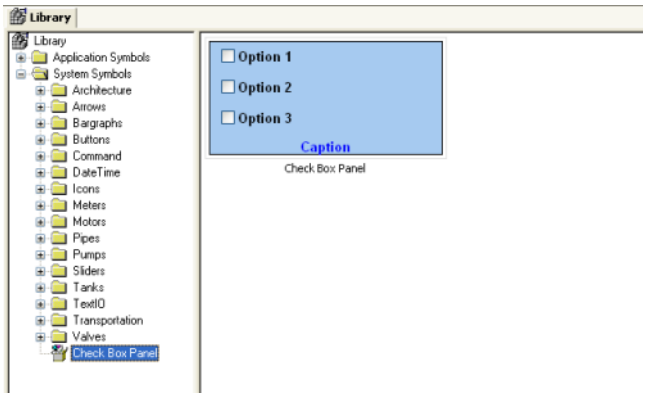
User-made symbols are normally available only in the project where they were initially created and saved. However, you can send a user-made symbol to the *System Symbols* folder of the Library, to make it available to all of your projects:

1. In the *Symbols* folder of the *Project Explorer*, right-click the desired symbol file ( **`.sym`** ) and then choose **Send to System Symbols** from the shortcut menu. A standard *Save As* dialog box is displayed, automatically pointing to the **`\Symbol`** sub-directory of the EcoStruxure Machine SCADA Expert program directory instead of the **`\Symbol`** sub-folder of your project folder.




***Saving a symbol***

2. Choose a location in which to save the symbol file. You can choose one of the existing categories/folders, or you can create a new one.
3. Click **Save**. The symbol file is saved in the specified location and the symbol is displayed in the **System Symbols** folder of the Symbols library.




***Saving a symbol***

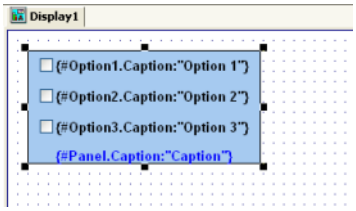
For more information, see Using the Library.

### Insert a symbol in a project screen

To insert a symbol in a project screen and then complete its custom properties:

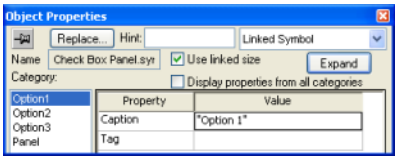
1. Open the desired project screen from the *Screens* folder, or insert a new screen. The screen is opened for editing.
2. Open the Symbols Library by doing one of the following:
   - On the Graphics tab of the ribbon, in the Libraries group, click **Symbols**;
   - Double-click **Symbols** in the Project Explorer; or
   - Right-click in the screen where you want to insert the symbol, and then click **Insert Linked Symbol** on the shortcut menu.

3. Select the symbol from the Symbols Library, and then click in the screen:



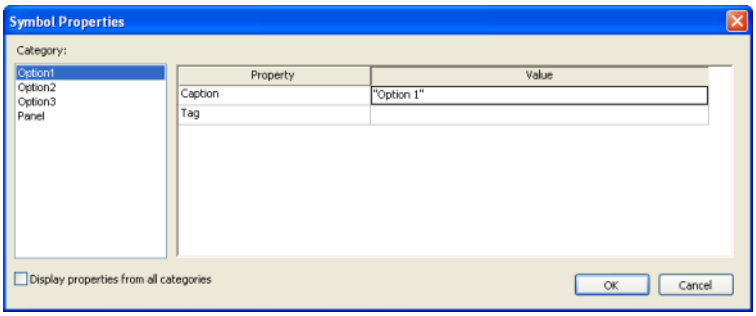

***Symbol placed in a project screen***

Once the symbol is inserted, you can manipulate it like any other object in the screen. You can align and distribute it with other objects, and you can apply Animations to it. However, the first thing to do is complete the custom properties for this instance of the symbol.



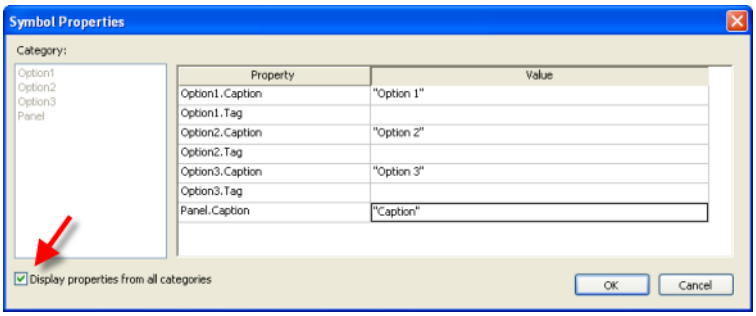

***Object Properties dialog box for the symbol***

4. Open the Object Properties for the symbol.
5. Click **Expand** to open the *Symbol Properties* dialog box.



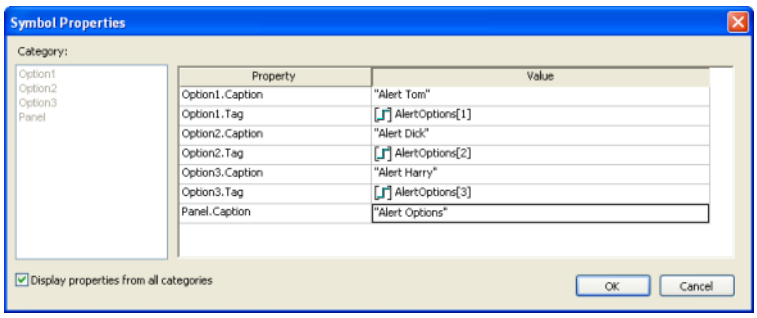

***Symbol Properties dialog box for the symbol***

To see all of the properties at the same time, select the **Display properties from all categories** check box.




***Displaying properties from all categories***

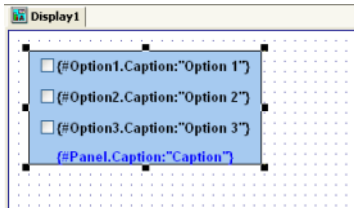
6. Enter the property values as needed. In this example, the three check boxes are used to determine whether to alert Tom, Dick and/or Harry. The captions are updated accordingly, and the check box tags are configured with the first three elements of a Boolean array called **`AlertOptions`**.




***Completed properties for the symbol***

7. Click **OK** to close the *Symbol Properties* dialog box, and then close the *Object Properties* dialog box.

The custom properties are resolved during runtime, as shown below.




***Symbol during editing***

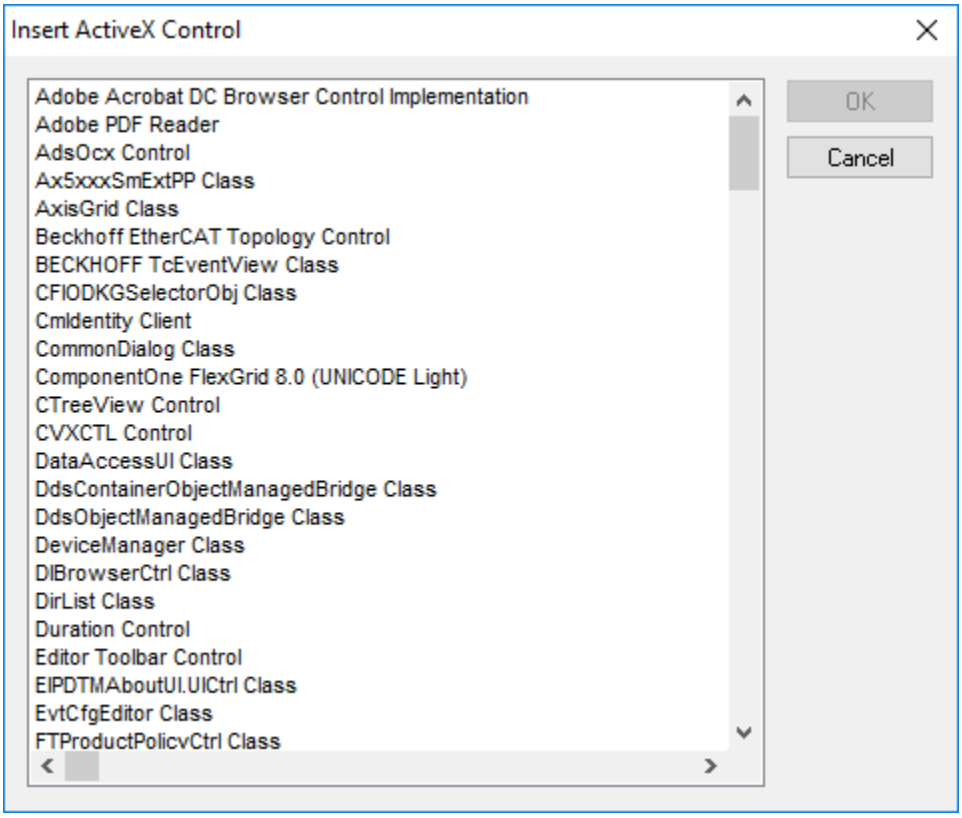
***Symbol during run time***

> **Note:** Remember, the completed custom properties on each instance of a symbol are independent from every other instance of that symbol, but if you make any changes to the master symbol file, those changes automatically propagate to every instance.

## ACTIVEX CONTROL OBJECT

On the **Graphics** tab, in the **Libraries** group, click **ActiveX Control** to open the *Insert ActiveX Control* dialog, which you can use to place ActiveX components on your screen.

When the dialog opens (as in the following figure), it contains a list of all ActiveX components that are registered on your PC.




***Insert ActiveX Control dialog***

**Note:** When you use ActiveX controls in your project, your runtime stations should have the same controls already installed and registered. Stations often have "auto download" and "auto install" features disabled for security reasons, so they may not be able to get ActiveX controls that are called by your project. Consult your hardware manufacturer and ActiveX controls provider for more information about how to manually install controls.

If you still want to enable automatic download of ActiveX controls, you can do so by manually editing your project file (*<project name>*.app) to include the following settings:

```
[UsedControls]
EnableDownload=1
Count=number of controls

[UsedControl1]
CLSID=class ID of the ActiveX control
Version=version of the ActiveX control
Codebase=URL of the ActiveX control file, or of the .CAB file that contains the
 ActiveX control files
RegFile1=name of a specific .OCX or .DLL file within the .CAB file; see below
RegFilen=name of a specific .OCX or .DLL file within the .CAB file; see below
…

[UsedControln]
…
```

The **CLSID** and **Version** settings are required for each ActiveX control, and they must match the ID and version of the actual control file(s) to which **Codebase** links. This allows a runtime station

to check the control against those that are already registered. If the settings do not match, then the runtime station may unnecessarily download the same control again.

If you don't know the **CLSID** and **Version** settings for an ActiveX control, you can find them in the registry key of an already installed and registered control. Search for the control file in **HKEY_CLASSES_ROOT\CSLID** in the Windows Registry.

Also, the URL for the **Codebase** setting can be either absolute or relative to the Web server's "home" directory. For example:

```
Codebase=http://server_address/AddOns/IndDateTimePick.ocx
```

...or...

```
Codebase=AddOns/IndDateTimePick.ocx
```

Finally, the **Regfile** settings are required only if **Codebase** links to a .CAB file. If it does, then use one or more **Regfile** settings to name the specific files within the .CAB file that must be downloaded and registered.

ActiveX controls are components designed according to a standard. Because EcoStruxure Machine SCADA Expert 2020 is an ActiveX container, you can configure and run ActiveX controls in project screens. ActiveX controls can provide the following interfaces:

- **Properties**: Variables whose values can be read and/or written for your project (e.g., Object Color, FileName, URL, and so forth)
- **Methods**: Functions from the ActiveX object that can be triggered by your project (e.g., open a dialog, execute a calculation, and so forth)
- **Events**: Internal messages that can trigger the execution of expressions in your project (e.g., Mouse_Click, Download_Completed, and so forth)

The name of the properties, methods and events supported by each ActiveX depends on its own implementation.

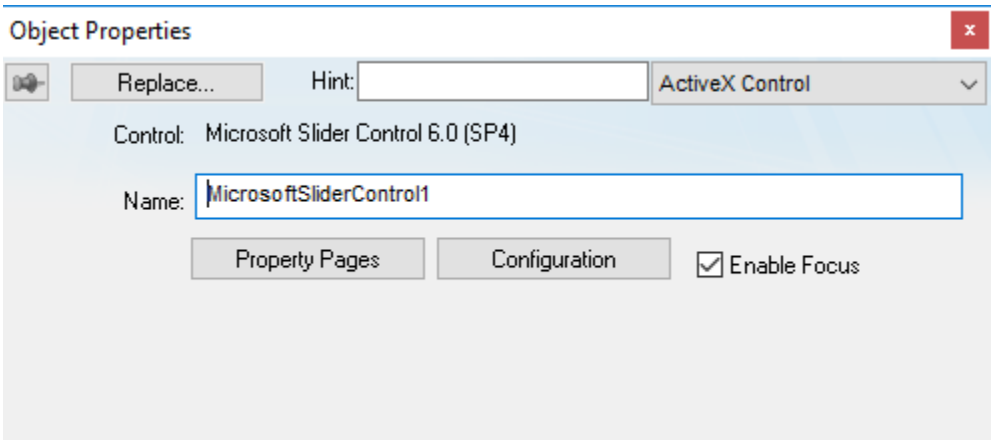
There are two different ways to interface your project with the ActiveX control:

- By using the ActiveX functions **XGet()**, **XSet()** and **XRun()**

  OR

- By using the *Object Properties* window to configure the object

Double-click on the ActiveX Control to open the *Object Properties* dialog.




***Object Properties: ActiveX Control***

The *Object Properties* window displays the name of the ActiveX control. Generally, each ActiveX control is either a **`*.dll`** or a **`*.ocx`** file registered in your local computer. You must assign a name (alias) to the ActiveX control on the Name field (e.g., MyControl). This name is used to reference the object when calling one of the ActiveX functions that are provided in the Built-in Scripting Language.

> **Note:** You should not configure two ActiveX controls on the same screen with the same name. For instance, if you insert two "Windows Media Player" ActiveX controls on the same screen, and assign the name MyMP1 to one object (Name field), you cannot assign the same name to the second object on the same screen. You would have to assign the name MyMP2, for example, to the second object.

The **Property Pages** button opens the standard window for configuring the Static Properties (if any). The layout and the options in this dialog depend on the implementeation of each ActiveX Control. Use this interface to set properties that should not be changed during runtime (fixed properties).

The **Configuration** button on the *Object Properties* window opens dialogs that allow you to do the following:

- Associate tags to properties of the ActiveX Control;
- Trigger methods from the ActiveX Control based on tag change; and
- Configure scripts, which are executed when Events from the ActiveX Control occur.

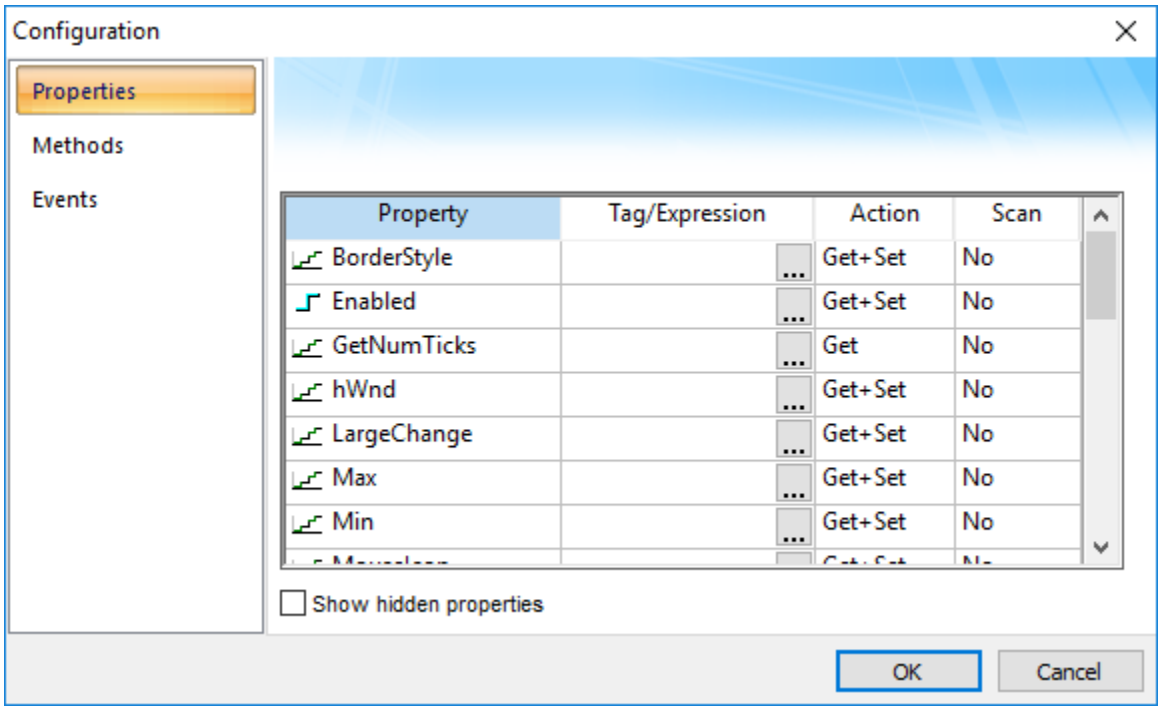
The following sections describe how to configure these interfaces.

> **Note:** Although the Configuration dialog displays the list of all properties, methods and events, you only have to configure the items that you need for your project.

The screen shots used in the following sections depict the Acrobat 3D Office control. The names of the properties, methods and events vary for each ActiveX control, but the configuration interface is the same. The concepts described here apply to all controls.

### Configuring Properties

The *Properties* tab provides a grid with the following fields:




***Configuration Dialog – Properties Tab***

- **Property**: Lists all properties available from the ActiveX object, and indicate their types:

| Property Icon | Property Type |
|---|---|
| | Boolean |
| | Integer |
| | Real |
| | String |

- **Tag/Expression**: The tag configured in this field is associated with the respective property of the ActiveX object. The Action column will define whether the value of this tag will be written to the ActiveX property, or if the value of the ActiveX property will be written to this tag (or both).

> **Note:** You can configure an expression in this field if you want to write the result of an expression to the property of the ActiveX object. However, in this case, the value of the property cannot be read back to one tag (unless you use the XGet() function). Therefore, an expression is configured in this field, the Scan field is automatically set to Set.

- **Action**: Defines the direction of the interface between the tag or expression configured in the Tag/Expression field and the ActiveX property, according to the following table:

| Action | Description |
|---|---|
| Get | Read the value of the ActiveX property and write it to the tag configured in the **Tag/Expression** field. |
| Set | Write the value from the tag or expression configured in the **Tag/Expression** field into the ActiveX property. |
| Get+Set | Executes both actions (**Get** and **Set**). However, when opening a screen with the ActiveX object, the **Get** command is executed before any **Set** command is executed. In other words, the tag configured in the **Tag/Expression** field is updated with the value of the ActiveX property when the project screen is opened. |
| Set+Get | Executes both actions (**Get** and **Set**). However, when opening a screen with the ActiveX object, the **Set** command is executed before any **Get** command is executed. In other words, the ActiveX property is updated with the value of the tag configured in the **Tag/Expression** field when the project screen is opened. |

> **Note:** When the value of the property is "Read-only" (cannot by overwritten by your project), the **Action** field is automatically set to **Get**.

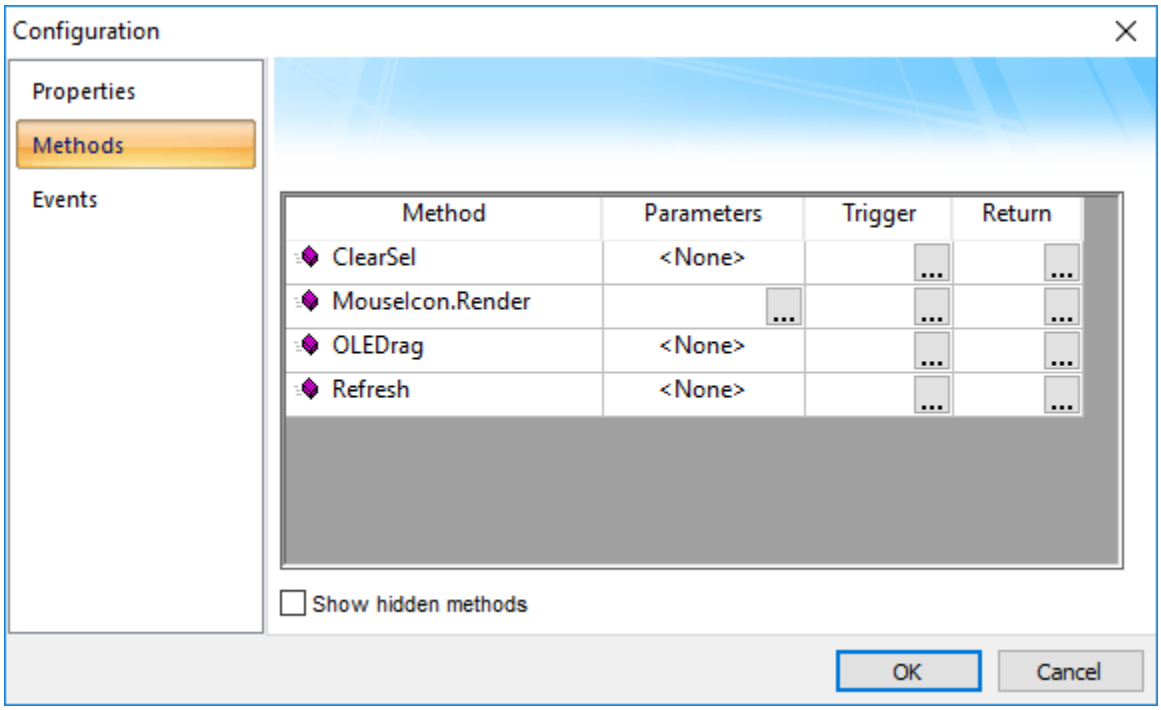
- **Scan**: Defines the polling method to get values from the ActiveX propreties, according to the following table:

| Scan | Description |
|---|---|
| No | The project runtime server gets the value only when the ActiveX object sends a message that the value has changed. |
| Always | The project runtime server continuously gets the value while the project screen that contains the ActiveX object is open. |

> **Note:** Some ActiveX controls are designed to send messages to their containers (e.g., your project) indicating that a property changed value and the new value should be read (Get) again. However, other ActiveX controls do not implement this algorithm. In this case, the only way to get the updated values of the ActiveX properties is to keep polling these values from the ActiveX control (Scan=Always).

### Configuring Methods

The Methods tab provides a grid with the following fields:




***Configuration Dialog – Methods Tab***

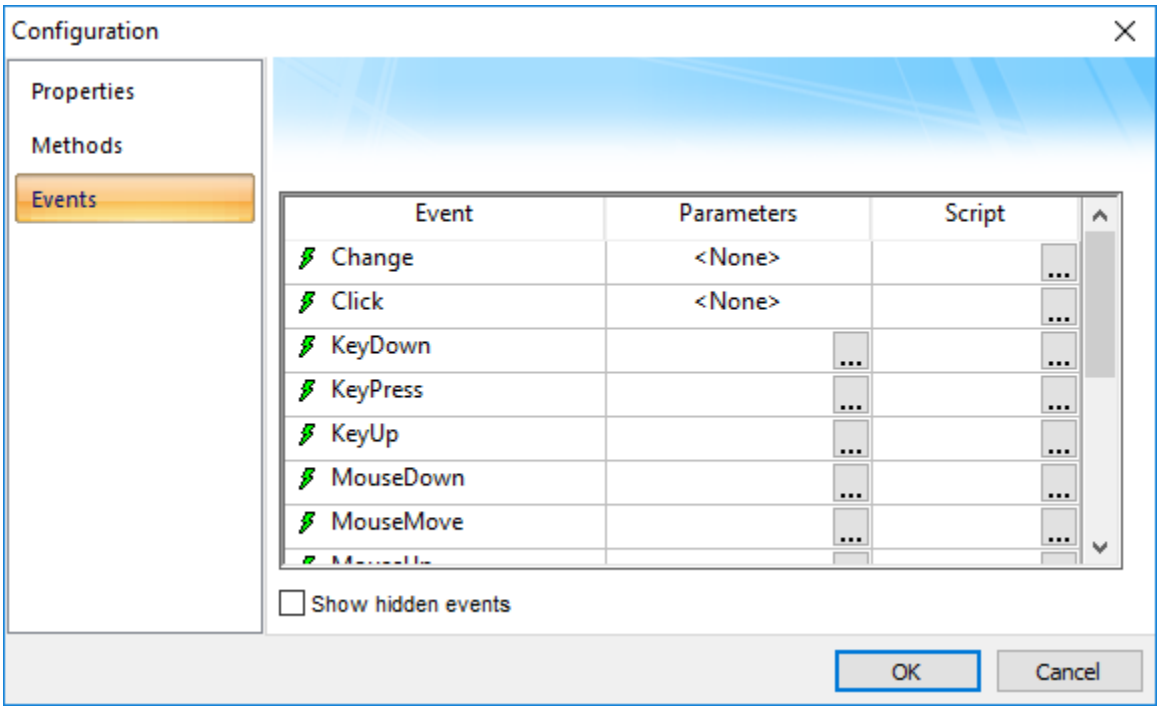
- **Method**: List all methods available from the ActiveX object.
- **Parameters**: The tags configured in this field are associated with the parameters of the method of the corresponding ActiveX object. If the method does not support any parameter, the fixed text <None> is displayed in the Parameters field. Otherwise, you can type the tags associated in the parameters of the ActiveX object. When the method has more than one parameter, you can type one tag for each parameter, separating them by a comma ( , ). For example, **`TagA , TagB , TagC`**. When the method is executed, either the value of the tags are written to the parameters of the method (input parameters), or, after the method is executed, the ActiveX writes the value of the parameters to the tags (output parameters).

  > **Tip:** When you click the Browse button (...), it will display the list of parameters supported by the method, allowing you to associate one tag with each parameter.

- **Trigger**: When the tag configured in this field changes value, the respective method of the ActiveX control is executed.
- **Return**: The tag configured in this field receives the value returned by the method (if any).

### Configuring Events

The Events tab provides a grid with the following fields:




*Configuration Dialog – Events Tab*

- **Event**: List all events available from the ActiveX object.
- **Parameters**: The tags configured in this field are associated with the parameters of the event of the corresponding ActiveX object. If the event does not support any parameter, the fixed text <None> is displayed in the Parameters field. Otherwise, you can type the tags associated with the parameters of the ActiveX object. When the event has more than one parameter, you can type one tag for each parameter, separating them by a comma (,). For example, **`TagA , TagB , TagC`**. When the event is generated, either the value of the tags are written to the parameters of the event (input parameters), or the parameter values are written to the tags (output parameters).

  > **Tip:** When you click the Browse button (...), it will display the list of parameters supported by the event, allowing you to associate one tag with each parameter.

- **Script**: The script configured in this field will be executed when the event is triggered by the ActiveX control.

  > **Tip:** When you click the Browse button (...), it will display a dialog with the complete script associated with the event. The main dialog displays only the expression configured in the first line of the script.

## .NET CONTROL OBJECT

.NET Components are designed according to the Microsoft .NET Framework, which is a standard for modular programming technologies. Because EcoStruxure Machine SCADA Expert is a .NET container, you can configure and run .NET Components in your project screens. The actual functions of a .NET Component are contained within a **.NET Control** object, which provides the configuration dialogs.

.NET Components include the following interfaces:

- **Properties**: Variables whose values can be read and/or written for your project (e.g., Object Color, FileName, URL, and so forth)

- **Methods**: Functions from the .NET Component that can be triggered by your project (e.g., open a dialog, execute a calculation, and so forth)
- **Events**: Internal messages that can trigger the execution of expressions in your project (e.g., Mouse_Click, Download_Completed, and so forth)

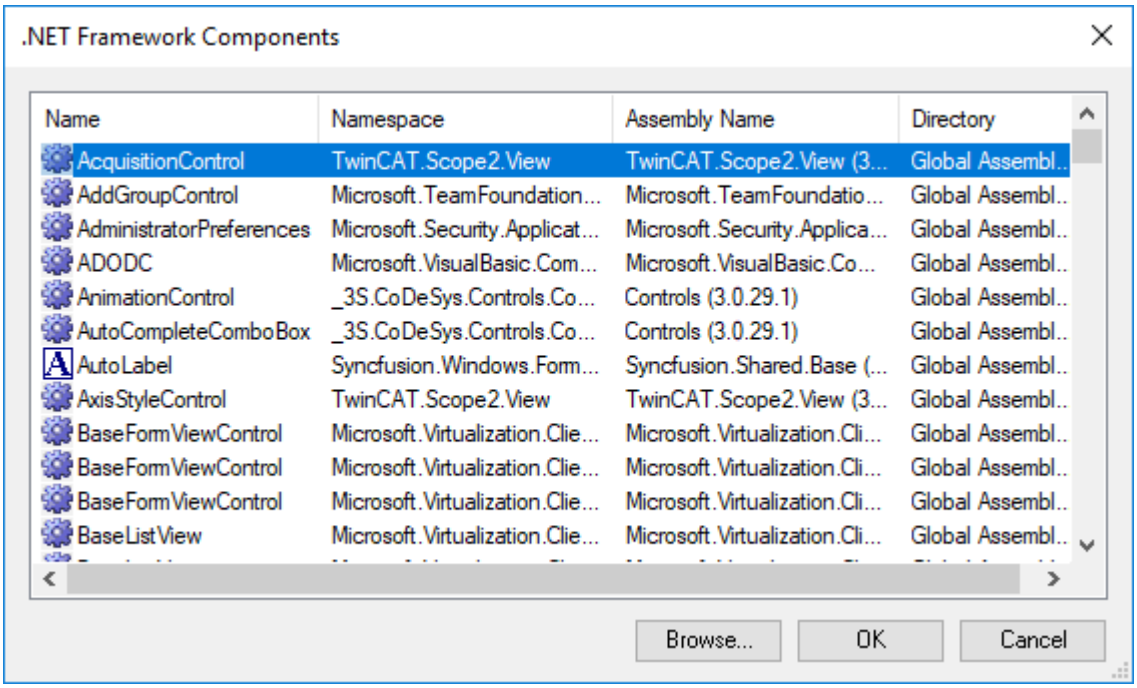
The properties, methods and events supported by each .NET Component vary according to the component's implementation.

When using .NET Components in your project, make sure that the target system (runtime station) can support the same components and that they are properly installed and registered. Your project includes links to the .NET Components; however, the installation of these components on the target system must be done separately. Furthermore, when .NET Components are used on screens open in remote Thin Clients, the .NET Components must also be manually installed on the Thin Client stations. The Microsoft Windows operating system installs a large selection of components by default, but additional components are offered by third-party providers. Consult your .NET Component provider for further information about how to install.

### Selecting and Placing a .NET Control Object

To select and place a .NET Control object in your project screen:

1. On the **Graphics** tab, in the **Libraries** group, click **.NET Control**. The *.NET Framework Components* dialog box is displayed.




***.NET Framework Components dialog***

This dialog box lists all of the .NET components that are installed and registered on your computer, but EcoStruxure Machine SCADA Expert 2020 does not necessarily support all of the listed components. In order to be placed in a project screen, a component must meet the following requirements:

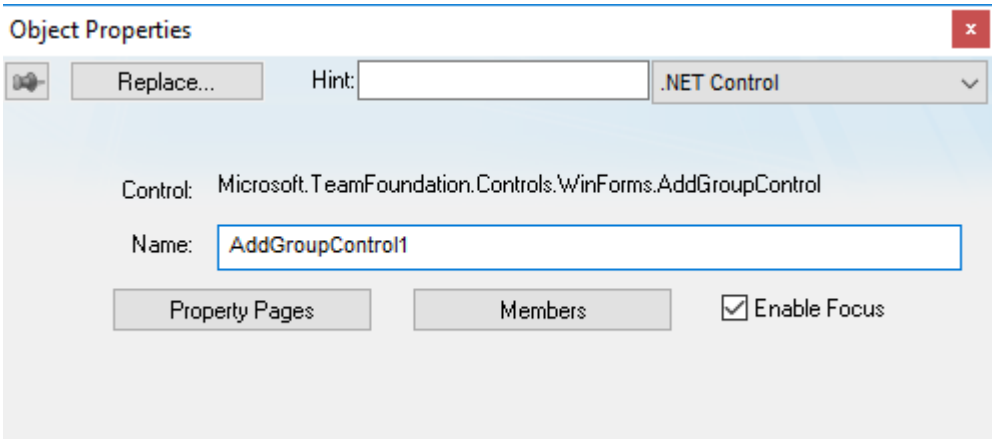
- It must be built with .NET Framework version 2.0, 3.0, or 3.5. Components that have been built with .NET Framework 4.0 or later are not supported.
- It must be designed using Windows Forms (WinForms) rather than Windows Presentation Foundation (WPF). Components that have been designed using WPF are not supported. You can use third-party development tools such as Visual Studio to "wrap" a WPF-based component, however, so that it has WinForms control layer and therefore can be used in EcoStruxure Machine SCADA Expert 2020.
- It must be designed as a User Control — that is, it must extend the System.Windows.Forms.UserControl class.

- The DataGrid and DataGridView controls are not supported in any case. As an alternative, use EcoStruxure Machine SCADA Expert 2020's own Grid object.

2. Select a component from the list, and then click **OK** to place it in your project screen. You can also click the **Browse...** button to find an unregistered component on your computer.

> **Tip:** Registered .NET Components are typically stored in the following directory:
>
> ```
> C:\WINDOWS\Microsoft.NET\Framework\
> ```
>
> However, you can have the application include unregistered components in the *.NET Framework Components* dialog by editing the `<project name>.APP` file to add this parameter:
>
> ```
> [Execution Environment]
> DotNetControlPath=OptionalPath
> ```
>
> For example:
>
> ```
> [Execution Environment]
> DotNetControlPath=C:\DOTNET CONTROLS BACKUP
> ```
>
> Thereafter, the *.NET Framework Components* dialog will list all registered components *and* all components found in the specified directory.

3. By default, a new .NET Control object is placed in the upper-left corner of your project screen. Click on the object and drag it to where you want it placed.
4. Once the object is placed, double-click on it to open its *Object Properties* dialog.




***Object Properties: .NET Control***

The *Object Properties* dialog shows the name of the .NET Component. You must assign a name (alias) to the component in the **Name** box (e.g., CheckBox1). This name is used to reference the component when using the scripting languages (VBScript and built-in scripting).

> **Note:** You should not configure two .NET Control objects on the same screen with the same name. For instance, if you place two CheckBox components on the same screen and assign the name CheckBox1 to one object (**Name** field), you cannot assign the same name to the second object on the same screen. You would have to assign the name CheckBox2, for example, to the second object.

The **Property Pages** button opens the standard window for configuring the Static Properties (if any). The layout and the options in this dialog depend on the implementeation of each .NET Component. Use this interface to set properties that should not be changed during runtime (fixed properties).

The **Members** button on the *Object Properties* dialog opens additional dialogs that allow you to do the following:

- Associate tags to properties of the .NET Component
- Trigger methods from the .NET Component based on tag change
- Configure scripts, which are executed when Events from the .NET Component occur

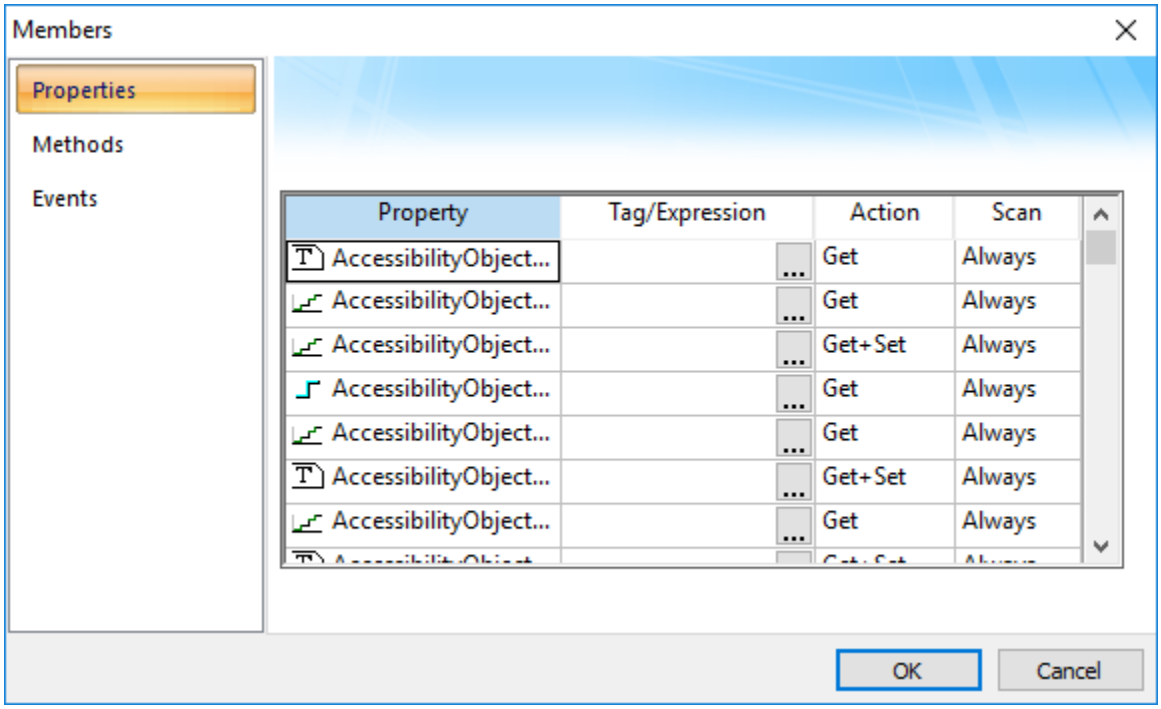
The following sections describe how to configure these interfaces.

> **Note:** Although the *Members* dialog displays the list of all properties, methods and events, you only have to configure the items that you need for your project.

The screen shots used in the following sections depict the CheckBox component. Although the names of properties, methods and events varies by component, the configuration interface is the same for any .NET Component. The concepts described here apply to all of them.

### Configuring Properties

The *Properties* tab provides a grid with the following fields:




***Members Dialog – Properties tab***

- **Property**: List all properties available from the .NET Component, and indicate their types:

| Property Icon | Property Type |
|---|---|
| | Boolean |
| | Integer |
| | Real |
| | String |

- **Tag/Expression**: The tag configured in this field is associated with the respective property of the .NET Component. The Action column will define whether the value of this tag will be written to the property, or if the value of the property will be written to this tag (or both).

- **Action**: Defines the direction of the interface between the tag or expression configured in the Tag/Expression field and the .NET property, according to the following table:

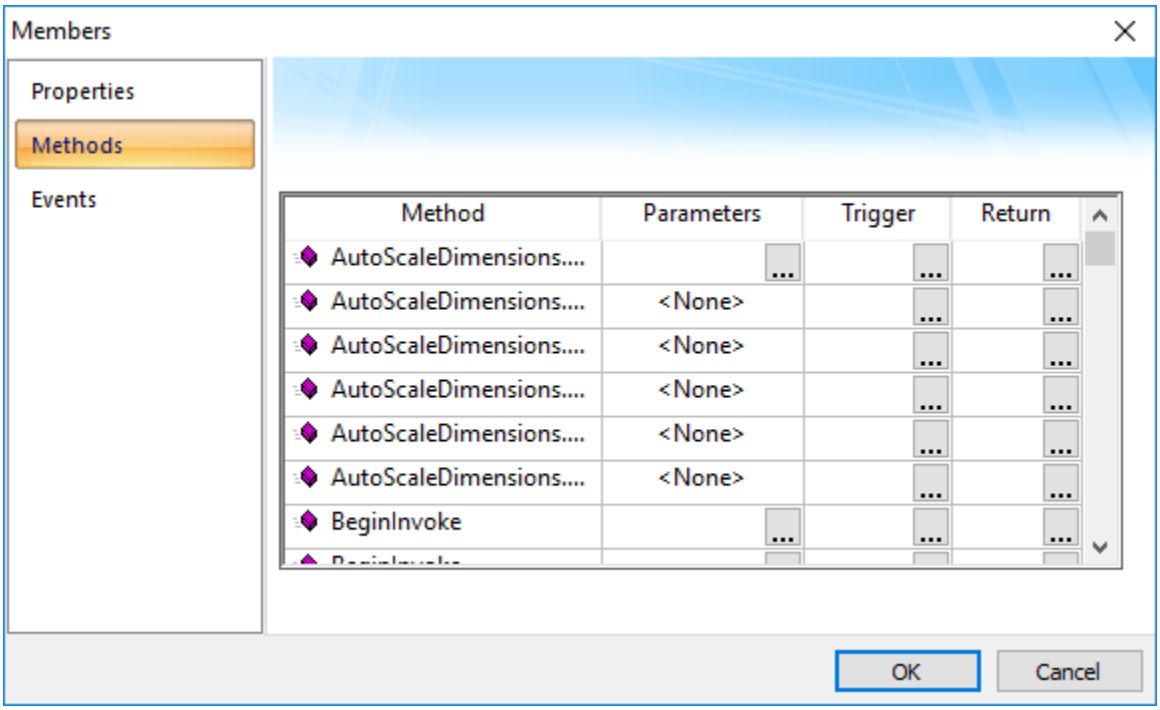
| Action | Description |
|---|---|
| Get | Read the value of the property and write it to the tag configured in the **Tag/Expression** field. |
| Set | Write the value from the tag or expression configured in the **Tag/Expression** field into the property. |
| Get+Set | Executes both actions (**Get** and **Set**). However, when opening a screen with the .NET Component, EcoStruxure Machine SCADA Expert executes the **Get** command before executing any **Set** command. That is, the tag configured in the **Tag/Expression** field is updated with the value of the property when EcoStruxure Machine SCADA Expert opens the screen where the .NET Component is configured. |
| Set+Get | Executes both actions (**Get** and **Set**). However, when opening a screen with the .NET Component, EcoStruxure Machine SCADA Expert executes the **Set** command before executing any **Get** command. That is, the property is updated with the value of the tag configured in the **Tag/Expression** field when EcoStruxure Machine SCADA Expert opens the screen where the .NET Component is configured. |

> **Note:** When the value of the property is "Read-only" (cannot by overwritten by your project), the **Action** field is automatically set to **Get**.

- **Scan**: Defines the polling method to get values from the propreties. For .NET Components, all properties scan **Always** by default. That is, EcoStruxure Machine SCADA Expert keeps polling the value of the property and updating the tag configured in the **Tag/Expression** field with this value.

### Configuring Methods

The *Methods* tab provides a grid with the following fields:




***Members Dialog – Methods tab***

- **Method**: Lists all methods available from the .NET Component.
- **Parameters**: The tags configured in this field are associated with the corresponding method. If the method does not support any parameter, then the fixed text **<None>** is displayed. Otherwise, you can enter the tags that you want to associate with the parameter. When the method has more than one parameter, you can enter one tag for each parameter, separating them by a comma (,). For example, **`TagA , TagB , TagC`**.

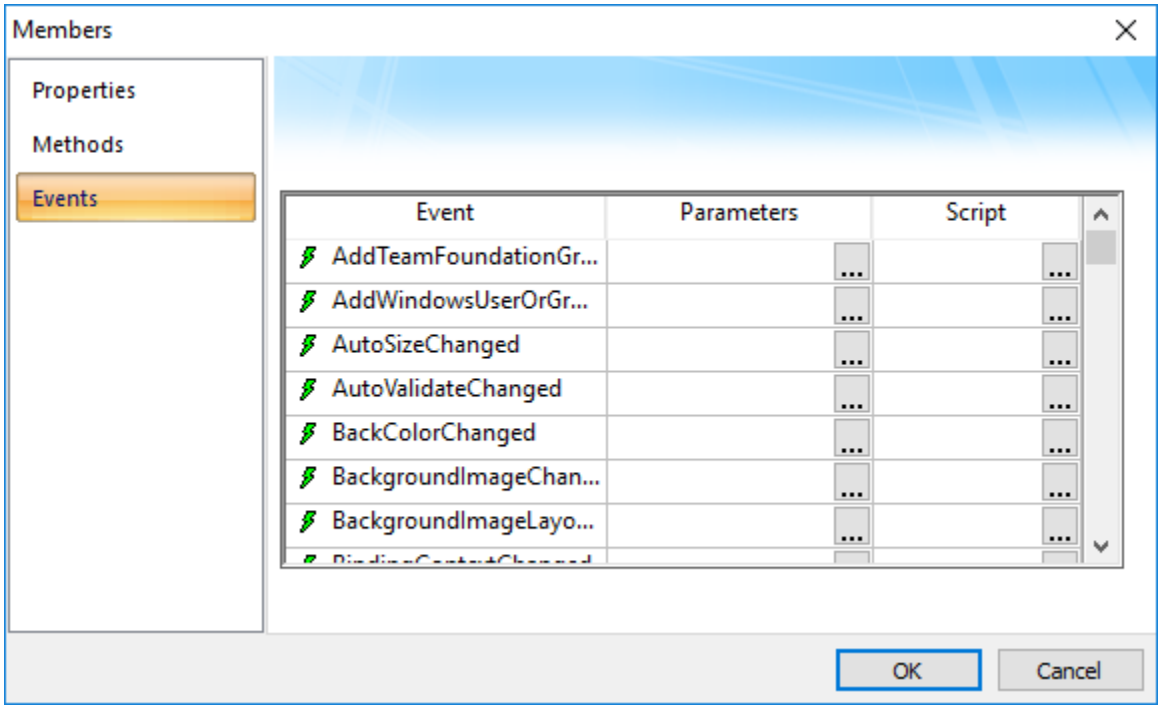
> **Tip:** When you click the Browse button (...), it will display the list of parameters supported by the method, allowing you to associate one tag with each parameter.

When the method is executed, either the value of the tags are written to the parameters of the method (input parameters), or, after the method is executed, the .NET Component writes the value of the parameters to the tags (output parameters).

- **Trigger**: When the tag configured in this field changes value, the respective method of the .NET Component is executed.
- **Return**: The tag configured in this field receives the value returned by the method (if any).

### Configuring Events

The *Events* tab provides a grid with the following fields:




***Members Dialog – Events tab***

- **Event**: Lists all events available from the .NET Component.
- **Parameters**: The tags configured in this field are associated with the corresponding event. If the event does not support any parameter, then the fixed text **<None>** is displayed. Otherwise, you can enter the tags that you want to associate with the parameter. When the event has more than one parameter, you can enter one tag for each parameter, separating them by a comma (,). For example, **TagA , TagB , TagC**.

> **Tip:** When you click the Browse button (...), it will display the list of parameters supported by the event, allowing you to associate one tag with each parameter.

When the event occurs, either the value of the tags are written to the parameters of the method (input parameters), or, after the event occurs, the .NET Component writes the value of the parameters to the tags (output parameters).

- **Script**: The script configured in this field will be executed when the event is triggered by the .NET Component.

> **Tip:** When you click the Browse button (...), it will display a dialog with the complete script associated with the event. The main dialog displays only the expression configured in the first line of the script.

## CUSTOM WIDGET

A custom widget is a type of screen object that displays an external, HTML5-compliant webpage within a frame in a project screen. The widget can do anything that the webpage could normally do when viewed in a browser.

Custom widgets are a platform-agnostic alternative to ActiveX and .NET controls, which are supported only on Microsoft Windows. In fact, each widget is essentially a small, embedded browser window that loads a specified webpage. Also in contrast to ActiveX and .NET controls, these webpages do not need to be compiled, installed, or registered on a computer before they can be used. They are included with the rest of the project files when you download your project to a target device.

You can create a library of custom widgets for your project, and then you can reuse those widgets as many times as you want in any of your project screens. Each instance of a widget is a discrete screen object with its own object properties.

When you create a new widget (or edit an existing widget), you can define properties and events for that widget:

- Properties are used to exchange data between the webpage and the rest of your project. They are similar to memory registers in a PLC: you can associate them with project tags, and then you can read/write them when their values change.
- Events are used to trigger actions in your project. Depending on how it is designed and used, the webpage can send events through the widget to your project, and then those events cause scripts to be executed in your project.

All instances of a custom widget have the same basic properties and events, because those instances are simply copies of the master in the library, but you can configure the object properties of each instance in order to associate different tags and attach different scripts.

As for the webpage itself, you can develop it to do anything you want using HTML5, CSS, and JavaScript. Studio will automatically create the web files in your project folder, when you create the new widget and add it to your project's library, but after that you can freely edit the the files.

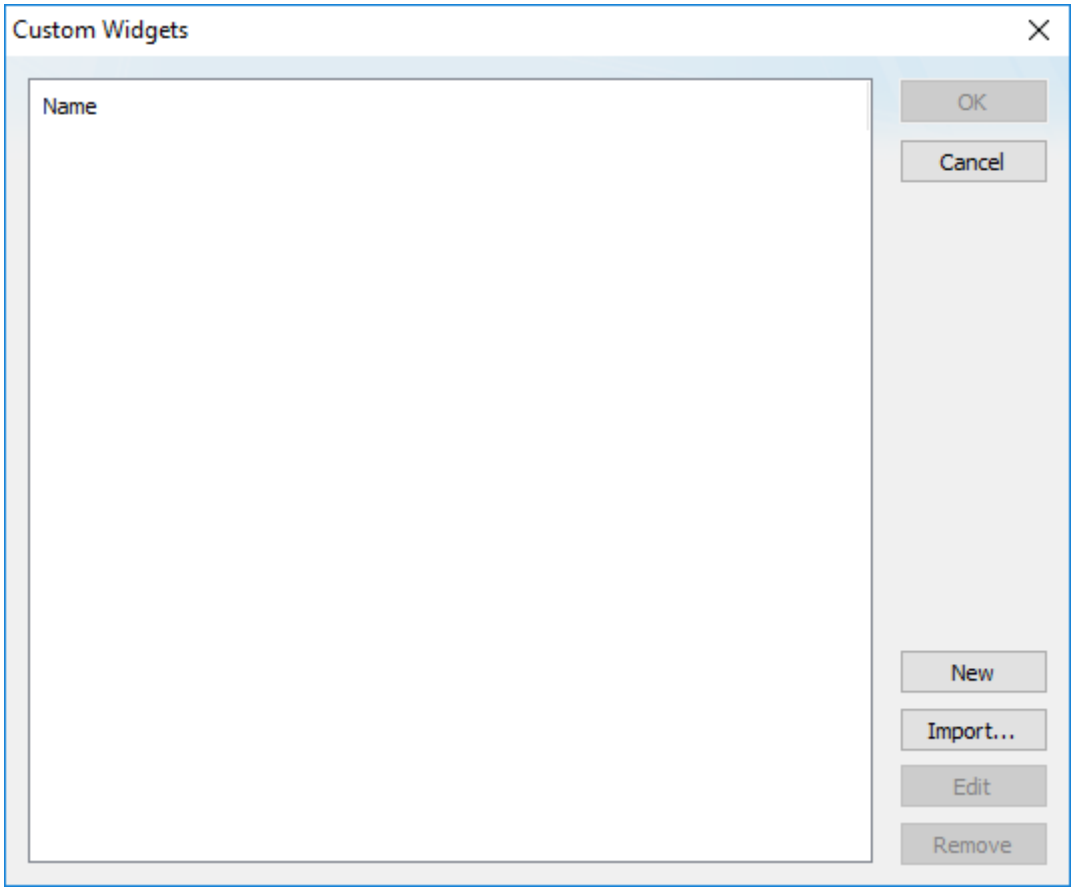
### Create a new custom widget

Use the **Custom Widgets** command to create a new custom widget and then add it to your project's library.

To create a new custom widget:

1. On the **Draw** tab of the ribbon, in the **Libraries** group, click **Custom Widget**.

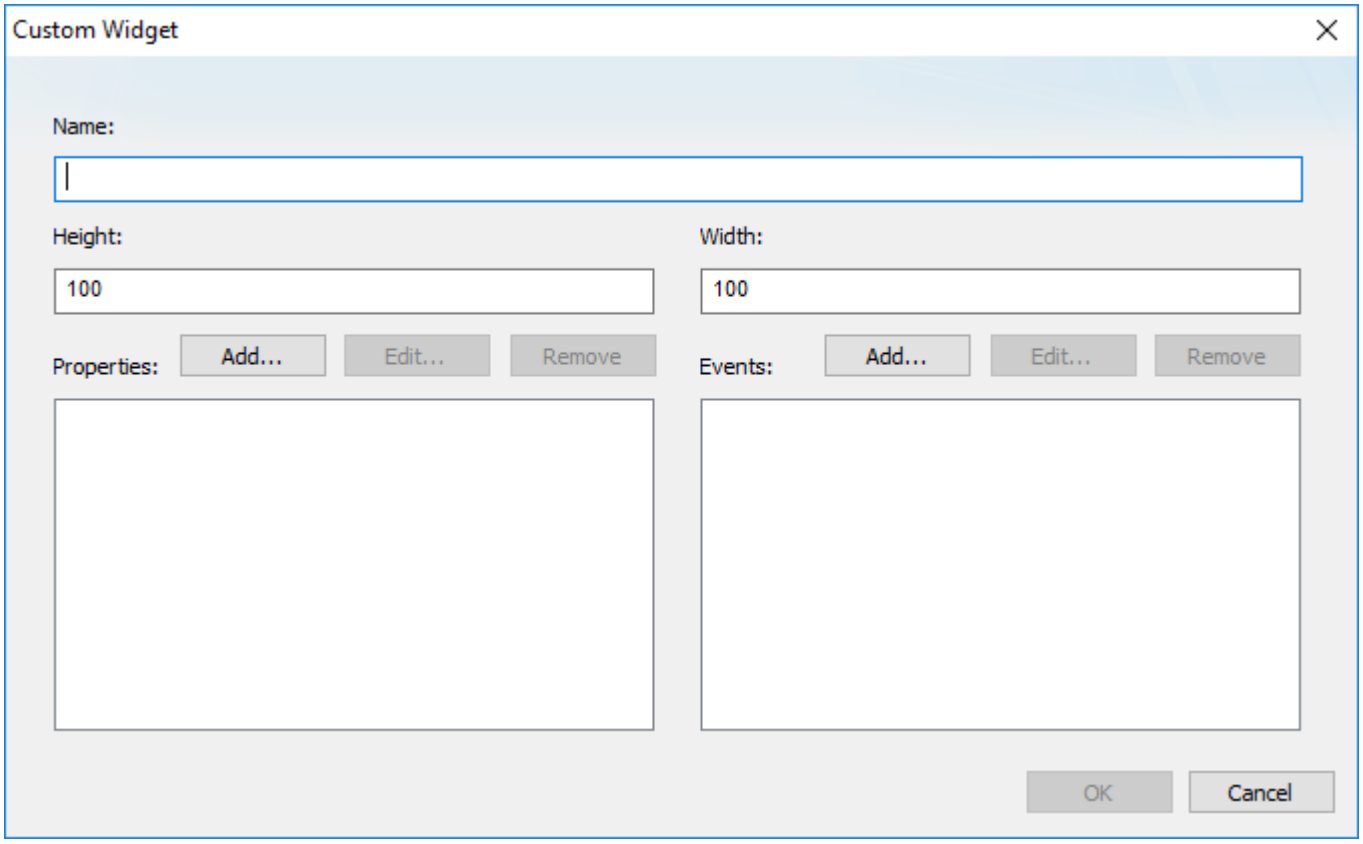
The *Custom Widgets* dialog box is displayed. This dialog box lists all of the widgets that have been added to your project's library.




***Custom Widgets dialog box***

2. Click **New**.

A *Custom Widget* dialog box is displayed for the new widget that you are creating.




***Custom Widget dialog box***

3. In the **Name** box, type a name for the widget.
   The name cannot contain any spaces.
4. In the **Height** and **Width** boxes, type the default height and width (in pixels) that the widget should have when it is added to a project screen.

   > **Note:** A custom widget, like most other screen objects, can be resized after it is inserted in a project screen.

5. To add a property to the widget, do the following:
   a) In the **Properties** area, click **Add**.
      The *Add* dialog box is displayed.
   b) In the **Name** box, type the name of the property.
   c) Click **OK** to add the property to the widget and then close the dialog box.
   d) Repeat for each property that you want to add.

   These are the basic properties that will be shared by all instances of the widget. To customize the properties on a specific instance of widget — that is, to associate different tags with the properties on that instance — you will need to configure the widget's object properties.
6. To add an event to the widget, do the following:
   a) In the **Events** area, click **Add**.
      The *Add* dialog box is displayed.
   b) In the **Name** box, type the name of the event.
   c) Click **OK** to add the event to the widget and then close the dialog box.

d) Repeat for each event that you want to add.

These are the basic events that will be shared by all instances of the widget. To customize the events on a specific instance of widget — that is, to attach different scripts to the events on that instance — you will need to configure the widget's object properties.

7. When you are done, click **OK** to create the widget.
The new widget is added to the list of widgets in the *Custom Widgets* dialog box. Also, the web files that actually make up the widget are automatically created in your project folder at:

***<project name>*\Web\Widgets*<widget name>***

8. If you want to immediately insert an instance of this widget, make sure it is selected in the list of widgets and then click **OK**.
The *Custom Widgets* dialog box is closed, and the widget is inserted in the project screen.
9. If you want to close the *Custom Widgets* dialog box without inserting an instance of this widget, click **Cancel**.

### Edit the web files for a custom widget

Edit the web files for a custom widget in order to develop the content of the widget and link the widget's properties and events.

Before you begin this task, you should be familar with how to develop webpages using HTML5, CSS, and JavaScript. Also, you must have already created the custom widget and added it to your project's library; the associated web files are automatically created in your project folder only after the widget is added to the library.

The files should be located at:

***<project name>*\Web\Widgets*<widget name>***

Each custom widget actually comprises three web files, but only two of the files are user-editable:

- `index.html` is the webpage itself. It is what is displayed within the widget's frame in the project screen. You may edit the entire body of the webpage (i.e., everything between **<body>** and **</body>**).
- `custom_widget.js` is the library of JavaScript functions that are associated with the webpage. You need to develop functions that link the widget's properties and events with the actual content and behavior of the webpage.

Do not edit the third file, *<widget name>*`.wjson`. It contains important settings for the custom widget.

Now, given the almost limitless ways in which you can develop an HTML5-compliant webpage, it is beyond the scope of this documentation to cover every possible step and option in editing these web files. Instead, the rest of this topic will feature a simple example of a custom widget that can load another webpage and then notify your project that the webpage was loaded. The widget — which you created earlier; see Create a new custom widget on page 301 — should have at least one property named **URL** and one event named **PageLoaded**.

To edit the web files for a custom widget:

1. Locate the widget's web files in your project folder.

   The files should be located at:

   ***<project name>*\Web\Widgets*<widget name>*\custom_widget.js**
   ***<project name>*\Web\Widgets*<widget name>*\index.html**

2. Use a text editor to open `index.html`.

   The default contents of **<body>** are a simple badge and label.

```
<!DOCTYPE html>
<html style="overflow: hidden;">
    <head>
        <script src="../Resources/Apis/Proxy.js" cwidget="MyWidget"></script>
```

```
        <script src="./custom_widget.js"></script>
        <title>MyWidget</title>
    </head>
    <body>
        <div style="width:96vw;height:95vh;background-color:white;text-
align:center;vertical-align:middle;line-height:98vh;border:solid;border-
width:thin;border-color:#e6e9eb">
            <div>
                <img src="../Resources/Images/HTML5.png"
 style="width:32px;height:32px"/>
                <p></p>
                <div style="height: 64px; top: 28px; width: 100%; position:
 absolute;">MyWidget</div>
            </div>
        </div>
    </body>
</html>
```

3. Delete the default contents of **<body>**.

```
<!DOCTYPE html>
<html style="overflow: hidden;">
    <head>
        <script src="../Resources/Apis/Proxy.js" cwidget="MyWidget"></script>
        <script src="./custom_widget.js"></script>
        <title>MyWidget</title>
    </head>
    <body>

    </body>
</html>
```

4. Insert your own HTML code into the body of the webpage.

   The entire body is displayed within the widget's frame in your project screen.

   In this example, you are inserting an **iframe** element that can be used to load other webpages. (The **iframe** element is a sort of browser window within the browser window, or in this case, a browser window within the custom widget.)

```
<!DOCTYPE html>
<html style="overflow: hidden;">
    <head>
        <script src="../Resources/Apis/Proxy.js" cwidget="MyWidget"></script>
        <script src="./custom_widget.js"></script>
        <title>MyWidget</title>
    </head>
    <body>
        <iframe id="myFrame" style="width: 100vw; height: 100vh;"></iframe>
    </body>
</html>
```

5. Save and close `index.html`.
6. Use the text editor to open `custom_widget.js`.

```
// Subscribes to receive property changes
cwidget.on("PropertyName", function() {

   // Gets property value
   console.log(cwidget.PropertyName);

   // Sets property value
   cwidget.PropertyName = "value"

   // Triggers an event
```

```
    cwidget.dispatchEvent("EventName");

});
```

This is an example of a JavaScript function that can be executed during project run time. **cwidget** is a JavaScript object that represents your custom widget, and it has various properties and methods associated with it. For example, **cwidget.on** is executed when the value of a property changes, **cwidget.PropertyName** (e.g., **cwidget.URL**) accesses the value of the property itself, **cwidget.dispatchEvent** notifies your project when an event has occurred, and so on.

You can copy this example as many times as you want, for however many properties and events you have added to your custom widget, and then develop each function to do something different. You can also develop entirely new functions and sub-routines, using your knowledge of JavaScript.

7. Delete the body of the function, so that you can insert your own commands.

```
cwidget.on("PropertyName", function() {

});
```

8. Replace **PropertyName** with the name of the property you added to your custom widget.

   In this example, you are using the **URL** property.

```
cwidget.on("URL", function() {

});
```

   Now this function will be executed whenever there is a change in the value of your widget's **URL** property (or more specifically, whenever there is a change in the value of the project tag associated with the **URL** property).

9. Develop your function.

   In this example, your function will load a webpage and then notify your project that the webpage was loaded.

```
cwidget.on("URL", function() {

    var myFrame = document.getElementById("myFrame");
    myFrame.onload = cwidget.dispatchEvent("PageLoaded");
    myFrame.src = cwidget.URL;

});
```

   The first line declares a new JavaScript object named **myFrame** and then links it to the **iframe** element that you inserted into your HTML file. The second line causes the **PageLoaded** event to be dispatched when a webpage is loaded. (In other words, the widget in your project is notified that the **PageLoaded** event occurred.) And the third line sets the source of **myFrame** to be equal to the widget's **URL** property, which causes the **iframe** element to load the specified URL.

   Remember, the entire function depends on the **cwidget.on** method, so whenever there is a change in the value of your widget's **URL** property, this function will be executed and a new webpage will be loaded.

10. Save and close `custom_widget.js`.

Of course there are many other things you can do with these web files, but that is beyond the scope of this documentation. More thorough descriptions and examples will be provided in future releases of this software. In the meantime, if you need help with developing your custom widgets, please contact your EcoStruxure Machine SCADA Expert 2020 software distributor.

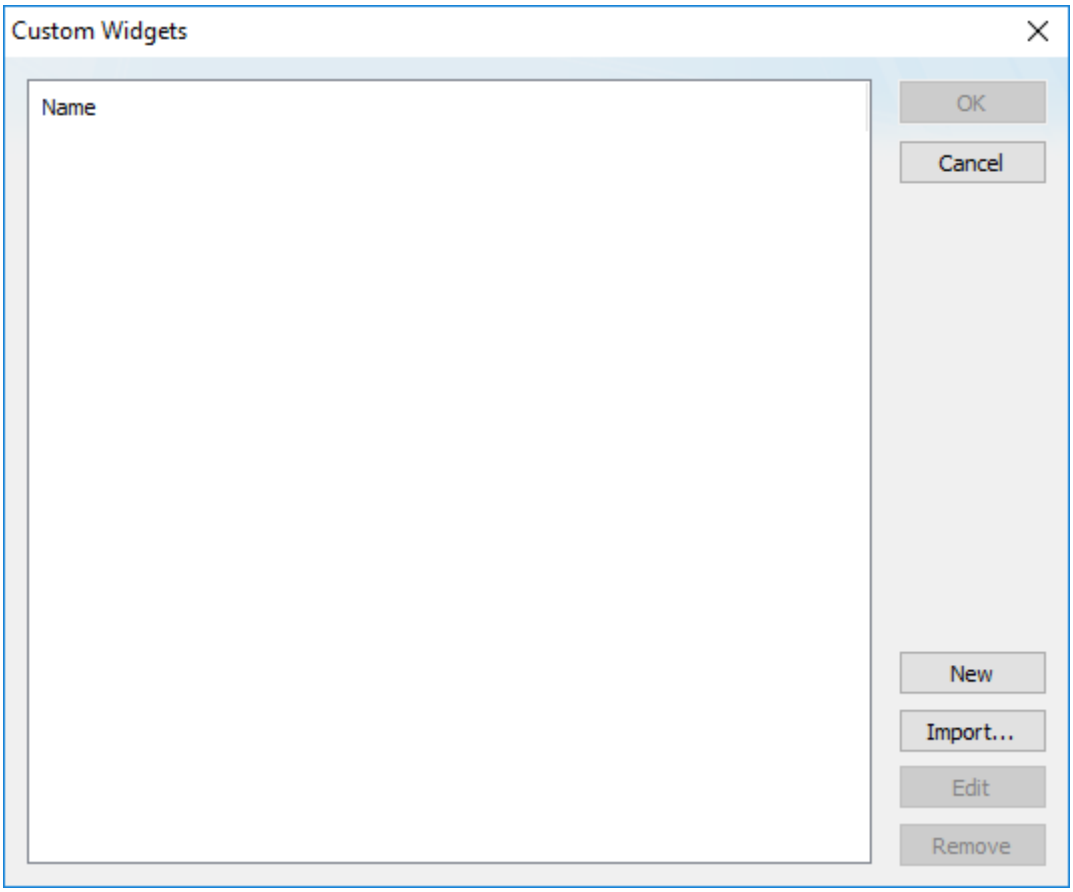
### Import a Custom Widget Package (CWP)

Import a Custom Widget Package — that is, a custom widget that has been packaged and distributed as a .cwp file — and add it to your project's library.

The project development software includes a number of premade widgets, and additional widgets might be made available by other sources. You can use any of these widgets in your project just as you would use widgets that you created yourself.

To import a Custom Widget Package:

1. On the **Draw** tab of the ribbon, in the **Libraries** group, click **Custom Widget**.
   The *Custom Widgets* dialog box is displayed. This dialog box lists all of the widgets that have been added to your project's library.




***Custom Widgets dialog box***

2. Click **Import**.
   A standard *Open* dialog box is displayed, and by default, it starts in the following location in your application folder:

   ```
   C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
    2020\Redist\CustomWidgetsLibrary
   ```

   This folder contains the premade widgets that are included with the project development software, but you can use the *Open* dialog box to navigate to any folder.

3. Locate and select the Custom Widget Package (.cwp file) that you want to import, and then click **Open**.

The custom widget files are extracted and copied to the following location in your project folder:

***<project name>*\Web\Widgets*<widget name>***

The custom widget is added to your project's library, and you can now insert it in a project screen. For more information about how to configure and use the custom widget, check the custom widget files in your project folder for a "help" or "readme" document.

For more information about how to package and distribute your own custom widgets, contact your EcoStruxure Machine SCADA Expert 2020 software distributor.

### Insert and configure a custom widget

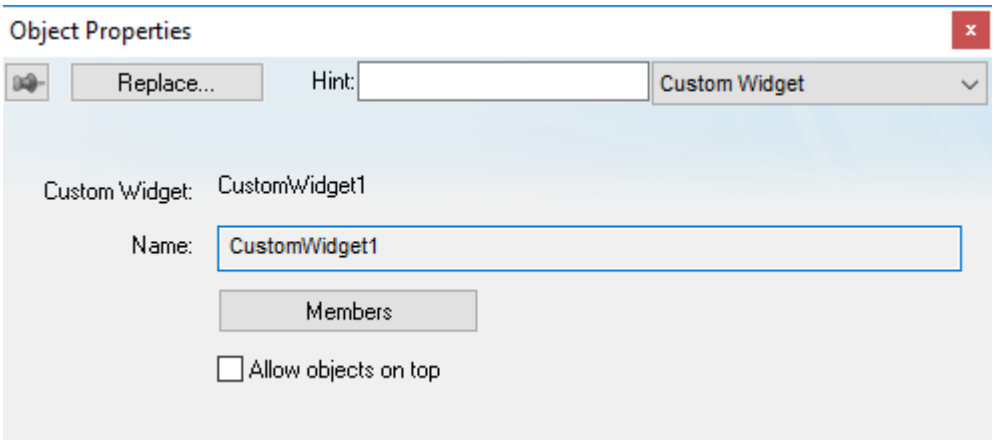
Use the **Custom Widget** command to select and insert an instance of custom widget in a project screen, and then configure the object properties of that instance.

Remember that all instances of a widget share the same basic properties and events, as you defined them when you created the widget and added it to your project's library, but you can customize these properties and events for each instance of the widget. Specifically, you can configure the object properties of an instance in order to associate tags with properties and attach scripts to events.

To insert and configure a custom widget:

1. Make sure you have the correct project screen open in the Screen/Worksheet editor.
2. On the **Draw** tab of the ribbon, in the **Libraries** group, click **Custom Widget**.
   The *Custom Widgets* dialog box is displayed. This dialog box lists all of the widgets that have been added to your project's library.
3. In the list, select the custom widget that you want to insert into the project screen, and then click **OK**.
   The dialog box is closed and the a new instance of the widget is inserted.
4. Use the standard screen editing tools to adjust the size and position of the widget, if necessary.
5. To open the widget's object properties, do one of the following:
   - Select the widget/object, and then on the **Draw** tab of the ribbon, in the **Editing** group, click **Properties**.
   - Right-click the widget/object, and then on the shortcut menu, click **Properties**.
   - Double-click the widget/object.

   The *Object Properties* dialog box for that widget is displayed. The name of the master widget is displayed in the **Custom Widget** box, and the name of this instance of the widget is displayed in the **Name** box.




***Object properties for a custom widget***

6. Click **Members**.
   The *Members* dialog box is displayed. It shows all of the properties and events that you defined when you created the custom widget and added it to your project's library.

7. Click the **Properties** tab, and then for each property in the list, type the tag/expression that should be associated with the property.

   Whenever the value of the tag/expression changes, it updates the value of the property in the widget's web files. Whenever the value of the property changes, it updates the associated tag.

   > **Tip:** You can double-click in the **Tag/Expression** box in order to open the Object Finder and use it to compose the tag/expression.

8. Click the **Events** tab, and then for each event in the list, attach a script:
   a) In the Script box, click the ... button.
   A standard script editor is displayed.
   b) Compose the script as you would in any other VBScript interface in Studio.

   > **Note:** Only VBScript is supported at this time.

   c) Click **OK** to save the script and close the script editor.

   Whenever the specified event is received from the widget's web files, it causes the script to be executed.

9. Click **OK** to close the *Members* dialog box.
10. Close the *Object Properties* dialog box.

### Configure the web server for custom widgets

If you use a web server to serve your project to thin clients, there is an additional step you must take in order to configure that web server for custom widgets.

Before you begin this task, you should be familiar with how to configure and run a web server, such as Internet Information Services for Windows.

Also, this task assumes that you have already configured your project for thin client access, that the web server is running, and that your remote users are using any of the standard thin clients. For more information, see Thin Clients and Mobile Access on page 727.

If you are only using the local Viewer module to view your project running on the same computer, you do not need to do anything and you may skip this task.

In short, when a user opens a project screen that contains a custom widget, the widget tries to load its web files from a specific URL. That URL can vary, depending on how you develop and deploy your project, so you must make sure the web server knows exactly where the web files are located on the server. To do that, you will create a direct link from your project's website to those web files.

To configure the web server for custom widgets:

1. From the Windows Control Panel, run **Administrative Tools** > **Internet Information Services (IIS) Manager**.
2. In Internet Information Services (IIS) Manager, right-click your project's website (see below), and then on the shortcut menu, click **Add Virtual Directory**.
   - If your remote users are using Thin Clients to access your project, you should have already configured **Default Web Site** so that its physical path (i.e., its root directory) points directly to your project folder. If this is true, right-click **Default Web Site**.
   - If your remote users are using Mobile Access to access your project, **Default Web Site** should contain a folder that was automatically created when you installed the Mobile Access Runtime software. For example, **Default Web Site** > **EMSE2020**. If this is true, right-click that folder.

   The *Add Virtual Directory* dialog box is displayed.
3. In the *Add Virtual Directory* dialog box, in the **Alias** box, type `CustomWidget`.
4. In the **Physical path** box, type the complete file path for your project folder, or click the browse button (...) on the right in order to open a standard Windows file browser that you can use locate and select the project folder.

For example:

```
C:\Users\<user name>\Documents\EcoStruxure Machine SCADA Expert 2020 Projects
\<project name>
```

5. Under **Pass-through authentication**, click **Connect as**.
   The *Connect As* dialog box is displayed.
6. In the *Connect As* dialog box, under **Path credentials**, select **Specific user**, and then to the right of the box, click Set.
   The *Set Credentials* dialog box is displayed
7. In the *Set Credentials* dialog box, type the user name and password of a Windows user on the computer that has permission to access the project folder.

   By default, the web server only has permission to access the files in the website's own physical path. This is to ensure that visitors to the website do not have unauthorized access to the rest of the computer. Therefore, if you create a virtual directory that points to a location outside of the website's physical path, you need to give the web server permission to access that location.
8. Click **OK** to close the *Set Credentials* dialog box and return to the *Connect As* dialog box.
   The name of the Windows user is displayed in the **Specific user** box.
9. Click **OK** to close the *Connect As* dialog box and return to the *Add Virtual Directory* dialog box.
10. Click **OK** to close the *Add Virtual Directory* dialog box.
    The virtual directory named **CustomWidget** is added to your project's website.
11. Exit Internet Information Services (IIS) Manager.

### Install the Custom Widget Framework on a client station

If your project screens include custom widgets, you might need to install Custom Widget Framework on some client stations to enable them to properly display the widgets.

This task applies only to stations on which you have already installed either Thin Client or Embedded HMI — in other words, stations that are using the Thin Client software to view your project.

Stations that are viewing your project through Mobile Access do not need to have Custom Widget Framework installed, because custom widgets are HTML5-based screen objects that can be displayed normally in the web browser.

Before you begin this task, you must have installed the full SCADA software on at least one Windows computer — typically, on your project development workstation — because doing so also unpacks the Custom Widget Framework installer. (Custom Widget Framework is not included in the Thin Client installer because it would greatly increase the file size of that installer, for a feature that not all projects use.)

You must have Administrator privileges on a computer or device in order to install any software.

To install the Custom Widget Framework on a client station:

1. Locate the Custom Widget Framework installer (`CustomWidgetFrameworkSetup.exe`) in your SCADA program folder.

   If SCADA was installed in its default location, the Custom Widget Framework installer should be located at:

   ```
   C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
    2020\Redist\CustomWidgetFramework\CustomWidgetFrameworkSetup.exe
   ```
2. Copy the installer to the client station, either over the network or on a portable hard drive.
3. Run the installer. You might need to do this as a user with Administrator privileges: right-click the installer, and then on the shortcut menu, click **Run as Administrator**.
4. Follow the installer's instructions. On the *Choose Destination Location* page of the installer, make sure the Bin folder of either Thin Client or Embedded HMI is selected. If it is not, click **Browse** and then use the file browser to locate and select the folder.

### Automatic resizing of custom widgets

Custom widgets can be automatically resized during project run time in order to fit changing contents.

When you create a new custom widget, you specify a default size for that widget. And then when you insert a Custom Widget object into a project screen, you can manually resize it like you would any other screen object. Given the nature of a custom widget, however — it is an embedded container for an external HTML file — the configured size of the widget might be too small to display the entire contents of the widget.

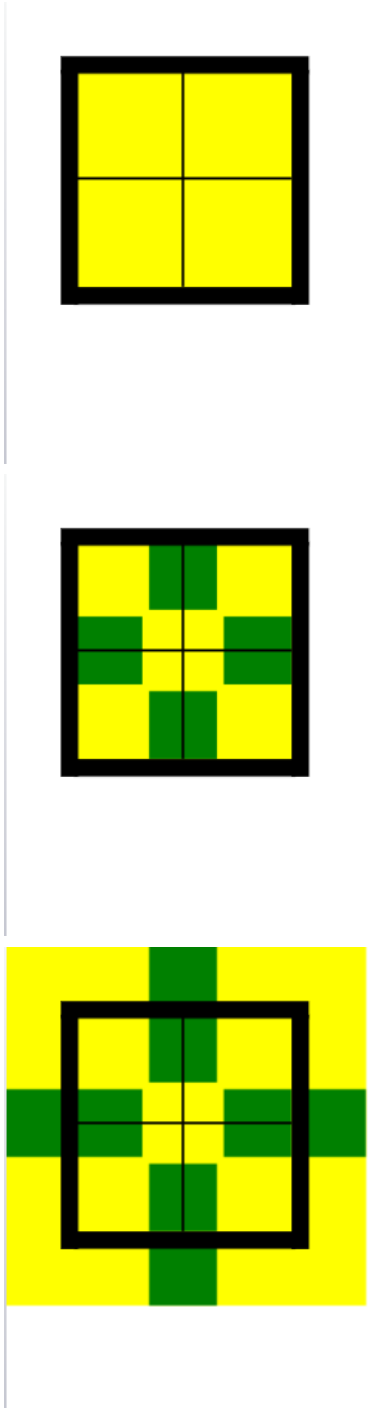
To handle this, automatic resizing of custom widgets has been implemented. When a screen is opened during project run time, each widget in the screen is immediately resized to fit the current contents of its associated HTML file. (The configured position of the widget does not change; its top-left corner remains the same as when you inserted the object in the screen.) Then, if/when the contents of the HTML file change, the widget is resized again to fit the new contents. This is especially useful if the HTML file has been designed so that it dynamically updates throughout the project run time.

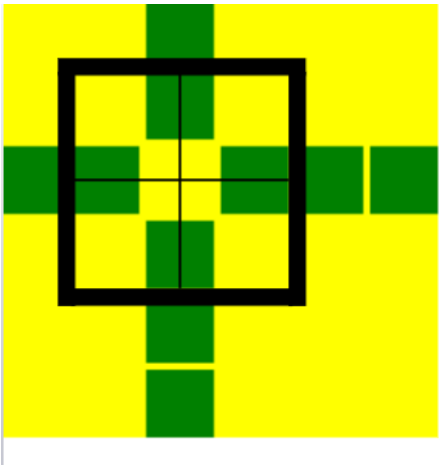
*Example of a "date picker" widget, before resizing*

*Example of a "date picker" widget, after resizing*

Please note the associated HTML file can be designed so that its contents change in size in any or all directions, not just to the bottom and right. The widget will still be resized to fit the contents, and this might appear to change the position of the widget but it does not; the actual position of the widget remains the same, as determined by its top-left corner, even though the contents appear to overflow it.

The following series of illustrations shows an example of a custom widget being resized as the contents change. The black frame represents the Custom Widget object as it was inserted into the project screen, and yellow-and-green pattern represents the contents of the associated HTML file.





Automatic resizing does not apply to custom widgets that are included in Linked Symbols. This is to prevent any disruption in the layout of the symbol.

Also, automatic resizing does not necessarily apply to custom widgets that are designed to display other webpages. This is because the webpages are typically displayed within an **iframe** element in the HTML file, and that element constrains the content of the webpages. The **iframe** element itself would have to change size in order to trigger the automatic resizing.

> **Tip:**
>
> By default, automatic resizing is enabled for all custom widgets. You can disable it on a widget-by-widget basis, however. To do this, edit the widget's associated JavaScript file (`custom_widget.js`) to include the following line:
>
> ```
> _proxy.autoResize = false;
> ```
>
> This change will apply to all instances of the same custom widget.
>
> For more information about editing `custom_widget.js`, see Edit the web files for a custom widget on page 304.

## LINK TO AN EXTERNAL IMAGE FILE

Use a Linked Picture screen object to link to an external image file, so that you can easily reuse the image in your project and/or change the image during run time.

This task assumes that you have a Screen worksheet open for editing.

Also, you must decide where exactly the image file will be stored:

- If you want the image file to be downloaded with the rest of the project files to the target system, it must be saved in your project folder at:

  **EcoStruxure Machine SCADA Expert 2020 Projects\<*project name*>\Web**

- If the image file will be located elsewhere on the network or target system, note the complete file path.

To link to an external image file:

1. On the **Draw** tab of the ribbon, in the **Libraries** group, click **Linked Picture**.
   A standard *Open* dialog box is displayed.
2. Use the dialog box to locate and select the image file, and then click **OK**.

   The following table shows which image file types are supported on each target platform:

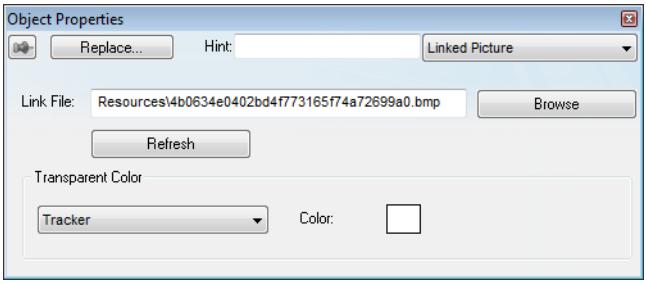
| File Type | Windows | Mobile Access |
|---|---|---|
| Windows Bitmap (.bmp) | # | # |
| Windows Metafile (.wmf, .emf) | # | |
| JPEG (.jpg, .jpeg) | # | # |
| PNG (.png) | # | # |
| TIFF (.tif, .tiff) | # | |
| GIF (.gif) | # | |
| AutoCAD DXF (.dxf), versions 10 and 11 only | # | |
| Exchangeable image file (.Exif), which is commonly used in digital cameras | | |

For maximum compatibility across all target platforms, you should use PNG files whenever possible.

The image is added to the worksheet as a Linked Picture screen object.

3. Double-click the screen object.

   The *Object Properties: Linked Picture* dialog box is displayed.




4. In the **Link File** box, examine the link.

   If the image file is located in the Web folder, the link is a relative file path. If the image file is located elsewhere, the link is an absolute file path.

   You can specify folders within the Web folder. For example, if you type `MyPictures\Picture1.bmp`, the project runtime will look for the image file at the following location:

   ***<project name>*\Web\MyPictures\Picture1.bmp**

> **Tip:**
>
> The file extension is not always required for the link to work. In projects that are configured to run on the Windows target platform, if no file extension is specified, `.png` is used by default. To change this, use a text editor to open your project file (`<project name>.APP`) and then edit the following property:
>
> **[Viewer]**
> **DefaultLinkedPictureExtension=*<image file extension>***
>
> In projects that are configured to run on the Embedded target platform, if no file extension is specified, .bmp is used by default. This cannot be changed.

5. If you want to change the link — and therefore change the picture — during project run time, replace the file path with a project tag:

   a) In the **Link File** box, select the file path, and then copy it to the clipboard.

b) Replace the file path in the **Link File** box with the name of a String tag in curly brackets (e.g., `{MyLinkedPicture}`).

c) If the tag does not exist, you will be prompted to create it. Make sure that you create it as a String tag.

d) Set the tag's startup value to be the file path that you copied to the clipboard, either by pasting the file path into the tag's Startup property (in the Project Tags Datasheet View) or by configuring the Startup Script to set the tag value when the project is run.

With a properly configured project tag, the link will be refreshed whenever the tag value changes during run time. Keep in mind that the tag value must have the same format as a normal link: a relative file path for a file located in the Web folder, or an absolute file path for a file stored elsewhere on the network or target system.

6. If you want some part of the picture to be transparent to the screen background, select a transparent color:
   a) In the **Transparent Color** group, select either **Color Code** or **Tracker** in the list.
   b) If you selected **Color Code**, type a tag/expression that will provide the 24-bit color code of the desired transparent color.
      For more information about Windows color codes, see Color Interface on page 74 and WdColor Enumeration.
   c) If you selected **Tracker**, click and drag the tracker on the screen object until it is positioned over a sample of the desired transparent color.
      The tracker is an additional handle on the screen object that initially appears just inside the bottom-right corner of the object. Moving the tracker on the object does not move or resize the object itself.

> **Note:** Transparent color is not supported for Windows Metafiles (.wmf, .emf)

7. Close the *Object Properties* dialog box.

Please note that if you enable performance control in the project settings, each image file will be cached in and then loaded from memory, if possible, rather than from its specified location in the project folder or on the network. This will improve run-time performance, because loading a file from RAM is faster than loading it from the hard drive. For more information, see Configure the performance control settings on page 112.

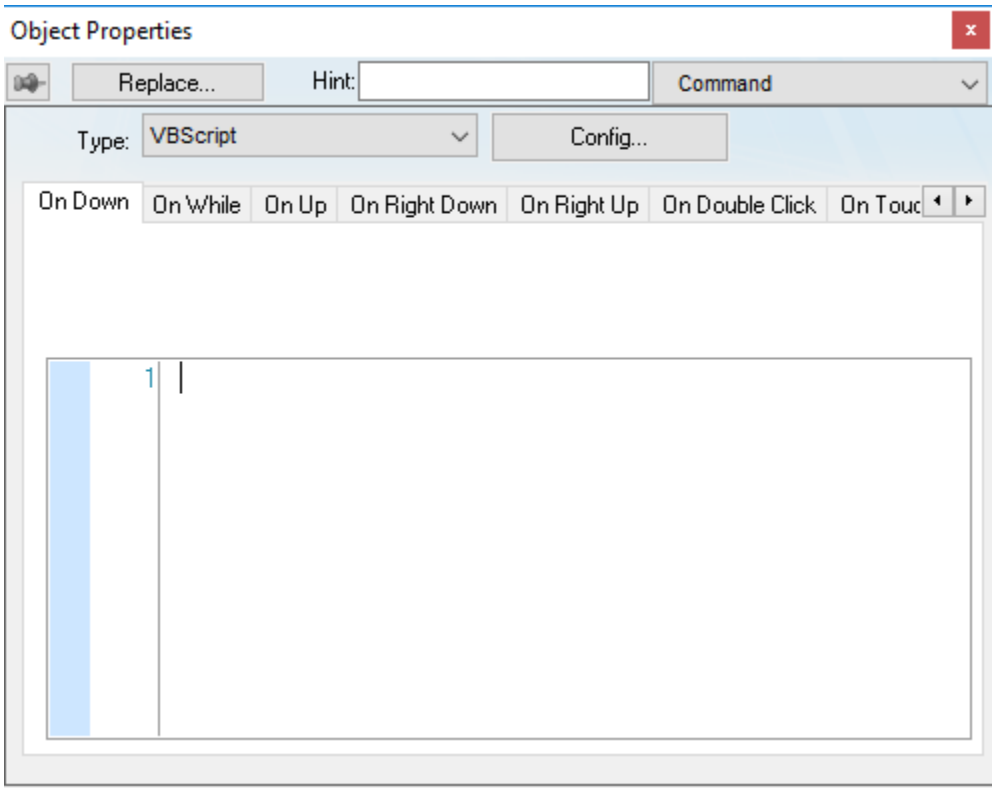
If you do not want the image files to be cached, however — in other words, if you want to ensure that the lastest versions of the files are always loaded from their specified locations — you should disable performance control. This is typically required when certain images (e.g., snapshots) are updated during project run time.

### *Applying animations to screen objects*

Use the *Animations* group to apply animations to a screen object or group of objects. Animations enable you to modify object properties on the fly (during runtime) according to tag values. Some animations also enable you to execute commands or insert values (set points) to the tags.

## COMMAND ANIMATION

On the **Graphics** tab, in the **Animations** group, click **Command** to add the animation to a selected object or group of objects. The animation enables you to click on the object or press a pre-defined key to execute the command at runtime. Double-click on the object to view its object properties.




***Object Properties: Command***

The Command animation provides one tag for each one of the events supported by it. Notice that more than one event can be configured simultaneously for the same Command animation:

- **On Down**: Executes the command/script once when the user clicks on the object with the left mouse button.
- **On While**: Keeps executing the command/script continuously while the mouse pointer is pressed on the object. The period (in milliseconds) of execution for the command/script is set in the Rate field from the Configuration dialog screen, except for the VBScript option, which is executed as fast as possible.
- **On Up**: Executes the command/script once when the user releases the left mouse button on the object.
- **On Right Down**: Executes the command/script once when the user clicks on the object with the right mouse button.
- **On Right Up**: Executes the command/script once when the user releases the right mouse button on the object.
- **On Double Click**: Executes the command/script once when the user double-clicks on the object with the left mouse button.
- **On Touch**, **On Touch Start**, **On Touch Delta**, **On Touch Complete**: These events are used for multi-touch gestures. For more information, see About Touch Events on page 364.

> **Note:** The events **On Right Down**, **On Right Up** and **On Double Click** are not supported in projects running on Windows Embedded target systems.

> **Tip:** An asterisk (*) on an event tab indicates that something is configured for that event. This makes it easier to see at a glance which events are configured.

**Type** menu: This setting defines the type of action that must be executed by the event of the Command animation. Notice that each event has its own type. Therefore, the same Command animation can be configured with different types of action for different events. The following types are supported:

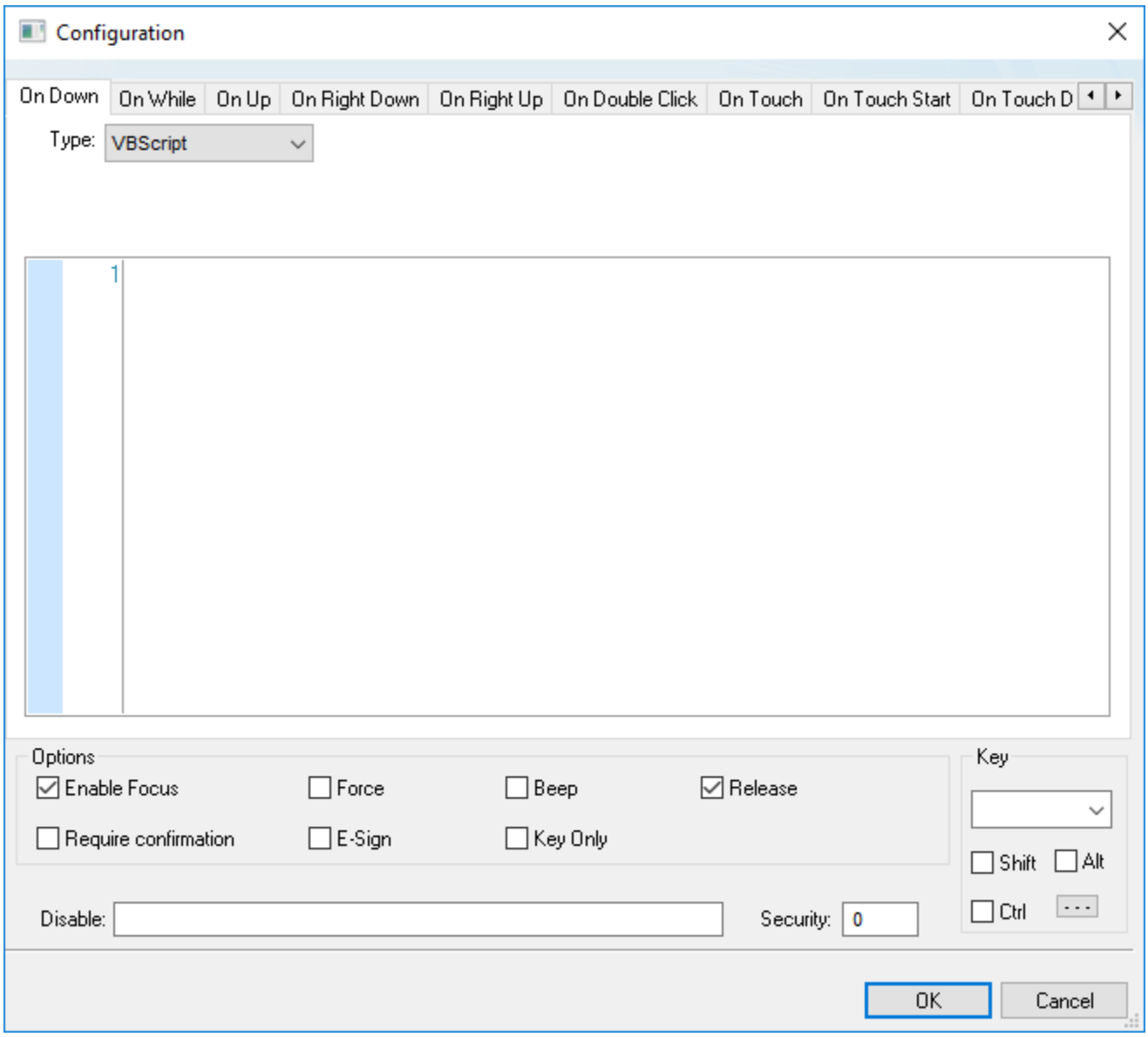
| Type | Description |
|---|---|
| **Built#in Language** | Allows you to configure a script using the EcoStruxure Machine SCADA Expert built-in language. When this type is selected, the user can configure up to 12 expressions for each event in the **Expression** column. The expressions are executed sequentially from the first row until the last one when the event is triggered. The result of each expression is written to the tag configured in the **Tag** column (if any). Consult the **Built-in Scripting Language** chapter for more information. |
| **VBScript** | Allows you to configure a script using the standard VBScript language. When this type is selected, the user can configure a script in the VBScript editor for the **Command** animation. Consult the **VBScript** chapter for further information about the VBScript language. |
| **Open Screen** | Allows you to configure the **Command** animation to open a specific screen when the event is triggered during runtime. This type is equivalent to the `Open` function. You can either type the screen name in the **Open Screen** field or browse it. Furthermore, you can type a string tag between curly brackets {TagName} in this field. When the event is executed, the project will attempt to open the named screen.<br>**Note:** The screen file extension (either `*.scc` or `*.scr`) is assumed, so you do not need to include it. However, if you have two screen files with the same name but different extensions in your project folder (e.g., `MyScreen.scc` and `MyScreen.scr`), the one with the preferred extension — as determined by whether the **Use .scr extension for screen files** option in the project settings is selected — will be opened. For more information, see Viewer tab on page 117. |
| **Close Screen** | Allows you to configure the **Command** animation to close a specific screen when the event is triggered during runtime. This type is equivalent to the `Close` function. You can either type the screen name in the **Close Screen** field or browse it. You can also type a string tag between curly brackets {TagName} in this field. When the event is executed, the project will attempt to close the named screen. |
| **Set Tag** | Allows you to configure the **Command** animation to set a tag when the event is triggered during runtime. You can either type the tag name in the **Set Tag** field or browse it. When the event is executed, the project will write the value 1 to the tag configured in this field. |
| **Reset Tag** | Allows you to configure the **Command** animation to reset a tag when the event is triggered during runtime. You can either type the tag name in the **Reset Tag** field or browse it. When the event is executed, the project will write the value 0 to the tag configured in this field. |
| **Toggle Tag** | Allows you to configure the **Command** animation to toggle a tag when the event is triggered during runtime. You can either type the tag name in the **Toggle Tag** field or browse it. When the event is executed, the project will toggle the value of the tag configured in this field. |

**Config** button: Launches the *Configuration* dialog, where the Command animation can be fully configured.

**Back to button**: Click to go back to the object properties of the underlying Button object.

### Configuration dialog

This dialog allows you to fully configure the Command animation…




***Configuration dialog***

The event tabs (e.g., On Down, On While, etc.) and the **Type** menu are the same as in the *Object Properties* dialog described above. The remaining settings are shared for all events:

- *Options* pane:
  - **Enable Focus** checkbox: When this option is checked, the object that the Command animation was applied to can receive the focus during runtime by the navigation keys.
  - **Force** checkbox: When this option is selected, any project tag that receives a value will trigger events as if the tag changed, even if the new value is equal to the old value. For example, if a tag has a value of 0 and the Command animation runs a procedure that writes 0 to that tag, all other tasks in the project runtime will recognize that the tag changed, even though it did not. This option is useful for making sure that events triggered by tag changes (e.g., **Write on Tag Change** on a communication driver) are always triggered when the Command animation is used.

Please keep in mind that if the tag's value does not actually change, the tag's timestamp (***tagname*->Timestamp**) is not updated either.

**Force** applies to both the procedure run by the Command animation itself and any global procedures called in that procedure, as long as they are run on the project runtime client where the Command animation is used (i.e., on the device where the button is pushed).

**Force** does not apply to global procedures that are run on the project runtime server using the function RunGlobalProcedureOnServer, even if the function is called in the procedure run by the Command animation. If you want to force tag changes in global procedures run on the server, use the function ForceTagChange.

- **Beep** checkbox: When this option is checked, a short beep is played when the Command is executed. This option is useful to provide an audio feed-back to the user, indicating that the Command was executed. It does not indicate, however, if the action triggered by the Command animation was successful or not.
- **Release** checkbox: When this option is checked, the On Up event is executed when you drag the cursor (or your finger) out of the object area (whether the button was released or not). This option is useful to make sure that the On Up event will always be executed after an On Down event, even if the user releases the mouse cursor out of the object area before releasing it.
- **Confirm** checkbox: When this option is checked, user will have to answer a confirmation question before executing the command. This option is useful for decreasing the accidental triggering of critical events during runtime.
- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the command.
- **Key Only** checkbox: When this option is checked, the user can *only* use the keyboard shortcut (configured in the *Key* pane described below) to execute commands.

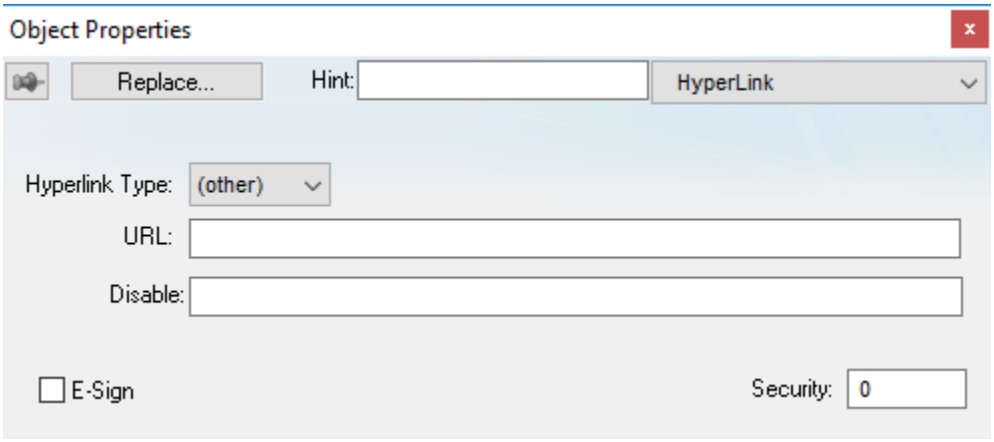
- **Disable**: Disables action by the user when the result of the expression configured in this field is TRUE (value different from 0).
- **Security**: Security access level required to use the Command animation.
- *Key* group: Shortcut used to trigger the events On Down, While Down and On Up using a keyboard. (In other words, pressing this keyboard shortcut is the same as clicking the left mouse button.) This option is especially useful when creating projects for runtime devices that do not provide a mouse or touch-screen interface — the keyboard is the only physical interface available to interact with your project during runtime.
  - **Shift**, **Ctrl**, or **Alt** boxes: Click to create a key combination key, meaning the Shift, Ctrl and/or Alt key must be pressed with the key specified in the drop-down list.
  - Click the browse button (...) to open the *Key Modifer* dialog, which enables you to modify your combination keys. You can choose **Left**, **Right** or **Left or Right** to specify the position on the keyboard of the Shift, Ctrl or Alt key in the key combination. If you choose **Left or Right**, the command will be executed any time either of these keys is pressed in combination with the key specified in the drop-down list.

> **Tip:** If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects.

## HYPERLINK ANIMATION

On the **Graphics** tab, in the **Animations** group, click **Hyperlink** to add the animation to a selected object or group of objects. Applying this animation allows you to click on the object(s) during execution to launch the default browser and load the specified URL.

Double-click on the object to open the *Object Properties* dialog.




***Object Properties: Hyperlink***

You can use this dialog to specify the following parameters:

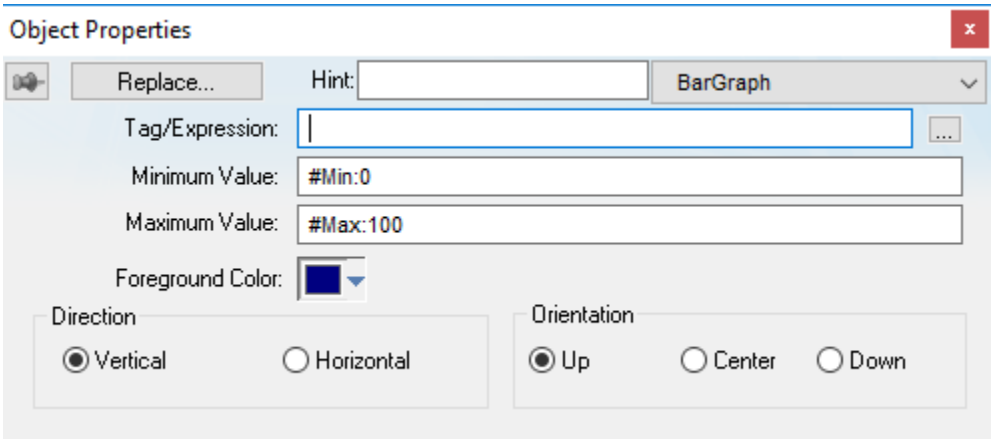
- **Hyperlink Type** combo-box: Click the combo-box button to select a URL protocol from the list. The project uses this protocol when it loads the URL.
- **URL** field: Type the URL address you want to load.

> **Tip:** You are not required to enter the protocol type in the URL field. When you select a protocol type from the **Hyperlink Type** list, the project automatically adds the protocol's prefix to the URL address.

- **Disable** field: Type a value greater than zero into this field to disable the hyperlink Command animation for the selected object(s).
- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the animation.
- **Security** field: Type a value into this field to specify a security level for the object(s). If a user logs on but does not have the required security level, the project disables the hyperlink command for the object(s).

## BARGRAPH ANIMATION

On the **Graphics** tab, in the **Animations** group, click **Bargraph** to add bar graph properties to a selected object, then double-click on the object to open the *Object Properties* dialog.




***Object Properties: BarGraph***

Use the *Object Properties* dialog to specify the following parameters:

- **Tag/Expression** field: Type a tag or expression that evaluates the bar graph level. You also can click the icon to browse your directories for an existing tag or expression.
- **Minimum Value** field: Type a numeric constant or a tag value into this field to define the minimum value used to calculate the height (if vertical) or width (if horizontal) of the bars.
- **Maximum Value** field: Type a numeric constant or a tag value into this field to define the maximum value used to calculate the height (if vertical) or width (if horizontal) of the bars.

  If you do not specify a value for this field, the application opens a dialog requesting you confirm creation of the tag.

> **Tip:** The application also allows you to enter constants in tag/numeric value fields. Constant values (defined by the # character) are equivalent to numeric values, except that constants display in the *Tag Replace* dialog. You may find constants useful for documentation purposes or for creating generic objects.
>
> For example: **`#Name:100`**.
>
> Where the value (**`100`**) following the semicolon ( **`:`** ) is the constant, and **`Name`** is a constant mnemonic only and not added to database.

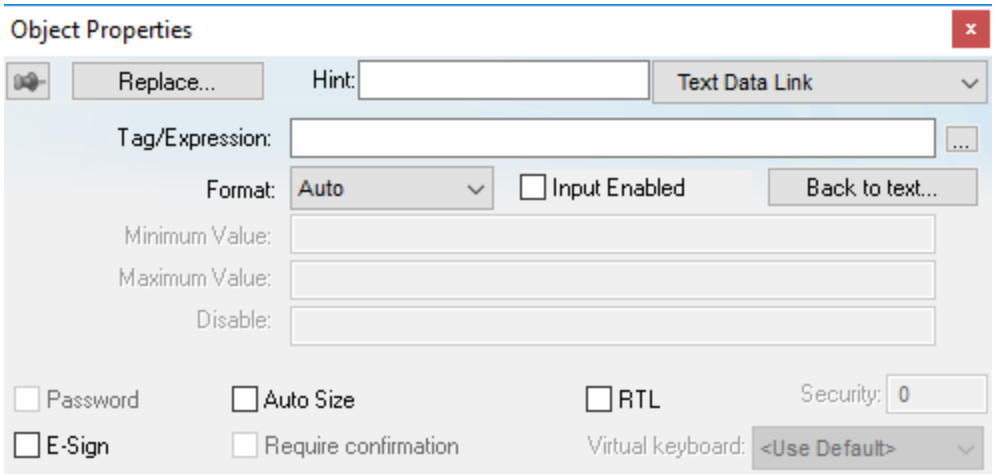
- **Foreground Color**: To specify a fill color for the bars, click the combo-box button. When the *Color* dialog displays, click on a color to select it, and then close the dialog.
- **Direction** area: Click the **Vertical** or **Horizontal** radio button to specify the direction of the bar graph.
- **Orientation** area: Click the **Up**, **Center**, or **Down** radio button to specify the orientation of the maximum and minimum values when drawing the bars.

## TEXT DATA LINK ANIMATION

On the **Graphics** tab, in the **Animations** group, click **Text Data Link** to add the animation input or output text property to a selected Text object. Applying the **Text Data Link** property allows you to insert and display tag values in real time if you are using the keyboard or on-screen keypad to run a project.

> **Note:** You can only apply this animation to Text objects that include one or more **#** characters. Each **#** represents one character of input/output. You can combine **#** characters with regular text in the same Text object — for example, **`MyLabel #####`** or **`$###.##`**.
>
> It's important to remember that the runtime project will always display the most significant digits of a numeric value, regardless of the number or placement of **#** characters in the text. That means if you do not have sufficient **#** characters to display the value, then it will be transformed in some way depending on the format of the value (as set by the **Fmt** option described below):
>
> - In Decimal format, the number of decimal places is determined by the position of the decimal separator in the **`###`** text. However, if you do not have enough **#** characters to the left of the decimal separator to display the whole value, then the whole value will overrun the fractional value. For example, if you try to display a value of 112.64 in **`#.##`**, you will see **`112`**.
> - In Hexa and Binary formats, if you have more **#** characters than you need to display the value, then the runtime project will fill in with leading zeroes. If you have less characters than you need, then the value will simply be truncated.
> - In Auto format, the runtime project will ignore the number of **#** characters and display the entire numeric or string value. Numeric values will be displayed in decimal format with their complete whole and fractional values, regardless of the placement of the decimal separator in the **`###`** text. Given an exceptionally large value or long string, this may disrupt the layout of your screens.

Double-click on the object to open the *Object Properties* dialog. You can use this dialog to specify the following parameters:




***Object Properties: Text Data Link***

- **Tag/Expression** text field: Type one of the following into the field:
  - The name of a tag on which to perform an input or output operation; or
  - An expression on which to perform an output operation only.

  You can also click the browse button ... to open the *Object Finder* to find an existing tag or expression.

> **Note:** If the configured tag/expression is invalid, then during runtime, the placeholder characters (**###**) will be displayed instead.

- **Format** combo-box: Click to select how the numeric value (if any) of the specified tag or expression will be formatted and displayed on-screen. Available options include **Decimal**, **Hexa** (i.e., hexadecimal), **Binary** and **Auto**. If you select **Auto**, then the value will be formatted according to the virtual table created by the `SetDecimalPoints` function.

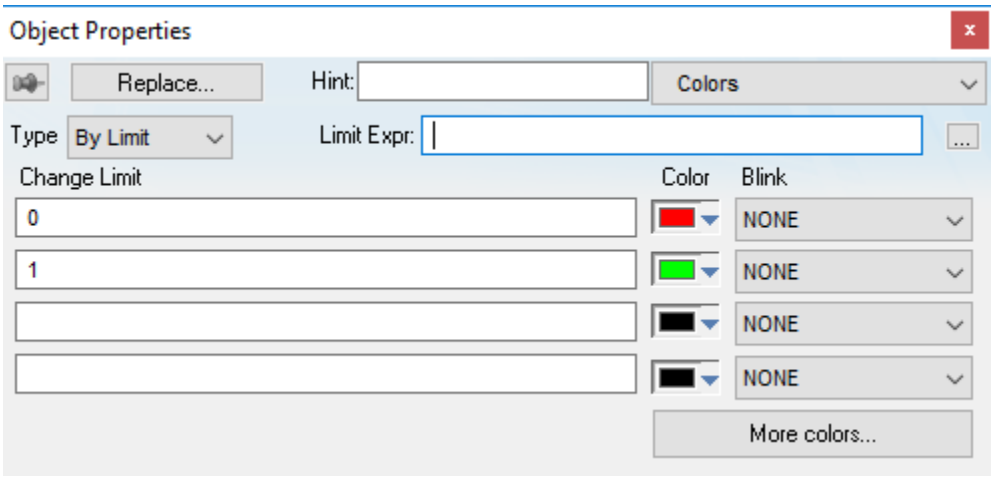
  This option does not actually change the specified tag or expression in any way. For example, **Tag/Expression** is set to a tag of Integer type, **Input Enabled** is checked, and **Fmt** is set to **Hexa**. You may input a new value in hexadecimal format, but it is saved in your project database as an integer.
- **Input Enabled** checkbox: Click (check) this option to allow user input to the specified tag. Disable (uncheck) this option to only display the output from the specified tag or expression.
- **Back to text**: Click to go back to the object properties of the underlying Text object.
- **Minimum Value** field: Enter a minimum value for the tag associated with this Text object. A user will not be permitted to input a number lower than this value.
- **Maximum Value** field: Enter a maximum value for the tag associated with this Text object. A user will not be permitted to input a number greater than this value.
- **Password** checkbox: Click (check) this option to hide password text entries by replacing the text with asterisks (*****).
- **Confirm** checkbox: Click (check) this option to require users to confirm any new values set during runtime.
- **Auto Size** checkbox: Click (check) this option to automatically resize the Text object to fit the output. This option is not available if **Input Enabled** is checked (see above).
- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before changing the tag value.
- **VK**: Virtual Keyboard type used for this object. You need to select the Virtual Keyboard option in the *Viewer* settings (**Viewer** on the Project tab of the ribbon) before configuring the Virtual Keyboard for this interface.
- **Disable** field: Type a value greater than zero in this field to disable the tag's data input property.

- **Security** field: Type a value in this field to specify the security level for a specific data input object (as defined in the Security section).

## COLOR ANIMATION

On the **Graphics** tab, in the **Animations** group, click **Color** to add the animation to a selected object. The Colors animation allows you to modify the color of a static object during runtime based on the value of a tag or expression.

Double-click on the object to open the Object Properties dialog.




***Object Properties: Colors***

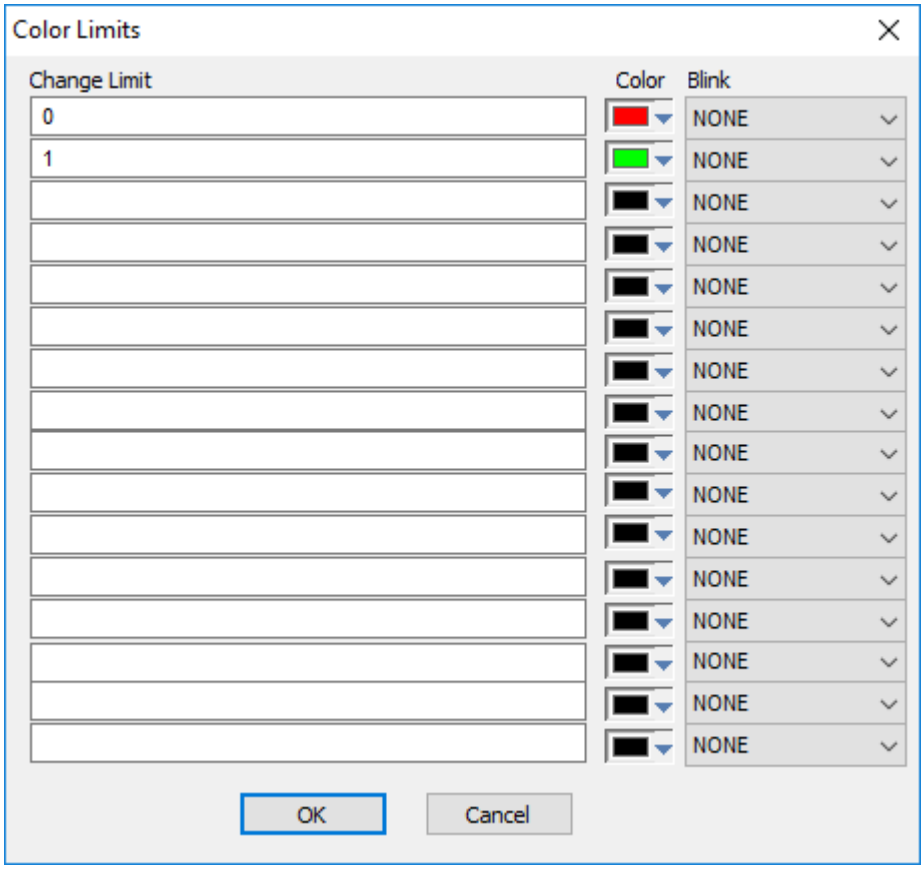
You can use this dialog to specify the following parameters:

- **Type**: Determines the mode in which this animation works:
  - **By Limit**: When selecting this type, you can specify up to four limits (Change Limit) for this animation and a color for each limit. When the value of the tag or expression configured in the Tag/Expr field reaches the limits, the color associated with the respective limit is applied to the object.
  - **By Color**: When selecting this type, you can specify the code of the color that must be applied to the object directly in the Tag/Expr field. Using this code, you can apply any color supported by your device to the object.

> **Tip:** You can configure the `RGBColor` function in the Tag/Expr field when **Type** = By Color. This allows you to configure the color by its RGB codes. See Color Interface for a table with the codes for the most commonly used colors.

- **Tag/Expression** field: Type the name of a tag or expression you want to monitor. When Type = By Limit, EcoStruxure Machine SCADA Expert compares the result of the tag/expression with the specified Change Limits to determine the proper color for the selected object. When Type = By Color, the result of this field sets the color that will be applied to the object.
- **Change Limit** field: Type a limit value (a numeric constant or tag) for the color change. The numbers must be configured in ascendant order according to the following sequence of the fields displayed on the Object

Properties dialog: Upper left, lower left, upper right and lower right field. If you click on the More button, you can configure up to 16 different limits for the color animation.




***Color Limits dialog***

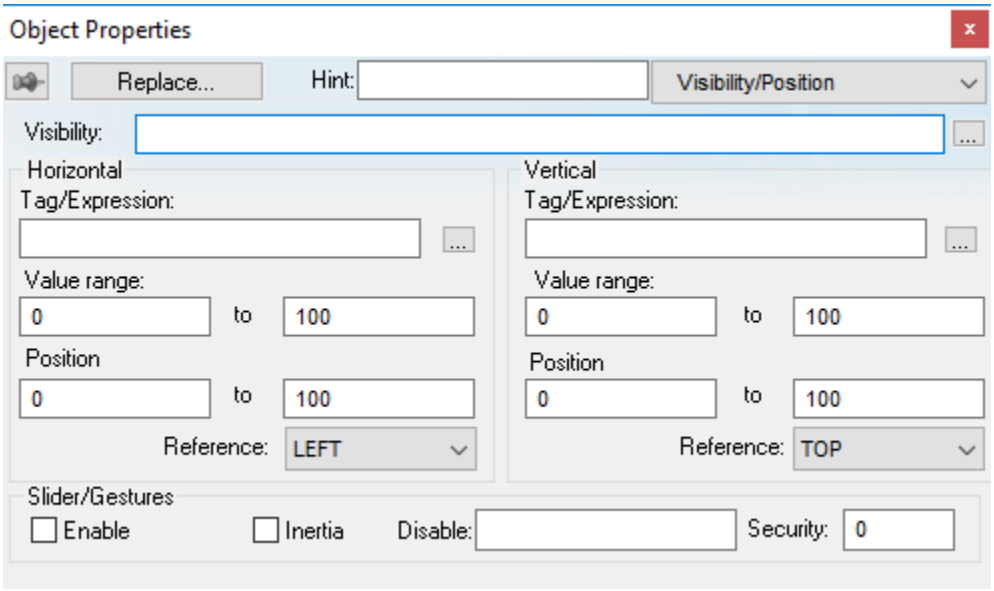
- **Color** combo box: Click the combo-box button to associate a color with each color change limit. When the Color dialog opens, click a color to select it, and then close the dialog.
- **Blink** combo-box: Click the combo-box button to specify whether the color change will blink, and how fast it will do so.

> **Note:** The following fields are automatically disabled (grayed out) when **Type** = By Color: **Change Limit**, **Color** and **Blink**.

## VISIBILITY/POSITION ANIMATION

The Visibility/Position animation allows you to move an object horizontally and/or vertically during run time.

On the **Graphics** tab, in the **Animations** group, click **Visibility/Position** to add the animation to an object. Double-click on the object to open its *Object Properties* dialog.




***Object Properties: Visibility/Position***

Use the dialog to configure the following properties:

- **Visibility** box: Configure a tag/expression in this box to control the visibility of the object. When the value of the tag/expression is 0 (FALSE), the object is hidden, and when the value is non-zero (TRUE) or the box is left empty, the object is visible.

  Objects that are hidden cannot be clicked/tapped and therefore cannot execute any Command animations applied to them.

  For some types of screen objects (i.e., all Shapes, Standard-style Buttons, and Linked Pictures), **Visibility** controls not just the visibility but the opacity of the object, and the value (from 0 to 1) of the tag/expression determines the percentage of opacity. For example, a value of 0.8 would give the object 80% opacity. The value can change during run time, so you can use it to make objects appear to fade in and out. Please note that you must have **Enable Enhanced Graphics** selected in the project settings in order to use this feature. For more information, see Project Settings: Viewer.

- **Horizontal** and **Vertical**: Configure these settings to determine how the object moves in the screen:

  - **Tag/Expression** boxes: Configure a tag/expression that will determine the position of the object during run time; as the value changes, the object is moved in the screen. Whether you can configure a tag or an expression depends on whether the **Slider/Gestures** option (see below) is selected:

    - If the **Slider/Gestures** option is not selected, then configure either a tag or an expression in this box.
    - If the **Slider/Gestures** option is selected, then configure only a project tag (Integer or Real type) in this box. When the end user manually moves the object, the new value is written back to the tag.

    For the horizontal position, the value increases as the object moves to the right and it decreases at the object moves to the left. For the vertical position, the value increases as the object moves to the bottom and it decreases as the object moves to the top.

  - **Value range** boxes: Enter the minimum and maximum values for the tag/expression. If the actual value goes outside of its range, then the value is ignored and the limit is used instead.

  - **Position** boxes: Enter values to specify how far (in pixels) the object can move from its starting position. The starting position is equal to "0,0". Values greater than 0 allow the object to move right and down, and values less than 0 allow the object to move left and up.

During run time, the object's position is proportional to the tag/expression value within its range. For example, if **Position** is `0` to `100` and **Range** is `0` to `10`, then each increment in the value will move the object 10 pixels. This is true for both **Horizontal** and **Vertical**.

- **Reference** drop-down lists: Select a reference point on the object. The following table shows how your selections for **Horizontal** and **Vertical** work in combination:

| | LEFT | CENTER | RIGHT |
|---|---|---|---|
| TOP | | | |
| CENTER | | | |
| BOTTOM | | | |

This reference point is meaningful only if you have the Resize animation added to the same object. The position of the object is always based on this reference point, regardless of the size or shape of the object.

- **Slider/Gestures**: When multi-touch gestures are enabled, the end user can use one- or two-finger "slide" gestures to move this object during run time. The changes in position are written back to the project tags configured in the **Tag/Expression** boxes above.
  - **Enable** option: Select to enable gestures on this specific object.

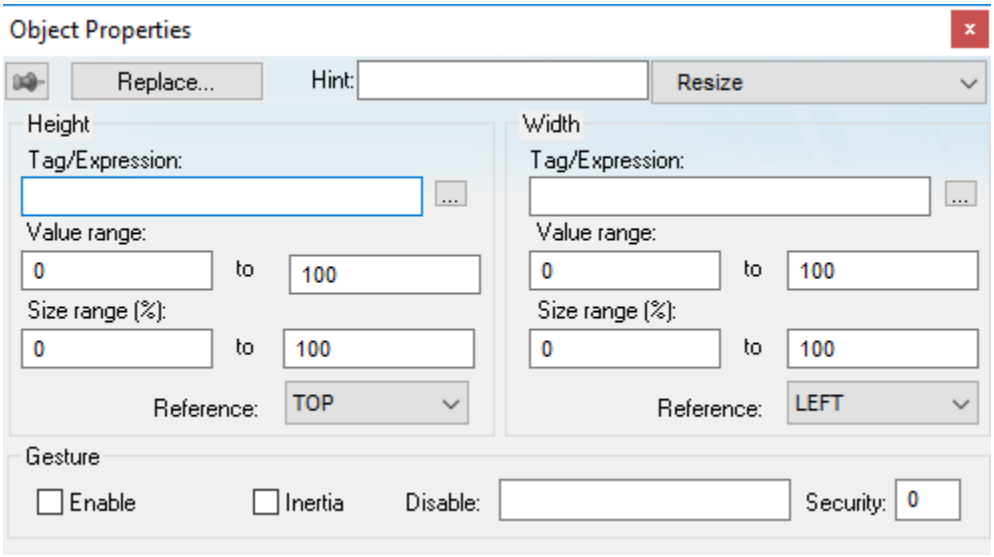
    Please note that Multi-Touch must also be enabled for the project and screen.
  - **Inertia** option: Select to apply inertia to this object, so that it slows down naturally rather than stops abruptly when the end user stops touching it.
  - **Disable** box: Configure a tag/expression. When its value is TRUE (i.e., not 0), then gestures are disabled on this object.
  - **Security** box: Type the minimum security level that the end user must have to use gestures on this object.

  For more information, see Multi-Touch on page 351.

## RESIZE ANIMATION

The Resize animation allows you to increase or decrease the size of an object during runtime.

On the **Graphics** tab, in the **Animations** group, click **Resize** to add the animation to an object. Double-click on the object to open its *Object Properties* dialog.

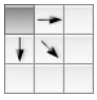
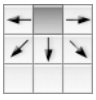
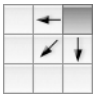



***Object Properties: Resize***

Use the dialog to configure the following properties:

- **Height** and **Width**: Configure these settings to determine how the object moves in the screen:
  - **Tag/Expression** boxes: Configure a tag/expression that will determine the size of the object during run time; as the value changes, the object is resized in the screen. Whether you can configure a tag or an expression depends on whether the **Gesture** option (see below) is selected:
    - If the **Gesture** option is not selected, then configure either a tag or an expression in this box.
    - If the **Gesture** option is selected, then configure only a project tag (Integer or Real type) in this box. When the end user manually resizes the object, the new value is written back to the tag.
  - **Value range** boxes: Enter the minimum and maximum values for the specified tag(s). If a tag's actual value goes outside of its range, then the value is ignored and the limit is used instead.
  - **Size range (%)** boxes: Enter the minimum and maximum values for the size of the object. The minimum value can be as low as 0% (making the object effectively invisible), and the maximum value can be as high as you want. 100% is the original size of the object when you draw it in the screen worksheet, 200% is double the original size, and so on.

    During run time, the object's size is proportional to the tag value within its range. For example, if **Size range (%)** is `0` to `100` and **Value range** is `0` to `10`, then each increment in the value will increase the object size by 10%. This is true for both Height and Width.
  - **Reference** drop-down lists: Select a reference point to determine the directions in which the object will change size. The following table shows how your selections for Height and Width work in combination:

| | LEFT | CENTER | RIGHT |
|---|---|---|---|
| TOP |  |  |  |

| | | | |
|---|---|---|---|
| CENTER | | | |
| BOTTOM | | | |

- **Gesture**: When multi-touch gestures are enabled, the end user may use two-finger "pinch" and "stretch" gestures to resize this object during run time. The changes in size are written back to the project tags configured in the **Tag/Expression** boxes above.
  - **Enable** option: Select to enable gestures on this specific object.

    Please note that Multi-Touch must also be enabled for the project and screen.
  - **Inertia** option: Select to apply inertia to this object, so that it slows down naturally rather than stops abruptly when the end user stops touching it.
  - **Disable** box: Configure a tag/expression. When its value is TRUE (i.e., not 0), then gestures are disabled on this object.
  - **Security** box: Type the minimum security level that the end user must have to use gestures on this object.

  For more information, see Multi-Touch on page 351.

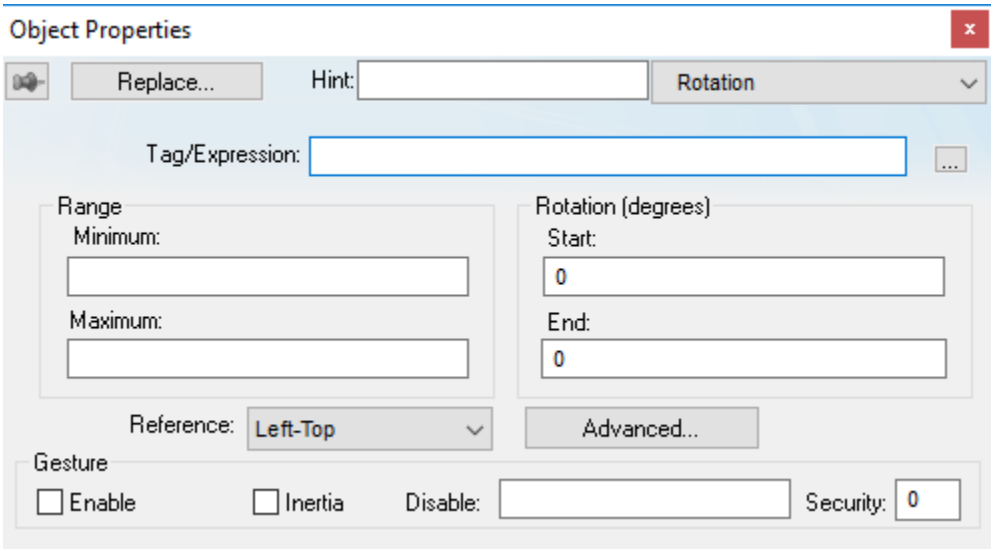
## ROTATION ANIMATION

Use the Rotation animation to rotate screen objects.

On the **Graphics** tab, in the **Animations** group, click **Rotation** to add the animation to a Line, Open Polygon, Closed Polygon, Bitmap, or Linked Picture object.

> **Note:**
>
> The Rotation animation does not work in a group of objects. If the animation is added to an object and then that object is grouped with others, it will be disabled.
>
> Also, the Rotation animation cannot be added to Bitmap and Linked Picture objects in projects running on Windows Embedded devices. For more information, see About target platforms, product types, and target systems on page 100.

After the animation is added, double-click the object to open the *Object Properties* dialog box.




***Object Properties: Rotation***

Use this dialog box to edit the following properties:

- **Tag/Expression** box: Specify a tag/expression that will determine the angle of the object during run time; as the value changes, the object is rotated in the screen. Whether you can specify a tag or an expression depends on whether the **Gesture** option (see below) is selected:
  - If the **Gesture** option is not selected, specify either a tag or an expression in this box.
  - If the **Gesture** option is selected, specify only a project tag (Integer or Real type) in this box. When the end user manually rotates the object, the new value is written back to the tag.
- *Range* area: Type the **Minimum** and **Maximum** values for **Tag/Expression**.
- *Rotation (degrees)* area: Type the **Start** and **End** positions (in degrees) of the object. The actual rotation is proportional to the value of **Tag/Expression** within *Range* — in other words, the **Start** position is equal to the **Minimum** value, and the **End** position is equal to the **Maximum** value. An object can rotate up to 360 degrees, and it rotates clockwise by default.

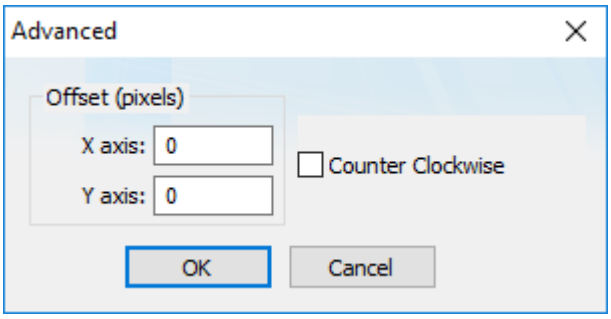
  For example, a Rotation animation has the following settings: **Minimum** is 0, **Maximum** is 100, **Start** is 0, and **End** is 180. If the current value of **Tag/Expression** is 50 (i.e., halfway between Minimum and Maximum), then the actual rotation of the object is 90 degrees (i.e., halfway between Start and End). A value of 25 is equal to 45 degrees, a value of 75 is equal to 135 degrees, and so on.

  Values less than **Minimum** and greater than **Maximum** are not ignored. Instead, they continue the rotation at the same proportional rate. Given the same settings as in the previous example, a value of -50 is equal to -90 degrees, a value of 150 is equal to 270 degrees, and so on.

  If you actually need to limit the value of **Tag/Expression**, reconfigure it to include the `Min` and `Max` functions.
- **Reference** combo-box: Select one of the following as a pivot point on which to rotate the object:
  - **Left-Top**: Upper-left corner of the object.
  - **Left-Bottom**: Lower-left corner of the object.
  - **Center**: Center of the object.
  - **Right-Top**: Upper-right corner of the object.
  - **Right-Bottom**: Lower-right corner of the object.

  You can fine tune the pivot point by configuring the *Offset* settings described below.

- **Advanced** button: Click to open the *Advanced* dialog box, where you can edit the following settings:




***Object Properties: Rotation – Advanced***

  - *Offset (pixels)* area: Enter the number of pixels by which to offset the **Reference** (i.e., pivot point) on the **X axis** and/or **Y axis**.
  - **Counter Clockwise** checkbox: Click (enable) this option to make the object rotate counterclockwise instead of clockwise.
- **Gesture**: When multi-touch gestures are enabled, the end user may use two-finger "turn" gestures to rotate this object during run time. The changes in angle are written back to the project tag specified in the **Tag/Expression** box above.
  - **Enable** option: Select to enable gestures on this specific object.

    Please note that Multi-Touch must also be enabled for the project and screen.
  - **Inertia** option: Select to apply inertia to this object, so that it slows down naturally rather than stops abruptly when the end user stops touching it.
  - **Disable** box: Specify a tag/expression. When its value is TRUE (i.e., non-zero), gestures are disabled on this object.
  - **Security** box: Type the minimum security level that the end user must have to use gestures on this object.

  For more information, see Multi-Touch on page 351.

### *Use custom properties to set property values when screens are opened*

Studio allows you to assign values, tags, or even expressions to screen objects properties or animation properties dynamically when opening the screens. This feature is based on the use of Custom Properties (formerly known as Mnemonics).

Custom Properties are place holders (aliases) that can configured to screen animations and objects properties. The built-in function $Open() can be used to set values, tags, or even expressions dynamically to the Custom Properties when opening the screen. Therefore, the same screen can be used to display different values, depending on the context in which it was opened.

The Custom Properties follow the syntax below:

```
#CustomPropertyName:CustomPropertyValue
```

...where:

**`CustomPropertyName`**

Identifier (alias name) of the custom property.

**`CustomPropertyValue`**

Actual (default) value of the custom property. It can be a literal value (numeric or alphanumeric), a tag, or even an expression between parentheses. It can also be omitted (no default value), so there is no default value for the custom property, but its value can still be set dynamically when opening the screen with the built-in function $Open().

During the runtime, only the CustomPropertyValue is used and the remaining text from the aforementioned syntax is ignored. Examples:

| Custom Property (full syntax) | Custom Property Name (alias used as identifier) | Custom Property Value (used during the runtime) |
|---|---|---|
| #MyNumValue:10 | **MyNumValue** | **10** |
| #MyTextValue:"ABC" | **MyTextValue** | **"ABC"** |
| #MyTag:Second | **MyTag** | **Second** |
| #MyExpression:(Minute*10) | **MyExpression** | **(Minute*10)** |

In most cases, the Custom Property value is completely replaced by the value passed by the built-in function $Open(). For example, assume you configured an object property from the screen MyScreen with the syntax:

```
#MyCustomProperty:
```

Then, you execute the following expression to open the screen:

```
$Open("MyScreen",-1,-1,-1,-1,0,0,"#MyCustomProperty:Second")
```

The screen will be opened, and the placeholder #MyCustomProperty: will be replaced by the tag Second during the runtime.

Additional examples:

| Custom Property configured on the objects and animations | Custom Property passed by the built-in function $Open() | Actual value executed during the runtime |
|---|---|---|
| #MyCustomProperty:Minute | #MyCustomProperty:Second | Second |
| #MyCustomProperty:Minute | #MyCustomProperty:Second->Quality | Second->Quality |
| #MyCustomProperty:Minute | #MyCustomProperty:Mytag[1].MyMember | Mytag[1].MyMember |
| #MyCustomProperty:Minute | #MyCustomProperty:Mytag[1].MyMember->Quality | Mytag[1].MyMember->Quality |

### Replace Custom Property value partially using tag fields configured on the objects and animations

Assume you have a screen where you configure the following syntaxes on different objects:

- #MyTag:Minute
- #MyTag:Minute->Min
- #MyTag:Minute->Max
- #MyTag:Minute->Unit

Notice that the same Custom Property (MyTag) is associated with different values (Minute, Minute->Min, Minute->Max, and Minute->Unit). It is valid, as long as all values are associated to different fields of the same tag. Conveniently, you can replace the tag Minute by another tag dynamically and keep the fields configuraton, calling the built-in function $Open() as follows:

```
$Open("MyScreen",-1,-1,-1,-1,0,0,"#MyCustomProperty:Second")
```

This function will replace values as indicated in the following table:

| Custom Property configured on the objects and animations | Custom Property passed by the built-in function $Open() | Actual value executed during the runtime |
|---|---|---|
| #MyCustomProperty:Minute | #MyCustomProperty:Second | Second |

| Custom Property configured on the objects and animations | Custom Property passed by the built-in function $Open() | Actual value executed during the runtime |
|---|---|---|
| #MyCustomProperty:Minute->Min | #MyCustomProperty:Second | Second->Min |
| #MyCustomProperty:Minute->Max | #MyCustomProperty:Second | Second->Max |
| #MyCustomProperty:Minute->Unit | #MyCustomProperty:Second | Second->Unit |

### Replace Custom Property value partially using class tags on the objects and animations

Assume you have a screen where you configure the following syntaxes on different objects:

- #MyTag:MyTagA.MemberX
- #MyTag:MyTagA.MemberY
- #MyTag:MyTagA.MemberZ

Notice that the same Custom Property (MyTag) is associated with different values (MyTagA.MemberX, MyTagA.MemberY, and MyTagA.memberZ). It is valid, as long as all values are associated to different members of the same class tag. Conveniently, you can replace the main tag name MyTagA by another tag dynamically and keep the respective members, calling the built-in function $Open() as follows:

```
$Open("MyScreen",-1,-1,-1,-1,0,0,"#MyCustomProperty:MyTagB")
```

This function will replace values as indicated in the following table:

| Custom Property configured on the objects and animations | Custom Property passed by the built-in function $Open() | Actual value executed during the runtime |
|---|---|---|
| #MyCustomProperty:MyTagA.MemberX | #MyCustomProperty:MyTagB | MyTagB.MemberX |
| #MyCustomProperty:MyTagA.MemberY | #MyCustomProperty:MyTagB | MyTagB.MemberY |
| #MyCustomProperty:MyTagA.MemberZ | #MyCustomProperty:MyTagB | MyTagB.MemberZ |

> **Note:** This partial replacement is valid ONLY if both the original class tag (MyTagA) and the target class tag (MyTagB) share the same class type.

### Replace Custom Property value partially using array tags

Assume you have a screen where you configure the following syntaxes on different objects:

- #MyTag:MyArray[1]
- #MyTag:MyArray[2]
- #MyTag:MyArray[3]

Notice that the same Custom Property (MyTag) is associated with different values (MyArray[1], MyArray[2], and MyArray[3]). It is valid, as long as all values are associated to different array positions from the same tag. Conveniently, you can replace the tag name MyArray by another tag dynamically and keep the respective array positions, calling the built-in function $Open() as follows:

```
$Open("MyScreen",-1,-1,-1,-1,0,0,"#MyCustomProperty:NewArray")
```

This function will replace values as indicated in the following table:

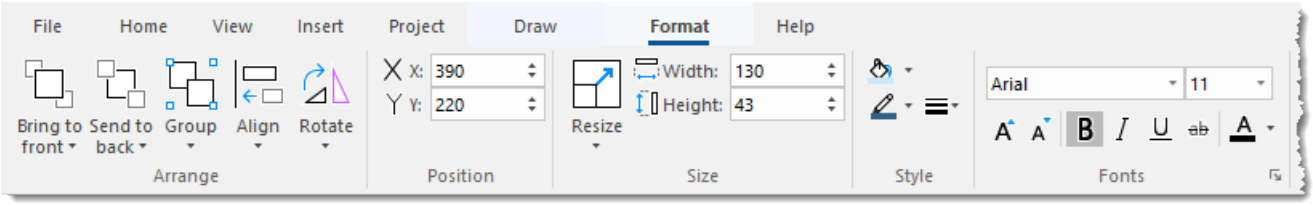
| Custom Property configured on the objects and animations | Custom Property passed by the built-in function $Open() | Actual value executed during the runtime |
|---|---|---|
| #MyCustomProperty:MyArray[1] | #MyCustomProperty:NewArray | NewArray[1] |

| Custom Property configured on the objects and animations | Custom Property passed by the built-in function $Open() | Actual value executed during the runtime |
|---|---|---|
| #MyCustomProperty:MyArray[2] | #MyCustomProperty:NewArray | NewArray[2] |
| #MyCustomProperty:MyArray[3] | #MyCustomProperty:NewArray | NewArray[3] |

**Note:** This partial replacement is valid ONLY if both the original tag (MyArray) and the target tag (NewArray) are array tags.

## Format tab

The **Format** tab of the ribbon is used to format and arrange objects in a project screen.




> **Note:** This tab is available only when you've selected one or more objects in a project screen.

The tools are organized into the following groups:

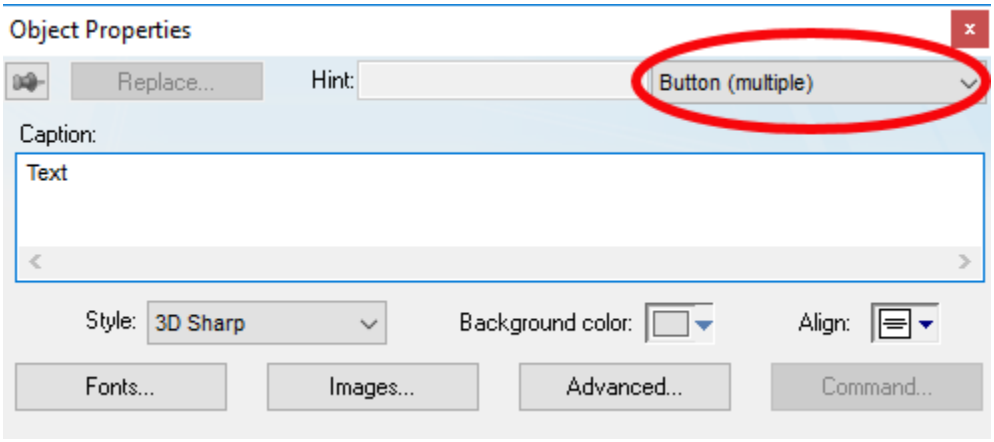
- **Arrange**: Arrange objects in a project screen, including bring to front and send to back, group, align, and rotate.
- **Position**: Precisely adjust the position of a screen object in a project screen.
- **Size**: Precisely adjust the size of a screen object.
- **Style**: Change the fill and line color of a screen object.
- **Fonts**: Change the caption font of a screen object.

### *Change the properties of multiple screen objects*

This task describes how to select two or more screen objects and then change the properties that are common to the selected objects.

Before you begin this task, you must have a project screen open in the screen editor.

Which properties you can change depends on whether you select multiple objects of the same type or of different types. If the objects are of the same type, you can change the properties that are specific to that type. For example, if you select multiple Button objects, then you can change the properties that are specific to Button objects.

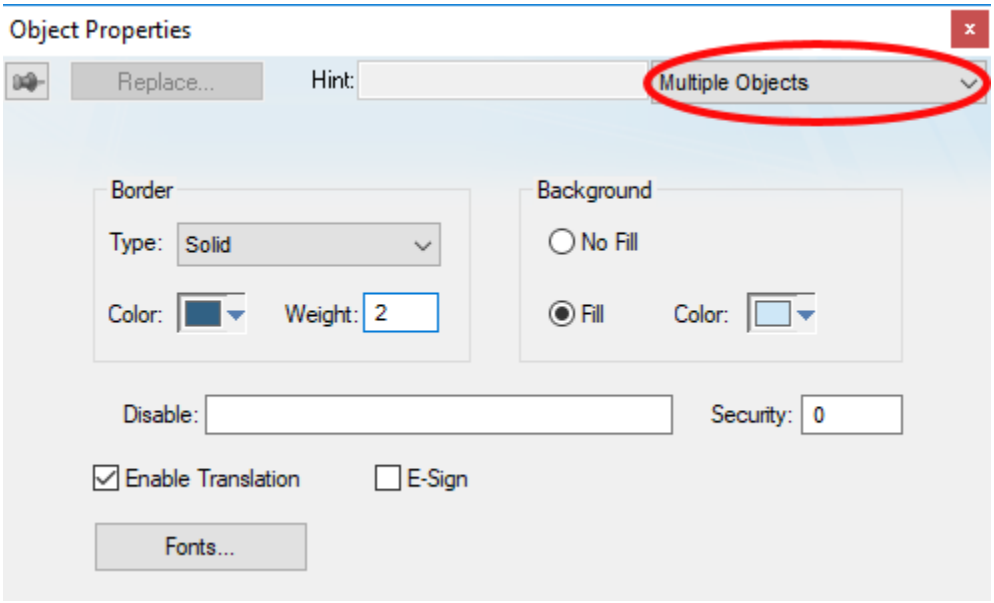



***Object properties of multiple selected Button objects***

For more information about the properties of a specific type of object, see the documentation for that object.

> **Note:** You can only use this method to change the properties of Shapes and Active Objects. You cannot use this method to change the properties of Data Objects, Animations, Library items, or objects in a group.

In contrast, if you select multiple objects of different types, you can change the properties that are common to all of the objects. This includes not only cosmetic properties like Border and Background, but also functional properties like Disable, Security, Enable Translation, and E-Sign. (Some properties may not apply to all objects. For example, Button objects do not have Border and Rectangle objects do not have Security.)




***Object properties for multiple objects of different types***

In both cases, the dialog box shows the current values of the properties of the last selected object.

It is only when you actually change the value of a property that the change is applied to the selected objects. All other properties are left unchanged, regardless the values shown in the dialog box.

To change the properties of multiple screen objects:

1. In the screen editor, do one of the following:
   - Press and hold either **Shift** or **Ctrl** on the keyboard, and then click each object that you want to change; or
   - Use the cursor to draw a selection box around all of the objects that you want to change.

   The objects are selected.
2. Do one of the following:
   - On the **Draw** tab of the ribbon, in the **Editing** group, click **Properties**;
   - Right-click the selected objects, and then on the shortcut menu, click **Properties**; or
   - Press **Alt+Enter** on the keyboard.

   > **Note:** You cannot double-click to open the *Object Properties* dialog box as you otherwise would, because clicking like that clears the selection.

   The *Object Properties* dialog box is displayed for the selected objects.
3. Change the property values that you want to change, and then close the dialog box.

The changes are applied to all of the selected objects.

### *Set the tab order of screen objects*

Set the tab order of screen objects to make a screen easier to use from a physical keyboard/keypad rather than from a mouse or touchscreen.

When you press **Tab** on the keyboard/keypad during project run time, the focus moves to the next object or field on the screen. (This is also known as an object becoming active.) Focus makes it possible to interact with that object or field using only the keyboard/keypad. For example:

- When the focus is on a button, you may press **Return** to click/tap that button;
- When the focus is on a text box, you may type a value into that box and then press **Return** to enter the value; and
- When the focus is on list or menu, you may use the arrow keys to navigate the menu and then press **Return** to make a selection.

If you repeatedly press **Tab**, then the focus will move through all of the objects in a screen according to the screen's tab order. By default, the tab order is the same as the layer order, starting with the layer farthest back (ID: 0) and proceeding to the front (ID: *n*). However, you can draw objects anywhere, rearrange them, and adjust their layers as you develop the screen, so the default tab order of a finished screen may seem to jump around at random rather than move from left to right and/or top to bottom as the user would expect.

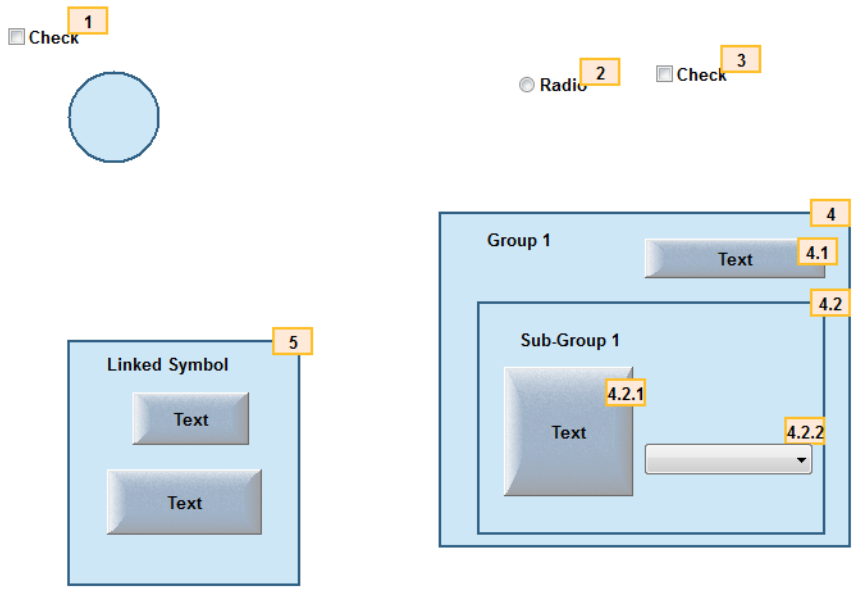
You can set the tab order to make it move through the screen exactly how you want it to, independent of the layer order.

To set the tab order:

1. Open a screen for editing.
2. Right-click on the screen background, and then click **Tab Order** on the shortcut menu.

> **Note:** If **Tab Order** is not available on the shortcut menu, it may be because you actually right-clicked on a screen object or you have a screen object selected. Make sure you have no screen object selected and you are right-clicking on the screen background.

The screen enters Tab Order mode, with the order number of each screen object shown in an orange badge at the top-right corner of the object.




***Example of Tab Order mode***

A group of objects has a single order number for the entire group (e.g., 4), and then the objects within the group have their own sub-order (e.g., 4.1 and 4.2). This continues as deeply as necessary to accomodate

nested groups. If you do not see the sub-order numbers, right-click again in the screen and then click **Expand All Groups** on the shortcut menu.

3. To quickly set the tab order for all objects, simply click on the objects in the order that you want. The order numbers will update as you click on the objects.
4. To set the order number for a specific object:
   a) Double-click on the object's badge.
   The badge changes to a text input box.
   b) Type the order number for that object. Be sure to included any necessary sub-orders.
   c) Press **Return** to apply the change.

> **Note:** You cannot change the sub-order of objects within a Linked Symbol, because it is only a copy of a Master Symbol that is shared across the entire project. Instead, you must edit the Master Symbol itself. For more information, see Save your own project symbols on page 279.

5. When you have finished setting the tab order and want to exit Tab Order mode, right-click on the screen and then click **Tab Order** on the shortcut menu.

### *Bring to front / Send to back*

Bring a screen object to the front or send it to the back of other overlapping objects, as part of arranging the objects on the project screen.

Before you begin this task, you must have a project screen open for editing and the screen must have two or more objects already on it.

> **Note:** In this section, "object" refers to both individual screen objects and object groups, but it does not refer to multiple objects that have been selected together but not grouped.

Screen objects are automatically assigned ID numbers, starting with ID 0, as you add the objects to a project screen. (To check the ID number of an object, simply select it. The ID number is displayed on the status bar at the bottom of the development application window.) The object with ID 0 is the furthest back, behind all other objects on the screen, and each additional object is displayed in front of it.

In other words, objects with higher ID numbers are displayed in front of objects with lower ID numbers.

On a finished screen, you will probably have objects arranged in overlapping layers, and you may want rearrange the objects so that they are displayed correctly. You can bring an object to the front, so that it is displayed in front of all other objects, or you can send an object to the back, so that it is displayed behind all other objects. You can also move an object forward or backward one layer at a time, if necessary.

*Bringing the red circle to the front*

*Sending the blue rectangle to the back*

When you rearrange objects, their ID numbers are automatically adjusted to reflect their new order. For example, if you have four objects on a screen and you send the frontmost object (ID 3) to the back, then that object becomes ID 0 and the ID numbers of the other three objects are increased accordingly.

Please note that windowed objects — that is, screen objects that include some kind of window displaying other data — are handled differently than shapes (e.g., Line, Rectangle) and simple active objects (e.g., Button, Check Box). The windowed objects on a screen will always be in front; you cannot send them behind the other types of objects. This is to ensure that the windowed objects will be rendered properly during project run time.

Windowed objects include:

- Text Box object
- Combo Box object
- Alarm/Event Control object
- Trend Control object
- Grid object
- Most ActiveX and .Net Control objects

If you have more than one windowed object on the same screen, then you can arrange those objects relative to each other. For example, you can send a Grid object behind a Trend Control object. Both windowed objects will still be in front of the other types of objects, however, and in most cases, you should not have overlapping windowed objects anyhow.

To bring a screen object to the front or send it to the back:

1. Select the screen object that you want to move forward or backward.
   "Handles" are displayed at the object's corners to show that it is selected.
2. To bring the object all the way to the front (i.e., increase it to the highest ID number), do one of the following:
   - On the **Format** tab of the ribbon, in the **Arrange** group, click **Bring to Front**; or
   - Right-click the object itself, and then click **Bring to Front** on the shortcut menu.
3. To bring the object one layer forward (i.e., to increase its ID number by 1), do one of the following:
   - On the **Format** tab of the ribbon, in the **Arrange** group, click and hold **Bring to Front** until the shortcut menu appears, and then click **Bring Forward** on the shortcut menu; or
   - Right-click the object itself, and then click **Bring Forward** on the shortcut menu.
4. To send the object all the way to the back (i.e., decrease it to ID 0), do one of the following:
   - On the **Format** tab of the ribbon, in the **Arrange** group, click **Send to Back**; or
   - Right-click the object itself, and then click **Send to Back** on the shortcut menu.
5. To send the object one layer backward (i.e., to decrease its ID number by 1), do one of the following:
   - On the **Format** tab of the ribbon, in the **Arrange** group, click and hold **Send to Back** until the shortcut menu appears, and then click **Send Backward** on the shortcut menu; or
   - Right-click the object itself, and then click **Send Backward** on the shortcut menu.

### *Group and ungroup screen objects*

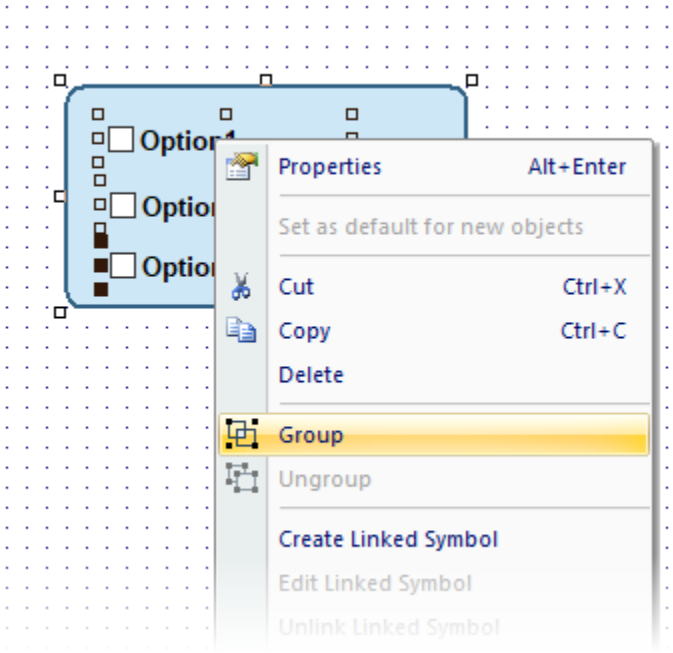
Use the **Group** command to group screen objects together.

A group of objects can be selected, moved, copied, and in some cases modified as if it is a single object, which makes it easier to manage in a busy project screen. Moreover, a group can be grouped with other groups to create increasingly complex groups.

To group two or more screen objects, select the objects that you want to group and then do one of the following:

- On the **Format** tab of the ribbon, in the **Arrange** group, click **Group**; or

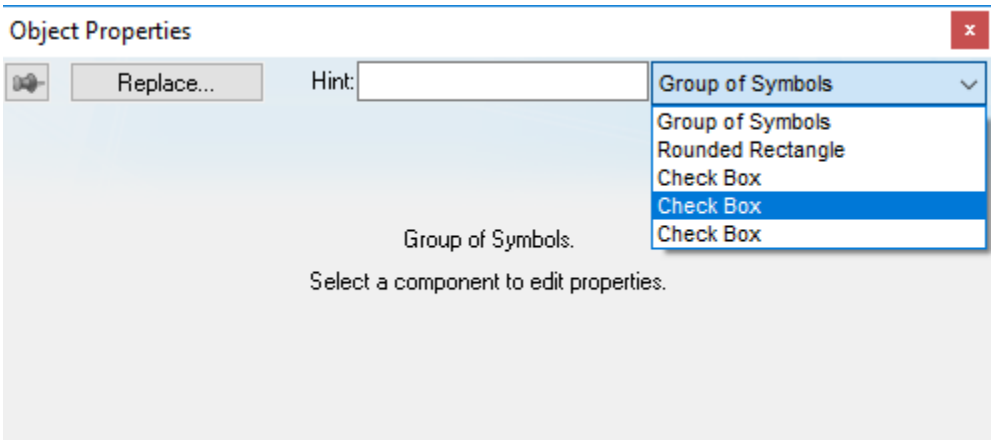
- Right-click the selected objects, and then click **Group** on the shortcut menu.




***Grouping the selected objects***

> **Tip:** If the **Format** tab of the ribbon is not visible, or if the **Group** command is not available on the shortcut menu, it is because you have not selected any objects in the screen editor.

To edit the object properties on a group, use the *Object Properties* dialog box just as you would on a single object. A group has more than one set of object properties, however, like an object that has animations added to it. As such, use the list in the top-right corner of the dialog box to select each set of object properties.




***Selecting a set of object properties in a group***

Once you have created a group — also called a symbol — you may choose to save a master of it in your Symbols Library and then reuse it elsewhere in your project. Each copy will be linked to the master so that if you change the master, all of the linked copies will also be changed. For more information, see Save your own project symbols on page 279.

To ungroup a group of objects, select the group and then do one of the following:

- On the **Format** tab of the ribbon, in the **Arrange** group, click and hold **Group** to access the menu, and then click **Ungroup** on the menu; or
- Right-click the selected group, and then click **Ungroup** on the shortcut menu.

> **Tip:** A complex group can comprise several subgroups, so to ungroup it completely, first ungroup the group and then ungroup the subgroups.

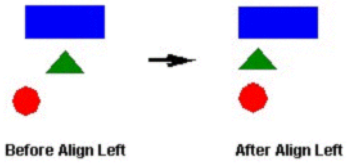
## *Align, Center and Distribute Tools*

When you select a series of objects (two or more), you can align those objects based on the location of the last object selected. As you select objects, solid handles display on the last object selected, and the handles on all previously selected objects become empty (unfilled) boxes.

> **Note:** In all of the figures provided, the rectangle represents the last object selected.

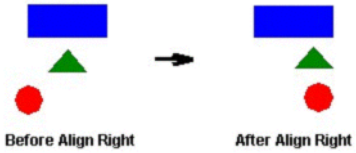
Use the following alignment tools to align a series of objects.

Click the **Align left** tool to align all selected objects to the left edge of the last object selected. For an example, see the following figure:



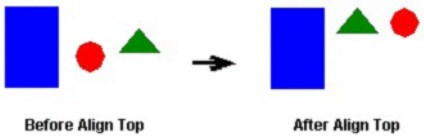

*Aligning Objects Left*

Click the **Align right** tool to align all selected objects to the right edge of the last object selected. For an example, see the following figure:



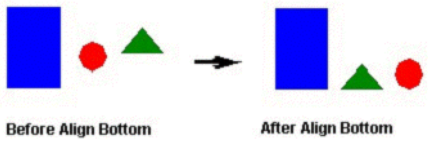

*Aligning Objects Right*

Click the **Align top** tool to align all selected objects to the top edge of the last objected selected. For an example, see the following figure:




*Aligning Object Tops*

Click the **Align bottom** tool to align all selected objects to the bottom edge of the last object selected. For an example, see the following figure:

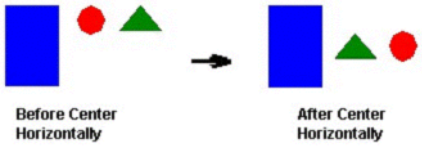



***Aligning Object Bottoms***

Click the **Center Vertically** tool to align all selected objects to the vertical center of the last object selected. For an example, see the following figure:

***Centering Objects Vertically***

Click the **Center Horizontally** tool to align all selected objects to the horizontal center of the last object selected. For an example, see the following figure:

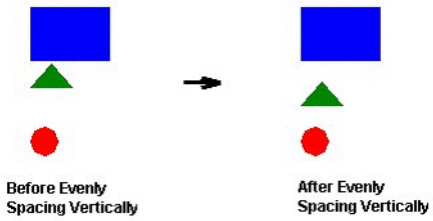



***Centering Objects Horizontally***

Click the **Evenly distribute horizontally** tool to put an equal amount of horizontal space between a series of objects (two or more). For an example, see the following figure:

***Distributing Objects Horizontally***

Click the **Evenly distribute vertically** tool to put an equal amount of vertical space between a series of objects (two or more). For an example, see the following figure:

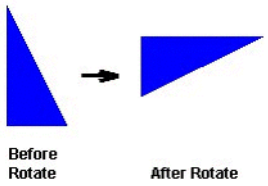



***Distributing Objects Vertically***

> **Note:** The distribution tools may move the last object selected (with solid handles) by no more than a few pixels to equally space all of the objects.

## *Rotate Tool*

Click the **Rotate** tool to rotate the selected object 90 degrees (a quarter turn) clockwise.

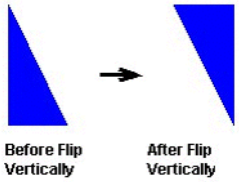



***Rotating Objects***

> **Note:** You can use this tool only with a single selected object or grouped object. You cannot use this tool with multiple objects selected.

### FLIP VERTICALLY TOOL

Click the **Flip Vertically** tool to invert the selected object vertically. The object rotates around an imaginary line through its vertical center until it is a mirror image of the original object. For an example, see the following figure:

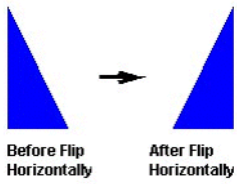



***Flipping Objects Vertically***

> **Note:** You can use this tool only with a single selected object or grouped object. You cannot use this tool with multiple objects selected.

### FLIP HORIZONTALLY TOOL

Click the **Flip Horizontally** tool to invert the selected object horizontally. The object rotates around an imaginary line through its horizontal center until it is a mirror image of the original object. For example, see the following figure:




***Flipping Objects Horizontally***

> **Note:** You can use this tool only with a single selected object or grouped object. You cannot use this tool with multiple objects selected.

## *Resize Tools*

Use the following ribbon options for resizing:

- Click the **Resize width** tool to set the width of all selected objects to the width of the last object selected, or to resize one selected object so that its width equals its height.
- Click the **Resize height** tool to set the height of all selected objects to the height of the last object selected, or to resize one selected object so that its height equals its width.

> **Tip:** You can use **Resize width** and **Resize height** to turn an ellipse into a circle or a rectangle into a square. Make sure you have only one object selected, however.

You also can use the mouse pointer and arrow keys to resize objects. When you select an object (or group of objects) with the pointer, handles are displayed at each corner and at the midpoint of each side. You can use these handles as follows:

- **To enlarge an object**, drag a handle in the direction you want to resize the object. Dragging a side handle resizes the object in one direction only (height only or width only). Dragging a corner handle resizes the entire object (height and width).

  When you drag a corner handle, the object's proportions are constrained by default. To freely resize the object, hold down the SHIFT key as you drag the handle.
- **To resize an object one pixel at a time**, click and hold a handle and then press the arrow keys. For the corner handles and the left and right side handles, press the LEFT ARROW and RIGHT ARROW keys. For the top and bottom handles, press the UP ARROW and DOWN ARROW keys.
- **To resize an Open or Closed Polygon**, draw a selection box around all of the polygon's points and group them. You can then resize the polygon like a normal object.

> **Note:** When you resize a Symbol, a Group, or any other collection of selected objects, all of the objects in the collection are resized in the same direction and to the same degree.

## *Fill Color Tool*

Click the **Fill Color** tool to specify a default fill color for the following objects:

- *Closed Polygons*
- *Ellipses*
- *Rounded Rectangles*
- *Rectangles*

> **Tip:** To save development time, select several objects (of any type specified in the preceding list) and use **Fill Color** to specify a default fill color for all of them at once.

### *Line Color Tool*

Click the **Line Color** tool to specify a line color for selected objects or to set a default color for new objects, including the following:

- Open Polygons
- Closed Polygons
- Lines
- Ellipses
- Rounded Rectangles
- Rectangles

When you click the **Line Color** tool, the *Line Selection* dialog displays. Use this dialog to specify line styles and color for the selected objects.

> **Tip:** To save development time, you can select several of the preceding objects and use the **Line Color** tool to specify a line color for all of the objects at once.

### *Fonts Tool*

Click the **Fonts** tool to specify the font and color for selected Text objects, or to specify a default font and color for new Text objects.

> **Tip:** To save development time, select several Text objects and use the **Fonts** tool to specify font and color settings for all of the objects at once. (You cannot use this function for grouped Text objects however.)

## Data Input

Project screens are often viewed on HMI panels and mobile devices that have touchscreens instead of physical keyboards. Therefore, the user must have some way to input data (i.e., numeric values and text) using only the touchscreen. This section describes how to configure the data input options for your project.

The following screen objects, animations, and functions can accept data input from the user:

- Text object with Text Data Link animation (if the **Input Enabled** option is selected in the object properties)
- Text Box object (if the **Input Enabled** option is selected in the object properties)
- Combo Box object (if the **Input Enabled** option is selected in the object properties)
- Alarm/Event Control object (for adding comments and filtering the list)
- Trend Control object (for changing the time and period of the trend graph)
- Grid object (in any column that has input enabled)
- KeyPad function
- ShowInplaceInput function

Also, if you have enabled the security system for your project, the user will occasionally need to type their user name and/or password — for example, to log on to the project, to e-sign an event, to change their password, and so on.

In each of these situations, when data input is required — that is, when an object is tapped or a function is called — you can choose to display a special, on-screen interface in front of the normal project screen. This interface serves two purposes. First, it ensures that the user knows the project is waiting for their input; some project screens are so full of objects that it can be difficult for the user to see a blinking cursor in a particular text box. And second, it provides a touchscreen keyboard or keypad on which the user can actually type their input.

The exact nature of the on-screen interface varies depending on the type of client that the user is using to access your project.

### *Data input in screens on Thin Clients*

To get data input from the user in screens viewed on Thin Clients, you can choose to display a Virtual Keyboard (VK) dialog box that is similar to the On-Screen Keyboard app in Windows.

You can enable the Virtual Keyboard separately for each type of Thin Client:

- For Secure Viewer — which includes the local Viewer that runs as part of the project runtime — configure your project's Viewer settings.

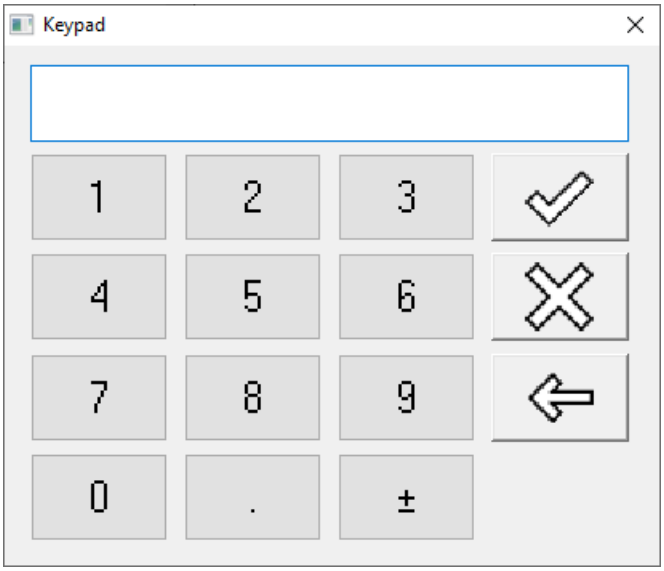
Keep in mind that when you enable the Virtual Keyboard for each type of Thin Client, it will be displayed on all client stations that use the same type of Thin Client to access your project, regardless of whether those stations actually have touchscreens instead of physical keyboards. Consider how this might affect the usability of your project, especially if there will be a mix of client stations.

#### Keyboard types and options

When you enable the Virtual Keyboard, you can also select a default keyboard type and some additional options. The following tables shows the possible combinations of keyboard types and options:

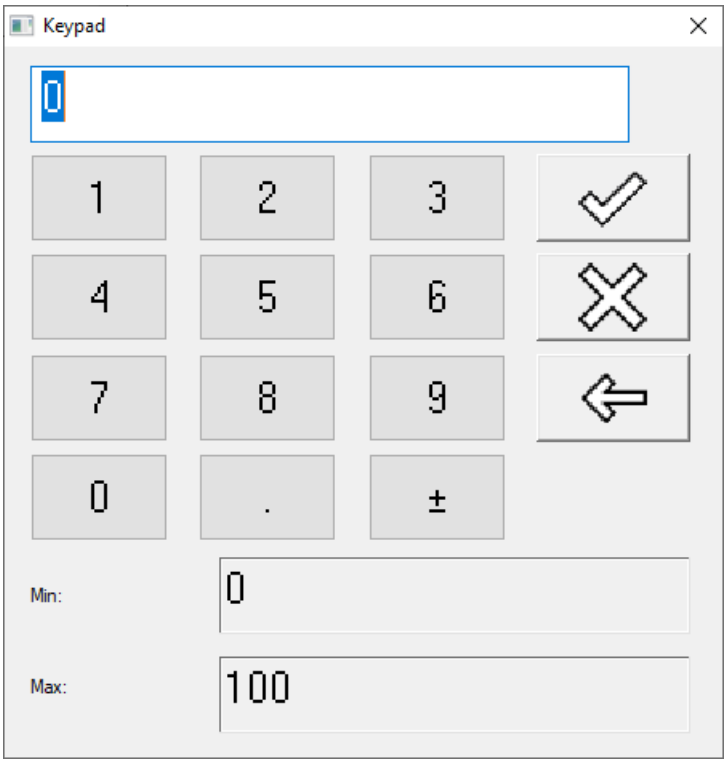
**Keypad**

This standard keypad is used to enter numeric values only.



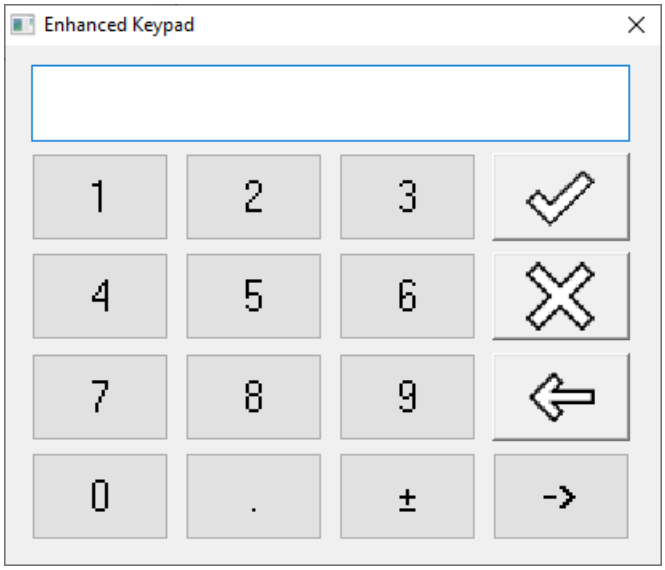

### Keypad with the Enable Min/Max fields option selected

This standard keypad is used to enter numeric values only. The minimum and maximum values allowed for the associated tag — as set in the tag properties — are displayed at the bottom of the keypad.




### EnhKeypad

This enhanced keypad is used to enter alphanumeric characters on devices that have small displays. The **->** button in the bottom-right corner lets the user proceed through sets of keys until they find the specific character they want.



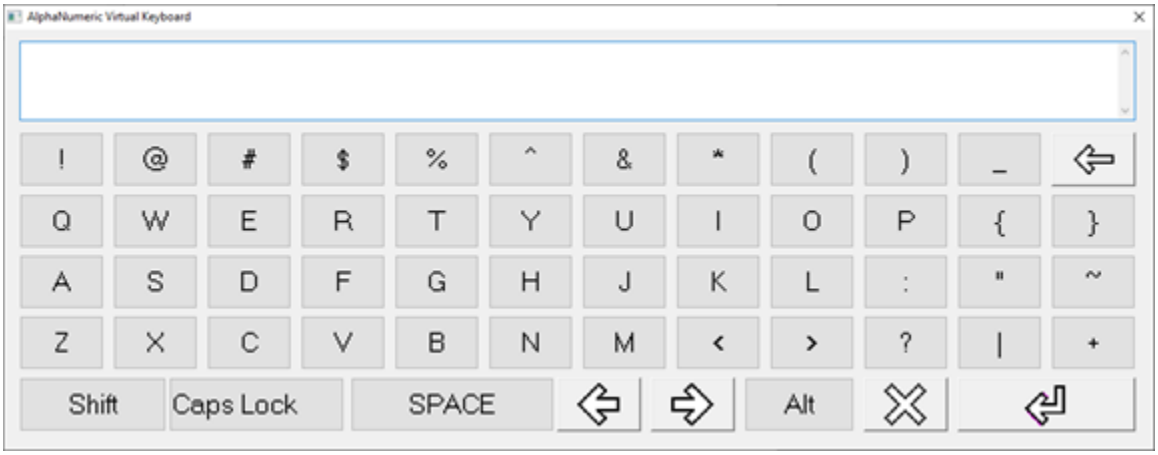

### AlphaNumeric

This full keyboard is used to enter alphanumeric characters.

> **Note:** The screen shot above has been scaled to fit this documentation. The actual size of the AlphaNumeric keyboard is comparable to the other keyboards described in this topic.

### AlphaNumeric with the Enable multi-line text input option selected

This full keyboard accepts multi-line text with line breaks (CR+LF). When Caps Lock is enabled, the OK/Accept key in the bottom-right corner of the keyboard becomes a Return key instead.




> **Note:** The screen shot above has been scaled to fit this documentation. The actual size of the AlphaNumeric keyboard is comparable to the other keyboards described in this topic.

Once you have enabled the Virtual Keyboard and selected a default keyboard type, it will be used for all data input in your project unless you override it for a specific screen object or function call. For a screen object, you can select another keyboard type in that object's properties. For a function call, you can use the optional function parameters to determine how the . For more information, see the documentation for each screen object and function.

### Change the size, position, or language of the Virtual Keyboard

By default, the Virtual Keyboard is displayed near the screen object that invoked it, but you can edit the following properties in your project file (*<project name>*.app) in order to force the keyboard to be displayed in a fixed size and/or position:

```
[Keypad]
VKType=<0-3> // 0 (Default) = The Virtual Keyboard will work in default mode
             // 1 (Auto Size) = The Virtual Keyboard will automatically resize
 according to the screen size
             // 2 (Manual Size/Position) = The entries below will be used

//Properties for AlphaNumeric
AlphaNumeric-PosX=   // The TOP coordinate (in pixels) where the Virtual Keyboard must
 be displayed
AlphaNumeric-PosY=   // The LEFT coordinate (in pixels) where the Virtual Keyboard must
 be displayed
AlphaNumeric-Width=  // Virtual Keyboard width (in pixels)
AlphaNumeric-Height= // Virtual Keyboard height (in pixels)

//Properties for EnhKeypad
EnKeyPad-PosX=
EnKeyPad-PosY=
EnKeyPad-Width=
EnKeyPad-Height=

//Properties for Keypad with Min/Max fields enabled
KeyPadMinMax-PosX=
KeyPadMinMax-PosY=
KeyPadMinMax-Width=
KeyPadMinMax-Height=

//Properties for Keypad with Min/Max fields not enabled
KeyPad-PosX=
KeyPad-PosY=
```

```
KeyPad-Width=
KeyPad-Height=
```

To change the language of the Virtual Keyboard during project run time, use the `SetKeyboardLanguage` function.

## *Data input in screens on Mobile Access*

To get data input from the user in screens viewed on Mobile Access, you can choose to display a customized Data Input dialog box. Displaying this dialog box automatically invokes the built-in keyboard on mobile devices.

> **Note:**
>
> At this time, the Data Input dialog box is displayed only for the following screen objects and functions:
>
> - Text object with Text Data Link animation (if the **Input Enabled** option is selected in the object properties)
> - Text Box object (if the **Input Enabled** option is selected in the object properties)
> - KeyPad function
>
> More objects and functions will be supported in the future, as the Mobile Access web interface is improved.

The Data Input dialog box is automatically displayed for all Text objects with Text Data Link animations. It cannot be disabled. There are several reasons for this, but the most important to you and your users is that it makes it clear when the project is waiting for input from the user. Without the dialog box, it might be difficult to see that a Text object has become active in the Mobile Access web interface, especially if a project screen has been scaled to fit a smaller display.

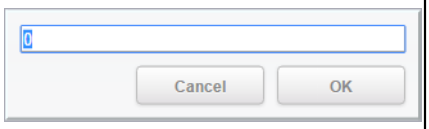
In contrast, the Data Input dialog box is optional for Text Box objects. It is enabled by default, but you can choose to disable it and then have the user type directly into each Text Box object like they would type into a text box in a standard web form. To disable the dialog box for Text Box objects, clear the **Always Use Data Input Dialog** option in the *Mobile Access Configuration* worksheet. For more information, see Configure the global settings for all areas on page 790.

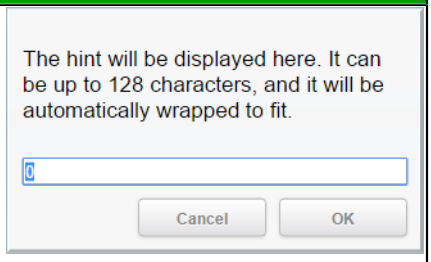
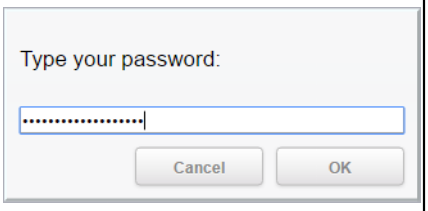
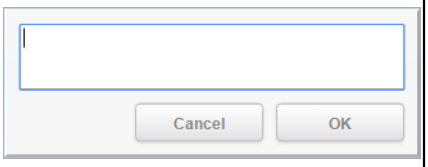
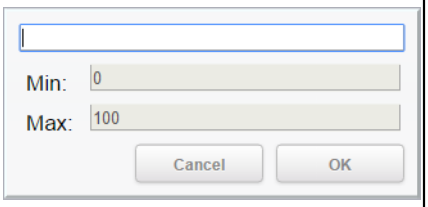
When the Data Input dialog box is invoked, it is displayed in the center of the web browser window. It is actually part of the Mobile Access web interface, so the user cannot move or close it like they might move or close a dialog box that is displayed by the underlying operating system. The user must acknowledge it — either by entering a value or by clicking/tapping **Cancel** — and the project screen is paused (i.e., graphics and tag values are not updated) until they do so.

The Data Input dialog box has the same appearance on all client stations, regardless of the station's web browser or operating system, because it is part of the Mobile Access web interface. However, displaying the dialog box on a mobile device also invokes that device's native virtual keyboard, and the appearance of that keyboard can vary greatly from device to device, depending on the device's operating system and settings.

### Examples of the Data Input dialog box

The appearance of the Data Input dialog box is determined by which options are selected in the object properties for a specific screen object. The following table shows some examples:

| Options Selected | Description | Appearance |
|---|---|---|
| • **Input Enabled** | Single-line data input dialog box. |   |
| • **Input Enabled** | Single-line data input dialog box with hint. | |

| Options Selected | Description | Appearance |
|---|---|---|
| • **Hint** (or *optStrHint* parameter for `Keypad` function) | |  |
| • **Input Enabled**<br>• **Hint** (or *optStrHint* parameter for `Keypad` function)<br>• **Password** (or *optNumIsPassword* parameter for `Keypad` function) | Single-line data input dialog box with hint and obfuscated password.<br>**Note:** Password obfuscation applies to text values only. In other words, the specified project tag must be String type. If it is not, the option is ignored. |  |
| • **Input Enabled**<br>• **Multi-line** (Text Box object only) | Multi-line data input dialog box. To include line breaks in the entered value, press **Return** on the keyboard (either physical or virtual). The value will not actually be entered until you click/tap **OK** in the dialog box.<br>**Note:** Multi-line applies to text values only. In other words, the specified project tag must be String type. If it is not, the option is ignored. |  |
| • **Input Enabled**<br>• **Minimum Value** (or *optNumMin* parameter for `Keypad` function)<br>• **Maximum Value** (or *optNumMax* parameter for `Keypad` function) | Single-line data input dialog box with the minimum and maximum values allowed. The user must enter a value that is between the minimum and maximum values, and if they do not, the entered value is highlighted in red and the **OK** button is disabled.<br>**Note:** The minimum and maximum values are valid for numeric values only. In other words, the specified project tag must be Integer or Real type. If it is not, the options are ignored. |  |

## Entering negative values

Some mobile device operating systems have limitations on how users can enter negative values.

For example, the default numeric keyboard on some Android devices does not allow the user to enter negative values, so it might be necessary to install/use a different keyboard.

Keep these limitations in mind and communicate them to your users, if necessary

## Multi-Touch

You can enable multi-touch gestures in project screens in order to provide your end users with additional interface options.

On an older touchscreen device that supports only a single touch point, the user's touch or tap is directly equivalent to a simple mouse click, so they cannot do anything that they could not otherwise do by connecting a mouse to the device.

On a newer touchscreen device that supports multiple touch points, however, the user can use two or more fingers at the same time in order to manipulate project screens and screen objects. The additional touch points provide context that is not available in a single touch point; two fingers working together can perform different gestures, and different gestures trigger different actions on-screen. For example:

- You can swipe two fingers together in the same direction to quickly pan through a window or scroll through a list;
- You can stretch and pinch with two fingers to resize a screen object or zoom in/out on a project screen; and
- You can "grip" a screen object with two fingers and then rotate it like a dial.

Moreover, if you are experienced with VBScript, you can use Touch Events to customize the behavior of multi-touch beyond the standard gestures described in this section. These Touch Events are actually VBScript sub-routines that receive the raw touch input data from the Windows API.

### Requirements and considerations

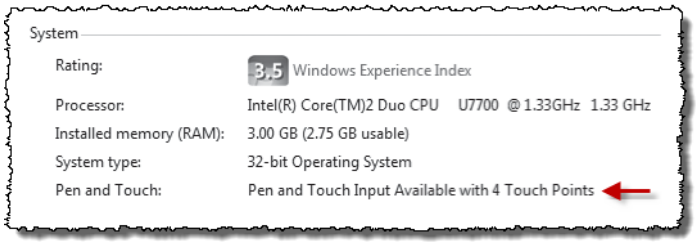
Keep in mind that these system requirements do not apply to the computer that you are using to develop your EcoStruxure Machine SCADA Expert 2020 project; if you can run the project development application and create a project, you can configure the necessary settings to include multi-touch in your project. Instead, these requirements apply to the project thin clients that your end users will use to access your project.

To support multi-touch gestures on a Windows computer or Windows Embedded device, the client device must have:

- A touchscreen display that is capable of reading two or more touch points;
- A version of Windows or Windows Embedded that includes support for touch input; and
- One of the following project viewers or thin clients:
  - Secure Viewer for Windows

  For more information about installing each of these, see Installation Guide on page 33.

To confirm that a Windows computer or Windows Embedded device can support multi-touch gestures, open the *System* control panel (**This PC** > **Control Panel** > **System and Security** > **System**) and then look for **Pen and Touch**. It must say that touch input is available with at least two touch points.




***Example of Pen and Touch in the System control panel***

To support multi-touch gestures on a mobile device (i.e., smartphone or tablet) that accesses your project through Mobile Access, the client device must have:

- A touchscreen display that is capable of reading two or more touch points;
- An operating system that includes support for touch input; and

- An HTML5-compatible web browser.

We cannot give further instructions for confirming that a specific device can support multi-touch gestures, but generally speaking, all of the latest Android and iOS devices should be able to.

If you will be running your project in a mixed environment — that is, if your end users will be using different types of client devices, including some that do not support touch input — then you should be careful about how you include multi-touch gestures in your project. Always provide a second way to manipulate a screen or object, using a keyboard, a mouse, or a single-finger tap.

### Limitations on support for Multi-Touch

At this time, multi-touch gestures are fully supported only on Windows computers and Windows Embedded Standard devices. Support for specific gestures on other project viewers and thin clients is limited. The specific limitations are described in their respective sections, but the following table provides a summary:

| Feature | Windows & Windows Embedded Standard | Mobile Access |
|---|---|---|
| Zoom and Pan gestures in project screens | Supported | Supported |
| Gestures with Alarm/Event Control object | Supported | Not supported |
| Gestures with Trend Control object | Supported | Not supported |
| Gestures with Grid object | Supported | Not supported |
| Gestures with Position animation | Supported | Supported |
| Gestures with Resize animation | Supported | Not supported |
| Gestures with Rotation animation | Supported | Not supported |
| Touch Events for up to 10 touch points | Supported | Not supported |

## *About the Multi-Touch settings for project screens*

The Multi-Touch settings determine how multi-touch gestures behave in each project screen.

You may configure default settings for your entire project, so that each new project screen has the same settings as all other screens of the same type, and you may also customize the settings for specific screens when the default settings would not be appropriate.

### CONFIGURE THE DEFAULT MULTI-TOUCH SETTINGS FOR ALL PROJECT SCREENS

Configure the default Multi-Touch settings for all project screens in your project, so that the screens all behave the same way during project run time.

The Multi-Touch settings come preconfigred for most common uses, so you need to configure them further only if:

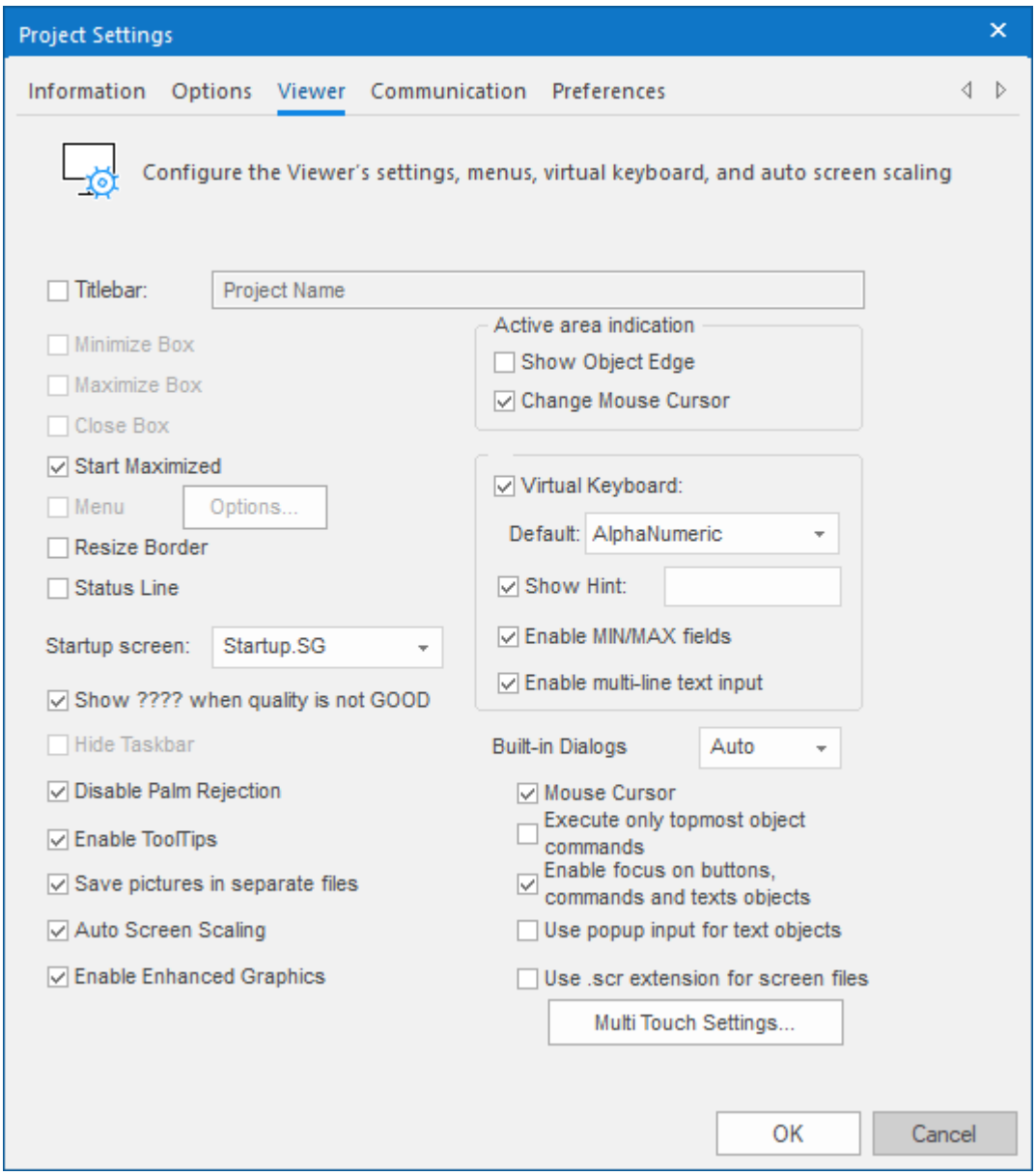
- You are not satisfied with the run-time behavior of these settings; and/or
- You want to use project tags to programmatically change the setting during project run time.

Keep in mind that these are the default settings for all screens in your project. If you only want to configure the settings for a specific screen, see Configure the Multi-Touch settings for a specific project screen on page 355.

To configure the default Multi-Touch settings for all project screens:

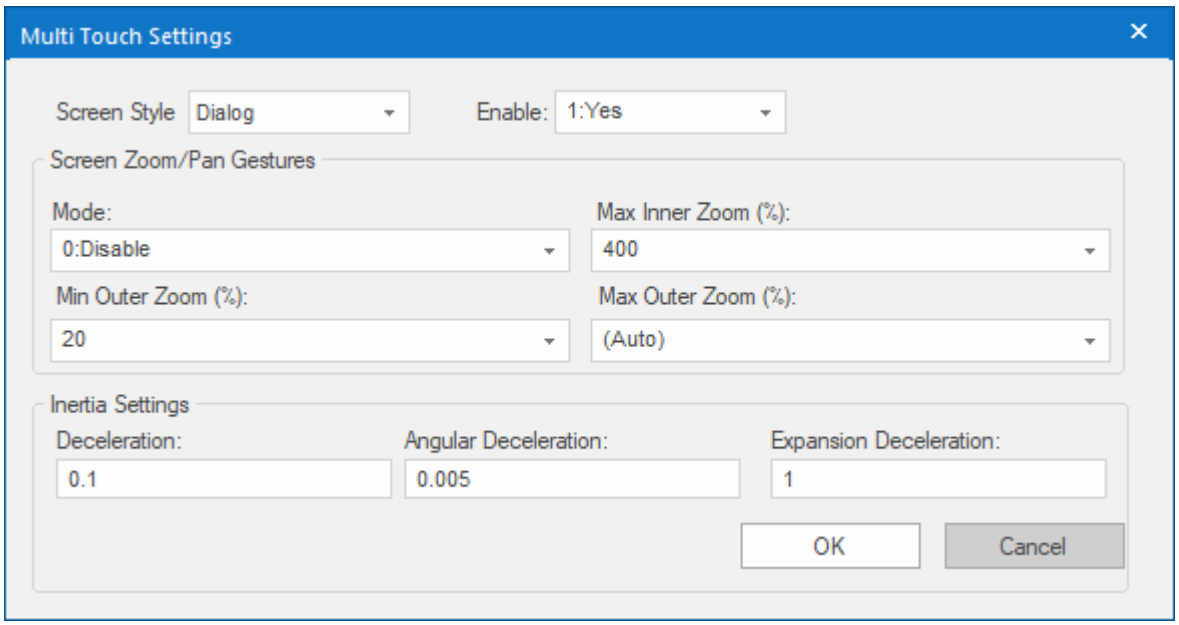
1. On the **Project** tab of the ribbon, in the **Settings** group, click **Viewer**.

   The *Project Settings* dialog is displayed with the **Viewer** tab selected.




2. Click **Multi-Touch Settings**.

   The *Multi-Touch Settings* dialog is displayed.




3. In the **Screen Style** list, click the style for which you want to configure the default settings.

   Every style has its own settings, so you might need to repeat the following steps for each style. For more information about the different styles of project screens, see Screen Attributes on page 231.

4. In the **Enable** list, either click an option or type the name of a project tag (Boolean or Integer type).

   This setting determines whether the Multi-Touch features in general (i.e., gestures, screen zoom/pan, inertia, touch events) are enabled for the selected screen style. If you typed the name of a project tag, the value of the tag will control the setting during project run time.

| Option | Description |
|---|---|
| **0:No** | The Multi-Touch features are disabled for the selected screen style. |
| **1:Yes** | The Multi-Touch features are enabled for the selected screen style. |

5. In the **Screen Zoom/Pan Gestures** area, configure the zoom settings.

   a) In the **Mode** box, either select an option or type the name of a project tag (Integer type).

   This setting determines the zoom/pan mode for the selected screen style. If you type the name of a project tag, the value of that tag will determine the mode during project run time.

| Option | Description |
|---|---|
| **0:Disable** | Zoom/Pan is disabled for the selected screen style. |
| **1:Inner** | Inner Zoom/Pan is enabled for the selected screen style. Zooming changes the scale of the screen's contents, and panning moves the viewable area within the screen's border. The screen itself does not change size or position. |
| **2:Outer** | Outer Zoom/Pan is enabled for the selected screen style. In practice, this is more like Resize/Move: zooming changes the size of the entire screen (automatically scaling the screen's contents to fit), and panning moves the screen in relation to the other open screens.<br><br>Please note this mode is not directly supported on Mobile Access. However, selecting the **Enable Screen View Zoom** option (in the Mobile Access Configuration settings) produces essentially the same effect for all screens. For more information, see Configure the global settings for all areas on page 790. |

For more information, see Using multi-touch gestures in project screens on page 358.

b) In the **Max Inner Zoom (%)** box, either select an option or type the name of a project tag (Integer type).

This will be the maximum magnification allowed for the viewable area inside the project screen, when **Mode** is set to **Inner**. If you type the name of a project tag, the value of that tag will determine the zoom during project run time. Values less than 100 (i.e., 100%) and greater than 1000 (i.e., 1000%) will be ignored.

c) In the **Min Outer Zoom (%)** box, either select an option or type the name of a project tag (Integer type).

This will be the minimum size allowed for the project screen (as a percentage of size specified in the screen attributes), when **Mode** is set to **Outer**. If you type the name of a project tag, the value of that tag will determine the zoom during project run time. Values less than 20 (i.e., 20%) and greater than 100 (i.e., 100%) will be ignored.

d) In the **Max Outer Zoom (%)** box, either select an option or type the name of a project tag (Integer type).

This will be the maximum size allowed for the project screen (as a percentage of the size specified in the screen attributes), when **Mode** is set to **Outer**. The default option is **(Auto)**, which means that the maximum screen size will be equal to the size of the display on which the project is viewed. If you type the name of a project tag, the value of that tag will determine the zoom during project run time. Values less than 100 (i.e., 100%) and greater than 1000 (i.e., 1000%) will be ignored.

6. In the **Inertia Settings** area, configure the deceleration values for the different types of movement.

All values are in pixels per second.

a) In the **Deceleration** box, either type a value or type the name of a project tag (Real type) that contains the value.

This value controls the deceleration from "slide to pan" and "slide to move" gestures.

b) In the **Angular Deceleration** box, either type a value or type the name of a project tag (Real type) that contains the value.

This value controls the deceleration from "turn to rotate" gestures.

c) In the **Expansion Deceleration** box, either type a value or type the name of a project tag (Real type) that contains the value.

This value controls the deceleration from "pinch/stretch to resize" and "pinch/stretch to zoom" gestures.

For a detailed explanation of how inertia is used in multi-touch gestures, see "Inertia Mechanics" at the Microsoft Developer Network website: msdn.microsoft.com/library/dd562169.aspx

7. Repeat from Step 3 for each style of screen that you want to configure.
8. Click **OK** to save the settings and close the dialog.

## CONFIGURE THE MULTI-TOUCH SETTINGS FOR A SPECIFIC PROJECT SCREEN

Configure the Multi-Touch settings for a specific project screen when the project's default settings would not be appropriate.

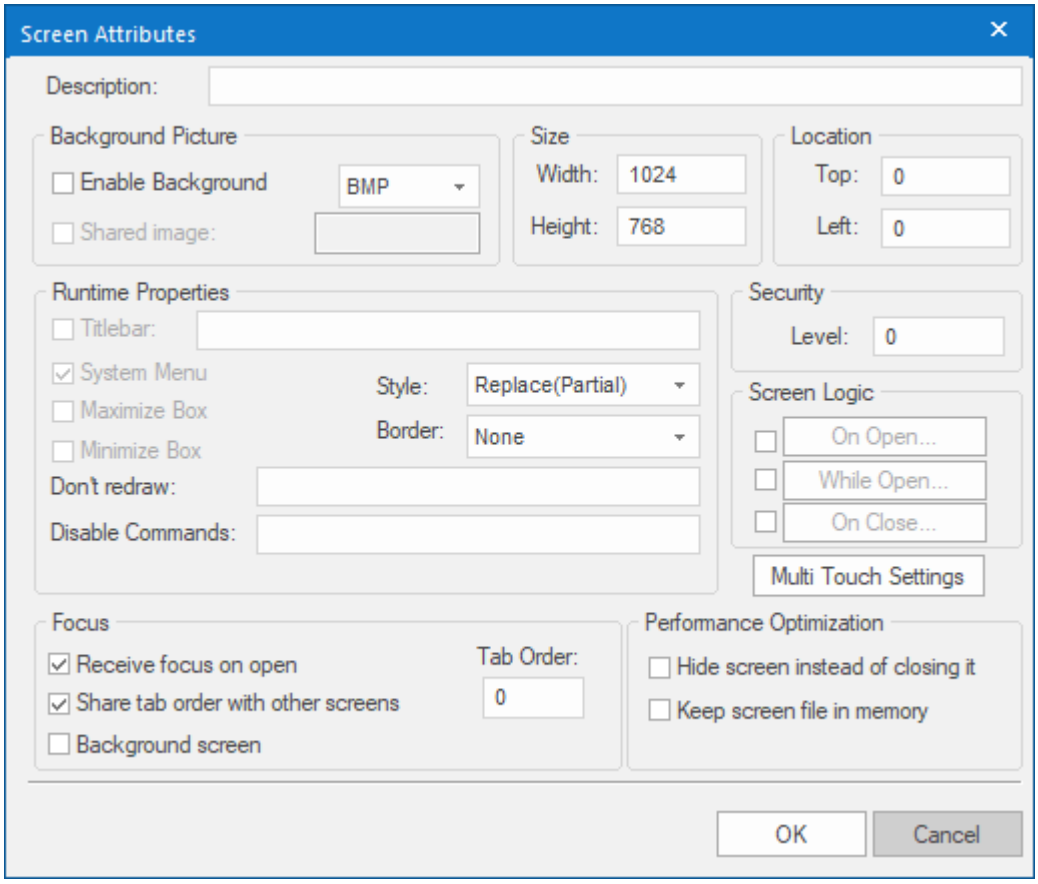
Before you begin this task, you should have the selected Screen worksheet open for editing.

Keep in mind that the project's default settings are there to ensure that the project screens all behave the same way during project run time. Consistency makes your project easier to use. (For more information, see Configure the default Multi-Touch settings for all project screens on page 352.) As such, you should change the settings for a specific project screen only when it is absolutely necessary to the purpose of that screen.

To configure the Multi-Touch settings for a specific project screen:

1. Do one of the following:
   - On the **Draw** tab of the ribbon, in the **Screen** group, click **Attributes**; or
   - Right-click anywhere in the Screen worksheet, and then click **Screen Attributes** on the shortcut menu.

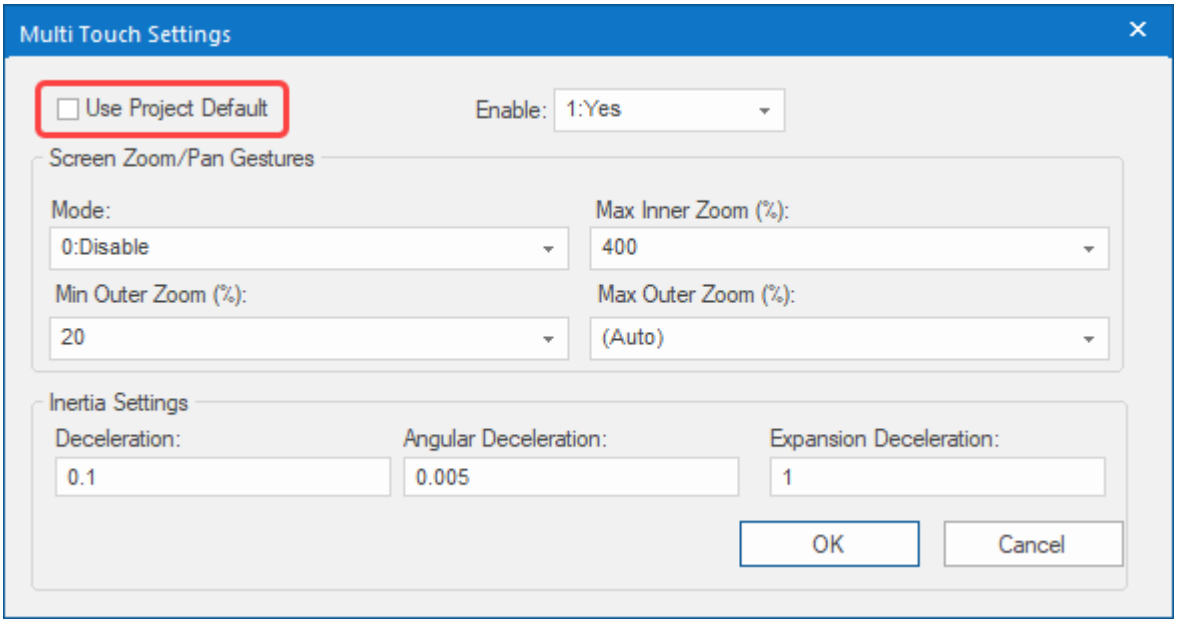
The *Screen Attributes* dialog is displayed.




***Screen Attributes dialog***

2. Click **Multi-Touch Settings**.
The *Multi-Touch Settings* dialog is displayed, with most of the settings disabled because the screen is using the project's default settings.

3. Clear the **Use Project Default** option.




The remaining settings are enabled for configuring.

4. In the **Enable** list, either click an option or type the name of a project tag (Boolean or Integer type).

   This setting determines whether the Multi-Touch features in general (i.e., gestures, screen zoom/pan, inertia, touch events) are enabled for the selected screen style. If you typed the name of a project tag, the value of the tag will control the setting during project run time.

| Option | Description |
|---|---|
| **0:No** | The Multi-Touch features are disabled for the selected screen style. |
| **1:Yes** | The Multi-Touch features are enabled for the selected screen style. |

5. In the **Screen Zoom/Pan Gestures** area, configure the zoom settings.
   a) In the **Mode** box, either select an option or type the name of a project tag (Integer type).

      This setting determines the zoom/pan mode for the selected screen style. If you type the name of a project tag, the value of that tag will determine the mode during project run time.

| Option | Description |
|---|---|
| **0:Disable** | Zoom/Pan is disabled for the selected screen style. |
| **1:Inner** | Inner Zoom/Pan is enabled for the selected screen style. Zooming changes the scale of the screen's contents, and panning moves the viewable area within the screen's border. The screen itself does not change size or position. |
| **2:Outer** | Outer Zoom/Pan is enabled for the selected screen style. In practice, this is more like Resize/Move: zooming changes the size of the entire screen (automatically scaling the screen's contents to fit), and panning moves the screen in relation to the other open screens.<br><br>Please note this mode is not directly supported on Mobile Access. However, selecting the **Enable Screen View Zoom** option (in the Mobile Access Configuration settings) produces essentially the same effect for all screens. For more information, see Configure the global settings for all areas on page 790. |

      For more information, see Using multi-touch gestures in project screens on page 358.

b) In the **Max Inner Zoom (%)** box, either select an option or type the name of a project tag (Integer type).

This will be the maximum magnification allowed for the viewable area inside the project screen, when **Mode** is set to **Inner**. If you type the name of a project tag, the value of that tag will determine the zoom during project run time. Values less than 100 (i.e., 100%) and greater than 1000 (i.e., 1000%) will be ignored.

c) In the **Min Outer Zoom (%)** box, either select an option or type the name of a project tag (Integer type).

This will be the minimum size allowed for the project screen (as a percentage of size specified in the screen attributes), when **Mode** is set to **Outer**. If you type the name of a project tag, the value of that tag will determine the zoom during project run time. Values less than 20 (i.e., 20%) and greater than 100 (i.e., 100%) will be ignored.

d) In the **Max Outer Zoom (%)** box, either select an option or type the name of a project tag (Integer type).

This will be the maximum size allowed for the project screen (as a percentage of the size specified in the screen attributes), when **Mode** is set to **Outer**. The default option is **(Auto)**, which means that the maximum screen size will be equal to the size of the display on which the project is viewed. If you type the name of a project tag, the value of that tag will determine the zoom during project run time. Values less than 100 (i.e., 100%) and greater than 1000 (i.e., 1000%) will be ignored.

6. In the **Inertia Settings** area, configure the deceleration values for the different types of movement.

All values are in pixels per second.

a) In the **Deceleration** box, either type a value or type the name of a project tag (Real type) that contains the value.

This value controls the deceleration from "slide to pan" and "slide to move" gestures.

b) In the **Angular Deceleration** box, either type a value or type the name of a project tag (Real type) that contains the value.

This value controls the deceleration from "turn to rotate" gestures.

c) In the **Expansion Deceleration** box, either type a value or type the name of a project tag (Real type) that contains the value.

This value controls the deceleration from "pinch/stretch to resize" and "pinch/stretch to zoom" gestures.

For a detailed explanation of how inertia is used in multi-touch gestures, see "Inertia Mechanics" at the Microsoft Developer Network website: msdn.microsoft.com/library/dd562169.aspx

7. Click **OK** to save the settings and close the dialog.

### *About the different types of multi-touch gestures*

This section describes the different types of multi-touch gestures and how they can be used in your project.

The gestures themselves — swipe, slide, pinch, stretch, and so on — are a standard part of many operating systems today, so you are probably already familiar with using them on a tablet or smartphone. And even if you are not, illustrations of the gestures are provided in this section.

What this section describes is which gestures can be used in which areas of your project. For example, the same "pinch" and "stretch" gestures can be used to resize a project screen, to resize a screen object with the Resize animation, or even to navigate through a trend graph.

Please note that for the purposes of this documentation, "multi-touch gesture" almost always means a gesture using two fingers. There are some exceptions, such as using a one-finger swipe to select cells in a Grid object, but those exceptions will be descibed in detail in their respective sections.

#### USING MULTI-TOUCH GESTURES IN PROJECT SCREENS

You can use multi-touch gestures to either zoom-and-pan or resize-and-move a project screen during project run time, depending on how the screen is configured.

Specifically, it depends on whether **Zoom/Pan Mode** for a given screen is set to **Inner** or **Outer**. For more information, see About the Multi-Touch settings for project screens on page 352.

> **Note:** For these gestures to work, at least two fingers must be touching the same project screen. If only one finger is inside the screen and the others are outside it, the touch input will be ignored.

### Inner Zoom/Pan

When Inner Zoom/Pan is enabled for a project screen, you can use two-finger gestures to zoom and pan the contents of that screen. Simply touch two fingers to any part of the screen, and then either pinch and stretch to zoom or slide to pan. Zooming changes the scale of the screen's contents, and panning moves the viewable area within the screen's border. The screen itself does not change size or position in relation to the other open screens.

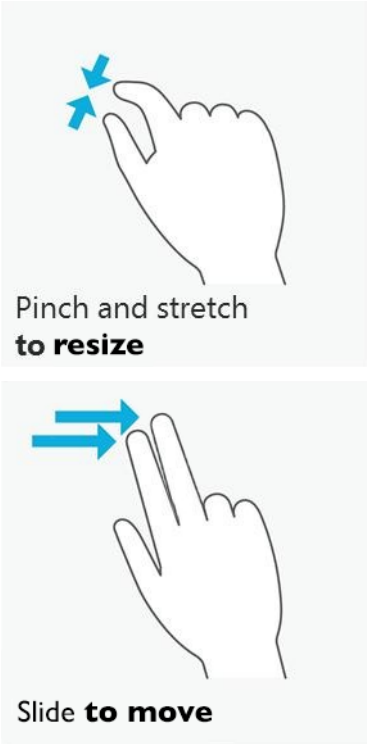
You can use Inner Zoom/Pan to get a closer look at part of a project screen. For example, if a large external image such as a photo or illustration is displayed in a small inset screen, you can use Inner Zoom/Pan to manipulate the image within that inset. Also, if the **Auto Screen Scaling** option is selected in your project settings and some screens are downscaled so much that they become illegible, you can use Inner Zoom/Pan to improve the view of those screens.

These gestures are **not** supported on Mobile Access.

### Outer Zoom/Pan

When Outer Zoom/Pan is enabled for a project screen, you can use two-finger gestures to resize and move that screen within the viewer window. Simply touch your fingers to any part of the screen, and then either

pinch and stretch to resize or slide to move. Zooming changes the size of the entire screen (automatically scaling the screen's contents to fit), and panning moves the screen in relation to the other open screens.




You can use Outer Zoom/Pan to change the layout of all open screens, just as you would arrange windows on the Windows desktop.

These gestures are **not** supported on Mobile Access.

> **Note:** As an alternative to Outer Zoom/Pan, you can enable the **Resizing** border for a project screen. That will also make the screen resizable and movable within the viewer window, although it adds a Windows-style border (including title bar) around the screen. For more information, see Screen Attributes on page 231.

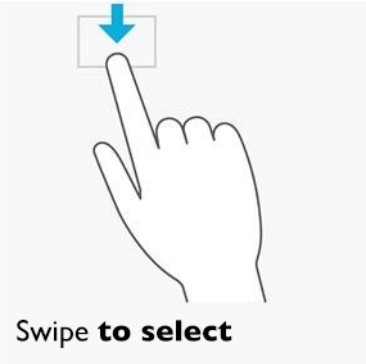
### USING MULTI-TOUCH GESTURES IN DATA OBJECTS

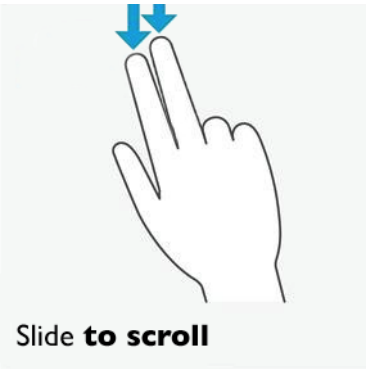
You can use multi-touch gestures to manipulate Alarm/Event, Trend, and Grid objects during project run time.

> **Note:** For these gestures to work, all fingers must be touching the same screen object. If only one finger is inside the object and the others are outside it, then the touch input will be ignored.

### Alarm/Event Control object

When Multi-Touch is enabled for a project screen, then any Alarm/Event Control object in that screen can be manipulated with multi-touch gestures. Specifically, you can swipe with one finger to select items in the list of alarms/events, and you can slide with one or two fingers to scroll through the list.




These gestures are **not** supported on Mobile Access.

### Trend Control object

When Multi-Touch is enabled for a project screen, then any Trend Control object in that screen can be manipulated with multi-touch gestures. Specifically, you can either pinch/stretch to zoom or slide to pan the viewable area of the trend.

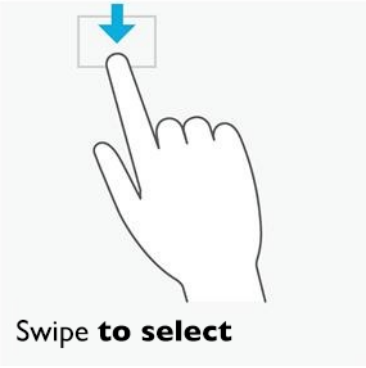
Zooming changes the period and scale of the trend (i.e., the X and Y axes) just as if you clicked any of the **Zoom** tools on the Trend Control object's toolbar. Consequently, if you click **Cancel Zoom** on the toolbar, then any zooming done by your gestures will be canceled and the period and scale will be reset.

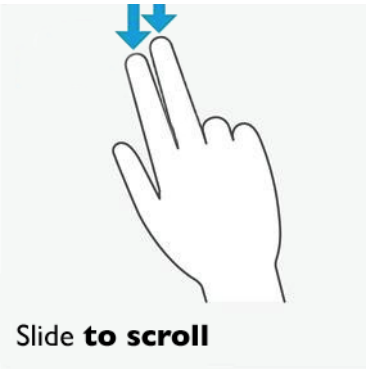
Panning works only when the Trend Control object is configured to show historical data.

These gestures are **not** supported on Mobile Access.

### Grid object

When Multi-Touch is enabled for a project screen, then any Grid object in that screen can be manipulated with multi-touch gestures. Specifically, you can swipe with one finger to select cells in the grid, and you can slide with two fingers to scroll/pan the viewable area of the grid.



These gestures are **not** supported on Mobile Access.

## USING MULTI-TOUCH GESTURES IN OBJECT ANIMATIONS

You can use multi-touch gestures to trigger Position, Resize, and Rotation animations during project run time.

> **Note:** For these gestures to work, all fingers must be touching the same screen object. If only one finger is inside the object and the others are outside it, then the touch input will be ignored.

### Position animation

When Multi-Touch is enabled for a project screen, any object with a Position animation in that screen can be manipulated with multi-touch gestures. Specifically, you can slide with one or more fingers to move the object.

This is essentially the same as when Multi-Touch is disabled, of course, because the purpose of the Position animation is to make the object movable. In this case, the primary benefit of enabling Multi-Touch is inertia, which makes the movement of the object more natural.

This gesture is supported on Mobile Access.

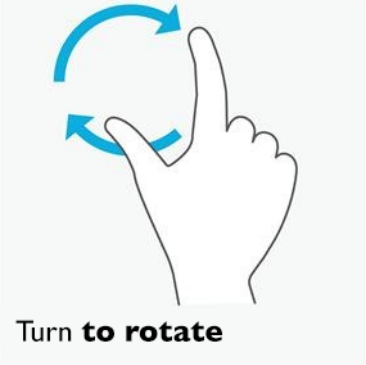
### Resize animation

When Multi-Touch is enabled for a project screen, any object with a Resize animation in that screen can be manipulated with multi-touch gestures. Specifically, you can pinch and stretch with two fingers to resize the object.

This gesture is **not** supported on Mobile Access.

### Rotation animation

When Multi-Touch is enabled for a project screen, you can use a two-finger gesture to "grip" and turn any object with a Rotation animation in that screen. Simply touch your fingers to the object and turn it.




This gesture is **not** supported on Mobile Access.

## *About Touch Events*

Touch Events are predefined VBScript sub-routines that you can add to screen objects and project screens to create custom touch behaviors.

These Touch Events are based directly on the Windows Touch API — specifically, on the **`_IManipulationEvents`** interface and the `ManipulationStarted`, `ManipulationDelta`, and `ManipulationCompleted` methods. For more information about these methods, go to: msdn.microsoft.com/library/dd562197

The Windows Touch API passes the raw input from up to 10 touch points to the project runtime, and then the project runtime interprets that input and passes it to your project as Touch Events.

In practice, Touch Events are essentially the same as any other VBScript interface in your project. You select a screen object or project screen, add the appropriate Touch Event depending on when you want the script to be executed (e.g., when the user starts or stops touching), and then develop the script to do whatever you want. The only real differences between Touch Events and the other VBScript interfaces are: Touch Events can only be executed as sub-routines, which means that they do not return values; and they can only receive the selected touch input data as arguments. If you can work within these guidelines, however, you can use Touch Events to develop custom touch behaviors far beyond the standard gestures.

Touch Events are **not** supported on Mobile Access.

## ADD A TOUCH EVENT TO A SCREEN OBJECT

Add a Touch Event to a screen object in order to process touch input on that object.

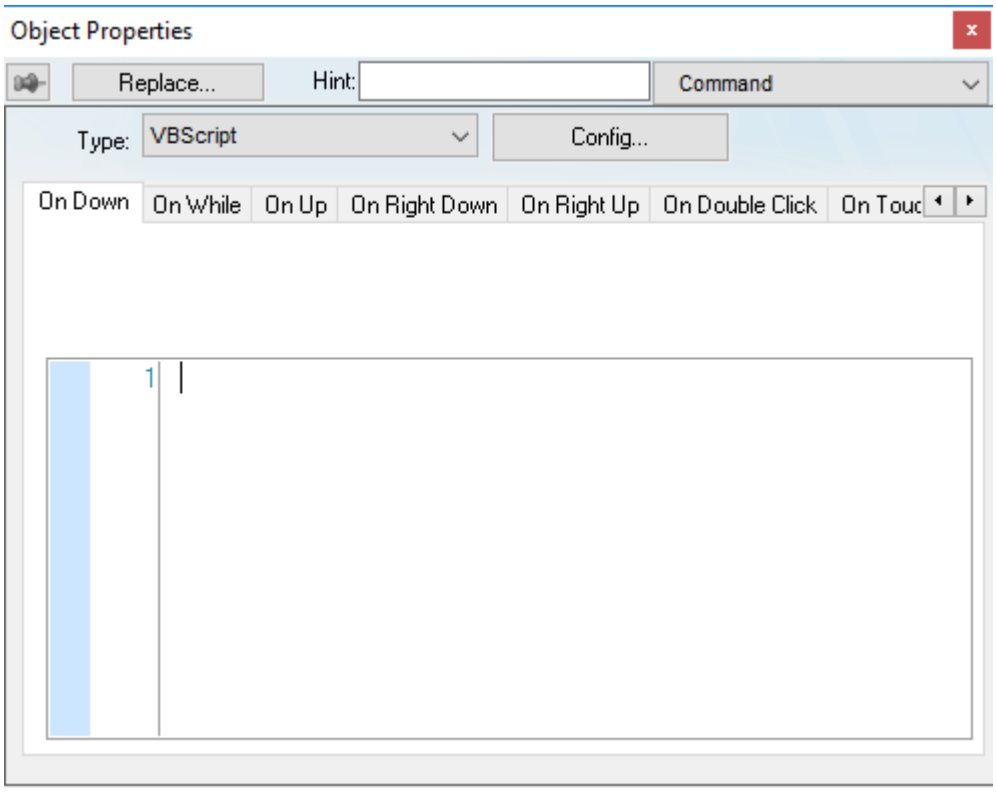
This task is a supplement to other topics that discuss how to use multi-touch gestures in your project. You must have the Multi-Touch feature enabled, either for your entire project or for a specific project screen, and you must have either Inner or Outer mode selected. For more information, see About the Multi-Touch settings for project screens on page 352.

Also, this task assumes you have already created the screen object to which you want to add the Touch Event, and it begins from that point.

Touch Events are added to a screen object through the Command animation.

To add a Touch Event to a screen object:

1. Click the screen object to which you want to add the Touch Event.
   The screen object is selected.
2. On the **Draw** tab of the ribbon, in the **Animations** group, click **Command**.
   The Command animation is added to the selected screen object.
3. Double-click the screen object.
   The *Object Properties* dialog is displayed.
4. If the Command animation properties are not already displayed, then in the properties list (in the top-right corner of the dialog), click **Command**.
5. If the VBScript event tabs are not already displayed, then in the **Type** list, click **VBScript**.




***VBScript event tabs in the Command animation properties***

6. In the VBScript event tabs, click the tab for the Touch Event that you want to add.
   You might need to use the arrow buttons to scroll left and right through the tabs.

| Tab | Description |
|---|---|
| On Touch | Process the raw data from the touch input. The sub-routine is continuously executed while the user touches the screen object. |
| On Touch Start | Perform an action once, when the user starts touching the screen object. |
| On Touch Delta | Perform an action each time the user moves their fingers on the screen object. |
| On Touch Completed | Perform an action once, when the user stops touching the screen object. |

7. Click in the text box below, and then type your VBScript code.

At this point, you can develop the Touch Event just as you would develop any other VBScript sub-routine in your project. For more information, see Overview of VBScript on page 1204.

The OnTouch sub-routines each have predefined parameters that are described in subsequent topics in this section. Do not edit those parameters; they receive data from the touch input and then make it available for use in your code.

## ADD A TOUCH EVENT TO A PROJECT SCREEN

Add a Touch Event to a project screen in order to process touch input on the screen in general, rather than on a specific object in the screen.

This task is a supplement to other topics that discuss how to use multi-touch gestures in your project. You must have the Multi-Touch feature enabled, either for your entire project or for a specific project screen, and you must have either Inner or Outer mode selected. For more information, see About the Multi-Touch settings for project screens on page 352.

Also, this task assumes you already have the Screen worksheet open for editing, and it begins from that point.

Touch Events are added to a project screen through the Screen Script worksheet.

To add a Touch Event to a project screen:

1. Do one of the following:
   - On the **Draw** tab of the ribbon, in the **Screen** group, click **Script**; or
   - Right-click anywhere in the screen worksheet, and then click **Screen Script** on the shortcut menu.

   The screen's associated Screen Script worksheet is opened for editing.
2. Right-click anywhere in the worksheet, and then point to **Add Touch Event** on the shortcut menu. A sub-menu of the available Touch Events is displayed.
3. On the sub-menu, click the Touch Event that you want to add.

| Option | Description |
|---|---|
| Sub Screen_OnTouch | Process the raw data from the touch input. The sub-routine is continuously executed while the user touches the project screen. |
| Sub Screen_OnTouchStart | Perform an action once, when the user starts touching the project screen. |
| Sub Screen_OnTouchDelta | Perform an action each time the user moves their fingers on the project screen. |
| Sub Screen_OnTouchCompleted | Perform an action once, when the user stops touching the project screen. |
| Sub Screen_OnSwipeRight | Perform an action once, when the user swipes horizontally from left to right on the project screen. |
| Sub Screen_OnSwipeLeft | Perform an action once, when the user swipes horizontally from right to left on the project screen. |

| Option | Description |
|---|---|
| **Sub Screen_OnSwipeDown** | Perform an action once, when the user swipes vertically from top to bottom on the project screen. |
| **Sub Screen_OnSwipeUp** | Perform an action once, when the user swipes vertically from bottom to top on the project screen. |

> **Tip:** "Swipe" means to quickly move your finger across the screen in a specific direction (up, down, left, right).

The Touch Event is inserted as a VBScript sub-routine in the worksheet.

4. Click in the sub-routine that you just inserted, and then type your VBScript code.

At this point, you can develop the Touch Event just as you would develop any other VBScript sub-routine in your project. For more information, see Overview of VBScript on page 1204.

The OnTouch sub-routines each have predefined parameters that are described in subsequent topics in this section. Do not edit those parameters; they receive data from the touch input and then make it available for use in your code.

The OnSwipe sub-routines do not have any parameters. They are executed as-is when the corresponding events occur.

## ONTOUCH

Use the sub-routine `OnTouch` in VBScript to process the raw touch point data that are provided while the user touches the project screen or screen object.

### Syntax

```
Sub OnTouch(arX,arY,arIDs,arFlags,arMask,arTime,arXContacts,arYContacts)
    …
End Sub
```

**arX**

An array of integer values, from **arX(0)** to **arX(*n*)**, providing the x-coordinates (in pixels from the left of the screen) of the currently active touch points.

**arY**

An array of integer values, from **arY(0)** to **arY(*n*)**, providing the y-coordinates (in pixels from the top of the screen) of the currently active touch points.

**arIDs**

An array of integer values, from **arIDs(0)** to **arIDs(*n*)**, providing the unique identifiers of the currently active touch points. Each discrete touch point receives its own identifier, even if it is the same finger touching, then lifting, then touching again. These identifiers are incremented from when the device is turned on, and they include all touches captured by the operating system, not just those captured by your project during run time.

**arFlags**

An array of integer values, from **arFlags(0)** to **arFlags(*n*)**, where each value is a set of bit flags that specify various aspects of touch point press, release, and motion.

For more information about the bit flags and their possible values, go to "TOUCHINPUT structure" on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317334.aspx

**arMask**

An array of integer values, from **arMask(0)** to **arMask(*n*)**, where each value is a set of bit flags that specify which of the optional parameters (i.e., arTime, arXContacts, arYContacts) contain

valid information. The availability of valid information is device-specific; for example, for the parameter arTime, some devices provide only the time elapsed since the device was turned on, rather than the actual system time.

For more information about the bit flags and their possible values, go to "TOUCHINPUT structure" on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317334.aspx

**arTime**

An array of integer values, from **arTime(0)** to **arTime(*n*)**, providing the timestamps (in milliseconds) of the currently active touch points.

**arXContacts**

An array of integer values, from **arXContacts(0)** to **arXContacts(*n*)**, providing the widths (in hundredths of a pixel) of the contact areas of the currently active touch points. The contact area of a touch point is the area actually touched by the user's fingertip.

**arYContacts**

An array of integer values, from **arYContacts(0)** to **arYContacts(*n*)**, providing the heights (in hundredths of a pixel) of the contact areas of the currently active touch points. The contact area of a touch point is the area actually touched by the user's fingertip.

### Returned value

This is a sub-routine (as opposed to a function) in VBScript, so it does not return any value.

### Notes

This sub-routine is based on the WM_TOUCH system message and the associated TOUCHINPUT data structure in the Windows API. For more information, go to "Windows Touch Input" on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317321.aspx

The sub-routine is executed continuously while the user is touching the project screen or screen object. There are no delta or cumulative values, so there is nothing to reset when the manipulation is completed. These are the raw data provided by the Windows API.

You are not required to use the received parameters in your code. They simply make the raw touch input data available to you, for you to use (or not) as you deem necessary.

In all of the parameters described above, the array elements represent the individual touch points on the screen, in the order that the user actually touches the screen. The first array element (position 0) is the first touch point, the second array element (position 1) is the second touch point, and so on up to the maximum number of touch points supported by the device.

Please note that the arrays are dynamically resized to fit to the current number of active touch points. In other words, the array elements do not exist until the user's fingers actually touch the screen and the corresponding touch points are added, and the array elements are subsequently eliminated when the touch points are removed. This can make it difficult to reference the array elements in your project unless you include the following code (or something similar) in the sub-routine:

```
n = UBound(arX)

For i = 0 to n
    $TouchX[i] = arX(i)
    $TouchY[i] = arY(i)
    $TouchID[i] = arIDs(i)
    $TouchTime[i] = arTime(i)
Next
```

The function **UBound** measures the current size of arX (although any of the parameters may be used), and then the **For** loop copies the values to appropriately named tag arrays (e.g., **TouchX**, **TouchY**) in your project tags database. Once this is done, you can reference the tag arrays rather than the parameters.

Unlike the parameters, the tag arrays are not dynamically resized, so garbage values may be left in the higher array positions when touch points are removed. To clean out those garbage values, you might also include the following code (or something similar) in the sub-routine:

```
s = $TouchX->Size

For i = (n+1) to s
    $TouchX[i] = 0
    $TouchY[i] = 0
    $TouchID[i] = 0
    $TouchTime[i] = 0
Next
```

By this time, you may have noticed that there is no graceful way to handle the elimination of array elements from anything other than the highest array position. If the user touches the screen with two fingers and then lifts their second finger, the second element of the array (position 1) is eliminated without issues. But if the user touches the screen with two fingers and then lifts their first finger, the first element of the array (position 0) is eliminated and the second element (position 1) becomes the first element (position 0).

You can use the unique identifiers provided by arIDs, rather than the array positions that will change as the arrays are dynamically resized, to handle specific touch points over time. The exact method for doing that, however, depends on how you develop the rest of your project and therefore is beyond the scope of this documentation.

## ONTOUCHSTART

Use the sub-routine `OnTouchStart` in VBScript to perform an action when the user starts touching the project screen or screen object.

### Syntax

```
Sub OnTouchStart(x,y)
    …
End Sub
```

**x**

The starting x-coordinate (in pixels from the left of the screen) of the first touch point.

**y**

The starting y-coordinate (in pixels from the top of the screen) of the first touch point.

### Returned value

This is a sub-routine (as opposed to a function) in VBScript, so it does not return any value.

### Notes

This sub-routine is based on the method ManipulationStarted in the Windows API. For more information, go to "Windows Touch Input" on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317321.aspx

The sub-routine is executed once when the manipulation is started — that is, when the first touch point is added to the project screen or screen object. Additional touch points after the first do not trigger this sub-routine.

You are not required to use the received parameters in your code. They simply make the raw touch input data available to you, for you to use (or not) as you deem necessary.

### ONTOUCHDELTA

Use the sub-routine `OnTouchDelta` in VBScript to perform an action each time the user manipulates the project screen or screen object.

### Syntax

```
Sub
 OnTouchDelta(x,y,deltaX,deltaY,deltaScale,deltaExpansion,deltaRotation,cumulativeX,cumulativeY,cum
    …
End Sub
```

**x**

The current x-coordinate (in pixels from the left of the screen) of the first touch point.

**y**

The current y-coordinate (in pixels from the top of the screen) of the first touch point.

**deltaX**

The change (in pixels) between the previous x-coordinate and the current x-coordinate of the first touch point.

**deltaY**

The change (in pixels) between the previous y-coordinate and the current y-coordinate of the first touch point.

**deltaScale**

The change (as a percentage) in the distance between the first and second touch points.

**deltaExpansion**

The change (in pixels) in the distance between the first and second touch points.

**deltaRotation**

The change in the angle of rotation (in radians) indicated by the first and second touch points.

**cumulativeX**

The total change (in pixels) between the starting x-coordinate and the current x-coordinate of the first touch point.

**cumulativeY**

The total change (in pixels) between the starting y-coordinate and the current y-coordinate of the first touch point.

**cumulativeScale**

The total change (as a percentage) in the distance between the first and second touch points, from the start of the manipulation.

**cumulativeExpansion**

The total change (in pixels) in the distance between the first and second touch points, from the start of the manipulation.

**cumulativeRotation**

The total change in the angle of rotation (in radians) indicated by the first and second touch points, from the start of the manipulation.

**inertiaEnabled**

A boolean value indicating whether inertia is enabled for the project screen or screen object.

### Returned value

This is a sub-routine (as opposed to a function) in VBScript, so it does not return any value.

### Notes

This sub-routine is based on the method ManipulationDelta in the Windows API. For more information, go to "Windows Touch Input" on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317321.aspx

The sub-routine is executed once for each discrete movement in the current manipulation. Changes in position require one touch point. Changes in size and/or rotation require two touch points.

You are not required to use the received parameters in your code. They simply make the raw touch input data available to you, for you to use (or not) as you deem necessary.

## ONTOUCHCOMPLETED

Use the sub-routine `OnTouchCompleted` in VBScript to perform an action when the user stops touching the project screen or screen object.

### Syntax

```
Sub
 OnTouchCompleted(x,y,cumulativeX,cumulativeY,cumulativeScale,cumulativeExpansion,cumulativeRota
    …
End Sub
```

**x**

The ending x-coordinate (in pixels from the left of the screen) of the first touch point.

**y**

The ending y-coordinate (in pixels from the top of the screen) of the first touch point.

**cumulativeX**

The total change (in pixels) between the starting x-coordinate and the ending x-coordinate of the first touch point.

**cumulativeY**

The total change (in pixels) between the starting y-coordinate and the ending y-coordinate of the first touch point.

**cumulativeScale**

The total change (as a percentage) in the distance between the first and second touch points, from the start to the end of the manipulation.

**cumulativeExpansion**

The total change (in pixels) in the distance between the first and second touch points, from the start to the end of the manipulation.

**cumulativeRotation**

The total change in the angle of rotation (in radians) indicated by the first and second touch points, from the start to the end of the manipulation.

### Returned value

This is a sub-routine (as opposed to a function) in VBScript, so it does not return any value.

### Notes

This sub-routine is based on the method ManipulationCompleted in the Windows API. For more information, go to "Windows Touch Input" on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317321.aspx

The sub-routine is executed once when the manipulation is completed — that is, when the last touch point is removed from the project screen or screen object.

You are not required to use the received parameters in your code. They simply make the raw touch input data available to you, for you to use (or not) as you deem necessary.

## Import a Studio XML Screen

Use the Import Wizard to import a Studio XML Screen, which is an external text file created with EcoStruxure Machine SCADA Expert 2020's custom XML schema.

Before you begin this task, you must have a properly formatted Studio XML Screen file that you can import.

A Studio XML Screen file contains the same information as a regular screen file. It is simply formatted as human-readable XML instead of binary data, which makes it more flexible and portable.

XML files can be manually created in any text editor, of course, but our intent is to enable you to quickly and programmatically create large numbers of screens in Microsoft Visual Studio using our custom XML schema and its associated C# API. This often requires additional training and support, so the schema and API are not included in the standard installation of EcoStruxure Machine SCADA Expert 2020. Instead, the schema, API, documentation, sample files, and support contract can be purchased as a separate toolkit. For more information, please contact your software vendor.

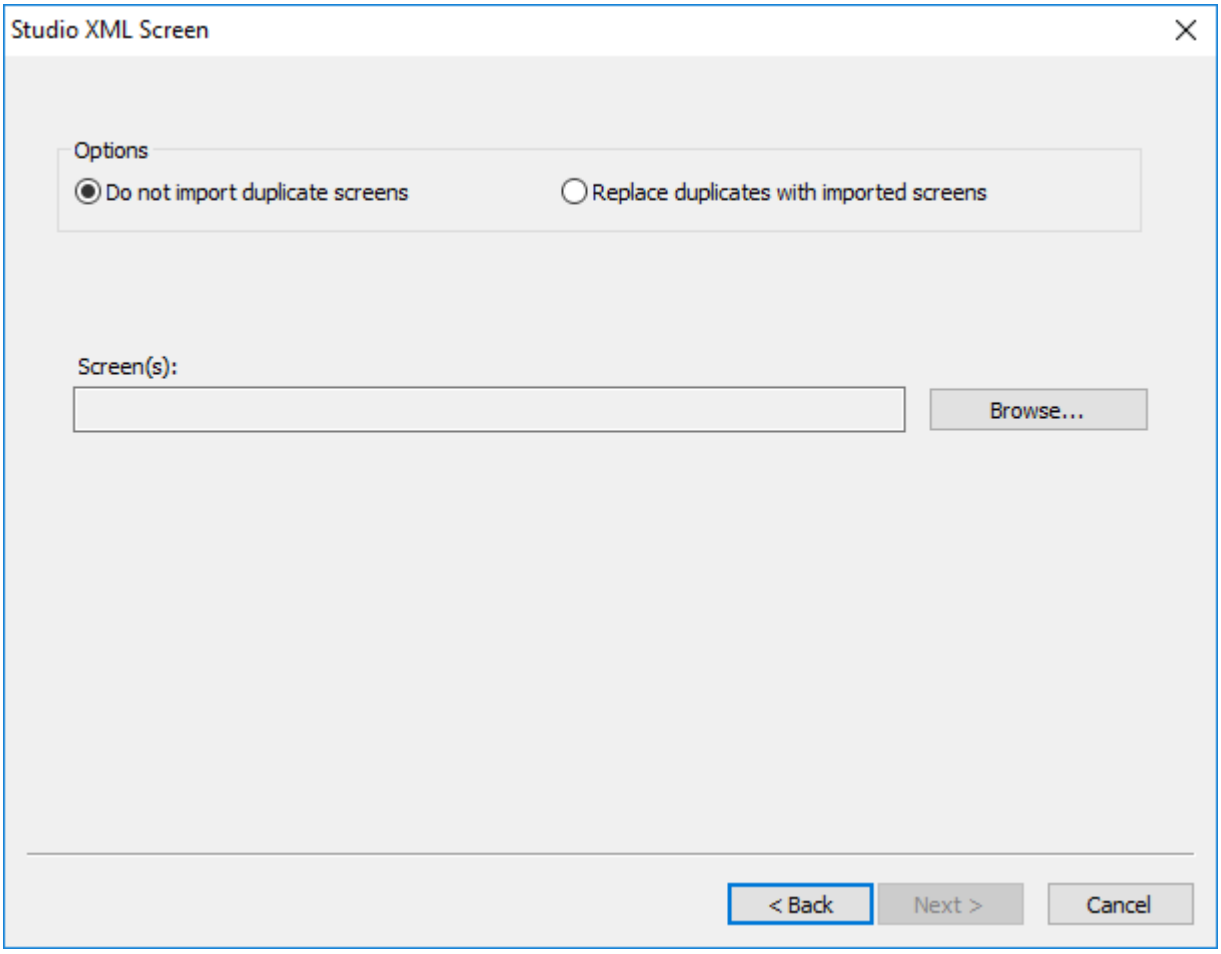
Once you have created your screens, you can use the Import Wizard to batch import them into your EcoStruxure Machine SCADA Expert project.

> **Tip:** You can also use the function `ImportXML` to import Studio XML Screen files during run time.

To import one or more Studio XML Screens:

1. On the **Home** tab of the ribbon, in the **Tools** group, click **Import Wizard**.
   The *Import Wizard* dialog box is displayed.
2. In the **Source Type** list, click **Studio XML Screen**, and then click **Next**.

The next step of the import wizard is displayed.




***Selecting the screens to import***

3. Under **Options**, choose whether imported screens should automatically replace existing screens in your project.

   Screens are considered to be duplicates if they have the same file name. For example, `Objects.xml` and `Objects.scc` would be duplicates.

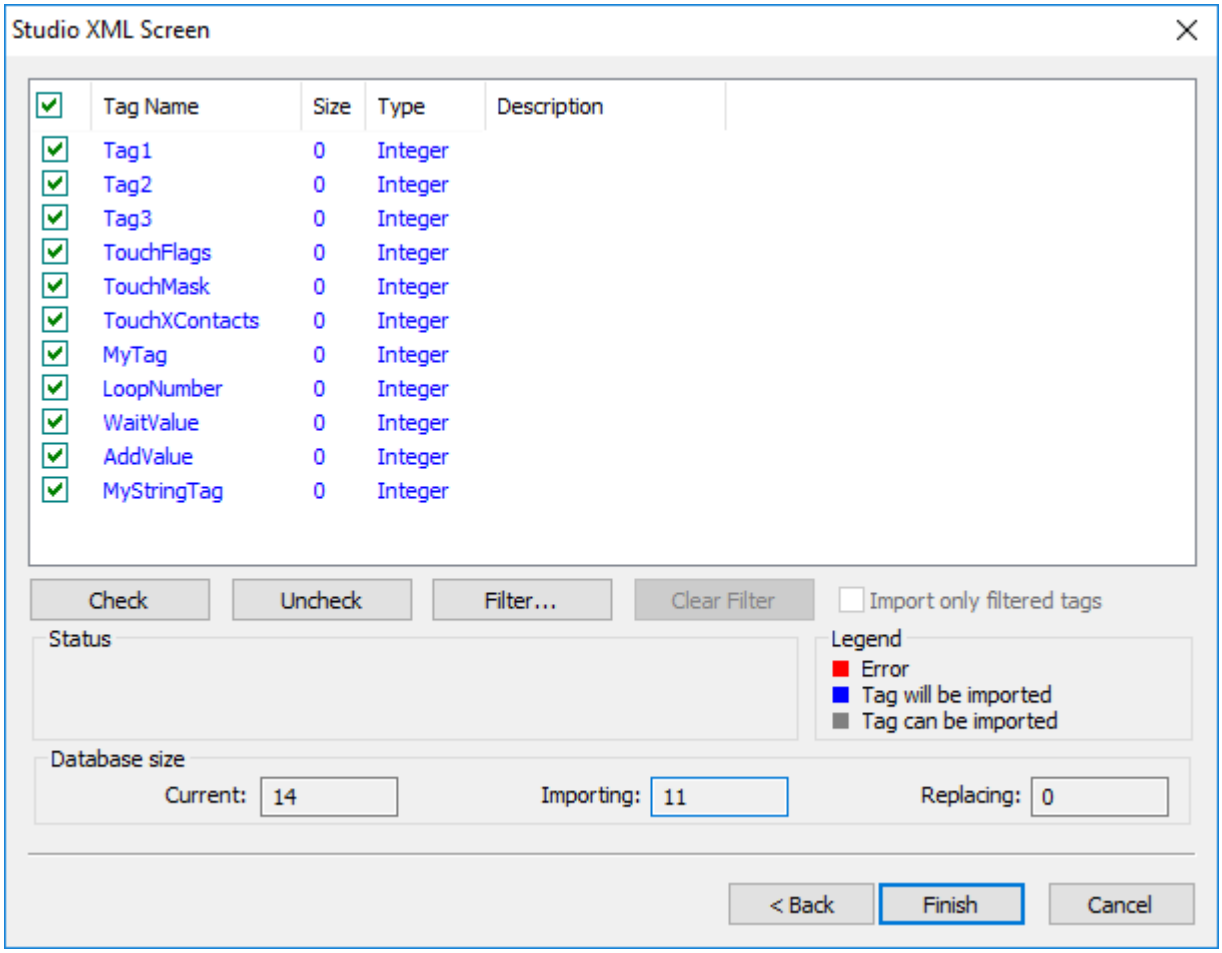
   - If you do not want the imported screens to replace existing screens in your project, select **Do not import duplicate screens**. A warning will be displayed for each duplicate that you try to import.
   - If you want the imported screens to automatically replace existing screens in your project, select **Replace duplicates with imported screens**.

4. Click **Browse**.
   A standard *Open* dialog box is displayed.
5. Use the file browser to locate and select the Studio XML Screen files that you want to import, and then click **Open**.

   You can Ctrl-click and Shift-click to select more than one file.

   The selected file(s) are displayed in the **Screen(s)** box.
6. Click **Next**.

The selected screens are processed, and then the next step of the import wizard is displayed showing the project tags that are included in the selected screens.




***Selecting the project tags to import***

7. In the list of project tags, select the tags that you want to import with the screens:
   - For each tag in the list, select or clear the check box to the left.
   - To select all of the check boxes, click **Check**. To clear all of the check boxes, click **Uncheck**.
   - To filter the list of tags, click **Filter** and then configure filter strings for one or more columns. You can use wildcard characters (`*` and `?`) in the filter strings.
8. Click **Finish**.

The screens and included tags are imported into your project. Also, the screens are automatically published for Thin Clients and Mobile Access — i.e., the corresponding `*.HTML`, `*.SCC`, and `*.SSMA` files are automatically generated and saved in your project's Web folder, so the imported screens should be immediately available for you to select.

# Alarms, Events, and Trends

The Alarm and Trend tasks are used to log historical data, and the Alarm/Event and Trend Control objects are used to display historical data on a project screen.

These two features are normally used together, but they do not need to be; project data may be logged without being displayed during runtime, and the data displayed during runtime may be taken from outside the project.

## Alarm worksheet

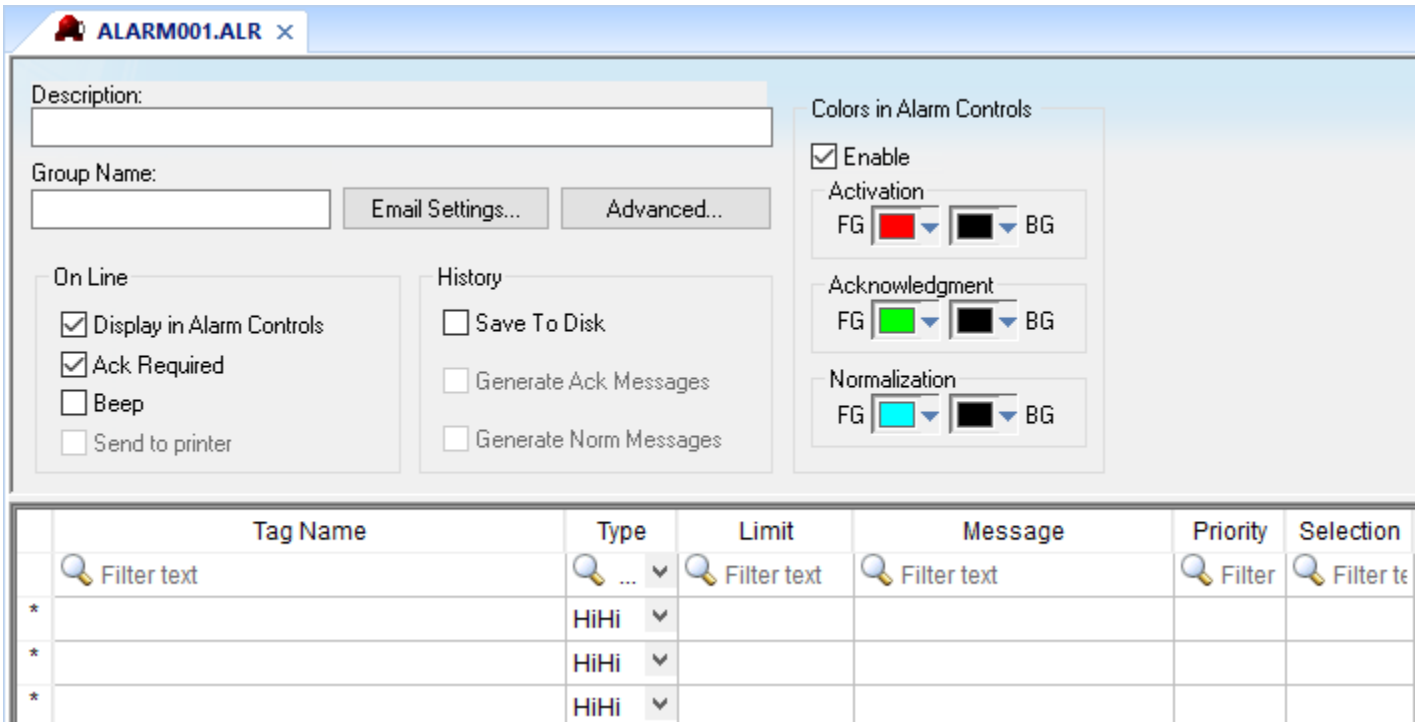
The Alarms folder enables you to configure alarm groups and tags related to each group. The Alarm worksheet defines the alarm messages generated by the project. The primary purpose of an alarm is to inform the operator of any problems or abnormal condition during the process so he can take corrective action(s).

The Alarm worksheet is executed by the Background Task module (see Runtime Tasks on page 136). It handles the status of all alarms and save the alarm messages to the history, if configured to do so, but it does not display the alarm messages to the operator; the Alarm/Event Control screen object, available on the Graphics tab of the ribbon, must be created and configured in a screen in order to display alarms.

To create a new Alarm worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Alarm**;
- Right-click the **Alarms** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the File menu, click the **File** tab, and then select **Alarm Worksheet**.

To edit an existing Alarm worksheet, double-click it in the Project Explorer.




***Alarm worksheet***

You can create multiple Alarm groups (worksheets) and each group can be configured with independent settings, such as message colors, history log enabled/disabled, and so forth.

Each Alarm worksheet is composed of two areas:

- **Header:** Settings applied to all tags and alarms configured in the same alarm group. These settings allow you to configure the formatting of the message and the actions that must be triggered based on alarm events (e.g., print alarms, send alarms by email, and so forth). For more information, see Header Settings.
- **Body:** Configure alarm messages and associate them to conditions linked to tags. For more information, see Body Settings.

> **Note:**

The Alarm task has been modified to avoid automatically acknowledging alarms by another alarm. For example, the Hi (Lo) alarm should not be automatically acknowledged when the HiHi (LoLo) alarm becomes active. To enable the previous behavior, set the following key in your project (.APP) file:

```
[Alarm]
UseLegacyPriorityAck=1
```

### *Alarm Worksheet Header*

The following table describes the Header settings on an Alarm worksheet:

| Field | Remarks | Syntax |
|---|---|---|
| Description | Description of the alarm group. It is displayed on the workspace. This field is used for documentation only. | Text (up to 80 chars) |
| Group Name | Name of the Alarm group. During runtime, the operator can filter alarms based on the Group Name by the built-in Filters dialog of the Alarm/Event control object. | Text (up to 32 chars) |
| Email Settings | Launches the *Email Settings* dialog, where you can configure the settings for emails sent automatically based on alarm conditions. | Button |
| Advanced | Launches the *Advanced Settings* dialog, where you can configure the settings for emails sent automatically based on alarm conditions. | Button |
| On Line > Display in Alarm Controls | When checked, the alarms are available to be displayed on the Alarm/Event Control object. | Checkbox |
| On Line > Ack Required | When checked, the alarms require acknowledgment. In this case, the alarms are displayed on the Alarm/Event Control object (Online mode) until they are acknowledged AND normalized. | Checkbox |
| On Line > Beep | When checked, the computer keeps beeping while there are alarm(s) to be acknowledged, currently active. | Checkbox |
| On Line > Send to printer | This option has been deprecated because it invokes the DOS print command (`prn`), which has been deprecated in Windows 7 and later. Now, when this option is selected, alarm messages are simply passed to an external batch file as soon as the alarms are created. You can edit the batch file to process the alarm messages as you see fit. It is located in the SCADA program folder at: `Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin\unprint.bat` | Checkbox |
| History > Save to Disk | When checked, the alarm messages are stored in the history log when they become active. | Checkbox |
| History > Generate Ack Messages | When checked, the alarm messages are stored in the history log when they are acknowledged. | Checkbox |
| History > Generate Norm Messages | When checked, the alarm messages are stored in the history log when they become normalized. | Checkbox |
| Colors in Alarm Controls > Enable | When checked, the alarms configured in this group will be displayed with the colors assigned to each alarm state (**Activation**, **Acknowledgement** or **Normalization**), according to the colors configured in the Alarm Group. | Color |
| Colors in Alarm Controls > FG and BG | You can configure the text foreground color (FG) and background color (BG) for the alarms displayed on the Alarms/Events Control object. Each alarm state can be displayed with a different color schema:<br>• **Activation**: Alarm active and not acknowledged<br>• **Acknowledgement**: Alarm active and acknowledged<br>• **Normalization**: Alarm no longer active and not acknowledged. | Color |

## CONFIGURE THE EMAIL SETTINGS FOR AN ALARM WORKSHEET

Your project can automatically send emails to specified addresses when alarm events occur. Configure the email settings for an Alarm worksheet in order to customize the format and frequency of those emails.

Before you begin this task, make sure you have configured an outgoing email server and account for your project. There are two ways to do this:

- Configure the default email settings for your project; or
- Configure a script that calls the `CnfEmail` function at least once. This is typically done in the Startup Script, so that the email settings are configured before anything in the project tries to send email, but it can be done again in other scripts in order to change the settings during project run time.

You do not need to have an email client application (e.g., Microsoft Outlook) installed on the computer or device that hosts the project runtime, because the project runtime can connect directly to the email server. In other words, the project runtime itself acts as an email client.

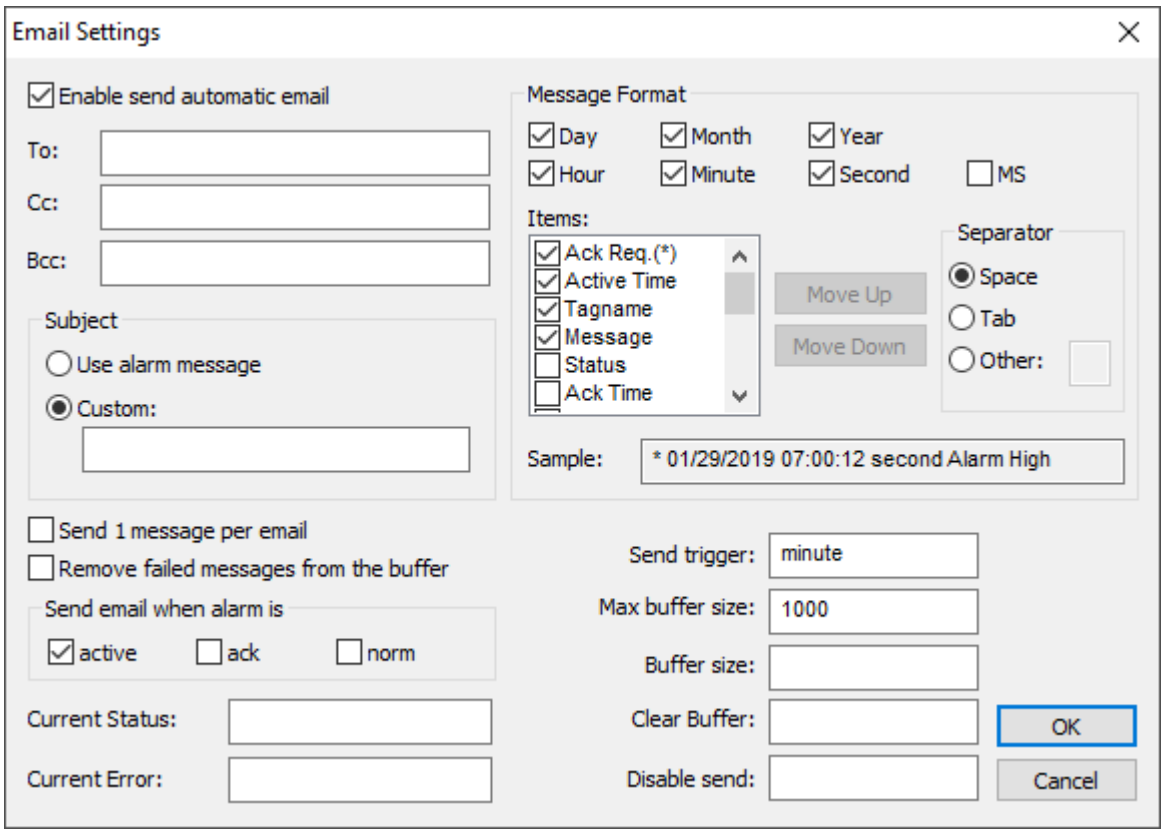
> **Note:** Keep in mind that the email settings for your project, as described above, are separate from the email settings for an Alarm worksheet, as described below.

This task assumes you have created an Alarm worksheet and that worksheet is open in the Screen/ Worksheet Editor. The procedure below begins at that point.

To configure the email settings for an Alarm worksheet:

1. In the header of the Alarm worksheet, click **Email Settings**.
   The *Email Settings* dialog box is displayed. Email is disabled by default for each Alarm worksheet, so most of the settings in this dialog box are not available until email is enabled.
2. Select the **Enable send automatic email** option.

   All of the settings in the dialog box become available.



3. In the **To** box, type the primary email address to which you want send alarm messages. If you want to send to more than one address, insert semicolons (`;`) to separate the addresses.

   You can also send "carbon copies" and "blind carbon copies" to other email addresses. Type those addresses in the **Cc** and **Bcc** boxes, respectively. The same email is sent to all of the addresses at the same time.

   The **To**, **Cc**, and **Bcc** settings each have a limit of 1024 characters. These settings accept string expressions, but when the expressions are evaluated, the resulting values are subject to the same character limit.

4. In the **Subject** area, in the **Custom** box, type the text that you want to use as the subject of each email.

   Alternatively, you can select **Use alarm message** in order to use the actual alarm message as the subject of the email, but this is not recommended because the alarm message is included in the body of the email.

The procedure above describes the minimum that you need to do in order to send email for the Alarm worksheet. When the alarms events that are configured in the worksheet occur during project run time, corresponding alarm messages are added to an internal memory buffer, and then those messages are periodically sent as email until the buffer is empty.

If you want to customize the format and frequency of those emails, review the additional settings described below:

**Send 1 message per email**

Select this option in order to send one alarm message (event) per email, each time sending is triggered (see **Send Trigger** below). Otherwise, all of the messages currently stored in the buffer are sent as a batch in a single email.

This option is automatically selected and cannot be cleared when the **Use alarm message** option is selected for the subject of the email.

**Remove failed messages from the buffer**

Select this option in order to remove alarm messages (events) from the buffer after the project runtime tries to send them, even if there is an error (failure) and the email is not sent successfully. Otherwise, the messages are kept in the buffer until the email is sent successfully, or until the buffer is full (see **Max buffer size** below).

**Send email when alarm is**

Select which alarm events should generate messages:

| Option | Description |
|---|---|
| **active** | When an alarm becomes active. |
| **ack** | When an alarm is acknowledged. |
| **norm** | When an alarm is normalized. |

Each type of event can be selected individually, but that selection applies to all alarms configured in the same worksheet. If you want to generate messages when some alarms but not others become active, separate those alarms into different worksheets.

**Current Status**

You can type the name of a project tag in this box. During project run time, this tag is updated with a numeric value that indicates the status of the latest email:

| Value | Description |
|---|---|
| -2 | Incorrect version of the `INDMail.DLL` library. |
| -1 | The `INDMail.DLL` library is corrupted. |
| 0 | `SendEmailExt` function is not being executed. |
| 1 | Sending email(s). |
| 2 | Last email was sent successfully. |
| 3 | There was an error sending the last email. |

**Current Error**

You can type the name of a project tag (String type) in this box. During project run time, this tag is updated with the error message describing the result of the last email that the project tried to send. The error message is a string, so the project tag must be String type in order to hold the message.

**Message Format**

This interface allows you to customize the format of the alarm messages that are included in the body of the email:

- **Day**, **Month**, **Year**, **Hour**, **Minute**, **Second**, **MS**: The options checked will compose the timestamp for each alarm message. **MS** stands for milliseconds.
- **Items**: The options checked will compose the actual message for each alarm. Use the **Move Up** and **Move Down** buttons to change the order of the items.
- **Separator**: The separator that is inserted between items in the alarm message. This is important if you have set up another program to receive the emails and then automatically parse the messages.

The **Sample** box displays an example of the message format as you customize it.

**Send Trigger**

When the value of this tag/expression changes, the alarm messages (events) currently stored in the buffer are sent according to these email settings. The default trigger is the **`Minute`** tag, which is a system tag that updates each minute of the clock on the computer or device that hosts the project runtime, and it prevents the messages from overloading the email server or "spamming" the email recipients. The default trigger might not be frequent enough to keep up with alarms as they occur, however, especially if you have selected the **Send 1 message per email** option; if there are 10 messages stored in the buffer and only one message is sent per minute, then it will take 10 minutes to send all of the messages. Therefore, you may configure a different tag/expression that can accomodate the actual behavior of your project.

**Max buffer size**

This is the maximum number of alarm messages (events) that can be stored in the buffer. The buffer stores messages on a "First In, First Out" (FIFO) basis, so if new messages are added after the maximum has been reached, then the older messages are removed without being sent. The default maximum is 1000 messages, but you can increase it if the computer or device that hosts the project runtime has sufficient memory.

**Buffer size**

You can type the name of a project tag in this box. During project run time, this tag is updated with the number of alarm messages (events) currently stored in the buffer.

**Clear Buffer**

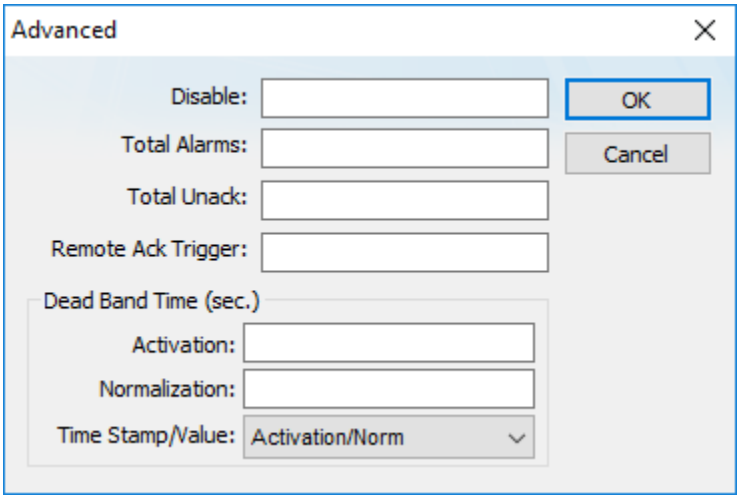
You can type a tag/expression in this box. When its value changes during project run time, all alarm messages (events) currently stored in the buffer are removed and will never be sent.

**Disable send**

You can type a tag/expression in this box. While its value is TRUE (i.e., not 0) during project run time, the sending of email is disabled for this Alarm worksheet. Alarm messages (events) currently stored in the buffer are not sent, regardless of **Send Trigger**, and new messages are not added to the buffer.

When you are done, click **OK** to save the settings and close the dialog box.

### ADVANCED SETTINGS FOR ALARM WORKSHEET




***Alarms Worksheet — Advanced***

The following table describes the Advanced settings on an Alarm worksheet:

| Field | Remarks | Syntax |
|---|---|---|
| Disable | When the value of the tag configured in this is TRUE, all alarms configured in this group are temporarily disabled. This option is useful to disable alarms under special conditions (e.g., during maintenance). | Tag |
| Total Alarms | The tag configured in this field, if any, is updated with the number of alarms from this group, which are currently active. | Tag |
| Total Unack | The tag configured in this field, if any, is updated with the number of alarms from this group, which are currently active AND have not been acknowledged yet. | Tag |
| Remote Ack Trigger | When the tag configured in this field change of value, all active alarms from this group are acknowledged. This option can be used to acknowledge alarms regardless of any action from the operator. | Tag |
| Dead Band Time > Activation | Each alarm must remain continuously in its alarm condition for the period of time specified in this field before becoming active. This option is useful to avoid generating alarms on intermittent conditions (e.g., noise). If this field is left in blank, the alarm becomes active as soon as its condition is true. | Tag or Number |
| Dead Band Time > Normalization | Each alarm must remain continuously out from its alarm condition for the period of time specified in this field before becoming normalized. This option is useful to avoid normalizing alarms on intermittent conditions (e.g., noise). If this field is left in blank, the alarm become normalized as soon as its condition is no longer true. | Tag or Number |
| Dead Band Time > Time Stamp/Value | Each alarm maintains a time stamp of the last significant activity, along with the value of the tag at that time. You can select the type of activity that updates the time stamp:<br>• **Activation/Norm** (default): The time when the dead band ended — that is, when the alarm becomes activated or normalized.<br>• **Last Tag Change**: The time when the value of the tag last changed during the dead band.<br>• **Start Condition**: The time when the dead band started. | Combo |

## *Alarm Worksheet Body*

The following table describes the Body settings on an Alarm worksheet:

| Field | Description | Syntax |
|---|---|---|
| **Tag Name** | Name of the tag associated with the alarm. | Tag |
| **Type** | Type of the alarm:<br>• **HiHi**: Activates the alarm if the tag value is greater than or equal to the specified limit.<br>• **Hi**: Activates the alarm if the tag value is greater than or equal to the specified limit. (For Boolean tags, if the value is 1.)<br>• **Lo**: Activates the alarm if the tag value is less than or equal to the specified limit. (For Boolean tags, if the value is 0.)<br>• **LoLo**: Activates the alarm if the tag value is less than or equal to the specified limit.<br>• **Rate**: Activates the alarm if the tag value changes more than the specified limit in a given period. (For Boolean tags, if the value changes at all.)<br>• **DevP**: Activates the alarm if the tag value is greater than or equal to the tag's deviation set point plus the limit.<br>• **DevM**: Activates the alarm if the tag value is less than or equal to the tag's deviation set point minus the limit.<br>If you select **Rate**, you must also specify the check frequency (e.g., once per minute) in the tag properties. If you select **DevP** or **DevM**, you must also specify the deviation set point in the tag properties. For more information, see Properties of Integer and Real tags on page 171. | Combo-box |
| **Limit** | Limit associated with each alarm.<br>The limits can be modified dynamically during run time, using the tag fields **`HiHiLimit`**, **`HiLimit`**, **`LoLimit`**, **`LoLoLimit`**, **`RateLimit`**, **`DevPLimit`**, and **`DevMLimit`**. For example: **`TagLevel->HiLimit`** | Number |
| **Message** | Message associated to the alarm. The message can be displayed on the Alarm/Event Control object and/or stored in the Alarm History and/or sent by Email, depending on the settings configured in the Header of the Alarm group. | Text and/or {Tag} (up to 256 chars) |
| **Priority** | Priority number associated to the alarm. When displaying alarms on the Alarm/Event Control object , the operator can filter and/or sort the alarms by priority. | Number (from 0 to 255) |
| **Selection** | Alias associated to the alarm (e.g., AreaA, AreaB, etc). When displaying alarms on the Alarm/Event Control object, the operator can filter and/or sort the alarms by their selection value. | Text (up to 7 characters)<br>**Tip:** If you need more characters or a different data type, use a custom field instead. |
| Custom fields | Additional custom fields that will be saved in history. The number of custom fields (up to 10) can be set in Project Settings: Options. | Any |

When you save an Alarm worksheet, only the header settings are saved as part of the worksheet file. All of the alarm configurations that make up the body of the worksheet are actually saved as tag properties. The next time you open that worksheet, the tags database is scanned for all alarm configurations that belong to the worksheet (i.e., the alarm group), and then that information is used to recreate the body of the worksheet. This happens quickly and automatically every time you open the worksheet, so it might seem like you are opening a static file but that is not the case.

You may think of the Alarm worksheet as an editor for those tag properties that are related to alarms. If you use either the *Tag Properties* dialog box or the `TagsDBSetAlarm` function to edit the same properties, the updated alarm configurations will be included in the body of the worksheet the next time you open it. In fact, you can set alarms on tags before you create any Alarm worksheets at all; when you do create the worksheets, they will be automatically populated with alarm configurations according to their group numbers.

You cannot configure more than one alarm of the same type on a given tag, and each alarm configuration cannot belong to more than one group/worksheet.

If you make extensive changes to the tags database after you save an Alarm worksheet, it might not be possible to recreate the body of the worksheet the next time you open it. For example, if you copy all of the tags from the tags database (in Datasheet View) to a spreadsheet program, use that program to sort the tags, and then copy the tags back to the tags database, most or all of the tag properties will be reset in the process.

### SORT OR FILTER THE ROWS IN A WORKSHEET

Sort or filter the rows in a worksheet in order to make it easier to browse the rows or find a specific item.

Before you begin this task, you must have already inserted a worksheet and opened it for editing. You should also be familiar with how sorting and filtering is done in general-purpose spreadsheet applications.

Please note that you can sort or filter rows only in the following types of worksheets:

- The Project Tags, Shared Tags, and System Tags datasheets;
- The Translation Table worksheet;
- All task worksheets except Report and Script, which do not have rows; and
- All communication worksheets.

None of the other worksheets have rows to sort or filter.

Sorting is done alphanumerically, by the selected column, in either ascending (0–9, A–Z) or descending (Z–A, 9–0) order.

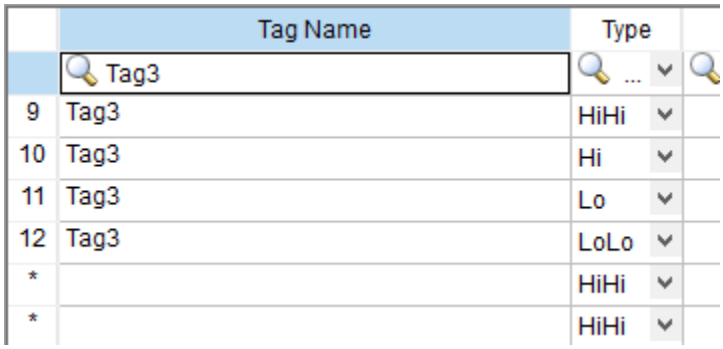
| | Tag Name | Type |
|---|---|---|
| | Filter text | ... |
| 1 | Tag1 | HiHi |
| 2 | Tag1 | Hi |
| 3 | Tag1 | Lo |
| 4 | Tag1 | LoLo |
| 5 | Tag2 | HiHi |
| 6 | Tag2 | Hi |
| 7 | Tag2 | Lo |
| 8 | Tag2 | LoLo |
| 9 | Tag3 | HiHi |
| 10 | Tag3 | Hi |
| 11 | Tag3 | Lo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

***Alarm worksheet rows in their original order***

| | Tag Name | Type |
|---|---|---|
| | Filter text | ... |
| 10 | Tag3 | Hi |
| 2 | Tag1 | Hi |
| 6 | Tag2 | Hi |
| 9 | Tag3 | HiHi |
| 5 | Tag2 | HiHi |
| 1 | Tag1 | HiHi |
| 3 | Tag1 | Lo |
| 11 | Tag3 | Lo |
| 7 | Tag2 | Lo |
| 8 | Tag2 | LoLo |
| 4 | Tag1 | LoLo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

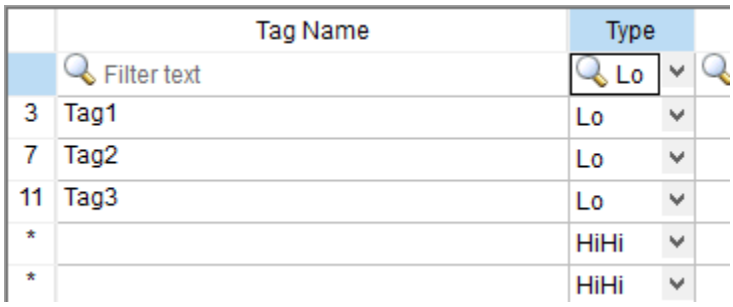
***Alarm worksheet rows sorted by Type***

Filtering is done according to whatever string you enter in the selected column. Only the rows that match the string will be displayed.



| | Tag Name | Type |
|---|---|---|
| | Tag3 | ... |
| 9 | Tag3 | HiHi |
| 10 | Tag3 | Hi |
| 11 | Tag3 | Lo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

***Alarm worksheet rows filtered where Tag Name is "Tag3"***



| | Tag Name | Type |
|---|---|---|
| | Filter text | Lo |
| 3 | Tag1 | Lo |
| 7 | Tag2 | Lo |
| 11 | Tag3 | Lo |
| * | | HiHi |
| * | | HiHi |

***Alarm worksheet rows filtered where Type is "Lo"***

> **Tip:** You can still delete rows while they are sorted or filtered.

To sort or filter rows:

1. To sort the rows, click the header of the column by which you want to sort. Click once to sort in ascending order, and then click again to sort in descending order.

   The current order (i.e., the direction of the sort) is indicated by the arrow to the right of the column name.

   > **Note:** You cannot sort by multiple columns.

2. To undo the sorting and restore the rows to their original order, click the header of the first (numbered) column.
3. To filter the rows, type the string that you want to match in the top (zero) row of the worksheet and then press either **Tab** or **Return**.

   You may include `*` and `?` as wildcard characters in your string:

   - `*` matches any number of characters, including none. For example, `Tag*` would match **`Tag`**, **`Tag3`**, **`Tag34567`**, **`TagA`**, and **`Tag_TEMP`**.
   - `?` matches exactly one character. For example, `Tag?` matches **`Tag3`** and **`TagA`**, while `Tag?????` matches **`Tag34567`** and **`Tag_TEMP`**.

   Also, you may filter by multiple columns. Only the rows that match the filter strings in all columns will be displayed.
4. To undo the filtering and restore the rows to their original order, delete the string that you typed and then press either **Tab** or **Return**.

Please keep in mind that sorting or filtering the rows of a worksheet only helps you to edit that worksheet. It does not change how the worksheet is executed during run time. The rows will be executed in their original numbered order (i.e., the leftmost column) unless you actually move or delete a row.

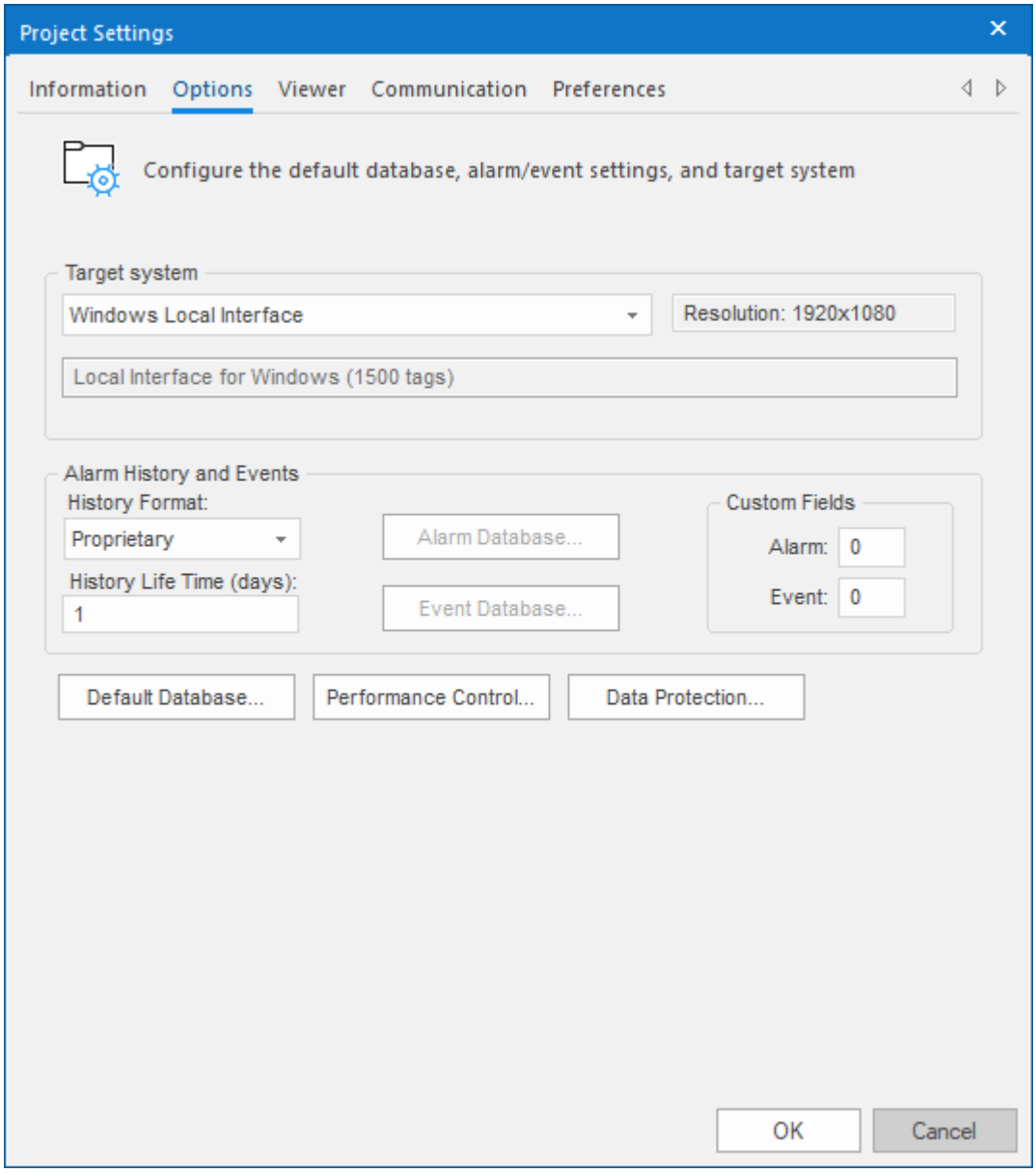
### *Saving your alarm history / event log to an external database*

By default, your project's alarm history and event log are saved to proprietary-format text files in your project's Alarms folder. However, you can change your project settings to save them to an external SQL database instead.

> **Note:** If your project was created with an earlier version of this software and then upgraded to the latest version, you should consider starting over with new database tables.
>
> In the latest version of this software, new database tables are automatically indexed by event time in order to improve run-time performance. Existing database tables cannot be indexed in this way, so if you can afford to discard that data, you should update your database configuration to create new tables.
>
> If you do this, you must also manually edit your project file (*<project name>*.APP) to correct the following setting:
>
> ```
> [Alarm]
> AddEventTimeColumn=1
> ```
>
> This setting exists in order to maintain backward compatibility, and it defaults to 0 for projects that were upgraded.

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**.

The *Project Settings* dialog is displayed.




***Project Settings: Options***

2. In the **Alarm History and Events** area, in the **History Life Time** box, type the number of days of history that you want to save.

   As the history exceeds the specified number of days, it will be automatically deleted in a first-in, first-out manner. If no number is specified — that is, if it is left blank or set to 0 — then history will never be deleted. There is no limit to how much history you can save, but the more you save, the more disk space it will take.

3. From the **History Format** list, select **Database**.

4. To configure a single, default database to be used for both the alarm history and the event log (as well as all other runtime tasks), in the **Default Database** area, click **Configure**.
   The *Default Database Configuration* dialog is displayed. Use the dialog to configure the database connection. For more information, see Configuring a default database for all task history.

5. To configure a separate database for either your event log or your alarm history, click **Event Database** or **Alarm Database**, respectively.
   In either case, a *Database Configuration* dialog is displayed. Use the dialog to configure the database connection. For more information, see Database Configuration.
6. Click **OK**.

### *Format of the alarm history*

This topic describes the location and format of the alarm history that is saved during project run time, if one or more alarm worksheets have been configured.

By default, a project's alarm and event history is saved in Proprietary format. You can configure your project settings to save the alarm and event history in Database format. For more information, see Saving your alarm history / event log to an external database on page 105.

When your project is configured to save the history in Proprietary format (i.e., when **History Format** is **Proprietary**), the alarm history is saved as sequentially numbered .alh files in your project folder. A new .alh file is created for each calendar day, and that file is named according to the following syntax:

```
ALYYMMDD.ALH
```

For example, the alarm history for 17 August 2018 is saved in your project folder at: *<project name>*\Alarm\AL180817.ALH

All of the alarm history for a given day is saved in the same file, regardless of whether the project is stopped and then restarted during that day. (In other words, restarting the project does not create an additional file for that day.) Each change in the alarm state — active, normalized, acknowledged — is appended to the file as a new line, and each line comprises pipe-separated values that describe the alarm. (The pipe character is also known as the vertical bar.) You can open the file with a text editor or spreadsheet program.

> **Tip:** To change the location where your project saves its alarm and event history files, use the SetAppAlarmPath function.

When your project is configured to save the history in Database format (i.e., when **History Format** is **Database**), the alarm history is saved in a table in the external database. All of the alarm history for the entire project is saved in a single table named ALARMHISTORY by default, although you can change that name in your project settings. Each change in the alarm state — active, normalized, acknowledged — is inserted into the table as a new row.

The fields/columns of the alarm history are described below:

| Proprietary | Database | | Description |
|---|---|---|---|
| Field Number | Column Name | Data Type | |
| P1 | — | Integer (int) | Internal version number of the alarm history format.<br>This value is saved only when **History Format** is **Proprietary**. You can use this information to differentiate between .alh files that were created by previous and current versions of this software.<br>**Note:** If the user deletes (hides) the alarm, the internal version number is replaced with **xxx**. For more information, see "Al_Deleted" below. |
| P2 | Al_Start_Time | Timestamp (datetime) | Timestamp that indicates when the alarm started (i.e., when the alarm became active).<br>When **History Format** is **Proprietary**, the start time is saved in the local time zone (i.e., in the time zone that is selected in the computer's settings) and in the following format:<br>**DD/MM/YYYY\|HH:MM:SS.mmm** |
| P3 | | | |

| Proprietary | Database | | Description |
|---|---|---|---|
| Field Number | Column Name | Data Type | |
| | | | The date information is saved in field P2, and the time information is saved in field P3. Please note the pipe character that separates the two fields.<br><br>When **History Format** is **Database**, the start time is saved in Coordinated Universal Time (UTC) by default and in the database's native datetime format. If the database does not support milliseconds in its datetime format, the milliseconds portion is saved in Al_Start_Time_ms. |
| — | Al_Start_Time_ms | Integer (int) | The milliseconds portion of Al_Start_Time.<br><br>This value is saved only when **History Format** is **Database** and the database does not support milliseconds in its datetime format. |
| P4 | Al_Tag | String (varchar) | Name of the project tag on which the alarm occurred. |
| P5 | Al_Message | String (varchar) | Alarm message. |
| P6 | Al_Ack | Boolean (int) | A numeric flag that indicates whether the alarm was acknowledged, where:<br>• `0` = Alarm was acknowledged or does not require acknowledgment.<br>• `1` = Alarm was not acknowledged. |
| P7 | Al_Active | Boolean (int) | A numeric flag that indicates whether the alarm is still active, where:<br>• `0` = Alarm is not active.<br>• `1` = Alarm is active.<br><br>This information is useful for that indicates when an alarm has been acknowledged but not normalized. |
| P8 | Al_Tag_Value | Real (float) | Value of the project tag when the alarm state changed. |
| P9 | Al_Group | Integer (int) | Group or worksheet number of the alarm. |
| P10 | Al_Priority | Integer (int) | Priority number of the alarm. |
| P11 | Al_Selection | String (varchar) | Alias for the alarm, by which the user can select and filter alarms in the Alarm/Event Control object. |
| P12 | Al_Type | Integer (int) | Type of alarm, where:<br>• `1` = HiHi<br>• `2` = Hi<br>• `4` = Lo<br>• `8` = LoLo<br>• `16` = Rate<br>• `32` = DevP (a.k.a. Deviation+)<br>• `64` = DevM (a.k.a. Deviation-) |
| P13 | Al_Ack_Req | Boolean (int) | A numeric flag that indicates whether the alarm requires acknowledgment, where:<br>• `0` = Alarm does not require acknowledgement.<br>• `1` = Alarm requires acknowledgement. |
| P14 | Al_Norm_Time | Timestamp (datetime) | Timestamp that indicates when the alarm was normalized.<br><br>When **History Format** is **Proprietary**, the norm time is saved in the local time zone (i.e., in the time zone that is selected in the computer's settings) and in the following format:<br><br>**`DD/MM/YYYY|HH:MM:SS.mmm`** |
| P15 | | | |

| Proprietary | Database | | Description |
|---|---|---|---|
| Field Number | Column Name | Data Type | |
| | | | The date information is saved in field P14, and the time information is saved in field P15. Please note the pipe character that separates the two fields.<br>When **History Format** is **Database**, the norm time is saved in Coordinated Universal Time (UTC) by default and in the database's native datetime format. If the database does not support milliseconds in its datetime format, the milliseconds portion is saved in Al_Norm_Time_ms. |
| — | Al_Norm_Time_ms | Integer (int) | The milliseconds portion of Al_Norm_Time.<br>This value is saved only when **History Format** is **Database** and the database does not support milliseconds in its datetime format. |
| P16<br>P17 | Al_Ack_Time | Timestamp (datetime) | Timestamp that indicates when the alarm was acknowledged by the user.<br>When **History Format** is **Proprietary**, the ack time is saved in the local time zone (i.e., in the time zone that is selected in the computer's settings) and in the following format:<br>`DD/MM/YYYY|HH:MM:SS.mmm`<br>The date information is saved in field P16, and the time information is saved in field P17. Please note the pipe character that separates the two fields.<br>When **History Format** is **Database**, the ack time is saved in Coordinated Universal Time (UTC) by default and in the database's native datetime format. If the database does not support milliseconds in its datetime format, the milliseconds portion is saved in Al_Ack_Time_ms. |
| — | Al_Ack_Time_ms | Integer (int) | The milliseconds portion of Al_Norm_Time.<br>This value is saved only when **History Format** is **Database** and the database does not support milliseconds in its datetime format. |
| P18 | Al_User | String (varchar) | User who was logged on to the station when and where the alarm occurred. (It is possible for different users to be logged on each time the alarm state changes.) |
| P19 | Al_User_Comment | String (varchar) | Comment submitted by the user when they acknowledged the alarm. |
| P20 | Al_User_Full | String (varchar) | Full name of the user who was logged on to the station when and where the alarm occurred. (It is possible for different users to be logged on each time the alarm state changes.) |
| P21 | Al_Station | String (varchar) | Name of the computer where the alarm occurred. |
| P22 | Al_Prev_Tag_Value | Real (float) | Value of the project tag immediately before the alarm state changed. |
| see description | Al_Deleted | Boolean (int) | A numeric flag that indicates whether the user "deleted" the on-screen alarm message. The alarm is not actually deleted from the alarm history, it is only hidden in the Alarm/Event Control object. The alarm history is updated to reflect this.<br>When **History Format** is **Proprietary**, the line in the history file is updated so that the internal version number in field P1 is replaced with **xxx**.<br>When **History Format** is **Database**, the row in the database table is updated so that Al_Deleted has one of the following values:<br>• 0 = Alarm message was not deleted (hidden) by the user.<br>• 1 = Alarm message was deleted (hidden) by the user. |
| see description | Al_Event_Time | Timestamp (datetime) | Timestamp that indicates when the most recent alarm state change occurred.<br>When **History Format** is **Proprietary**, this information is not actually saved in the alarm history. Instead, it is automatically derived from the other three alarm times (Start, Norm, Ack).<br>When **History Format** is **Database**, the event time is saved in Coordinated Universal Time (UTC) by default and in the database's native datetime format. |

| Proprietary | Database | | Description |
|---|---|---|---|
| Field Number | Column Name | Data Type | |
| | | | If the database does not support milliseconds in its datetime format, the milliseconds portion is saved in Al_Event_Time_ms. |
| — | Al_Event_Time_ms | Integer (int) | The milliseconds portion of Al_Event_Time.<br>This value is saved only when **History Format** is **Database** and the database does not support milliseconds in its datetime format. |
| P23 | Bias | Integer (int) | The difference (in minutes) between the local time zone and UTC, when the alarm is logged and the alarm times (Start, Norm, Ack) are saved.<br>When **History Format** is **Proprietary**, this information is always saved because the alarm times are always saved in the local time zone.<br>When **History Format** is **Database**, this value is saved only if the **Local Time + Time Difference** option (in your project settings) is also selected. |
| P24 | — | Integer (int) | A numeric flag that indicates which project runtime task generated the alarm, where:<br>• `0` = Background Task<br>• `1` = OPC UA Client Runtime<br>• `2` = Driver Runtime<br>This value is saved only when **History Format** is **Proprietary**. |
| P25 | — | String (varchar) | A multi-part code that describes the specific data element on which the alarm occurred. The format of the code varies depending on which project runtime task generated the alarm.<br>When the alarm is generated by the OPC UA Client Runtime task, the value saved in this field contains an additional pipe character that separates parts of the code. The value actually ends with a Start of Text control character (**^B** or **#**). Keep this in mind when you try to read or parse the history file, because the additional pipe character effectively splits one field into two and increases the total number of fields in each line of the file.<br>This value is saved only when **History Format** is **Proprietary**. |
| P26 | — | String (varchar) | The substate of the alarm (e.g., Unacknowledged, Acknowledged, Unconfirmed, Confirmed).<br>This value is saved only when **History Format** is **Proprietary** and the alarm is generated by the OPC UA Client Runtime task. |
| P27 | — | n/a | This field is an artifact created by how the custom fields are appended to the line in the Proprietary format. It contains no value. |
| P28 … P37 | Al_Custom1 … Al_Custom10 | String (varchar) | Up to 10 custom fields. For more information about custom fields, see Options tab on page 104. |
| — | Last_Update | Timestamp (datetime) | Timestamp that indicates when the database entry was created or last updated.<br>This value is saved only when **History Format** is **Database**. The update time is saved in Coordinated Universal Time (UTC) and in the database's native datetime format. If the database does not support milliseconds in its datetime format, the milliseconds portion is saved in Last_Update_ms.<br>This information is used to synchronize databases when the project is configured to use the Secondary Database in addition to the Primary Database. |
| — | Last_Update_ms | Integer (int) | The milliseconds portion of Last_Update.<br>This value is saved only when **History Format** is **Database** and the database does not support milliseconds in its datetime format. |

**Tip:**

To customize the column names in the database table, manually edit your project file (*<project name>*.APP) as follows:

```
[Alarm]
<default column name>=<custom column name>
```

For example:

```
[Alarm]
Al_Message=Alarm_Message
Al_Ack=Acknowledgment
```

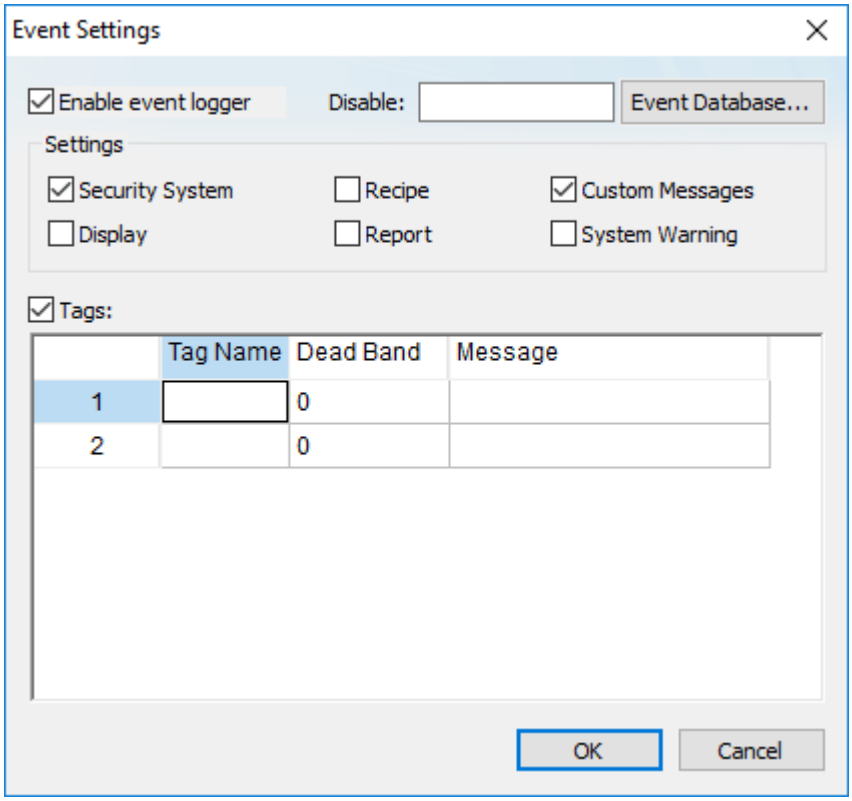
# Events

This section describes EcoStruxure Machine SCADA Expert's logging and event-retrieval features. An event can be any tag change, generating reports or recipes, opening and closing screens, logging onto and logging off the security system, and so forth. EcoStruxure Machine SCADA Expert saves all of these events in a log file, which can then be retrieved by the Alarm/Event Control object.

## *Enable the event logger*

Event logging is disabled by default, to conserve runtime resources. To enable the saving of events to the history file, use the Event Logger in the Project Explorer.

1. In the Project Explorer, on the **Global** tab, double-click **Event Logger**.
   The *Event Settings* dialog is displayed.




***Event Settings dialog***

2. Select **Enable event logger**.
3. In the **Disable** box, type the name of a project tag. Whenever the value of the tag is TRUE (i.e., non-zero) during runtime, event logging will be suspended.
4. In the **Settings** area, select which types of events that you want to log to the history file.

| Option | Description |
|---|---|
| **Security System** | Events generated by your project's security system, including:<br>• Log On / Log Off users<br>• User created/removed by calling the `CreateUser` or `RemoveUser` functions |

| Option | Description |
|---|---|
| | • User blocked/unblocked by calling the `BlockUser` or `UnblockUser` functions<br>• User blocked by the security system after several attempts to enter an invalid password<br>• Password expired<br>• Password modified<br>• Invalid Log On attempt |
| **Display** | Open Screen and Close Screen events. |
| **Recipe** | Recipes loaded, saved, initialized, or deleted. |
| **Report** | Reports saved to disk or sent to printer. |
| **Custom Messages** | Events generated by calling the `SendEvent` function. |
| **System Warning** | Various runtime warnings and errors, including:<br>• Errors that occur when sending alarms by email<br>• Tag was blocked/unblocked<br>• Division by zero<br>• Connection/Disconnection of the remote security system |

5. To log changes in specific project tags, select **Tags**, and then in the table, specify the tags.

| Column | Description |
|---|---|
| **Tag Name** | The name of the project tag that you want to log to the history file. |
| **Dead Band** | A value to filter changes against, so that only changes greater than this value are logged.<br>For example, if you specify a **Dead Band** value of 5 for a tag value of 50 and the tag value changes to 52, then the system will not register this variation in the event log, because the change is less than 5. However, if the tag value change is equal to or greater than 5, then the system will log the new value to the history file. |
| **Message** | A string (message) related to this tag change. You can specify tags in messages using the ***{tagname}*** syntax. |

   The **Tags** option is useful for logging events that are not important enough to be alarm conditions (for example, `Motor On`, `Motor Off`, and so on).

6. Click **OK**.

By default, the event log is saved as a series of text files in your project's Alarms folder. For more information, see Format of the event history on page 398.

Alternatively, you can save the event log to an external SQL database. For more information, see Saving your alarm history / event log to an external database on page 105.

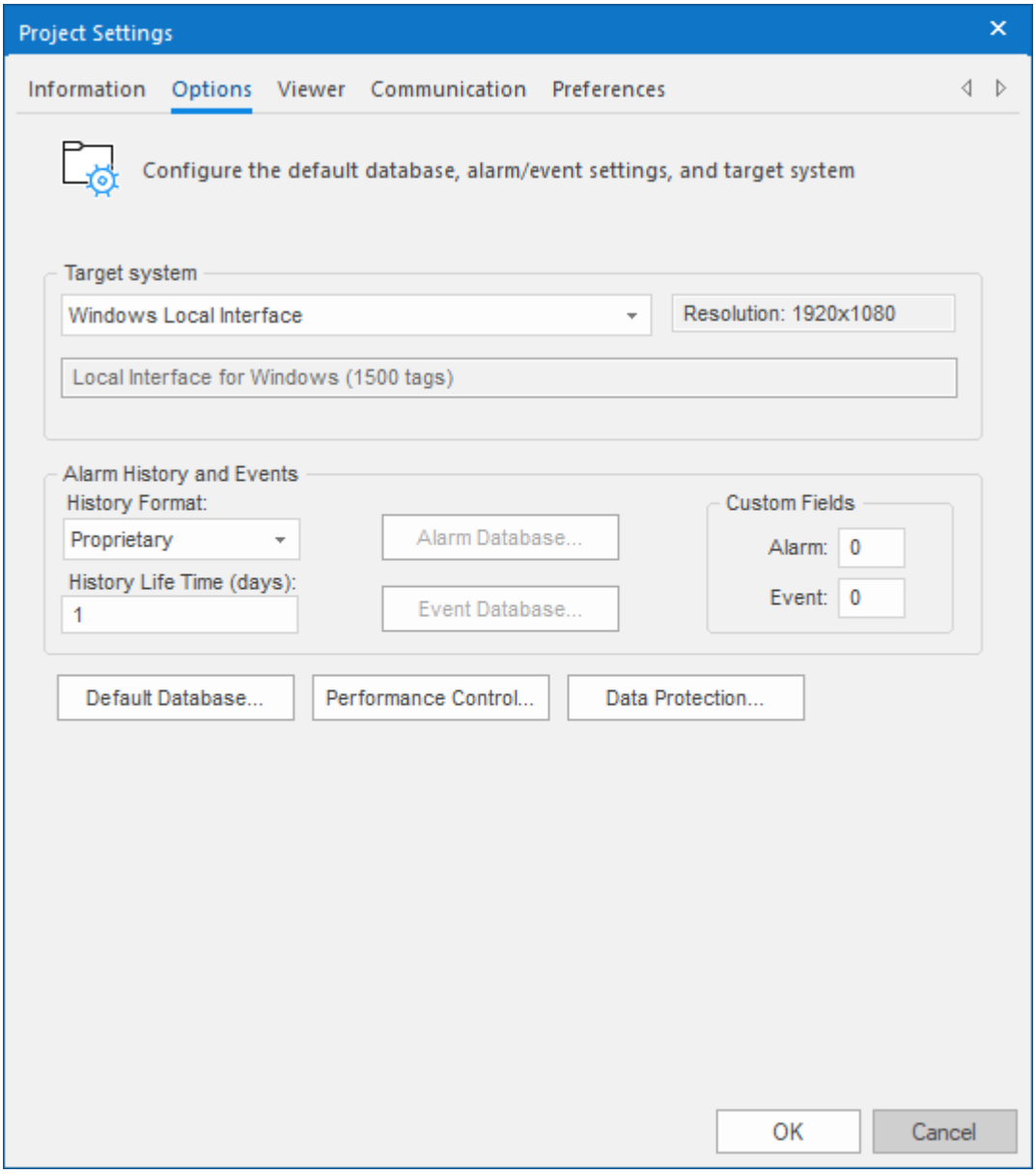
### *Saving your alarm history / event log to an external database*

By default, your project's alarm history and event log are saved to proprietary-format text files in your project's Alarms folder. However, you can change your project settings to save them to an external SQL database instead.

> **Note:** If your project was created with an earlier version of this software and then upgraded to the latest version, you should consider starting over with new database tables.
>
> In the latest version of this software, new database tables are automatically indexed by event time in order to improve run-time performance. Existing database tables cannot be indexed in this way, so if you can afford to discard that data, you should update your database configuration to create new tables.
>
> If you do this, you must also manually edit your project file (*<project name>*.APP) to correct the following setting:
>
> ```
> [Alarm]
> AddEventTimeColumn=1
> ```
>
> This setting exists in order to maintain backward compatibility, and it defaults to 0 for projects that were upgraded.

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**.

The *Project Settings* dialog is displayed.




***Project Settings: Options***

2. In the **Alarm History and Events** area, in the **History Life Time** box, type the number of days of history that you want to save.

   As the history exceeds the specified number of days, it will be automatically deleted in a first-in, first-out manner. If no number is specified — that is, if it is left blank or set to 0 — then history will never be deleted. There is no limit to how much history you can save, but the more you save, the more disk space it will take.

3. From the **History Format** list, select **Database**.

4. To configure a single, default database to be used for both the alarm history and the event log (as well as all other runtime tasks), in the **Default Database** area, click **Configure**.
   The *Default Database Configuration* dialog is displayed. Use the dialog to configure the database connection. For more information, see Configuring a default database for all task history.

5. To configure a separate database for either your event log or your alarm history, click **Event Database** or **Alarm Database**, respectively.
   In either case, a *Database Configuration* dialog is displayed. Use the dialog to configure the database connection. For more information, see Database Configuration.
6. Click **OK**.

## *Format of the event history*

This topic describes the location and format of the event history that is saved during project run time, if the event logger is enabled.

By default, a project's alarm and event history is saved in Proprietary format. You can configure your project settings to save the alarm and event history in Database format. For more information, see Saving your alarm history / event log to an external database on page 105.

When your project is configured to save the history in Proprietary format (i.e., when **History Format** is **Proprietary**), the event history is saved as sequentially numbered .evt files in your project folder. A new .evt file is created for each calendar day, and that file is named according to the following syntax:

```
EVYYMMDD.EVT
```

For example, the event history for 17 August 2018 is saved in your project folder at: `<project name>\Alarm\EV180817.EVT`

All of the event history for a given day is saved in the same file, regardless of whether the project is stopped and then restarted during that day. (In other words, restarting the project does not create an additional file for that day.) Each event is appended to the file as a new line, and each line comprises tab-separated values that describe the event. You can open the file with a text editor or spreadsheet program.

> **Tip:** To change the location where your project saves its alarm and event history files, use the `SetAppAlarmPath` function.

When your project is configured to save the history in Database format (i.e., when **History Format** is **Database**), the event history is saved in a table in the external database. All of the event history for the entire project is saved in a single table named EVENTHISTORY by default, although you can change that name in your project settings. Each event is inserted into the table as a new row.

The fields/columns of the event history are described below:

| Proprietary | Database | | Description |
|---|---|---|---|
| Field Number | Column Name | Data Type | |
| P1 | — | Integer (int) | Internal version number of the event history format.<br>This information is saved only when **History Format** is **Proprietary**. You can use this information to differentiate between .evt files that were created by previous and current versions of this software. |
| P2 | Ev_Type | Integer (int) | Type of event:<br>• 1 = Security System<br>• 2 = Display<br>• 3 = Recipe<br>• 4 = Report<br>• 5 = Custom Message<br>• 6 = System Warning<br>• 7 = Tag |
| P3 | Ev_Time | Timestamp (datetime) | Timestamp that indicates when the event occurred. |

| Proprietary | Database | | Description |
|---|---|---|---|
| Field Number | Column Name | Data Type | |
| | | | When **History Format** is **Proprietary**, the event time is saved in the local time zone (i.e., in the time zone that is selected in the computer's settings) and in the following format:<br>**`MM/DD/YYYY HH:MM:SS.mmm`**<br>When **History Format** is **Database**, the event time is saved in Coordinated Universal Time (UTC) by default and in the database's native datetime format. If the database does not support milliseconds in its datetime format, the milliseconds portion is saved in Ev_Time_ms. |
| — | Ev_Time_ms | Integer (int) | The milliseconds portion of Ev_Time.<br>This information is saved only when **History Format** is **Database** and the database does not support milliseconds in its datetime format. |
| P4 | Ev_Info | String (varchar) | Name of the project tag that is associated with the event. |
| P5 | Ev_Value | Real (varchar) | Value of the project tag when the event occurred. |
| P6 | Ev_Source | String (varchar) | Reserved. |
| P7 | Ev_User | String (varchar) | User who was logged on to the station when and where the event occurred. |
| P8 | Ev_User_Full | String (varchar) | Full name of the user who was logged on to the station when and where the event occurred. |
| P9 | Ev_Message | String (varchar) | Event message. |
| P10 | Ev_Station | String (varchar) | Name of the station where the event occurred. |
| P11 | Ev_Comment | String (varchar) | Comment entered by the user when the event occurred. User comments are optional. |
| P12 | Ev_Prev_Value | Real (varchar) | Value of the project tag before the event occurred. |
| — | Ev_Deleted | Boolean (int) | A numeric flag that indicates whether the user deleted the event message:<br>• `0` = FALSE; event message was not deleted.<br>• `1` = TRUE; event message was deleted.<br>This information is saved only when **History Format** is **Database**, and it causes a new update time to be saved in Last_Update. |
| P13 | Bias | Integer (int) | Difference (in minutes) between the local time zone and UTC, when the event is logged and the event time is saved.<br>When **History Format** is **Proprietary**, this information is always saved because the event time is always saved in the local time zone.<br>When **History Format** is **Database**, this information is saved only if the **Local Time + Time Difference** option (in your project settings) is also selected. |
| P14 … P23 | Ev_Custom1 … Ev_Custom10 | String (varchar) | Up to 10 user-defined custom fields. |
| — | Last_Update | Timestamp (datetime) | Timestamp that indicates when the database entry was created or last updated.<br>This information is saved only when **History Format** is **Database**. The update time is saved in Coordinated Universal Time (UTC) and in the database's native datetime format. If the database does not support milliseconds in its datetime format, the milliseconds portion is saved in Last_Update_ms.<br>This information is used to synchronize databases when the project is configured to use the Secondary Database in addition to the Primary Database. |
| — | Last_Update_ms | Integer (int) | The milliseconds portion of Last_Update. |

| Proprietary | Database | | Description |
|---|---|---|---|
| Field Number | Column Name | Data Type | |
| | | | This information is saved only when **History Format** is **Database** and the database does not support milliseconds in its datetime format. |

**Tip:**

To customize the column names in the database table, manually edit your project file (*<project name>*.APP) as follows:

```
[EventLogger]
<default column name>=<custom column name>
```

For example:

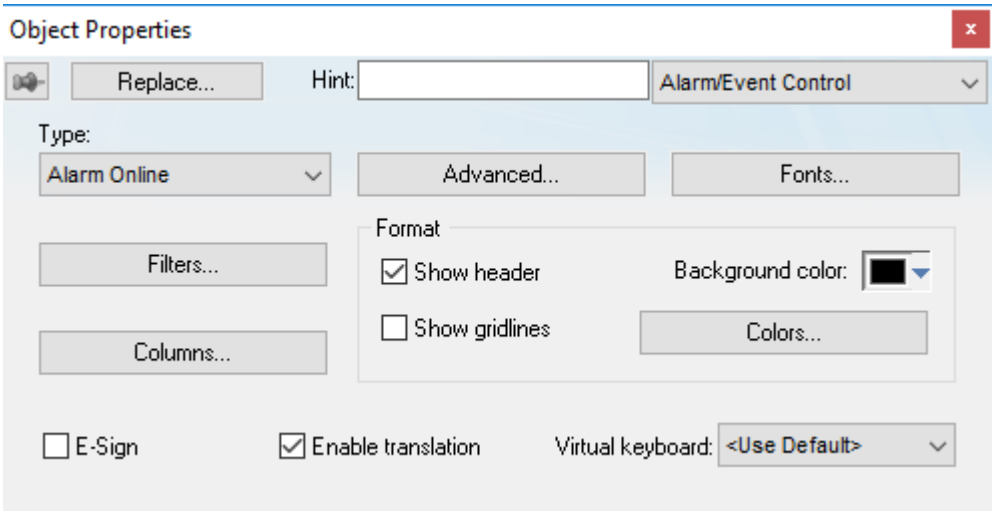
```
[EventLogger]
Ev_Info=TagName
Ev_Message=Message
```

## Alarm/Event Control object

Use the **Alarm/Event Control** tool to add an Alarm or Event Control object to a project screen.

To create and configure an Alarm/Event Control object:

1. On the **Draw** tab of the ribbon, in the **Data Objects** group, click **Alarm/Event Control**.
2. Click in the display, and drag the mouse to create and adjust the object's shape.
3. Double-click on the object to open the following *Object Properties* dialog box.




***Object Properties: Alarm/Event Control***

You can use this dialog box to specify the following parameters:

- Select an alarm object mode in the *Type* group:
  - **Alarm Online**: Display only current alarm messages, as configured in your Alarm worksheets.
  - **Alarm History**: Display only alarm messages from the Alarm History database.
  - **Alarm History + Event**: Display both alarm messages from the Alarm History database and logged events from the Event History database.
  - **Event**: Display only logged events from the Event History database.
- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the animation.
- Click (enable) the **Enable translation** checkbox to enable the external translation of messages using the Translation Table. (For more information, see Project Localization on page 681.)
- **VK**: Virtual Keyboard type used for this object. You need to select the Virtual Keyboard option in the *Viewer* settings (**Viewer** on the Project tab of the ribbon) before configuring the Virtual Keyboard for this interface.

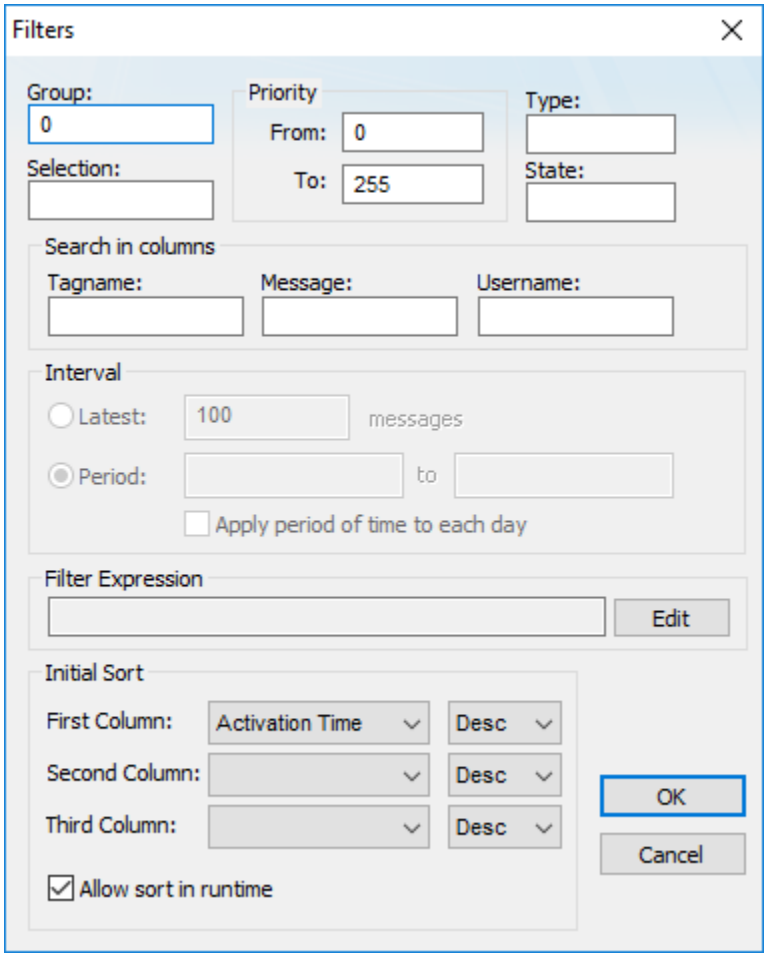
> **Tip:** By default, an Alarm/Event Control object can display up to 16,000 messages at the same time in projects running on the Windows target platform, and up to 1,024 messages at the same time in projects running on the Windows Embedded target platform. If necessary, however, you can increase the maximum number of messages in a project running on Windows. To do this, manually edit your project file (*<project name>*.app) to add the following setting:
>
> ```
> [Objects]
> MaxMessagesAlarmControl=<number of messages>
> ```

We have successfully tested projects with Alarm/Event Control objects that display up to 30,000 messages at the same time, but you can specify any number as long as the target device has sufficient system resources to run your project.

Please note this setting applies to Windows only; the maximum number of messages cannot be changed for Windows Embedded.

### Filters

To filter alarm messages during runtime, click the **Filters** button. The *Filters* dialog box displays so you can specify filtering parameters for the Alarm Control object.




***Filters dialog box***

- Use the **Group** field to filter messages by the Alarm group/worksheet number. The worksheets are organized in the Alarms folder, in the Tasks tab of the *Project Explorer*, starting with 1. If you specify a Group of 0, then all of the worksheets will be displayed. You can use commas and/or dashes to specify a range of groups (e.g.,`1,3,5-6`).
- Use the **Selection** field to filter messages by the Selection text configured on the Alarm worksheet.
- In the *Priority* group, use the **From** and **To** fields to filter messages by the Priority configured on the Alarm worksheet. Type numerical values into the these fields to delimit the priority range.
- Use the **Type** field to filter messages by the alarm type (e.g., HiHi, Hi, Lo, LoLo, Rate, Dev+, Dev-). You can use commas to specify more than one type; for example,`HiHi,LoLo`.
- Use the **State** field to filter messages by the alarm status:

| Value | Description |
|---|---|
| 0 | All alarms (default) |
| 1 | All active and unacknowledged alarms |
| 2 | All active and acknowledged alarms |
| 3 | All inactive and acknowledged alarms |
| 4 | All inactive and unacknowledged alarms |

Leaving this field blank is effectively the same as entering a value of 0.

- In the *Search in columns* group, use the **Tagname**, **Message**, and/or **Username** text fields to specify criteria for filtering messages. Type a tagname, message, and/or user name into the text field for which you want EcoStruxure Machine SCADA Expert to search.
- Use the settings in the *Interval* group to filter messages by the last ***x*** messages (**Latest**) or based on a period of time (**Period**). If you do not specify any interval at all, then only the alarms for the current day will be displayed.

> **Note:**
>
> - You can specify String tags in curly brackets (e.g. {*tagname*}) in the **Group**, **Selection**, **Tagname**, **Message**, and **Username** fields, to change these values during runtime.
> - You must specify String tags *without* curly brackets (e.g. *tagname*) in the **Type** field and the **Period** fields of the *Interval* group. These fields cannot take values directly.
> - You can specify Integer tags in the **From** and **To** fields *Priority* group, the **State** field, and the **Latest** field from the *Interval* group.
> - You can use wildcards ( ***** and **?** ) when specifying values for the **Selection**, **Tagname**, **Message**, and **Username** fields.

- Use the **Filter Expression** to configure an expression that will filter unwanted messages out of the display. Only messages that satisfy the expression will be shown.

  To enter a filter expression, click **Edit**; the *Alarm Filter Expression* dialog box is displayed. The filter expression must follow the basic syntax of...

  **[*columnname*]*operator*'*value*'**

  ...where the *columnname* is the name of a column in the Alarm/Event Control object and *operator* is any of the standard relational operators (e.g., **=**, **<**, **>**, **<=**, **>=**, **<>**). For example:

  **[Activation Time]>'08/17/2007 15:00'**

  This filter will only show alarm messages with activation times greater (later) than 15:00 on 08/17/2007.

> **Note:** The **Display Value** and **State** columns are not supported in the filter expression.

> **Tip:** It is not necessary to use square brackets when *columnname* is one word (e.g., **Value**), but doing so can make the filter expression easier to read.

You can combine several conditions simultaneously by using the logical operators AND, OR, and NOT. You can also use parentheses to establish the order of operations. For example:

```
[Type]='HiHi' OR [Type]='LoLo' AND [Activation Time]>'08/17/2007 15:00'
```

```
([Tag Name] Like 'Tag1') AND ([Tag Name] NOT Like 'Tag2') AND
 [Custom1]='MyCustomArea'
```

You can use wildcards (***** and **?**) in the filter expression. You can also change the filter expression during run time by specifying project tags and/or built-in functions in curly brackets. For example:

```
[Value]='{AlarmFilterValue}'
```

```
[ActivationTime]<='{DateTime2UTC(Date + " " + Time)}'
```

Please note that filtering by time works only if the Alarm/Event Control object is configured to show alarm history, and all times should be specified in Coordinated Universal Time (UTC), because that is how the alarm history is saved. If you need to convert between local time and UTC, you can use the Date & Time functions.

The *Alarm Filter Expression* dialog box has a limit of 1024 characters. You can configure a filter expression of up to 2048 characters, however, by using the curly brackets syntax described above. The contents of the brackets will be evaluated during run time. So, for example, you can specify two or more project tags (String type) that contain parts of the overall expression and then use AND operators to combine the parts:

```
{MyFilterExpression1} AND {MyFilterExpression2}
```

> **Note:** If your filter expression includes any dates, they must be in the current date format. If they are not, you might see unexpected behavior during run time. For more information, see About the date format and how to change it on page 694.

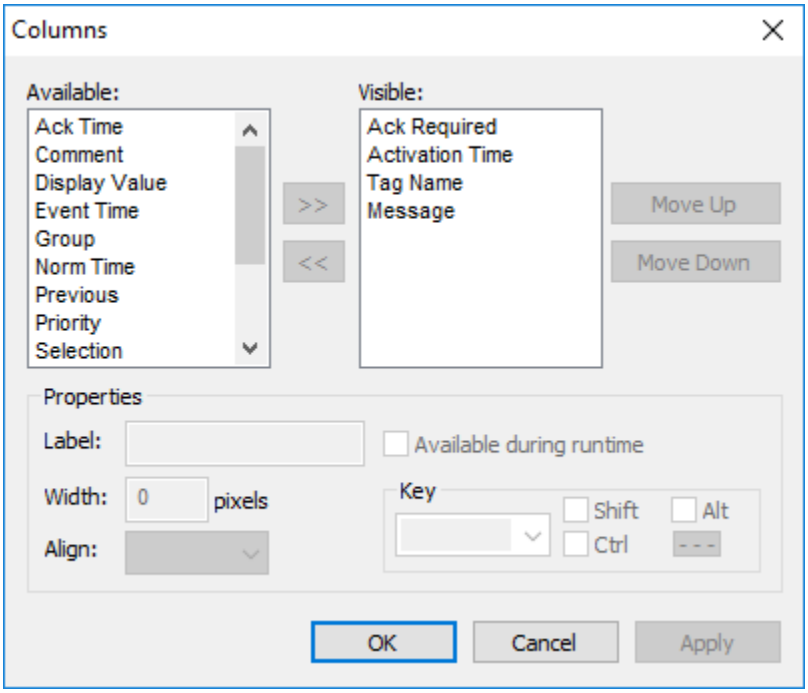
- Use the settings in the *Initial Sort* group to set the default sorting order. Select a sort type from the **Column** combo-box, and then select **Asc** or **Desc** to sort in ascending or descending order. You can configure up to three levels of sorting.

  > **Note:** If you configure all three levels with sort types other than Activation Time, then the project will automatically sort on a fourth level according to Activation Time, in descending order.
  >
  > You cannot change the type of this fourth-level sort, but you can toggle its default order — from descending to ascending — by manually editing your project file (*<project name>*.app) to change the following setting:
  >
  > ```
  > [Objects]
  > DescendingAlarmListTime=TRUE or FALSE
  > ```
  >
  > **TRUE** sorts in descending order, **FALSE** sorts in ascending order.

  Click the **Allow sort in runtime** checkbox if you want to allow the user to change the sort order during runtime.

### Columns

Click the **Columns** button to open the *Columns* dialog box where you can specify display properties for columns in the object.




***Columns dialog box***

- The *Available* list contains all of the column types available for this object. The *Visible* list contains all of the column types currently in use for the object.

  Click the **»** and **«** buttons to move selections between the two lists.

  > **Tip:** You can configure an Alarm Control object to display recently replaced values together with their new values. To do so, move both **Value** and **Previous** to the *Visible* list.

  Click the **Move Up** or **Move Down** buttons to rearrange the order of columns in the *Visible* list.
- Use the **Label** and **Width** fields in the *Properties* group to change the default column labels and widths at runtime.
- Use the **Align** combo box to specify alignment (**Left**, **Center**, or **Right**) for the alarm message text within a specified column.
- Click (enable) the **Available during runtime** checkbox to allow the user to add selected columns to the visible list during runtime.
- Use the **Key** box to assign a shortcut to each column. This allows you to sort the information on the Alarm Control object by any column, using keyboard keys instead of the mouse cursor.

When you are finished, click **OK** to close the *Column* dialog box.

> **Tip:** You can associate text labels with priority values, so that more meaningful information is displayed in the Priority column of the Alarm Control. To do this, manually edit your project file (*<project name>*.APP) to add the following entries:
>
> ```
> [Alarm]
> PriorityLabelCount=N (total number of labels)
> PriorityValue1=value
> PriorityLabel1=label
> ```

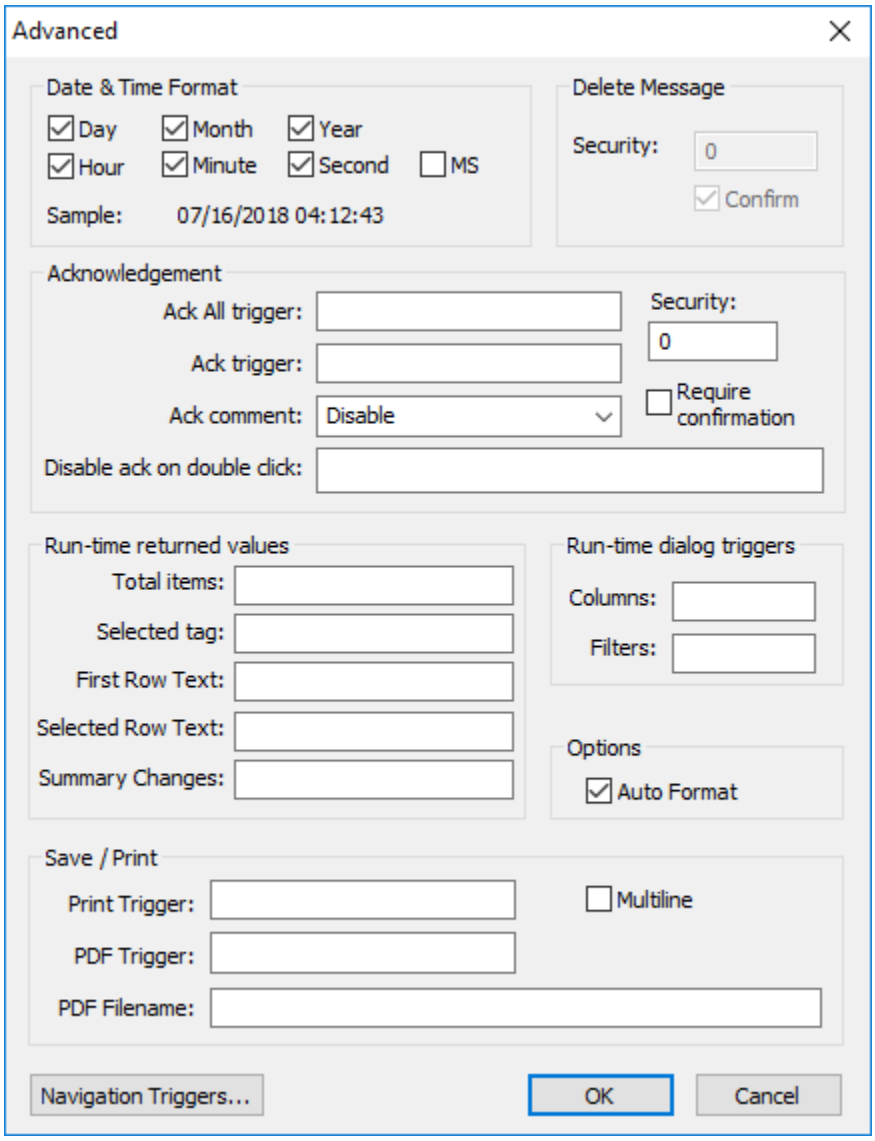
```
...
PriorityValueN=value
PriorityLabelN=label
```

Here is an example:

```
[Alarm]
PriorityLabelCount=3
PriorityValue1=490
PriorityLabel1=ALMTest
PriorityValue2=480
PriorityLabel2=Test2
PriorityValue3=470
PriorityLabel3=Test100
```

### Advanced

Click the **Advanced** button to open the *Advanced* dialog box where you can specify advanced properties for the Alarm Control object.




***Advanced dialog box***

- Use the settings in the **Date & Time Format** group to control which date and time information displays in the alarm message. Click (enable) a checkbox to include that element in the display. Note: **MS** stands for milliseconds.

> **Tip:** Watch the **Sample** text to preview how the information will look in the alarm message.

- Use the settings in the **Delete Message** group to control who can delete alarm messages from the Alarm History:
  - **Security**: Use this field to specify which security level can delete alarm messages. Only those users with the specified security level will be allowed to delete an alarm message.

- **Confirm**: Select this option to require the user to confirm a message deletion before EcoStruxure Machine SCADA Expert actually deletes the selected alarm message.

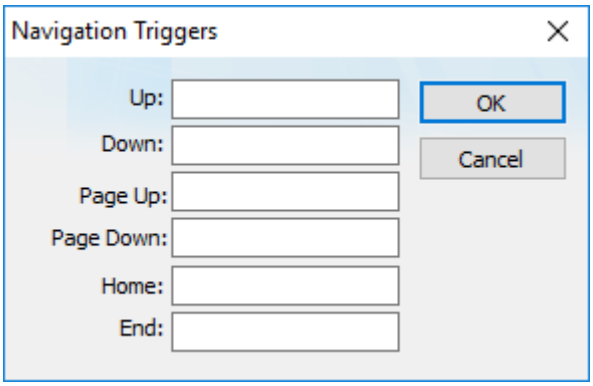
- Use the settings in the **Acknowledgement** group to control how alarms are acknowledged:
  - **Ack All trigger** box: Type a tag to receive a value. When the tag value changes, it indicates that all messages in the alarm object have been acknowleged.
  - **Ack trigger** box: Type a tag to receive a value. When the tag value changes, it indicates that the message at the top of the alarm object has been acknowleged.
  - **Ack comment** list: Select **Disabled**, **Optional**, or **Mandatory** to determine whether the user can or must enter comments after acknowledging alarms.
  - **Disable ack on double click** box: Type either a numeric value or a tag/expression. When it evaluates as TRUE (non-zero), the user cannot acknowledge alarms by double-clicking (or double-tapping) them. This option may be used, for example, either to force the operator to click another button to acknowledge the alarm or to prevent alarms from being acknowledged on thin clients.
  - **Security** box: Type a numeric value to specify which security level can acknowlege an alarm message. Only those users with the specified level can respond.
  - **Require confirmation** checkbox: Select this option to display a confirmation dialog when the user tries to acknowledge a single alarm.
- Use the settings in the **Run-time returned values** group to get information about the alarms during run time:
  - **Total items** box: Type an integer tag to see how many alarms remain after EcoStruxure Machine SCADA Expert filters the alarm object using parameters specified on the *Filters* dialog box.
  - **Selected tag** box: Type a string tag to enable the end user to click on an alarm message to see the name of the tag associated with that message.
  - **First Row Text** box: Type the name of a project tag or array (String type). That tag or array will receive the contents of the columns of the first row of the Alarm/Event Control. If you specify a tag, the columns will be separated by tabs. If you specify an array, the array elements will each receive one column. If the array is not large enough to receive all of the columns, the remaining columns will be discarded. Whenever the first row changes — either due to a new Alarm/Event, or simply because the rows are reordered — the specified array is updated.
  - **Selected Row Text** box: Type the name of a project tag or array (String type). That tag or array will receive the contents of the columns of the selected row (i.e., the row that the user has clicked/tapped) of the Alarm/Event Control. If you specify a tag, the columns will be separated by tabs. If you specify an array, the array elements will each receive one column. If the array is not large enough to receive all of the columns, the remaining columns will be discarded. Whenever the selected row changes — that is, whenever the user clicks/taps another row — the specified array is updated.
  - **Summary Changes** box: Type the name of a project tag (Integer type). That tag will receive a running count of the number of changes in the Alarm/Event Control. For example, when a new Alarm occurs or when an Alarm is acknowledged, the value of the configured tag will be incremented. Reordering the rows is not counted as a change.
- Use the settings in the **Run-time dialog triggers** group to allow the user to customize the object during run time:
  - **Columns** box: Type a tag name. When the tag value changes, it opens a dialog box allowing the user to customize the columns visible in the object.
  - **Filters** box: Type a tag name. When the tag value changes, it opens a dialog box allowing the user to filter the columns visible in the object.
- **Auto Format** checkbox: When checked, decimal values in the **Display Value**, **Previous** and **Value** columns of the object will be formatted according to the virtual table created by the function `SetDecimalPoints`.
- Use the **Save / Print** group to control the printing of alarms during run time:
  - **Print Trigger**: When the tag configured in this field is toggled, the current state of the Alarm/Event Control object is sent to the default printer.
  - **PDF Trigger** box: When the tag configured in this field is toggled, the current state of the Alarm/Event Control object is saved as a PDF file at the location specified by **PDF Filename**.

- **PDF Filename** box: Enter a complete file path and name where the PDF file is to be saved. You can also enter a tag name using the **{tag}** syntax.

  > **Note:** **PDF Trigger** and **PDF Filename** are not supported in projects running on Windows Embedded or Thin Client.

- **Multiline** checkbox: When this option is checked, the print output or PDF will be formatted according to the available column space, and the text within each cell will be wrapped so that all of it is shown.

- Click the **Navigation Triggers** button to open the following dialog box:




***Navigation Triggers dialog box***

  You can make the on-screen Alarm Control object scroll up, scroll down, page up, page down, go to home (beginning) of page, or go to end of page by configuring tags in the corresponding fields. Whenever the values of the configred tags change, the Alarm Control object will navigate that way. This is useful for adding navigation controls to the screen; for example, if you configure the same tag to the **Up** field in this dialog box and a Pushbutton object, then the Alarm Control object will scroll up whenever the Pushbutton object is pressed.

When you are finished, click **OK** to close the *Advanced* dialog box.

### Fonts

Click the **Fonts** button to open a standard Fonts interface where you can specify display properties for the message text.

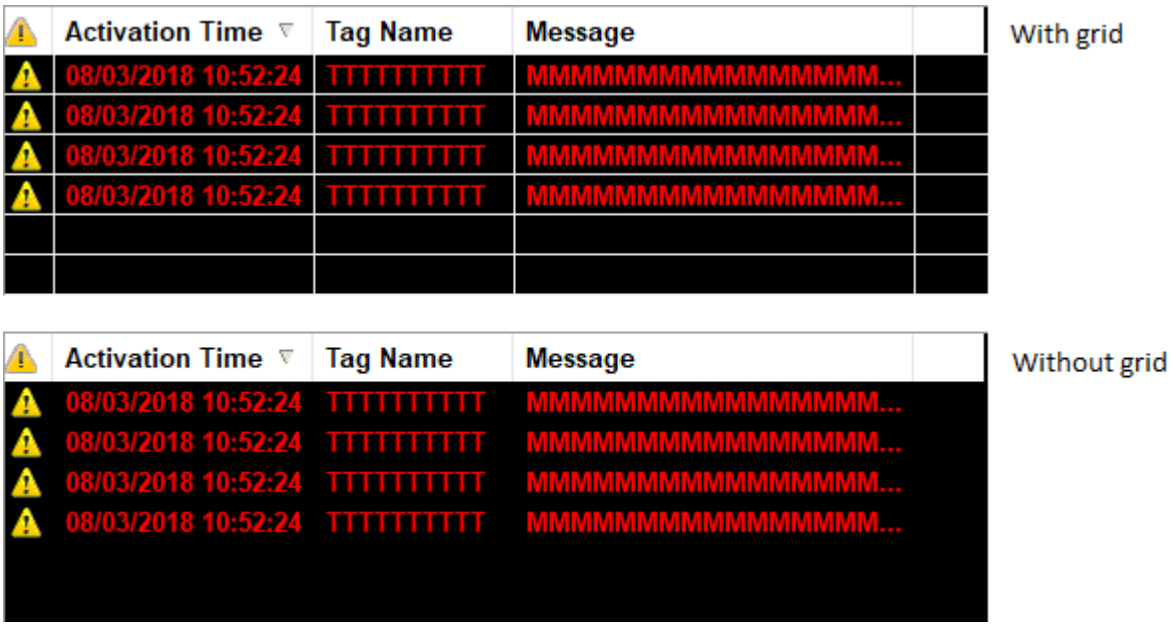
### Format

Use the *Format* area of the object properties to configure the appearance of the Alarm/Event Control object during run time:

- Select **Show header** to show a header on the object. The header displays the column labels.

***Showing the header***

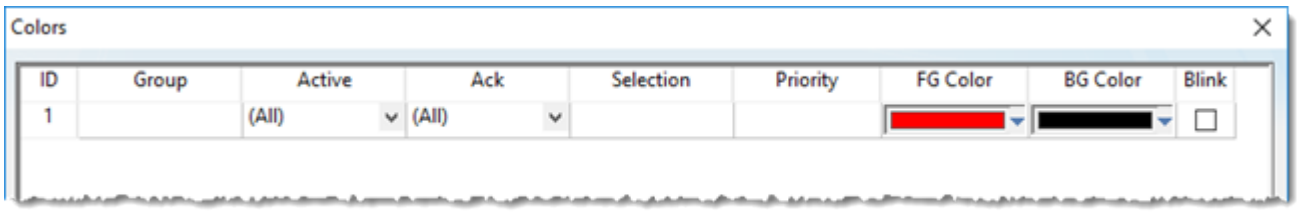
- Select **Show gridlines** to show gridlines in the object. Gridlines can make it easier to distinguish individual rows and columns in the object.




***Showing the gridlines***

- Use the **Background color** box to select a background color for the object. Click the box to open the color palette pop-up, then click a color to select it.

- Click **Colors** to open the *Colors* dialog box, which you can use to select background and foreground colors for specific alarms. These colors will override the default colors that you selected in your Alarm worksheet(s). This is useful for highlighting special alarms.




***Colors dialog box***

> **Note:** This feature is not supported in projects running on Windows Embedded target systems.

In each row of this table, you can configure a subset of alarms using similar criteria as in the *Filters* dialog box, and then for that subset you can select custom background and foreground colors:

1. In the **Group** box, type the number of the Alarm group/worksheet. You may use commas and/or dashes to specify a range of groups.
2. In the **Active** box, select **All** (both active and normalized alarms), **Active** (active alarms only), or **Norm** (normalized alarms only).
3. In the **Ack** box, select **All** (both acknowledged and unacknowledged alarms), **Acked** (acknowledged alarms only), or **Unacked** (unacknowledged alarms only).
4. In the **Selection** box, type the selection text that you configured in the Alarm worksheet. You may leave this box empty.
5. In the **Priority** box, type the priority number that you configured in the Alarm worksheet. You may leave this box empty.
6. Click the **BG Color** box to open a color picker, and then select the color that you want to be the background color of the alarm message.
7. Click the **FG Color** box to open a color picker, and then select the color that you want to be the foreground color of the alarm message.
8. Select **Blink** if you want the alarm message to blink. This will make it more noticeable.

Please note that the subsets you configure here must pass any filters that you previously configured in the *Filters* dialog box. For example, if you configured the filters to show groups `1-6` in the Alarm/Event Control object, then configuring a subset of group `7` here will have no effect.

Also, the colors that you select will be used to indicate all possible alarm states (i.e., activation, acknowledgement, and normalization), if you configure the subset to include all of those states. Therefore, if you want different colors for different alarm states, then you must configure additional subsets.

## *Customize the audible alarm*

You can customize the sound and frequency of the audible alarm that is played on thin clients when an alarm is active.

Before you begin this task, your custom alarm sound should be saved as a .wav file in your project's Web folder. For example: `<project folder>\Web\CustomAlarm.wav`

By default, the audible alarm is a standard system beep that is played once per second (1000 milliseconds). This is determined by settings in your project file, which you can manually edit. You can also completely disable the audible alarm.

> **Note:** This feature is currently supported only on Thin Clients (including the project runtime's local Viewer module). It is not supported on Mobile Access.

To customize the audible alarm:

1. Save and close your project, and then exit the project development software.
   You should not manually edit your project file while it is open in the software.
2. Use a text editor to open your project file, which should be located at: *<project folder>**<project name>*.app
3. In your project file, locate the following settings:

```
[Alarm]
AlarmBeep=1
AlarmBeepTime=1000
AlarmSound=
```

4. Edit the settings as needed:

   **AlarmBeep**

   A boolean flag that enables the audible alarm. The default value is 1 for new projects. If you change this setting to 0, the audible alarm will be disabled on all thin clients.

   **AlarmBeepTime**

   The frequency of the audible alarm, in milliseconds. The default value is 1000 for new projects. If you change this setting, make sure the time is long enough to allow your custom alarm sound to play properly.

   **AlarmSound**

   The name of the .wav file that contains your custom alarm sound (e.g., `CustomAlarm.wav`). The default value is empty for new projects, and if it is empty, the standard system beep will be played instead. The specified file must be located in your project's Web folder.
5. Save and close your project file.

## Trend worksheet

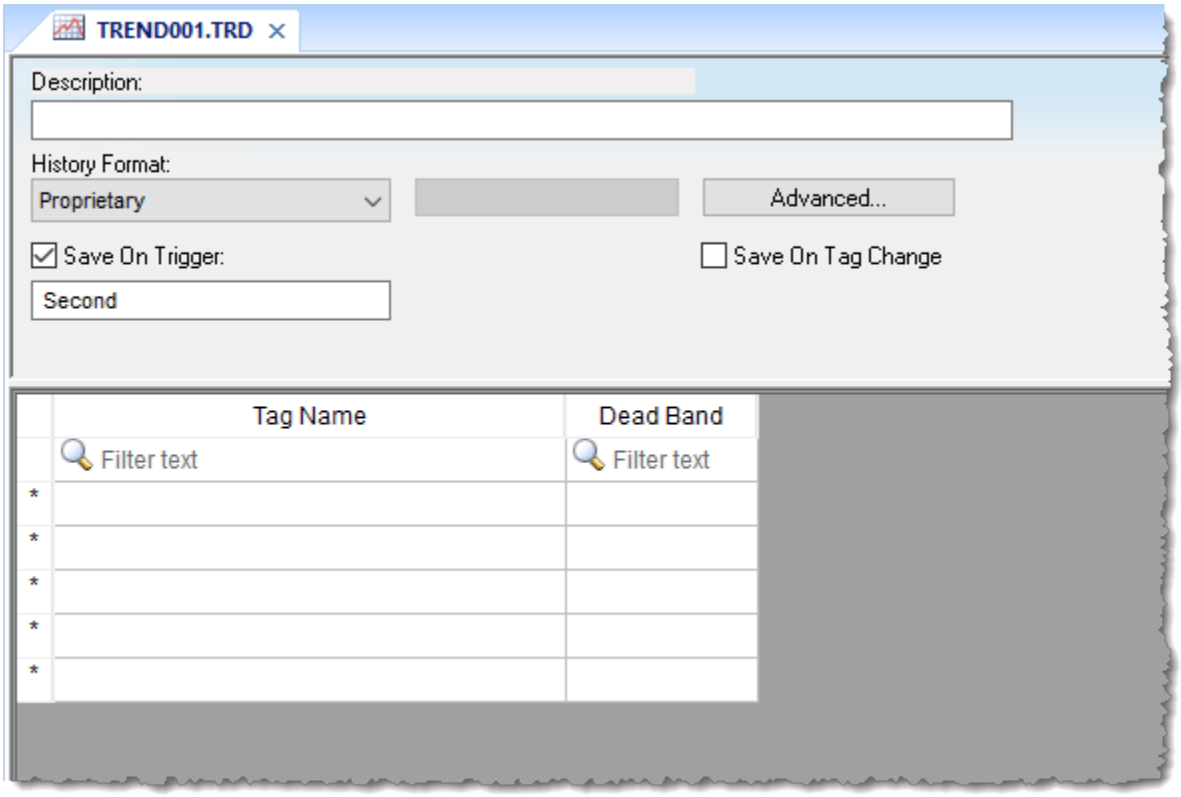
The Trend folder enables you to configure history groups that store trend curves. You can use the Trend worksheet to declare which tags must have their values stored on disk, and to create history files for trend graphs. The project stores the samples in a binary history file (*.hst), and shows both history and on-line samples in a screen trend graph.

The Trend worksheet is executed by the Background Task module (see Runtime Tasks on page 136). It handles the saving of trend data to the history, but it does not display that data to the operator; the Trend Control screen object, available on the Graphics tab of the ribbon, must be created and configured in a screen in order to display trend data.

To create a new Trend worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Trend**;
- Right-click the **Trends** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the File menu, click the **File** tab, and then select **Trend Worksheet**.

To edit an existing Trend worksheet, double-click it in the Project Explorer.




***Trend worksheet***

The Trend worksheet is divided into two areas:

- Header area (top section), which contains information for the entire group
- Body area (bottom section), where you define each project tag in the group. This section contains several columns (only two are shown in the preceding figure).

### Header

Configure the following settings in the worksheet header:

#### Description

Type a description of the worksheet for documentation purposes.

#### History Format

Click the arrow button to select a trend history format from the list. The available options are:

**Proprietary**

Save trend history in a proprietary, binary file. The file is saved in your project folder (on the project runtime server) at: `[…]\<project name>\Hst\GGYYMMDD.hst`

- **GG** = Trend worksheet number
- **YY** = Last two digits of the year
- **MM** = Month
- **DD** = Day

A new history file is created for each calendar day that the project runs.

The utility programs `HST2TXT.EXE` and `TXT2HST.EXE` are provided in order to convert history files from binary ( `*.hst` ) to plain text ( `*.txt` ) and vice versa. For more information, see Converting Trend History Files from Binary to Text on page 421 and Converting Trend History Files from Text to Binary on page 422.

**Database**

Save trend history in an external SQL database of your choice. After you select this format, click **Database Configuration** to open the *Database Configuration* dialog box, where you can configure the connection to the database. For more information, see Database Configuration on page 108 and Database Interface on page 815.

By default, the history is saved in the table **TRENDGGG** ( **GGG** = Trend Worksheet Number; e.g., **TREND001** for the Trend Worksheet 001).

**Historian**

Save trend history to a Historian database or AVEVA Insight. After you select this format, click **Historian Configuration** to open the *Historian* dialog box, where you can configure the connection to the database. The trend history for each project tag is saved separately in the Historian database, but you can use **Prefix** in the database connection settings in order to keep the tags grouped together. For more information, see Support for AVEVA Insight and Historian on page 837.

> **Note:**
>
> You can specify String tags in many fields of the Trend worksheet, to change those values during run time, but doing so may affect how those values are saved in the trend history:
>
> - When the history format is **Proprietary**, the value of the String tag is converted to a numeric value (if possible) and then saved in the history file. If numeric conversion is not possible, then a value of 0 is saved.
> - When the history format is **Database** or **Historian**, the actual value of the String tag is saved in the database.

#### Save On Trigger

Click (enable) and type a tag name to save trend samples when someone changes the specified tag. (Tag change can be an event from the Scheduler.)

#### Save On Tag Change

Click (enable) to always save the trend sample when a value change occurs in any of the tags from that group.

When the history format is **Proprietary** or **Database**, all of the tags in the group are saved after each change. When the history format is **Historian**, only the tag that changed is saved.

#### Advanced

Click to display the Trend Advanced Settings dialog. For information about completing the fields in this window, see Batch History Configuration.

### Body

For each project tag, configure the following settings in the worksheet body:

#### Tag Name

The name of the project tag for which trend history will be saved.

#### Dead Band

Type a value to filter acceptable changes when **Save on Tag Change** is used. For example, Dead Band has value = 5. If the tag value is 50 and changes to 52, the system will not register this variation in the database, because it is less than 5. If the change is equal to or greater than 5, the new value will be saved to the history file.

#### Field

When **History Format** is **Database**, this is the name of the field (in the SQL database table) where the trend history will be saved. If this field is left blank, the project tag name will be used.

For array tags and classes, special characters (`[].`) will be replaced by underscores (_), as shown in the examples below:

| Tag Name | Field Name |
|---|---|
| `MyArray[1]` | `MyArray_1` |
| `MyClass.Member1` | `MyClass_Member1` |
| `MyClass[3].Member2` | `MyClass_3_Member2` |

#### Historian Tag

When **History Format** is **Historian**, this is the name of the tag (in the Historian database) where the trend history will be saved. If this field is left blank, the project tag name will be used.

When you save a Trend worksheet, only the header settings are saved as part of the worksheet file. All of the trend configurations that make up the body of the worksheet are actually saved as tag properties. The next time you open that worksheet, the tags database is scanned for all trend configurations that belong to the worksheet (i.e., the trend group), and then that information is used to recreate the body of the worksheet. This happens quickly and automatically every time you open the worksheet, so it might seem like you are opening a static file but that is not the case.

You may think of the Trend worksheet as an editor for those tag properties that are related to history. If you use either the *Tag Properties* dialog box or the `TagsDBSetTrend` function to edit the same properties, the updated trend configurations will be included in the body of the worksheet the next time you open it. In fact, you can set trends on tags before you create any Trend worksheets at all; when you do create the worksheets, they will be automatically populated with trend configurations according to their group numbers.

You cannot configure more than one trend on a given tag, and each trend configuration cannot belong to more than one group/worksheet.

If you make extensive changes to the tags database after you save an Trend worksheet, it might not be possible to recreate the body of the worksheet the next time you open it. For example, if you copy all of the tags from the tags database (in Datasheet View) to a spreadsheet program, use that program to sort the tags, and then copy the tags back to the tags database, most or all of the tag properties will be reset in the process.

> **Note:** The Trend task can accept only up to 1000 tags in a single worksheet. If you manually configure more than 1000 tags in the same worksheet, the Trend task will generate an error during project run time.

### *Sort or filter the rows in a worksheet*

Sort or filter the rows in a worksheet in order to make it easier to browse the rows or find a specific item.

Before you begin this task, you must have already inserted a worksheet and opened it for editing. You should also be familiar with how sorting and filtering is done in general-purpose spreadsheet applications.

Please note that you can sort or filter rows only in the following types of worksheets:

- The Project Tags, Shared Tags, and System Tags datasheets;
- The Translation Table worksheet;
- All task worksheets except Report and Script, which do not have rows; and
- All communication worksheets.

None of the other worksheets have rows to sort or filter.

Sorting is done alphanumerically, by the selected column, in either ascending (0–9, A–Z) or descending (Z–A, 9–0) order.

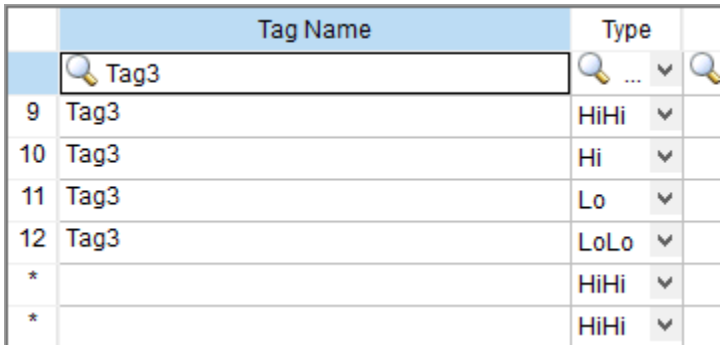
| | Tag Name | Type |
|---|---|---|
| | Filter text | ... |
| 1 | Tag1 | HiHi |
| 2 | Tag1 | Hi |
| 3 | Tag1 | Lo |
| 4 | Tag1 | LoLo |
| 5 | Tag2 | HiHi |
| 6 | Tag2 | Hi |
| 7 | Tag2 | Lo |
| 8 | Tag2 | LoLo |
| 9 | Tag3 | HiHi |
| 10 | Tag3 | Hi |
| 11 | Tag3 | Lo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

***Alarm worksheet rows in their original order***

| | Tag Name | Type |
|---|---|---|
| | Filter text | ... |
| 10 | Tag3 | Hi |
| 2 | Tag1 | Hi |
| 6 | Tag2 | Hi |
| 9 | Tag3 | HiHi |
| 5 | Tag2 | HiHi |
| 1 | Tag1 | HiHi |
| 3 | Tag1 | Lo |
| 11 | Tag3 | Lo |
| 7 | Tag2 | Lo |
| 8 | Tag2 | LoLo |
| 4 | Tag1 | LoLo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

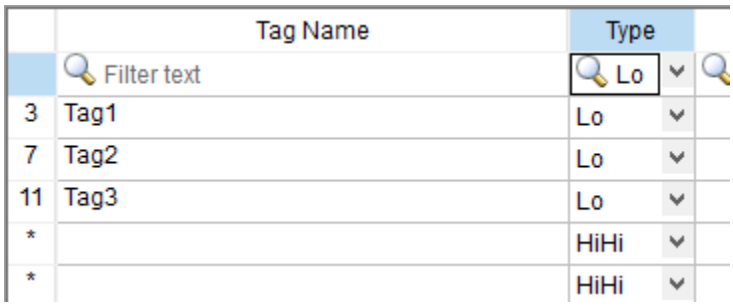
***Alarm worksheet rows sorted by Type***

Filtering is done according to whatever string you enter in the selected column. Only the rows that match the string will be displayed.



| | Tag Name | Type |
|---|---|---|
| | Tag3 | ... |
| 9 | Tag3 | HiHi |
| 10 | Tag3 | Hi |
| 11 | Tag3 | Lo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

***Alarm worksheet rows filtered where Tag Name is "Tag3"***



| | Tag Name | Type |
|---|---|---|
| | Filter text | Lo |
| 3 | Tag1 | Lo |
| 7 | Tag2 | Lo |
| 11 | Tag3 | Lo |
| * | | HiHi |
| * | | HiHi |

***Alarm worksheet rows filtered where Type is "Lo"***

> **Tip:** You can still delete rows while they are sorted or filtered.

To sort or filter rows:

1. To sort the rows, click the header of the column by which you want to sort. Click once to sort in ascending order, and then click again to sort in descending order.

   The current order (i.e., the direction of the sort) is indicated by the arrow to the right of the column name.

   > **Note:** You cannot sort by multiple columns.

2. To undo the sorting and restore the rows to their original order, click the header of the first (numbered) column.
3. To filter the rows, type the string that you want to match in the top (zero) row of the worksheet and then press either **Tab** or **Return**.

   You may include `*` and `?` as wildcard characters in your string:

   - `*` matches any number of characters, including none. For example, `Tag*` would match **`Tag`**, **`Tag3`**, **`Tag34567`**, **`TagA`**, and **`Tag_TEMP`**.
   - `?` matches exactly one character. For example, `Tag?` matches **`Tag3`** and **`TagA`**, while `Tag?????` matches **`Tag34567`** and **`Tag_TEMP`**.

   Also, you may filter by multiple columns. Only the rows that match the filter strings in all columns will be displayed.
4. To undo the filtering and restore the rows to their original order, delete the string that you typed and then press either **Tab** or **Return**.

Please keep in mind that sorting or filtering the rows of a worksheet only helps you to edit that worksheet. It does not change how the worksheet is executed during run time. The rows will be executed in their original numbered order (i.e., the leftmost column) unless you actually move or delete a row.

### *Creating Batch History*

EcoStruxure Machine SCADA Expert provides powerful tools that enable the user to create and manage batch historical information. The user is able to create batches by using the following formats:

- **Proprietary**: When using the proprietary format, each batch will be stored on a different historical file. The user can save historical data in both the normal historical file and batch files at the same time (see Trend Folder for more information about these files).
- **Database**: The historical data used for the batch is saved in the same table as the normal historical data; an additional table called BatchHistory keeps registers with the information about the batches. The list below describes the fields on the BatchHistory table:

| Field Name | Data Type | Description |
|---|---|---|
| Group_Number | Integer | Trend group number. This is the number of the worksheet that you are creating to specify the tags that will be stored on your batch history. |
| Batch_Name | String | Name of the batch |
| Start_Time | TimeStamp | Date and Time that the batch was started. |
| End_Time | TimeStamp | Date and Time that the batch was finished |
| Pri_Table | String | Reserved |
| Sec_Table | String | Reserved |
| Description | String | Batch description |
| Deleted | Boolean | **0**: Batch has not been deleted<br>**1**: Batch has been deleted |

**Tip:** You can customize the name of the table and the name of the columns created in the database by editing the *<project name>*.APP file, as follows:

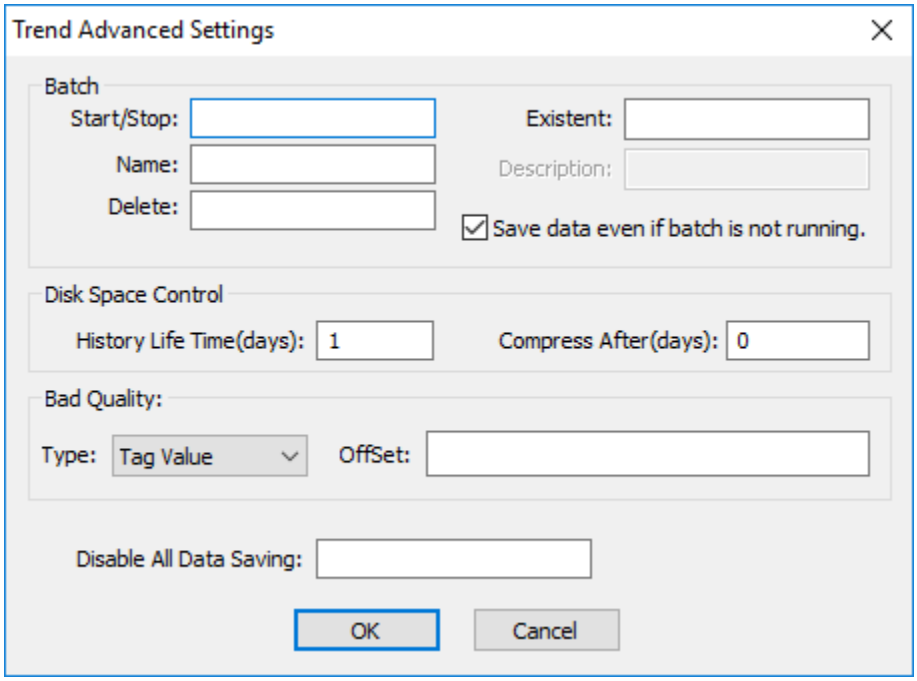
```
[Trend]
DefaultName=NewName

[TrendGroupPRI|SEC]
BatchHistory=TableName
```

For example:

```
[TREND001PRI]
BatchHistory=MyTableForPrimaryDB

[TREND001SEC]
BatchHistory=MyTableForSecondaryDB

[Trend]
Group_Number=Trend_Worksheet
Batch_Name=Load_Number
```

### Batch History Configuration

When you add a Trend worksheet (see Trend folder) and click the Advanced button, the following window will display:




***Trend Advanced Settings***

In the **Batch** pane, you can configure the saving of the batch history:

- **Start/Stop (input):** Enter the tag that will start/stop your batches. When the tag in this field is set to TRUE (different from 0), EcoStruxure Machine SCADA Expert will either start saving data to your batch file (if you are using proprietary format), or add a new register to the BatchHistory table on your database, indicating that a batch has been started. Note that historical data will be saved according to the configuration in the fields **Save Trigger** and **Save On Tag Change** options on the Trend Worksheet.
- **Name (input):** This field represents the batch name; its meaning depends on the format selected on the Trend Worksheet:
  - If you selected **Proprietary** in the **Type** field, the **Name** should comply with the format [Path]<FileName>, where:
    - **Path**: An optional field. If the path is not specified, the batch history file will be stored in the same path as the ***<project name>*.app** file.
    - **FileName**: Name of the batch history file.
  - If you selected **Database** in the **Type** field, the value in this field will be stored in the **Batch_Name** field of the *BatchHistory* table.

> **Tip:** You can enter tag names between curly brackets in this field (e.g., `C:\MyBatches\{MyTagWithName}{MyTagWithNumber}.hst`).

- **Delete (input):** When the tag specified in this field changes its value, the batch will be deleted. With the **Proprietary** format, the batch history file will be removed. With the **Database** format, it will set the **Delete** field in the *BatchHistory* table to true, but the saved historical data will remain. The Trend object only sees batches that have the delete field set to `0` (zero).
- **Existent (output)**: The tag entered on this field will receive the value `1` if the batch specified in the **Name** field already exists; otherwise the tag will receive the value `0`.

- **Description (output)**: This field is available only when using the **Database** format. When the tag in the **Start/Stop** field changes to TRUE, the register that is added to the *BatchHistory* table will display the string in this field.

> **Tip:** You can enter tag names between curly brackets in this field (e.g., {MyTag})

- **Save data even if batch is not running**: If this field is unchecked, the historical data will be saved only when the tag in the **Start/Stop** field is TRUE.

> **Tip:** The Batch Historical data can be displayed to the user in either Graphical or Table format. See Trend Folder or Grid Object to display information in these formats.

In the **Disk Space Control** area, you can control disk usage:

- **History Life Time (days)** field: Specify how many days to keep the history file on the disk. After the specified period, EcoStruxure Machine SCADA Expert automatically erases the file. Use this option only for files based on a date.
- **Compress After (days)** field: Specify how many days to keep the trend history file ( **`*.hst`** ) on the disk before compressing the file. After the specified period, EcoStruxure Machine SCADA Expert automatically compresses the file. Use this option only for files based on a date. This option is not available for Windows Embedded target systems.

In the **Bad Quality** area, you can determine what value will actually be saved in the batch history when the tag quality is BAD:

| Type | Description |
|---|---|
| **Tag Value** | The actual value of the project tag when the tag quality was BAD, plus the specified **Offset** (if any). |
| **Min Value** | The minimum historical value of the project tag, minus the specified **Offset** (if any). |
| **Max Value** | The maximum historical value of the project tag, plus the specified **Offset** (if any). |
| **Value** | The specified **Value** only. |
| **NaN** | Not a number.<br>Please note that when **History Format** is **Database** and **Bad Quality** is **NaN**, all of the database fields will be saved as Float type.<br>Also, if a Trend Control screen object is configured to use the history generated by this Trend worksheet, then NaN entries are counted as 0 for the purpose of calculating a trend's statistical average and deviation. |

> **Note:** The **Bad Quality** feature cannot be used in projects running on Windows Embedded target systems.

Finally, in the **Disable All Data Saving** box, type the name of a project tag. When the value of the tag is TRUE (non-zero) during runtime, all data saving is disabled for this worksheet. Other Trend worksheets are not affected.

## *Converting Trend History Files from Binary to Text*

By default, EcoStruxure Machine SCADA Expert saves trend history files in a binary format (**`.hst`**). Because you may want to have these files in **`.txt`** format, EcoStruxure Machine SCADA Expert provides the **HST2TXT.EXE** program to convert trend history files from binary into text format.

To convert a file, use the following procedure:

1. At the command prompt, change directory (`cd`) to the Bin sub-folder of the Studio program folder, typically at:

```
cd C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
 2020\Bin
```

2. At the command prompt, copy the **`Hst2txt.exe`** into the same directory where the **`.hst`** file is located.
3. At the command prompt, type **`Hst2txt.exe`** and specify the following parameters:

- ***filename***: Name of the trend history file to convert
- ***[separator]***: Data separator character (*default* is **<TAB>**)
- ***[/e]***: Extended functionality (convert data with more than 10 characters)
- ***[/i:HH:MM:SS]***: Start time in hours (**HH**), minutes (**MM**), and seconds (**SS**)
- ***[/f:HH:MM:SS]***: Finish time in hours (**HH**), minutes (**MM**), and seconds (**SS**)
- ***[/m]***: Include milliseconds in the **Time** column (Type **1** to print the milliseconds value in the text file created from the **.hst** file.)

  For example:

  ```
  Hst2txt.exe 01952010.hst
  ```

  The program creates a **.hdr** (header) file and the **.txt** file, which are both plain text files that can be viewed using any text editor (for example, Notepad).
- The **.hdr** file contains the name of the tags configured in the Trend Worksheet.
- The **.txt** file contains the tag values saved in the history file.

4. After the program converts the file, type **Exit** to close the DOS window.

> **Note:** Alternatively, you can use the **HST2TXT** math script in a Math worksheet to convert binary files into text format automatically without having to use a DOS window.

**See also**:

- Converting Trend History Files From Text to Binary
- Creating Batch History
- Configuring a Default Database for All Task History

### *Converting Trend History Files from Text to Binary*

Use the `TXT2HST.exe` program to convert text-based history files back into binary format.

Before you begin this task, you must have both of the following:

- The text file (e.g., `02950201.txt`) that contains the historical data; and
- The corresponding header file (e.g., `02950201.hdr`) that provides the column numbers and names.

To convert a file, use the following procedure:

1. At the command prompt, change directory (`cd`) to the Bin sub-folder of the Studio program folder, typically at:

   ```
   cd C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
    2020\Bin
   ```

2. At the command prompt, copy the `TXT2HST.exe` into the same directory where the text and header files are located.
3. At the command prompt, type `TXT2HST.exe` and specify the following parameters:
   - ***filename***: Name of the text file
   - ***[separator]***: Data separator character (default is **<TAB>**)
   - ***[/e]***: Extended functionality (data value with more than 10 characters)
   - ***[/i:HH:MM:SS]***: Start time of data value in hours (**HH**), minutes (**MM**), and seconds (**SS**)
   - ***[/f:HH:MM:SS]***: Finish time of data value in hours (**HH**), minutes (**MM**), and seconds (**SS**)

For example:

```
TXT2HST.exe 02950201.txt
```

The program combines the text file and header file in order to create a binary file with the same name (e.g., `02950201.hst`).

4. When the program is finished, type **`exit`** to close the command prompt.

> **Note:** You cannot create a math script for the `TXT2HST.exe` program and use it in a Math worksheet in order to convert text files into binary format, as you can for the `HST2TXT.exe` program. The math script shortcut is available for binary files only.

### *Make trend history accessible through OPC HDA*

Use the built-in OPC HDA server to make historical data generated by Trend worksheets accessible to other computers.

> **Note:** This feature is not supported in projects that are configured to run on the Embedded target platform. For more information, see About target platforms, product types, and target systems on page 100.

The OPC Historical Data Access (OPC HDA) specification is used to exchange archived process data, such as the historical data generated by Trend worksheets in your project. This is in contrast to other OPC specifications — for example, OPC DA (also known as OPC Classic) and OPC UA — that are used to exchange real-time data.

The project runtime software for Windows includes a built-in OPC HDA server that you can use to make your project's trend history accessible to OPC HDA clients. The server only works with trend history that has been saved in the **Proprietary** format, as opposed to the **Database** and **Historian** formats. When the server is enabled, it automatically scans the project folder for all saved history files (.hst) and then makes the contents of those files accessible.

There are no user-configurable settings for the OPC HDA server itself, but to enable the server, you must ensure the **HDA OPC Server** task is started during project run time. You can configure the task to start automatically when the project is run, or you can manually start the task after the project is running. For more information, see Runtime Tasks on page 136.

Once your project is running and the task is started, you should be able to use any compatible OPC HDA client program to access the historical data. The OPC HDA server address is the same as your project's data server address, its port number is 135 (DCOM), and it should appear to the client as "Studio.Scada.HDA.OPC".

This feature only supports the 1.0 version of the OPC HDA specification.

## Trend Control object

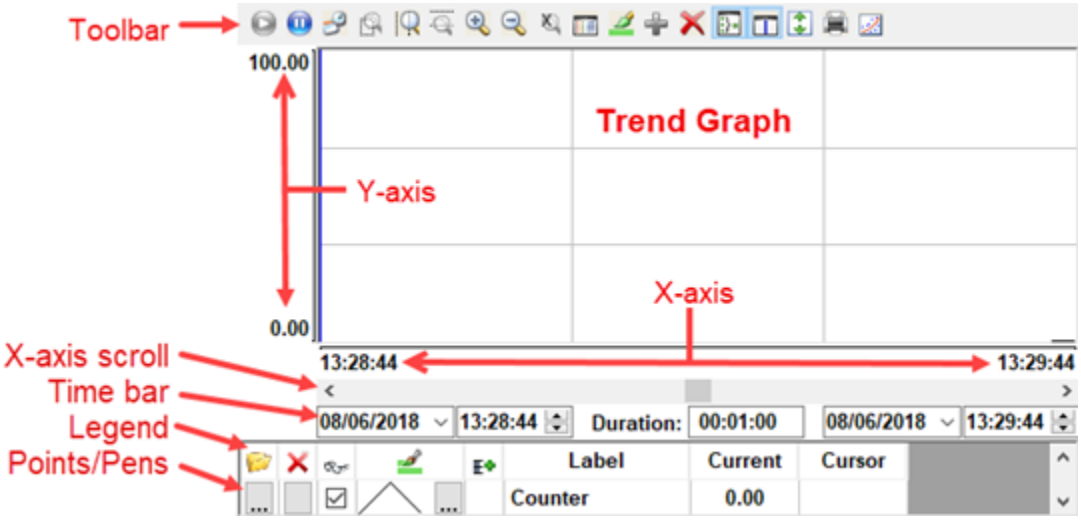
The Trend Control object displays data points (values) from different data sources in a graphic format.

The main features provided by the Trend Control object are:

- Display of multiple pens simultaneously
- Support for different Data Sources, such as Tag, Batch, Database and Text File
- Capability to generate X/Y graphs from the configured data sources (please refer to Appendix A for an example of an X/Y chart).
- Simultaneous display of an unlimited number of data points. This feature might be limited by the hardware used since available memory and performance will vary.
- Built-in toolbar, which provides interfaces for the user to interact with the Trend Control object during runtime
- Built-in legend, which displays the main information associated to each pen linked to the object
- Zooming and auto-scaling tools
- Horizontal and vertical orientation

### *About the trend control runtime interface*

During project runtime, a trend control has its own built-in interface that the operator can use to change how trends are displayed. This section describes the major parts of the interface and how they are used.

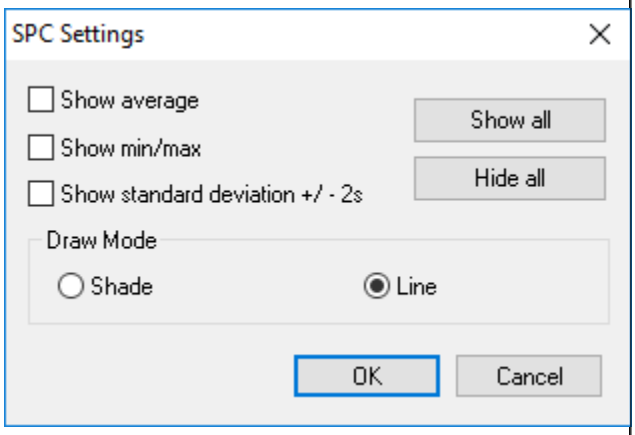



*Trend control runtime interface*

#### Toolbar

| Command/Tool | Icon | Description | Activation Tag |
|---|---|---|---|
| Run | | Sets the trend control to Run Mode (a.k.a. Online Mode). In this mode, the X-axis continues to scroll with the passage of time and the trends are updated with current tag values. | 0 = stop trend<br>1 = run trend<br>The Activation Tag is only one for both Run and Stop commands and it is configured on the "Run/Stop" field |
| Stop | | Sets the trend control to Stop Mode (a.k.a. Historical Mode). In this mode, the X-axis is stopped and the trends display only historical data. | 0 = stop trend<br>1 = run trend |

| Command/Tool | Icon | Description | Activation Tag |
|---|---|---|---|
| | | If decimation is enabled for one or more trends, the calculation and redrawing is done only in this mode. | The Activation Tag is only one for both Run and Stop commands and it is configured on the "Run/Stop" field |
| Period | | Opens a dialog which can be used to modify the X-axis scale main settings. | 1 = open dialog<br>Resets to 0 after open. |
| Window Zoom | | Allows the user to click on the trend graph and drag the cursor to select the area that must be visible when the cursor is released.<br>This option is disabled when the **Multiple Section** option (for the Y scale) is active. | 0 = disable zoom<br>1 = enable zoom<br>Resets to 0 after user input. |
| Horizontal Zoom | | Allows the user to click on two points in the trend graph, defining the horizontal scale that must be available. | |
| Vertical Zoom | | Allows the user to click on two points in the trend graph, defining the vertical scale that must be available.<br>This option is disabled when the **Multiple Section** option (for the Y scale) is active. | |
| Zoom In | | Zooms in (i.e., halves the current X and Y scales) each time the user clicks the tool. | 1 = execute command<br>Resets to 0 after execution. |
| Zoom Out | | Zooms in (i.e., doubles the current X and Y scales) each time the user clicks the tool. | |
| Cancel Zoom | | Cancels the current zoom and returns the trend graph to its original scale. | |
| Legend Properties | | Opens a dialog which can be used to modify the Legend main settings. | |
| Pen Style | | Opens a dialog which can be used to modify the pen style of the selected trend. | |
| Add Pen | | Opens a dialog which can be used to add a new trend to the trend control. | |
| Remove Pen | | Removes the selected trend from the trend control. | |
| Multiple Sections | | Switches the Y scale between Multiple Sections (a section for each trend) and Single Section (all trend share the same Y scale section). | 0 = Single Section<br>1 = Multiple Sections |
| Cursor | | Switches the cursor (ruler) between visible and hidden. | 0 = Cursor hidden<br>1 = Cursor visible |
| Auto Scale | | Changes the Y axis scale to fit all values from the trends that are currently being monitored. | 1 = execute command<br>Resets to 0 after execution. |
| Print | | Prints the current state of the trend control. (Historical data are not printed.) | |

| Command/Tool | Icon | Description | Activation Tag |
|---|---|---|---|
| SPC | | Opens a dialog which can be used to show the statistical process control (SPC) information for the selected trend:<br>  <br><br>• Draw Mode…<br>  • **Shade**: Draws the average value as a dashed line, and draws the min/max values and standard deviation as shaded areas.<br>  • **Line**: Draws the average value and standard deviation as dashed lines, and draws the min/max values as solid lines.<br>• **Show average**: Show the calculated average of all of the trend's historical values.<br>  **Note:** When a value is not a number (NaN) — for example, when a tag is flagged as BAD quality — it is counted as 0 for the purpose of calculating the average.<br>• **Show min/max**: Show the minimum and maximum historical values of the trend.<br>• **Show standard deviation**: Show the standard deviation of the trend. A low standard deviation indicates that the actual value tends to stay close to the average; a high standard deviation indicates that the actual value tends to vary greatly from the average. | The tag's Bit properties (**B0**–**B4**) can be used to open the dialog and pre-select options:<br>• ***tagname*->B0**<br>  • 1 = open dialog<br>  Resets to 0 after open.<br>• ***tagname*->B1**<br>  • 0 = **Draw Mode: Line** selected<br>  • 1 = **Draw Mode: Shade** selected<br>• ***tagname*->B2**<br>  • 0 = **Show average** cleared<br>  • 1 = **Show average** selected<br>• ***tagname*->B3**<br>  • 0 = **Show standard deviation** cleared<br>  • 1 = **Show standard deviation** selected<br>• ***tagname*->B4**<br>  • 0 = **Show min/max** cleared<br>  • 1 = **Show min/max** selected |

**Note:** Activation tags are configred in the trend control's object properties.

For more information, see Toolbar dialog on page 440.

### Time bar

The time bar displays the start date/time and end date/time of the graph, or in other words, the period of the X axis. For more information, see Axes dialog on page 436.

If you have configured the trend control to display historical data from a batch file, you can change these dates and times during project run time and therefore increase or decrease the period of the graph. The longer the period, the more data the graph will display but at lower resolution. The shorter the period, the less data the graph will display but at higher resolution. This is similar to using the Horizontal Zoom tool.

**Note:** If you enter a date (start or end) before 01/01/2000, it will be automatically adjusted to 01/01/2000. And if you enter a date (start or end) after the current date, it will be automatically adjusted to the current date.

All dates must be formatted according to the current date format. For more information, see About the date format and how to change it on page 694.

### Legend

| Command | Icon | Description |
|---|---|---|
| Selection | | Launches a dialog, where the user can replace the data point associated with the selected trend on the legend |
| Remove | | Removes the selected trend from the trend control |
| Hide | | When checked, the selected trend is visible; otherwise, it is hidden. |
| Pen Style | | Launches an embedded dialog, where the user can modify the pen style of the selected trend. |
| Scale | | When this box is checked, the Y axis scale is visible; otherwise, it is hidden. The scale can be hidden only when the Multiple Sections option is off. |

For more information, see Legend dialog on page 443.

## *Object Properties: Trend Control dialog*

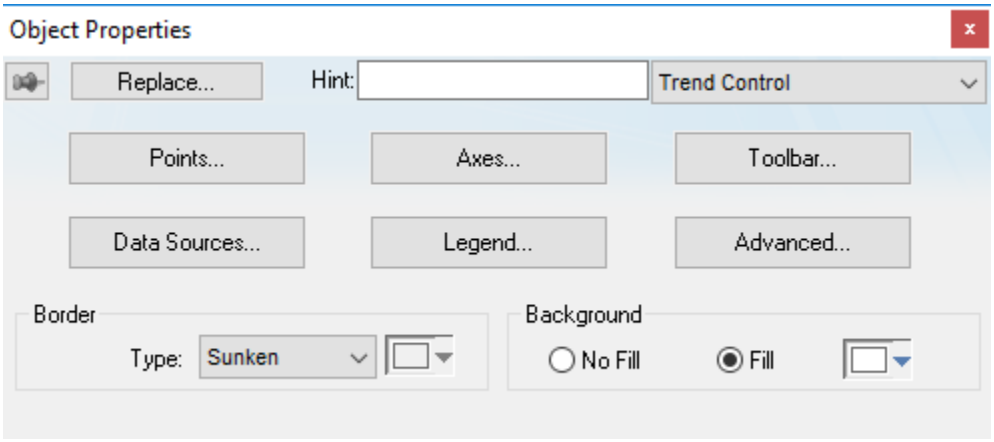
The *Object Properties: Trend Control* dialog is used to configure the basic properties of a Trend Control screen object.

### Accessing the dialog box

To access the *Object Properties* dialog box for a screen object, do one of the following:

- Select the screen object, and then on the **Draw** tab of the ribbon, in the **Editing** group, click **Properties**;
- Right-click the screen object, and then click **Properties** on the shortcut menu; or
- Double-click the screen object.

### The dialog box in detail




***Object Properties: Trend Control dialog***

In addition to the elements that are common to all *Object Properties* dialog boxes, the *Object Properties: Trend Control* dialog box contains the following elements:

| Area / Element Name | | Description |
|---|---|---|
| **Border** | **Type** | Sets the type of border around the graph area of the trend control. (There are no borders around the trend control's legend or toolbar.) |
| | **Color** | Sets the color of the border, if the border type is **Solid**. For more information, see Selecting colors and fill effects. |

| Area / Element Name | | Description |
|---|---|---|
| **Background** | **No Fill / Fill** | Enables the background fill for the graph area of the trend control. (There are no backgrounds for the trend control's legend or toolbar.) If the fill is not enabled, then the graph is transparent to whatever other screen objects are behind the trend control. |
| | **Color** | Sets the color and fill effect of the background fill, if it is enabled. For more information, see Selecting colors and fill effects. |
| **Points** | | Opens the, which allows configuration of the trend control's data points (or pens). For more information, see *Trend Control: Points* dialog. |
| **Axes** | | Allows configuration of the trend control's X and Y axes, as well as its horizontal or vertical orientation. For more information, see *Trend Control: Axes* dialog. |
| **Toolbar** | | Allows configuration of the user toolbar that is displayed above the trend control. For more information, see *Trend Control: Toolbar* dialog. |
| **Data Sources** | | Allows configuration of multiple data sources for the trend. For more information, see *Trend Control: Data Sources* dialog. |
| **Legend** | | Allows configuration of the legend that is displayed below the trend control. For more information, see *Trend Control: Legend* dialog. |
| **Advanced** | | Allows configuration of the trend control's advanced properties, such as runtime options and tag triggers. For more information, see *Trend Control: Advanced* dialog. |

Although the Trend Control object supports flexible configurations to meet the specific needs of your project, most of the settings are set by defaults based on the most common interfaces. Therefore, in many cases, you will only configure data points (displayed during runtime), which can be done easily by clicking the **Points** button from the *Object Properties* window.

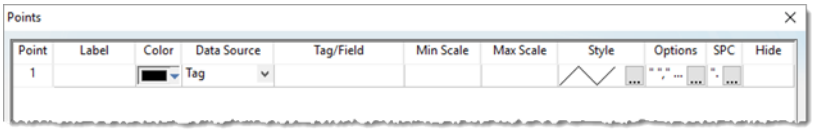
## POINTS

The *Points* dialog is used to configure the data points for a Trend Control screen object. The value of each data point is represented as a pen in the trend display. You can dynamically change which data points are visible during run time, regardless of how many data points are associated with the screen object.

### Accessing the dialog

To access the *Points* dialog for a specific Trend Control screen object, first access the *Object Properties* dialog for that screen object and then click **Points**.

### The dialog in detail




***Points dialog***

The following table summarizes the properties of each data point:

| Column Name | Description |
|---|---|
| **Point** | A unique ID number for the point, which is assigned automatically when the point is created in this interface. |
| **Label** | The label associated with the Point can be displayed on the Legend, during run time, providing a short reference to the user for each Point. |
| **Color** | The color of the pen usedColor of the pen used to draw the values of the Point on the Trend Control object. |
| **Data Source** | The data source for this point. **Tag** is available by default, but all other sources must be configured in the *Data Sources* dialog. |
| **Tag/Field** | The meaning of this parameter depends on the **Data Source** type associated with the data point:<br>• If **Data Source** is **Tag**, type the name of the tag with values to be displayed. If the tag is configured in a Trend task worksheet, its history will be automatically retrieved and displayed. Otherwise, only the tag's online values — that is, the tag's actual values during |

| Column Name | Description |
|---|---|
| | run time — will be displayed. (Please note this means that the tag's trend line cannot be redrawn after zooming; only new values can be drawn as they are received. For more information about zooming, see About the trend control runtime interface on page 424.)<br>• If **Data Source** is **Batch**, type the name of the tag with values to be retrieved from the Batch History file that is generated by the Trend task worksheet.<br>• If **Data Source** is **Database**, type the name of the field (column) in the SQL Relational Database that contains the point's values. For more information, see Using the Data Source Database on page 452.<br>• If **Data Source** is **Text File**, type the number of the column in the text file that contains the point's values. The number 0 refers to the first column, 1 refers to the second column, and so on. For more information, see Using the Data Source Text File on page 449. |
| **Min Scale** / **Max Scale** | The scale of the Y-axis for this point. This overrides the default scale that is set in the *Axes* dialog.<br>**Note:** The **Min Scale** and **Max Scale** properties can hold real numeric values up to six decimal places. If you need more precision than that, you must configure the **Min Scale** and **Max Scale** properties with Real tags and then store the values in those tags. |
| **Style** | The line and marker styles for this point; click the … button to open the *Pen Style* dialog. |
| **Options** | Additional options for this point; click the … button to open the *Options* dialog. |
| **SPC** | Calculated statistics to be used in statistical process control (SPC); click the … button to open the *SPC* dialog. |
| **Hide** | Tag trigger — when the value is TRUE, the data point is hidden in the trend display. |

## Pen Style dialog box

Use the *Pen Style* dialog box to customize the style of the pen that draws the point's values during run time.

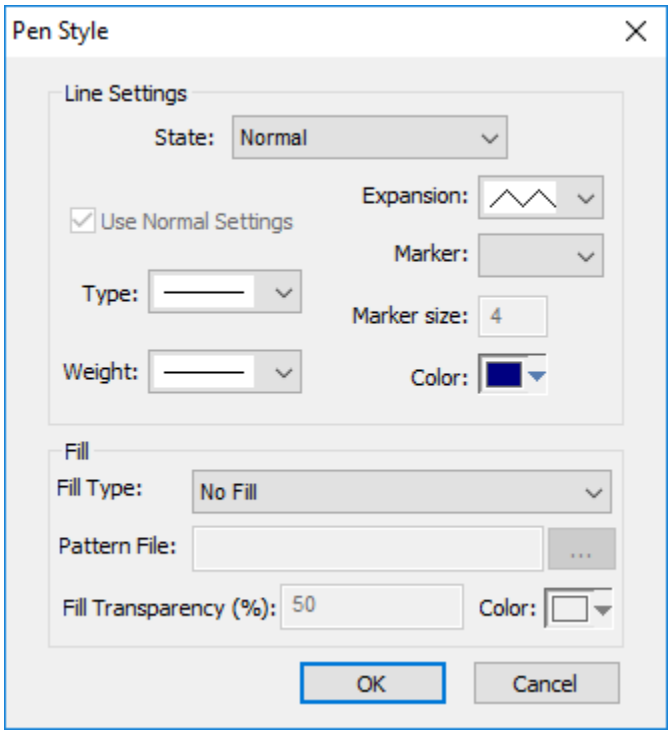
### Accessing the dialog box

To access the *Pen Style* dialog box for a specific point, do the following:

1. In the screen editor, select the Trend Control object, and then open its object properties.
2. In the object properties, click **Points**.
3. In the *Points* dialog box, select a point, and then click the **Style** column for that point.

This dialog box can also be opened on the project client during run time, by clicking the Pen Style tool in either the toolbar or the legend of the trend control. For more information, see About the trend control runtime interface on page 424.

### The dialog box in detail



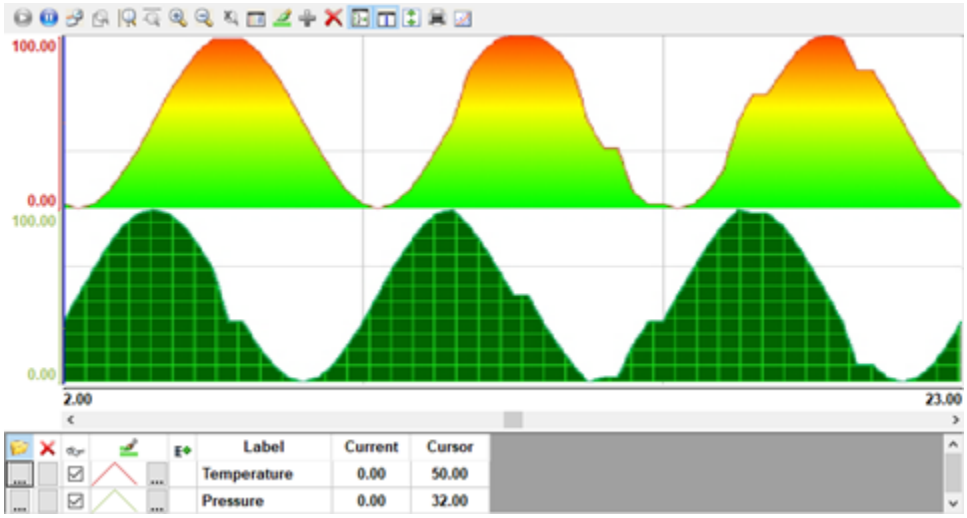

***Pen Style dialog box***

The *Pen Style* dialog box includes the following elements:

#### Elements in the *Pen Style* dialog box

| Area / Element Name | | Description |
|---|---|---|
| **Line Settings** | **State** | You have the option of defining a Hi Limit and a Lo Limit for each data Point, with the Options dialog. The Pen Style Dialog allows you to configure different settings for the pen (e.g., color), both when its values are within the limits (Normal State) and not within the limits (Out of Limits state). |
| | **Use Normal Settings** | Available only for the **Out of Limits** state. When checked, the pen will always be displayed with the settings for the Normal state, even if the data point values are not within the limits configured for it. |
| | **Type** | The type of line (e.g., solid, dashed, dotted) that connects the data points. |
| | **Weight** | The weight of the line that connects the data points. |
| | **Expansion** | The algorithm used to connect the points, as follows:<br>• : Consecutive points are directly connected to each other by an analog line. This option is suitable for numerical values.<br>• : Consecutive points are connected only through horizontal or vertical steps (depending on the orientation of the trend display). This option is suitable for Boolean values. |
| | **Marker** | The shape used to mark each data point. If no shape is selected, then only the connecting line between points is displayed. |
| | **Marker size** | The size of the data point marker. |
| | **Color** | The color of the trend line and data point markers. |
| **Fill** | **Fill Type** | The type of fill between the trend line and the number line. |
| | **Pattern File** | The graphic file used to fill the trend area. Available only Fill Type is set to **Custom Pattern**. |

| Area / Element Name | | Description |
|---|---|---|
| | | Click the browse button to open a Windows file browser and then select the desired graphics file. The file should be located in your project folder.<br><br>See below for an example of trends with custom fill patterns.<br><br>**Note:** This feature should be used only with small images that can be tiled to fill the trend area. If you select a large image instead, with the intent of having it fill the entire trend area by itself, you might see unexpected behavior during project run time — for example, the image might not align properly within the trend area or it might ovelap itself in strange ways, depending on the size and position of the Trend Control object in the screen. |
| | **Color** | The color used to fill the trend area. Available only when Fill Type is set to **Solid Color**. |
| | **Fill Transparency (%)** | The transparency level of the fill. (If the fill is transparent, then other trends behind it can be seen through it, making the entire graph easier to read.) Available for both **Custom Pattern** and **Solid Color**. |




***Trends with custom fill patterns***

**Note:** When your project is viewed on either a Windows Embedded device or a Thin Client (any OS), the *Pen Style* dialog box — which can be opened during run time; see "Accessing the dialog box" above —allows the user to change the pen color only.

**Tip:** To programmatically modify these pen style settings during run time, configure **Style Modifer** in the *Options* dialog box. For more information, see Modify the pen style of a point during run time on page 434.

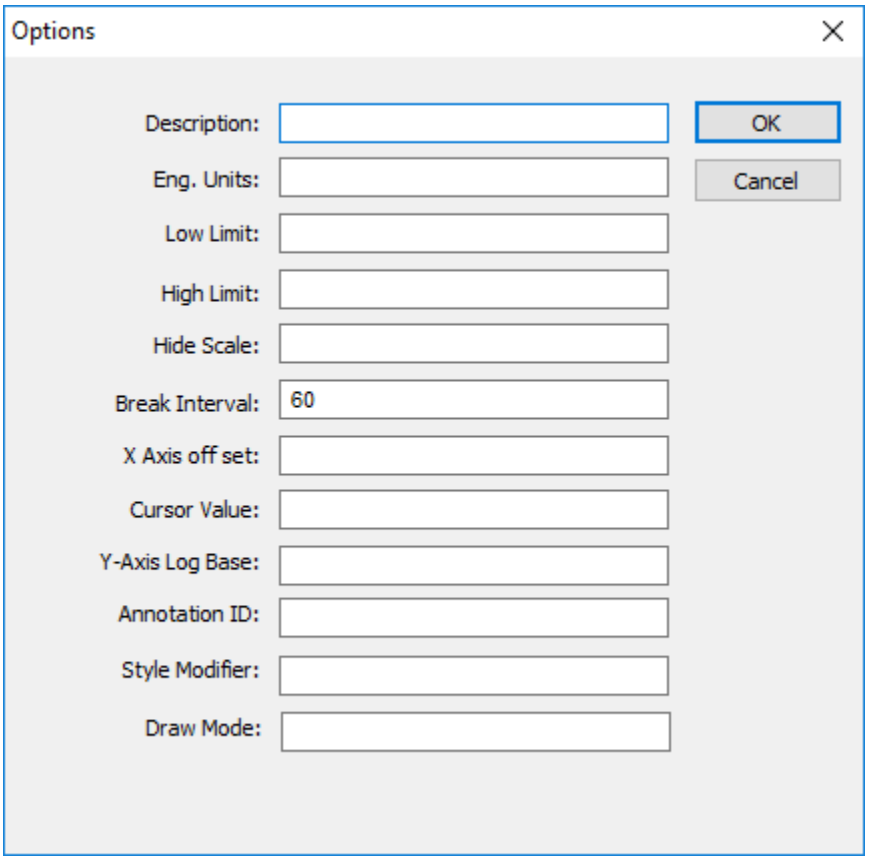
## Options

Use the *Options* dialog box to configure additional options for a specific point in a Trend Control object.

### Accessing the dialog box

To access the *Options* dialog box for a specific point: first access the *Points* dialog box, where all of the points in a Trend Control object are configured, and then click the **Options** column for that point. For more information, see Trend Control object on page 424.

### The dialog box in detail




***Options dialog box***

The *Options* dialog box includes the following settings:

**Description**

This text can be displayed in the legend of the Trend Control object during run time, providing a brief description of the trend.

If you specify a tag name in curly brackets (e.g., `{MyTag}`), that tag's Description property is used.

**Eng. Unit**

This text can be displayed in the legend of the Trend Control object during run time, providing the engineering unit (i.e., the unit of measurement) associated with the trend.

If you specify a tag name in curly brackets (e.g., `{MyTag}`), that tag's Engineering Unit property is used.

**Lo Limit**

Type a tag name or numerical value. When the trend falls below this value during run time, it can be drawn in a different style (e.g., color). For more information, see Pen Style dialog box on page 429.

If you specify a tag name in curly brackets (e.g., `{MyTag}`), that tag's LoLimit property is used.

**Hi Limit**

Type a tag name or numerical value. When the trend rises above this value during run time, it can be drawn in a different style (e.g., color). For more information, see Pen Style dialog box on page 429.

If you specify a tag name in curly brackets (e.g., `{MyTag}`), that tag's HiLimit property is used.

**Hide Scale**

Type a tag name or numerical value. When the value is TRUE (i.e., not zero), the Y-axis scale associated with this trend is hidden during run time.

> **Note:** This setting applies only when the Trend Control object is configured to display a single section in the Y axis. In other words, the **Multiple Section** option in the *Axes* dialog box must be cleared. For more information, see Axes dialog on page 436.

**Break Interval**

Type a numerical value (default is 7200). This is the maximum interval allowed between two consecutive points in a trend. If the interval between the two points is greater than this value, the Trend Control object assumes that no data were collected during the interval and it does not draw a line connecting the two points.

If the X axis is configured to be numeric, the value specified here is taken as a numeric scalar value. If the X axis is configured to be date/time, the value specified here is taken as seconds.

This setting accepts some special values:

| Value | Description |
|---|---|
| -1 | Do not connect any points. |
| -2 | Connect only points that have ascending values. |

**X Axis Offset**

Type a tag name or numerical value. The value is the offset from the X-axis scale configured for the Trend Control object. This setting is useful when you want to display data from two or more trends using a different scale for each trend, so that you can compare them.

If the X axis is configured to be numeric, the value specified here is taken as a numeric scalar value. If the X axis is configured to be date/time, the value specified here is taken as seconds.

**Cursor Value**

Type the name of a project tag. During run time, the tag is continuously updated with the value of the trend where it is intersected by the vertical cursor (if any) in the Trend Control object.

**Y-Axis Log Base**

Type a tag name or numerical value. If the value is 0 (or the box is left empty), the Y axis of the trend is a normal linear scale. If the value is anything other than 0, the Y axis is a logarithmic scale with a log base equal to that value. The most common log base is 1, which gives a scale of 1, 10, 100, 1000, and so on, but you can specify any log base.

**Annotation ID**

Type a unique ID with which annotations can be associated. This setting is optional; annotations can also be associated with the point's tag/field, but it is better to associate them with the annotation ID in case the tag/field is changed. For more information, see Display text- and image-based trend annotations in a trend control on page 459.

**Style Modifier**

Modify the pen style's expansion, line color, line weight, or line type. For more information, see Modify the pen style of a point during run time on page 434.

**Draw Mode**

Type a tag name or numerical value. If the value is 1, the historical data for this trend are decimated before the trend is drawn in the Trend Control object. That means the trend's X axis is divided into a number of intervals (as determined by **Max Points** in the Advanced settings), and then all of the data points within each interval are averaged together to be drawn as a single point.

This is similar to the **Decimation** option in the Advanced settings, except that the decimation is done only for this trend rather than for all trends in the Trend Control object.

> **Note:** If decimation is enabled and the X axis is configured to be numeric rather than date/time, the data used in the X axis must be properly sorted. For more information, see the Data Sources settings.

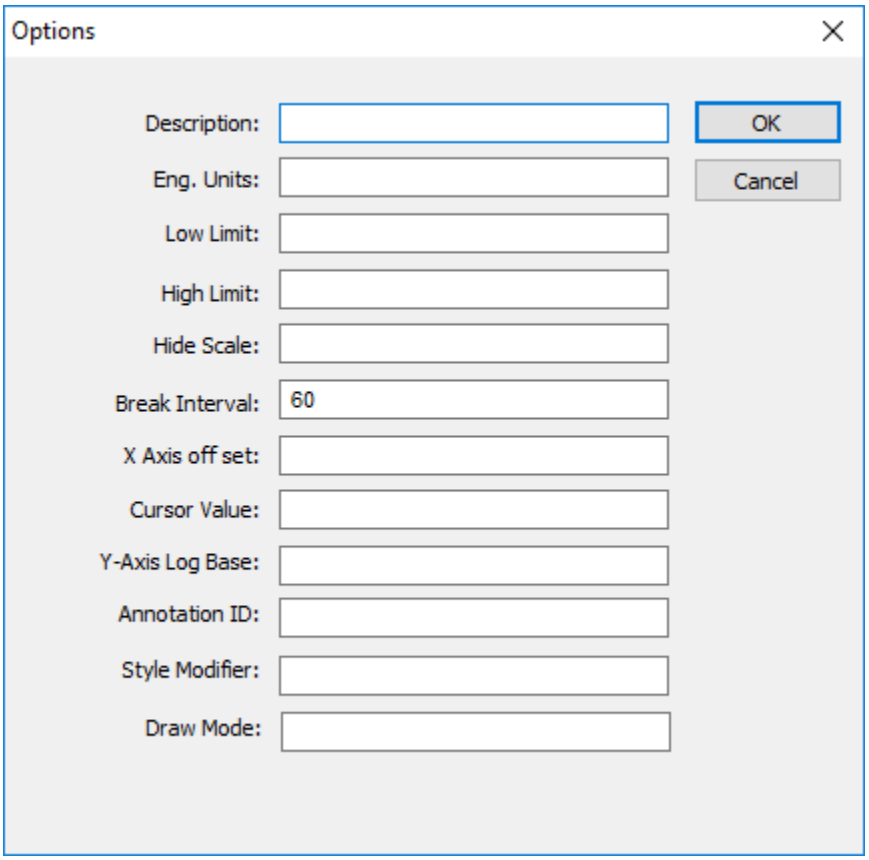
> **Note:** If the data source for the point is a tag from a Trend worksheet that has been configured to save to a Historian database or AVEVA Insight, decimation must be enabled.

### *Modify the pen style of a point during run time*

Use **Style Modifier** to modify the pen style of a point in a trend control.

By default, the data points in a trend control are drawn with a solid, black line. You can change the style of the line — more specifically, you can change the style of the pen that draws the line — by changing the settings in the *Pen Style* dialog box. The user can also open the *Pen Style* dialog box during run time and change the settings then. For more information, see Pen Style dialog box on page 429

Alternatively, you can use **Style Modifier** (in the *Options* dialog box) to programmatically modify some of the pen style settings during run time. In other words, you can specify project tags that will determine the pen style settings, and then you can use scripts or user input to change the tag values during run time.




***Options dialog box***

The **Style Modifier** box accepts a text string that includes one or more parameters, and each parameter modifies one element of the pen style. The text string must have this basic format:

```
<Parameter1>=<Value1>;<Parameter2>=<Value2>;…;<ParameterN>=<ValueN>
```

The following table lists the supported parameters and their accepted values:

| Parameter | Description | Accepted Values |
|---|---|---|
| `Expansion` | The method or algorithm used to connect the data points. | • `0` (smooth/analog)<br>• `1` (squared/digital) |
| `Type` | The type of line (e.g., solid, dashed, dotted) that connects the data points. | • `0` (solid)<br>• `1` (dashed)<br>• `2` (dotted)<br>• `3` (dash-dot)<br>• `4` (dash-dot-dot) |
| `Weight` | The weight (i.e., thickness) of the line, in pixels. | from `0` to `10` |
| `Color` | The color of the line.<br>For more information about colors in EcoStruxure Machine SCADA Expert, see Color Interface on page 74. | a 24-bit color value from `0` to `16777215` |

> **Note:** These parameters are the same as the settings in the *Pen Style* dialog box.

You could specify literal values for any or all of the parameters, but that is effectively the same as using the *Pen Style* dialog box to configure those settings. The key to programmatically modifying the pen style is specifying tag names or expressions enclosed in curly brackets (**{}**) for the parameter values. Then, whenever the value of a specified tag/expression changes, the pen style is modified.

For example, in the **Style Modifier** box, type the following string:

```
Expansion={ExpansionTag};Color={RGBColor(0,0,ColorTag)}
```

For the parameter **Expansion**, you specified the project tag **ExpansionTag** enclosed in curly brackets. There are only two accepted values (0 and 1) for **Expansion**, so **ExpansionTag** could be Boolean type. Then, whenever the value of **ExpansionTag** changes during run time, the expansion method is modified accordingly.

For the parameter **Color**, you specified an expression that calls the function `RGBColor` to convert RGB color values to a 24-bit color value. The red and green color values remain constant at 0, but the blue color value is determined by the project tag **ColorTag**. Then, whenever the value of **ColorTag** changes during run time, the value returned by the function also changes and the line color is modified accordingly.

Following this example, you can specify any project tags or expressions for the parameters as long as their returned values are within the accepted values for the parameters.

## SPC

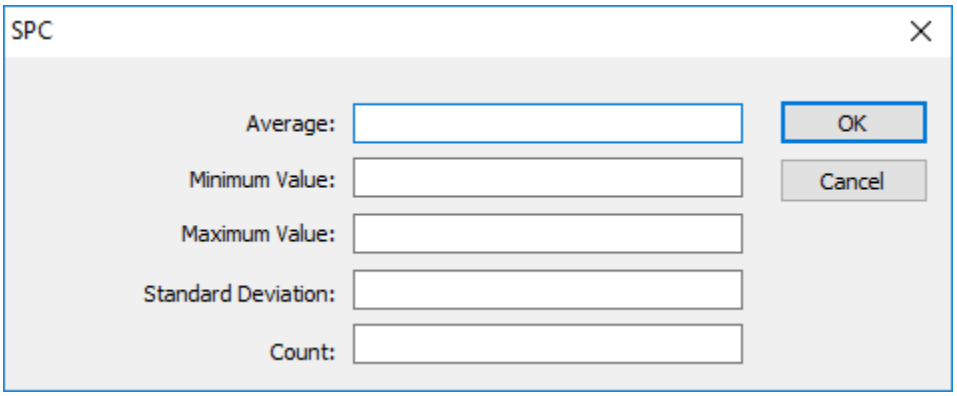
Use the *SPC* dialog box to specify project tags that will receive certain statistical values that are calculated from the entire history of a trend. These statistics are used in statistical process control (SPC), which is a method for monitoring processes and ensuring that they operate efficiently.

### Accessing the dialog box

To access the *SPC* dialog box for a specific point: first access the *Points* dialog box, where all of the points in a Trend Control object are configured, and then click the **SPC** column for that point. For more information, see Trend Control object on page 424.

> **Note:** If the data source for the point is a tag in a Trend worksheet that has been configured to save to a Historian database or AVEVA Insight, SPC is not supported.

### The dialog box in detail




***SPC dialog box***

The *SPC* dialog box includes the following settings:

**Average**

Type the name of a project tag (Real type) that will receive the calculated average of all of the data point's historical values.

> **Note:** When a value is not a number (NaN) — for example, when a tag is flagged as BAD quality — it is counted as 0 for the purpose of calculating the average.

**Minimum Value**

Type the name of a project tag (Real type) that will receive the minimum historical value of the data point.

**Maximum Value**

Type the name of a project tag (Real type) that will receive the maximum historical value of the data point.

**Standard Deviation**

Type the name of a project tag (Real type) that will receive the standard deviation of the data point. A low standard deviation indicates that the value of the data point tends to stay close to the average; a high standard deviation indicates that the value tends to vary greatly from the average.

**Count**

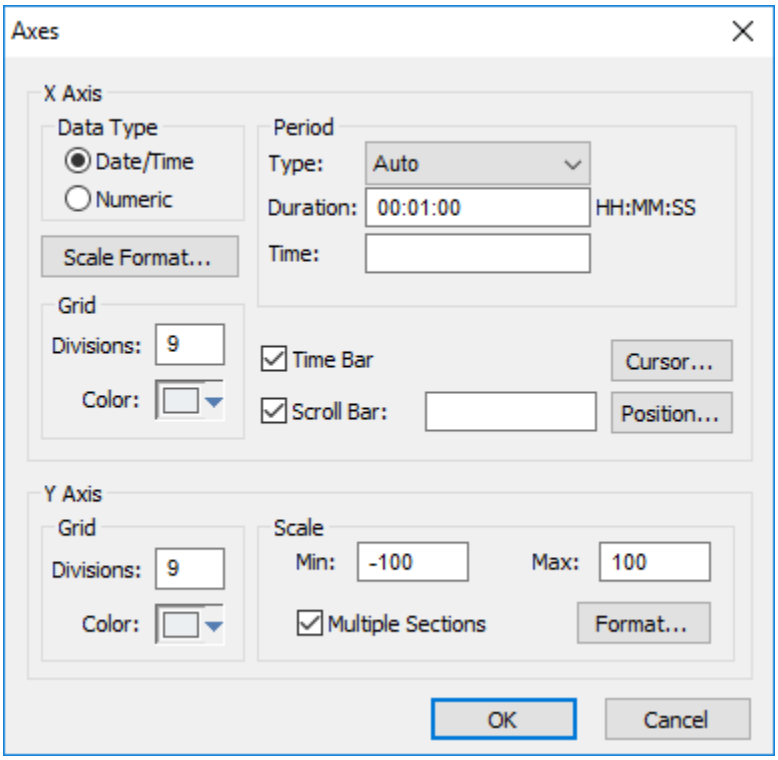
Type the name of a project tag (Integer or Real type) that will receive the total number of historical values, or samples, for the data point. The count will increase as the project runs and the historical database grows.

## *AXES* DIALOG

### Accessing the dialog

To access the *Axes* dialog for a specific Trend Control screen object, first access the *Object Properties* dialog for that screen object and then click **Axes**.

### The dialog in detail




***Axes dialog***

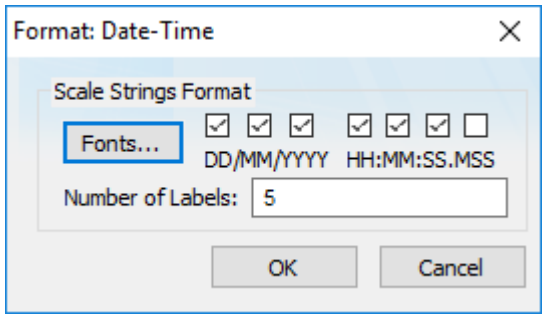
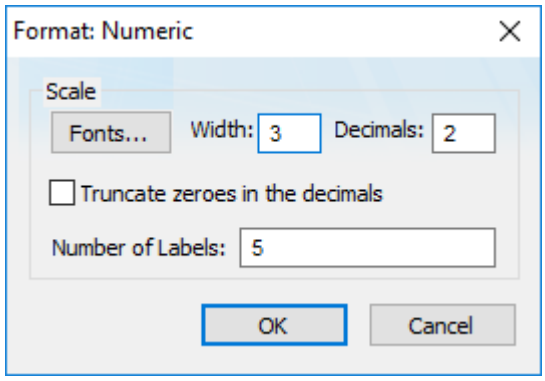
The *Axes* dialog contains the following elements:

| Group / Setting | | | Description |
|---|---|---|---|
| **X Axis** | **Data Type** | **Date/Time** | |
| | | **Numeric** | |
| | | **Scale Format** | |
| | **Period** (when Data Type is Date/Time) | **Type** | • **Auto**: When this option is selected, the Trend Control object works with **Start Date/Time** when is it triggered to Pause Mode, and it works with **Time Before Now** when it is triggered to Play Mode.<br>• **Start Date/Time**: When this option is selected, the value of the tag configured in the **Time** field defines the starting Date/Time for the data displayed on the object.<br>• **Time Before Now**: When this option is selected, the value of the tag configured in the **Time** field defines the amount of time before the current Date/Time, which will be used as the starting Date/Time for the data displayed on the object. |
| | | **Duration** | Defines the Period of data displayed on the object. You can configure a string tag in this field, so you can change the duration dynamically during runtime by changing the value of this tag. The format of the value supported by this property is HH:MM:SS. For example, `36:00:00` (thirty six hours). |
| | | **Time** | This field is optional. The value of the tag configured in this field represents a period of time, rather than a specific date or time. The meaning of this value depends on the option set for the **Type** property.<br>• When the **Type** is set as Start Date/Time, the value of the tag configured in this field must comply with the format Date Time. For example, `02/10/2005 18:30:00`. |

| Group / Setting | | | Description |
|---|---|---|---|
| | | | • When the **Type** is set as Time Before Now, the value of the tag configured in this field must comply with one of the following formats:<br>1. Time (string value). For example, **48:00:00** (forty eight hours).<br>2. Number of hours (real value). For example, **2.5** (two hours and thirty minutes).<br>If the **Time** field is left blank (or if the tag configured in this field has the value **0**), the object displays data up to the current Date/Time. |
| | **Period** (when Data Type is Numeric) | **Min** / **Max** | Minimum and maximum values displayed on the X axis.<br>The **Min** and **Max** properties can hold real numeric values up to six decimal places. If you need more precision than that, then you must configure the **Min** and **Max** properties with Real tags and then store the values in those tags. |
| | | **Eng. Units** | Engineering Unit (e.g., Kg, BTU, psi) that is associated with the X axis during runtime. |
| | **Grid** | **Divisions** | You can configure the number of divisions (vertical or horizontal lines) drawn on the object for the X and/or Y axis respectively, as well as the color of these lines. |
| | | **Color** | The color of the vertical grid lines. |
| | | **Time Bar** | When checked, the Time bar is displayed below the X axis during runtime; otherwise, it is hidden. The time bar is a standard interface that can be used by the operator to change the X axis scale during runtime. |
| | | **Scroll Bar** | When checked, the Scroll bar is displayed below the X axis during runtime; otherwise, it is hidden. The time bar is a standard interface that can be used by the operator to navigate through the X axis scale during runtime. Optionally, you can configure a tag in the Scroll bar field, which defines the period for the scroll bar. If this field is left empty, the period is equal to the current value for Duration of the X axis. |
| | | **Cursor** | The cursor is an optional ruler orthogonal to the X axis, which can be used during runtime to obtain the value of any pen at a specific point (intersection of the pen with the cursor). When you click this button, the Cursor dialog launches, where you can configure the settings for the optional vertical cursor as follows: |
| | | **Positon** | Defines the position of the X axis, as well as its direction and orientation, as follows: |
| **Y Axis** | **Grid** | **Divisions** | You can configure the number of divisions (vertical or horizontal lines) drawn on the object for the X and/or Y axis respectively, as well as the color of these lines. |
| | | **Color** | The color of the horizontal grid lines. |
| | **Scale** | **Min** / **Max** | Default minimum and maximum values displayed in the Y axis. Used when more than one pen shares the same scale (Multiple Sections disabled), and/or for the points whose Min and Max fields are not configured (left blank).<br>Please note that if you configure a trend point to have a logarithmic scale (see **Points - Options**), then the value configured here for **Min** should be greater than 0. Even if you configure a minimum value less than or equal to 0 (which is impossible for a logarithm), a minimum value of 0.00000000000000000001 will be used automatically during run time. This will not change any of the object properties that you have configured. |
| | | **Multiple Selections** | When checked, the Y scale is divided automatically into one section for each pen; otherwise, all pens share the same Y scale. |
| | | **Format** | Launches a dialog for configuring the format of the labels displayed by the Y axis. |

**Note:** The tags configured in the **Period/Range** fields are automatically updated when the user changes the X scale dynamically during runtime, using the Time bar embedded in the object.

- **Data Type**: The X axis can display either Date/Time values or numeric values, according to this setting.

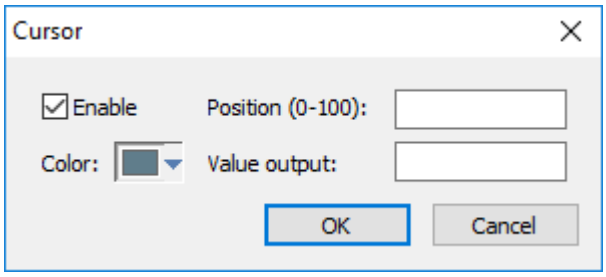
| Data Type | Scale Format |
|---|---|
| Date/Time |  |
| Numeric |  |





> **Note:** The number of decimal points for the X or Y scale (Decimals) can be configured with a tag. Therefore, this setting can be modified dynamically during runtime.

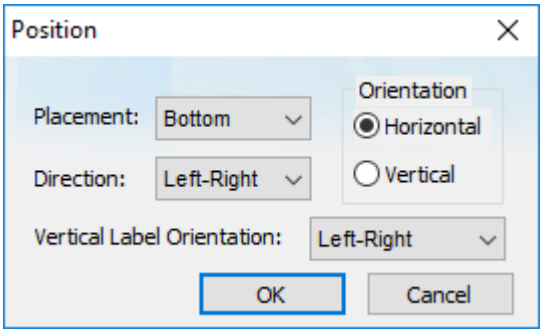
- **Cursor**: The cursor is an optional ruler orthogonal to the X axis, which can be used during runtime to obtain the value of any pen at a specific point (intersection of the pen with the cursor). When you click this button, the Cursor dialog launches, where you can configure the settings for the optional vertical cursor as follows:




***Cursor Dialog***

| Property | Description |
|---|---|
| **Enable** | When checked, the vertical cursor is visible during runtime. |
| **Color** | Color of the line drawn for the cursor. |
| **Position (0#100)** | You can configure a numeric tag in this field, which is proportional to the position of the cursor on the X axis, from 0 to 100%. When this value is changed, the position of the cursor is automatically modified. |
| **Value Output** | You can configure a string tag in this field that returns the value of the X axis in which the cursor is currently positioned. |

- **Position**: Defines the position of the X axis, as well as its direction and orientation, as follows:




***Position Dialog***

| Property | Description |
|---|---|
| **Placement** | Side of the trend control on which the X axis will be placed. |
| **Direction** | Direction of the X axis. |
| **Orientation** | Orientation of the X axis. |
| **Vertical Label Orientation** | The orientation of the text labels on the vertical axis, regardless of whether the vertical axis is X or Y. |

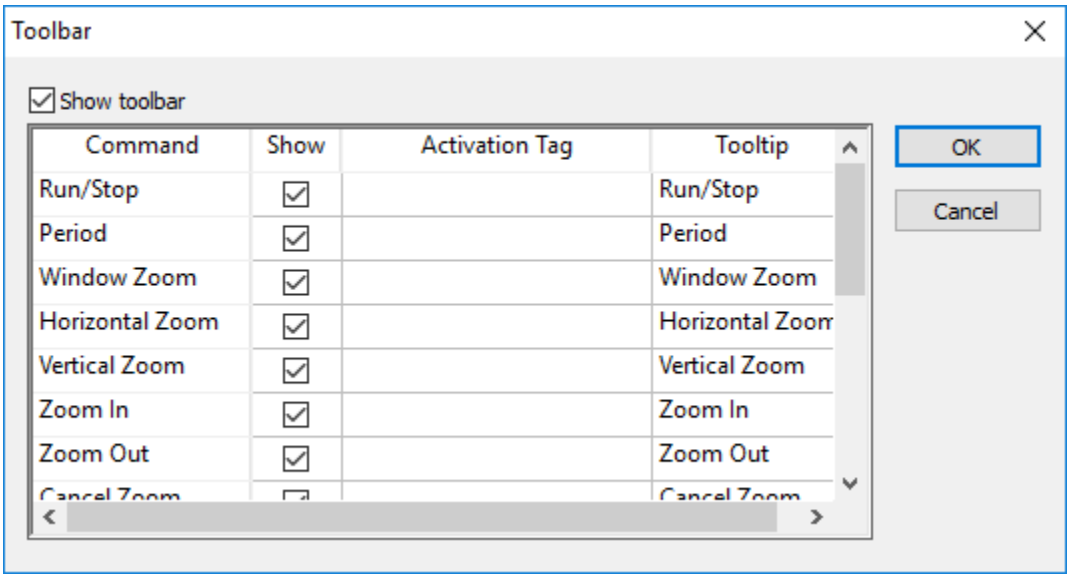
## *TOOLBAR* DIALOG

The *Toolbar* dialog is used to customize the toolbar on the Trend Control screen object.

### Accessing the dialog

To access the *Toolbar* dialog for a specific Trend Control screen object, first access the *Object Properties* dialog for that screen object and then click **Toolbar**.

### The dialog in detail




***Toolbar dialog***

The **Show toolbar** option controls whether the entire toolbar is shown during runtime. You may hide the toolbar to save space or to prevent users from changing the trend display.

Also, each command/tool in the toolbar has the following properties:

| Column Name | Description |
|---|---|
| **Command** | The name of the command/tool. For more information about each tool, see |
| **Show** | The option to show the tool on the toolbar. |
| **Activation Tag** | An optional tag trigger — when the value of the tag changes from FALSE (0) to TRUE (any non-zero value), the command is activated as if the operator clicked the tool.<br>This can be used to script changes in the trend display during runtime. |
| **Tooltip** | The tooltip that is displayed when the mouse cursor hovers over the tool. |

For more information, see About the trend control runtime interface on page 424.

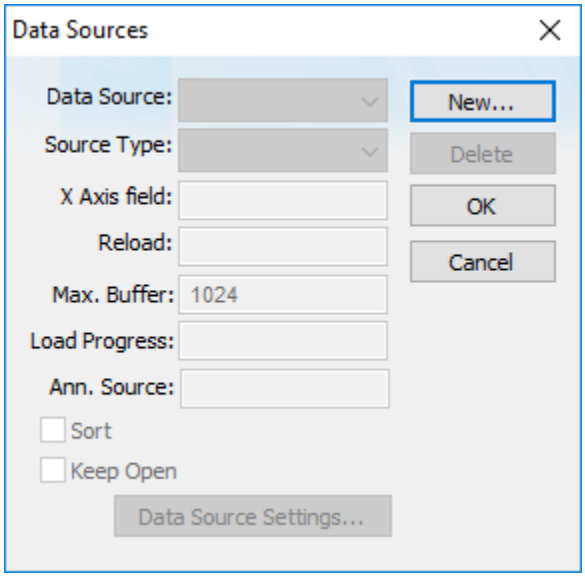
## DATA SOURCES

Use the *Data Sources* dialog box to configure one or more data sources for a Trend Control object.

### Accessing the dialog box

To access the *Data Sources* dialog box for a specific Trend Control object, first access the *Object Properties* dialog for that object and then click **Data Sources**.

### The dialog box in detail




***Data Sources dialog***

The data source defines the location of the values from the data point(s) associated with it. Many points can share the same data source — you do not need to create one data source for each data point.

The data source tag is available by default to the Trend Control object. You can add additional data sources with the **New** button. The name you enter will be used as an alias to link the data points to this new data source.

For more information about adding data sources, see:

- Using the Data Source Text File on page 449
- Using the Data Source Database on page 452

The other fields in this dialog allow you to edit the data source settings:

**Source Type**

Select the source type of the location of the data point values. For more information about the different types of sources, see the table below.

**X Axis field**

If the X axis of the trend graph is set to be numeric instead of date/time (in the Axes settings), then enter the name of the field (column) of the data source that holds the data for the X axis.

> **Note:** If you have enabled decimation (either in the Advanced settings for all trends or in the Options settings for a single trend), then the field that you have specified here must be sorted in ascending order. The procedure to do this varies by source type (e.g., text editor, spreadsheet application, external database, etc.), so for more information, see the documentation for your specific type.

**Max. Buffer**

The maximum amount of data (in bytes) that will be held in runtime memory.

**Load Progress**

The tag in this field will receive a real value (0-100) that represents the percentage of the Data Source load progress.

**Ann. Source**

The name of the database table that contains text and image annotations to be displayed in the trend control. This must be a table in the same database that is configured to be the trend control's data source. Annotations are not supported for other types of data sources. For more information, see Display text- and image-based trend annotations in a trend control on page 459.

**Sort**

This option is useful for plotting data from a text file. When enabled (checked), it sorts the data and shows the Cursor column value until the **Max. Buffer** is filled. When disabled (unchecked), the data are not sorted and the Cursor column value is not shown.

**Keep Open**

This option keeps the data source open as long as the screen that contains the Trend Control object is open. This improves the performance of the runtime project, but keeping the data source open may cause other problems like database connection errors (when **Source Type** is **Database**) and file write conflicts (when **Source Type** is **Batch** or **Text File**). To close the data source after the data has been loaded, clear (uncheck) this option.

**Data Source Settings**

Click to define the settings for the selected **Source Type**

The following table summarizes the settings for each **Source Type**:

| Source Type | Description | X-Axis field | Data Source Settings |
|---|---|---|---|
| **Batch** | Batch generated by the **Trend** task of EcoStruxure Machine SCADA Expert | Disabled. The X-Axis data will be retrieved automatically on the correct position from the proprietary Batch file generated by EcoStruxure Machine SCADA Expert. | Enter the data point values in **Batch Name** for their retrieval. You can configure a tag between curly brackets in this field to change this setting dynamically during runtime. |
| **Database** | SQL Relational Database | The name of the field that contains the X-axis data. | Configure the settings to link this Data Source to the SQL Relational Database that holds the data point values. For more information, see Using the Data Source Database on page 452. |

| Source Type | Description | X-Axis field | Data Source Settings |
|---|---|---|---|
| **Text File** | Text file (e.g., CSV file) with data point values separated by a specific delimiter | Number of the column that holds the X-Axis data. The number 0 refers to the first column, 1 refers to the second column, and so on. | Enter the name of the text file that holds the data points. The default path is the current project folder. You can configure a tag between curly brackets in this field to change this setting dynamically during runtime.<br><br>You can also choose one or more delimiters for the data stored in the text file. The value of each row is written in the text file between two delimiters. When using a comma as a delimiter, the grid object is able to read data from CSV files. You can even choose a custom delimiter by checking the **Other** option. For more information, see Using the Data Source Text File on page 449. |

**Note:**

There is a default query timeout of 120 seconds, to prevent the project client from hanging on an unusually long data source query. To adjust the timeout period, manually edit your project file (`<project name>.APP`) and change the following setting:

```
[Trend]
QueryTimeout=120
```

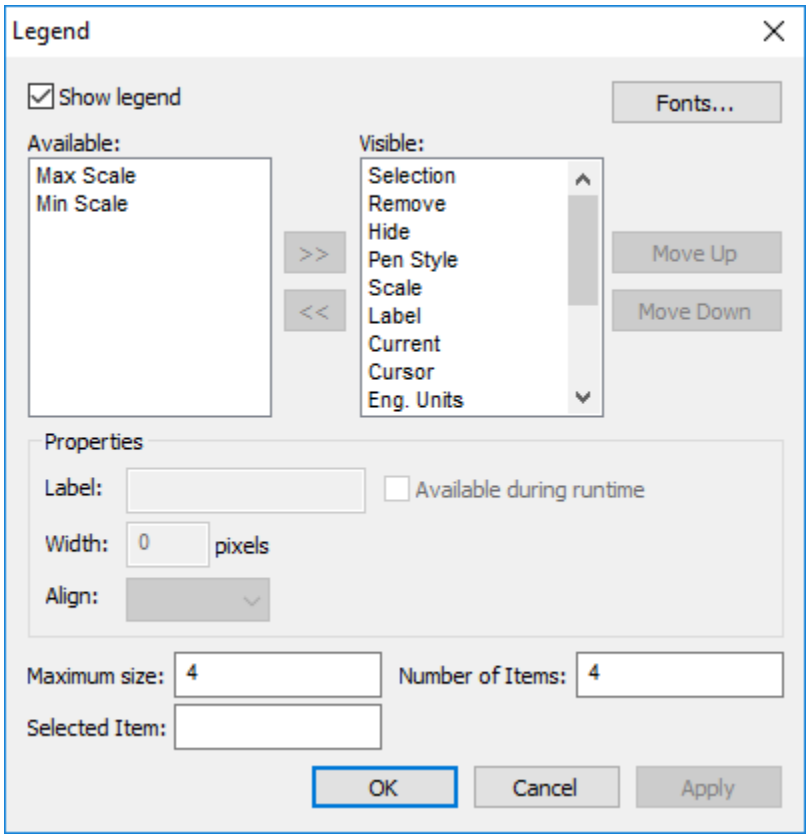
If you change the value to `0`, then there will be no timeout at all; data source queries will always continue until they are completed.

## *LEGEND* DIALOG

### Accessing the dialog

To access the *Legend* dialog for a specific Trend Control screen object, first access the *Object Properties* dialog for that screen object and then click **Legend**.

### The dialog in detail




***Legend dialog***

The *Legend* dialog contains the following elements:

- **Show**: When checked, the embedded legend is displayed during runtime. This interface provides useful information associated with the pens currently linked to the object.
- **Available / Visible**: The items in the **Visible** box are displayed in the legend during runtime. You can add items to and remove them from the **Visible** box using the **»** and **«** buttons respectively. Moreover, you can use the **Move Up** and **Move Down** buttons to change the order in which the items are displayed in the legend during runtime.

  The following table lists the available legend items:

| Item | Legend Icon | Description |
|---|---|---|
| Eng Units | | The tag/pen's Engineering Units. |
| Min | | The tag/pen's minimum possible value. |
| Max | | The tag/pen's maximum possible value. |
| Selection | | Press button to select another tag for this pen. |
| Remove | | Press button to completely remove this pen from the legend and the Trend chart. |
| Hide | | Select (check) option to hide this pen in the Trend chart. |
| Pen Style | | Press button to change the pen's line style, weight, color, markers, and so on. |

| Item | Legend Icon | Description |
|---|---|---|
| Scale | | Select (check) option to show the pen's scale on the Trend chart. |
| Description | | Description of the tag/pen. |
| Current | | The current value of the tag configured to the pen. |
| Cursor | | The value of the pen where it intersects the cursor line. |

- **Properties**: Allows you to configure the properties for the field highlighted in the **Available** or **Visible** box:

| Property | Description |
|---|---|
| **Label** | Label for the field displayed during runtime |
| **Width** | Width for the field (in pixels) during runtime. |
| **Align** | Alignment of the data displayed in the field. |
| **Available during runtime** | When this option is checked, the user can show or hide the field during runtime. |

- **Maximum size**: Defines the size of the legend in terms of number of rows. For instance, the user might have 8 points being displayed in the trend object, if the maximum size is set to two, the legend will have a scroll bar to allow the user to scroll to the other points.
- **Number of items**: Number of points (default) displayed on the legend. You can allow the user to add/remove points during runtime regardless of the value in this field.
- **Selected Item**: You can configure a numeric tag in this field. The object writes in this tag the number of the selected row. In addition, you can select different rows by writing their values in this tag.
- **Fonts**: Sets the font for the text displayed in the legend.

For more information, see About the trend control runtime interface on page 424.

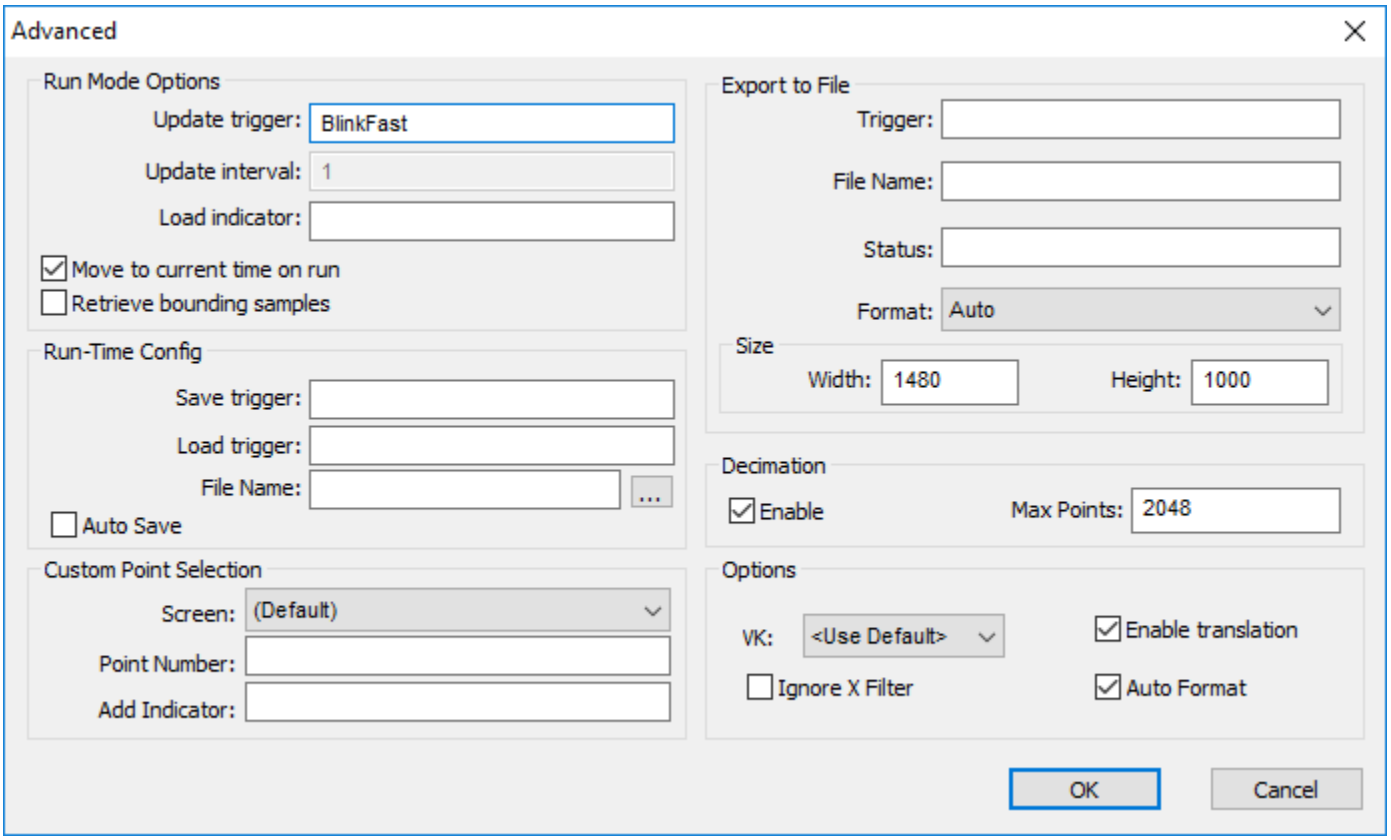
## ADVANCED

Use the *Advanced* dialog box to configure advanced settings for a Trend Control object.

### Accessing the dialog box

To access the *Advanced* dialog box for a Trend Control object, first access the object properties for that object and then click **Advanced**. For more information, see Trend Control object on page 424.

### The dialog box in detail




***Advanced dialog box***

The *Advanced* dialog box includes the following settings:

| Group | Setting | Description |
|---|---|---|
| **Run Mode Options** | **Update trigger** | When the tag configured in this field changes value, the trend object is updated (refreshed). |
| | **Update interval** | When the update trigger is issued and the X Axis if of type numeric, the value on this field will be added to the minimum and maximum values of the X Axis. |
| | **Load indicator** | Type the name of a project tag. While the trend control is loading external data, the tag receives a value of 1, and when the trend control has finished loading the data, the tag receives a value of 0. |
| | **Move to current time on run** | When this box is checked, X axis shifts to the current time automatically when the object is triggered to Play mode, during runtime. |
| | **Retrieve bounding samples** | When this box is checked, the object retrieve the data outbound the object (first points only). Uncheck this option can improve the performance, since the points outbound the object will not be retrieved from the history. On the other hand, the object will not draw lines linking the first and last samples to the extremities of the object. |
| **Run-Time Config** | **Save trigger** | The configuration of a Trend Control object can be changed during run time, using the object's on-screen tools. You can then save the new configuration and load it again at a later time. This allows you to do things like:<br>• Keep the configuration consistent when the user closes and then reopens a project screen, even after restarting the project; or<br>• Create standard configurations for different situations and then load the appropriate configuration during run time, based on a predefined condition or the user's selection. |

| Group | Setting | Description |
|---|---|---|
| | | **Note:** This feature is not available on Mobile Access.<br><br>When the tag configured in the **Save trigger** box is toggled (i.e., changes value), the current configuration of the Trend Control object is saved to an external file (as specified in the **File Name** box). The following settings are saved:<br>• Points: Color, Tag/Field (in case you are using indirect tags), Min Scale, Max Scale, Hide<br>• Axes: Period (start time, end time, etc.), Grid Divisions, Number of Labels, Enable Cursor, and the current zoom<br>• Toolbar: Show/hide state of all commands that do not have activation tags configured<br>• Legend: Visible columns with widths, Maximum Size, Number of Items, Selected Item |
| | **Load trigger** | When the tag configured in the **Load trigger** box is toggled (i.e., changes value), the previously saved configuration will be loaded from the external file (as specified in the **File Name** box) and then applied to the Trend Control object. |
| | **File Name** | The name of the external file that will be saved and loaded. If you do not specify a file extension, the default extension is .stmp. If you do not specify any file name at all (i.e., if you leave this box empty), the default name is:<br>`<screen name><object ID>TrendControl.stmp`<br>For the local Viewer running on the project runtime server, the file is saved in the project's Web folder. For example:<br>`<project name>\Web\MyScreen10TrendControl.stmp`<br>For Thin Clients running on all other stations, the file is saved in the standard Temp directory. For example:<br>`C:\Users\<current user>\AppData\Local\Temp\MyScreen10TrendControl.stmp`<br>To change the file name during run time, type an appropriate tag/expression enclosed in curly brackets (e.g., `{UserName}TrendControl`). The current value of the tag/expression will be used whenever **Save trigger** or **Load trigger** is activated.<br>**Note:** If you specify a file extension other than .stmp, the resulting save files will not be recognized or converted when you upgrade to a newer version of EcoStruxure Machine SCADA Expert 2020. To ensure the files are converted, manually change the extensions on all of the files to .stmp before you upgrade. Then, after you upgrade the software and your project, you may change the extensions back to what they were before. |
| | **Auto Save** | If this option is selected, the current configuration of the Trend Control object is automatically saved when the project screen is closed. If this option is cleared, the configuration is saved only when **Save trigger** is activated. |
| **Custom Point Selection** | **Screen** | This interface allows you to create your custom dialog to modify or insert pens to the object.<br>Name of the screen which must be launched when the user triggers a command to modify or insert a new pen to the object during runtime. |
| | **Point Number** | Point number (from the Points dialog), indicating the point associated to the pen that will be inserted or modified during runtime. |
| | **Add Indicator** | Flag that indicates that the user triggered an action to insert a new pen (value 1) instead of modifying a pen that is already been visualized (value 0). |
| **Export to File** | **Trigger** | When the project tag specified in this box changes value (i.e., toggles), the current state of the trend control is exported to an image file. In other words, a screen shot is taken, but only of the trend control. The toolbar, scroll bar, legend and time display are not included. |
| | **File Name** | The file path and name of the exported file. |

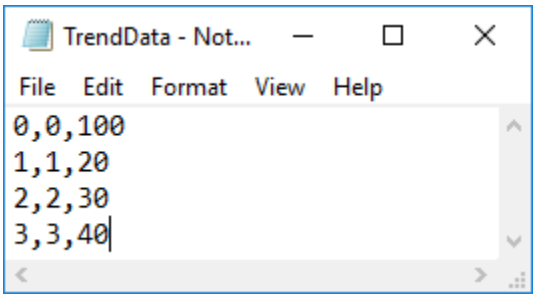
<table>
<tr><th>Group</th><th>Setting</th><th>Description</th></tr>
<tr><td></td><td></td><td>If no path is specified, the file is saved in the Web sub-folder of the project folder. If no extension is specified, it is determined by Format (see below).<br><br>To programmatically change the file name during run time, specify a project tag or expression enclosed in curly brackets (e.g., <code>{MyFileName}</code>). The value of the specified tag/expression is used.</td></tr>
<tr><td></td><td>Status</td><td>The project tag specified in this box receives a status code that indicates the success or failure of the most recent export.<br><br>The status code can be one of the following possible values:<br><br>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>-2</td><td>Out of memory. The specified image size is too large. See Size below.</td></tr>
<tr><td>-1</td><td>Error during export. Either the specified image size is invalid (e.g., 0) or the file could not be saved.</td></tr>
<tr><td>0</td><td>Export has started.</td></tr>
<tr><td>1</td><td>Image file exported successfully.</td></tr>
</table>
</td></tr>
<tr><td></td><td>Format</td><td>The graphic format of the exported image file.<br><br>If you select Auto, the format is determined by the file extension specified in File Name (see above). If you select Auto but do not specify a file extension, the default format is BMP.</td></tr>
<tr><td></td><td>Size</td><td>The image file is exported at full size by default. However, you can specify the Width and Height (in pixels).</td></tr>
<tr><td>Decimation</td><td>Enable</td><td>When this option is selected, the trends in the Trend Control object that are configured to show historical data will have their data decimated before the trends are drawn. This means that for each trend, the X axis is divided into a number of intervals (determined by Max Points) and all of the data points within each interval are averaged together to be drawn as a single point. This can improve runtime performance when there is a large amount of historical data to display, and it can make the trends easier to read.<br><br>Decimation only works when the trend control is in Stop Mode (a.k.a. Historical Mode).<br><br>Please note that when this option selected, decimation is done for all trends that are configured to show historical data. To do it for only for a single trend, use Draw Mode in the Points – Options settings.<br><br>
<table>
<tr><td> </td><td>Note: If decimation is enabled and the X axis is set to be numeric rather than date/time, then the data used in the X axis must be properly sorted. For more information, see the Data Sources settings.</td></tr>
</table>
<br>
<table>
<tr><td> </td><td>Note: If the data sources for one or more points are tags from a Trend worksheet that has been configured to save to a Historian database or AVEVA Insight, decimation must be enabled at least for those points.</td></tr>
</table>
</td></tr>
<tr><td></td><td>Max Points</td><td>The maximum number of data points used to draw each trend. Default is 2048.</td></tr>
<tr><td>Options</td><td>VK (Virtual Keyboard)</td><td>Virtual Keyboard type used for this object.</td></tr>
<tr><td></td><td>Ignore X Filter</td><td>When this box is checked, the X Filter is ignored to avoid adding the WHERE or querying clause to the Data Sources.</td></tr>
<tr><td></td><td>Enable translation</td><td>Enable the external translation for the text displayed by this object.</td></tr>
<tr><td></td><td>Auto Format</td><td>When checked, decimal values in the Current, Cursor, Max, Min and Scale columns will be formatted according to the virtual table created by the function <code>SetDecimalPoints</code>.<br><br>
<table>
<tr><td> </td><td>Note: For the Auto Format to work, decimals formatting on the X axis must be disabled — that is, the Decimals box in the Axes settings must be left empty.</td></tr>
</table>
</td></tr>
</table>

### *Using the Data Source Text File*

The Trend Control can generate trend charts from any Text File that has the values organized in columns and rows. The columns should be separated from each other by special characters (usually the comma). Each sample (pair of values representing a point in the graph) is represented by a row (a line in the file). Suppose that the user wants to display a chart with the information in the following table:

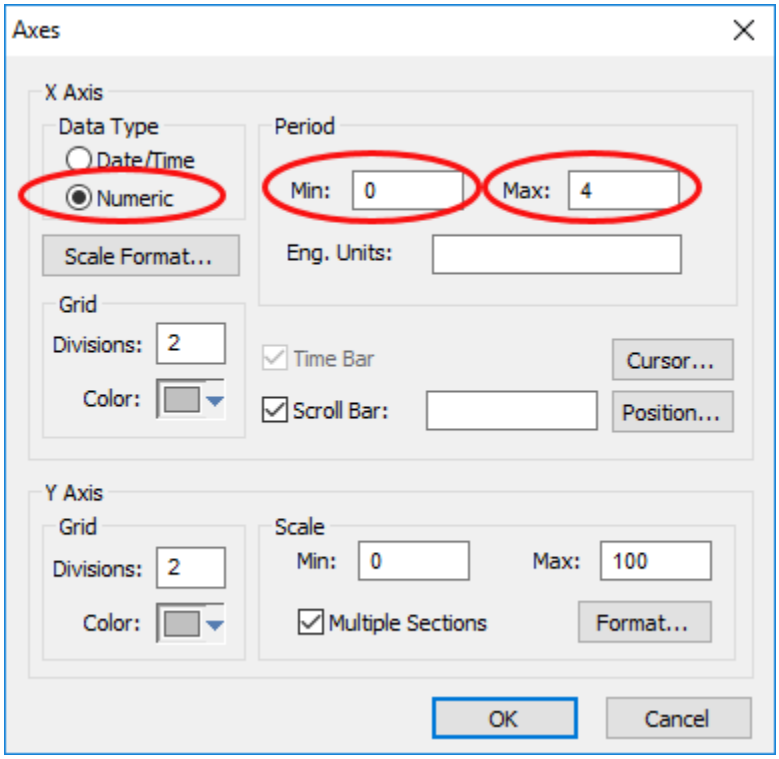
| X Value | Y1 Value | Y2 Value |
|---|---|---|
| 0 | 0 | 10 |
| 1 | 1 | 20 |
| 2 | 2 | 30 |
| 3 | 3 | 40 |

We have one variable that represents the X Axis and two variables (Y1 and Y2) that will represent different lines in the chart. The first step is to convert the data into a text file. If we adopt the comma as our separator the file will be as shown below:

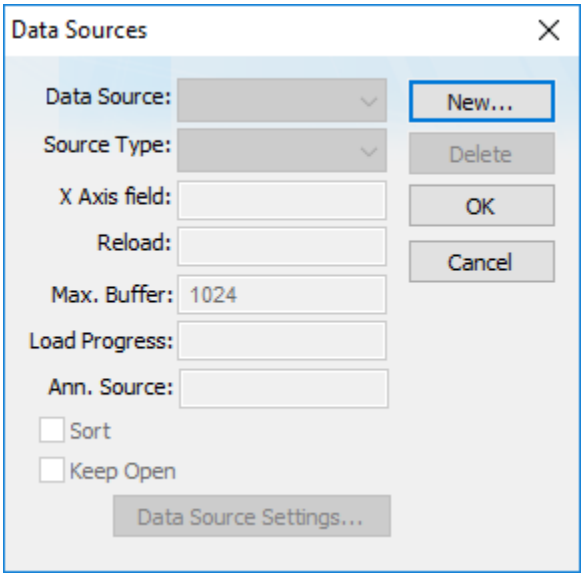



We strongly recommend that you save the file in the same folder where the project is. By doing so, you do not have to specify the entire path and your project will still work, even if it is copied to a different computer.

Once you have added the Trend Control to your screen, double-click on the object to open the *Object Properties* and click on **Axis**.... Change the *Data Type* of the *X Axis* to **Numeric**, and set the ranges as shown in the picture below:

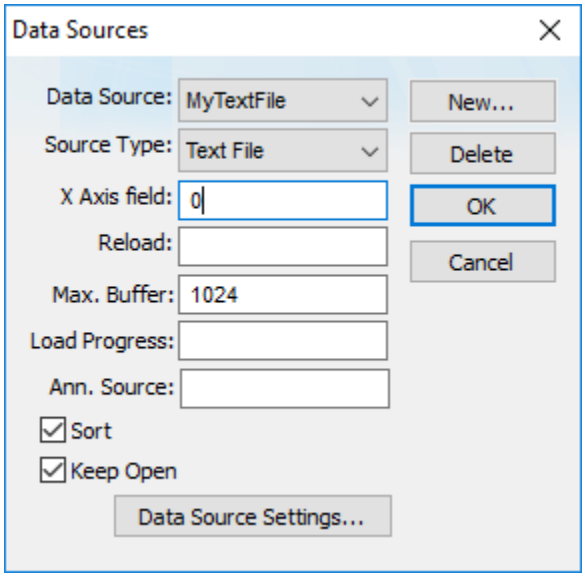



Click **OK** on this window and then, in the *Object Properties* window, click on the **Data Sources**... button. The following window will display:



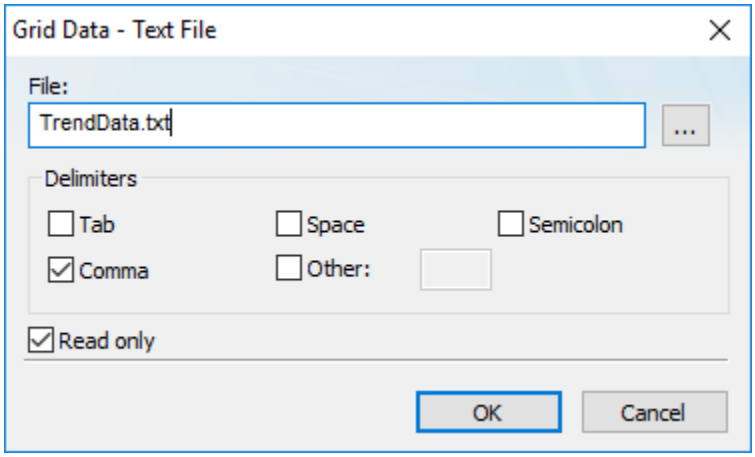

***Trend Control – Data Sources dialog***

We need to create a data source in order to access to the text file. Click on the **New**... button, specify the **Data Source** name **MyTextFile** and then click **Create**. You should see the following information now:




***Setting X Axis field to 0***

On the **X Axis field** we need to indicate which column in our text file represents the X Axis. In our example we are using column zero, so enter `0` for this field, then click **Data Source Settings**.... The following window will display:

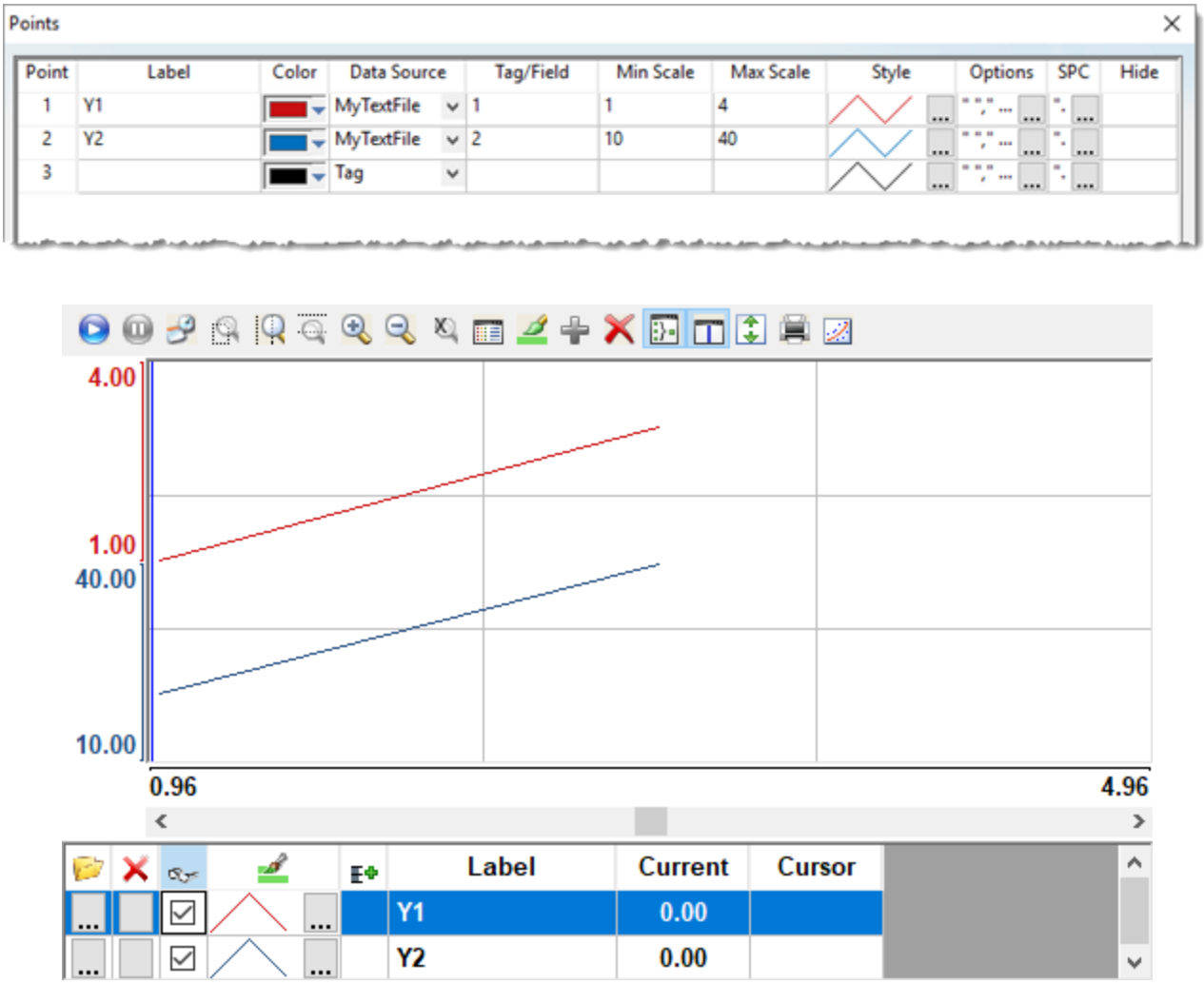



***Selecting the text file***

If you have copied the text file to the project folder, you only have to specify the file name, otherwise, enter with the complete path where the file is located (use the browse button as needed). Click **OK** on this window and **OK** again to finish the data source configuration and close the *Data Source* configuration Window.

Now we need to define our Y1 and our Y2. They will be represented by points on our Trend Control. Double-click on the Trend Control again to access the *Object Properties* window and then click on **Points**.... Your next step is to define the points according to the following figure:

After following these steps, run your project and you should see something similar to the figure below:




### *Using the Data Source Database*

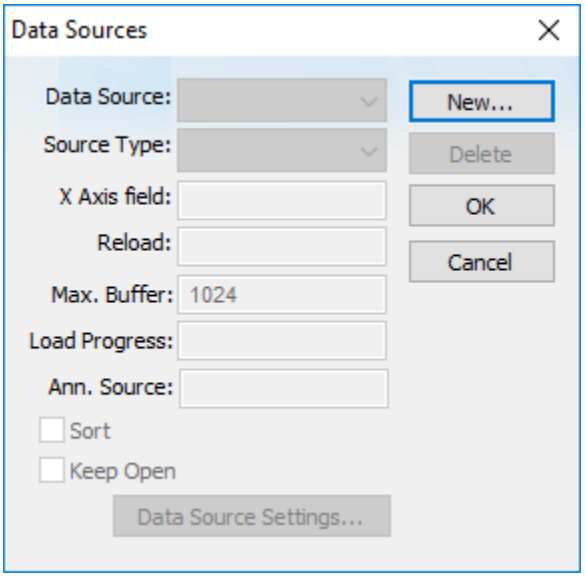
The Trend Control can generate trend charts from any Relational Database that can be accessed through the ADO.Net technology. This Appendix illustrates how to access a Microsoft Access Database; if you are using another type of database, almost all the definitions will apply, however you will need to configure your connection on a different way. For information on how to configure other databases, please refer to the Appendixes in the Database Interface section of this manual.

Suppose that you have an access database at your C drive named **mydata.mdb** and that you want to generate a chart based on the information in the following table:

TrendData

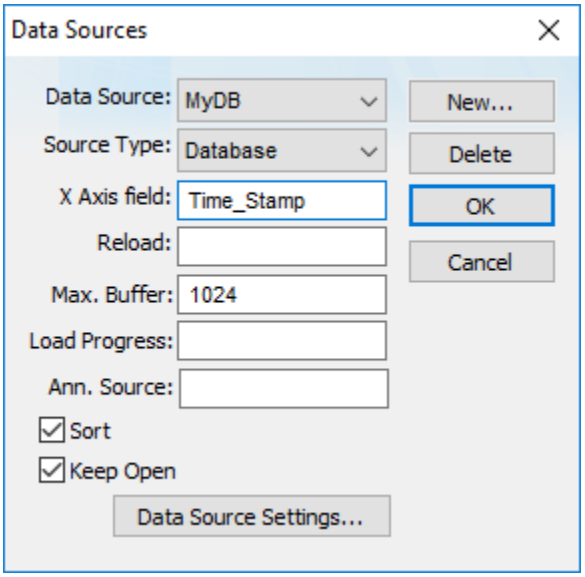
| Time_Stamp | Temperatur | Pressure |
|---|---|---|
| 7/17/2018 1:00:00 PM | 80 | 30 |
| 7/17/2018 1:00:01 PM | 40 | 31 |
| 7/17/2018 1:00:02 PM | 50 | 32 |
| 7/17/2018 1:00:03 PM | 60 | 30 |
| 7/17/2018 1:00:04 PM | 70 | 33 |
| 7/17/2018 1:00:05 PM | 90 | 34 |
| 7/17/2018 1:00:06 PM | 100 | 44 |
| 7/17/2018 1:00:07 PM | 101 | 44 |
| 7/17/2018 1:00:08 PM | 90 | 50 |
| 7/17/2018 1:00:09 PM | 95 | 44 |
| 7/17/2018 1:00:10 PM | 96 | 40 |
| 7/17/2018 1:00:11 PM | 99 | 40 |
| 7/17/2018 1:00:12 PM | 95 | 45 |
| 7/17/2018 1:00:13 PM | 90 | 56 |
| 7/17/2018 1:00:14 PM | 80 | 44 |
| 7/17/2018 1:00:15 PM | 75 | 46 |
| 7/17/2018 1:00:16 PM | 64 | 44 |
| 7/17/2018 1:00:17 PM | 55 | 48 |
| 7/17/2018 1:00:18 PM | 56 | 50 |
| 7/17/2018 1:00:19 PM | 54 | 51 |
| 7/17/2018 1:00:20 PM | 50 | 52 |

The first step is to add the Trend Control to your screen. Now double-click on the object to open then Object Properties and click on **Data Sources**.... The following window will display:



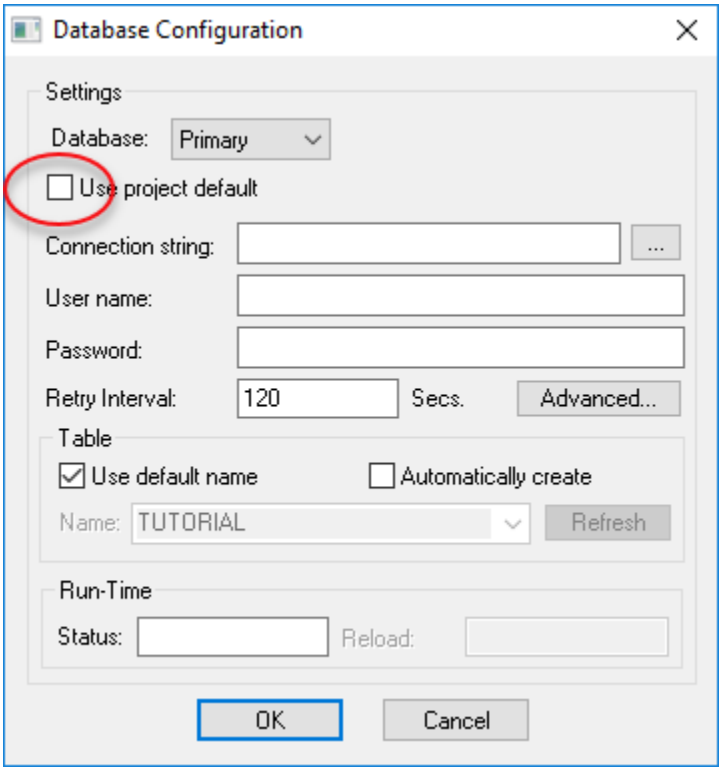

***Trend Control – Data Sources dialog***

We need to create a data source in order to access to the database. Click the **New**... button, specify the **Data Source** name **MyDB** and then click **Create**. You should see the following information now:



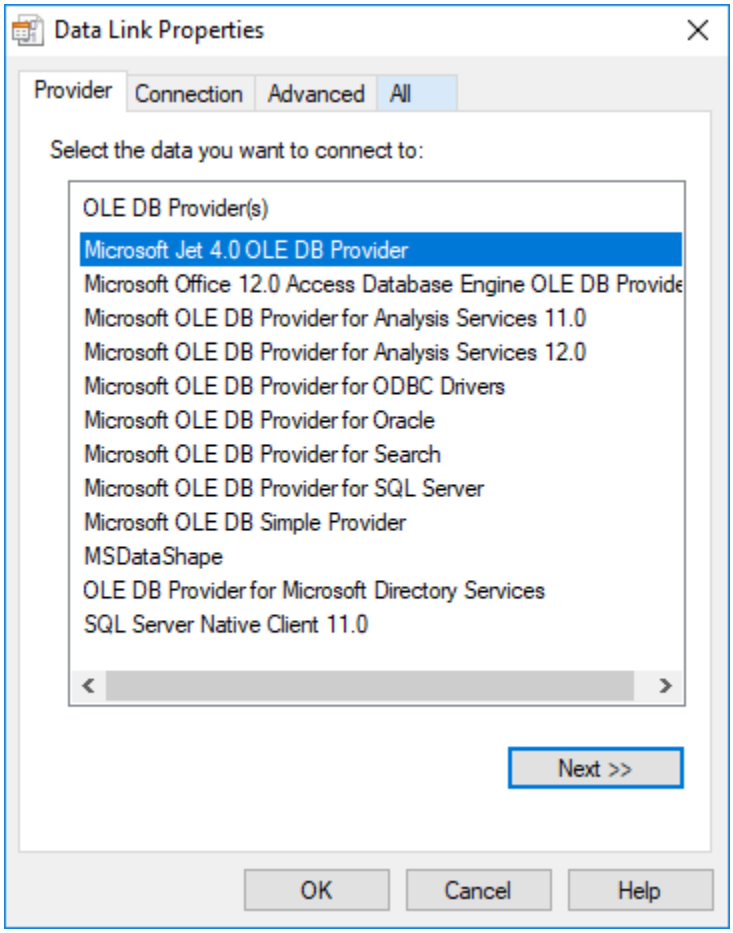

***Setting X Axis field to `Time_Stamp`***

Change the **Source Type** to **Database** and specify `Time_Stamp` in the **X Axis field**. Then click on the **Data Source Settings**... button, the following window will display:



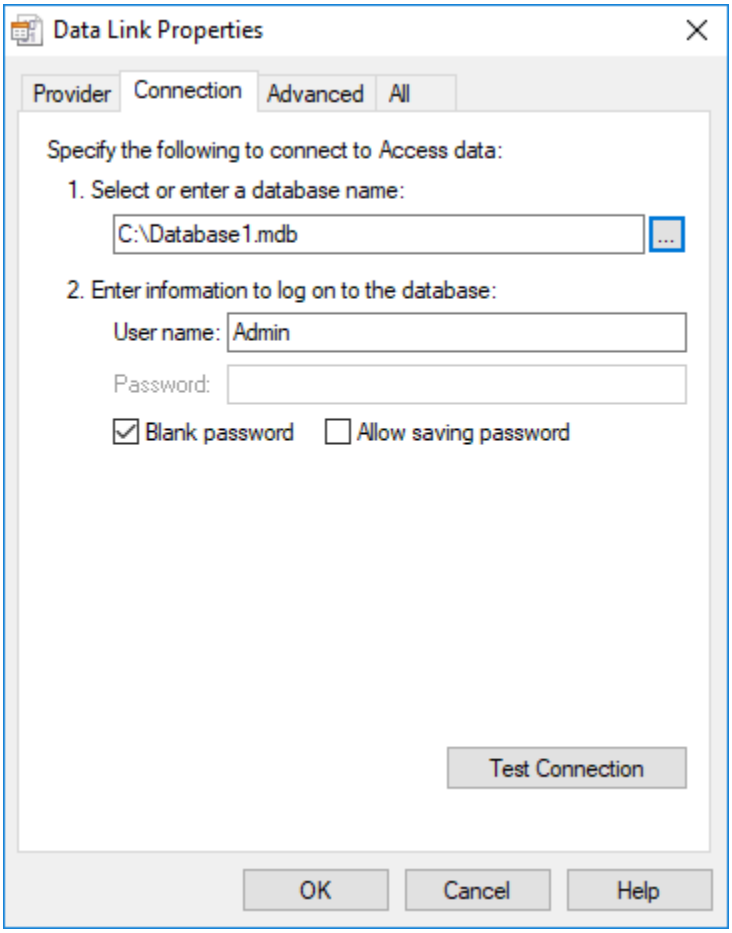

***Clearing the Use project default option***

Uncheck the checkbox **Use project default** and click on the browse button ... in order to configure the connection string. The following window will display:



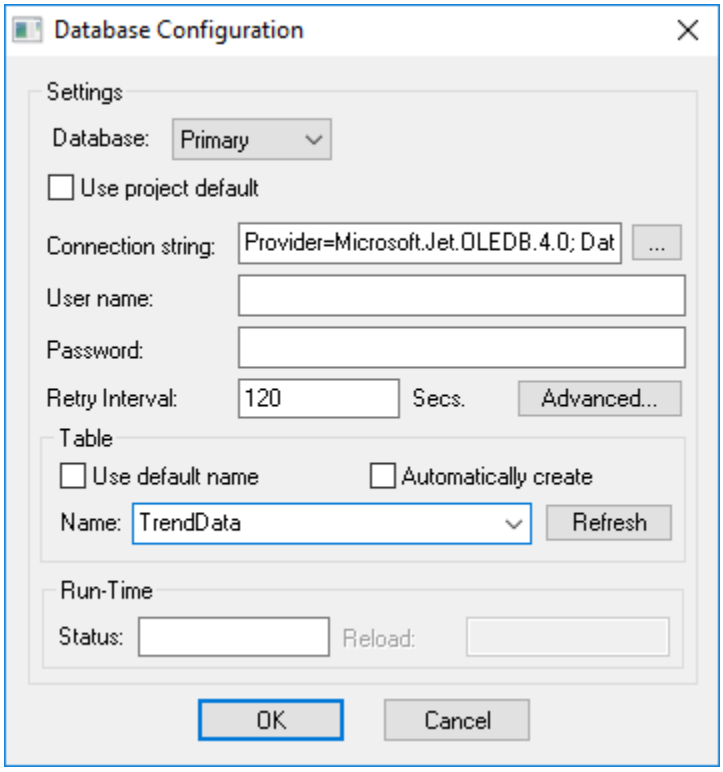

***Selecting the OLE DB Provider***

Select the Microsoft Jet 4.0 OLE DB Provider and click **Next »**. In the following window, you should specify the database path:




***Selecting the database file***

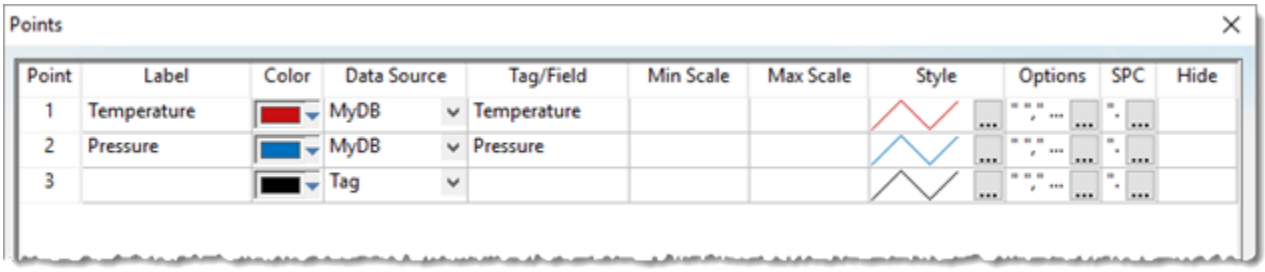
Click **OK** to finish the Connection String configuration. Now uncheck the option **Use default name** and select the table from your database as shown below:



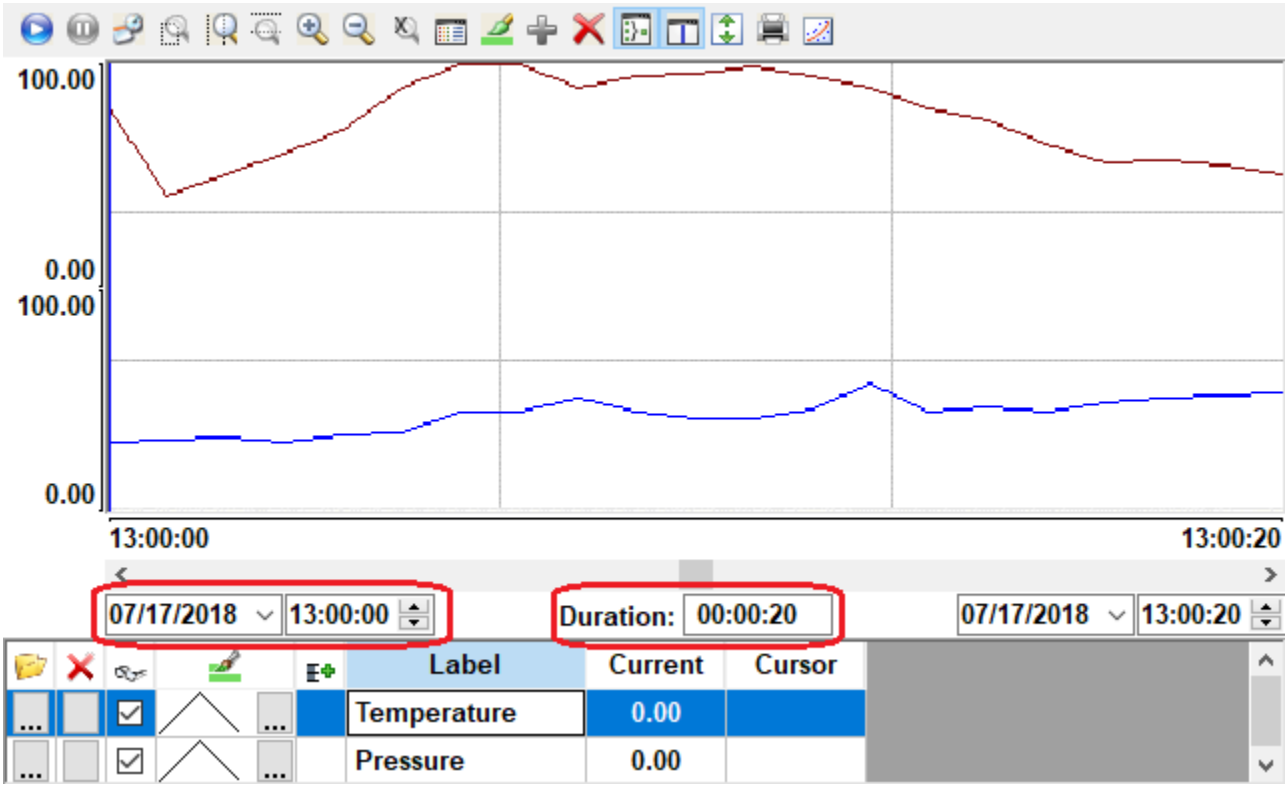

***Selecting the table in the database***

Click **OK** on this window and **OK** again to finish the data source configuration and close the *Data Source* configuration window.

Now we need to define Temperature and Pressure. They will be represented by points on our Trend Control. Double-click on the Trend Control again to access the Object Properties window and then click **Points**.... Your next step is to define the points according to the following figure:



If you run the trend, it will start with the current date/time. In order to see the data in the chart you will have to properly configure the start date/time as shown below:




### *Display text- and image-based trend annotations in a trend control*

Use trend annotations to display additional text and images in a trend control during project run time. The annotations' content and settings are stored in the same database that stores the historical data.

> **Note:** This feature is not supported in projects that are configured to run on the Embedded target platform. For more information, see About target platforms, product types, and target systems on page 100.

When you configure the data source for a Trend Control object, you can connect to an external database that stores the historical data that you want to display. The historical data might be generated by a Trend worksheet in the same project, or they might be provided by some other source, but in either case, they are typically stored in a single database table with one field (column) for each trend point.

A database can have many tables, however, so you can create another table to store your trend annotations. The table can be named anything (e.g., `TrendNotes`), but it must have the following fields (columns):

| Field | Description |
|---|---|
| **`Type`** | The type of annotation:<br>• `0` = Image<br>• `1` = Text |
| **`AnnotationID`** | The annotation ID for the specific trend point with which the annotation should be associated. Multiple annotations can be associated with the same point.<br>Associating an annotation with a point ensures that the annotation will be displayed correctly in the trend control. If the trend control is configured to display multiple sections, the annotations associated with each point will be displayed in that point's section. Also, if a point is hidden or removed from the |

| Field | Description |
|---|---|
| | trend control during run time, its associated annotations will also be hidden or removed.<br><br>The annotation ID for a point can be configured in that point's options. For more information, see Options on page 431. If no annotation ID has been configured for a point, you can associate annotations with the point's tag/field instead. |
| **AnnotationContent** | The content of the annotation:<br><br>• If the value in the **Type** field is `0`, the value in this field should be the name of the image file (e.g., `image.jpg`). The file should be located Web sub-folder of your project folder (e.g., *<project name>*`/Web/image.jpg`), on the computer that hosts your project runtime server.<br><br>• If the value in the **Type** field is `1`, the value in this field should be a plain text comment. |
| **X1** | The left border of the annotation box, specifed in the same units as the trend control's X axis. If the X axis is set to **Date/Time**, the value in this field should be an appropriate time stamp (e.g., `10/10/2009 10:00:00`). For more information, see Axes dialog on page 436. |
| **X2** | The right border of the annotation box, specified in the same units as the trend control's X axis. If the X axis is set to **Date/Time**, the value in this field should be an appropriate time stamp (e.g., `10/10/2009 10:00:00`). For more information, see Axes dialog on page 436. |
| **Y1** | The top border of the annotation box, specified in the same units as the trend control's Y axis. For more information, see Axes dialog on page 436. |
| **Y2** | The bottom border of the annotation box, specified in the same units as the trend control's Y axis. For more information, see Axes dialog on page 436. |
| **Z** | The Z-index of the annotation box, which determines whether it is drawn in front of or behind other annotation boxes. The greater the Z-index, the more "forward" the annotation box will be.<br><br>**Note:** If the value of **Z** is negative, the annotation box will be drawn behind the trend control. |
| **ImageFitMode** | How the image should fit in the annotation box:<br><br>• `0` = Use the specified image as a fill pattern.<br><br>• `1` = Scale the specified image to fit the annotation box.<br><br>• `2` = Resize the annotation box to fit the specified image. The bottom-left corner defined by **X1** and **Y2** remains fixed, while the top and right borders defined by **X2** and **Y1** are moved as needed.<br><br>The value in this field is significant only if the value in the **Type** field is `0`. |
| **TextBaseWidth** | The width of the text label (in pixels). If this value is not the same as the width of the annotation box (which is **X2** minus **X1**), the text label will be scaled horizontally to fit.<br><br>The value in this field is significant only if the value in the **Type** field is `1`. |
| **TextBaseHeight** | The height of the text label (in pixels). If this value is not the same as the height of the annotation box (which is **Y1** minus **Y2**), the text label will be scaled vertically to fit.<br><br>The value in this field is significant only if the value in the **Type** field is `1`. |
| **Font** | The font settings, in the following format:<br>*size* **:** *color* **:** *alignment*<br><br>*size* is the font size in points. The default font size is the same as the size specified for the Y-axis labels. |

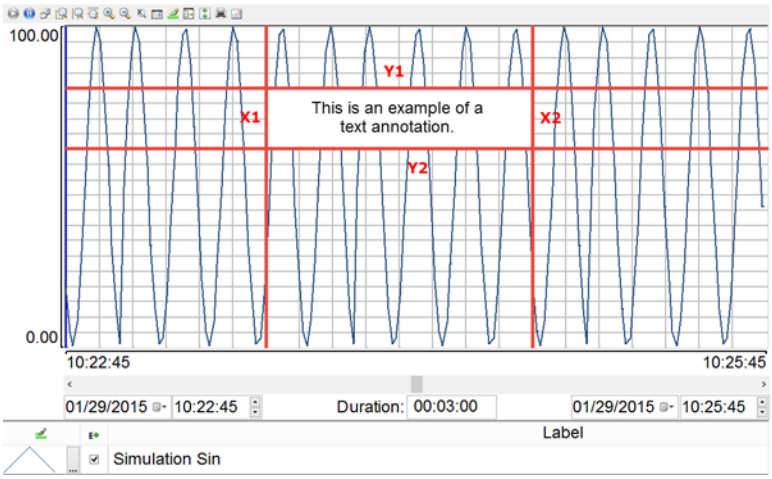
| Field | Description |
|---|---|
| | *color* is the font color, specified as an RGB code (e.g., `0,128,128`). For more information about RGB codes, see Color Interface on page 74. The default font color is the same as the color specified for the Y-axis labels.<br><br>*alignment* is the alignment of the text within the text label:<br>• `0` = Top-left (default)<br>• `1` = Top-center<br>• `2` = Top-right<br>• `3` = Middle-left<br>• `4` = Middle-center<br>• `5` = Middle-right<br>• `6` = Bottom-left<br>• `7` = Bottom-center<br>• `8` = Bottom-right<br><br>**Note:** The font style cannot be specified separately. It will be the same as the style specified for the Y-axis labels.<br><br>The value in this field is significant only if the value in the **`Type`** field is `1`. |

With the table's fields (columns) configured like this, each record (row) in the table can store the content and settings for an single annotation. Then, in the trend control's **Data Source** settings, in the **Ann. Source** box, type the name of this table (e.g., `TrendNotes`). For more information, see Data Sources on page 441.

With everything properly configured, the project runtime server will get the data stored in this table and use them to display your annotations in the trend control during project run time. For example, given a record (row) with the following values:

| Field (Column) | Value |
|---|---|
| **`Type`** | `1` |
| **`AnnotationID`** | `SimulSin` |
| **`AnnotationContent`** | `This is an example of a text annotation.` |
| **`X1`** | `01/29/2015 10:23:36` |
| **`X2`** | `01/29/2015 10:24:45` |
| **`Y1`** | `81` |
| **`Y2`** | `62` |
| **`Z`** | `1` |
| **`ImageFitMode`** | `0` |
| **`TextBaseWidth`** | `381` |
| **`TextBaseHeight`** | `87` |
| **`Font`** | `11:0,0,0:4` |

...the annotation will be displayed in the trend control like this:




***An example of a text-based trend annotation displayed in a trend control***

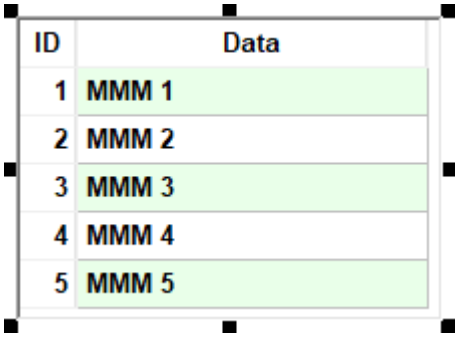
Please note that the red lines and captions in the example above are included only to highlight the borders (X1, X2, Y1, Y2) of the annotation box. They would not actually be displayed like this during project run time.

## Grid object

The Grid object allows you to read/write data in a tabular format from the data source configured in the object.

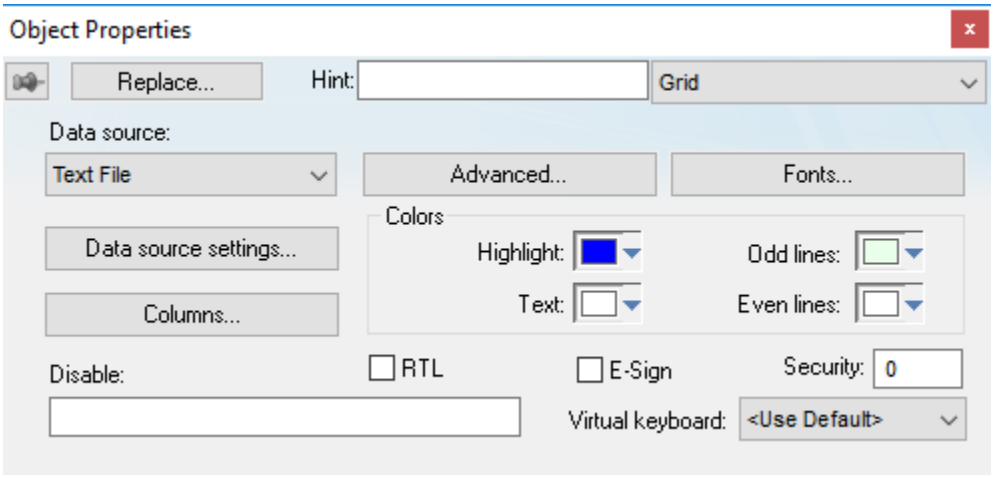
To draw one, do the following:

1. On the **Draw** tab of the ribbon, in the **Data Objects** group, click **Grid**.
2. Click on the screen worksheet, and then draw a box of the desired size (while holding down the mouse button).
3. Release the mouse button, and the Grid Object will display.




***Sample Grid Object***

Right-click on the Grid Object, and select **Properties** from the menu. The Object Properties dialog will open. Use this dialog to configure the Grid Object's parameters:




***Object Properties: Grid***

- **Data Source**: Select the data source type. The object supports three data sources:

| Data Source | Description |
|---|---|
| Text File | Displays data from a text file in the ASCII or Unicode format (e.g., CSV text files). |
| Class Tag | Displays values from a Class Tag, where the members of the tag are fields (columns) of the grid object, and each array position is one row of the grid object. |
| Database | Displays data from an SQL Relational Database, using ADO (ActiveX Database Object) to exchange data with the database. |

- **Data source settings**: Click to launch the *Data* dialog, where you can specify a data source for the Grid object.

- **Columns**: Click to launch the *Columns* dialog, where you can configure the settings (such as label, column, width, etc.) for the columns of the Grid object.
- **Advanced**: Click to launch the *Advanced* dialog, where you can configure several settings for the Grid object.
- **Fonts**: Click to launch the *Fonts* dialog, where you can configure the font settings for the text displayed in the Grid object.

> **Tip:** By default, the same text color is used for both the header and the body of the grid. If you want to set a different text color for the header, then manually edit the project file (*<project name>*.APP) to add the following setting:
>
> ```
> [Objects]
> GridHeaderTextColor=value
> ```
>
> *value* must be a hexadecimal RGB color value, such as FF0000 for red. Please note that this setting will apply to all grid headers in your project.

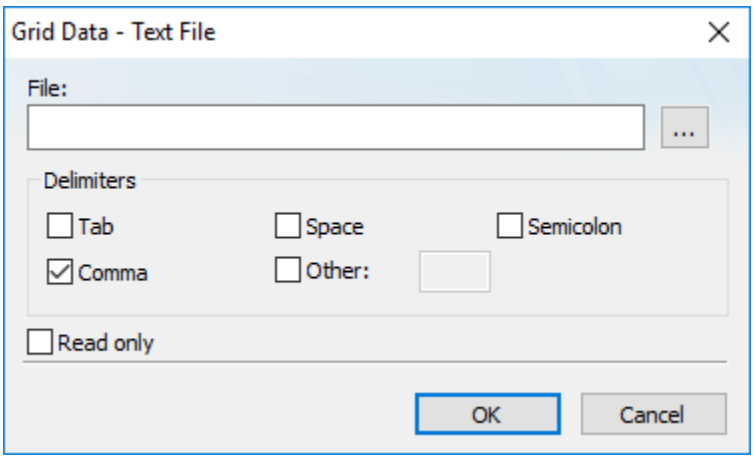
- **Highlight**: Select a background color for the selected row, during runtime.
- **Text**: Select a text color for the selected row, during runtime.
- **Odd lines**: Select a background color for the odd rows.
- **Even lines**: Select a background color for the even rows.
- **Disable**: You can enter an expression in this field to disable data input or action by the user.
- **E-Sign**: When you check this option, the user will be prompted to enter an electronic signature before entering or modifying data on the object.
- **Security**: Enter the security system access level required for the object/animation.
- **Virtual keyboard**: Select a Virtual Keyboard type used for this object. The option <Use Default> selects the default Virtual Keyboard configured in the *Viewer* settings (**Viewer** on the Project tab of the ribbon). You can also specify a different Virtual Keyboard for this Grid object.

## *Data dialog*

This dialog allows you to configure the data source for a Grid object.

### Grid Data – Text File

When the **Data Source** type is set to Text File, you can configure the following settings:



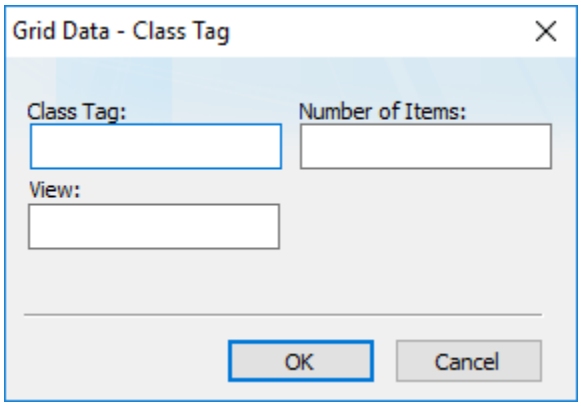

- **File**: Enter the name of the text file source. You can either type the file name and its path or click the ... button to browse for it. (If the file is stored in your project folder, you can omit the path in the name.)

> **Tip:** You can configure tag names between curly brackets {TagName} in the **File** field.

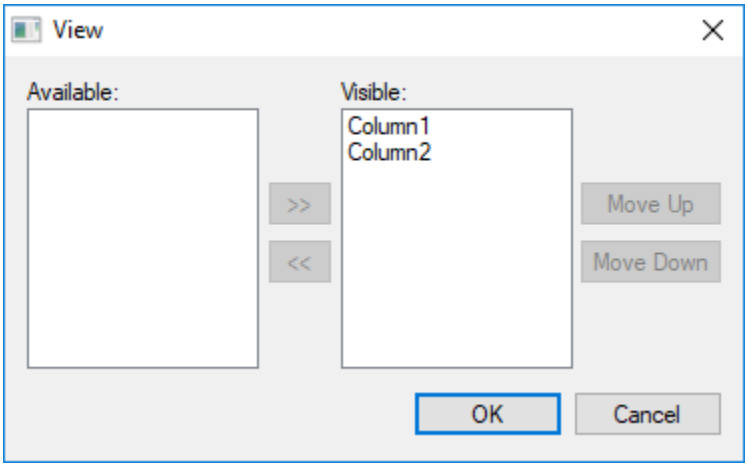
- **Delimiters**: Set the delimiter(s) used in the data source file. For instance, if the data will be read from a CSV (comma separated values) file, you would select the **Comma** option. You can even choose a custom delimiter by checking the **Other** option and typing the custom delimiter in the field beside it.
- **Read only** checkbox: When this option is checked, the Grid object will only read data from the specified file. The object will not write anything to the file.

### Grid Data – Class Tag

When the **Data Source** type is set to Class Tag, you can configure the following interface:

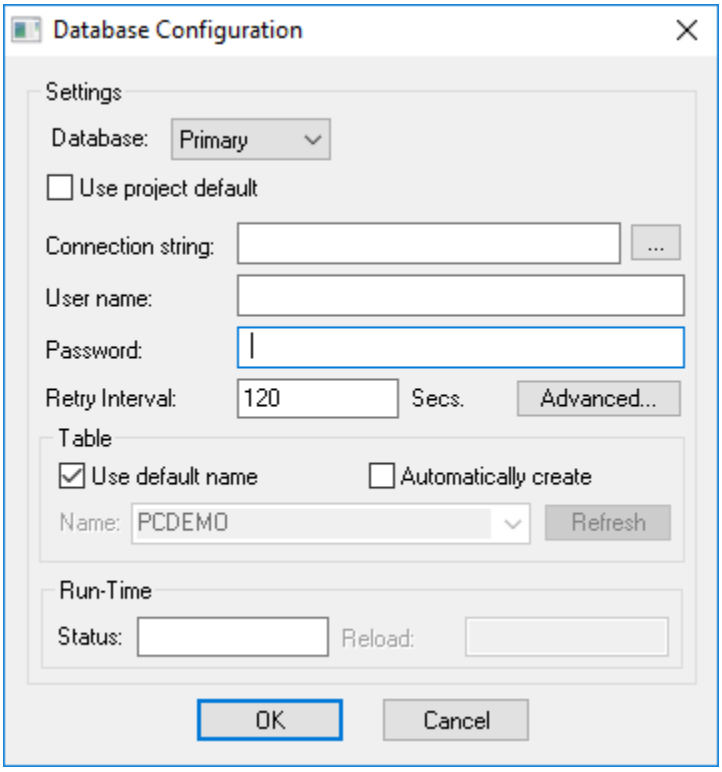



- **Class Tag**: Enter the name of the main class tag source. (Do not specify a specific member of the class tag.) You can specify the initial array position in this field (e.g., `Mytag[10]`); otherwise, 0 (zero) will be used as the initial position by default.
- **Number of Items**: Enter the number of array positions from the **Class Tag** that should be displayed.
- **View**: When the tag configured in the optional field changes value (e.g., toggles) during runtime, the grid object launches a dialog, allowing the user to show/hide each column or modify their positions.



### Grid Data – Database Configuration

When the **Data Source** type is set to Database, you can configure the following settings:

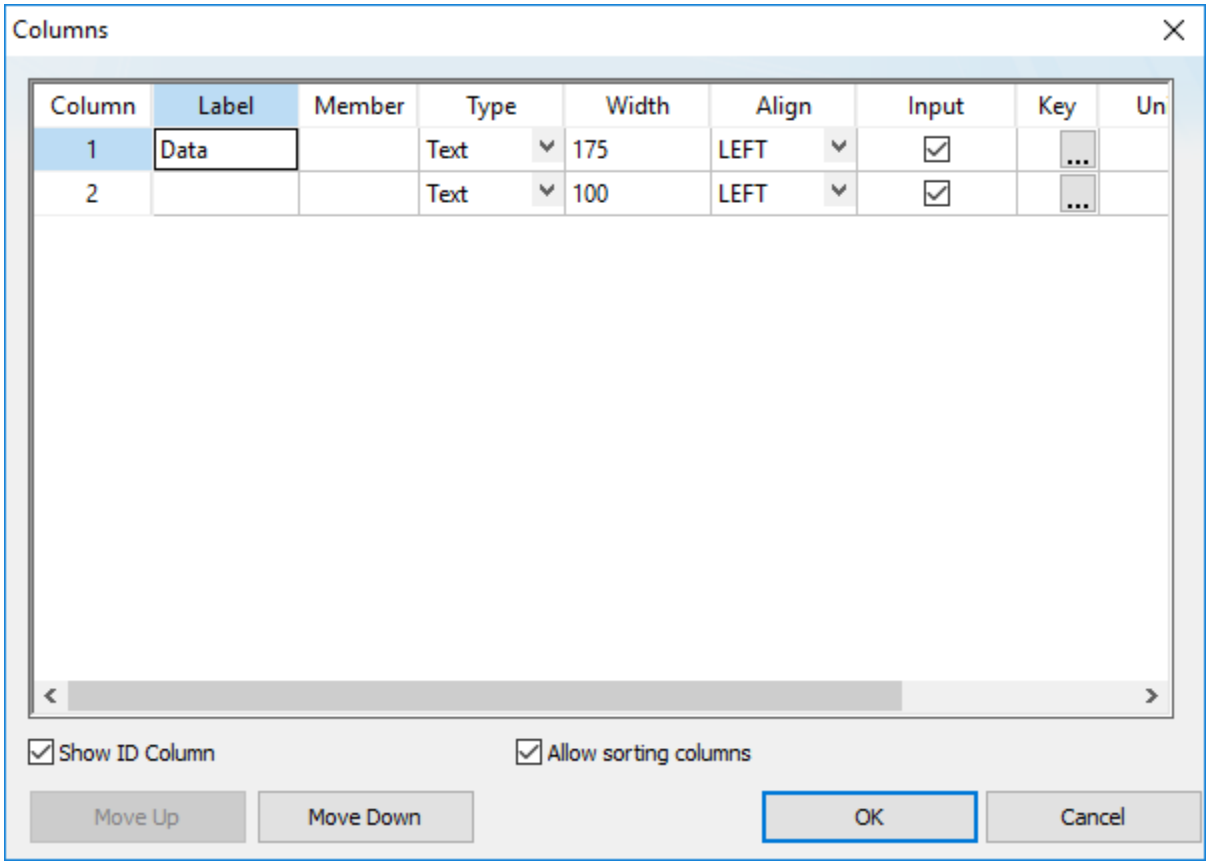



Please refer to the *Database Configuration* dialog for further information about this dialog.

> **Note:** Configuring a secondary (redundant) database for a Grid object will make the content of the object read-only — that is, the object can read and display data from the primary or secondary database, as needed, but changes made in the object cannot be written to either database. This also means that the options **Insert Trigger** and **Save on data change** will have no effect. For more information about those options, see Advanced dialog on page 471.

### *Columns dialog*

This topic describes the Columns properties of the Grid object.




***Column properties for a Grid object***

**Column**

This number indicates the order of the columns in the grid, from left to right. To reorder the columns, use the **Move Up** and **Move Down** buttons at the bottom of the dialog box.

**Label**

Type a label for the column, which will be displayed in the header row of the grid.

You can type a string expression for this setting (e.g., `{MyLabel}`). When the value of the expression changes during project run time, the label is changed to match.

When the label is blank, the width of the entire column is set to 0. You can do this to hide columns during project run time.

This option is not available if the **Show header** option in the *Advanced* settings is not selected.

**Member**

Enter the name of the class member to which this column will be linked. If this field is left in blank, the text configured in the **Label** field will be used as a default member name.

This setting is available only when the Grid object's data source is set to **Class Tag**.

**Field**

Enter the name of the field (column) in the database table to which this column will be linked. If this field is left in blank, the text configured in the **Label** field will be used as a default field name.

You can configure valid SQL statements directly in the field (e.g., `List(DISTINCT [Cell_Name]) AS [Cell Name]`). You can also type a string expression that provides the SQL statement.

This setting is available only when the Grid object's data source is set to **Database**.

### Type

Select the data type of the values that will be displayed in the column. In particular, when the Grid object's data source is set to **Database**, make sure the **Type** configured for each column of the grid matches the data type of the corresponding field in the database. The following types are available:

| Type | Description |
|---|---|
| **Text** | Display string values (i.e., alphanumeric text). |
| **Numeric** | Display numeric values. |
| **Picture** | Display the specified image file (.bmp or .ico) located in the Web sub-folder of your project folder. For example, if the value from the data source is **`MyFile.bmp`**, the Grid object will display the image file located at *`<project name>`*`/Web/MyFile.bmp`.<br><br>If it is a .bmp file, the image may be resized or stretched to fit the column. If it is an .ico file, which can contain several scaled versions of the same image, the version that best fits the column will be automatically selected and then resized if necessary. In both cases, resized images cannot be aligned in the column (i.e., the **Align** setting for the column will have no effect).<br><br>If the specified image file cannot be found during project run time, a default image is displayed instead. To change the default image, manually edit your project file (*`<project name>`*`.APP`) to change the following property:<br><br>**`[Objects]`**<br>**`GridDefaultPicture=<file name>`** |
| **Check-box** | Display the values, typically Boolean, as check boxes. While a value is TRUE (i.e., non-zero), the check box is selected. While a value is FALSE (i.e., zero) or NULL, the check box is cleared.<br><br>If the **Input** option is selected for the column, and the user selects or clears the check box during project run time, the value is set to 1 or 0, respectively. This is true even if the value is not Boolean. Therefore, it is possible for a large numeric value (e.g., 1000) or even a string value to be reset to 1 when the check box is cleared and then selected again. |
| **Time** | Display the values in the time format (e.g., **`HH:MM:SS`**).<br><br>This type is available only when the Grid object's data source is set to **Database**, because each value is assumed to be a database timestamp that can be parsed to get the actual time. |
| **Date** | Display the values in the date format (e.g., **`MM/DD/YYYY`**).<br><br>This type is available only when the Grid object's data source is set to **Database**, because each value is assumed to be a database timestamp that can be parsed to get the actual date. |
| **Date/Time** | Display the values in the date/time format (e.g., **`MM/DD/YYYY HH:MM:SS`**).<br><br>This type is available only when the Grid object's data source is set to **Database**, because each value is assumed to be a database timestamp that can be parsed to get the actual date and time. |
| **Time – UTC** | Display the values in the time format (e.g., **`HH:MM:SS`**). |

| Type | Description |
|---|---|
| | This type is available only when the Grid object's data source is set to **Database**, because each value is assumed to be a database timestamp that can be parsed to get the actual time. Also, the timestamp is assumed to be in Coordinated Universal Time (UTC), and it is automatically converted to the local time zone in the project viewer / thin client. |
| **Date – UTC** | Display the values in the date format (e.g., **MM/DD/YYYY**).<br>This type is available only when the Grid object's data source is set to **Database**, because each value is assumed to be a database timestamp that can be parsed to get the actual date. Also, the timestamp is assumed to be in Coordinated Universal Time (UTC), and it is automatically converted to the local time zone in the project viewer / thin client. |
| **Date/Time – UTC** | Display the values in the date/time format (e.g., **MM/DD/YYYY HH:MM:SS**).<br>This type is available only when the Grid object's data source is set to **Database**, because each value is assumed to be a database timestamp that can be parsed to get the actual date and time. Also, the timestamp is assumed to be in Coordinated Universal Time (UTC), and it is automatically converted to the local time zone in the project viewer / thin client. |

For more information about the date format, see About the date format and how to change it on page 694.

**Width**

Enter the width of the column, in pixels.

You can type the name of an Integer tag for this setting (e.g., `Column1Width`). When the value of the tag changes during project run time, the width is changed to match.

> **Note:** When the Grid object's data source is set to **Class Tag**, if the **Label** field is configured but the **Member** field is not, then the **Width** setting is ignored and the column is auto-sized to fit its contents.

**Align**

Select the horizontal alignment for the data displayed in the column. There are three options: **Left**, **Right**, or **Center**.

**Input**

Select this option to allow the user to enter data in the column during project run time.

**Key**

Click the More button (...) and then use the *Key Modifier* dialog box to define a shortcut key for the column. When the user presses the shortcut key during project run time, the rows of the grid are sorted according to the values in the column. This is an alternative to clicking or tapping the column's label, so it is especially useful if you are developing your project for a device that has a keyboard but not a mouse or touchscreen.

The shortcut key is useful only if **Allow sorting columns** option is selected (see below).

**Unit**

Enter the name of the engineering unit (i.e., the unit of measurement), if any, that applies to the data to be displayed in the column.

You can type a string expression for this setting (e.g., `{MyUnit}`). When the value of the expression changes during project run time, the engineering unit is changed to match.

**Decimal Points**

Enter the number of decimal places to be displayed in the column, if the column is configured to display numeric values.

You can type the name of an Integer tag for this setting (e.g., `Column1Decimals`). When the value of the tag changes during project run time, the number of decimal places is changed to match.

**Show ID Column**

This option, which is selected by default, displays an additional column of ID numbers for the rows of the grid.

**Allow sorting columns**

This option, which is selected by default, allows the user to sort the rows of the grid according to the values in a selected column. To select a column during project run time, the user can either click/tap the column's label or press the column's shortcut key.

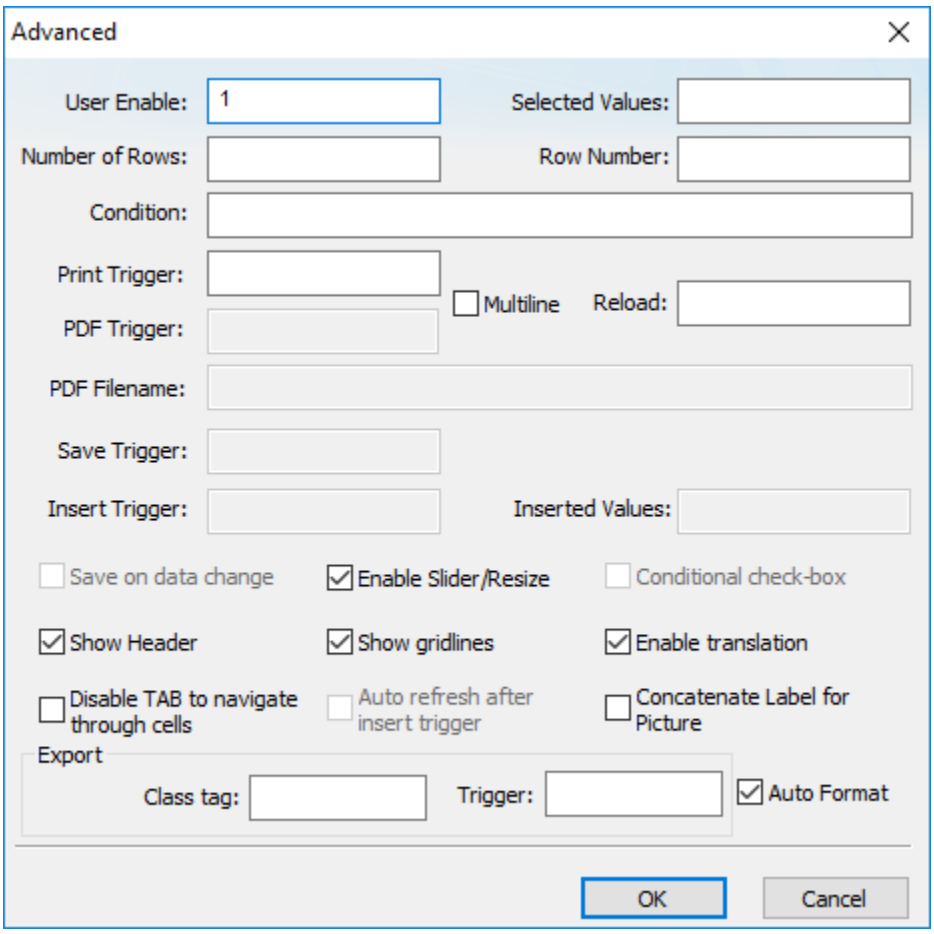
**Note:**

When the Grid object's data source is set to **Class Tag**, if the entire *Columns* dialog box is left blank, then the grid automatically displays the values from all class members with the following default settings:

| Setting | Value |
|---|---|
| **Label** | The name of the class member. |
| **Type** | **Text** |
| **Width** | The minimum size to display the name of the class member on the header row of the grid. |
| **Align** | **Center** |
| **Input** | Enabled (selected). |
| **Key** | None. |
| **Unit** | The engineering unit of the class member. |

### *Advanced dialog*

This dialog allows you to configure the advanced settings for a Grid object.




***Advanced dialog***

**User Enable**

If the value of this tag is TRUE (different from 0), the user can select different rows of the object by clicking on them during run time. This box can be configured with a tag or with a numeric value.

**Selected Values**

The values from each column of the selected row are written to each position of the array tag configured in this box. Moreover, you can modify the value of the cells currently selected in the Grid object by changing the value of array tag configured in this box. The initial array position (offset) can be configured in this box.

**Number of Rows**

The grid object writes the number of rows currently available in the grid object to the tag configured in this box.

**Row Number**

The Grid object writes the number of the row currently selected during run time. In addition, you can select different rows by writing their values in this tag.

**Condition**

Enter an expression to filter the grid data; only rows that match the expression will be displayed. The expression must use the following syntax:

```
[Column] Operator Value
```

For example…

```
[ColumnX] > 200
```

When **Data Source** (in the *Grid Object Properties* dialog) is set to Text File or Class Tag, the ***Column*** is the value specified in the Label. When **Data Source** is set to Database, the column is the value specified in the Field. (In this case, if the Field is left blank, then the column value specified is the Label.)

Also, expressions for Database must be formatted like a SQL Where statement. The following table shows which operators should be used:

**Condition Expression Operators**

| Comparison | Data Source is Text File… | Data Source is Database… |
|---|---|---|
| equal to | = | LIKE |
| not equal to | <> | NOT LIKE |
| wildcard, single character | ? | _ |
| wildcard, unlimited characters | * | % |

As such, the following expression for Text File…

```
[C1] = 'ab?d'
```

…means the same as the following expression for Database…

```
[C1] LIKE 'ab_d'
```

Finally, you can combine several expressions simultaneously in the **Condition** box, using the logic operators **AND**, **OR**, and **NOT**. For example:

```
[ColumnAge] > '10' OR [ColumnName] = 'John' AND [ColumnDate] > '05/20/2003'
```

> **Note:** If you have configured multiple columns to contain date/time values but each column is of a different type (i.e., a different time zone), the filter will convert the values in all of the columns to match the type of the last column. For example, if columns 2 and 3 are configured as **Date/Time** but column 4 is configured as **Date/Time - UTC**, the values in columns 2 and 3 will be converted to UTC for the purposes of filtering.

> **Tip:** You can configure tags between curly brackets **{TagName}** in the **Condition** box to change the filtering condition during run time.

**Print Trigger**

When the tag configured in this box is toggled, the current state of the Grid object is sent to the default printer.

**PDF Trigger**

When the tag configured in this box is toggled, the current state of the Grid object is saved as a PDF file at the location specified by **PDF Filename**.

**PDF Filename**

Enter a complete file path and name where the PDF file is to be saved. You can also enter a tag name using the `{tag}` syntax.

> **Note:** **PDF Trigger** and **PDF Filename** are not supported in projects running on Windows Embedded or Thin Client.

**Multiline**

When this option is selected, the print output or PDF will be formatted according to the available column space, and the text within each cell will be wrapped so that all of it is shown.

**Reload**

When the tag configured in this box is toggled, the object reloads the data from the data source and displays it.

**Save Trigger**

When the tag configured in this box is toggled, the data source (Text File or Database) is updated with the current values of the grid object. (This box is not available when the Data Source type is Class Tag, because the values are automatically updated in the tags as you change a cell in the grid.)

**Insert Trigger**

When the **Auto refresh after insert trigger** option is selected, the tag configured in this box is used as a trigger to refresh the database table. Whenever the value of the tag changes, a new row is added to the table and the values of the array configured in the **Inserted Values** box are automatically inserted.

**Inserted Values**

If the **Insert Trigger** is being used, then the array tag configured in this box provides the values that will be inserted. This box must only contain an array tag, although it can be of any size.

**Save on data change**

When this option is selected, the values are updated on the data source (Text File or Database) as soon as the user enters a new value on the grid, during run time. (This option is disabled when the Data Source type is Class Tag, because the values are automatically updated in the tags as the user changes the value of the cells in the grid.)

**Enable Slider/Resize**

If this box is not checked, the user is unable to scroll the list by dragging the slider button, or to change the cell's size during run time.

**Conditional check-box**

When this option is selected, the user cannot check a checkbox on the Grid during run time, unless all preceding checkboxes in the same column are also checked. This option is especially useful when you want to oblige the user to follow a pre-defined sequence. This box is not available when the Data Source type is Class Tag.

**Show Header**

When this option is selected, the header of the Grid object is visible during run time, displaying the label of each column.

**Show gridlines**

When this option is selected, the gridlines of the Grid object are visible during run time.

**Enable translation**

When this option is selected, the text displayed by the Grid object will be subject to translation by the Translation Tool during run time.

> **Note:** This does not include columns which have been configured to accept user input (i.e., for which the **Input** option in the *Columns* dialog box has been selected).

**Disable TAB to navigate through cells**

When this option is selected, the user can only navigate through the cells of the Grid Object with the arrow keys, rather than the Tab key. You should disable the Tab key for navigation if you want it to be used for switching to the next object that supports focus on the screen.

**Auto refresh after insert trigger**

See **Insert Trigger** above.

**Concatenate Label for Picture**

When this option is selected, the reference name for the picture is the result of the concatenation of the name in the Field column with the value of the Label column. The result will be <Label name>_<Field value>.

**Export**

This interface allows you to export the data from the grid object to a class-array tag, regardless of the Data Source selected for the object. The following settings must be configured to support this feature:

| Setting | Description |
|---|---|
| Class tag | Type the main tag name of the class-array tag that will receive the exported values. Each row from the grid object will be exported to one array position of the array tag, by matching column labels. The initial array position can be configured in this box; 0 is the default. |
| Trigger | When the tag configured in this box changes value (e.g., toggles), the data is exported from the Grid object to the class-array tag configured in the Class tag field. |

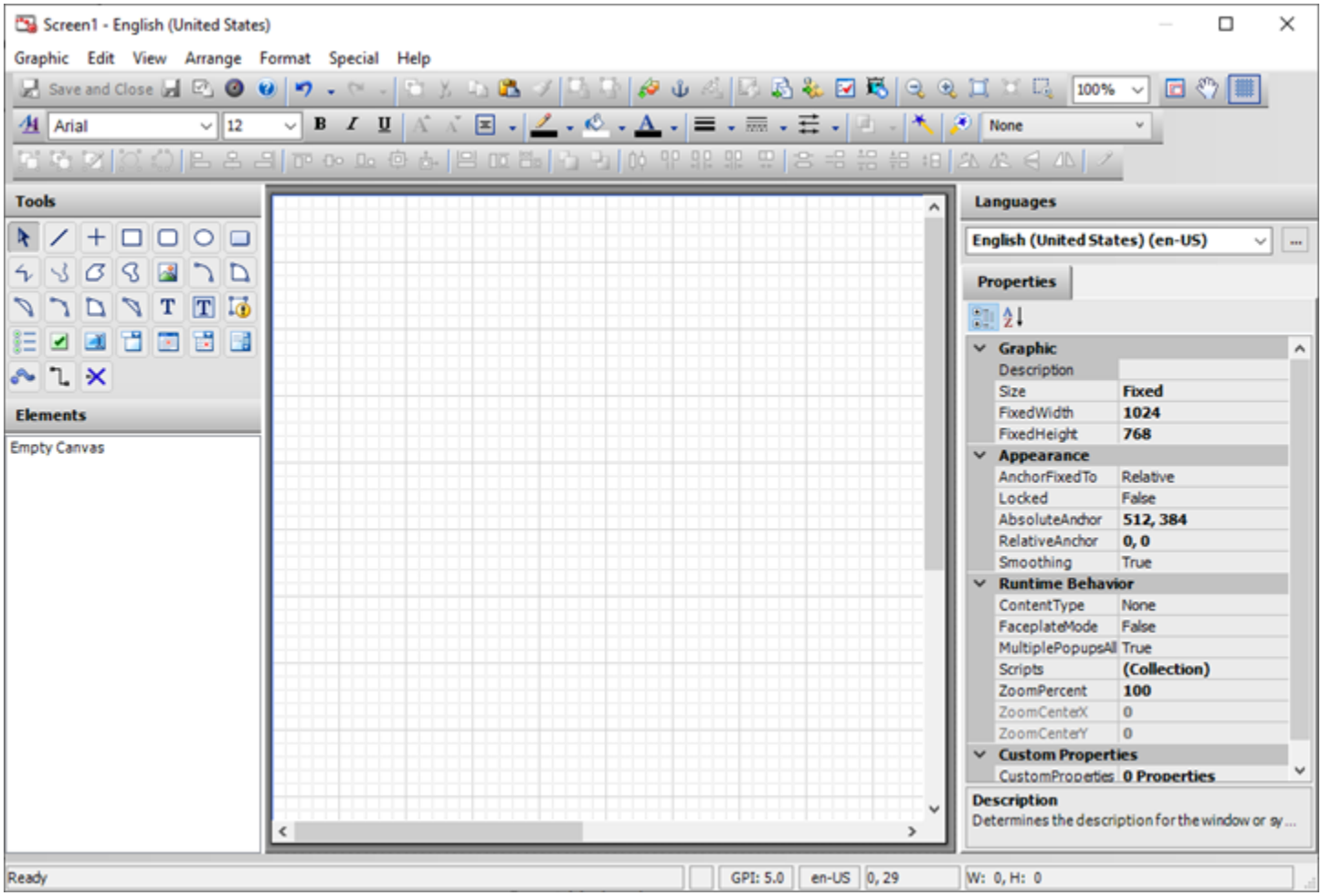
> **Tip:** The Export feature is an easy and powerful tool to transfer data from different data sources to tags. After exporting the data to tags, you can use different tasks to manipulate the data, such as the FileWrite() function, or the Recipe or Report tasks to save the data in text files (e.g., CSV files).

**Auto Format**

When this option is selected, decimal values in columns of **Numeric** type will be formatted according to the virtual table created by the `SetDecimalPoints()` function. This option will work only in columns for which **Decimal Points** are not already configured. For more information, please see Grid Object: *Columns* dialog.

# Industrial Graphics

This section describes how to use the Industrial Graphics editor and symbol library to create Industrial Graphics screens that you can use in your projects.




The Industrial Graphics editor works as a companion to the native graphics tools in Studio. It provides new features and options that were not previously available in Studio, it lets you reuse symbols that you created in other applications, and it increases interoperability across your entire automation solution.

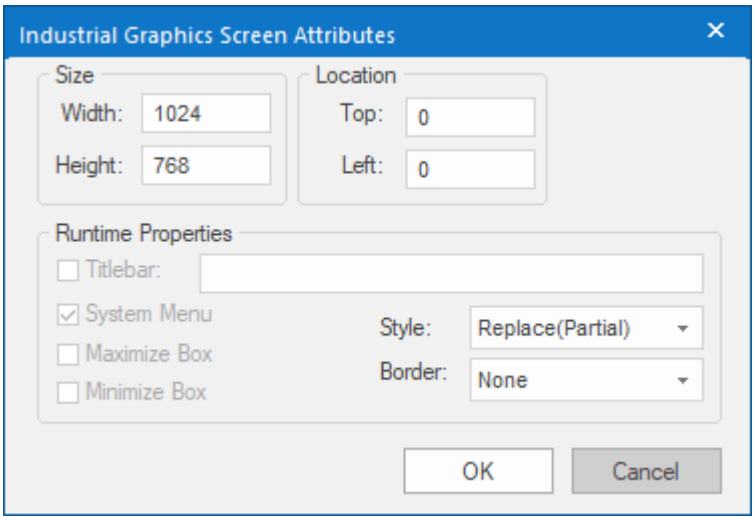
## Create a new Industrial Graphics screen

This task describes how to create a new Industrial Graphics screen and then open it for editing.

To create a new Industrial Graphics screen:

1. Do one of the following:
   - On the **Insert** tab of the ribbon, in the **Graphics** group, click **Industrial Graphics Screen**; or
   - On the **Graphics** tab of the *Project Explorer*, right-click **Screens**, and then on the shortcut menu, select **Insert Industrial Graphics Screen**.

   The *Industrial Graphics Screen Attributes* dialog box is displayed.




2. Configure the size and location of the screen when it is opened:
   a) Under **Size**, specify the **Width** and **Height** (in pixels) of the screen when it is opened.
      The default size is equal to the resolution that you selected when you created your project.
   b) Under **Location**, specify the distances (in pixels) between **Top** and **Left** sides of the screen and corresponding edges of the viewer window.
      This is also known as the X/Y position of the screen within the project viewer.

   You may choose to let the user resize or move the screen after it is opened, depending on which option you select for **Border** below.

3. For **Style**, choose one of the following:

| Option | Description |
|---|---|
| **Overlapped** | Opens the screen without closing any other screens. |
| **Popup** | Forces the screen in front of all other screens but does not close them. |
| **Replace (Partial)** | Opens the screen and closes all other **Replace** screens that it partially covers. This is the default for all new screens. |
| **Dialog** | Similar to **Popup**, except that the other screens are also disabled until the dialog is closed by the user. |
| **Replace (Complete)** | Similar to **Replace (Partial)**, except that it closes only other **Replace** screens that it completely covers. |

4. For **Border**, choose one of the following:

| Option | Description |
|---|---|
| **None** | No border; the screen is a flat, immovable rectangle within the project viewer. This is the default for all new screens. |
| **Thin** | A thin border that makes the screen a movable window within the project viewer. Includes the title bar. |
| **Resizing** | A thick border that makes the screen a movable, resizable window within the project viewer. Includes the title bar. |

5. If you chose either **Thin** or **Resizing** for **Border**, you can now select the **Titlebar** option to add a title bar to the screen, and then after you do that, you can do the following:
   a) In the box to the right of the option, type the title of the screen.
      It is useful to specify a title even if the title bar is not shown, because the title is always included when the screen is printed. You can also type a string expression for this setting (e.g., `{MyScreen}`).
   b) Configure the remaining options for the title bar:
      **System Menu**
      Shows a menu of basic window commands at the left end of the title bar.
      **Maximize Box**
      Shows the **Maximize** button at the right end of the title bar.
      **Minimize Box**
      Shows the **Minimize** button at the right end of the title bar.
6. Click **OK**.

The new screen is added to the **Screens** folder in the *Project Explorer*, and then it is automatically opened for editing in the Industrial Graphics editor.

For more information about how to use the Industrial Graphics editor to edit your screen, see the help system that is available within the editor itself: in the editor, go to **Help**, and then select **Help Topics**.

When you are done editing the screen, click **Save and Close** in the editor's toolbar. The screen is saved, and you are returned to the project development environment where you can proceed to use the screen in your project as you normally would. The screen is also automatically saved as HTML for run time, so you do not need to do that manually like you do for native screens.

> **Note:** The Industrial Graphics editor window is modal, which means only one editor window can be open at a time and while it is open, the rest of the project development environment is disabled.

To edit the screen again, open it from the *Project Explorer* and then do one of the following:

- On the **Draw** tab of the ribbon, in the **Industrial Graphics** group, click **Edit Symbol**;
- Right-click in the screen, and then on the shortcut menu, select **Edit Symbol**; or
- Double-click anywhere in the opened screen.

To delete or rename the screen, right-click the screen in the *Project Explorer*, and then on the shortcut menu, select the appropriate command.

It is not possible for two or more screens to have the same name, even if they are different types of screens and have different file extensions in your project folder (e.g., .scc or .scr for native screens, .sca for Industrial Graphics screens), because all of the screens are eventually saved as HTML for run time. If you want to reuse screen files from an existing project in your new project, you should rename those screens before you try to add them to your project folder.

## Create a new Industrial Graphics symbol

This task describes how to create a new Industrial Graphics symbol and then open it for editing. A symbol is a premade object or group of objects that can be reused.

If you want to add a symbol to a toolset (i.e., a folder) in the *Project Explorer*, you must create the toolset first because symbols and toolsets cannot be moved after they have been created. For more information, see Create a new Industrial Graphics toolset on page 479.

To create a new Industrial Graphics symbol:

1. On the **Graphics** tab of the *Project Explorer*, do one of the following:
   - Right-click **Industrial Graphics Symbols**, and then on the shortcut menu, select **New Symbol**; or
   - Select a toolset under **Industrial Graphics Symbols**, right-click it, and then on the shortcut menu, select **New** > **Symbol**.

   The *New Symbol* dialog box is displayed.
2. In the **Name** box, type the name of the symbol, and then click **OK**.

   The name must start with a letter and can contain only letters, numbers, and underscore characters. For example, `Symbol_001`.

After you click **OK**, the new symbol is added to the **Industrial Graphics Symbols** folder in the *Project Explorer*, and then it is automatically opened for editing in the Industrial Graphics editor.

For more information about how to use the Industrial Graphics editor to edit symbols, see the help system that is available within the editor itself: in the editor, go to **Help**, and then select **Help Topics**.

When you are done editing the symbol, click **Save and Close** in the editor's toolbar. The symbol is saved, and you are returned to the project development environment where you can proceed to use the symbol in an Industrial Graphics screen.

> **Note:** The Industrial Graphics editor window is modal, which means only one editor window can be open at a time and while it is open, the rest of the project development environment is disabled.

To open the symbol again at any time, double-click it in the *Project Explorer*.

To delete, rename, or duplicate a symbol that you previously created, do the following: right-click the symbol in the *Project Explorer*, and then on the shortcut menu, select the appropriate command.

## Create a new Industrial Graphics toolset

This task describes how to create a new Industrial Graphics toolset and then insert it into your project files.

In the context of Industrial Graphics, a "graphic toolset" is essentially a folder or collection of folders that you can use to organize your Industrial Graphics symbols.

To create a new Industrial Graphics toolset:

1. On the **Graphics** tab of the *Project Explorer*, right-click **Industrial Graphics Symbols**, and then on the shortcut menu, select **New Graphic Toolset**.
   The *New Toolset* dialog box is displayed.
2. In the **Name** box, type the name of the toolset, and then click **OK**.
   The name must start with a letter and can contain only letters, numbers, and underscore characters. For example, `Toolset_001`.

After you click **OK**, the new toolset is inserted under **Industrial Graphics Symbols** in the *Project Explorer*.

You can nest these toolsets as many levels deep as you want. To create a new folder within an existing folder, do the following: right-click the existing toolset, and then on the shortcut menu, select **New** > **Graphic Toolset**.

You cannot move toolsets after they have been created, however; if you accidentally create a toolset in the wrong location, you need to delete it and then create it again in the right location. As such, we recommend you outline your entire folder structure before you begin creating toolsets.

To delete or rename a toolset that you previously created, do the following: right-click the folder in the *Project Explorer*, and then on the shortcut menu, select the appropriate command.

## Import an Industrial Graphics symbol library

This task describes how to import an Industrial Graphics symbol library into your project

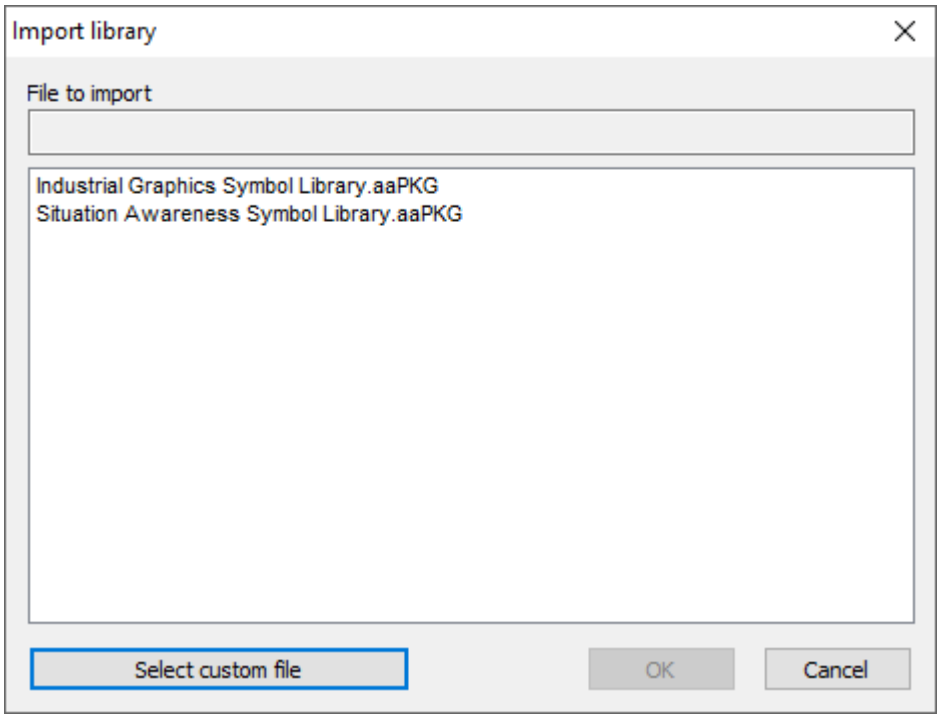
Industrial Graphics symbol libraries are distributed as .aaPKG files.

Two libraries of commonly used symbols are included with this software, and you are free to use these symbols in your project. Importing a symbol library into a project can significantly increase the size of that project, however.

To import an Industrial Graphics symbol library:

1. On the **Graphics** tab of the *Project Explorer*, right-click **Industrial Graphics Symbols**, and then on the shortcut menu, select **Import Library**.

   The *Import Library* dialog box is displayed.




   Studio will automatically detect and list any .aaPKG files that are at the following location on your computer:

   ```
   C:\Program Files (x86)\EcoStruxure Machine SCADA Expert 2020\Bin\GraphicContent
   ```

2. Do one of the following:
   - Select one of the listed .aaPKG files, and then click **OK**; or
   - Click **Select custom file**, use the file browser to locate and select an .aaPKG file, and then click **Open**.

The selected .aaPKG file is imported into your project. This can take a long time to complete.

Once the import is completed, the library is shown as a new toolset under **Industrial Graphics Symbols** in the *Project Explorer*.

## Embed an Industrial Graphics symbol in a screen

This task described how to embed an Industrial Graphics symbol in an Industrial Graphics screen.

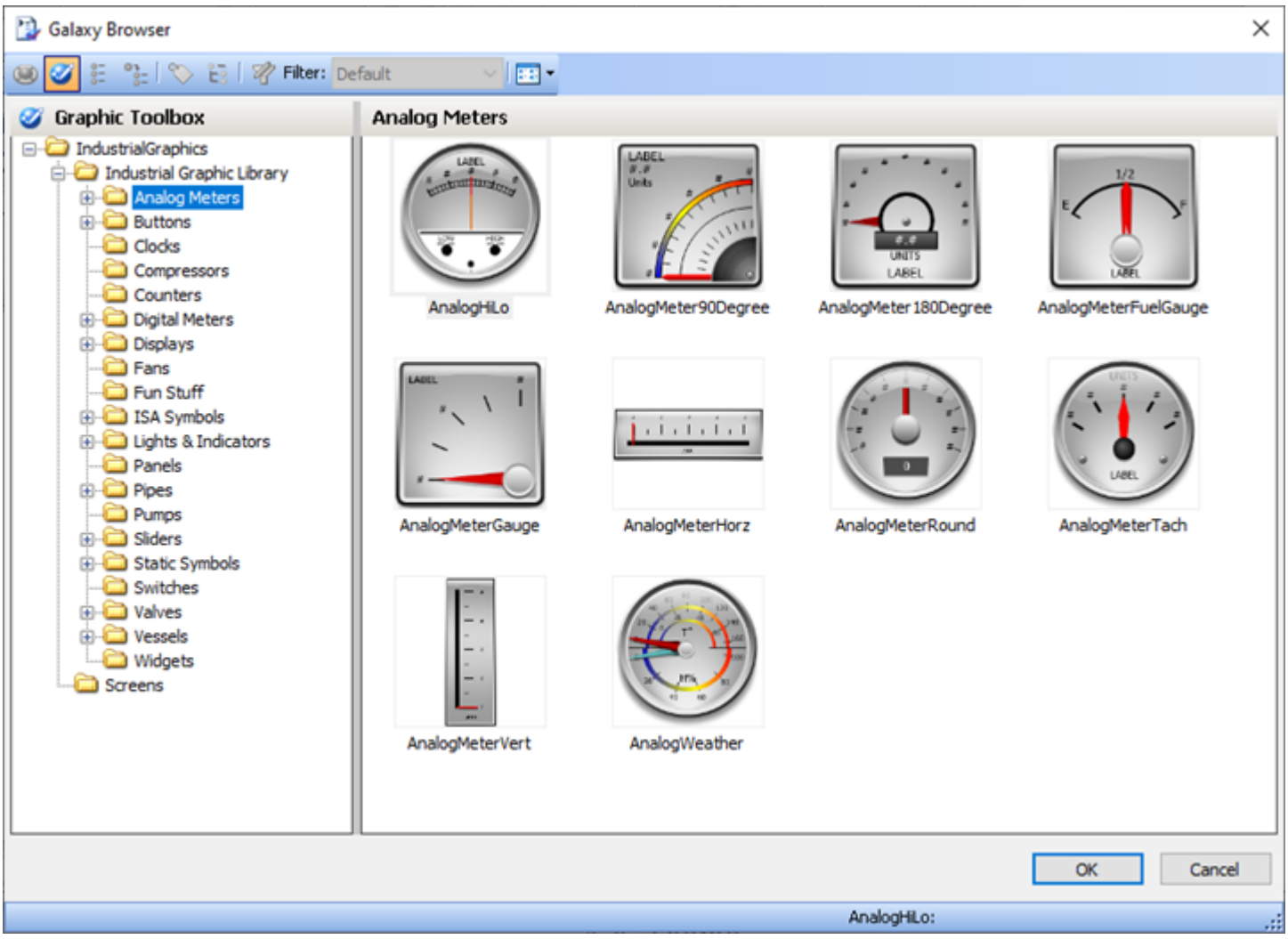
Before you begin this task, you must have already created both an Industrial Graphics screen and an Industrial Graphics symbol. Industrial Graphics symbols cannot be used in native screens, just as native symbols cannot be used in Industrial Graphics screens.

To embed an Industrial Graphics symbol in a screen:

1. Open the Industrial Graphics screen from the *Project Explorer*, and then do one of the following:
   - On the **Draw** tab of the ribbon, in the **Industrial Graphics** group, click **Edit Symbol**;
   - Right-click in the screen, and then on the shortcut menu, select **Edit Symbol**; or
   - Double-click anywhere in the opened screen.

   The Industrial Graphics editor is displayed, and the screen is opened for editing.
2. Do one of the following:
   - Go to **Edit**, and then select **Embed Graphic**; or
   - On the editor's toolbar, click **Embed Graphic**

   The *Galaxy Browser* dialog box is displayed, and your project's own library of toolsets and symbols is listed in the *Graphic Toolbox* pane on the left.




3. In the *Graphic Toolbox* pane, select a toolset (folder).
   The symbols contained in that toolset are displayed in the pane on the right.
4. Select a symbol, and then click **OK**.
   The *Galaxy Browser* dialog box is closed, and then your pointer appears in paste mode.
5. Click anywhere in the open screen.
   The selected symbol is pasted into the screen.

Once the symbol is embedded in the screen, you can edit it as you would edit any other element. For more information about how to use the Industrial Graphics editor, see the help system that is available within the editor itself: in the editor, go to **Help**, and then select **Help Topics**.

## Using project tags in Industrial Graphics screens

You can use your project tags in Industrial Graphics screens and symbols.

You can do this anywhere in the Industrial Graphics editor that you would normally reference an attribute, such as when you configure an animation.

Technically speaking, your project tags belong to the "HMI" namespace in the Industrial Graphics editor. What this means in practice is that if you type `HMI:` first, a list of available tags will automatically pop up for you to select from. This is similar to how IntelliSense works in the VBScript editor. Project and system tags are provided in a combined list.

The "HMI" namespace is the default, however, so you do not need to include the `HMI:` prefix. You can type the full name of the tag you want to use, if you know it.

Only project tags (including arrays) and system tags are supported at this time, and the scope of the project tags must be set to **Server**. Shared tags (i.e., tags added to the project through tag integration), tag properties, classes, and built-in functions are not supported at this time. If you need to access an unsupported item from an Industrial Graphics screen, you can create a Math or Script worksheet that does what you need and then use a project tag to control the execution of that worksheet.

For more information about how to use the Industrial Graphics editor to edit screens and symbols, see the help system that is available within the editor itself: in the editor, go to **Help**, and then select **Help Topics**.

## Known limitations of Industrial Graphics

This is a list of known limitations on how the Industrial Graphics editor can be used to create screens and symbols, and on how Industrial Graphics screens can be used in projects.

### Runtimes

Industrial Graphics screens can be used only in projects that run in the SCADA runtime edition for Windows and Windows Server. None of the other runtime editions are supported at this time.

### Thin Clients

You cannot use Thin Clients to view projects that include Industrial Graphics screens. You can use Studio's built-in Viewer to test your Industrial Graphics screens as you develop your project, but you should not use it during your actual project run time.

In all cases, for both local and remote clients, you must use Mobile Access as your project viewer. Specifically, we recommend you use the Mobile Access add-on for Internet Information Services (IIS), because the Mobile Access add-on for CGI is not supported. For more information about how to set it up, see Mobile Access on page 756.

Industrial Graphics screens are automatically saved as HTML, so you do not need to do that manually as you would for native screens.

### Arrays

You can use project tag arrays in Industrial Graphics screens and symbols. However, if you do not include the `HMI:` prefix when you type the name of an array, a warning message is displayed. For example:

Could not resolve reference: 'tag1[2]'. Save anyway?

You can choose to save the reference and it will work as expected during run time, but you can avoid the message altogether if you include the `HMI:` prefix.

### Element Styles

The default list of Element Styles is included in the Industrial Graphics editor, and you can apply any of those styles to elements. You cannot modify or import styles at this time, however.

### User Input Keypad

Due to how Industrial Graphics screens are displayed in the project viewer, you must make sure any screen that includes a User Input animation is large enough to display the keypad invoked by that animation. If the screen is too small, the keypad will not be displayed correctly and the user might not be able to proceed.

### AlarmClient Graphic

The AlarmClient graphic is not supported at this time. In most cases, this is an issue only if you are using an imported symbol from the Situational Awareness Library.

### Embedding Symbols Within Symbols

You can embed a symbol within another symbol, but if you make changes to the embedded symbol, those changes may not immediately appear in the symbol that contains it. You must save, close, and then reopen all of the affected symbols to make sure the changes propagate correctly.

### Layout Tool

The Layout tool does not support Industrial Graphics screens at this time.

If you want to use the tool to lay out native and Industrial Graphics screens together, we recommend you create additional native screens to serve as placeholders for the Industrial Graphics screens. Then, when you are done laying out the screens, you can copy the Size and Location attributes from the placeholder screens to the Industrial Graphics screens.

### Translation Tool

The Translation tool does not support Industrial Graphics screens at this time, even if you have configured it to translate the native screens in your project.

# Background Tasks

Background tasks are, as the name implies, project features that run in the background, as opposed to the graphical screens with which the user interacts.

The background tasks are executed by the Background Tasks module (see Runtime Tasks), and they are defined by task worksheets in the Project Explorer.

## Alarm worksheet

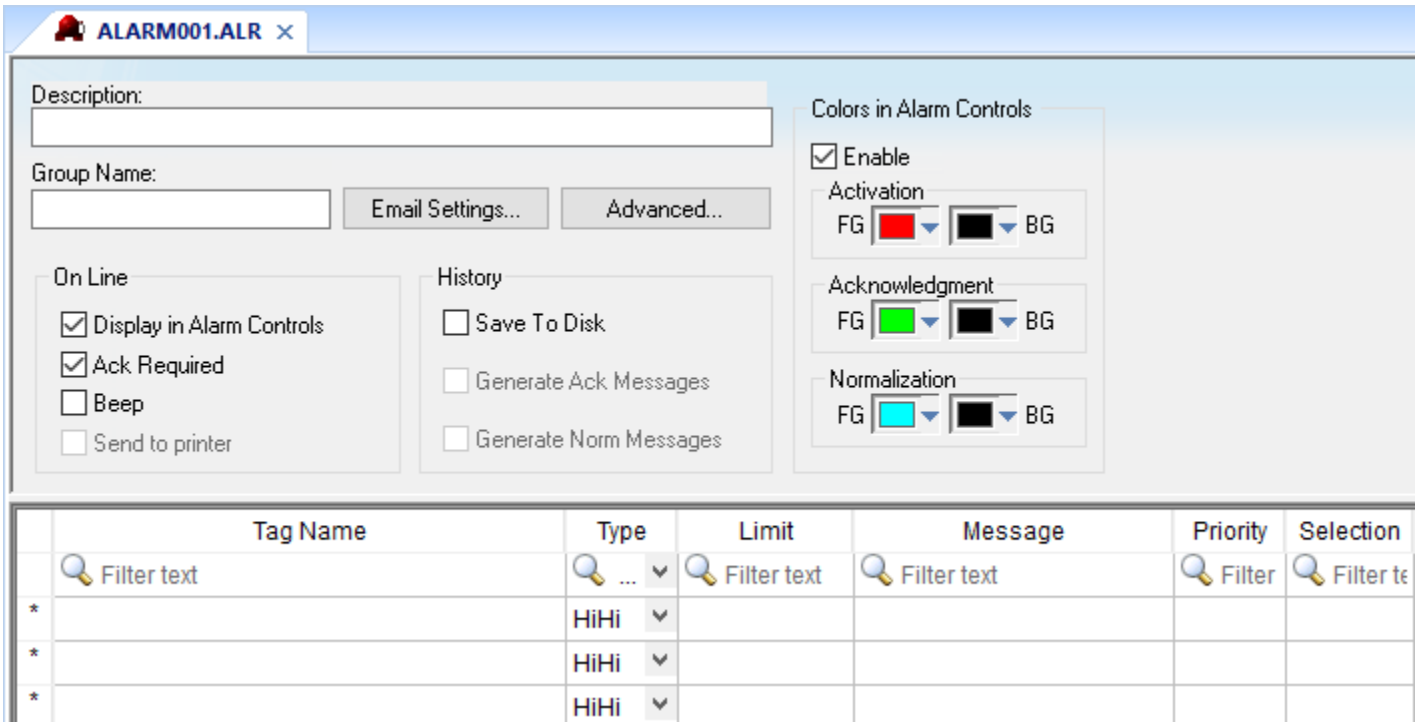
The Alarms folder enables you to configure alarm groups and tags related to each group. The Alarm worksheet defines the alarm messages generated by the project. The primary purpose of an alarm is to inform the operator of any problems or abnormal condition during the process so he can take corrective action(s).

The Alarm worksheet is executed by the Background Task module (see Runtime Tasks on page 136). It handles the status of all alarms and save the alarm messages to the history, if configured to do so, but it does not display the alarm messages to the operator; the Alarm/Event Control screen object, available on the Graphics tab of the ribbon, must be created and configured in a screen in order to display alarms.

To create a new Alarm worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Alarm**;
- Right-click the **Alarms** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the File menu, click the **File** tab, and then select **Alarm Worksheet**.

To edit an existing Alarm worksheet, double-click it in the Project Explorer.




***Alarm worksheet***

You can create multiple Alarm groups (worksheets) and each group can be configured with independent settings, such as message colors, history log enabled/disabled, and so forth.

Each Alarm worksheet is composed of two areas:

- **Header:** Settings applied to all tags and alarms configured in the same alarm group. These settings allow you to configure the formatting of the message and the actions that must be triggered based on alarm events (e.g., print alarms, send alarms by email, and so forth). For more information, see Header Settings.
- **Body:** Configure alarm messages and associate them to conditions linked to tags. For more information, see Body Settings.

> **Note:**

The Alarm task has been modified to avoid automatically acknowledging alarms by another alarm. For example, the Hi (Lo) alarm should not be automatically acknowledged when the HiHi (LoLo) alarm becomes active. To enable the previous behavior, set the following key in your project (.APP) file:

```
[Alarm]
UseLegacyPriorityAck=1
```

## Trend worksheet

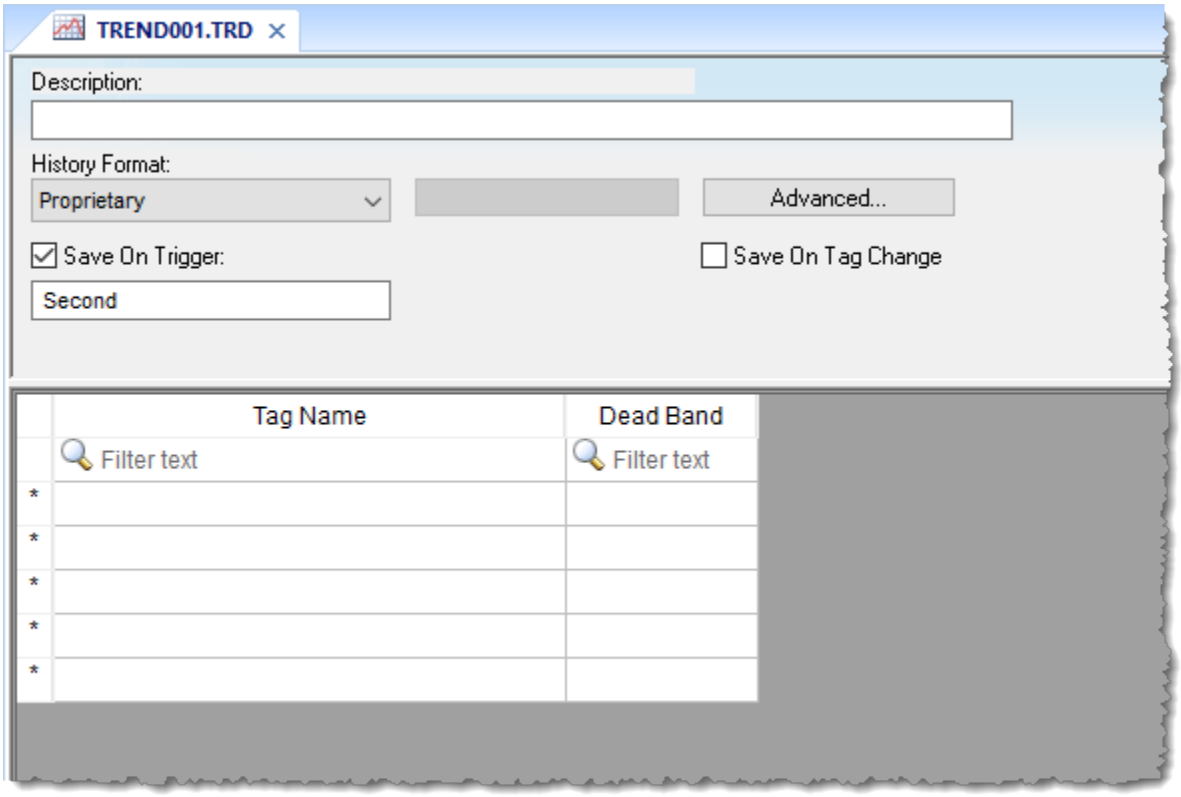
The Trend folder enables you to configure history groups that store trend curves. You can use the Trend worksheet to declare which tags must have their values stored on disk, and to create history files for trend graphs. The project stores the samples in a binary history file (*.hst), and shows both history and on-line samples in a screen trend graph.

The Trend worksheet is executed by the Background Task module (see Runtime Tasks on page 136). It handles the saving of trend data to the history, but it does not display that data to the operator; the Trend Control screen object, available on the Graphics tab of the ribbon, must be created and configured in a screen in order to display trend data.

To create a new Trend worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Trend**;
- Right-click the **Trends** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the File menu, click the **File** tab, and then select **Trend Worksheet**.

To edit an existing Trend worksheet, double-click it in the Project Explorer.




***Trend worksheet***

The Trend worksheet is divided into two areas:

- Header area (top section), which contains information for the entire group
- Body area (bottom section), where you define each project tag in the group. This section contains several columns (only two are shown in the preceding figure).

### Header

Configure the following settings in the worksheet header:

#### Description

Type a description of the worksheet for documentation purposes.

#### History Format

Click the arrow button to select a trend history format from the list. The available options are:

**Proprietary**

Save trend history in a proprietary, binary file. The file is saved in your project folder (on the project runtime server) at: `[…]\<project name>\Hst\GGYYMMDD.hst`

- **GG** = Trend worksheet number
- **YY** = Last two digits of the year
- **MM** = Month
- **DD** = Day

A new history file is created for each calendar day that the project runs.

The utility programs `HST2TXT.EXE` and `TXT2HST.EXE` are provided in order to convert history files from binary ( `*.hst` ) to plain text ( `*.txt` ) and vice versa. For more information, see Converting Trend History Files from Binary to Text on page 421 and Converting Trend History Files from Text to Binary on page 422.

**Database**

Save trend history in an external SQL database of your choice. After you select this format, click **Database Configuration** to open the *Database Configuration* dialog box, where you can configure the connection to the database. For more information, see Database Configuration on page 108 and Database Interface on page 815.

By default, the history is saved in the table **TRENDGGG** ( **GGG** = Trend Worksheet Number; e.g., **TREND001** for the Trend Worksheet 001).

**Historian**

Save trend history to a Historian database or AVEVA Insight. After you select this format, click **Historian Configuration** to open the *Historian* dialog box, where you can configure the connection to the database. The trend history for each project tag is saved separately in the Historian database, but you can use **Prefix** in the database connection settings in order to keep the tags grouped together. For more information, see Support for AVEVA Insight and Historian on page 837.

> **Note:**
>
> You can specify String tags in many fields of the Trend worksheet, to change those values during run time, but doing so may affect how those values are saved in the trend history:
>
> - When the history format is **Proprietary**, the value of the String tag is converted to a numeric value (if possible) and then saved in the history file. If numeric conversion is not possible, then a value of 0 is saved.
> - When the history format is **Database** or **Historian**, the actual value of the String tag is saved in the database.

#### Save On Trigger

Click (enable) and type a tag name to save trend samples when someone changes the specified tag. (Tag change can be an event from the Scheduler.)

#### Save On Tag Change

Click (enable) to always save the trend sample when a value change occurs in any of the tags from that group.

When the history format is **Proprietary** or **Database**, all of the tags in the group are saved after each change. When the history format is **Historian**, only the tag that changed is saved.

#### Advanced

Click to display the Trend Advanced Settings dialog. For information about completing the fields in this window, see Batch History Configuration.

### Body

For each project tag, configure the following settings in the worksheet body:

#### Tag Name

The name of the project tag for which trend history will be saved.

#### Dead Band

Type a value to filter acceptable changes when **Save on Tag Change** is used. For example, Dead Band has value = 5. If the tag value is 50 and changes to 52, the system will not register this variation in the database, because it is less than 5. If the change is equal to or greater than 5, the new value will be saved to the history file.

#### Field

When **History Format** is **Database**, this is the name of the field (in the SQL database table) where the trend history will be saved. If this field is left blank, the project tag name will be used.

For array tags and classes, special characters (`[].`) will be replaced by underscores (_), as shown in the examples below:

| Tag Name | Field Name |
|---|---|
| `MyArray[1]` | `MyArray_1` |
| `MyClass.Member1` | `MyClass_Member1` |
| `MyClass[3].Member2` | `MyClass_3_Member2` |

#### Historian Tag

When **History Format** is **Historian**, this is the name of the tag (in the Historian database) where the trend history will be saved. If this field is left blank, the project tag name will be used.

When you save a Trend worksheet, only the header settings are saved as part of the worksheet file. All of the trend configurations that make up the body of the worksheet are actually saved as tag properties. The next time you open that worksheet, the tags database is scanned for all trend configurations that belong to the worksheet (i.e., the trend group), and then that information is used to recreate the body of the worksheet. This happens quickly and automatically every time you open the worksheet, so it might seem like you are opening a static file but that is not the case.

You may think of the Trend worksheet as an editor for those tag properties that are related to history. If you use either the *Tag Properties* dialog box or the `TagsDBSetTrend` function to edit the same properties, the updated trend configurations will be included in the body of the worksheet the next time you open it. In fact, you can set trends on tags before you create any Trend worksheets at all; when you do create the worksheets, they will be automatically populated with trend configurations according to their group numbers.

You cannot configure more than one trend on a given tag, and each trend configuration cannot belong to more than one group/worksheet.

If you make extensive changes to the tags database after you save an Trend worksheet, it might not be possible to recreate the body of the worksheet the next time you open it. For example, if you copy all of the tags from the tags database (in Datasheet View) to a spreadsheet program, use that program to sort the tags, and then copy the tags back to the tags database, most or all of the tag properties will be reset in the process.

> **Note:** The Trend task can accept only up to 1000 tags in a single worksheet. If you manually configure more than 1000 tags in the same worksheet, the Trend task will generate an error during project run time.

## Recipes

Use a Recipe worksheet to load tag values from and/or save tag values to an external data file during project run time. It is typically used to execute process recipes that comprise many predefined settings, but you can also use it to take snapshots of the project state or store other types of data.

The external data file can be one of two file types: a standard .xml file or a space-separated .dat file. Each type of file has its own benefits and limitations. The .xml file stores the data in an easy-to-read XML format that can be processed by other programs or viewed in a web browser. (An .xsl file is saved and associated with the .xml file, and the web browser uses that .xsl file to style the data as a web page.) Furthermore, the .xml file can handle large arrays of values without issues. The tag names are saved with the tag values, however, so the values can only be loaded back into the same tags.

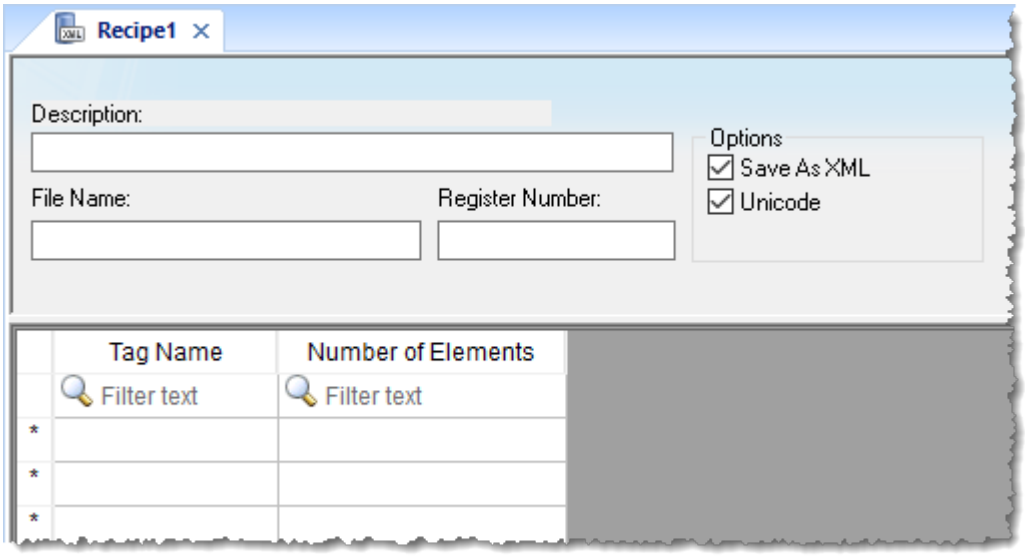
In contrast, the .dat file stores the tag values as raw data, without tag names. (An .rcp file that contains the recipe configuration is saved and associated with the .dat file.) That means you can configure one Recipe worksheet to save tag values to the file and then configure another Recipe worksheet to load the saved values into different tags. You must be careful about how the project tags are ordered in their respective worksheets, however, or else the worksheets will conflict with each other over how they parse the data. Furthermore, because of how the raw data is saved line-by-line, the .dat file cannot handle large arrays of values.

In both cases, the files for all recipes should located in the Web sub-folder of your project folder (e.g., `<project name>\Web\Recipes`).

To create a new Recipe worksheet:

1. Do one of the following:
   - On the **Insert** tab of the ribbon, in the **Task Worksheets** group, click **Recipe**;
   - Right-click the **Recipes** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
   - Click **New** on the File menu, click the **File** tab, and then select **Recipe Worksheet**.

   A new Recipe worksheet is opened for editing in the Screen/Worksheet Editor.




***Recipe worksheet***

2. In the **Description** box, type a description of the recipe.
   This is for documentation purposes only and does not affect the execution of the worksheet.
3. In the **File Name** box, type the name of the external file that will store the data.
   You can type either a specific file name (e.g., `Recipe001`) or the name of a project tag enclosed in curly brackets (e.g., `{MyFileName}`), so that the file name can be programmatically changed during project run time. Do not include the file extension (.xml or .dat), because that will be automatically determined by whether the **Save As XML** option is selected.

4. In the **Register Number** box, type a tag to define the register number to be read from or written to a database file.

   > **Note:** This setting is for legacy purposes only, and it should not be used in a new Recipe worksheet.

5. Select the **Save As XML** option to save the data to a standard .xml file, or clear the option to save the data to a space-separated .dat file.

   This option is selected by default.

6. Select the **Unicode** option to save the data in Unicode format (two bytes per character), or clear the option to save the data in ANSI format (one byte per character).

   This option is selected by default.

7. In the body of the worksheet, configure a row for each project tag that you want to include in the recipe:

   a) In the **Tag Name** column, type the name of the project tag.

      If the tag is an array, a class, or both, then all of its array elements and class members are included by default. To include only a specific element and/or member, type the full name including array position and/or member name (e.g., `MyArray[3].MyMember`).

   b) In the **Number of Elements** column, type the number of elements that you want to include from the specified tag.

      This is starting from the array position that you specified in the **Tag Name** column, or from position 0 if you did not specify a position. You can type either a literal value (e.g., `10`) or the name of a project tag enclosed in curly brackets (e.g., `{MyNumberOfElements}`), so that the number of elements can be programmatically changed during project run time.

8. When you are done, save and close the worksheet.

To execute a Recipe worksheet during project run time, call the `Recipe` function. Unless you are using an existing data file that has been copied from another project, you must call the function at least once to create a new data file and save the initial values of the included tags.

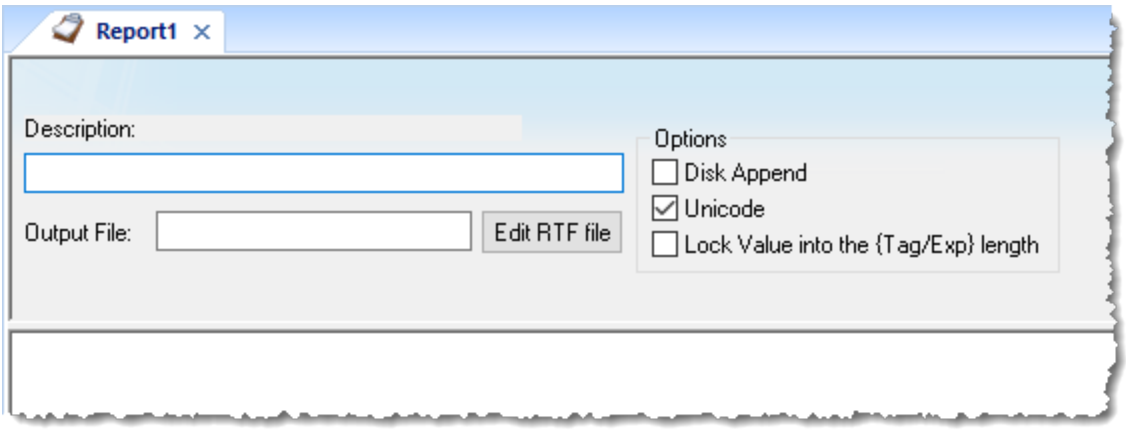
## Report worksheet

A Report worksheet is used to design a report that is dynamically generated during runtime (using the current values of the included tags) and then either sent to a printer or saved to a file.

To create a new Report worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Report**;
- Right-click the **Reports** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the File menu, click the **File** tab, and then select **Report Worksheet**.

To edit an existing Report worksheet, double-click it in the Project Explorer.




***Report worksheet***

The *Report* worksheet is divided into two areas:

- *Header* area (top section), which contains information for the whole group; and
- *Body* area (bottom section), where you define each tag in the group.

Use the Header parameters on this worksheet as follows:

- **Description** field: Type a description of the worksheet for documentation purposes.
- **Output File** field: Type a tag name for the output file (using the **`{tag}`** syntax) where data is stored when you are printing to a file. Where the tag value is part of the file name.

  For example: **`report{Day}.out`**. Where the generated file might be **`report1.out`**, **`report2.out`**, **`report3.out`**, and so on, according to the tag day value.

  > **Note:** A report configuration file uses **`.RCP`** as the default extension. The **Output File** field is the file where data is stored.

- **Edit RTF file** button: Click to access the report as an RTF file, which you can edit for layout modification and so forth.
- **Disk Append** checkbox: When printing to a file
  - Check the box to add (amend) the new report to the end of an existing file
  - Uncheck the box to replace the existing report in that file with the new report
- **Unicode** checkbox: Click (enable) to save the report in Unicode format (two bytes per character) or (disable) to save the report in ASCII format (one byte per character).
- **Lock Value into the {Tag/Exp} length** checkbox: Click (enable) to automatically truncate the values of Tags/Expressions in the report to fit between the curly brackets, as they are positioned in the Body of the report

(see below). This helps to preserve the layout of the report. If this option is left unchecked, the full values of Tags/Expressions in the report will be displayed.

Use the *Body* portion of this worksheet for report formatting. You can configure a report using data in the system and indicating where to print the tag values. Each tag name will replace the **{*tag_name*}** tag name. For Real type tags, use the following syntax: **{*tag_name n*}**, where ***n*** is the number of decimal places you want printed.

If you are using the standard report editor (text only: ASCII or Unicode), the number of characters reserved for the tag value will be equal to the number of characters used to type the tag name (including the two "curly" brackets). For example, if you configure **{TagA}** in the report body, reserve six characters for the tag value in the report file. This behavior is not valid for reports in RTF format.

To execute a Report worksheet, use the `Report` function anywhere an expression is allowed.

**Note:** After you create and edit a Report worksheet, you can save it with a custom name. The name should not contain spaces, however, because if it does, the `Report` function will not be able to execute it.

## Create a new Math worksheet

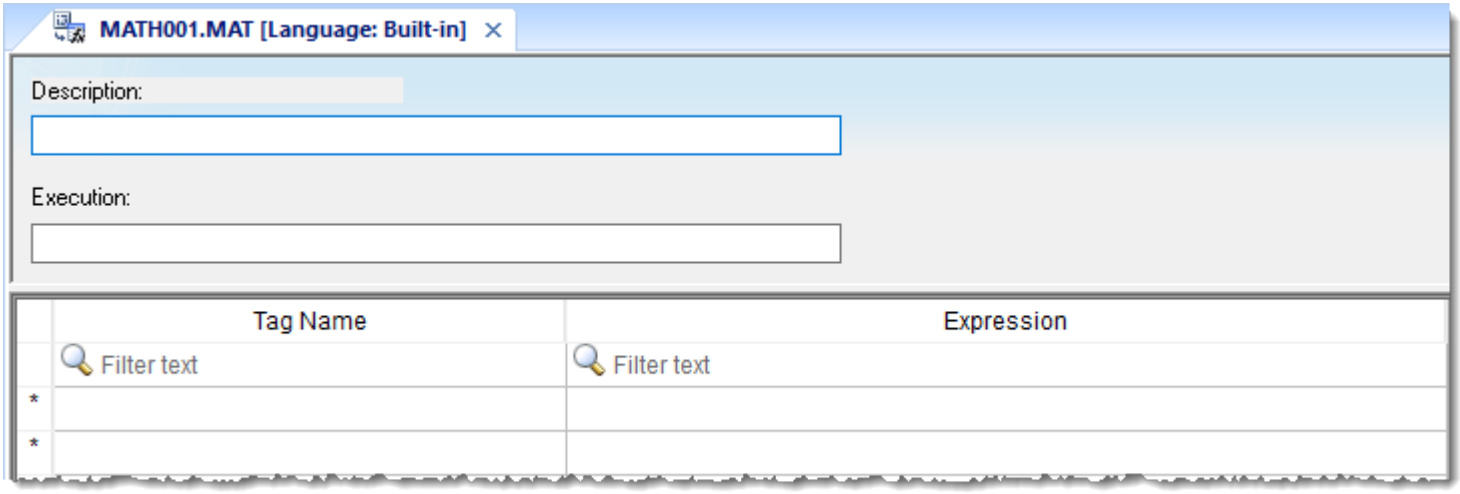
Create a Math worksheet to implement program logic that should be executed periodically in the background during project run time, rather than on a specific action or event like opening a screen or clicking a button.

This worksheet is functionally similar to the Script worksheet, except that it uses the built-in scripting language instead of VBScript.

To create a new Math worksheet:

1. Do one of the following:
   - On the **Insert** tab of the ribbon, in the **Task Worksheets** group, click **Math**;
   - On the **Tasks** tab of the Project Explorer, right-click the **Math** folder, and then click **Insert** on the shortcut menu; or
   - Click **New** on the File menu, click the **File** tab, select **Math Worksheet**, and then click **OK**.

   A new Math worksheet is opened for editing in the Screen/Worksheet Editor.




***Math worksheet***

2. In the **Description** box, type a description of the worksheet.

   This is for documentation purposes only and does not affect the execution of the worksheet.

3. In the **Execution** box, type a tag/expression that will control the execution of the worksheet during project run time.

   The worksheet is scanned periodically as part of the Background Task module. Each time the worksheet is scanned, if the specified tag/expression evaluates as TRUE (i.e., non-zero), then the worksheet is executed. In practice, this means the worksheet is executed continuously while the tag/expression evaluates as TRUE.

   To ensure the tag/expression always evaluates as TRUE, so that the worksheet is executed continuously without exception or interruption, type a literal value of 1.

4. Complete the body of the worksheet, which uses the Built-in Language interface.

5. Save and close the worksheet.

   When you save a new worksheet for the first time, you will be prompted to assign it a unique sheet number. This number determines the order in which the worksheet will be scanned relative to all other worksheets of the same type. For example, if you assign a new Math worksheet the sheet number 5, it will be scanned after sheet numbers 1–4 and before sheet numbers 6–*n*.

The completed Math worksheet is saved in the Math folder in the Project Explorer.

To open an existing Math worksheet for further editing, double-click it in the Project Explorer.

You can also execute a Math worksheet on demand, independent of whatever is configured in the **Execution** box of that worksheet, by calling the `Math` function anywhere an expression is allowed. Using this method, you can even call other Math worksheets like sub-routines from within a Math worksheet.

### *About the Built-in Language interface*

The Built-in Language interface is a standard tool for developing program logic that uses our Built-in Language. It is offered as an alternative to the VBScript interface.

The Built-in Language interface is available in the following places in the project development environment:

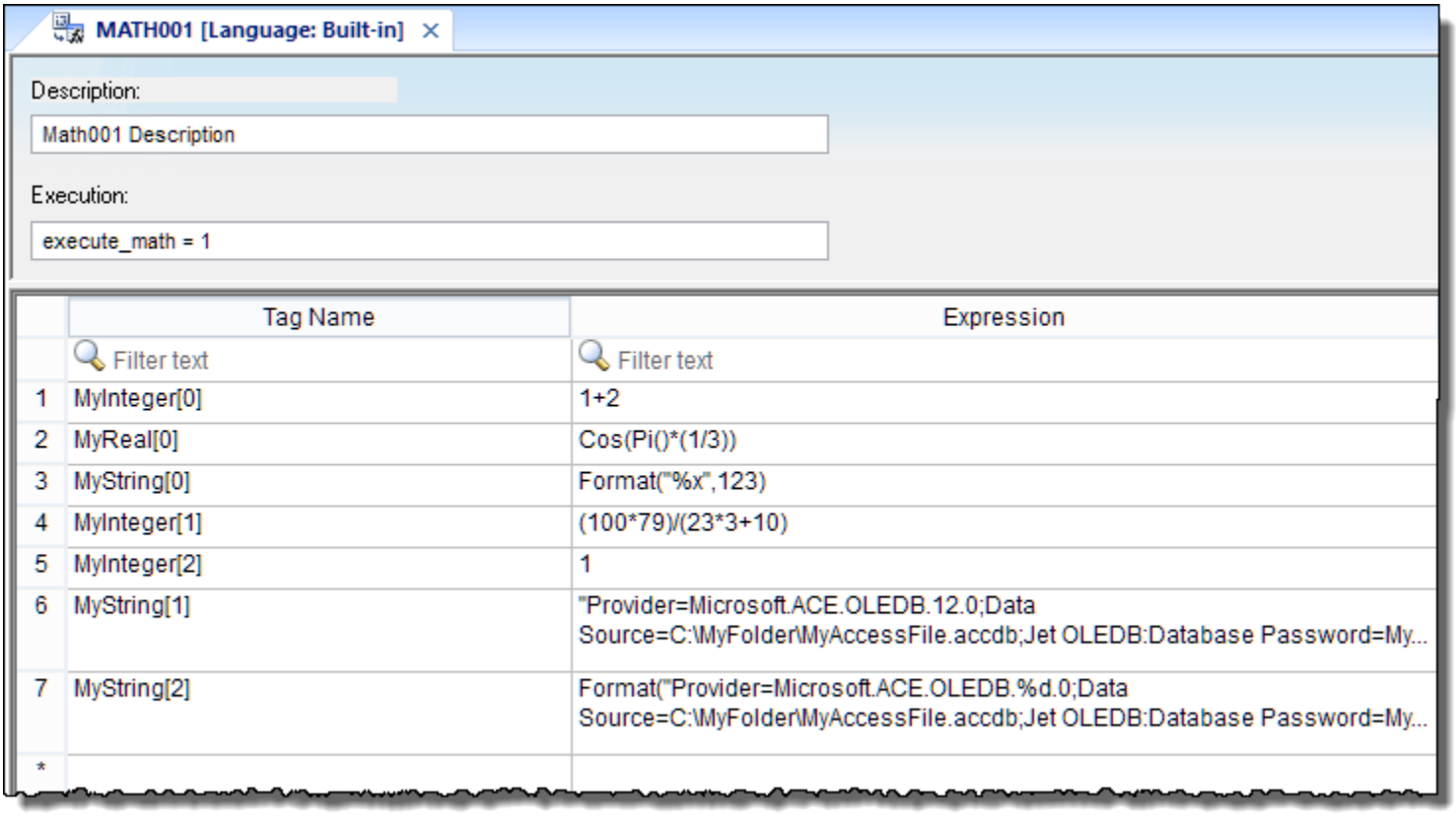
| Interface | Is Executed… |
|---|---|
| Math worksheet | …periodically, while the worksheet's execution condition is TRUE. |
| Scheduler worksheet | …at a specified date/time, at a specified time interval, or when the value of a specified tag changes. |
| Screen Logic (in Screen Attributes) | …when the screen is opened, while the screen is open, and/or when the screen is closed. |
| ActiveX or .NET Control object | …when a selected event on that object is triggered. |
| Command animation | …when the underlying screen object (typically a Button object) is clicked. |

In some of these places, you will be prompted to select the type of interface. Remember to select **Built-in** or **Built-in Language**, as opposed to **VBScript** or others.

The interface itself is arranged as a two-column table. Each row of the table is essentially a line of code, and the rows are executed sequentially from first to last.

When a row is executed, the expression in the **Expression** column is evaluated, and if it returns a value, that value is written to the project tag in the **Tag Name** column.

An expression can be almost anything, from a simple arithmetic or logic operation to a series of nested function calls, as long as it follows the syntax for our Built-in Language. The following example shows a variety of expressions.




***A variety of expressions in a Math worksheet***

| Row | Tag Name | Expression | Description |
|---|---|---|---|
| 1 | `MyInteger[0]` | `1+2` | A simple arithmetic operation that returns an integer value. The value is written to `MyInteger[0]`. |
| 2 | `MyReal[0]` | `Cos(Pi()*(1/3))` | A trigonometric expression, using the `Cos` and `Pi` functions, that returns a real value. The value is written to `MyReal[0]`. |
| 3 | `MyString[0]` | `Format("%x",123)` | An integer value (123) that is reformatted as a string using the `Format` function. The string is written to `MyString[0]`. |
| 4 | `MyInteger[1]` | `(100*79)/(23*3+10)` | A more complex arithmetic expression that uses parentheses to change the order of operations. The resulting value is written to `MyInteger[1]`. |
| 5 | `MyInteger[2]` | `1` | A literal value of 1 that is written to `MyInteger[2]`. |
| 6 | `MyString[1]` | `"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\MyFolder\MyAccessFile.accdb;Jet OLEDB:Database Password=MyDbPassword;"` | A database connection string that is written to `MyString[1]`. |
| 7 | `MyString[2]` | `Format("Provider=Microsoft.ACE.OLEDB.%d.0;Data Source=C:\MyFolder\MyAccessFile.accdb;Jet` | The same database connection string as above, except that the `Format` function is used to get the value of `DBVersion` and then insert it into the string. The returned value — that, is new string — is written to `MyString[2]`. |

| Row | Tag Name | Expression | Description |
|---|---|---|---|
| | | OLEDB:Database Password=MyDbPassword;",DBVersion) | |

For a list of available functions and a complete description of each function, see Appendix: Built-in Language on page 896.

In the example above, there is no relationship between the expressions; each row can be executed and each expression can be evaluated without affecting any of the others. The rows of the Built-in Language interface are executed sequentially, however, so you can also configure expressions that progress from one row to the next. Specifically, an expression can use the values of project tags that were set in previous rows. The example below shows how to populate the elements of an array so that the value of each element is incrementally greater than the previous one.

| | Tag Name | Expression |
|---|---|---|
| | Filter text | Filter text |
| 1 | tag[1] | 1 |
| 2 | tag[2] | tag[1]+1 |
| 3 | tag[3] | tag[2]+1 |
| 4 | tag[4] | tag[3]+1 |
| 5 | tag[5] | tag[4]+1 |
| 6 | tag[6] | tag[5]+1 |
| 7 | tag[7] | tag[6]+1 |
| 8 | tag[8] | tag[7]+1 |
| 9 | tag[9] | tag[8]+1 |
| 10 | tag[10] | tag[9]+1 |
| 11 | tag[11] | tag[10]+1 |

***Configuring expressions that use tags set in previous rows***

Finally, like in traditional code, you can use comments and empty fields to organize the content of the Built-in Language interface. The following example shows a few different methods for doing this.

| | Tag Name | Expression |
|---|---|---|
| | Filter text | Filter text |
| 1 | execute_math | tag[0] |
| 2 | | // comment |
| 3 | tag[1] | (Rand()+0.01)*100 |
| 4 | | |
| 5 | | 150 |
| 6 | tag[2] | |
| 7 | | |
| 8 | tag[3] | 123+456 // comment |

***Using comments and empty fields to organize the content***

A comment can be placed in a row by itself or appended to an expression. The double-slash (//) indicates that everything following it should not be evaluated.

Empty fields are not evaluated at all, which means the following:

- If the **Tag Name** column is empty and the **Expression** column contains an expression, the expression will be evaluated but the returned value (if any) will be discarded.
- If the **Tag Name** column contains a project tag and the **Expression** column is empty, the value of the project tag will remain unchanged.
- If the **Tag Name** and **Expression** columns are both empty, the entire row serves only as blank space to separate other groups of rows.

### *Using the Goto…Label structure in a Math worksheet*

You can use the Goto and Label system tags to create a Goto…Label structure, which controls the flow of program logic in a Math worksheet.

> **Note:** The Goto…Label structure can be used only in Math worksheets. It cannot be used in any other Built-in Language interface, such as an ActiveX or .NET Control object, a Command animation, or Screen Logic (in Screen Properties).

The rows of a Math worksheet are executed sequentially from first to last. If a Goto tag is encountered, however, it causes the execution to go to the matching Label tag in the same worksheet. "Matching" in this case means the Goto and Label tags have the same value at the moment when the Goto tag is encountered.

The Goto and Label tags are somewhat different from other system tags. Their names are reserved by the system, so that you cannot use the same names for your project tags, but their values are not defined by the system. Instead, you define their values when you configure them in a Math worksheet.

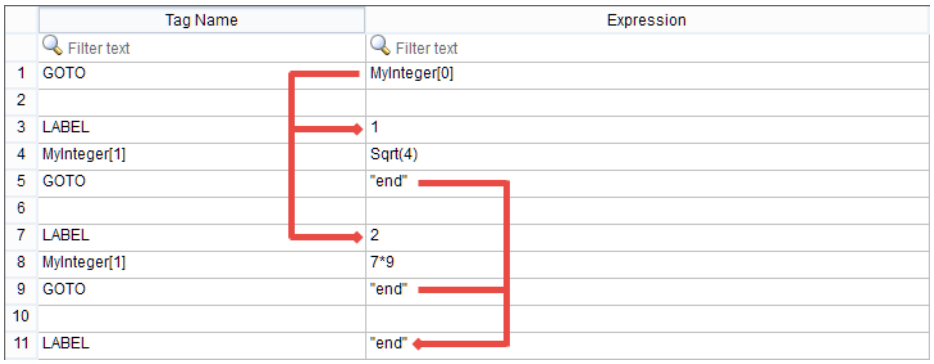
In their most basic usage, the Goto and Label tags can have the same literal value (either numerical or string). In the example below, the execution encounters a Goto tag in the first row, evaluates the corresponding expression to get the literal value **`"end"`**, and then goes to the matching Label tag in the last row.

| | Tag Name | Expression |
|---|---|---|
| | Filter text | Filter text |
| 1 | GOTO | "end" |
| 2 | | |
| 3 | LABEL | 1 |
| 4 | MyInteger[1] | Sqrt(4) |
| 5 | GOTO | "end" |
| 6 | | |
| 7 | LABEL | 2 |
| 8 | MyInteger[1] | 7*9 |
| 9 | GOTO | "end" |
| 10 | | |
| 11 | LABEL | "end" |

***GOTO goes directly to the end of the worksheet***

Of course, if the execution goes directly from the first row to the last row, all of the rows in between are ignored and the worksheet serves no real purpose. The key to the Goto…Label structure is to configure the expression for the Goto tag so that it performs some sort of program logic. In the next example, the first Goto

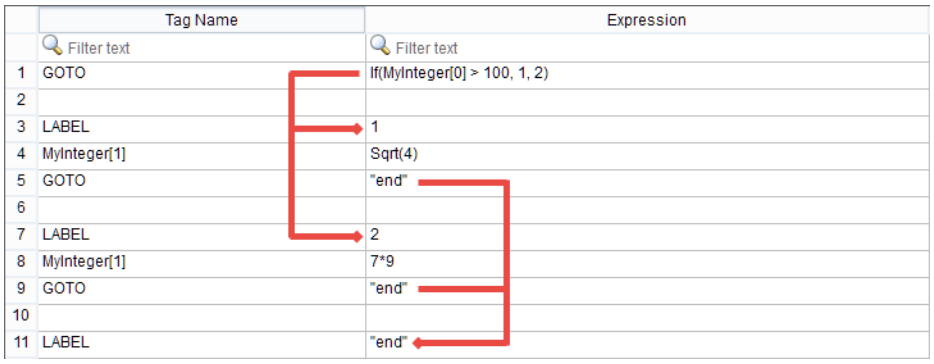
tag is configured to get the value of a project tag (e.g., **`MyInteger[0]`**), and then the execution goes to the Label tag that has the same value (e.g., either **`1`** or **`2`**).



| | Tag Name | Expression |
|---|---|---|
| | Filter text | Filter text |
| 1 | GOTO | MyInteger[0] |
| 2 | | |
| 3 | LABEL | 1 |
| 4 | MyInteger[1] | Sqrt(4) |
| 5 | GOTO | "end" |
| 6 | | |
| 7 | LABEL | 2 |
| 8 | MyInteger[1] | 7*9 |
| 9 | GOTO | "end" |
| 10 | | |
| 11 | LABEL | "end" |

***GOTO gets the value of a project tag***

From there, you can develop the expression further to perform more complex program logic. In the final example, the `If` function is used to not only get the value of a project tag but also test it for some condition.



| | Tag Name | Expression |
|---|---|---|
| | Filter text | Filter text |
| 1 | GOTO | If(MyInteger[0] > 100, 1, 2) |
| 2 | | |
| 3 | LABEL | 1 |
| 4 | MyInteger[1] | Sqrt(4) |
| 5 | GOTO | "end" |
| 6 | | |
| 7 | LABEL | 2 |
| 8 | MyInteger[1] | 7*9 |
| 9 | GOTO | "end" |
| 10 | | |
| 11 | LABEL | "end" |

***GOTO gets the value of a project tag and then tests it***

In this way, using the Goto…Label structure, you can configure multiple sub-routines in a single Math worksheet and then use program logic to go from one to the next.

Please be aware that if a Goto tag does not have a matching Label tag — that is, if the execution encounters a Goto tag but cannot find a Label tag that has the same value — then a message will be sent to the project runtime log and the execution will continue to the next row after the Goto tag. You should thoroughly test each Goto…Label structure to make sure there is a matching Label tag for every possible value of the Goto tag.

> **Note:** Technically, the data type of the Goto and Label tags is String, which means they can only receive string values. If a Goto or Label tag receives a numerical value — either a literal value or a value returned by evaluating a tag/expression — that value is converted to a string value before it is actually stored in the tag. This conversion should have no practical effect on the run-time performance of your project, however, because the Goto and Label tags should not be used in any other context.

### *Using the For…Next loop in a Math worksheet*

You can use the `For` function and the Next tag to create a For…Next loop, which repeatedly executes some part of a Math worksheet.

> **Note:** The For…Next loop can be used only in Math worksheets. It cannot be used in any other Built-in Language interface, such as an ActiveX or .NET Control object, a Command animation, or Screen Logic (in Screen Properties).

The rows of a Math worksheet are executed sequentially from first to last. If a `For` function is encountered, however, it causes the execution of certain rows — specifically, the rows between the `For` function and its associated Next tag — to be repeated a specified number of times. When the Next tag is encountered, the execution returns to the `For` function at the start of the loop.

The Next tag is somewhat different from other system tags. Its name is reserved by the system, so that you cannot use the same name for one of your project tags, but its value is not defined by the system. In fact, the Next tag cannot receive any value at all. It exists only to be the end of the For…Next loop.

In the example below, a simple For…Next loop populates the elements of an array with values (e.g., multiples of 10):

| | Tag Name | Expression |
|---|---|---|
| | Filter text | Filter text |
| 1 | Index | For(0, MyArray->Size, 1) |
| 2 | MyArray[Index] | Index * 10 |
| 3 | Next | |

***A single For…Next loop that sets the values of array elements***

Using the Size property of the array ensures the loop will repeat for every element in the array, regardless of the size of the array. You could also specify a literal value for the second parameter of the `For` function in order to limit the number of times the loop will repeat. In either case, remember the array elements are numbered starting from position 0; for example, **`For(0,9,1)`** will cause the loop to repeat ten times for array positions 0–9.

In the next example, each element of the array is also a class with multiple members, so a second For…Next loop is nested inside the first in order to populate all of the class members with values:

| | Tag Name | Expression |
|---|---|---|
| | Filter text | Filter text |
| 1 | NumberOfMembers | TagsDBGetClassMemberCount("cTest") |
| 2 | Index | For(1, MyArray->Size, 1) |
| 3 | Index2 | For(1, NumberOfMembers, 1) |
| 4 | MemberName | "MyArray[" + Index + "].M" + Index2 |
| 5 | @MemberName | Index * Index2 * 100 |
| 6 | Next | // End of second loop (start at Index2) |
| 7 | Next | // End of first loop (start at Index) |

***Nested For…Next loops that set the values of class members***

The `TagsDBGetClassMemberCount` function gets the number of members in the class, just as the Size property gets the number of elements in the array, and this ensures the second, nested loop will repeat for every class member of every array element. Also, the second loop uses an indirect tag in order to compose the name of the class member and then set the value of that member.

Finally, in the last example, a Goto…Label structure is inserted in order to provide an escape from the For… Next loop(s), such as if the value of the most recently set class member exceeds a specified limit:

| | Tag Name | Expression |
|---|---|---|
| | Filter text | Filter text |
| 1 | NumberOfMembers | TagsDBGetClassMemberCount("cTest") |
| 2 | Index | For(1, MyArray->Size, 1) |
| 3 | Index2 | For(1, NumberOfMembers, 1) |
| 4 | MemberName | "MyArray[" + Index + "].M" + Index2 |
| 5 | @MemberName | Index * Index2 * 100 |
| 6 | Goto | If(@MemberName > 999999999, "end", "continue") |
| 7 | Label | "continue" |
| 8 | Next | // End of second loop (start at Index2) |
| 9 | Next | // End of first loop (start at Index) |
| 10 | Label | "end" |

***A Goto…Label structure inserted into the For…Next loops***

For more information about the `For` function and its syntax, see For…Next on page 1037.

## Script worksheet

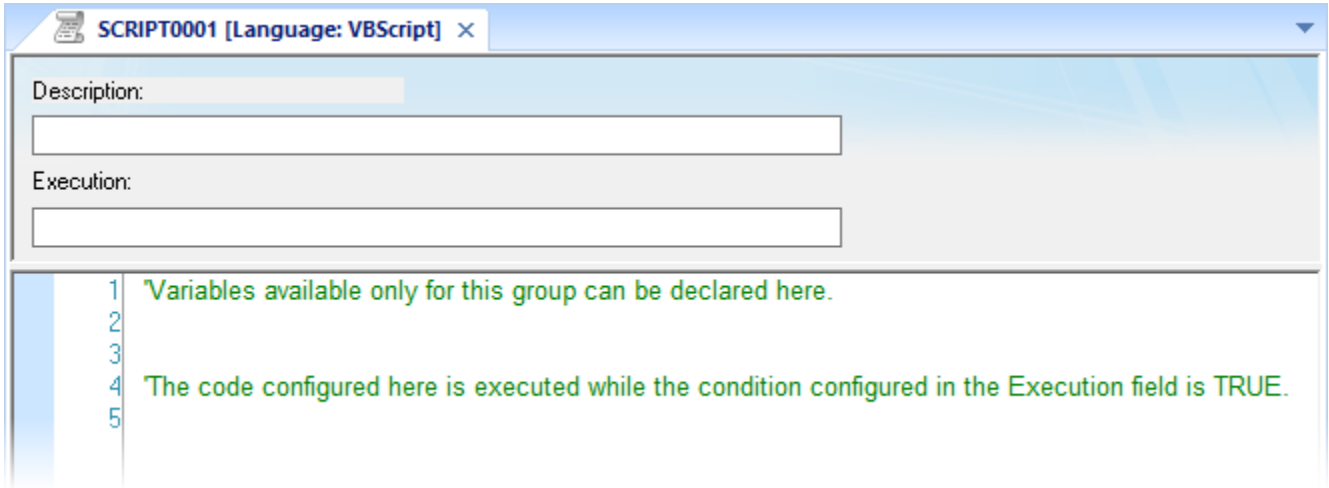
A Script worksheet is used to implement program logic (using VBScript) that should be continuously executed during runtime, rather than on specific actions like the user pressing a button on a screen.

> **Note:** The Script worksheet is functionally similar to the Math worksheet, except that it uses VBScript instead of the Built-in Scripting Language.

To create a new Script worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Script**;
- Right-click the **Script** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the File menu, click the **File** tab, and then select **Script Worksheet**.

To edit an existing Script worksheet, double-click it in the Project Explorer.




***Script worksheet***

The code configured in each Script worksheet is executed by the Background Task. The project scans the worksheets sequentially (based on the worksheet number) and executes only the groups in which the condition configured in the **Execution** field of the worksheet is TRUE (i.e., non-zero).

> **Note:** You must use the syntax supported by the Built-in Scripting Language in the **Execution** field. Only the body of the worksheet supports VBScript.

Variables declared in the worksheet have local scope for that specific group only. They are not available for any other VBScript interface.

You cannot define procedures (i.e., functions and subs) in the Script worksheet. However, you can call procedures defined in the Global Procedures or in the Startup Script.

Example:

```
'Variables available only for this group can be declared here
Dim myVar, myTest
myTest = 1

'The code configured here is executed while the condition configured in the Execution
 field is TRUE
myVar = $FindFile("c:\*.txt")
If MyVar > 0 Then
  $TagNumOfFiles = myVar
```

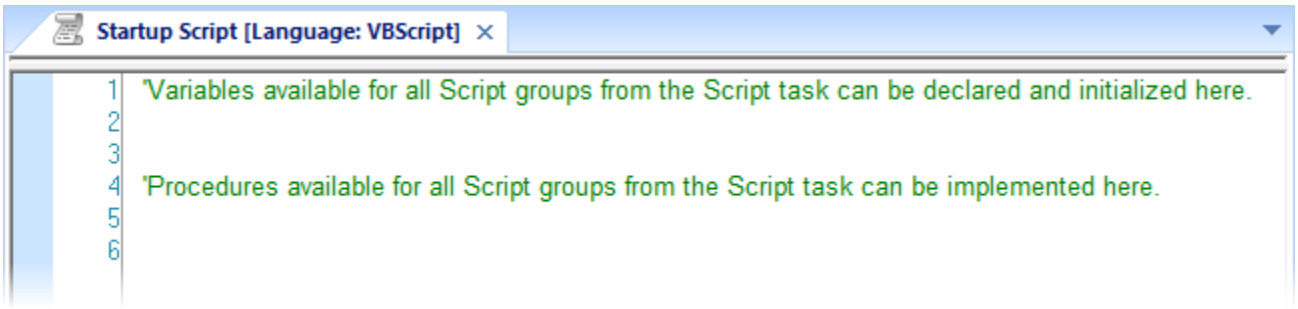
```
End If
```

> **Note:** When any Script worksheet is saved during runtime (on-line configuration), the Startup Script will be executed again and the current value of the local variables of any Script worksheet will be reset.

### *Startup Script worksheet*

The Startup Script worksheet is a VBScript interface that is automatically executed when the project is run.

To edit the Startup Script worksheet, double-click it in the Project Explorer. (It is located on the Tasks tab, in the Script folder.) The worksheet is displayed:




***Startup Script worksheet***

The code configured in this worksheet is executed just once when the Background Task module (BGTask) is started. This interface is useful for initializing variables or executing logics that must be implemented when the project is run.

You can declare and initialize variables and define procedures. However, variables or procedures declared in this interface will be available ONLY to the Script worksheets executed by the Background Task module — they are not available to any VBScript interface from the Graphic Module.

Example:

```
'Variables available for all Script groups from the Script task can be declared and
 initialized here
Dim MyVar, Counter
MyVar = 100

'Procedures available for all Script groups from the Script task can be implemented
 here

Function AreaEquTriangle(base, high)
  AreaEquTriangle = (base * high) / 2
End Function

Sub CheckLimits(myValue, myHiLimit, myLoLimit)
  If (myValue > myHiLimit Or myValue < myLoLimit) Then
    MsgBox("Value out of range")
  End If
End Sub

'The code configured here is executed just once when the Background task is started
If $GetOS() = 3 Then
  MsgBox ("Welcome! This project is running under Microsoft Windows Embedded operating
 system.")
Else
  MsgBox("Welcome! This project Is running under Microsoft Windows desktop operating
 system.")
End If
```

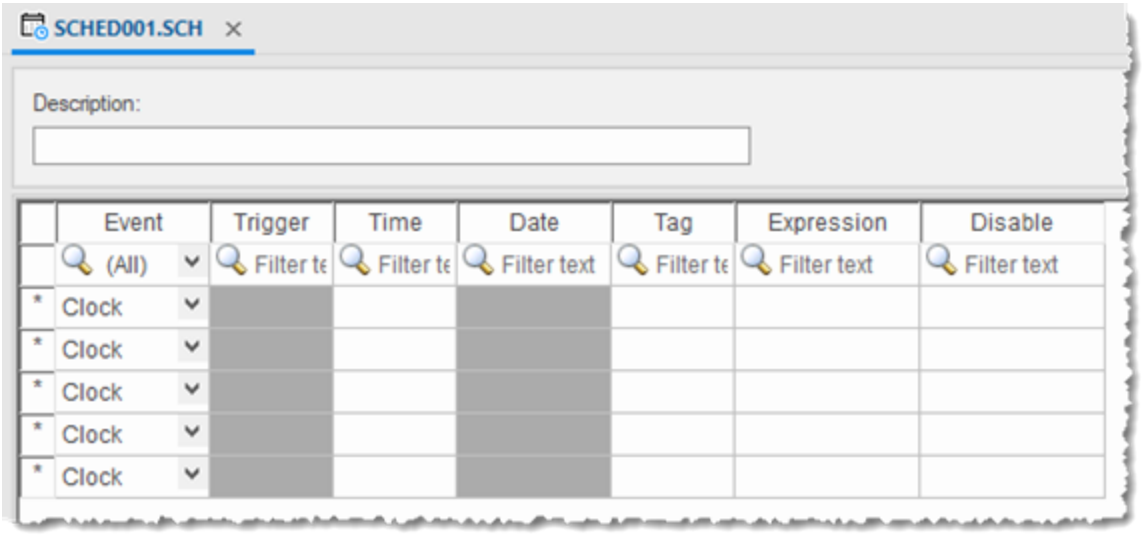
## Scheduler worksheet

A Scheduler worksheet is used to execute program logic (using the Built-in Scripting Language) at a specific date/time, on a regular time interval, or upon a triggering event.

To create a new Scheduler worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Scheduler**;
- Right-click the **Scheduler** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the File menu, click the **File** tab, and then select **Scheduler Worksheet**.

To edit an existing Scheduler worksheet, double-click it in the Project Explorer.




***Scheduler worksheet***

The *Scheduler* worksheet is divided into two areas:

- **Header** area (top section), which contains information for the whole group
- **Body** area (bottom section), where you define each tag, expression, and condition for the group.

Use the parameters on this worksheet as follows:

- **Description** field: Type a description of the worksheet for documentation purposes.
- **Event** drop-down list: Click to select an event type from the following:
  - **Calendar**: Generates time bases greater than 24 hours. For example, You can define an event that prints a report every Friday at a specific time.

    > **Note:** Be sure to complete the **Date** field if you want a specific date for event execution.

  - **Clock**: Generates time bases smaller than 24 hours (intervals in minutes or seconds). This function is frequently used with trend graphics. For example, you can define a tag that will be incremented each hour.
  - **Change**: Event related to the change of a tag in the **Trigger** field.

    > **Note:** This only works for tag changes on the project server, regardless of a tag's defined scope.

- **Trigger** field: This field is used only with the **Change** Event type. Type the name of a project tag in this field, and when the value of the tag changes, the specified **Expression** is evaluated.
- **Time** field: This field is used with the **Calendar** and **Clock** Event types.

  If the Event type is **Calendar**, then **Time** is a specific time of the day on **Date**. When that **Date** and **Time** occurs, the specified **Expression** is evaluated.

  If the Event type is **Clock**, then **Time** is a time interval starting from when the project was run. Every time the interval occurs, the specified **Expression** is evaluated.

  Either way, type a time using the **`HH:MM:SS.ms`** format. Valid values are **`00`** to **`23`** for hours, **`00`** to **`59`** for minutes, **`00`** to **`59`** for seconds, and **`1`** to **`9`** for milliseconds. (Milliseconds are optional.) Examples: **`03:00:00`** is every three hours, **`00:00:00.1`** is every 100 milliseconds.
- **Date** field: This field is used only with the **Calendar** Event type. Type a specific date formatted according to the current date format on the project runtime server; for more information, see About the date format and how to change it on page 694. When the specified **Date** and **Time** occurs, the specified **Expression** is evaluated.

  If the field is left blank, then the event occurs daily at the specified **Time**.
- **Tag** field: Type a tag that will receive the value returned by **Expression** (if any).
- **Expression** field: Type an expression to be evaluated. This field is used by all events.
- **Disable** field: Contains a disable condition for the specified function. Leave this field blank or use an expression value equal to zero to execute the function. Use an expression value equal to one and the function will not execute (Disable = 1).

## Database/ERP worksheet

This software uses Microsoft .NET ActiveX Data Objects (ADO.NET) to interface between the project tags database and other external databases. You can configure a Database/ERP worksheet to associate project tags with external database fields.

> **Note:** For more information about ADO.NET support in this software — including how to communicate with remote databases using the Database Gateway — please see Database Interface on page 815.

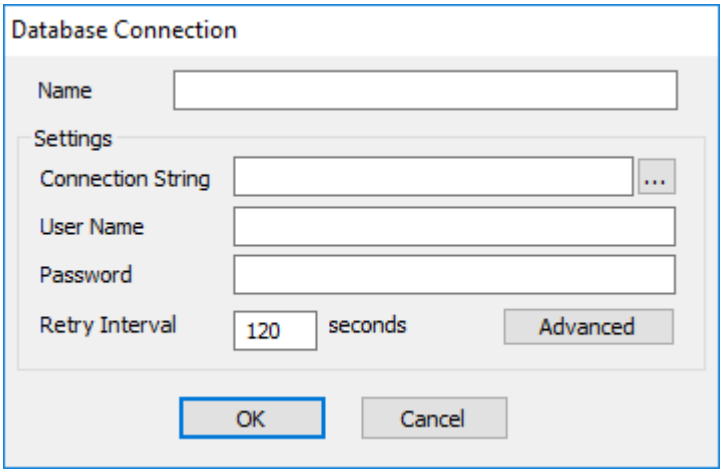
To interface with an external database, you must first configure a connection to the database and then build a worksheet that associates project tags with the database fields.

### Database Connections

To create a new connection to a target database:

1. In the *Project Explorer*, open the **Database/ERP** folder and then right-click on **Connections**.
2. Choose **Insert** from the shortcut menu.

   The *Database Connection* dialog is displayed.



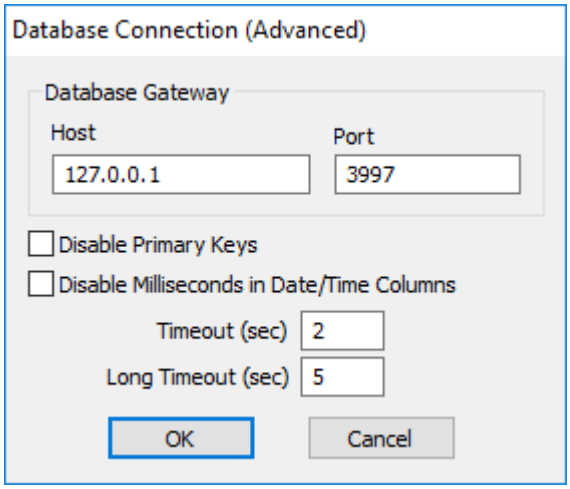

***Database Connection dialog***

3. In the **Name** field, enter the name that you want to use to reference the target database. You can create multiple database connections, but each connection must have a unique name.
4. In the **Connection String** field, click the browse button ... to open a standard *Data Link Properties* dialog. Use the dialog to configure a connection string for the target database.

   > **Note:** The list of Database Providers shown in the *Data Link Properties* dialog depends on the providers actually installed and available in the station where you are running the development application. For more information about using the *Data Link Properties* dialog, please refer to Windows Help.

5. In the **User Name** and **Password** fields, enter an appropriate login for the target database. The login should already be created on the database server, and it should have enough privileges to read from and write to the database tables.

6. If you are connecting to a remote database through the Studio Database Gateway, then click the **Advanced** button to open the advanced settings dialog, as shown below.




***Database Connection (Advanced) dialog***

7. In the **Host** field, enter the IP address of the station that is running the Database Gateway software (**`STADOSvr.exe`**). In the **Port** field, enter the port number on which the software has been configured to run.

   Other settings to configure, if necessary:

   - **Disable Primary Keys** checkbox: the project runtime will try to define a primary key to the table in order to speed up the queries. If you are using a database that does not support primary keys (e.g., Microsoft Excel), then you should check this box.
   - **Disable Milliseconds in Date/Time Columns** checkbox: the project runtime will try to include milliseconds when saving a date/time in the database. If you are using a database that does not support milliseconds, then you should check this box.

8. Click **OK** to close the dialog and save the connection configuration.

Database connections are saved as XML files in the `\<project name>\Config` sub-folder. Each file is given the same name as the name of the connection (as entered in the **Name** field of the *Database Connection* dialog), with the **`.XDC`** file extension. For example, the connection configuration `DB1` is saved in the file…

**`\<project name>\Config\DB1.XDC`**

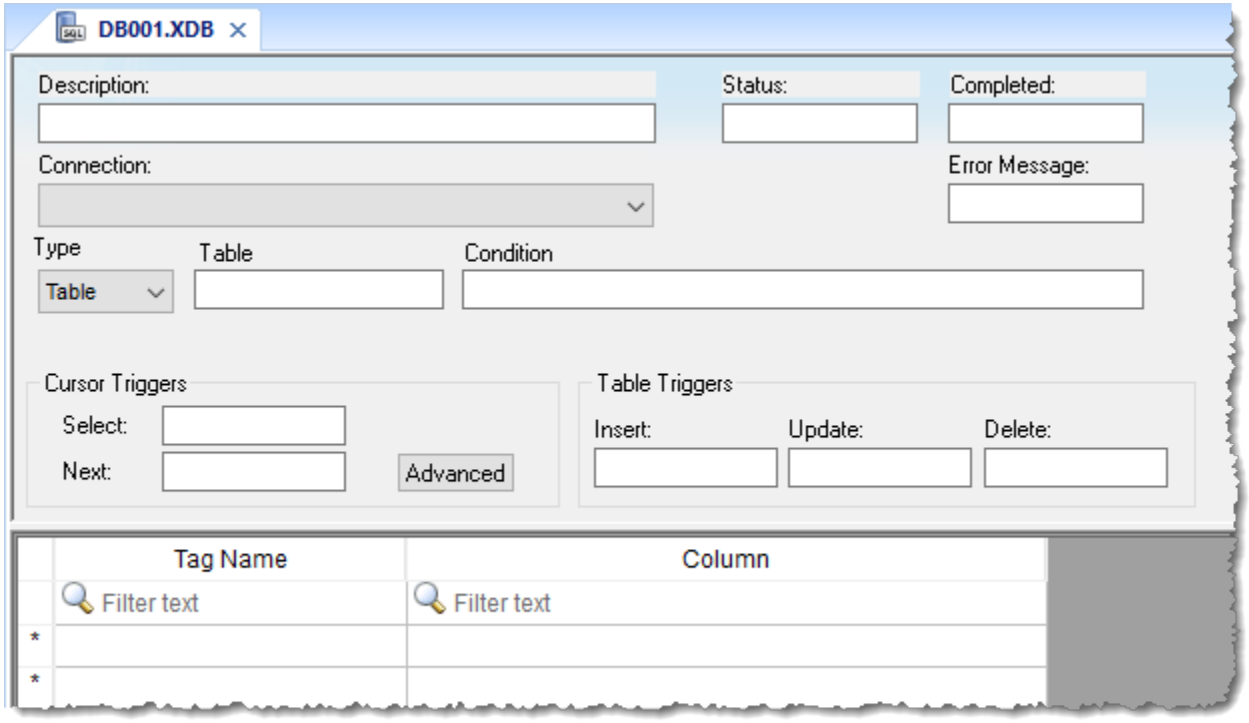
## Database Worksheet

> **Note:** This feature emulates Structured Query Language (SQL) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

Database worksheets allow asynchronous execution of database operations, and they offer a user-friendly interface for building SQL commands. Use one of the following methods to create a new database worksheet:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Database**; or
- Right-click on the **Database/ERP** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or

A new worksheet is displayed, as shown below:




***Database worksheet***

Database worksheets are saved with the **.XDB** file extension, in the Config sub-folder of your project folder. Each new worksheet is automatically numbered in the order of its creation. For example, the first worksheet created is saved in your project folder at: *<project name>*\Config\DB001.XDB

Database worksheets are executed under the *Database Client Runtime* task. However, creating a new worksheet does not automatically enable the task; you must use the *Runtime Tasks* dialog box (**Tasks** on the Home tab of the ribbon) to configure the task to start at runtime. For more information, please see Runtime Tasks.

Also, database worksheets run only on the server, and all triggers must be configured with server tags.

**Worksheet Header**

The header of the database worksheet is configured as follows:

- **Description** field: Enter a description of the worksheet, for documentation purposes.
- **Status** field: Enter the name of a numeric tag that will receive status codes for database operations during runtime:

**Status codes for external database operations**

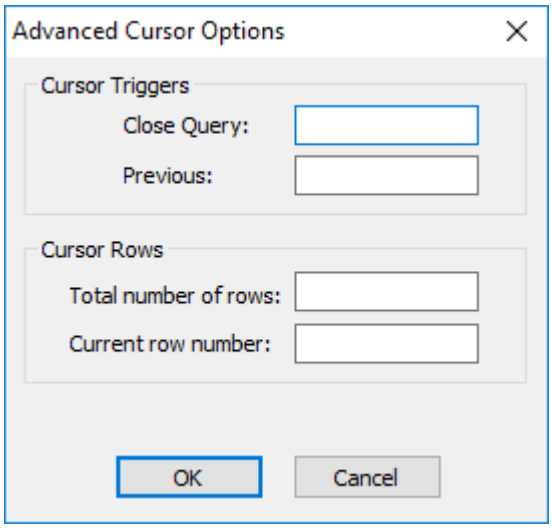
| Status Code | Description |
|---|---|
| 4 | Result set is empty |
| 3 | Cursor released and query successfully closed |
| 2 | Beginning of result set reached, usually while moving cursor to previous row |
| 1 | End of result set reached, usually while moving cursor to next row |
| 0 | No errors; status normal |
| -1 | Error while connecting to specified database (see **Connection** below) |
| -2 | Error while selecting result set |

| Status Code | Description |
|---|---|
| -3 | Error while moving cursor to next row (see **Next** trigger below) |
| -4 | Error while moving cursor to previous row (see **Previous** trigger below) |
| -5 | Error while closing the query (see **Close Query** trigger below) |
| -6 | Error while inserting rows in result set (see **Insert** trigger below) |
| -7 | Error while updating result set (see **Update** trigger below) |
| -8 | Error while deleting result set (see **Delete** trigger below) |

- **Completed** field: Enter the name of a numeric tag that will be toggled when database commands are successfully executed.
- **Error Message** field: Enter the name of a string tag that will receive detailed error messages, if errors occur during runtime.
- **Connection** combo-box: Click to select a connection to the target database. All available connections are listed, as configured with the *Database Connection* dialog described above.
- **Type** combo-box: Click to specify how the result set will be selected for the worksheet:
  - **Table**: Enter a table name and an optional filter condition. (The filter condition is equivalent to the SQL "Where" clause.) All rows of the table that match the filter condition are selected.
  - **SQL**: Enter a custom SQL "Select" statement.

> **Note:** For **Table**, **Condition** and **SQL Statement**, you can enter the names of project tags that contain the desired information. This lets you programmatically change the selection during runtime. However, tag names must be enclosed in curly brackets ({}) to distinguish them from literal strings. Also, you must release an existing selection before you open a new one; see **Close Query** below.

- *Cursor Triggers* area…
  - **Select** field: Enter any tag; when the value of the tag changes, a new cursor opens the first row of the result set and copies those values to the tags configured in the worksheet body.
  - **Next** field: Enter any tag; when the value of the tag changes, the cursor moves to the next row of the result set and copies those values to the tags configured in the worksheet body.
  - **Advanced** button: Click to open the *Advanced Cursor Options* dialog…




***Advanced Cursor Options dialog***

  - **Close Query** field: Enter any tag; when the value of the tag changes, the cursor releases the result set.

- **Previous** field: Enter any tag; when the value of the tag changes, the cursor moves to the previous row of the result set and copies those values to the tag configured in the worksheet body.
- **Total number of rows** field: Enter a numeric tag that will receive the total number of rows in the result set.
- **Current row number** field: Enter a numeric tag that will receive the number of the current row (i.e., the position of the cursor). When a result set is first opened using the **Select** trigger, this number is 1. Each **Next** trigger increments this number, and each **Previous** trigger decrements it.

- *Table Triggers* area…
  - **Insert** field: Enter any tag; when the value of the tag changes, a new row is inserted with the current values of the tags configured in the worksheet body.
  - **Update** field: Enter any tag; when the value of the tag changes, all rows of the result set are overwritten with the current values of the tags configured in the worksheet body.
  - **Delete** field: Enter any tag; when the value of the tag changes, all rows of the result set are deleted.

> **Note:** Table triggers are available only when **Type** is set to **Table**, because these operations work on the entire table row.

**Worksheet Body**

In the body of the worksheet, you can map project tags to the columns (fields) of the result set. For each row of the body, enter a **Tag Name** and its corresponding **Column**. Which columns are available depends on how the result set is selected, and how it is selected may change during runtime, so be sure to map all necessary columns.

> **Note:** You may have up to 2048 rows per worksheet. If you need more than that, then try creating additional worksheets and adjusting the result set for each worksheet.

## Sort or filter the rows in a worksheet

Sort or filter the rows in a worksheet in order to make it easier to browse the rows or find a specific item.

Before you begin this task, you must have already inserted a worksheet and opened it for editing. You should also be familiar with how sorting and filtering is done in general-purpose spreadsheet applications.

Please note that you can sort or filter rows only in the following types of worksheets:

- The Project Tags, Shared Tags, and System Tags datasheets;
- The Translation Table worksheet;
- All task worksheets except Report and Script, which do not have rows; and
- All communication worksheets.

None of the other worksheets have rows to sort or filter.

Sorting is done alphanumerically, by the selected column, in either ascending (0–9, A–Z) or descending (Z–A, 9–0) order.

| | Tag Name | Type |
|---|---|---|
| | Filter text | ... |
| 1 | Tag1 | HiHi |
| 2 | Tag1 | Hi |
| 3 | Tag1 | Lo |
| 4 | Tag1 | LoLo |
| 5 | Tag2 | HiHi |
| 6 | Tag2 | Hi |
| 7 | Tag2 | Lo |
| 8 | Tag2 | LoLo |
| 9 | Tag3 | HiHi |
| 10 | Tag3 | Hi |
| 11 | Tag3 | Lo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

***Alarm worksheet rows in their original order***

| | Tag Name | Type |
|---|---|---|
| | Filter text | ... |
| 10 | Tag3 | Hi |
| 2 | Tag1 | Hi |
| 6 | Tag2 | Hi |
| 9 | Tag3 | HiHi |
| 5 | Tag2 | HiHi |
| 1 | Tag1 | HiHi |
| 3 | Tag1 | Lo |
| 11 | Tag3 | Lo |
| 7 | Tag2 | Lo |
| 8 | Tag2 | LoLo |
| 4 | Tag1 | LoLo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

***Alarm worksheet rows sorted by Type***

Filtering is done according to whatever string you enter in the selected column. Only the rows that match the string will be displayed.

| | Tag Name | Type |
|---|---|---|
| | Tag3 | ... |
| 9 | Tag3 | HiHi |
| 10 | Tag3 | Hi |
| 11 | Tag3 | Lo |
| 12 | Tag3 | LoLo |
| * | | HiHi |
| * | | HiHi |

*Alarm worksheet rows filtered where Tag Name is "Tag3"*

| | Tag Name | Type |
|---|---|---|
| | Filter text | Lo |
| 3 | Tag1 | Lo |
| 7 | Tag2 | Lo |
| 11 | Tag3 | Lo |
| * | | HiHi |
| * | | HiHi |

*Alarm worksheet rows filtered where Type is "Lo"*

> **Tip:** You can still delete rows while they are sorted or filtered.

To sort or filter rows:

1. To sort the rows, click the header of the column by which you want to sort. Click once to sort in ascending order, and then click again to sort in descending order.

   The current order (i.e., the direction of the sort) is indicated by the arrow to the right of the column name.

   > **Note:** You cannot sort by multiple columns.

2. To undo the sorting and restore the rows to their original order, click the header of the first (numbered) column.
3. To filter the rows, type the string that you want to match in the top (zero) row of the worksheet and then press either **Tab** or **Return**.

   You may include `*` and `?` as wildcard characters in your string:

   - `*` matches any number of characters, including none. For example, `Tag*` would match **`Tag`**, **`Tag3`**, **`Tag34567`**, **`TagA`**, and **`Tag_TEMP`**.
   - `?` matches exactly one character. For example, `Tag?` matches **`Tag3`** and **`TagA`**, while `Tag?????` matches **`Tag34567`** and **`Tag_TEMP`**.

   Also, you may filter by multiple columns. Only the rows that match the filter strings in all columns will be displayed.
4. To undo the filtering and restore the rows to their original order, delete the string that you typed and then press either **Tab** or **Return**.

Please keep in mind that sorting or filtering the rows of a worksheet only helps you to edit that worksheet. It does not change how the worksheet is executed during run time. The rows will be executed in their original numbered order (i.e., the leftmost column) unless you actually move or delete a row.

# Communication

Communication tasks/worksheets are used to exchange tag values with other EcoStruxure Machine SCADA Expert projects, remote devices such as PLCs and transmitters, and any other systems that implement supported protocols like OPC.

## Configuring direct communication with a remote device

A communication driver is a DLL containing specific information about the remote equipment and implements the communication protocol. Drivers for dozens of common and not-so-common devices are installed with EcoStruxure Machine SCADA Expert 2020.

( Schneider Electric also provides a toolkit to develop new communication drivers. For more information, please contact Customer Support.)

The Drivers task/worksheet allows you to define the communication interface (or interfaces) between the project and remote equipment; such as a PLC, a single-loop, and transmitters.

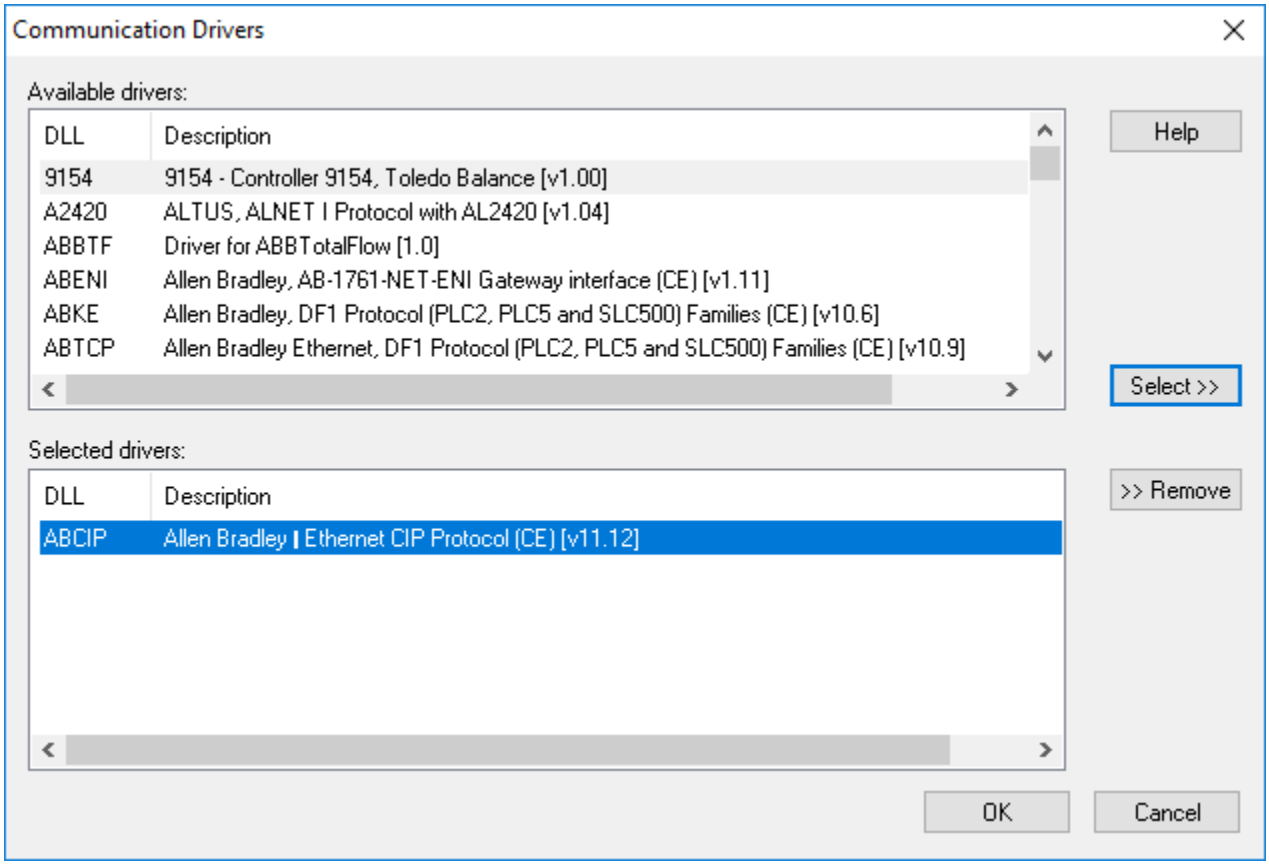
> **Note:** Consult the **Help** menu for a description of the functions and characteristics that are standard for all drivers. When developing a project, you can also refer to the specific documentation provided with each communication driver. This documentation is usually located in the **DRV** directory.

To configure a communication driver, you must specify the interface parameters (for example, the station address and the baud rate), specify the equipment addresses, and then link them to project tags.

Use one of the following methods to add or remove a driver:

- On the Insert tab of the ribbon, in the Communication group, click **Add/Remove Driver**; or
- Right-click the **Drivers** folder in the Project Explorer, and then click **Add/Remove drivers** on the shortcut menu.

Both methods open a *Communication Drivers* dialog, which displays a list of available drivers.




***Communication Drivers dialog***

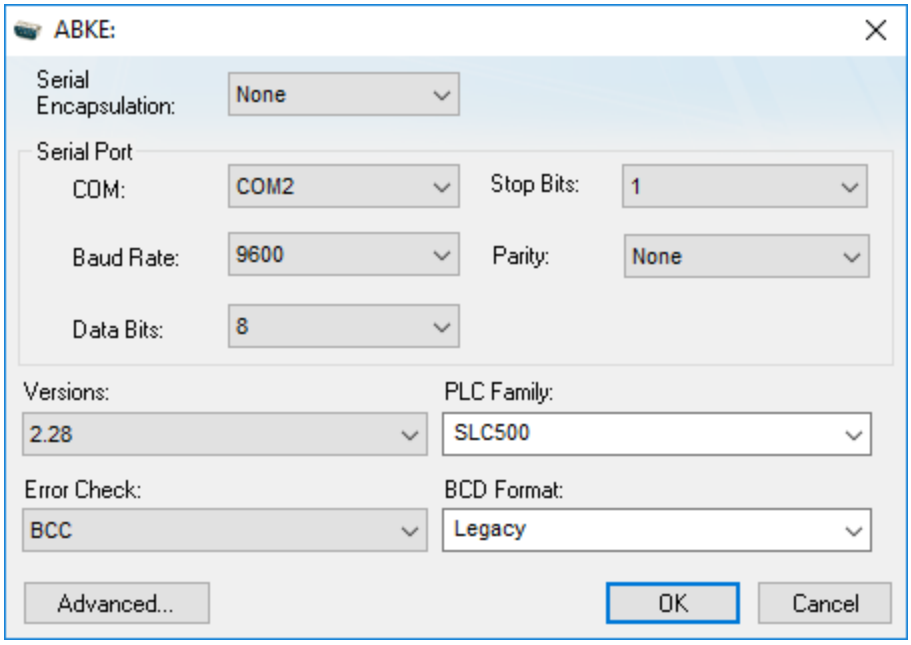
Use the parameters on this dialog, as follows:

- **Available Drivers** field: Lists all available drivers and a brief description of each.

- **Help** button: Click to open the **Help** menu, which contains detailed configuration instructions for the driver currently highlighted in the **Available Drivers** field.
- **Select** button: Click to select the driver currently highlighted in the **Available Drivers** field.
- **Selected Drivers** field: Lists all selected drivers and their descriptions (if available).
- **Remove** button: Click to remove a driver currently highlighted in the **Selected Drivers** field.

When you click **OK** in the *Communications Driver* dialog, you create a subfolder for the selected driver(s) in the *Drivers* folder located on the *Comm* tab.

You can right-click on a driver subfolder to access the **Settings** option, which opens the *Communications Parameters* dialog.




***Sample Communications Parameters dialog***

Use the parameters on this dialog, as follows:

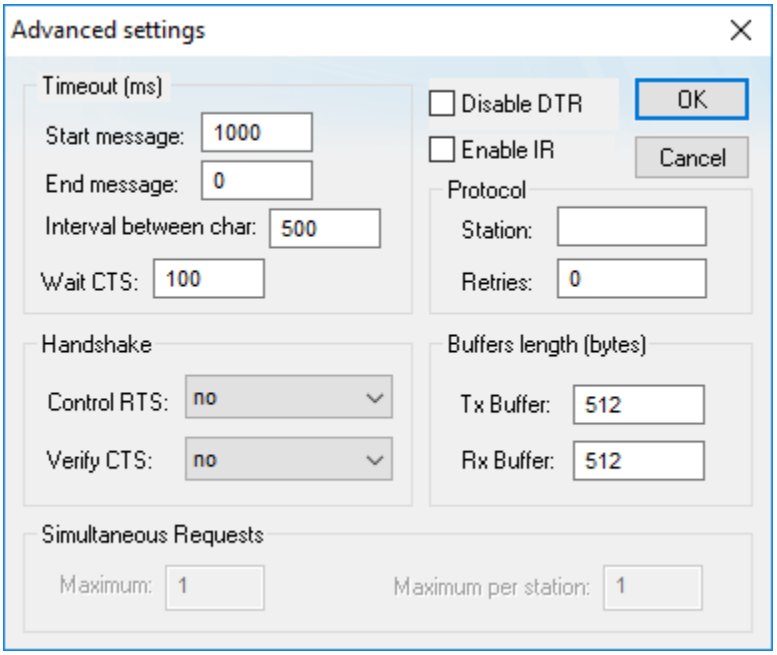
- **Serial Encapsulation** field: Enables serial drivers to communicate with modem, TCP/IP or UDP connections. This setting is supported only for serial drivers developed with the UNICOMM library, which includes most of the serial drivers available in the product.

> **Note:** The **Modem** option is not supported for Pocket PC v3.00 or older.

> **Note:** This section covers only the **None** option, which enables the driver to connect using a normal serial channel. Please refer to "Using TCP/IP and UDP Encapsulation" and "Using Modem Connections" below for more information about other encapsulation modes. "Serial Encapsulation Tests" below lists the drivers that have been tested with modem, TCP/IP and UDP modes.

- **COM** field: Click to select a serial communication port.
- **Baud Rate**, **Data Bits**, **Stop Bits**, **and Parity** fields: Click to select parameters for a serial port configuration.
- **Long1**, **Long2**, **String1**, and **String2** fields: These fields are driver custom settings. In the example above, the driver uses **Long1** to set up the error detection method and **String1** to define the PLC family type.

- **Advanced** button: Click to open the *Advanced settings* dialog. Use this dialog to change the default driver parameters.




***Advanced Settings dialog***

Specify or change the default driver parameters as follows:

- *Timeout (ms)* area:
  - **Start message** field: Specify the timeout for the message start.
  - **End message** field: Specify the timeout for the message end.
  - **Interval between char** field: Specify the timeout between each character.
  - **Wait CTS** field: Specify the timeout for the Clear to Send wait.
- *Handshake* area:
  - **Control RTS** drop-down list: Specify whether to use the "Request to Send" control.
  - **Verify CTS** drop-down list: Specify whether to use the "Clear to Send" type of verification.
  - **Disable DTR** checkbox: Click (enable) this box to disable the DTR function (the driver will not set the DTR signal before starting the communication).
  - **Enable IR** checkbox (*only available on Windows Embedded target systems*): Click (enable) this box to enable the serial driver to use an Infrared interface (COM2 port) instead of a standard serial port to communicate with the device (such as the PLC, I/O, hand-held computers, and so forth).
- *Protocol* area:
  - **Station** field: Some slave drivers such as the Modbus Slave (MODSL) require a slave network address. Use this field to specify the slave address.
  - **Retries** field: Type a numeric value to specify how many times the driver will attempt to execute the same communication command before considering a communication error for this command.
- *Buffers length (bytes)* area:
  - **Tx Buffer** field: Specify the transmission buffer length (in bytes).
  - **Rx Buffer** field: Specify the reception buffer length (in bytes).
- *Simultaneous Requests* area (available only on selected drivers):

- **Maximum** field: Specify the maximum number of requests that may be sent simultaneously to all connected devices.
- **Maximum per station** field: Specify the maximum number of requests that may be sent simultaneously to a single device.

> **Note:** The maximum number of simultaneous requests depends on the device and protocol specifications. Please consult the device manufacturer's documentation.

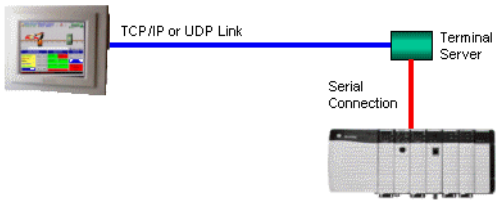
The development application provides two interfaces, which you can use to configure the driver (associating project tags to device addresses):

- **MAIN DRIVER SHEET**: Provides the easiest method for configuring communication between project tags and device addresses. This interface allows you to automatically group tags to provide the best performance during runtime. You cannot use this interface to control the time needed to scan a group of tags individually.
- **STANDARD DRIVER SHEETS**: Allows you to control the time needed to scan a group of tags individually.

You can use both sheets at the same time.

### Using TCP/IP and UDP Encapsulation

Most of the serial drivers allow the use of TCP/IP or UDP/IP encapsulation. The encapsulation mode has been designed to provide communication with serial devices connected to terminal servers on your ethernet or wireless networks. A terminal server can be seen as a virtual serial port. It converts TCP/IP or UDP/IP messages on your Ethernet or Wireless network to serial data. Once the message has been converted to a serial form, you can connect standard devices that support serial communications to the terminal server. The following diagram provides one example of applying this solution:



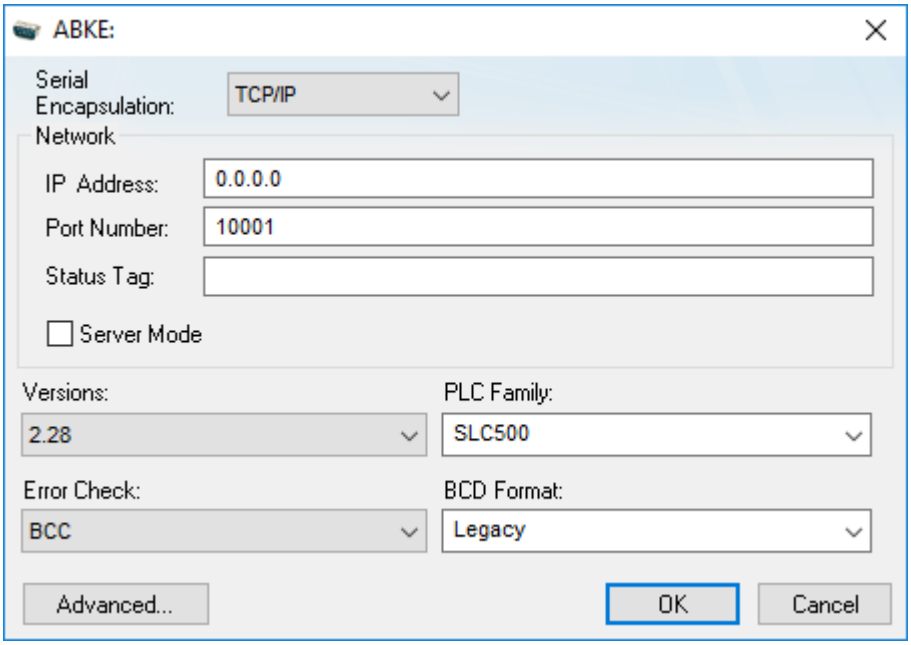

*TCP/IP Encapsulation*

You can enable the encapsulation by following the steps below:

1. Right-click on the driver's folder, and then choose **Settings** from the shortcut menu.

   This will give you access to the communication parameters.

2. In the **Serial Encapsulation** field, select **TCP/IP** or **UDP/IP**:

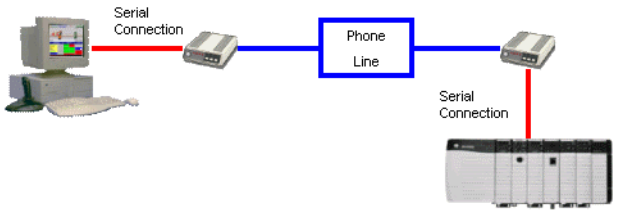



The following fields are available:

- **IP Address** field: Specify the IP Address for the Terminal Server. This field accepts tags in curly brackets.
- **Port Number** field: Enter the TCP/IP or UDP/IP port number.
- **Status Tag** field: This field is available only when using TCP/IP. The tag on this field receives the value 1 when the TCP/IP connection is established; otherwise, it receives 0.
- **Server Mode** field: The TCP/IP encapsulation allows the Server Mode, making the remote client responsible for establishing the connection to enable the communication.

## Using Modem Connections

Most of the serial drivers allow the use of modem connections. The modem connection has been designed to enable communications with remote serial devices connected through a phone line. The following diagram provides one example of applying this solution:



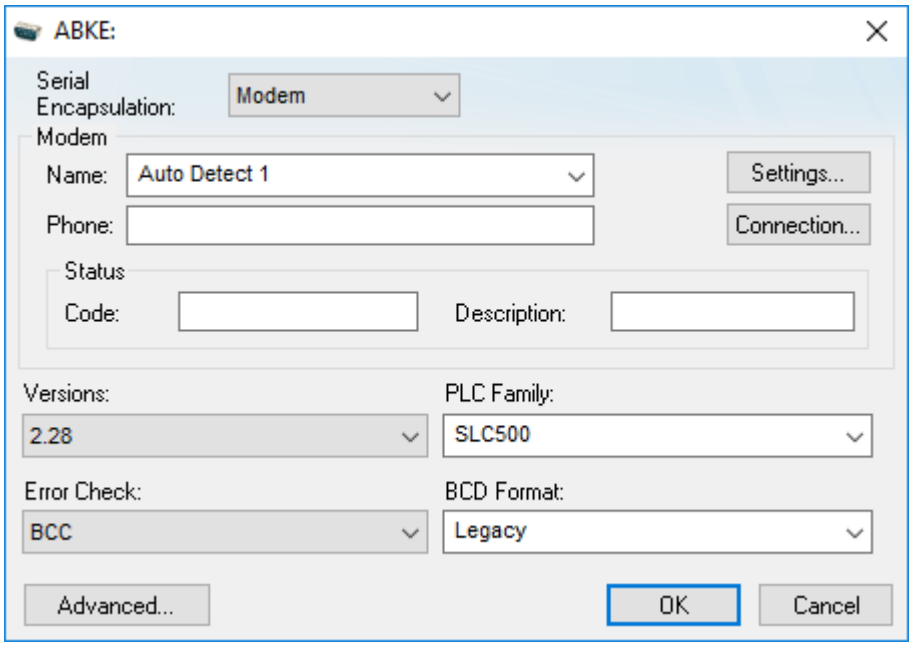

***Modem Connection***

You can enable the modem connection by following the steps below:

1. Right-click on the driver's folder, and then choose **Settings** from the shortcut menu.

   This will give you access to the communication parameters.

2. In the **Serial Encapsulation** menu, select **Modem**:



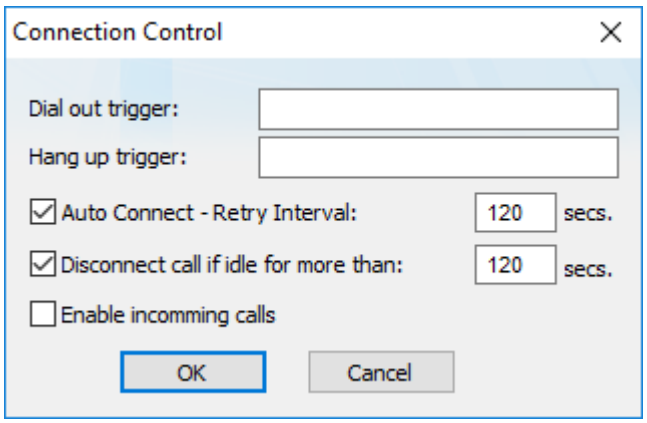

> **Note:** The **Modem** option is not supported for Pocket PC v3.00 or older.

The following fields are available:

- **Name** drop-down list: Select the modem that the driver will use to establish the connection. If you do not know the modem name, use the Auto Detect option. The **Auto Detect 1** will use the first modem available, **Auto Detect 2** will use the second, **Auto Detect 3** will use the third, and **Auto Detect 4** will use the fourth.
- **Phone** field: Enter a phone number that the driver will use to connect to the remote device. This field accepts tags between curly brackets.
- **Settings** button: Click on this button to configure the modem settings. The window that displays when you click on this button depends on the operating system that you are using and on the modem type.

  > **Note:** The settings configured by clicking on this button are not saved with your project. The information is saved on the operating system registry, and they are valid only in the computer that you are interacting with. If you install your project on another computer, you will have to reconfigure these settings.

- **Connection** button: Click to open the *Connection Control* window. The default connection settings should suffice for most of the projects. However, you can take full control over the connection, and also enable incoming calls, by clicking on this button.




***Connection Control dialog***

  - **Dial out trigger** field: When the value of the tag configured in this field changes, the driver will try to connect to the remote device. If the connection has already been established, the command is ignored. You do not have to use this field if you are using Auto Connect.
  - **Hang up trigger** field: When the value of the tag configured in this field changes, the driver will disconnect from the remote device. If the device is disconnected the command is ignored. You do not have to use this field if you are using Disconnect call if idle for more than.
  - **Auto Connect** field: When this option is enabled, the driver will try to connect to the remote device before sending any information. If the connection fails, the next attempt will be made after the Retry Interval has expired.
  - **Disconnect call if idle for more than** field: When this option is checked, the driver will automatically disconnect from the remote device if no communication is performed after the time you specified.
  - **Enable incoming calls** field: Check this option if you want to enable the driver to receive calls from the remote device. You can use the Hang up trigger to drop the call once it has been established. Notice that one driver can use both incoming calls and outgoing calls.
- *Status* area
  - **Code** field: Enter with a tag that will receive one of the following codes when the driver is running:
    - 0 = Disconnected
    - 1 = Connected
    - 2 = Dialing
    - 3 = Dropping
    - 4 = Closing Line
  - **Description** field: Enter with a tag that will receive a complete description of the current status. The description is associated with the **Code** field; however, it brings some additional information about the current status.

### Serial Encapsulation Tests

Most of the serial drivers should work with every serial encapsulation mode. However, most of the drivers were developed before the encapsulation modes had been created. The following table lists the drivers fully tested with certain encapsulation modes; if the driver that you intend to use is not listed and you are unsure whether it will work, please contact your distributor.

| Driver | Modem | TCP/IP | UDP/IP |
|---|---|---|---|
| MODSL | X | X | X |

| Driver | Modem | TCP/IP | UDP/IP |
|---|---|---|---|
| ABKE | X | X | X |
| MODBU | X | | X |
| OMETH | X | | |

**X** = Item has been tested

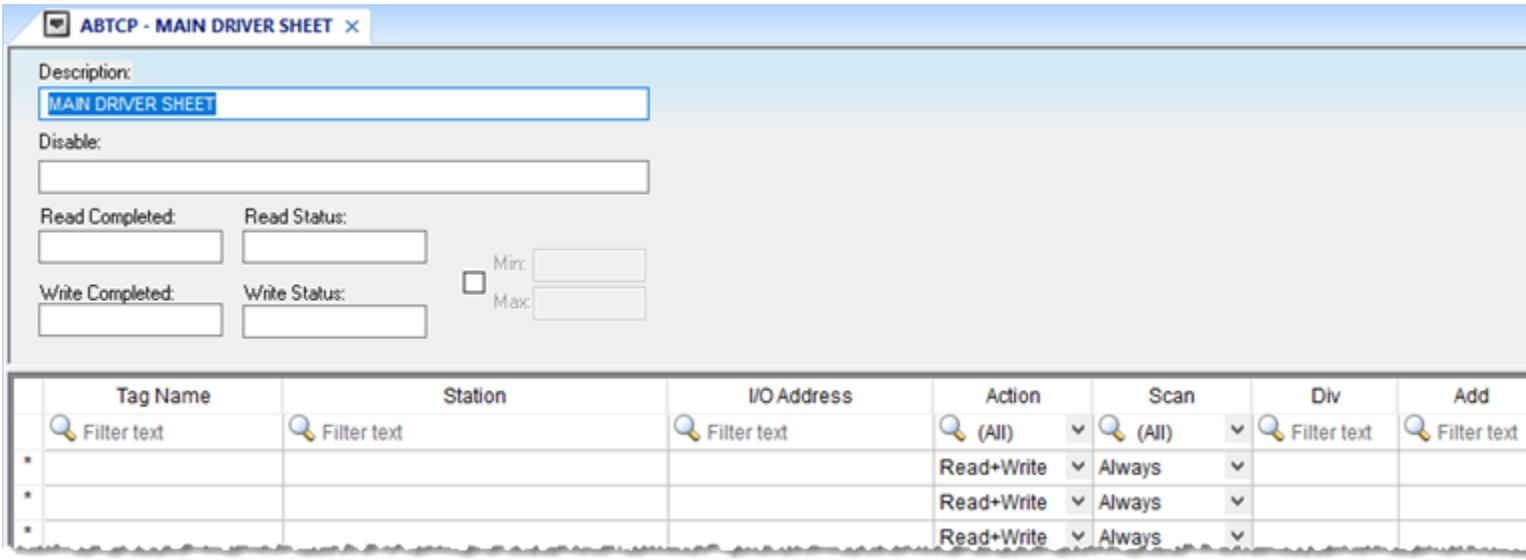
### *Main Driver Sheet*

The development application automatically inserts the MAIN DRIVER SHEET into the driver folder as soon as you add the driver to your project.

> **Note:** The MAIN DRIVER SHEET is not available for all drivers.

To configure the MAIN DRIVER SHEET, right-click on the icon, and select **Open** from the pop-up or just double-click on the icon.

The MAIN DRIVER SHEET dialog displays (see the following figure).




***Sample MAIN DRIVER SHEET***

The *MAIN DRIVER SHEET* worksheet is divided into two areas:

- *Header* area (top section), contains parameters that affect the all tags configured in the Body area of this worksheet; and
- *Body* area (bottom section), where you define the relationship between tags in the project and their field equipment address.

Use the *Header* area parameters as follows:

- **Description** field: Type a description of the MAIN DRIVER SHEET for documentation purposes.
- **Disable** field: Type a tag or an expression to enable and disable the communication of each MAIN DRIVER SHEET on the fly.
  - Type a value (or expression result) that is greater than zero to disable the MAIN DRIVER SHEET.
  - Type a zero (or leave this field blank) to enable the MAIN DRIVER SHEET.
- **Read Completed** field: Type in a tag and the communication driver toggles the tag when it completes a **`read`** command.

- **Read Status** field: Type in a tag, which is updated with the status of the last **`read`** command.
- **Write Completed** field: Type in a tag and the communication driver toggles the tag when it completes a **`write`** command.
- **Write Status** field: Type in a tag, which is updated with the status of the last **`write`** command.
- **Min and Max** checkbox: Click (check) to specify minimum and maximum values for data from the field equipment.
- **Min and Max** fields (become active only when you enable the **Min and Max** checkbox): Type a range of values, which can be converted into an engineering format.

  The project uses these fields to determine a minimum/maximum range of values for data from the field equipment. The scaling is done automatically. You must configure the engineering range using the **Min** and **Max** parameters on the *Tag Properties* dialog. This range affects all tags in the worksheet, except those with customized **Min** and **Max** values, as specified in the Body area of the driver sheet (**Min** and **Max** columns).

Use the *Body* area parameters as follows:

- **Tag Name** field: Type the name of a project tag to be used by the communication driver.
- **Station** field: Type the number of the equipment station within the network. The syntax in this field varies with each communication driver. Refer to the appropriate driver's documentation for further information.

  **Tip:** For some drivers, if you've configured the driver to do serial encapsulation via TCP/IP or UDP/IP, then the station may be specified using the following format:

  ***`IP_address:port_number|station`***

  For example:

  **`10.169.25.18:1234|Station5`**

  To see if this feature is supported on your selected driver, refer to the driver's documentation.

  **Tip:** You can configure a tag name (string) between curly brackets in this field. In this case, the tag value will be the Station used by the driver. Therefore, you can change the station dynamically during runtime.

  Configuring a string tag between curly brackets in the Station field of the Main Driver Sheet (MDS) is especially useful when configuring projects for redundant PLCs. Changing the value of the tag configured in the Station field, you can switch automatically from one PLC to the other in case of a failure of the primary PLC (hot/Stand-by).

- **I/O Address** field: Type the address of the field equipment related to the project tag. The syntax in this field varies with each communication driver. Refer to the appropriate driver's documentation for further information.
- **Action** field: Specify the communication direction, using one of the following options:
  - **Read** (the project continuously reads the address from the field device and updates the **Tag** value.)
  - **Write** (the project writes the tag value to the field device when the tag value changes.)
  - **Read+Write** (Combines the procedures of both the **Read** and **Write**parameters.)
- **Scan** field: Specify the condition under which the tag value is read from the remote device or server and then updated in the project database, using one of the following options:
  - **Always** means the tag is read and updated during every scan of the communication worksheet, regardless of whether the tag is used in any other project screens, scripts, or worksheets.

    This option is recommended for tags that must be continuously monitored in the background, such as tags that trigger alarms, tags used in recipes, tags that are recorded in the historical database, and so on.

- **Screen** means the tag is read and updated only if it is being used in at least one open project screen, either locally or on another client station.

  This option is recommended for tags that are used in screen objects, because the project may not need to update tags that are not being visualized anywhere. Selecting this option can improve project performance.

- **Auto** means the project will automatically choose either **Always** or **Screen**, depending on where the tag is used in your project. If the tag is only used in a screen object on a project screen, then the scan will default to **Screen**. But if the tag is configured in any other interface (e.g., Script, Math, Alarm, Trend, Recipe, Report, Scheduler), then the scan will switch to **Always** and remain there until the project is stopped.

If you are not sure of which option to select, select **Always**. This will guarantee the tag is read and updated.

- **Div** field: Specify the division constant when scale adjustment is required. This value is a division factor in a **`read`** operation and a multiplication factor in a **`write`** operation. Do not use this field if you are already using **Min** or **Max** in the configuration body.
- **Add** field: Specify the addition constant when scale adjustment is required. This value is an addition factor in a **`read`** operation and a subtraction factor in a **`write`** operation. Do not use this field if you are already using **Min** or **Max** in the configuration body.

> **Note:** The Main Driver Sheet can have up to 32767 rows. If you need to configure more than 32767 communication addresses, then either configure additional Standard Driver Sheets or create additional instances of the driver.

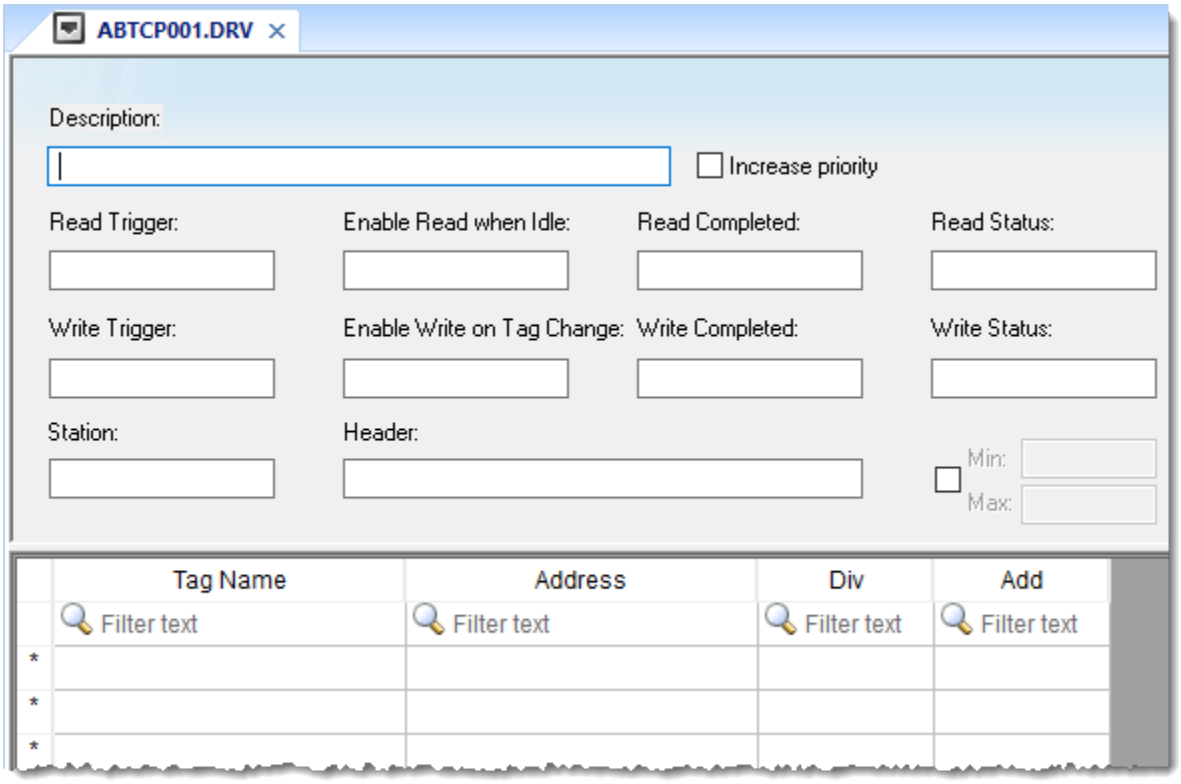
> **Tip:** By default, the project will scan the communication worksheet every 600 milliseconds, which is the rate at which the system tag **`BlinkSlow`** toggles. To adjust the rate, manually edit the project file (i.e., *<project name>*.APP) to add the following entry:
>
> ```
> [Options]
> MainDrvAlwaysTrigger=tagname
> ```
>
> *tagname* can be either another system tag (e.g., **`BlinkFast`**, **`Second`**, **`Minute`**) or a tag that you have created. Whenever the value of the tag changes, the worksheet will be scanned and the tags will be read.

### *Standard Driver Sheets*

In addition to the unique MAIN DRIVER SHEET that is available for each driver, you can create several STANDARD DRIVER SHEETS for each driver. The STANDARD DRIVER SHEETS provide additional fields, which you can use to control communication.

> **Note:** You can have a total of 9,999 Standard Driver Sheets for all drivers in your project.

To open a STANDARD DRIVER SHEET, right-click on a driver subfolder and select **Insert** from the resulting popup (see the following figure).




***Sample STANDARD DRIVER SHEET***

The *STANDARD DRIVER SHEET* dialog is divided into two areas:

- *Header* area (top section), contains parameters that affect the all tags configured in the Body area of this worksheet
- *Body* area (bottom section), where you define the relationship between tags in the project and their field equipment address.

Use the *Header* area parameters as follows:

- **Description** field: Type a description of the STANDARD DRIVER SHEET for documentation purposes.
- **Increase Priority** checkbox: Click (check) to keep the reading and writing commands for this sheet on the top of the communication queue whenever they are triggered.

> **Note:** You must give special attention to this worksheet when you enable the **Increase Priority** option. If the worksheet keeps triggering communication commands, the project may never be able to execute the other driver sheets.

- **Read Trigger** field: Type a tag that triggers the project to read the worksheet automatically when you change this tag's value.
- **Enable Read when Idle** field: Type a tag or constant value. Use a tag (or constant) value greater than zero, to enable reading from the equipment.

> **Note:** If you use a constant value (other than zero), be sure that your project requires a continuous reading because this value places a reading request in every communication scan.

- **Read Completed** field: Type in a tag and the communication driver toggles the tag when it completes a **`read`** command.

- **Read Status** field: Type in a tag and the communication driver updates the tag with the status of the last **`read`** command.
- **Write Trigger** field: Type a tag value to activate a group reading. Whenever you change this tag value, the program writes an equipment worksheet.
- **Enable Write on TagChange** field: Type a tag or constant value (not zero) to enable the communication driver to check the worksheet continuously for changes in the tag value. If a change occurs, the project writes this value to an address in the field equipment.
- **Write Completed** field: Type in a tag and the communication driver toggles the tag in this field when a **`write`** command completes.
- **Write Status** field: Type in a tag and the communication driver updates the tag with the status of the last **`write`** command.
- **Station** field: Type the equipment station number within the network. The syntax in this field varies with each communication driver. Refer to the appropriate driver's documentation for further information.

  > 👉 **Tip:** For some drivers, if you've configured the driver to do serial encapsulation via TCP/IP or UDP/IP, then the station may be specified using the following format:
  >
  > ***`IP_address:port_number|station`***
  >
  > For example:
  >
  > **`10.169.25.18:1234|Station5`**
  >
  > To see if this feature is supported on your selected driver, refer to the driver's documentation.

- **Header** field: Specify the data type and/or initial address to be read or written in the equipment. The syntax in this field varies with each communication driver. Refer to the appropriate driver's documentation for further information.

  > 👉 **Note:** You can use text in the **Station** and **Header** fields with tag values using the **`text {tag}`** syntax.

- **Min** and **Max** checkbox (not labeled): Click (check) to specify the minimum and maximum values for field equipment data.
- **Min** and **Max** fields (become active only when you enable the **Min and Max** checkbox): Type a range of values to be converted into an engineering format. These fields determine the minimum and maximum range of values. These values affect all tags in the worksheet.

  For example, Memory holds values from 0 to 4095, which means 0% to 100% in the user interface. So for this example, you must specify 0 to 100 for the min and max tag parameters.

Use the *Body* area parameters as follows:

- **Tag Name** field: Type a tag name for the communication driver to use.
- **Address** field: Type a field equipment address (or address offset) related to the project tag. The syntax in this field varies with each communication driver. Refer to the appropriate driver's documentation for further information.
- **Div** field: Specify a division constant to use when scale adjustment is required. The project uses this value as a division factor in a read operation and a multiplication factor in a write operation. Do not use this field if you are already using **Min** or **Max** in the configuration body.
- **Add** field: Specify an addition constant to use when scale adjustment is required. The project uses this value as an addition factor in a read operation and a subtraction factor in a write operation. Do not use this field if you are already using **Min** or **Max** in the configuration body.

  For read operations:

  ***`tag`*** **`=`** **`(`*`value in the equipment`*`)`** **`/ Div + Add`**

For write operations:

```
value in the equipment = (tag - Add) * Div
```

If you leave the cells empty in the **Div** and **Add** fields, this function is ignored.

### Notes

Each Standard Driver Sheet can have up to 4096 rows. However, the **Read Trigger**, **Enable Read When Idle**, and **Write Trigger** commands attempt to communicate the entire block of addresses that is configured in the sheet, so if the block of addresses is larger than the maximum block size that is supported by the driver protocol, then you will receive a communication error (e.g., "invalid block size") during run time. Therefore, the maximum block size imposes a practical limit on the number of rows in the sheet, and that limit varies by driver. For more information, please refer to the driver documentation for your selected driver.

Also, keep in mind that when you use the **Write Trigger** feature with memory-based drivers (e.g., MODBU, MOTCP, ABTCP, OMETH, SIETH), the driver writes to the entire block of registers from the first address through the last. If a specific register has not been declared in the worksheet but its address is within the block, the register will receive a zero (0) value. Check the worksheet for gaps in the address range. This does not apply to name-based drivers (e.g.,TWCAT, COSYS, ABCIP).

## *Read/write status codes for direct communication drivers*

This is a list of common status codes that are generated by communication drivers during read/write operations.

| Status Code | Description | Possible Causes | Procedure To Solve |
|---|---|---|---|
| 0 | OK | Communicating without error. | None required. |
| -1 | Invalid serial port | • The selected serial port is invalid or unaccessible.<br>• The port is already in use | • Select another serial port in driver settings.<br>• Check the serial port settings. |
| -2 | Invalid baud rate | The selected baud rate is invalid. | Select a baud rate in the valid range. |
| -3 | Invalid number of bits | The selected number of bits for the serial port is invalid. | Select a number of bits in the valid range. |
| -4 | Invalid number of stop bits | The selected number of stop bits for the serial port is invalid. | Select a number of stop bits in the valid range. |
| -5 | Invalid parity | The selected parity for the serial port is invalid. | Select a parity in the valid range. |
| -6 | Invalid IRQ | The selected IRQ for the serial port is invalid. | Select an IRQ in the valid range. |
| -7 | Serial port already in use | The selected port is being used by other process. | Choose another serial port for communication. |
| -8 | Invalid buffer size | The buffer size entered is not allowed. | Check the documentation for allowed buffer sizes. |
| -9 | Memory not enough | Out of memory in the system. | Close other processes that may be consuming memory. |
| -10 | Tx buffer empty | The CE device was not able to write the message on the serial port. | Check if the serial port is valid and accessible. |
| -11 | Tx buffer full | The TX buffer has more data than allowed. | Increase the TX buffer size in driver settings. |
| -12 | Rx buffer empty | No data was received. | Check if the serial port is valid and the device is sending data. |
| -13 | Rx buffer full | The RX buffer has more data than allowed. | Increase the RX buffer size in driver settings. |
| -14 | Timeout waiting CTS | The CTS was not received in the expected time. | • Verify if the device should send the CTS, otherwise disable the Verify CTS option.<br>• Increase the Wait CTS timeout in Advanced Driver Settings |

| Status Code | Description | Possible Causes | Procedure To Solve |
|---|---|---|---|
| -15 | Timeout waiting for message to start | • Disconnected cables.<br>• PLC is turned off, in stop mode, or in error mode.<br>• Wrong station number.<br>• Wrong parity (for serial communication).<br>• Wrong RTS/CTS configuration (for serial communication). | • Check cable wiring.<br>• Check the PLC mode — it must be RUN.<br>• Check the station number.<br>• Increase the timeout in the driver's advanced settings.<br>• Check the RTS/CTS configuration (for serial communication). |
| -16 | Timeout waiting for message to finish | The message took too long to be received as completed. | Increase the timeout for End message in driver settings. |
| -17 | Timout between rx character | The time between charactes is too long. | Increase the Interval between char in driver settings. |
| -18 | Timout between tx character | • The time to write the buffer has exceeded the limit.<br>• The driver was not able to write the buffer | Check the serial port settings. Increase the Interval between char in driver settings. |
| -19 | No carrier detected | Invalid serial port configuration or hardware fault. | Check the serial port configuration and status. |
| -20 | No DSR detected | Invalid serial port configuration or hardware fault. | Check the serial port configuration and status. |
| -21 | Could not find a 8250 in address | Invalid serial port configuration or hardware fault. | Check the serial port configuration and status. |
| -22 | Tx line is busy | The TX buffer is full and unable to write to the serial buffer. | • Check the serial port configuration and status.<br>• Check the device status. |
| -23 | User abort | User has aborted the request. | Retry the request without aborting it. |
| -24 | Function not supported | Serial port received an unsupported function. | Check the serial port configuration and status. |
| -25 | Overrun | Invalid serial port configuration or hardware fault. | Check the serial port configuration and status. |
| -26 | Parity | Invalid serial port configuration or hardware fault. | Check the serial port configuration and status. |
| -27 | Overrun and parity | Invalid serial port configuration or hardware fault. | Check the serial port configuration and status. |
| -28 | Framing | Invalid serial port configuration or hardware fault. | Check the serial port configuration and status. |
| -29 | Framing and overrun | Invalid serial port configuration or hardware fault. | Check the serial port configuration and status. |
| -30 | Framing and parity | Invalid serial port configuration or hardware fault. | Check the serial port configuration and status. |
| -31 | Framing, overrun, and parity | Invalid serial port configuration or hardware fault. | Check the serial port configuration and status. |
| -32 | Timeout waiting for a tx message to finish | The message could not be sent completely or partially. | • Check the serial port configuration and status.<br>• Check the device configurations. |
| -33 | Invalid driver configuration file | The driver configuration file (*drivername*.ini) is missing or corrupt. | Reinstall the driver. |
| -34 | Invalid address | The specified address is invalid or out of range. | Check the supported range of addresses described in this document, and then correct the address. |
| -35 | Driver API not initialized | The driver library was not initialized by the driver. | Contact technical support. |

| Status Code | Description | Possible Causes | Procedure To Solve |
|---|---|---|---|
| -36 | Invalid data type | The specified data type is invalid or out of range. | Check the supported data types described in this document, and then correct the data type. |
| -37 | Invalid header | The specified header in the driver worksheet is invalid or out of range. | Check the supported range of headers described in this document, and then correct the header. |
| -38 | Invalid station | The specified station in the driver worksheet is invalid or out of range. | Check the supported station formats and parameters described in this document, and then correct the station. |
| -39 | Invalid block size | Worksheet is configured with a range of addresses greater than the maximum block size. | Check the maximum block size number of registers described in this document, and then configure your driver worksheet to stay within that limit. Keep in mind that you can create additional worksheets.<br>**Note:** If you receive this error from a Main Driver Sheet or Tag Integration configuration, please contact Technical Support. |
| -40 | Invalid bit write | Writing to a bit using the attempted action is not supported. | • Writing to a bit using Write Trigger is not supported in some drivers. Modify the driver worksheet to use Write On Tag Change.<br>• The bit is read-only. |
| -42 | Invalid bit number | The bit number specified in the address is invalid. The limit for the bit number depends on the registry type. | Check the addresses to see if there are bit numbers configured outside the valid range for the registry. |
| -43 | Invalid byte number | The byte number specified in the address is invalid. The limit for the byte number depends on the registry type. | Check the addresses to see if there are byte numbers configured outside the valid range for the registry. |
| -44 | Invalid byte write | Writing to a byte using the attempted action is not supported. | The byte is read-only or inacessible. |
| -45 | Invalid string size | The string is more than 1024 characters. | Modify the addresses that have string data type to be less than 1024 characters. |
| -50 | Invalid modem | The selected modem name is invalid. | Select a valid modem name in driver settings. |
| -51 | Error initializing TCP | The TCP library is not available for serial encapsulation. | Reinstall the driver. |
| -52 | Error listening to port | The port selected in serial encapsulation settings is not available. | • Check for other processes that might be using the same port number.<br>• Check the selected port number. |
| -53 | Error initializing UDP | The UDP library is not available for serial encapsulation. | Reinstall the driver. |
| -54 | Error sending data packet (TCP or UDP) | The TCP/UDP library was not able to send the packet. | Check the serial encapsulation settings. |
| -55 | Error not connected | Modem is not connected. | • Check the modem settings.<br>• Check if the modem dial number is available. |
| -56 | Invalid connection handle | The connection is no longer valid. | Please contact Technical Support. |
| -57 | Message could not be sent | The socket was unable to send the TCP or UDP message. | • Check the station IP address and port number.<br>• Confirm that the device is active and accessible. Try to ping the address. |
| -58 | TCP/IP could not send all bytes | The TCP/IP stack was not able to send all bytes to destination. | • Check the station IP address, port number and/or ID number.<br>• Confirm that the device is active and accessible.<br>• Try to ping the address. |

| Status Code | Description | Possible Causes | Procedure To Solve |
|---|---|---|---|
| -60 | Error to establish TCP/IP connection | Error while establishing a TCP/IP connection with the slave device. Possibly incorrect IP address or port number in the specified station. | • Check the station IP address, port number and/or ID number.<br>• Confirm that the device is active and accessible.<br>• Try to ping the address. |
| -61 | TCP/IP socket error | The TCP/IP connection has been closed by the device. | Confirm that the device is active and accessible. Try to ping the address. |
| -62 | UDP/IP receive call returned error | The UDP socket is in error. | • Check the station IP address, port number and/or ID number.<br>• Confirm that the device is active and accessible.<br>• Try to ping the address. |
| -63 | UDP/IP error initializing | The UDP socket initialization failed. | Confirm that the operating system supports UDP sockets. |
| -64 | UDP/IP receive call returned error | The UDP socket is in error. | • Check the station IP address, port number and/or ID number.<br>• Confirm that the device is active and accessible.<br>• Try to ping the address. |
| -65 | UDP/IP bind error, port number may already be in use | The driver was not able to bind the UDP port. | • Check the port number used by the driver.<br>• Check for other programs that might be bound to the UDP port. |
| -66 | Unlicensed driver | Your runtime license does not allow you to run this driver. | Contact your EcoStruxure Machine SCADA Expert 2020 software distributor. |

## Notes

For more information about driver-specific status codes, see the documentation for that driver: on the **Help** tab of the ribbon, in the **Documentation** group, click **Communication Drivers**.

## Tag Integration

Tag Integration is an enhanced framework for communication with third-party applications and devices.

Tag Integration is built on the same communication drivers that are described in the Drivers section, but instead of manually configuring driver worksheets to associate project tags with device registers, you can use the Object Finder to browse a Tag Integration source and then import device registers directly into your project.

Device registers imported in this way appear as integrated tags in your project's Shared Database folder, and they count against your project's tag limit as determined by its target system. Integrated tags are "live", which means they are continuously and bilaterally updated during project run time as long as the Tag Integration source is also running and connected. In most cases, you can use integrated tags in the same ways that you would normally use project tags you created.

Tag Integration is available only for certain applications and devices, because additional work is required to upgrade a traditional communication driver to support this feature. Many of the drivers included with this software can be upgraded, however, so if the one you want is not listed in the Tag Integration settings, please contact your software distributor and ask about custom development.

Tag Integration is configured in the Communication tab of your project settings.

**Tip:**

By default, the project runtime server will update integrated tags every 600 milliseconds, which is the rate at which **BlinkSlow** toggles. To adjust the rate, manually edit your project file (*<project name>*.APP) to add the following entry:

```
[Options]
MainDrvAlwaysTrigger=<tag name>
```

*<tag name>* can be either another system tag (e.g., **BlinkFast**, **Second**, **Minute**) or a project tag you created. Whenever the value of that tag changes, the integrated tags are updated.

This works because the project runtime server automatically creates a virtual Main Driver Sheet to manage integrated tags. The same trigger updates all Main Driver Sheets in your project, however, so be careful if you are using both Tag Integration and traditional communication drivers to communicate with devices. (Standard Driver Sheets have separate, configurable triggers.)

### *Integrate tags from TwinCAT*

This task describes how to add a TwinCAT PLC as a tag integration source in your EcoStruxure Machine SCADA Expert project. You can use either online or offline tag retrieval to get the tags from the target PLC. This feature supports all versions of TwinCAT up to version 3.1.

Before you begin this task, you should do the following:

- Use the TwinCAT project development software to note the AMS Net ID and runtime port number of the target PLC;
- Install and configure the TwinCAT ADS software on the computer or device that will host your EcoStruxure Machine SCADA Expert project; and
- Make sure the target PLC is actually running and available on your network before you try to communicate with it.

If you plan to use online tag retrieval — that is, if you want to get live tags while the target PLC is running — you must also install and configure the TwinCAT ADS software on the same computer that you are using to develop your EcoStruxure Machine SCADA Expert project. The software makes the computer a node on the TwinCAT AMS network, so that it can communicate with other nodes including the target PLC. For more information, see Install and configure the TwinCAT ADS software on page 536.

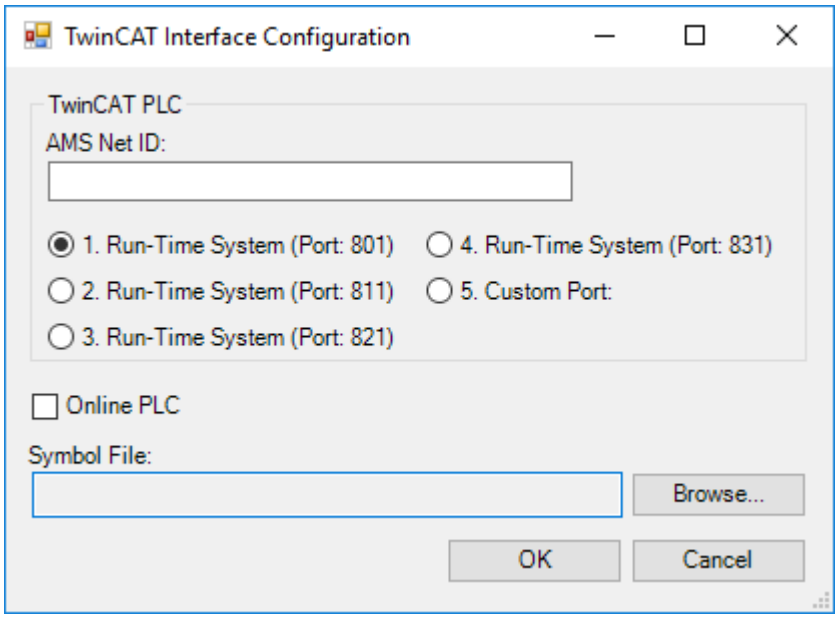
If you plan to use offline tag retrieval — that is, if you want to browse the tags while the target PLC is not running — use the TwinCAT project development software to export a symbol file from the TwinCAT project and then copy that file to your EcoStruxure Machine SCADA Expert project folder. The file provides essential information about the TwinCAT project database. EcoStruxure Machine SCADA Expert 2020 can open

TwinCAT Project Symbol files (.tpy), TwinCAT Module Class files (.tmc), and TwinCAT Module Instance files (.tmi), but .tpy files are preferred because they include nested structures. For more information about symbol files, see the TwinCAT project development software documentation.

You can set up both online and offline tag retrieval for the same tag integration source. They should not conflict with each other. For a large TwinCAT project, however, it is faster to use offline tag retrieval.

To configure TwinCAT tag integration:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.
   The *Project Settings* dialog box is displayed, with the **Communication** tab selected.
2. In the *Tag Integration* area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** list, select **Integrated** if it is not already selected.
4. In the **Provider** list, select **TwinCAT**.
5. In the **Name** box, type a name for the source.
   This name will be added as a prefix to the names of the integrated tags. For more information, see How integrated tags may be renamed in your project on page 575.
6. Click **Add**.
   The *TwinCAT Interface Configuration* dialog box is displayed.




***TwinCAT Interface Configuration dialog box***

7. In the **AMS Net ID** box, type the AMS Net ID of the target PLC.
   For example: `5.0.112.206.1.1`
8. Select the port on which the PLC runtime has been configured to run.

   For TwinCAT 2.x, you should select one of the standard ports: 801, 811, 821, 831.

   For TwinCAT 3.x, select **Custom Port** and then type the port number. The default for new TwinCAT 3.x projects is port 851.
9. If you plan to use online tag retrieval, select **Online PLC**.

   The TwinCAT system service must be installed and running on the same computer. If it is not, an alert will be displayed and you will not be able to finish configuring the tag integration.
10. If you plan to use offline tag retrieval, do the following:
    a) To the right of the **Symbol File** box, click **Browse**.
       A standard *Open* dialog box is displayed.

b) Use the *Open* dialog box to locate and select the symbol file that you previously exported from the TwinCAT project.

If the file does not appear to be available, make sure the correct file type (.tpy, .tmc, .tmi) is selected.

c) Click **Open**.

The location of the selected file is displayed in the **Symbol File** box.

11. Click **OK** to finish the configuration.

If the configuration is successful, the target PLC's tags will be immediately available in the Object Finder.

## INSTALL AND CONFIGURE THE TWINCAT ADS SOFTWARE

This section describes how to install and configure the TwinCAT Automation Device Specification (ADS) software that is required for communication with TwinCAT PLCs and runtimes.

### Download and install the ADS software on the local computer

To communicate with TwinCAT PLCs and runtimes, you must have the ADS software installed and configured on the same computer or device that hosts the EcoStruxure Machine SCADA Expert project runtime server (hereafter called "the local computer"). The ADS software allows the local computer to present itself as a TwinCAT node on the network, and then your project communicates through it.

The ADS software is installed as part of the full TwinCAT software, so if you already have the full TwinCAT software installed on the local computer, there is nothing more you need to do. Otherwise, you need to install and configure the ADS software separately.

At the time this document was written, you could download the ADS software installer from the following location: www.beckhoff.com/english.asp?twincat/tc1000.htm

After you download the installer, run it and follow the instructions. You will need to restart the local computer to finish the installation, and when you do, the software will run automatically. By default, the software is installed at: `C:\TwinCAT\`

Your use of the ADS software is subject to the License Agreement that is installed with the software. For more information about the License Agreement, please contact Beckhoff.

### Add an AMS route between the local computer and the target

To establish communication between the local computer and a target PLC or runtime, you need to add an AMS route between the two. This can be done on either the local computer or the target, as long as both have valid AMS Net IDs.

Each TwinCAT node on the network — in other words, each PLC, runtime, or computer that has the ADS software installed — has a unique AMS Net ID that consists of six numeric values separated by periods (e.g., `5.7.46.126.1.1`). When you install the ADS software on a computer, that computer is given a default AMS Net ID based on the computer's IP address. The AMS Net ID is separate from the IP address, however, and if you change the IP address, the AMS Net ID is not updated to match. You can manually change the AMS Net ID, if necessary.

To add the AMS route on the local computer:

1. In the notification area of the Windows taskbar, right-click the **TwinCAT** icon, and then on the shortcut menu, click **Router** > **Edit Routes**. (You might need to expand the notification area if the **TwinCAT** icon is hidden.) The *TwinCAT Static Routes* dialog box is displayed.
2. In the *TwinCAT Static Routes* dialog box, click **Add**. The *Add Route* dialog box is displayed.
3. If the target is located on the same network as the local computer, you should be able to select it:
   a. Click **Broadcast Search** to get a list of targets that broadcast their presence on the network.
   b. Select your target in the list. The route settings are automatically configured for the selected target.
4. If the target is not located on the same network as the local computer, you need to manually configure the route settings:
   a. In the **Route Name (Target)** box, type a name for the target. This is the name that will be displayed in the local computer's list of routes, after you finish adding the route.

b. In the **Route Name (Remote)** box, type a name for the local computer. This is the name that will be displayed in the target's list of routes, after you finish adding the route. The default name is the local computer's host name, but you can change it if necessary.

c. In the **AmsNetID** box, type the target's ID.

   If you do not know the target's ID, either use the TwinCAT programming software to get it or use **Broadcast Search** on another computer on the target's network.

d. In the **Transport Type** list, select the network's transport type or protocol. In most cases, you should select **TCP_IP** (i.e., TCP/IP). For all other options, please contact Beckhoff.

e. In the **Address Info** box, type the host name or IP address of the target, and then below the box, make sure the corresponding option — **Host Name** or **IP Address** — is selected.

> **Tip:** You can use the `ping` command, at the Windows command prompt, to confirm that the specified host name or IP address is valid and accessible.

5. Click **Add Route**. The *Add Route* dialog box is closed, and the route is added to the local computer's list of routes.
6. Close the *TwinCAT Static Routes* dialog box.

Alternatively, if you want to add the AMS route on the target, see the manufacturer's documentation for that PLC or runtime.

### Test the AMS route that you added

After you have added the AMS route between the local computer and the target, you should test the route itself to make sure they can communicate with each other. To test the route:

1. On the local computer, locate and run the ADS test program (`TcAdsTest.exe`). The *TcAdsTest* window is displayed.

> **Tip:** There are three copies of `TcAdsTest.exe` included in the ADS software. If the software was installed at its default location, the three copies should be located at:
>
> - `C:\TwinCAT\AdsApi\TcAdsDll\TcAdsTest.exe`
> - `C:\TwinCAT\AdsApi\TcAdsTest\TcAdsTest.exe`
> - `C:\TwinCAT\Common32\TcAdsTest.exe`
>
> All three copies function the same, so you can use any one of them.

2. In the *TcAdsTest* window, click **AdsPortOpen**. An alert message is displayed to inform you that the computer's ADS port has been opened for communication.
3. In the *TcAdsTest* window, click **Test**. The *Test* window is displayed.
4. In the *Test* window, in the **AmsNetId** box, type the ID of the target.
5. Click **Start** to start the test. The number of successful operations (e.g., *n* `Successful`) should be displayed in the **Output** box, and the number should keep increasing as long as the test is running.
6. Click **Stop** to stop the test.
7. Close the *Test* window, and then close the *TcAdsTest* window.

If the test results confirm that the local computer and the target can communicate with each other, your project should also be able to communicate with the target through the ADS software.

## *Integrate tags from CoDeSys*

This task describes how to add a CoDeSys 2.x or CoDeSys 3.x project as a tag integration source in your EcoStruxure Machine SCADA Expert project.

Before you begin this task, you must do the following:

- Configure your CoDeSys project to generate a new symbol file with the correct variables. The symbol file is like an index of the CoDeSys tags that you want to integrate into your project. For more information, see

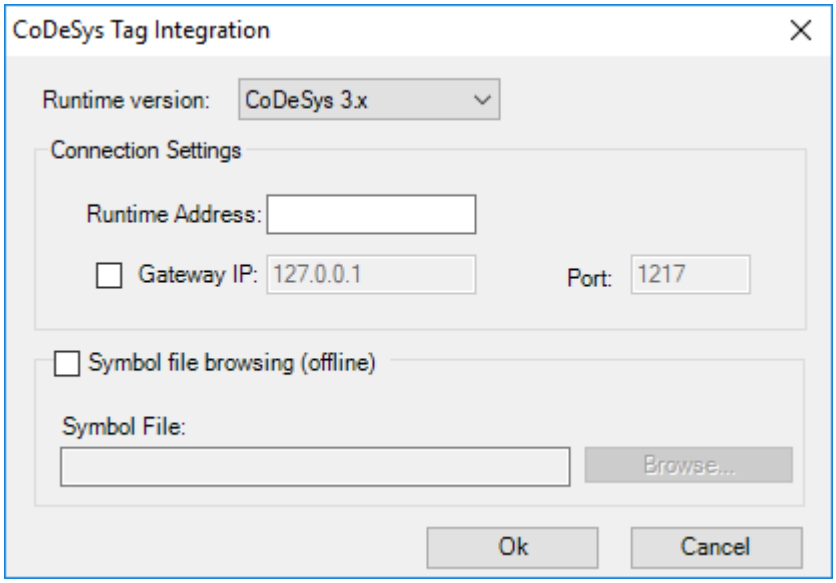
either Configure your CoDeSys 3.x project for tag integration on page 539 or Configure your CoDeSys 2.x project for tag integration on page 542.

- Rebuild your CoDeSys project, and then send it to the PLC.
- Make sure the PLC is running and available on your network, and then note its IP address and/or runtime address.

If you want to be able to browse the CoDeSys tags offline — that is, when the PLC is not running — then you must also copy the new symbol file to your EcoStruxure Machine SCADA Expert project folder.

To integrate tags from a CoDeSys project:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.
   The *Project Settings* dialog box is displayed, with the **Communication** tab selected.
2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** list, select **Integrated** if it is not already selected.
4. In the **Provider** list, select **CoDeSys**.
5. In the **Name** box, type an appropriate name for this tag integration source.
   This name will be added as a prefix to the names of the integrated tags. For more information, see How integrated tags may be renamed in your project on page 575.
6. Click **Add**.
   The *CoDeSys Tag Integration* dialog box is displayed.
7. In the **Runtime version** list, click either **CoDeSys 3.x** or **CoDeSys 2.x**, depending on the version of your CoDeSys project.
8. If you selected **CoDeSys 3.x**, configure the corresponding settings.



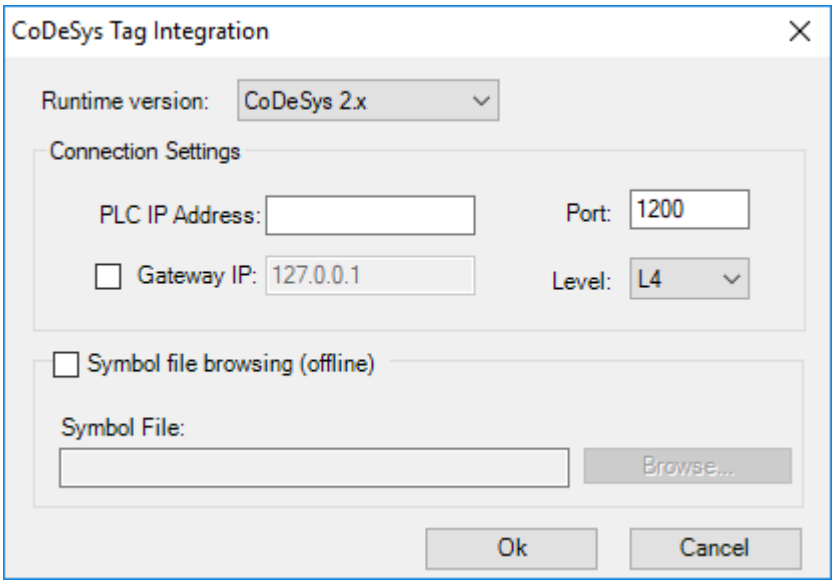

***CoDeSys Tag Integration dialog box for CoDeSys 3.x***

a) In the **Runtime Address** box, type the address of the CoDeSys runtime.
   To find this address, use the CoDeSys programming software to open your CoDeSys project.
   A typical runtime address is 0194.

b) If you are using a gateway server to manage communication with the CoDeSys runtime, select the **Gateway IP** check box, and then type the IP address and port number of the gateway server.
   Please keep in mind that the address of the gateway server is relative to your EcoStruxure Machine SCADA Expert project runtime server, so if both of them run on the same computer, you can use the default address `127.0.0.1` (i.e., localhost). Otherwise, if the gateway server and the EcoStruxure

Machine SCADA Expert project runtime server run on different computers, or if you are testing your EcoStruxure Machine SCADA Expert project on a development workstation before you send it to another computer, you should specify the actual address of the gateway server.

9. If you selected **CoDeSys 2.x**, configure the corresponding settings.




***CoDeSys Tag Integration dialog box for CoDeSys 2.x***

a) In the **PLC IP Address** and **Port** boxes, type the IP address and port number of the PLC that is running your CoDeSys project.

b) If you are using a gateway server to manage communication with the PLC, select the **Gateway IP** check box, and then type the IP address and port number of the gateway server.

Please keep in mind that the address of the gateway server is relative to your EcoStruxure Machine SCADA Expert project runtime server, so if both of them run on the same computer, you can use the default address `127.0.0.1` (i.e., localhost). Otherwise, if the gateway server and the EcoStruxure Machine SCADA Expert project runtime server run on different computers, or if you are testing your EcoStruxure Machine SCADA Expert project on a development workstation before you send it to another computer, you should specify the actual address of the gateway server.

c) In the **Level** list, select the appropriate protocol to communicate with the PLC.

In most cases, you should select **L4** (i.e., Level 4).

10. If you also want to browse the CoDeSys tags when the PLC is not running, enable offline browsing and locate the symbol file that you copied to your EcoStruxure Machine SCADA Expert project folder.

a) Select the **Symbol file browsing (offline)** check box.

b) Click **Browse**, and then locate the symbol file.

11. Click **OK** to finish the configuration and close the dialog box.

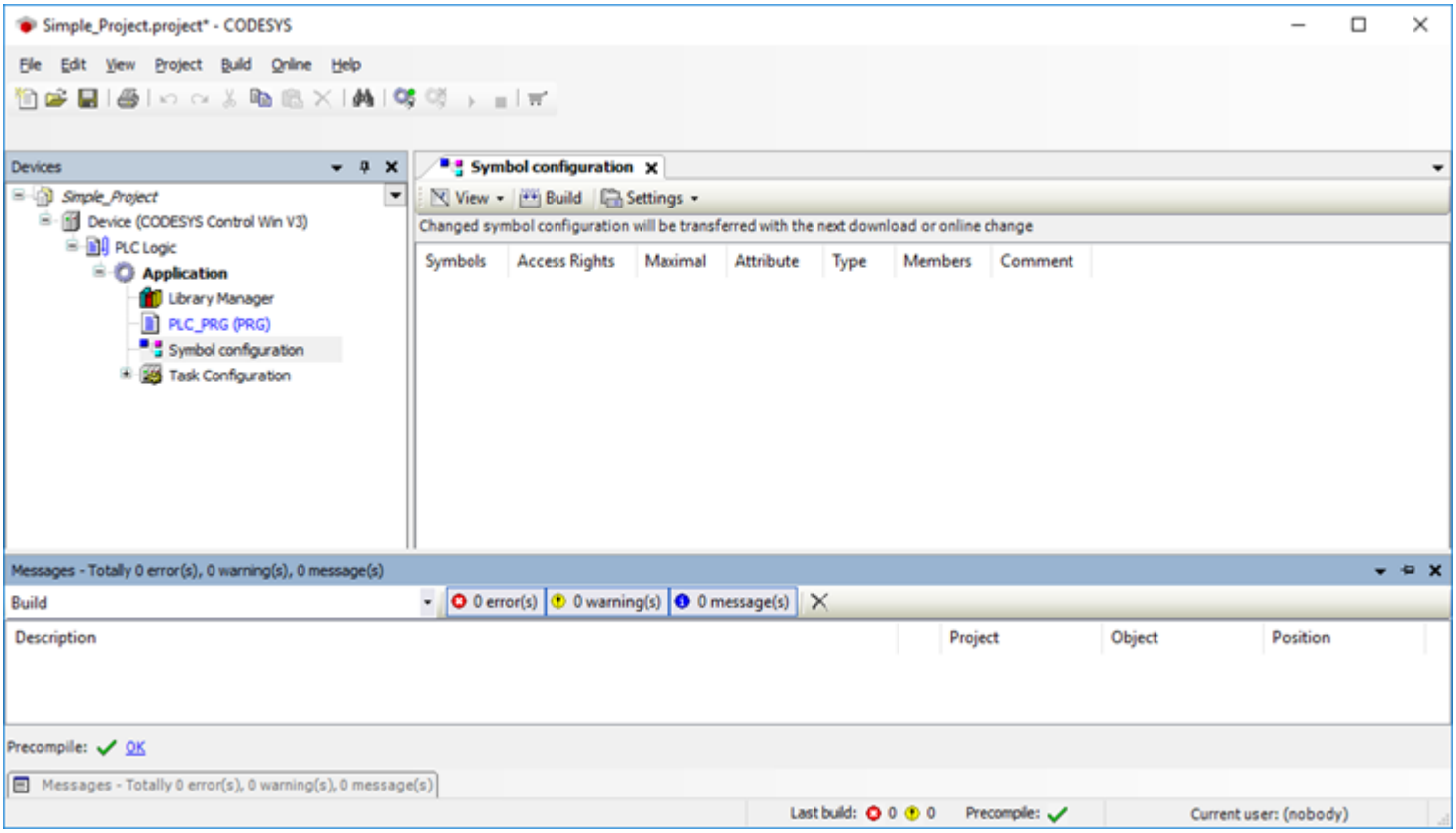
If the configuration is successful, the CoDeSys tags will be immediately available in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 573.

## CONFIGURE YOUR CODESYS 3.X PROJECT FOR TAG INTEGRATION

This task describes how to configure a CoDeSys 3.x project to communicate with external programs, such as EcoStruxure Machine SCADA Expert 2020, during runtime. It is a prerequisite to integrating CoDeSys tags into your EcoStruxure Machine SCADA Expert project.

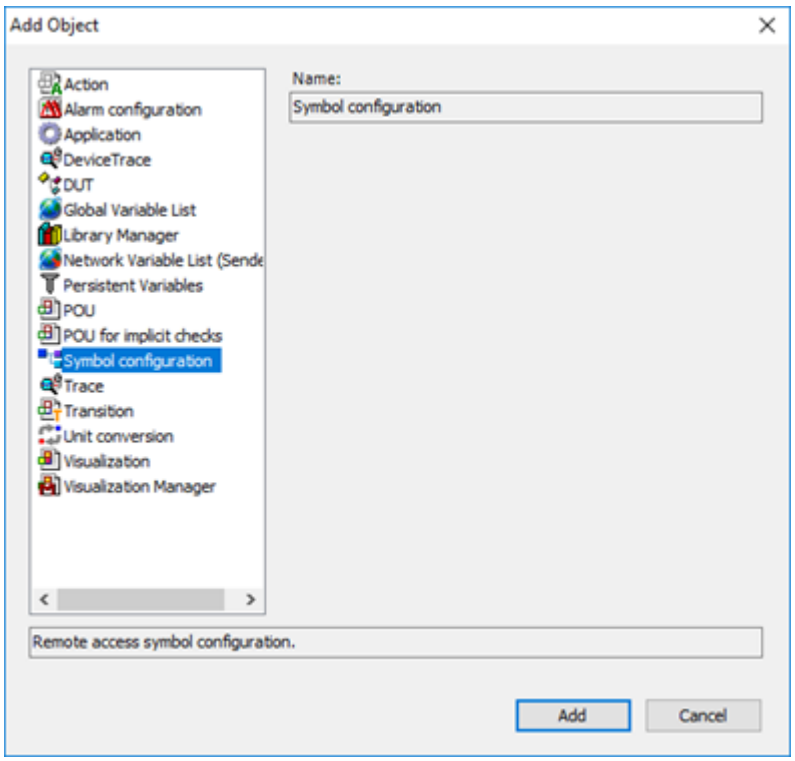
By default, the CoDeSys 3.x project development software does not generate a symbol file when you rebuild your CoDeSys project. You must add a Symbol Configuration object to your CoDeSys project and then configure the object to include the variables that you want to export to the symbol file.

1. Open your CoDeSys 3.x project.

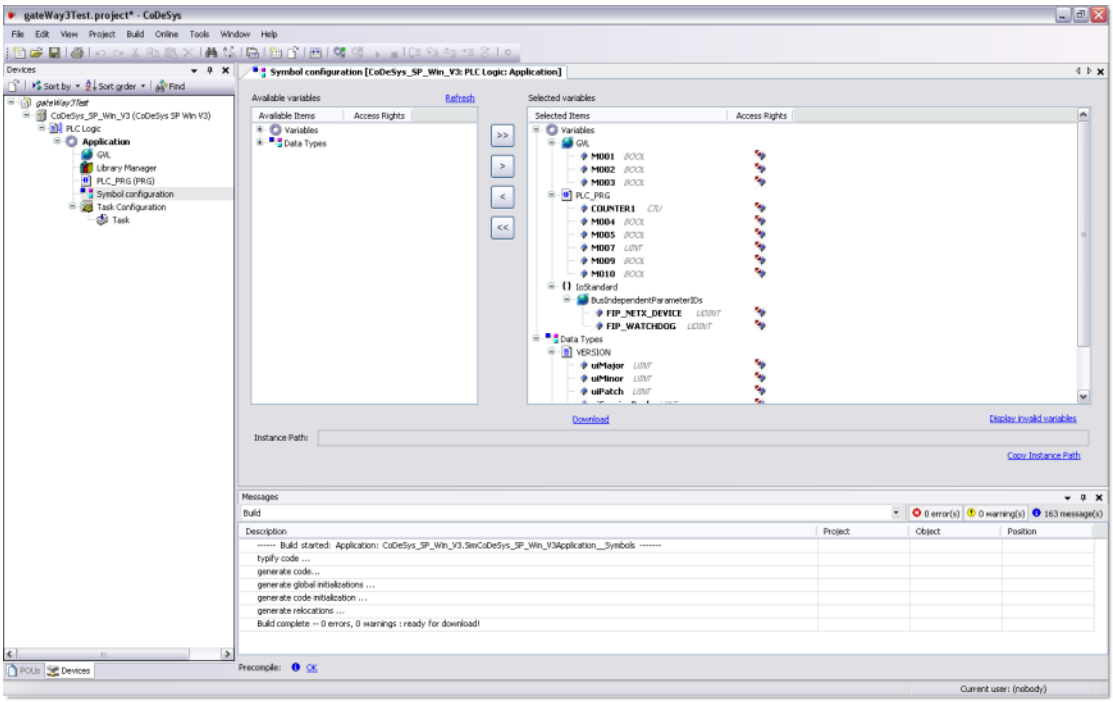



***Opening a CoDeSys 3.x project***

2. In the project explorer, right-click **Application**, and then click **Add Object** on the shortcut menu.
   The *Add Object* dialog is displayed.




3. From the list of objects, select **Symbol configuration**, and then click **Open**.
   A new Symbol Configuration object is added to your project and it is opened for editing.
4. In the Symbol Configuration object, add the variables you want to communicate with. You need to move them from the **Available variables** list on the left to the **Selected variables** list on the right.




***Moving variables in the Symbol Configuration object***

If you do not see your variables in the **Available variables** list, check the following:

- For Local Variables (POU variables), the POU containing them must be called in a Task:
  1. Add a Task Configuration object to the application.
  2. Add a Task to the Task Configuration object.
  3. Add the POU to the Task.
- For Global Variables, at least one of the variables from the Group must be used in at least one POU that is being called by one Task.

5. Close the Symbol Configuration object.
6. On the **Build** menu, click **Rebuild Application**.

Once the CoDeSys project is configured to export the selected variables, the next time you send the project to the PLC, it will include the new symbol file and your EcoStruxure Machine SCADA Expert project to be able to communicate with it.

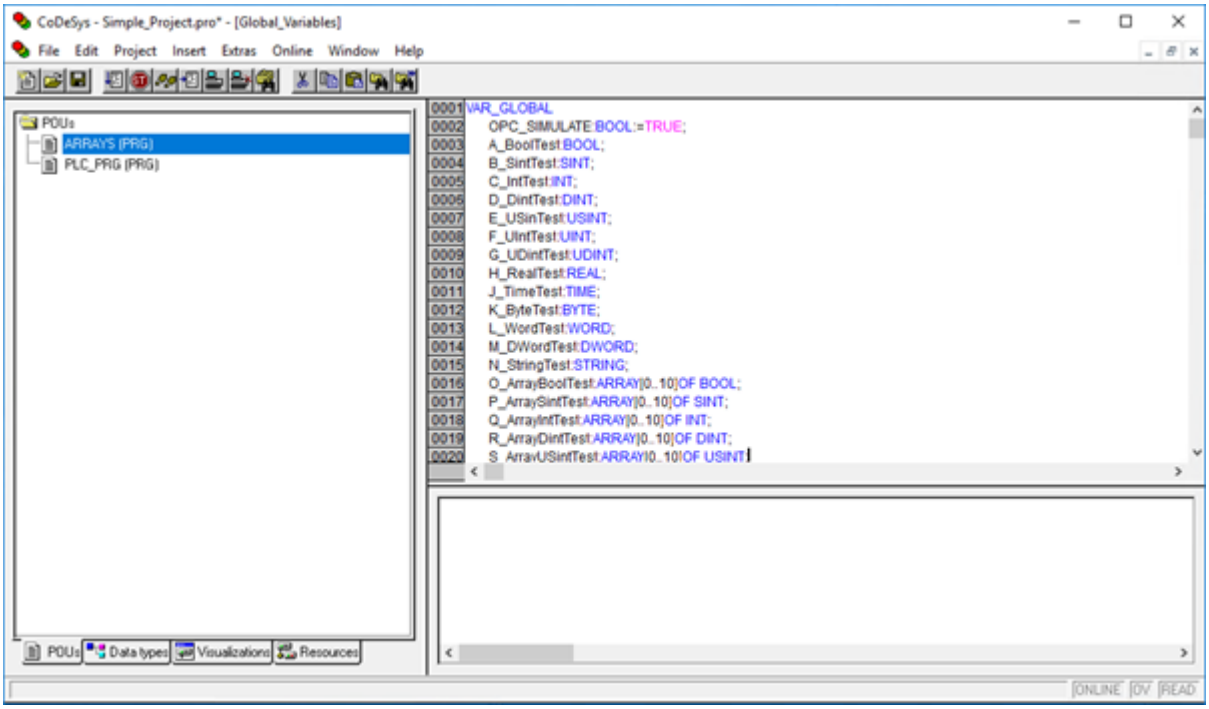
You can also copy the CoDeSys symbol file to your EcoStruxure Machine SCADA Expert project folder, which will allow you to browse the tags when the PLC is not running.

## CONFIGURE YOUR CODESYS 2.X PROJECT FOR TAG INTEGRATION

This task describes how to configure a CoDeSys 2.x project to communicate with external programs, such as EcoStruxure Machine SCADA Expert 2020, during runtime. It is a prerequisite to integrating CoDeSys tags into your EcoStruxure Machine SCADA Expert project.

The CoDeSys 2.x project development software automatically exports its project database to a symbol file every time you rebuild your CoDeSys project. However, CoDeSys exports the entire database by default, including many system and library variables that EcoStruxure Machine SCADA Expert cannot import. You must reconfigure your CoDeSys project options to export only the POUs and Global Variables and then rebuild your CoDeSys project to generate a fresh symbol file.

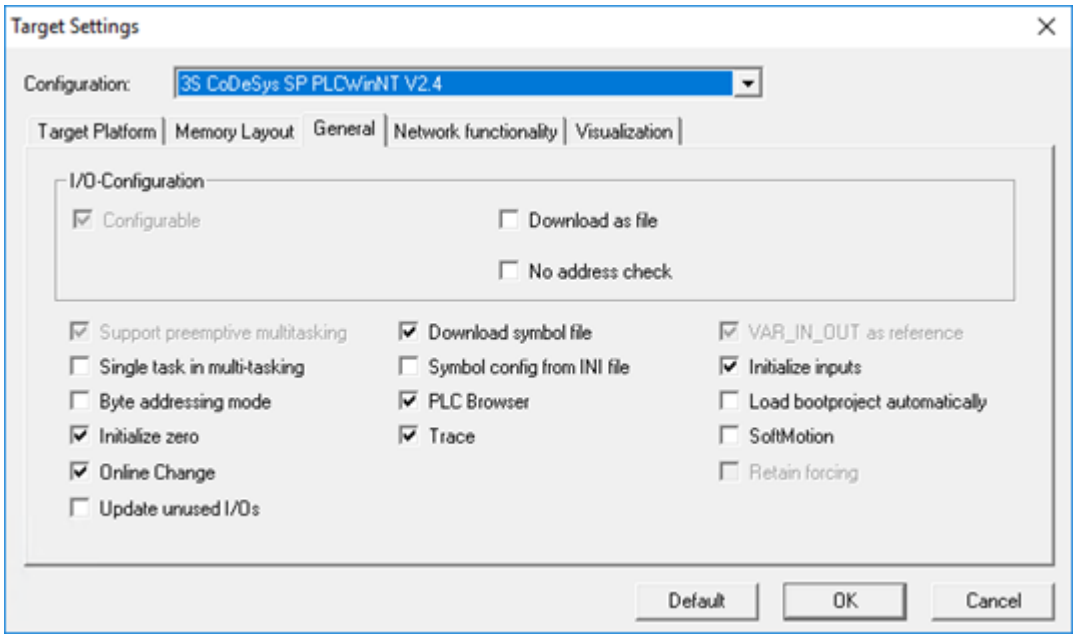
1. Open your CoDeSys 2.x project.




***Opening a CoDeSys 2.x project***

2. In the project explorer on the right, click the **Resources** tab.

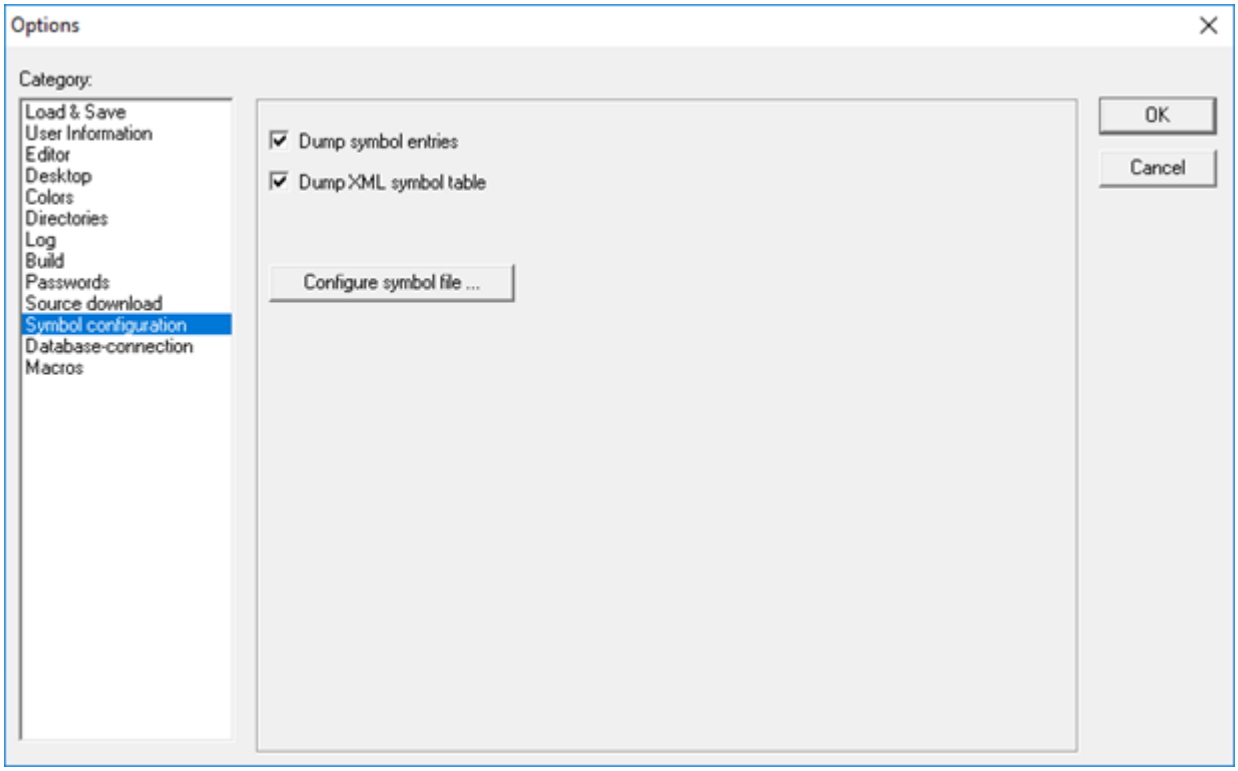
3. In the list of resources, double-click **Target Settings**.
   The *Target Settings* dialog is displayed.
4. Click the **General** tab.




***General tab of the Target Settings dialog***

5. Make sure that **Download symbol file** is selected.
6. Click **OK** to close the *Target Settings* dialog.
7. On the **Project** menu, click **Options**.
   The *Options* dialog is displayed.

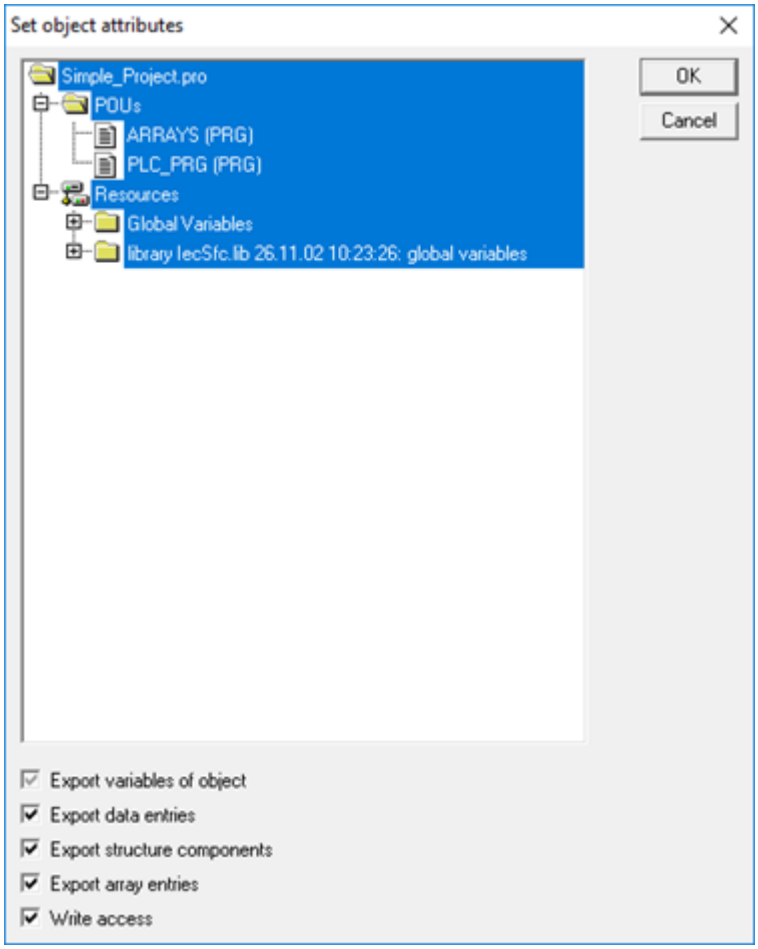
8. In the **Category** list, click **Symbol configuration**.




***Selecting "Symbol configuration"***

9. Select **Dump symbol entries** and **Dump XML symbol table**.
10. Click **Configure symbol file**.
    The *Set object attributes* dialog is displayed.
11. For the sake of expediency, you should first disable the export of all objects and then reenable only the objects that you want to export to EcoStruxure Machine SCADA Expert — typically, the POUs and Global Variables. Select all of the objects in the tree and then clear all options for them at the bottom of the dialog.

You might need to select **Export variables of object** in order to activate the other checkboxes before clearing them.




***Clearing the options for all objects***

12. Reselect only the POUs and Global Variables that you want to export to EcoStruxure Machine SCADA Expert. Do not select libraries. With the objects selected, select all of the options at the bottom of the dialog.
13. Click **OK** to close the *Set object attributes* dialog, and then click **OK** again to close the *Options* dialog.
14. On the **Project** menu, click **Clean All**.
15. On the **Project** menu, click **Rebuild All**.
    The CoDeSys development software will rebuild the project, generating a symbol file that contains only the selected POUs and Global Variables.

Once the CoDeSys project is configured to export the selected variables, the next time you send the project to the PLC, it will include the new symbol file and your EcoStruxure Machine SCADA Expert project to be able to communicate with it.

You can also copy the CoDeSys symbol file to your EcoStruxure Machine SCADA Expert project folder, which will allow you to browse the tags when the PLC is not running.

## *Integrate tags from RSLogix 5000 Family*

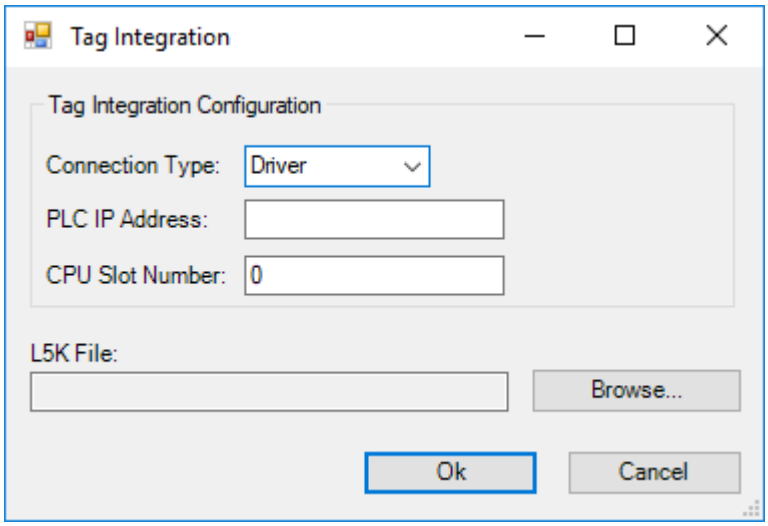
This task describes how to add an RSLogix 5000 PLC (from Allen-Bradley and Rockwell Software) as a tag integration source in your project.

Before you begin this task, you should do the following:

- Review the manufacturer's documentation for your RSLogix 5000 PLC;
- Use the PLC programming software to export a new symbol file (`*.L5K`) from your PLC program;
- Rebuild your PLC program, and then download it to the PLC; and
- Make sure the PLC is running and available on your network, and note its network address.

To add an RSLogix 5000 PLC as a tag integration source:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.
   The *Project Settings* dialog box is displayed, with the **Communication** tab selected.
2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** list, select **Integrated** if it is not already selected.
4. In the **Provider** list, select **RSLogix 5000 Family**.
5. In the **Name** box, type an appropriate name for this tag integration source.
   This name will be added as a prefix to the names of the integrated tags. For more information, see How integrated tags may be renamed in your project on page 575.
6. Click **Add**.
   The *RSLogix Tag Integration* dialog box is displayed.




7. If you want to communicate directly with the PLC using the RSLogix communication driver, then do the following:
   a) In the **Connection Type** list, click **Driver**.
   b) In the **PLC IP Address** box, type the address of the PLC.
   c) In the **CPU Slot Number** box, type the number of the PLC slot in which the CPU module is installed.
      The default is slot 0.
8. If you want to communicate with the PLC through an OPC server, then do the following:
   a) In the **Connection Type** list, click **OPC**.
   b) In the **OPC Server** list, select the type of server.
      At this time, only two OPC servers support RSLogix 5000 PLCs: Software Toolbox and Rockwell Automation's own RSLinx.
   c) In the **CPU Slot Number** box, type the number of the PLC slot in which the CPU module is installed.
      The default is slot 0.
   d) In the **Remote Server** box, type the address of the OPC server.

9. Select the symbol file that you exported from your PLC program.
   a) Click **Browse**.
   A standard *Open* dialog box is displayed.
   b) Locate and select the symbol file (`*.L5K`).
   In most cases, the file should be saved in the Config sub-folder of your project folder.
   c) Click **Open**.
   For more information, see Export symbol file for RSLogix 5000 Family on page 547.
   The selected file is displayed in the **L5K File** box.
10. Click **OK** to finish the configuration and add the source.

If the source is added successfully, then the RSLogix 5000 PLC tags will be immediately available in the Object Finder.

> **Note:** Some complex tag structures, such as arrays of nested structures and aliases of members of modules, are not supported.

### EXPORT SYMBOL FILE FOR RSLOGIX 5000 FAMILY

This task describes how to export a symbol file from an RSLogix 5000 PLC program.

The symbol file contains information about all of the tags used in the PLC program, and that information is used to add the PLC as a tag integration source in your project.

> **Note:** The procedure below was written using the original RSLogix 5000 PLC programming software, which supports up to V20. The specific steps might be different if you are using Studio 5000 Logix Designer, which currently supports V21 to V32, but the basic procedure should be the same.
> For more information, go to: https://www.rockwellautomation.com/rockwellsoftware/products/studio5000-logix-designer.page

To export the symbol file:

1. Run the PLC programming software, and then open your PLC program.
2. On the **File** menu, click **Save As**.
   A *Save As* dialog box is displayed.
3. Use the file browser to locate where you want to save the file.
   In most cases, you should save it in the Config sub-folder of your project folder at: `EcoStruxure Machine SCADA Expert 2020 Projects\<project name>\Config\`
4. In the **File name** box, type a name for the file.
5. In the **Save as type** list, click **RSLogix 5000 Import/Export File (*.L5K)**.
6. Click **Save**.

The file is saved at the specified location.

## *Integrate tags from Allen-Bradley PLC5, SLC500*

This task describes how to add an Allen-Bradley PLC2, PLC5, or SLC500 as a tag integration source in your EcoStruxure Machine SCADA Expert project.

This tag integration is based on the ABTCP driver, which communicates with Allen-Bradley devices (and others) using the DF1 protocol.

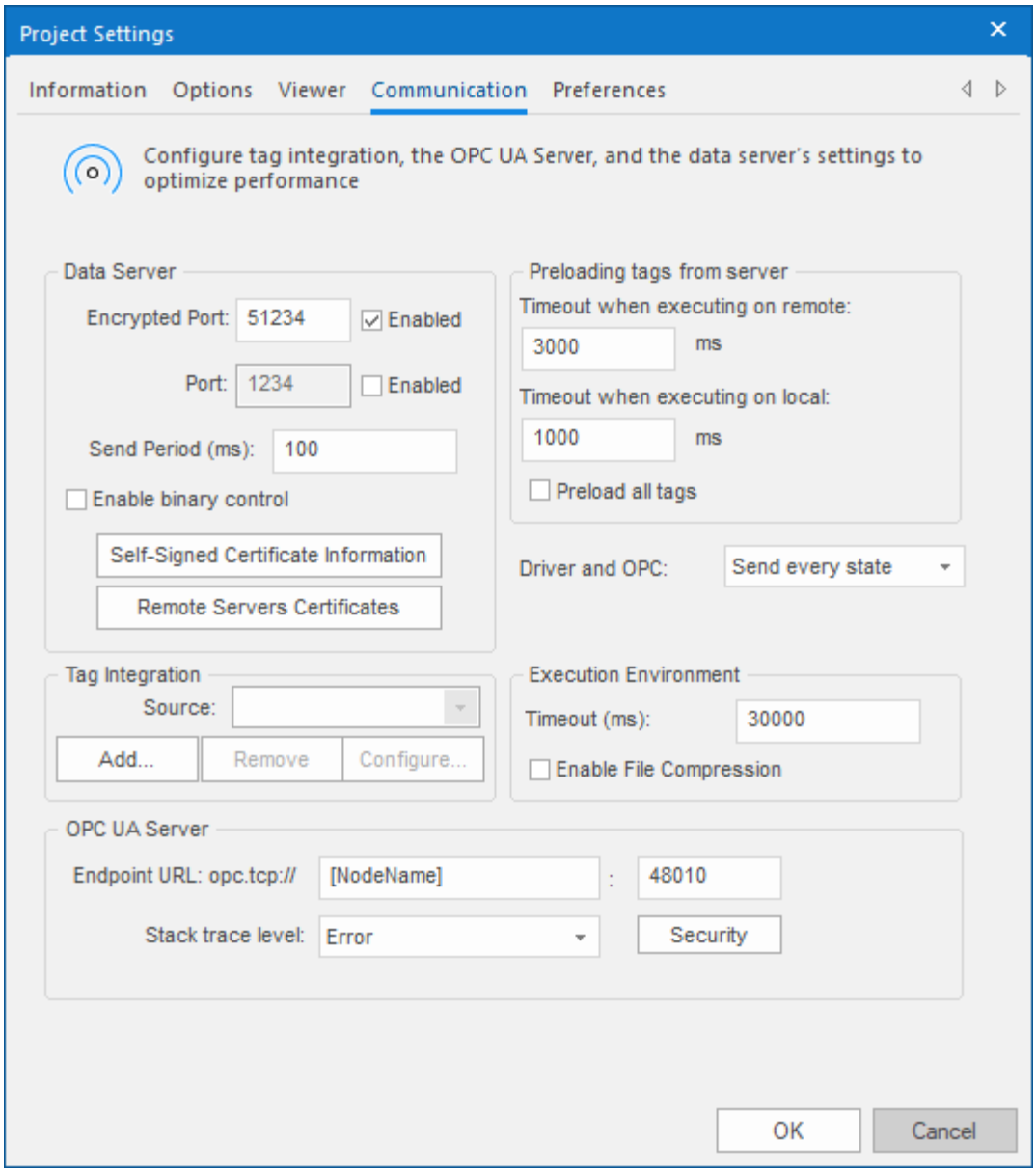
Before you begin this task, you should do the following:

- Review the manufacturer's documentation for your Allen-Bradley device;
- Read the ABTCP driver documentation (on the **Help** tab of the ribbon, in the **Documentation** group, click **Communication Drivers**, and then select **ABTCP**);

- Familiarize yourself with how memory areas — that is, groups of memory addresses — are configured on Allen-Bradley devices; and
- Make sure the source device is running and available on your network, and note its network address.

To add an Allen-Bradley device as a tag integration source:

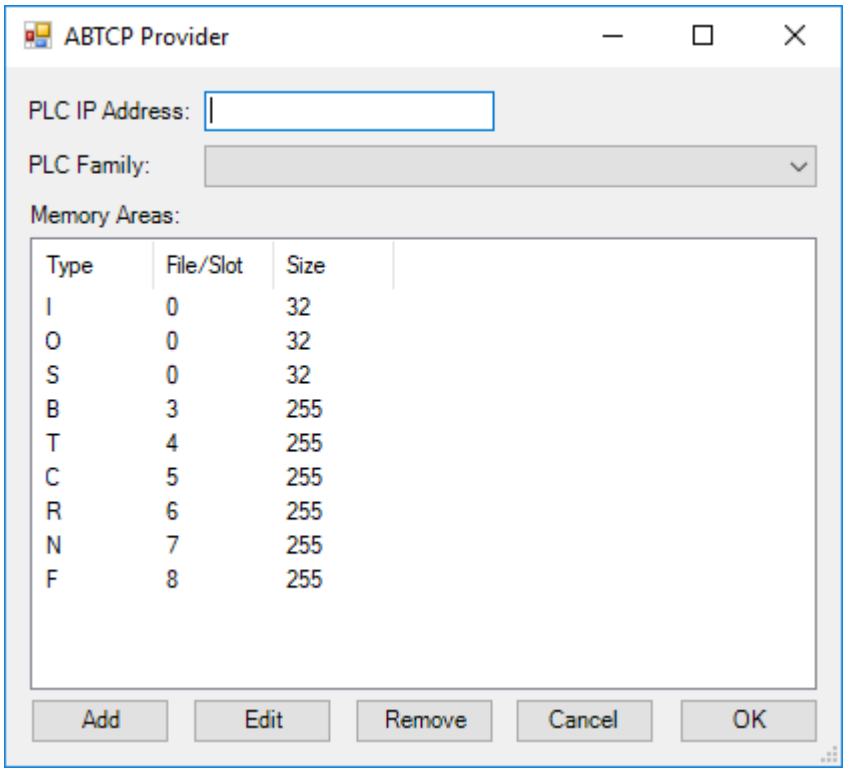
1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.
   The *Project Settings* dialog is displayed, with the **Communication** tab selected.




***Communication tab of the Project Settings dialog***

2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** list, click **Integrated**.
4. In the **Provider** list, click **Allen-Bradley PLC5, SLC500**.
5. In the **Name** box, type an appropriate name for this tag integration source.
   Note that the name will be used as a prefix on names of the integrated tags.

6. Click **Add**.
   The *ABTCP Provider* dialog box is displayed.



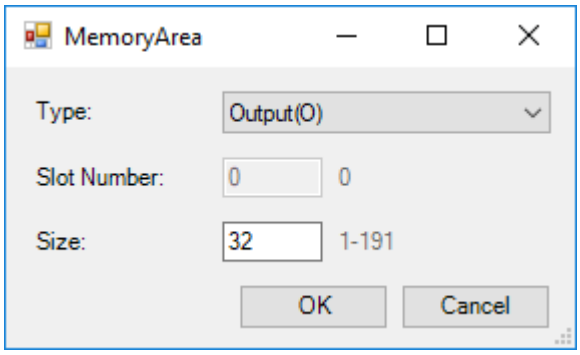

***ABTCP Provider dialog box***

7. In the **PLC IP Address** box, type the network address of the Allen-Bradley device.
   By default, port 2222 is assumed. If the device uses another port, then include it in the address.
8. In the **PLC Family** list, click the device family.

| Option | Description |
|---|---|
| **PLC2** | Allen-Bradley PLC-2 Control System |
| **PLC5** | Allen-Bradley PLC-5 Control System |
| **PLC5 as SoftPLC** | SoftPLC running a converted PLC-5 system |
| **SLC500** | Allen-Bradley SLC-500 Control System |

9. Check the default memory areas. (The defaults are the same for all device families.) If you need to add another memory area to match how you have configured your device, then do the following:
   a) Click **Add**.

The *Memory Area* dialog box is displayed.




***Memory Area dialog box***

b) In the **Type** list, click the memory address type.

c) In the **Slot Number / File Number** box, type the number of the slot (for O, I, S) or file (for B, N, T, C, R, F, ST).

> **Note:** If the family is PLC2 or PLC5 and the type is O, I, or S, then the slot number is automatically 0.

d) In the **Size** box, type the size (in bits) of the memory area.

10. Click **OK** to close the Memory Area dialog box, and then repeat the previous step as needed.
11. Click **OK** to finish the configuration and add the source.

If the source is added successfully, then the Allen-Bradley PLC tags will be immediately available in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 573.

### *Integrate tags from AutomationDirect Do-more H2 Series*

This task describes how to add a Do-more H2 Series PLC (supplied by AutomationDirect) as a tag integration source in your EcoStruxure Machine SCADA Expert project.

This tag integration is based on the DOMOR driver, which communicates with Do-more H2 Series PLCs over Ethernet using the Modbus/TCP protocol.

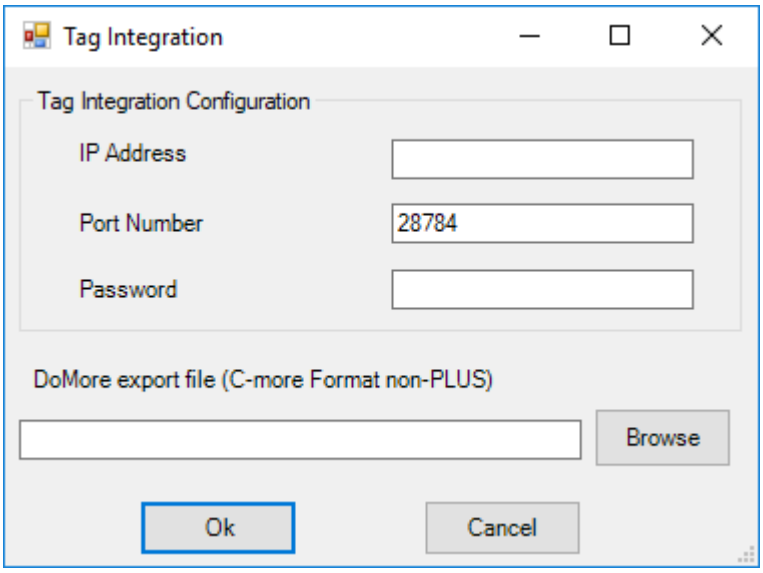
Before you begin this task, you should do the following:

- Review the manufacturer's documentation for your Do-more H2 Series PLC;
- Download and install the Do-more Designer programming software from AutomationDirect, and then use it to export your PLC program as a CSV file;
- Read the DOMOR driver documentation (on the **Help** tab of the ribbon, in the **Documentation** group, click **Communication Drivers**, and then select **DOMOR**); and
- Make sure the PLC is running and available on your network, and then note its network address.

To add a Do-more H2 Series PLC as a tag integration source:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.
   The *Project Settings* dialog box is displayed, with the **Communication** tab selected.
2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** list, select **Integrated**, if it is not already selected.
4. In the **Provider** list, select **AutomationDirect Do-more**.
5. In the **Name** box, type an appropriate name for the tag integration source.
   This name will be added as a prefix to the names of the integrated tags. For more information, see How integrated tags may be renamed in your project on page 575.

6. Click **Add**.
   The *Tag Integration* dialog box is displayed.




***Tag Integration dialog box***

7. In the **IP Address** box, type the IP address of the PLC.
8. In the **Port Number** box, type the port number on which the PLC program is running.
   The default port number for programs running on Do-more H2 Series PLCs is 28784, but that can be changed in the PLC programming software.
9. In the **Password** box, type the password for the PLC program, if it has been configured to require one.
10. In the **Do-more export file** box, specify the CSV file that you exported from your PLC program.
    a) Click **Browse**.
       A standard *Open* dialog box is displayed.
    b) Locate and select the CSV file (`*.CSV`).
       In most cases, the file should be saved in the Config sub-folder of your EcoStruxure Machine SCADA Expert project folder at: `EcoStruxure Machine SCADA Expert 2020 Projects\<project name>\Config\`
    c) Click **Open**.

    For more information, see Export CSV file for AutomationDirect Do-more on page 551.
    The selected file is displayed in the **Do-more export file** box.
11. Click **OK** to finish the configuration and add the source.

If the source is added successfully, then the Do-more H2 Series PLC tags will be immediately available in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 573.

## EXPORT CSV FILE FOR AUTOMATIONDIRECT DO-MORE

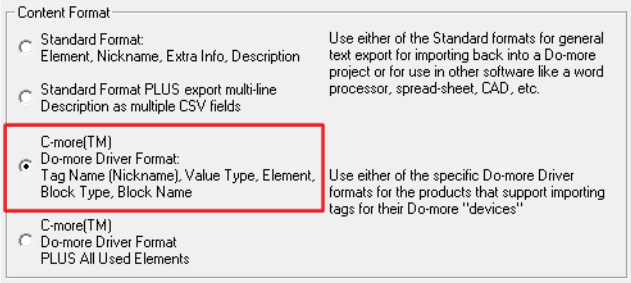
Export a CSV file from your Do-more H2 Series PLC program in order to be able to add the PLC as a tag integration source.

The CSV file (`*.CSV`) is exported from the Do-more Designer programming software. It contains information about all of the tags in your PLC program, so that the information can be imported into your EcoStruxure Machine SCADA Expert project.

> **Tip:** You can download the Do-more Designer programming software from AutomationDirect at: support.automationdirect.com/products/domore.html

To export the CSV file:

1. Run the Do-more Designer programming software, and then open the project file (`*.DMD`) for your PLC program.
2. Click **File**, and then on the **File** menu, click **Export** > **Element Documentation**.
   The *Export Documentation* dialog box is displayed.
3. In the **Content Format** group, click the **C-more™ Do-more Driver Format** radio button.




***Selecting the correct driver format***

4. Use the file browser to locate where you want to save the CSV file.
   In most cases, you should save it in the Config sub-folder of your EcoStruxure Machine SCADA Expert project folder at: `EcoStruxure Machine SCADA Expert 2020 Projects\<project name>\Config\`
5. In the **File name** box, type a name for the CSV file.
   By default, this will be the same name as your Do-more Designer project file.
6. Click **Save**.

The file is saved at the specified location, and the *Export Documentation* dialog box is closed.

## *Integrate tags from AutomationDirect Koyo*

This task describes how to add a Koyo DirectLOGIC PLC (supplied by AutomationDirect) as a tag integration source in your EcoStruxure Machine SCADA Expert project.

This tag integration is based on the KOYO driver, which communicates with Koyo DirectLOGIC PLCs over both serial communication and Ethernet.

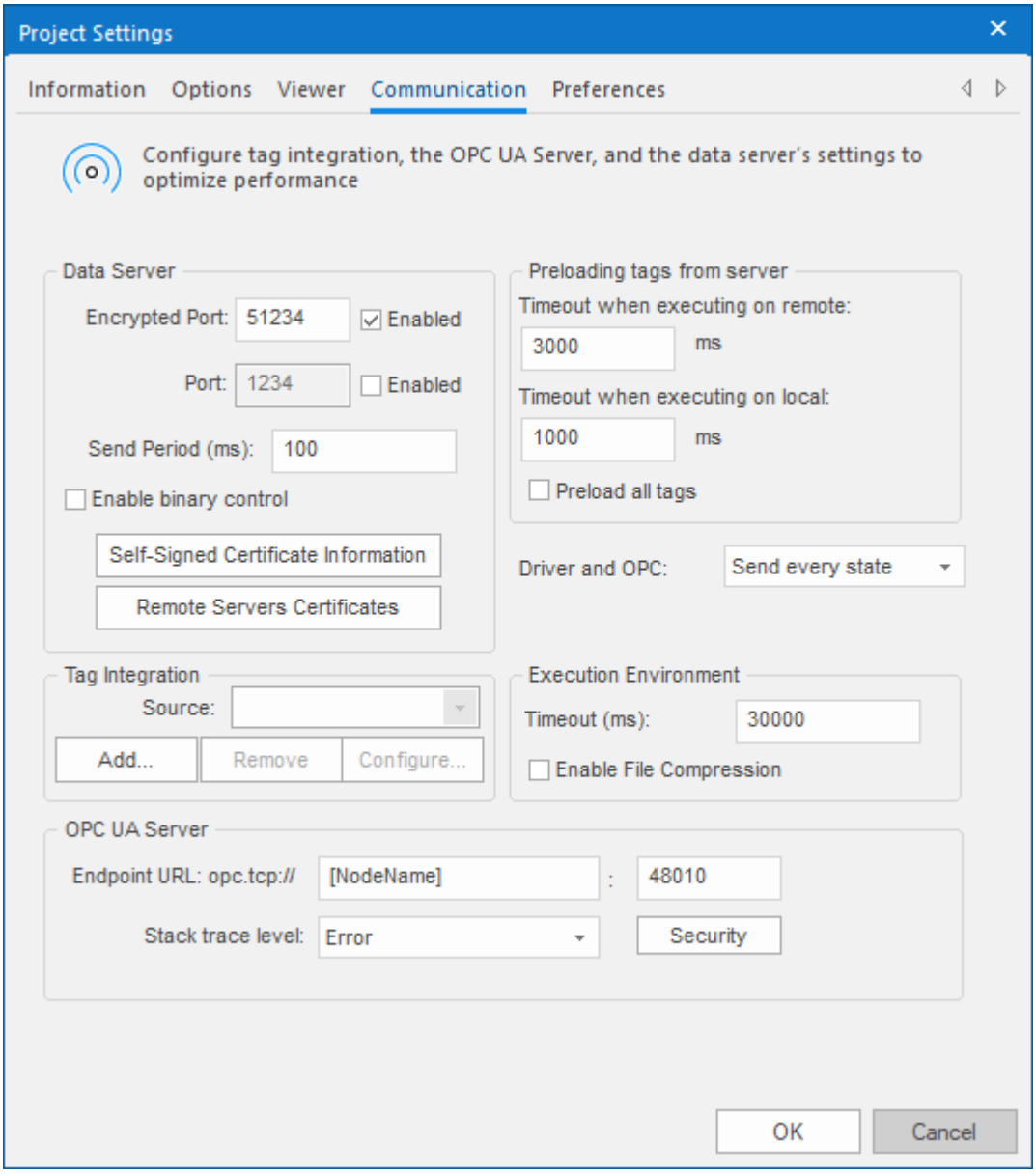
Before you begin this task, you should do the following:

- Review the manufacturer's documentation for your Koyo DirectLOGIC PLC;
- Download and install the DirectSOFT programming software from AutomationDirect, and then use it to export your PLC program as a CSV file;
- Read the KOYO driver documentation (on the **Help** tab of the ribbon, in the **Documentation** group, click **Communication Drivers**, and then select **KOYO**); and
- Make sure the source device is running and available on your network, and note its network address.

To add Koyo DirectLOGIC PLC as a tag integration source:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.

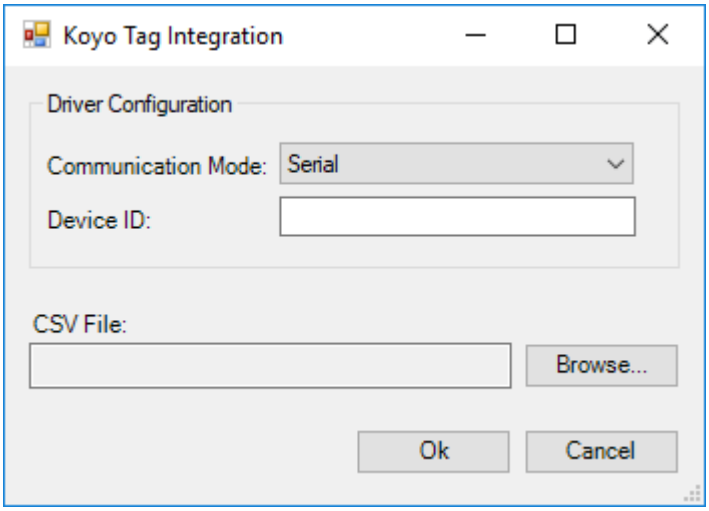
The *Project Settings* dialog is displayed, with the **Communication** tab selected.




***Communication tab of the Project Settings dialog***

2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** list, click **Integrated**.
4. In the **Provider** list, click **AutomationDirect Koyo**.
5. In the **Name** box, type an appropriate name for this tag integration source.
   Note that the name will be used as a prefix on names of the integrated tags.
6. Click **Add**.

The *Koyo Tag Integration* dialog box is displayed.




***Koyo Tag Integration dialog box***

7. In the **Communication Mode** list, click the platform of the source device.

| Option | Description |
|---|---|
| **Serial** | Direct serial communication |
| **EthernetName** | Ethernet communication by Module Name, when a broadcast router is used |
| **EthernetIP** | Ethernet communication by IP address |
| **EthernetID** | Ethernet communication by Module ID, when a broadcast router is used |

8. In the **Device** box, type the name, address, or ID of the source device.
   The format of **Device** depends on what you selected for **Communication Mode**.
9. Select the CSV file that you exported from your PLC program.
   a) Click **Browse**.
   A standard *Open* dialog box is displayed.
   b) Locate and select the CSV file (`*.CSV`).
   In most cases, the file should be saved in the Config sub-folder of your EcoStruxure Machine SCADA Expert project folder at: `EcoStruxure Machine SCADA Expert 2020 Projects\<project name>\Config\`
   c) Click **Open**.
   For more information, see Export CSV file for AutomationDirect Koyo on page 554.
   The selected file is displayed in the **CSV File** box.
10. Click **OK** to finish the configuration and add the source.

If the source is added successfully, then the Koyo DirectLOGIC PLC tags will be immediately available in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 573.

## EXPORT CSV FILE FOR AUTOMATIONDIRECT KOYO

Export a CSV file from your Koyo DirectLOGIC PLC program in order to be able to add the PLC as a tag integration source.

The CSV file (`*.CSV`) is exported from the DirectSOFT programming software. It contains information about all of the elements in your PLC program, and the information can be imported into your EcoStruxure Machine SCADA Expert project.

> **Tip:** You can download the DirectSOFT programming software from AutomationDirect at: support.automationdirect.com/products/directsoft.html

To export the file:

1. Run the DirectSOFT programming software, and then open your PLC program.
2. Click **File**, and then on the File menu, click **Export** > **Element Documentation**.
   The *Export Documentation* dialog box is displayed.
3. Use the file browser to locate where you want to save the file.
   In most cases, you should save it in the Config sub-folder of your EcoStruxure Machine SCADA Expert project folder at: `EcoStruxure Machine SCADA Expert 2020 Projects\<project name>\Config\`
4. In the **File name** box, type a name for the file.
5. Click **Save**.

The file is saved at the specified location.

### *Integrate tags from AutomationDirect P Series*

This task describes how to add an AutomationDirect Productivity Series (a.k.a. P Series) Programmable Automation Controller (PAC) as a tag integration source in your project.

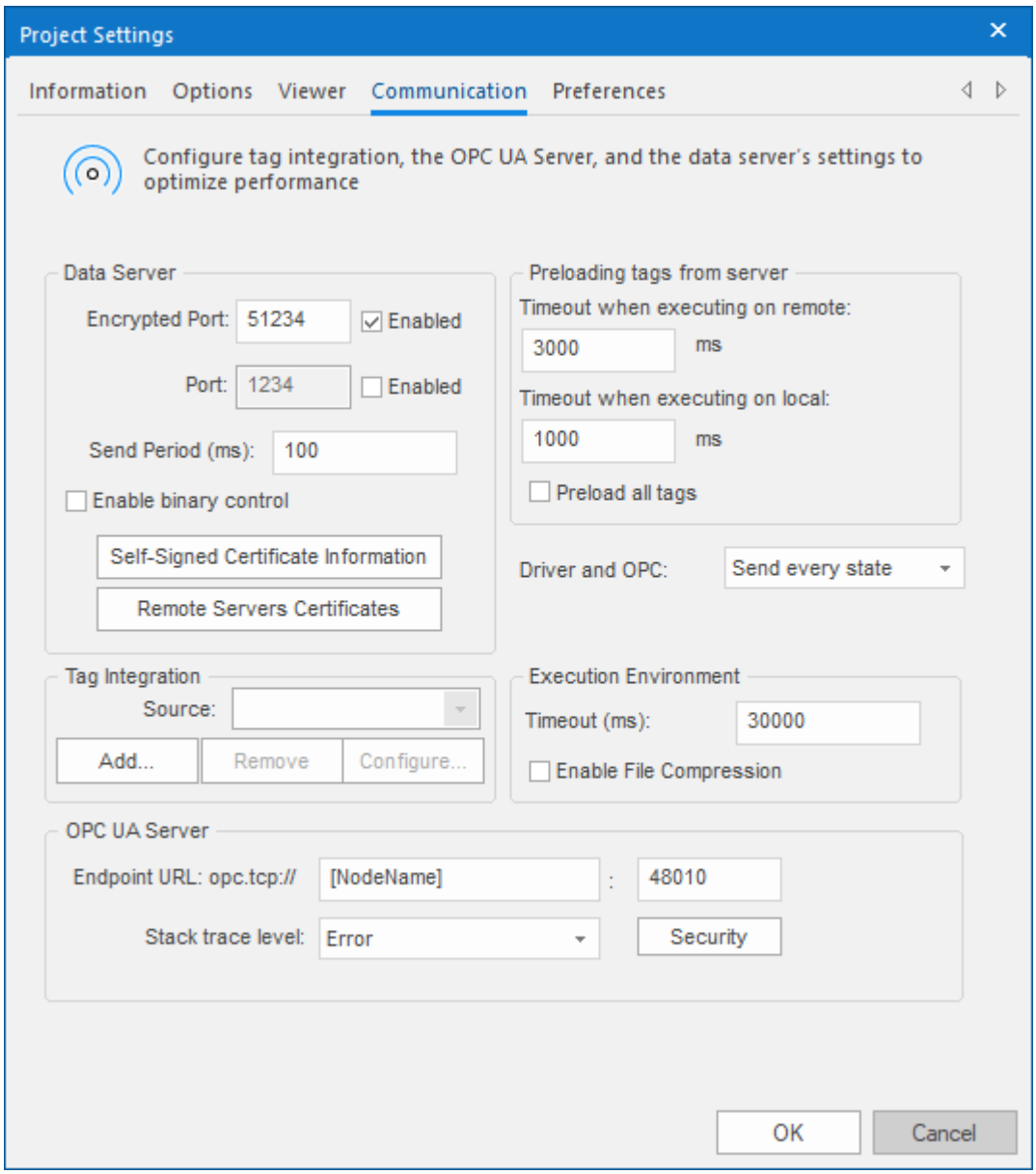
This tag integration is based on our ADPRO driver, which communicates with the PAC over Ethernet using the Modbus Extended protocol.

Before you begin this task, you should do the following:

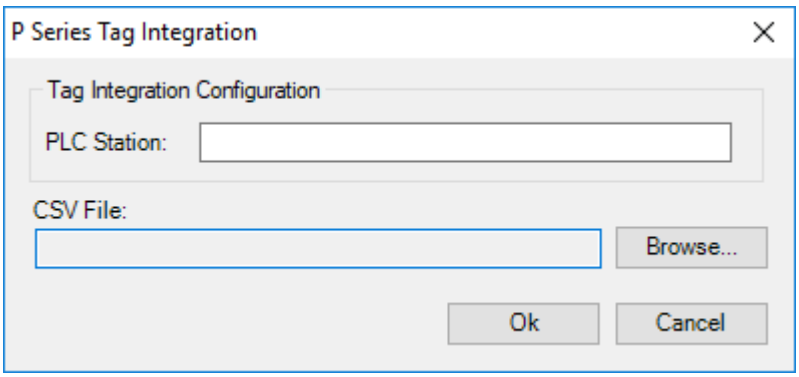
- Review the manufacturer's documentation for your AutomationDirect P Series PAC;
- Download and install AutomationDirect's Productivity Suite Programming Software, and then use it to export the PAC program's tag information as a comma-separated values (CSV) file;
- Read the documentation for the ADPRO driver (on the **Help** tab of the ribbon, in the **Documentation** group, click **Communication Drivers**, and then select **ADPRO**); and
- Make sure the PAC is running and accessible on your network, and then note its network address.

To add an AutomationDirect P Series PAC as a tag integration source:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.
   The *Project Settings* dialog is displayed, with the **Communication** tab selected.




2. Under **Tag Integration**, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** list, select **Integrated**.
4. In the **Provider** list, select **AutomationDirect P Series**.
5. In the **Name** box, type an appropriate name for the tag integration source.
   This name will be added as a prefix to the names of the integrated tags. The default is `DEV`, which is short for "device".
6. Click **Add**.
   The *P Series Tag Integration* dialog box is displayed.




7. In the **PLC Station** box, type the IP address of the PAC.
8. Select the .csv file that you exported from your PAC program:
   a) Click **Browse**.
   A standard *Open* dialog box is displayed.
   b) Locate and select the .csv file.
   In most cases, you should select the "basic" file and not the "extended" file, and that file should be located in the Config sub-folder of your project folder (e.g., `<project name>\Config\<PAC program name>_Basic.csv`).
   c) Click **Open**.
   For more information, see Export tag information from an AutomationDirect P Series PAC program on page 557.
   The selected file is displayed in the **CSV File** box.
9. Click **OK** to finish the configuration and add the source.

If the source is added successfully, the PAC's tags will be immediately available in the Object Finder in the project development environment. For more information, see Use the Object Finder to select integrated tags on page 573.

## EXPORT TAG INFORMATION FROM AN AUTOMATIONDIRECT P SERIES PAC PROGRAM

Export tag information from an AutomationDirect Productivity Series (a.k.a. P Series) PAC program, so that the PAC can be added as a tag integration source.

The tag information is exported as a comma-separated values (CSV) file, which you will subsequently import into your project. You can use AutomationDirect's Productivity Suite Programming Software to export the file, and you can download the software from the AutomationDirect website at: support.automationdirect.com/products/p3000.html

In fact, the AutomationDirect software will automatically export the same tag information as two separate .csv files:

- A "basic" file (e.g., `<PAC program name>_Basic.csv`) which has a format that is easy to import into your project but that cannot support complex data structures like classes and multi-dimensional arrays. When you import this file into your project, each item in those structures will be imported as a simple tag.
- An "extended" file (e.g., `<PAC program name>_Extended.csv`) which has a format that can support complex data structures but that cannot be imported into your project without additional work. If you want to use this file, please contact your EcoStruxure Machine SCADA Expert 2020 software distributor.

To export the tag information from the PAC program:

1. Run the Productivity Suite Programming Software, and then use it to open your PAC program (e.g, `<PAC program name>.adpro`).
2. Go to **File**, and then select **Export** > **Tags**.
   The *Export Tag Database* dialog box is displayed.
3. Click **Browse**, and then use the file browser to specify where you want to save the files.

In most cases, you should save the files in the Config sub-folder of your project folder (e.g., *<project name>*\Config).

4. In the **File name** box, type a name for the files.
   In most cases, you should type the same name as the PAC program itself. That will make it easier to indentify the files later.
5. Click **Select**.
   The file browser is closed, and the selected location is displayed in the **To File** box.
6. Select **Include I/O Tags**.
7. Click **Export**.

The .csv files are saved at the specified location (e.g., *<project name>*\Config*<PAC program name>*_Basic.csv).

### *Integrate tags from AutomationDirect PAC 3000*

This task describes how to add an AutomationDirect Productivity3000 Programmable Automation Controller (a.k.a. PAC 3000) as a tag integration source in your project.

This tag integration is based on our PAC3K driver, which communicates with the PAC over Ethernet using the Modbus Extended protocol.

> **Note:** This feature has been deprecated and is included only to maintain backward compatibility with existing projects. For all new projects, please use the ADPRO driver instead. For more information, see Integrate tags from AutomationDirect P Series on page 555.

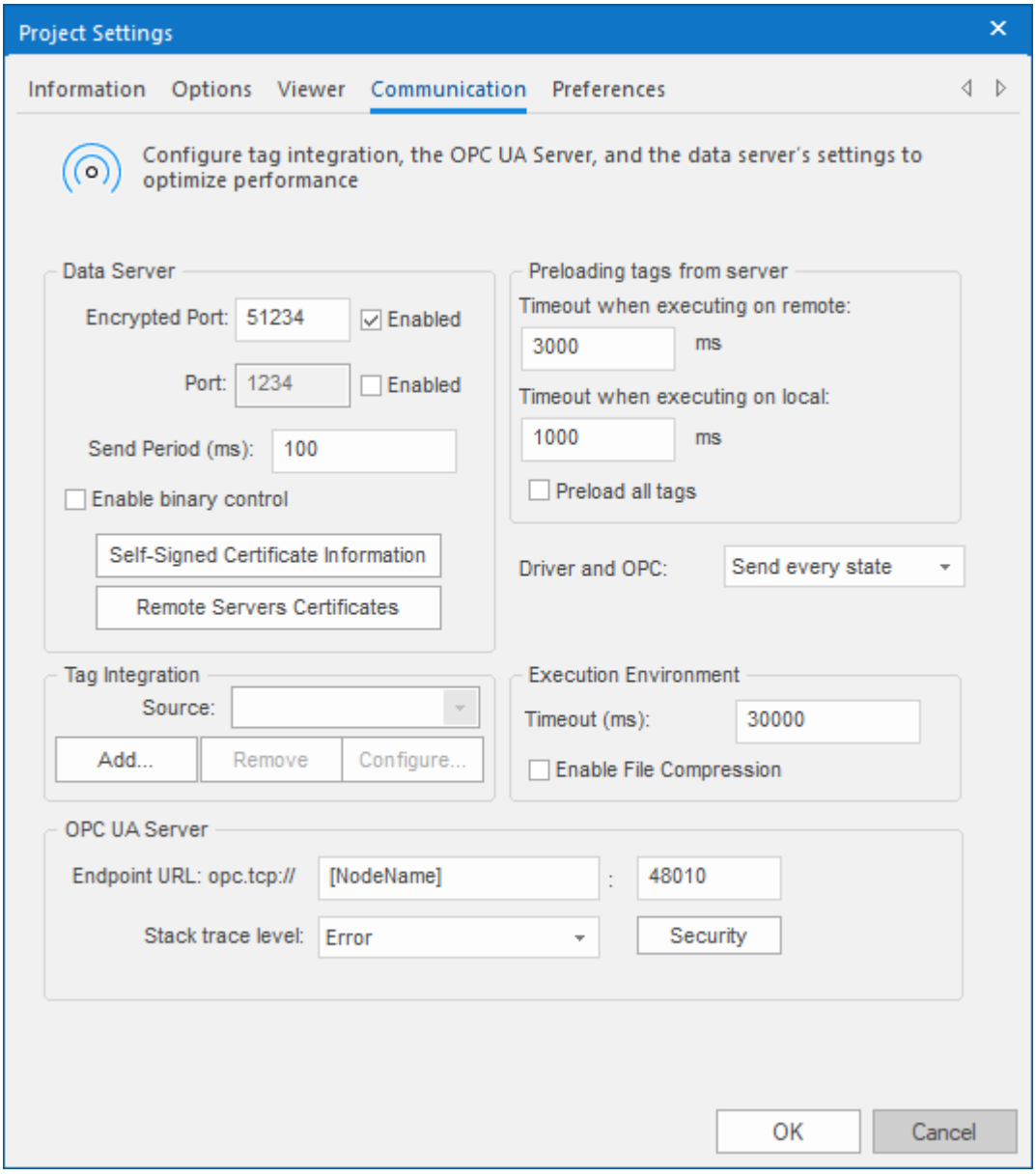
Before you begin this task, you should do the following:

- Review the manufacturer's documentation for your AutomationDirect PAC 3000;
- Download and install the Productivity Suite Programming Software from AutomationDirect, and then use it to export the PAC program's tags as a comma-separated values (CSV) file;
- Read the documentation for the PAC3K driver (on the **Help** tab of the ribbon, in the **Documentation** group, click **Communication Drivers**, and then select **PAC3K**); and
- Make sure the PAC is running and accessible on your network, and then note its network address.

To add an AutomationDirect PAC 3000 as a tag integration source:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.

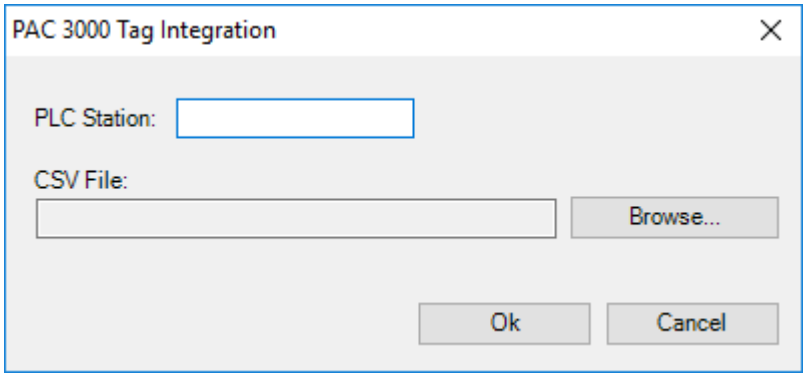
The *Project Settings* dialog is displayed, with the **Communication** tab selected.




***Communication tab of the Project Settings dialog***

2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** list, click **Integrated**.
4. In the **Provider** list, click **AutomationDirect PAC 3000**.
5. In the **Name** box, type an appropriate name for the tag integration source.
   This name will be used as a prefix on the names of the integrated tags. The default is `DEV`, which is short for "device".
6. Click **Add**.

The *PAC 3000 Tag Integration* dialog box is displayed.




***PAC 3000 Tag Integration dialog box***

7. In the **PLC Station** box, type the IP address of the PAC.
8. Select the CSV file that you exported from your PAC program:
   a) Click **Browse**.
   A standard *Open* dialog box is displayed.
   b) Locate and select the CSV file (`.csv`).
   In most cases, the file should be saved in the Config sub-folder of your project folder (e.g., *<project name>*`\Config\`*<PAC program name>*`.csv`).
   c) Click **Open**.

   For more information, see Export tag information from an AutomationDirect P Series PAC program on page 557.
   The selected file is displayed in the **CSV File** box.
9. Click **OK** to finish the configuration and add the source.

If the source is added successfully, the PAC's tags will be immediately available in the Object Finder in the Studio development environment. For more information, see Use the Object Finder to select integrated tags on page 573.

## EXPORT TAG INFORMATION FROM AN AUTOMATIONDIRECT P SERIES PAC PROGRAM

Export tag information from an AutomationDirect Productivity Series (a.k.a. P Series) PAC program, so that the PAC can be added as a tag integration source.

The tag information is exported as a comma-separated values (CSV) file, which you will subsequently import into your project. You can use AutomationDirect's Productivity Suite Programming Software to export the file, and you can download the software from the AutomationDirect website at: support.automationdirect.com/products/p3000.html

In fact, the AutomationDirect software will automatically export the same tag information as two separate .csv files:

- A "basic" file (e.g., *<PAC program name>*`_Basic.csv`) which has a format that is easy to import into your project but that cannot support complex data structures like classes and multi-dimensional arrays. When you import this file into your project, each item in those structures will be imported as a simple tag.
- An "extended" file (e.g., *<PAC program name>*`_Extended.csv`) which has a format that can support complex data structures but that cannot be imported into your project without additional work. If you want to use this file, please contact your EcoStruxure Machine SCADA Expert 2020 software distributor.

To export the tag information from the PAC program:

1. Run the Productivity Suite Programming Software, and then use it to open your PAC program (e.g, *<PAC program name>*`.adpro`).
2. Go to **File**, and then select **Export** > **Tags**.
   The *Export Tag Database* dialog box is displayed.

3. Click **Browse**, and then use the file browser to specify where you want to save the files.

   In most cases, you should save the files in the Config sub-folder of your project folder (e.g., *<project name>*\Config).

4. In the **File name** box, type a name for the files.

   In most cases, you should type the same name as the PAC program itself. That will make it easier to indentify the files later.

5. Click **Select**.
   The file browser is closed, and the selected location is displayed in the **To File** box.

6. Select **Include I/O Tags**.

7. Click **Export**.

The .csv files are saved at the specified location (e.g., *<project name>*\Config*<PAC program name>*_Basic.csv).

### *Add GE PACSystems RX3i as a tag integration source*

This task describes how to add a GE PACSystems RX3i programmable automation controller (PAC) as a tag integration source in your project.

This tag integration is based on our SRTP driver, which communicates with the target PAC over Ethernet using the Service Request Transfer Protocol (SRTP).

Before you begin this task, you should do the following:

- Read the SRTP driver documentation (on the **Help** tab of the ribbon, in the **Documentation** group, click **Communication Drivers**, and then select **SRTP**);
- Review the manufacturer's documentation for your GE PACSystems RX3i PAC;
- Note the network address of the target PAC; and
- Use GE Automation's Proficy Machine Edition software to export the target PAC's variables as an .xml file. For more information, see Export .xml file for GE PACSystems RX3i tag integration on page 564.

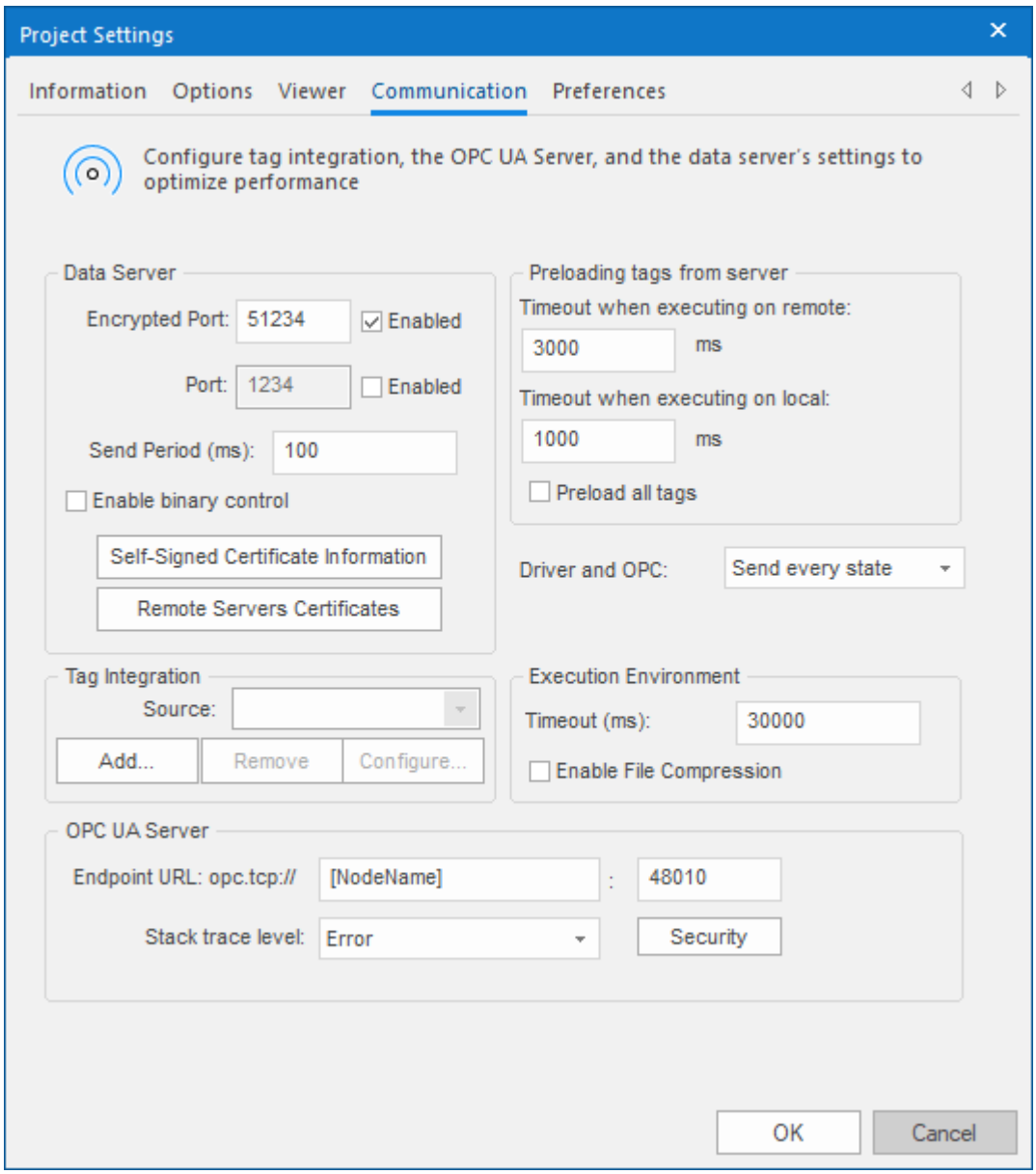
The target PAC does not need to be running when you add it as a tag integration source in your project, because live browsing of tags is not supported for this type of tag integration. Your project will use the information contained in the .xml file to reference the PAC's variables.

The target PAC does need to be running during project run time, however, so that your project can communicate with it.

To add a GE PACSystems RX3i PAC as a tag integration source:

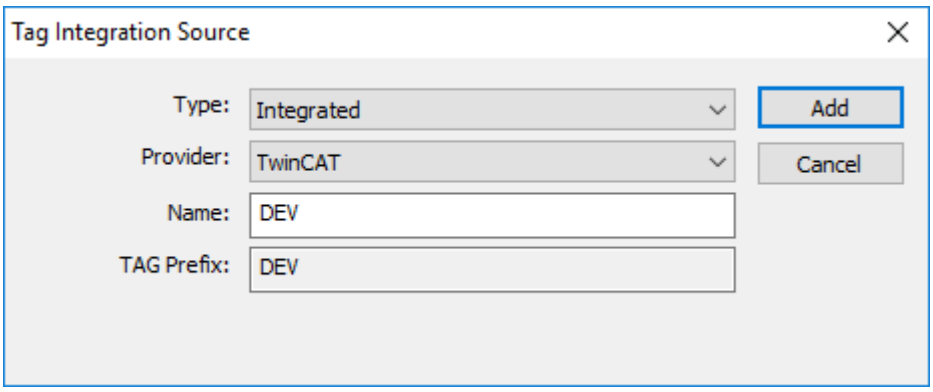
1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.

   The *Project Settings* dialog is displayed, with the **Communication** tab selected.

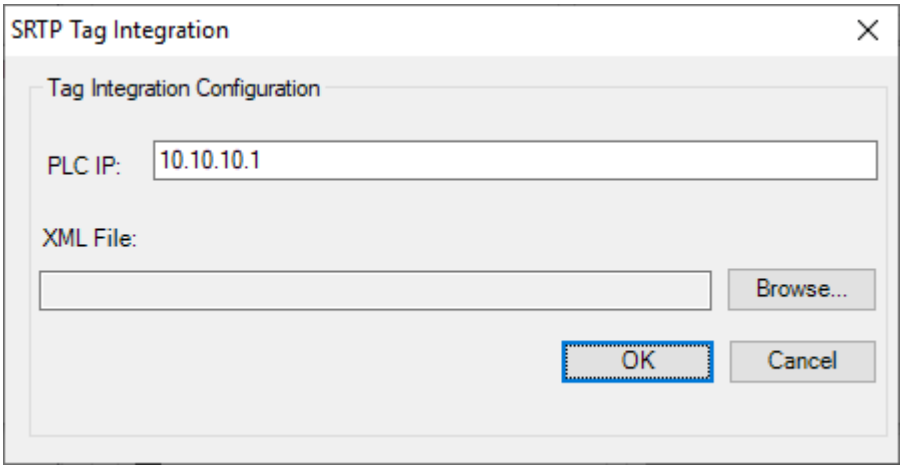



2. Under **Tag Integration**, click **Add**.

   The *Tag Integration Source* dialog box is displayed.




3. In the **Type** list, select **Integrated** if it is not already selected.
4. In the **Provider** list, select **GE PACSystems SRTP Symbolic Tags**.
5. In the **Name** box, type an appropriate name for the tag integration source.
   This name will be added as a prefix to the names of the integrated tags. The default is `DEV`, which is short for "device".
6. Click **Add**.
   The *SRTP Tag Integration* dialog box is displayed.




7. In the **PLC IP** box, type the network address of the target PAC.
8. In the **XML File** box, select the .xml file that you exported from the target PAC's program:
   a) Click **Browse**, to the right of the box.
   A standard *Open* dialog box is displayed.
   b) Locate and select the .xml file.
   In most cases, the file should be saved in the Config sub-folder of your project folder.
   c) Click **Open**.
   The location of the selected file is displayed in the box.
9. Click **OK** to finish the configuration and add the source.

If the source is added successfully, the target PAC's variables will be immediately available in the Object Finder in the project development environment. For more information, see Use the Object Finder to select integrated tags on page 573.

Live browsing is not supported for this type of tag integration, which means that if the target PAC's program is updated and its variables are changed, those changes will not be reflected in the tag integration. To see the changes, you will need to export the variables again as a new .xml file. The new file should be saved in the same location as the old one; if it is not, you will also need to update the **XML File** setting in the tag integration.

### EXPORT .XML FILE FOR GE PACSYSTEMS RX3I TAG INTEGRATION

This task describes how to export information about the variables used in a GE PACSystems RX3i PAC program, so that the target PAC can be added as a tag integration source in your project.

The information is exported as an .xml file, which you will subsequently import into your project. You can use GE Automation's Proficy Machine Edition software to export the .xml file, and you can download that software from GE's website at: https://digitalsupport.ge.com/en_US/Article/Proficy-Machine-Edition-Landing-Page

To export the .xml file:

1. Run the Proficy Machine Edition software, and then use it to open the program for the target PAC.
2. On the **Variables** tab of the ribbon, in the **Variable Tools** group, click **Export**.
   A standard *Save* dialog box is displayed.
3. Use the file explorer to select the location where you want to save the file, type the file name, and then click **Save**.

   In most cases, you should save the file in the Config sub-folder of your project folder.

   For example: `<project name>\Config\GE_RX3i.xml`
   After you click **Save**, the *Export Variables* dialog box is displayed.
4. In the **Target to Export From** list, select the target PAC.
   For example, **GE_RX3i**.
5. Review the export options, and then click **OK**.

The .xml file is saved at the specified location.

## *Integrate tags from Schneider Unity Modbus*

This task describes how to add a Schneider Modicon M340 PAC or Modicon Premium PAC as a tag integration source in your EcoStruxure Machine SCADA Expert project.

This tag integration is based on the SCHNE driver, which communicates with Schneider Modicon devices using the Modbus protocol over Ethernet.

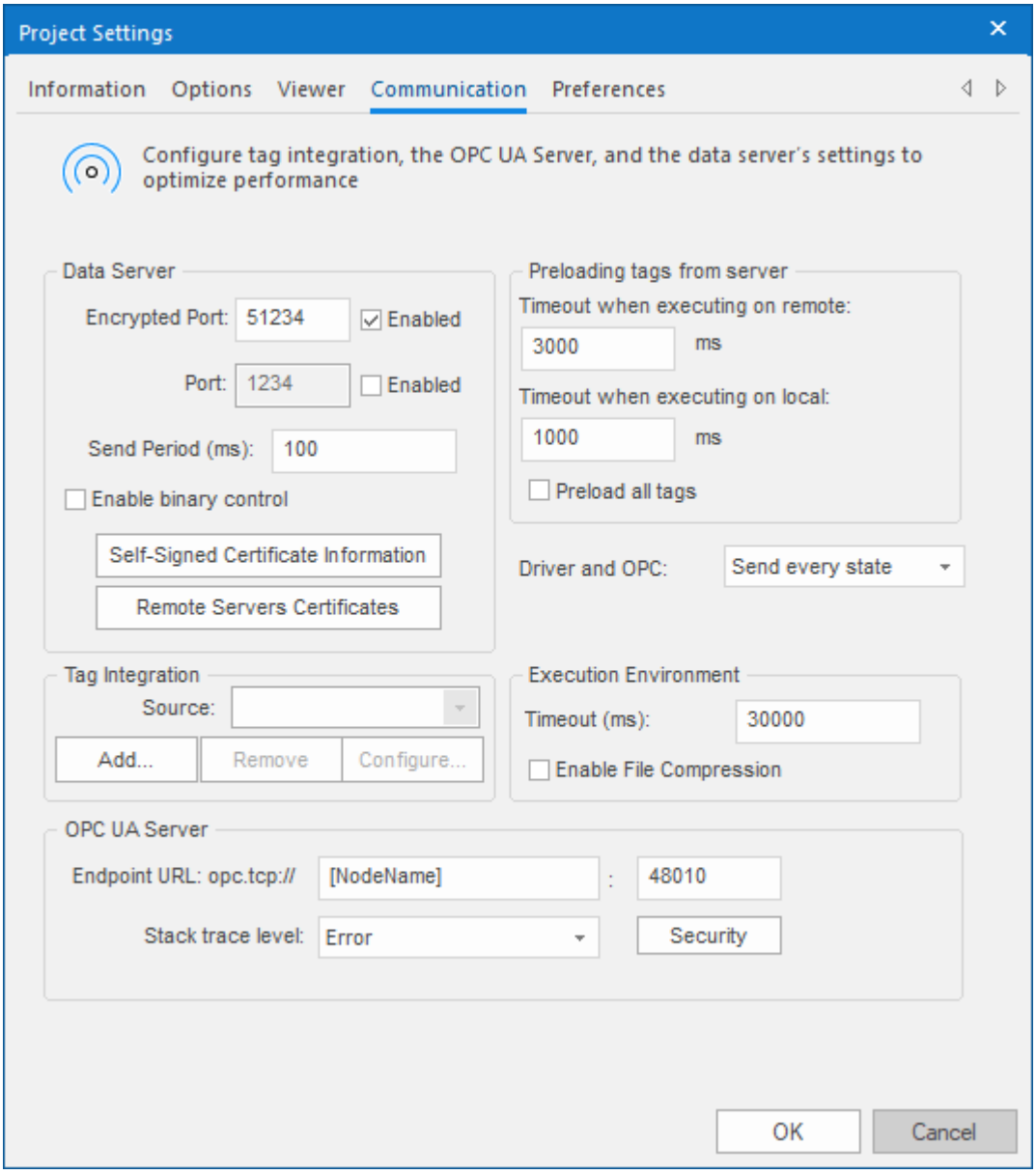
Before you begin this task, you should do the following:

- Review the manufacturer's documentation for your Schneider Modicon device and the Schneider Unity Pro software;
- Use the Schneider Unity Pro software to export an I/O configuration file (`*.XSY`) from your PLC program;
- Read the SCHNE driver documentation (on the **Help** tab of the ribbon, in the **Documentation** group, click **Communication Drivers**, and then select **SCHNE**); and
- Make sure the source device is running and available on your network, and note its network address.

To add a Schneider Modicon device as a tag integration source:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.

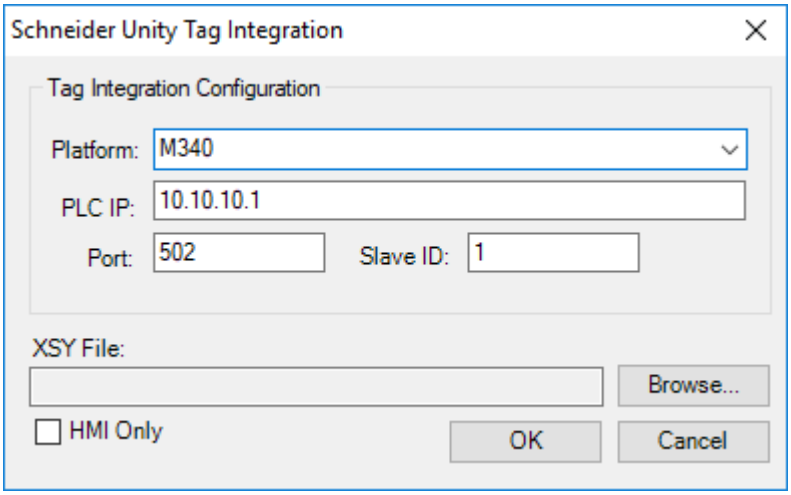
The *Project Settings* dialog is displayed, with the **Communication** tab selected.




***Communication tab of the Project Settings dialog***

2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** list, click **Integrated**.
4. In the **Provider** list, click **Schneider Unity Modbus**.
5. In the **Name** box, type an appropriate name for this tag integration source.
   Note that the name will be used as a prefix on names of the integrated tags.
6. Click **Add**.

The *Schneider Unity Tag Integration* dialog box is displayed.




***Schneider Unity Tag Integration dialog box***

7. In the **Platform** list, click the platform of the source device.

| Option | Description |
|---|---|
| **M340** | Schneider Modicon M340 PAC |
| **Premium** | Schneider Modicon Premium PAC |

8. In the **PLC IP** box, type the network address of the source device.
9. In the **Port** box, type the port number of the source device if it is different from the default port 502.
10. In the **Slave ID** box, type the Modbus slave ID of the source device if it is different from the default ID 1.
11. Select the I/O configuration file that you exported from your PLC program.
    a) Click **Browse**.
    A standard *Open* dialog box is displayed.
    b) Locate and select the I/O configuration file (`*.XSY`).
    In most cases, the file should be saved in the Config sub-folder of your EcoStruxure Machine SCADA Expert project folder at: `EcoStruxure Machine SCADA Expert 2020 Projects\<project name>\Config\`
    c) Click **Open**.
    For more information, see Export I/O configuration file for Schneider Unity Modbus on page 566.
    The selected file is displayed in the **XSY File** box.
12. If you want to get only tags that are flagged as HMI variables, then select **HMI Only**.
13. Click **OK** to finish the configuration and add the source.

If the source is added successfully, then the Schneider Modicon device tags will be immediately available in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 573.

## EXPORT I/O CONFIGURATION FILE FOR SCHNEIDER UNITY MODBUS

Export an I/O configuration file from your Schneider Modicon PLC program in order to be able to add the PLC as a tag integration source.

The I/O configuration file (`*.XSY`) is exported from the Schneider Unity Pro development software. It contains information about all of the tags used in your PLC program, and the information can be imported into your EcoStruxure Machine SCADA Expert project.

To export the file:

1. Run the Schneider Unity Pro development software, and then open your PLC program.

2. In the program browser, right-click the **Variables & FB Instances** file, and then click **Export** on the shortcut menu.
   A standard *Export* dialog is displayed.
3. Use the file browser to locate where you want to save the file.
   In most cases, you should save it in the Config sub-folder of your EcoStruxure Machine SCADA Expert project folder at: `EcoStruxure Machine SCADA Expert 2020 Projects\<project name>\Config\`
4. In the **File name** box, type a name for the file.
5. Click **Export**.

The file is saved at the specified location.

### *Integrate tags from Siemens S7-1200/S7-1500*

This task describes how to add a Siemens S7-1200 or S7-1500 controller as a tag integration source in your project.

This tag integration is based on our SITIA driver, which communicates with the Siemens S7-1200 or S7-1500 controller (firmware 4.0 or later) over Ethernet using the SIMATIC protocol.

Before you begin this task, you should do the following:

- Review the manufacturer's documentation for your Siemens S7-1200 or S7-1500 controller;
- Read the documentation for our SITIA driver (on the **Help** tab of the ribbon, in the **Documentation** group, click **Communication Drivers**, and then select **SITIA**);
- Make sure the controller is updated to firmware 4.0 or later; and
- Make sure the controller is running and accessible on your network, and then note its network address.

You might also need to update the controller's program to enable it to communicate with your project. Specifically, use TIA Portal — the programming software for Siemens controllers — to do the following:

- In the controller's **Protection** settings (**General** > **Protection**), make sure the access level is **Full access (no protection)**;
- Also in the controller's **Protection** settings, make sure **Permit access with PUT/GET communication from remote partner (PLC, HMI, OPC, ...)** is selected; and
- Make sure the **Visible in HMI** and **Accessible in HMI** options are selected for all program tags that you want to integrate into your project.

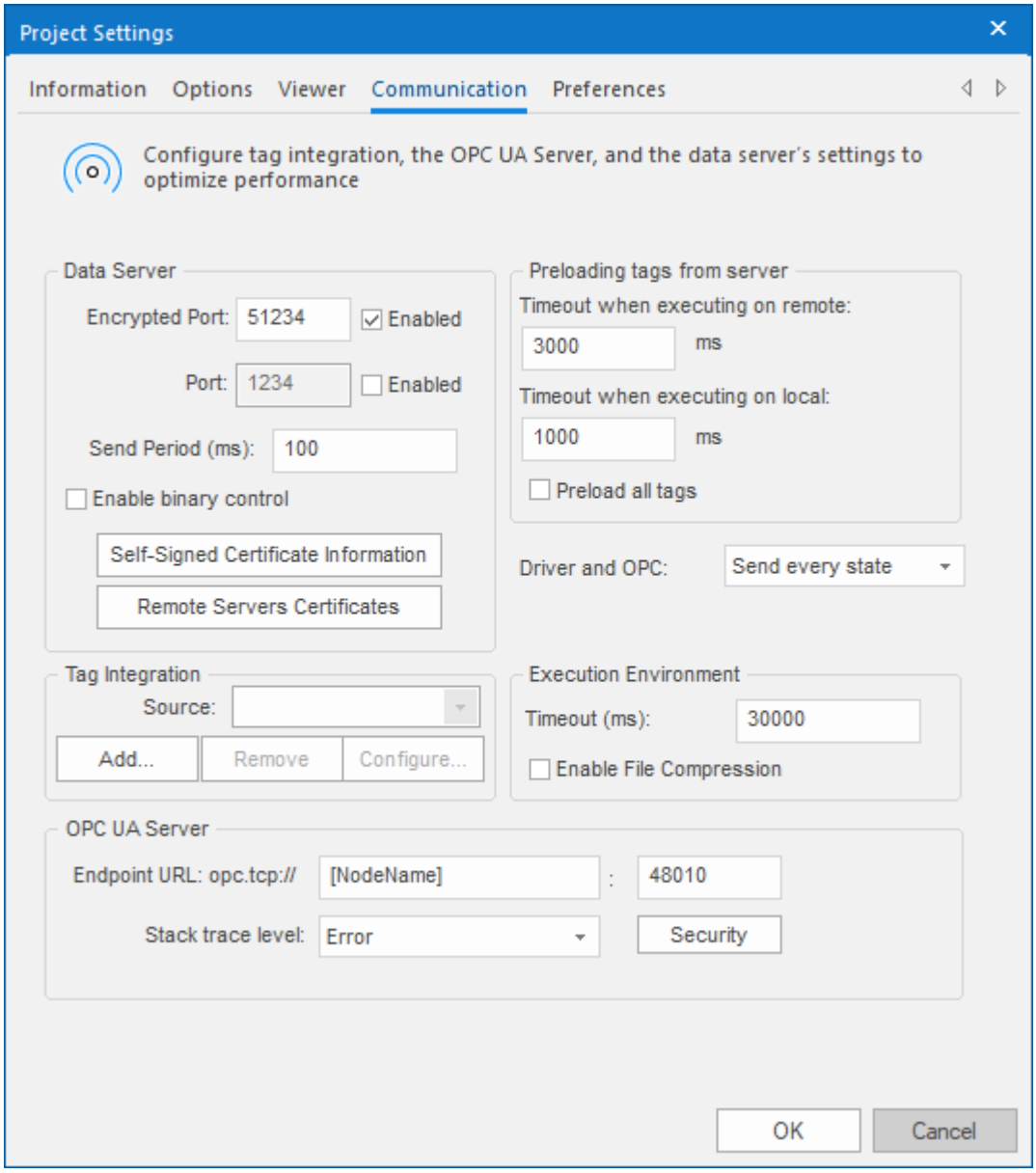
Remember that you need to recompile the controller's program and then download it to the controller whenever you change program settings or create new program tags.

> **Note:** Tag integration does not support password authentication. If you cannot set the controller's access level to **Full access (no protection)** as described above, or if the controller is configured with a fail-safe CPU (also known as a safety CPU or F-CPU, which provides increased protection and always requires a password to connect), then you cannot add that controller as a tag integration source. As an alternative, use the SITIA driver to establish direct communication with the controller. For more information, see Configuring direct communication with a remote device on page 518.

To add a Siemens S7-1200 or S7-1500 controller as a tag integration source:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.

The *Project Settings* dialog is displayed, with the **Communication** tab selected.

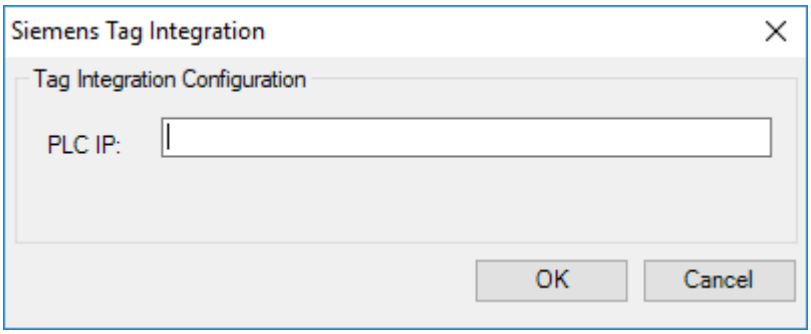



***Communication tab of the Project Settings dialog***

2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** list, select **Integrated**.
4. In the **Provider** list, select **Siemens S7-1200/S7-1500**.
5. In the **Name** box, type an appropriate name for the tag integration source.

   This name will be used as a prefix on the names of the integrated tags. The default is `DEV`, which is short for "device".
6. Click **Add**.

The *Siemens Tag Integration* dialog box is displayed.




***Siemens Tag Integration dialog box***

7. In the **PLC IP** box, type the IP address of the controller.
8. Click **OK** to finish the configuration and add the source.

If the source is added successfully, the controller's tags will be immediately available in the Object Finder in the Studio development environment. For more information, see Use the Object Finder to select integrated tags on page 573.

## *Add an OPC DA server as a tag integration source*

This task describes how to add an OPC server as a tag integration source in your project. It is based on our built-in OPC XML/DA client, which can communicate with OPC servers over Ethernet using the OPC Data Access (OPC DA) specification.

Before you begin this task, you should be familiar with the OPC DA specification (a.k.a. OPC Classic). For more information, go to: opcfoundation.org/about/opc-technologies/opc-classic/

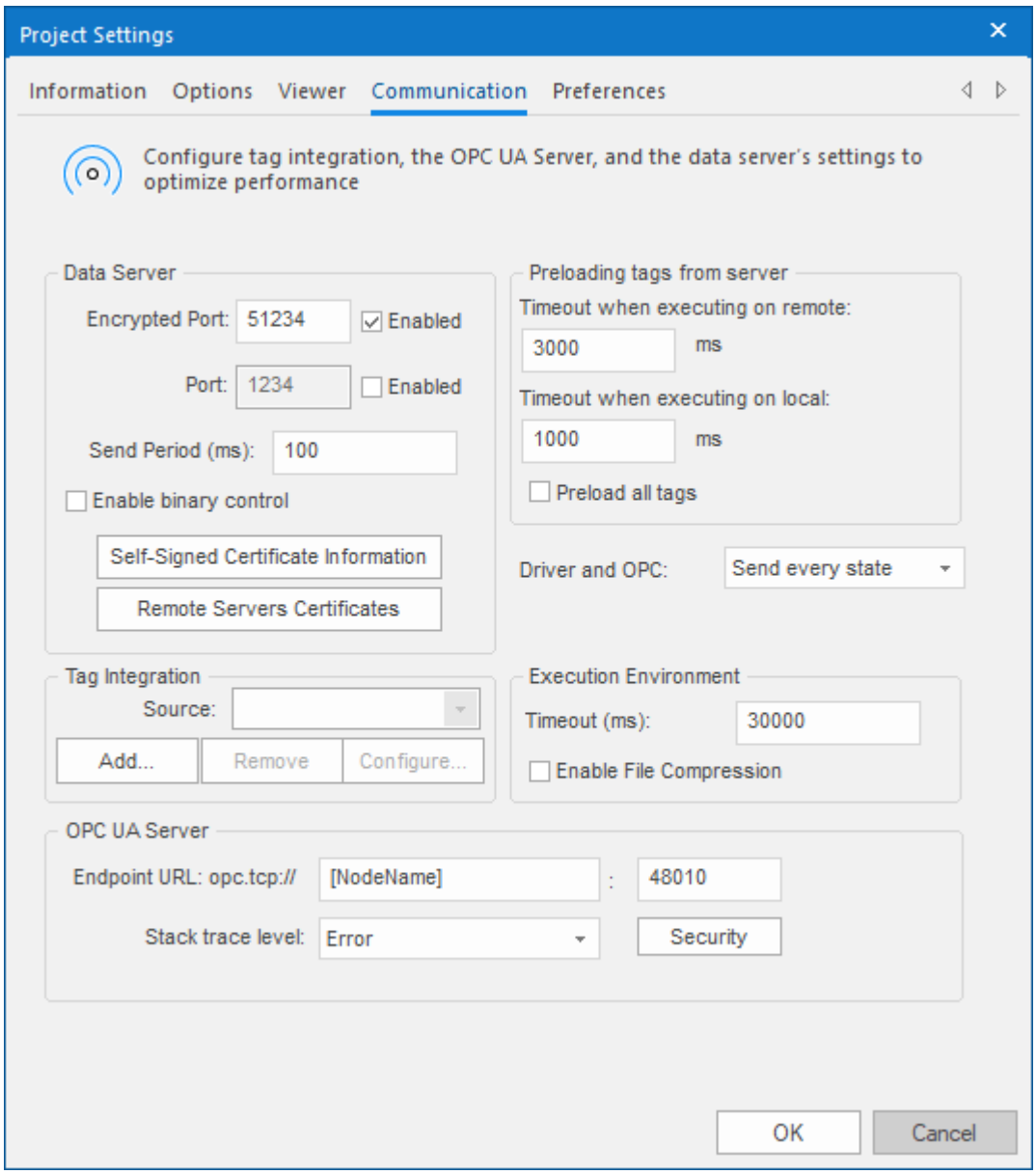
You should also review the documentation for your specific OPC server software. Make sure the server is running and accessible on your network, and then note its network address.

> **Note:** This task applies to OPC DA servers only. If you have an OPC UA server, see Add an OPC UA server as a tag integration source on page 571.

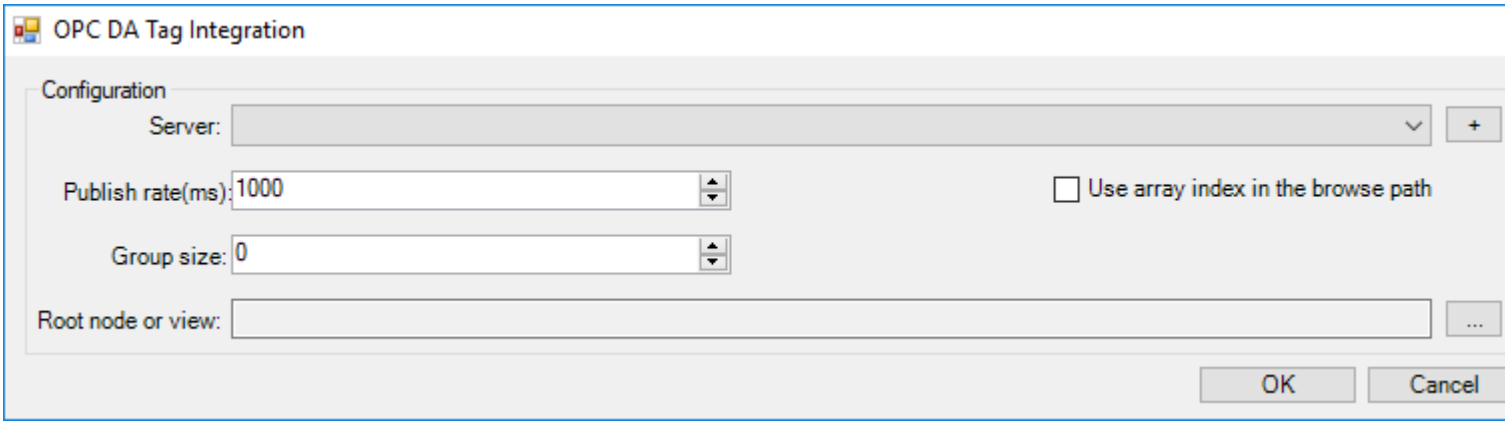
To add an OPC DA server as a tag integration source in your project:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.

   The *Project Settings* dialog is displayed, with the **Communication** tab selected.




2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** box, select **Integrated** if it is not already selected.
4. In the **Provider** box, select **OPC DA**.
5. In the **Name** box, type an appropriate name for the tag integration source.
   This name will be added as a prefix to the names of the integrated tags. The default is `DEV`, which is short for "device".
6. Click **Add**.
   The *OPC DA Tag Integration* dialog box is displayed.




7. In the **Server** box, select the connection to the OPC server.

   OPC DA tag integration is based on our built-in OPC XML/DA client, so you are selecting from the connections created for that client. You can select a connection that you previously created, or you can click the "add" button (**+**) to the right of the **Server** box in order to create a new connection. For more information, see Create a new OPC XML/DA connection on page 605. When you create a connection, make sure you select the appropriate OPC specification — either **Data Access 2.XX** or **Data Access 3.00** — to match the OPC server configuration.

8. In the **Publish rate** box, type the frequency (in milliseconds) at which the OPC server should publish updates to the client (i.e., your project).

   The publish rate is actually set when the client establishes its connection to the server.

9. In the **Group size** box, type the maximum number of items that may be read or written in a single operation.

   The default group size is 0, which means there is no limit and changes in item values will be immediately read and written (according to the publish rate). This is acceptable in most cases, but if you use a large number of items in your project and the values of those items change frequently, then read/write operations might overload the project and decrease run-time performance. You can specify a group size (e.g., 64) in order to moderate those read/write operations and restore run-time performance.

10. If the OPC server is configured to handle array elements as separate items, select the **Use array index in browse path** option. Conversely, if the OPC server is configured to handle each array as a single item with multiple elements, make sure the option is cleared.

    You can review the project runtime log (e.g., in the *Output* window) to see if there are any OPC communication errors when your project tries to read/write arrays. If there are, you might need to toggle the **Use array index in browse path** option.

11. In the **Root node or view** box, specify the server node that will serve as the root for all items. You can click the "more" button (...) to the right of the box in order to browse the server and then select a node.

    This step is optional, but specifying a root node makes it easier to find items and increases run-time performance.

12. Click **OK** to finish the configuration and add the source.

If the OPC server is successfully added as a tag integration source, its items will be immediately available for selection in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 573.

### *Add an OPC UA server as a tag integration source*

This task describes how to add an OPC server as a tag integration source in your project. It is based on our built-in OPC UA client, which can communicate with OPC servers over Ethernet using the OPC Unified Architecture (OPC UA) specification.

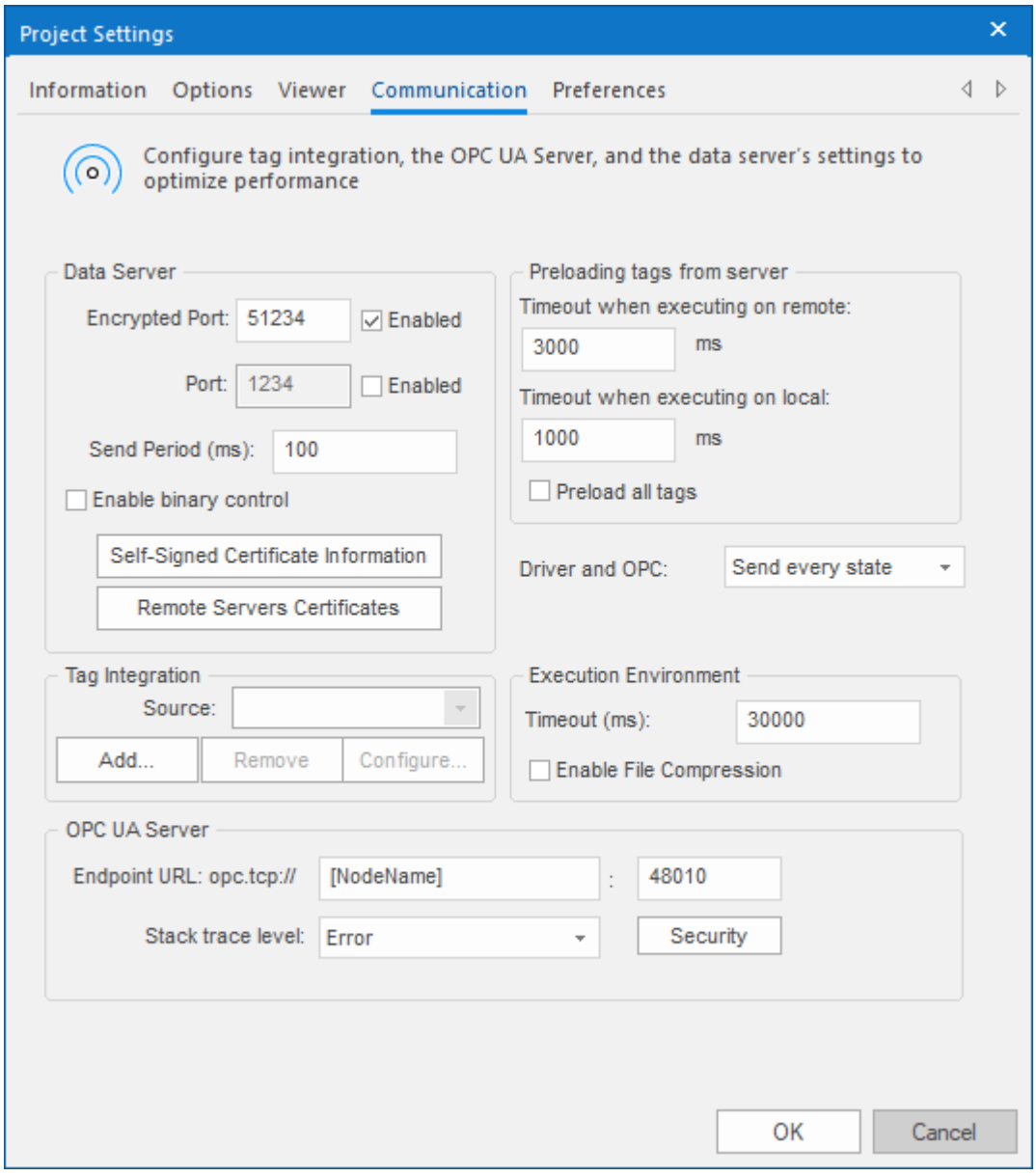
Before you begin this task, you should be familiar with the OPC UA specification. For more information, go to: opcfoundation.org/about/opc-technologies/opc-ua/

You should also review the documentation for your specific OPC server software. Make sure the server is running and accessible on your network, and then note its network address.

> **Note:** This task applies to OPC UA servers only. If you have an OPC DA server, see Add an OPC DA server as a tag integration source on page 569.

To add an OPC UA server as a tag integration source in your project:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.
   The *Project Settings* dialog is displayed, with the **Communication** tab selected.

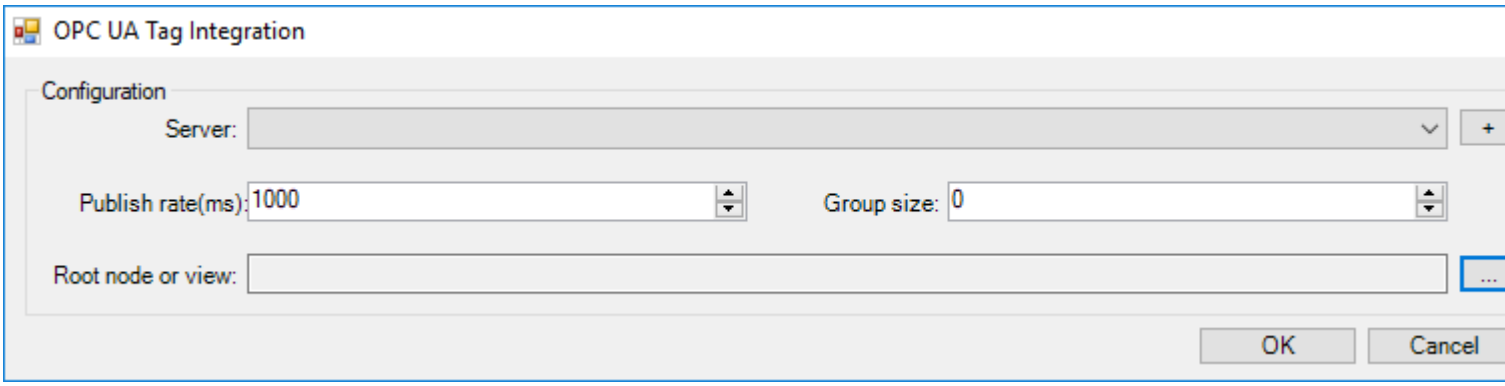



2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.
3. In the **Type** box, select **Integrated** if it is not already selected.
4. In the **Provider** box, select **OPC UA**.
5. In the **Name** box, type an appropriate name for the tag integration source.

This name will be added as a prefix to the names of the integrated tags. The default is `DEV`, which is short for "device".

6. Click **Add**.

   The *OPC UA Tag Integration* dialog box is displayed.




7. In the **Server** box, select the connection to the OPC server.

   OPC UA tag integration is based on our built-in OPC UA client, so you are selecting from the connections created for that client. You can select a connection that you previously created, or you can click the "add" button (**+**) to the right of the **Server** box in order to create a new connection. For more information, see Create a new OPC UA connection on page 576.

8. In the **Publish rate** box, type the frequency (in milliseconds) at which the OPC server should publish updates to the client (i.e., your project).

   The publish rate is actually set when the client establishes its connection to the server.

9. In the **Group size** box, type the maximum number of items that may be read or written in a single operation.

   The default group size is 0, which means there is no limit and changes in item values will be immediately read and written (according to the publish rate). This is acceptable in most cases, but if you use a large number of items in your project and the values of those items change frequently, then read/write operations might overload the project and decrease run-time performance. You can specify a group size (e.g., 64) in order to moderate those read/write operations and restore run-time performance.

10. In the **Root node or view** box, specify the server node that will serve as the root for all items. You can click the "more" button (...) to the right of the box in order to browse the server and then select a node.

    This step is optional, but specifying a root node makes it easier to find items and increases run-time performance.

11. Click **OK** to finish the configuration and add the source.

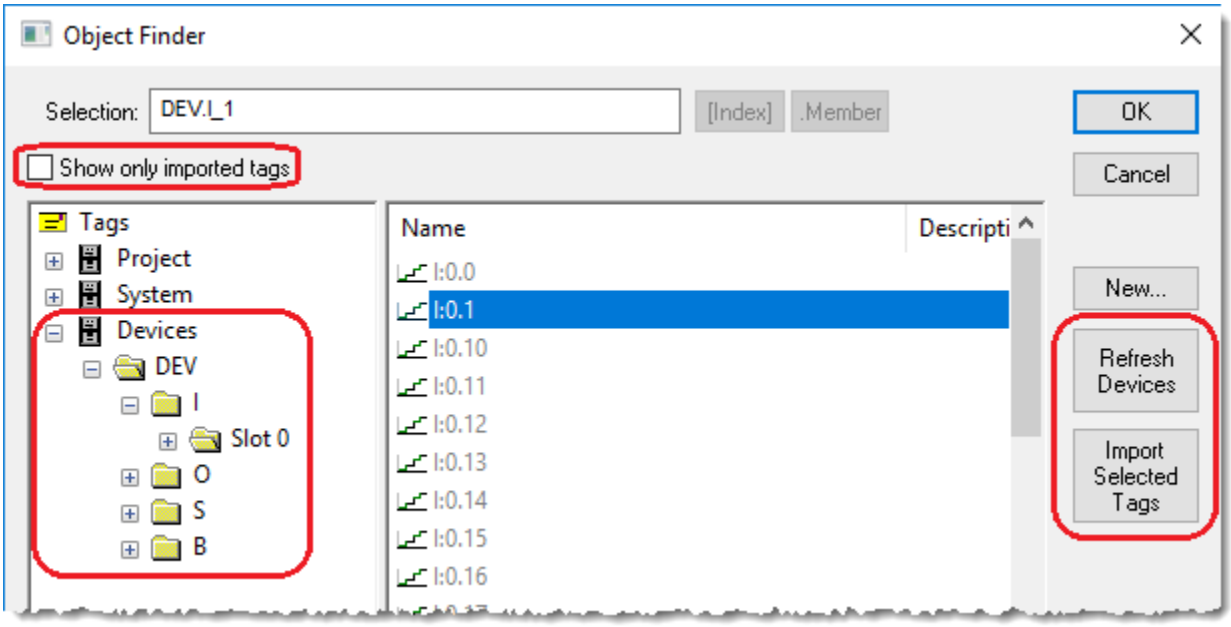
If the OPC server is successfully added as a tag integration source, its items will be immediately available for selection in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 573.

### *Use the Object Finder to select integrated tags*

After you add a tag integration source to your project, you can use the Object Finder to browse and select tags from that source like you would do for any other project tags.

> **Tip:** To open the Object Finder, double-click in any box or field where you would normally insert a project tag.

Integrated tags are listed under **Devices** in the Object Finder tree-view, with each device folder being a tag integration source that you added. The name of the device folder is the name that you specified when you added the device.




***Browsing integrated tags in the Object Finder***

A device folder may be marked with a badge that indicates the status of that device, as shown in the following table:

| Folder Icon | Description |
|---|---|
| DEV | Device is not "live"; tag information was extracted from a symbol file and then cached. |
| DEV | Device is "live"; tag information is currently being received from the connected device. |
| DEV | Connection to the device has failed; tag information might not be up-to-date. |

Additional information about connected devices may be shown at the bottom of the dialog box, below the tree-view.

The sub-folder structure for each device is determined by that device, and it cannot be changed from within the project development environment. If you need to change it for any reason, you must use the programming software for that device and you should do so before you use any of that device's tags in your project.

When you select a device folder or sub-folder, its list of available tags is shown to the right.

To use a specific tag in your project, select it in the list of available tags and then click **OK**. The selected tag is imported into your project's Shared Database folder, and then it is inserted into the box or field from which you opened the Object Finder.

Tags listed in black are already imported into your project. Tags listed in gray have not yet been imported. You can select the **Show only imported tags** option to filter the list of available tags, if you want to find a tag that has been imported and you do not want to scroll through the entire list.

To refresh the devices and their tags — either by reconnecting to the devices or by reloading their symbol files — click **Refresh Devices** on the right side of the dialog box. All tag integration sources will be refreshed at the same time, regardless of when or how they were added to the project.

To quickly import multiple tags, select all of them in the list of available tags and then click **Import Selected Tags** on the right side of the dialog box. Importing tags in this way does not insert the tags into the box or field

from which you opened the Object Finder; inserting can be done only one tag a time. Nevertheless, imported tags count against your project's tag limit, so you should not import tags that you do not intend to use later.

> **Tip:** To select multiple items in a list, hold the **Control** or **Shift** key while you click the items.

Tag integration does not support non-zero-based arrays (i.e., arrays that start at positions greater than 0) or pointer variables from devices. Such items will be included in the list of available tags, but they will be marked with red "X" badges to indicate that they cannot be imported or used.

Any changes in a tag integration source might break tag references in your project. To find broken tag references, verify your project.

### *How integrated tags may be renamed in your project*

When EcoStruxure Machine SCADA Expert integrates tags from third-party devices and software, it cannot directly transcribe the tag names. Some changes are made to improve tag management and to adhere to the local tag name syntax.

> **Tip:** The full, original name of an integrated tag can always be retrieved by referencing the **`DeviceTag`** property on the tag. For example, ***`tagname`*`->DeviceTag`**.

#### Inserting the tag prefix

First of all, since your project may connect to multiple devices that have the same control program and device tags, EcoStruxure Machine SCADA Expert will automatically insert the tag prefix that you specified when you added the tag integration source. For example, for an integrated tag named...

```
MyDeviceTAG[1].ClassMember
```

...the corresponding EcoStruxure Machine SCADA Expert project tag will be named...

```
tagprefix.MyDeviceTAG[1].ClassMember
```

This allows you to differentiate between similar tags from different sources.

#### Multidimensional arrays

EcoStruxure Machine SCADA Expert does not support multidimensional arrays, so for integrated tags that have more than one array index, each index after the first will be represented with ***`_Index_`***. For example, for an integrated tag named...

```
MyDeviceTAG[1][2][3].ClassMember
```

...the corresponding EcoStruxure Machine SCADA Expert project tag will be named...

```
tagprefix.MyDeviceTAG[1]_2__3_.ClassMember
```

#### Nested classes

EcoStruxure Machine SCADA Expert does not support nested classes, so for integrated tags that have more than one class member, each class member after the first will be represented with ***`_ClassMember`***. For example, for an integrated tag named...

```
MyDeviceTAG.ClassMember.ClassMember2
```

...the corresponding EcoStruxure Machine SCADA Expert project tag will be named...

```
tagprefix.MyDeviceTAG.ClassMember_ClassMember2
```

## OPC Clients and Servers

Open Platform Communications (OPC) is an interoperability standard for exchanging real-time process data. This software includes a variety of OPC client and OPC server features that you can use to exchange data between your project and other OPC-compatible systems.

"OPC" originally stood for "OLE for Process Control" because the standard depended on Microsoft's proprietary DCOM and OLE technologies. As the standard has evolved, however, it has moved from those proprietary technologies to cross-platform technologies like XML and SOAP for web services. The latest version of the OPC standard is OPC Unified Architecture (OPC UA).

For more information about OPC, go to: https://opcfoundation.org/

### *OPC UA Client*

Use the OPC UA Client worksheet and runtime task to establish communication between your project and a data exchange server that supports the OPC UA interoperability standard.

This OPC UA Client feature uses the new OPC Unified Architecture standard introduced by the OPC Foundation. According to the foundation:

> The OPC Unified Architecture (UA), released in 2008, is a platform independent service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework.
>
> This multi-layered approach accomplishes the original design specification goals of:
>
> - Functional equivalence: all COM OPC Classic specifications are mapped to UA
> - Platform independence: from an embedded micro-controller to cloud-based infrastructure
> - Secure: encryption, authentication, and auditing
> - Extensible: ability to add new features without affecting existing applications
> - Comprehensive information modeling: for defining complex information

In other words, OPC UA is intended to be a platform- and language-independent standard. For more information, go to: opcfoundation.org/about/opc-technologies/opc-ua/

> **Note:** This feature includes cryptographic software written by Eric Young (eay@cryptsoft.com).

#### CREATE A NEW OPC UA CONNECTION

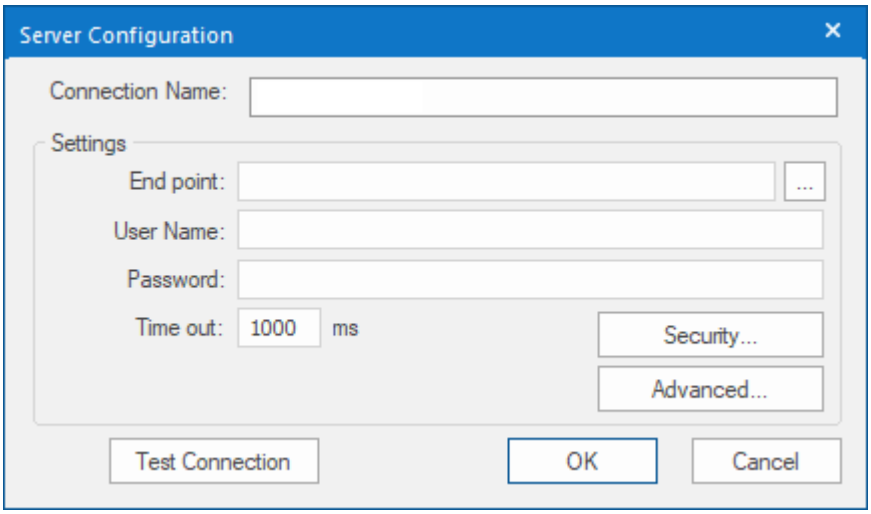
When you configure an OPC UA Client worksheet, you must select the connection that the client will use. This task describes how to create that connection.

Before you begin this task, you should know the communication and security settings for the OPC server to which you want to connect. If you do not, contact the server administrator.

To create a new connection to an OPC UA server:

1. Do one of the following:
   - On the **Insert** tab of the ribbon, in the **Communication** group, click **OPC Client**, and then select **OPC UA Connection** from the drop-down list; or
   - In the **Comm** tab of the Project Explorer, expand the **OPC UA** folder, right-click the **Connections** folder, and then click **Insert** on the shortcut menu.

   The *Server Configuration* dialog box is displayed.

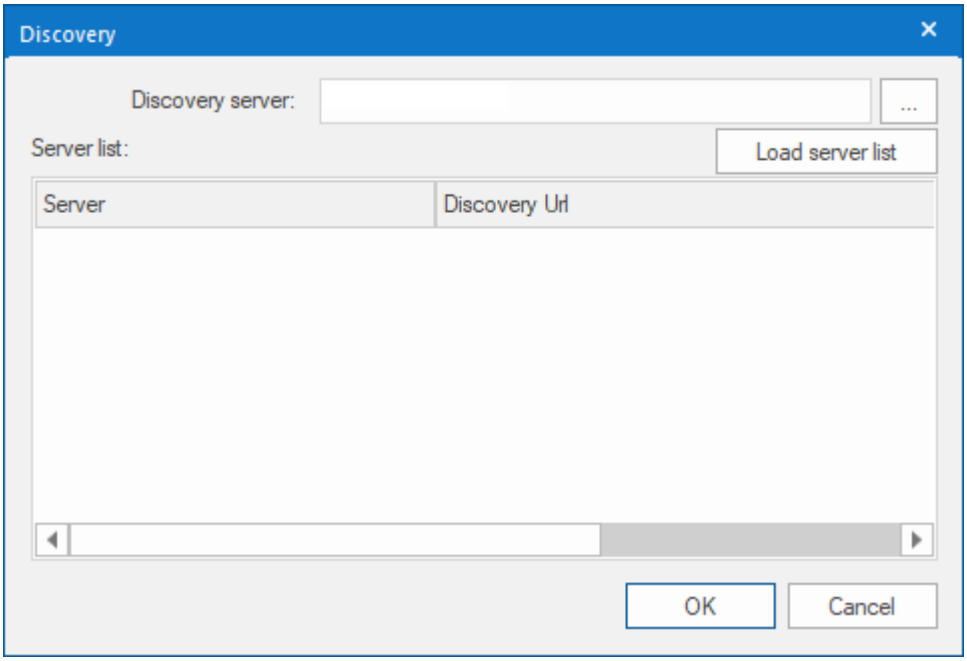



2. In the **Connection Name** box, type a name for the connection.

   This name will be displayed in the **OPC UA** > **Connections** folder in the Project Explorer, and it is the name you will look for when you configure the OPC UA Client worksheet.

3. In the **End point** box, do one of the following:

   - If you know the URL of the OPC server to which you want to connect, type it in the box.
   - If you do not know the URL, click the More button (...) to open the *Discovery* dialog box, use it to find a discovery server that publishes a list of OPC servers on your network, and then select the server to which you want to connect.




   - If you want to be able to change the URL during project run time, type an appropriate string expression (e.g., `{MyEndpointUrl}`). When the OPC UA Client Runtime task is started, it will get the value of the string expression and then connect to that URL. This happens only when the task is started, which typically happens when the project itself is run. If the value of the string expression changes after the

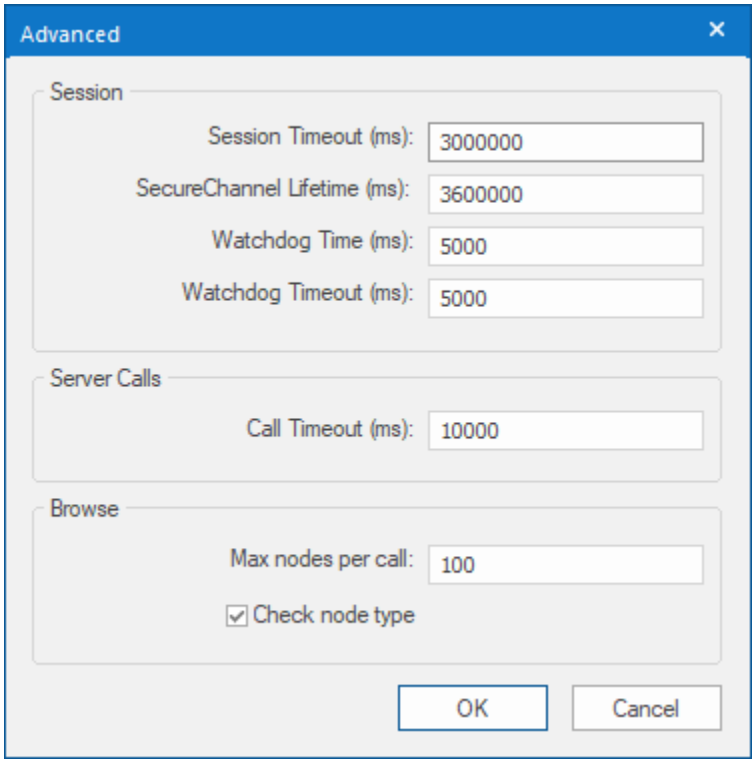
task is started, the task must be restarted in order to get the new value and then connect to the new URL. To restart the task while the project is running, use either the *Runtime Tasks* dialog box or the `EndTask` and `StartTask` functions.

> **Note:** At this time, the OPC UA Client feature in this software supports only binary communication with OCP.TCP end points (e.g., `opc.tcp://<host name or IP address>:8000/<server name>`). SOAP-based communication with HTTP and HTTPS end points is not supported.

4. In the **User Name** and **Password** boxes, type your credentials for the OPC server.
   You can leave these boxes empty if you want to connect anonymously and the server is configured to accept anonymous connections.
5. If your OPC server is configured to require secure communication (also called a "SecureChannel" in the OPC UA specification), you must take extra steps to configure the security settings and program certificates. The steps differ somewhat depending on whether you are using self-signed certificates or certificates signed by a certificate authority (CA):
   - Configure an OPC UA connection to use self-signed certificates on page 580
   - Configure an OPC UA connection to use CA-signed certificates on page 583

   > **Note:** OPC UA security is supported in projects running on Windows and Windows Embedded Standard. For more information, see About target platforms, product types, and target systems on page 100.
6. Click **Test Connection**.
   If the program can successfully connect to the OPC server using these settings, a confirmation message is displayed.
7. Click **OK** to save your changes and close the *Server Configuration* dialog box.
   The connection is saved in the **OPC UA** > **Connections** folder in the Project Explorer.

In certain situations, if the connection does not behave as expected during project run time — and especially if you see OPC communication errors in the runtime log — you might need to adjust the connection's advanced settings: in the *Server Configuration* dialog box, click **Advanced**, and then in the *Advanced* dialog box, review and configure the settings.




**Session**

Various session timeouts for the connection itself:

**Session Timeout**

The session timeout (in milliseconds) for the connection. If a session times out from inactivity, a new session must be started to resume communication.

**SecureChannel Lifetime**

The time (in milliseconds) after which the SecureChannel between the client and server is automatically renewed.

For more information about the SecureChannel Services, see the OPC UA specification.

**Watchdog Time**

The time (in milliseconds) between watchdog checks.

**Watchdog Timeout**

The timeout (in milliseconds) for a specific watchdog check.

**Server Calls**

Advanced settings that control individual calls to the OPC server:

**Call Timeout**

The timeout (in milliseconds) for individual server calls.

**Verify time synchronization**

When this option is selected, the time stamp on each value received from the server is compared against the system time on the client, and if the difference between the times is equal to the specified number of hours (i.e., the number of time zones between the server and client), the server and client are deemed to be synchronized and the value is accepted. Otherwise, if the difference is not equal to the specified number of hours, the value is rejected and communication is interrupted.

The default number of hours is 25, which is a special setting that means the hours portion of the time stamp should be disregarded and only the minutes and seconds portions should be compared. This enables the client to receive values from the server without knowing the server's time zone.

When this option is cleared, the time stamp on each value received from the server is ignored and the client's system time is used instead to save the value in the project database.

**Browse**

Advanced settings that control how you browse server items/nodes on the OPC server as you configure an OPC UA Client worksheet:

**Max nodes per call**

When browsing for an item/node on the server, the maximum number of nodes to be returned per server call.

**Check node type**

When browsing for an item/node on the server, check the node's data type. In some situations, the OPC server might not be able to get data types from field devices, and that might cause browsing to become slow and unusable. If that happens, clear the **Check node type** option.

This option can also interfere with tag expansion, so if tag expansion is used in any of the OPC UA Client worksheets that depend this connection, clear the **Check node type** option.

## CONFIGURE AN OPC UA CONNECTION TO USE SELF-SIGNED CERTIFICATES

You can configure an OPC UA connection to communicate securely using self-signed certificates.

This task is a supplement to another task, Create a new OPC UA connection on page 576. It assumes you have already created a new connection and are now configuring the security settings for that connection.

A certificate is used to identify a program that is trying to communicate securely. If another program is configured to trust the certificate, it accepts that the first program is what it claims to be and agrees to communicate with it. Therefore, when two programs trust each other's certificates, they can establish secure, two-way communication with each other.

There are two ways to configure a program to trust a given certificate. First, you can manually add the certificate to the program's trust list, so that when the certificate is presented during communication, the program can check it against the list. Second, you can instruct the program to trust the certificate authority (CA) that signed the certificate, so that when the certificate is presented during communication, the program automatically trusts it.

A self-signed certificate is a certificate signed by the program that created it, rather than by a certificate authority, so it must be manually added to the other program's trust list. Self-signed certificates are safe and convenient to use as long as you control both programs — for example, both the OPC UA client in your project and the OPC UA server itself.

There are some potential issues with using self-signed certificates, however:

- If you do not control both programs, you do not control their respective trust lists;
- If you have many programs that all communicate with each other, you must add each program's certificate to every other program's trust list; and
- If any program's certificate changes or expires, you must renew it and then add it again to every other's program's trust list.

In other words, self-signed certificates are only suitable for limited use among a handful of programs that you control. If you think you will encounter any of the issues listed above, consider using CA-signed certificates instead. For more information, see Configure an OPC UA connection to use CA-signed certificates on page 583.

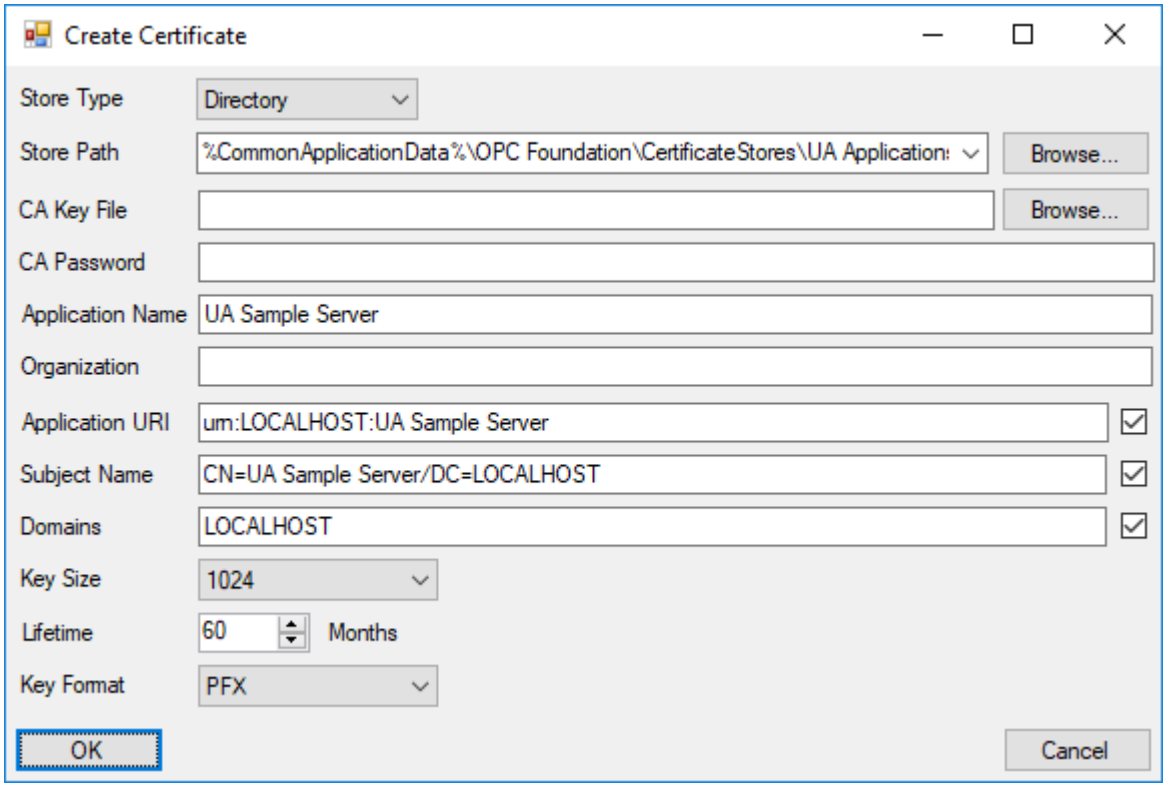
OPC UA uses a file-based certificate store, which means that the certificates are saved as files in folders rather than as entries in a database or system registry. This is important to know because this task involves copying certificate files from one folder to another. For more information, see "Certificate Management" in *OPC Unified Architecture Specification, Part 2: Security Model*. You can download that document from the OPC Foundation website at: opcfoundation.org/developer-tools/specifications-unified-architecture/part-2-security-model/

In this task, "OPC UA server" is a generic reference to any of the available hardware or software products that can run as an OPC UA server. For more information about how to complete certain steps, consult the manufacturer's documentation for your specific server. We have provided examples based on the free UA Sample Server and UA Configuration Tool that are offered by the OPC Foundation (opcfoundation.org), but these examples should be used only as a guide. The UA Configuration Tool is sometimes redistributed by other manufacturers with their own products, so these examples might appear to apply to your specific server, but you should still review each example before you proceed.

> **Note:** OPC UA security is supported in projects running on Windows and Windows Embedded Standard. For more information, see About target platforms, product types, and target systems on page 100.

To configure an OPC UA connection to use self-signed certificates:

1. In your OPC UA server, create a new, self-signed certificate for the server.
   Example using UA Sample Server and UA Configuration Tool:
   a) In the UA Configuration Tool, click the **Manage Application** tab.
   b) In the **Application To Manage** list, make sure **Opc.Ua.SampleServer** is selected, and then click **Create Application Certificate**.
   The *Create Certificate* dialog box is displayed.
   c) In the **Store Type** list, make sure **Directory** is selected.
   d) In the **Store Path** box, make sure **UA Applications** is selected.
   e) Complete the certificate information (e.g., Application Name, Organization, etc.) as needed, but leave the **CA Key File** and **CA Password** boxes empty.
   That is what will make the certificate self-signed.




***Example of self-signed certificate settings for UA Sample Server running on localhost***

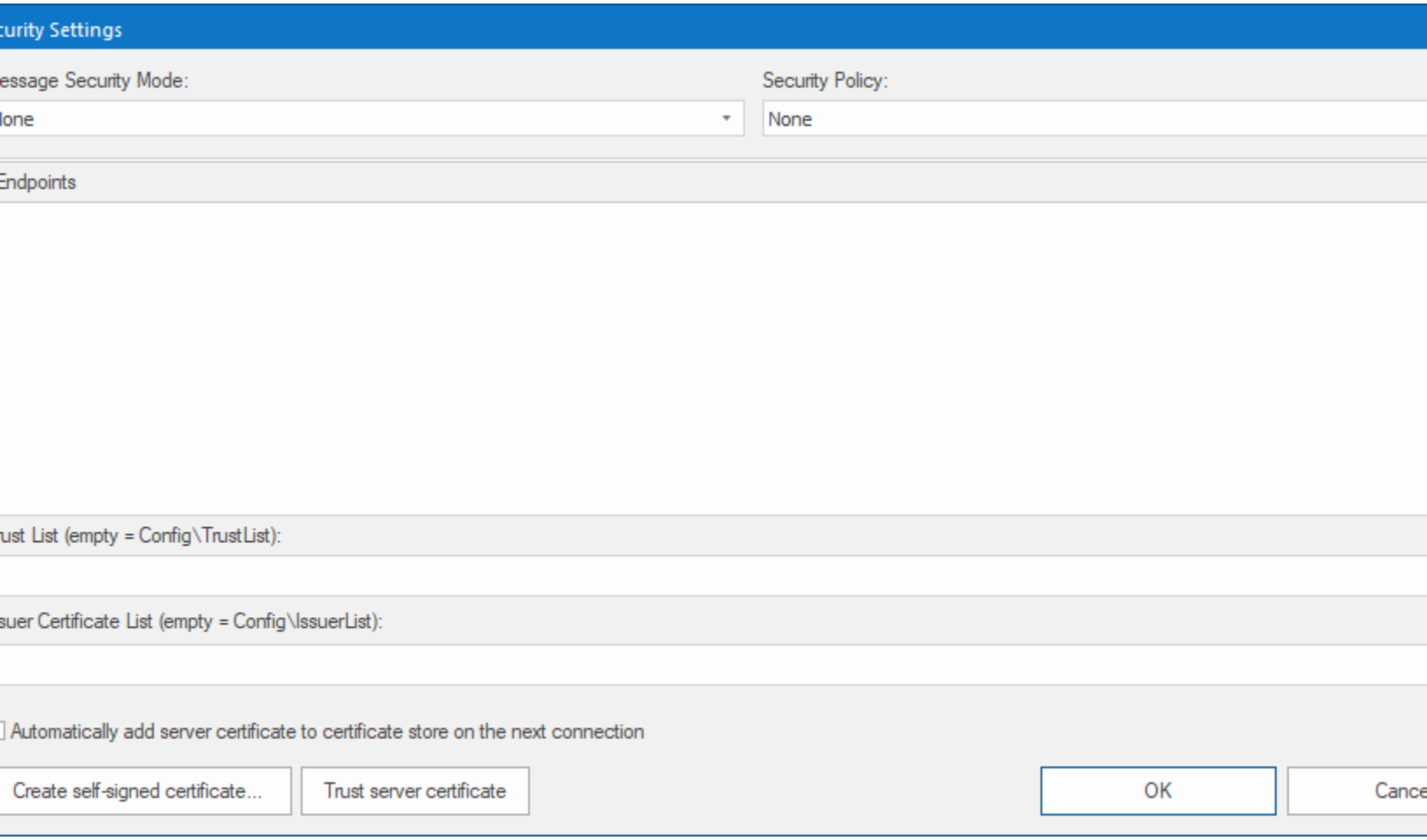
   f) Click **OK**.

The server certificate is created. If there is an old certificate in the certificate store, you might be prompted to overwrite or delete it. It should be safe to do so as long as there are no clients connected to the server.

g) Restart the UA Sample Server to make sure it uses the new certificate.

2. In EcoStruxure Machine SCADA Expert 2020, in the *Server Configuration* dialog box, click **Security**.

   The *Security Settings* dialog box is displayed.




3. If the OPC UA server is configured to broadcast the security modes and policies that it supports, a list of those will be displayed in the **Endpoints** box and you can select the server configuration that you want to use. Otherwise, you need to manually select the security mode and policy in the **Message Security Mode** and **Security Policy** boxes, respectively. These settings must match the server configuration.
   In a typical server configuration, the security mode might be **Sign and Encrypt** and the security policy might be **Basic128Rsa15**.
4. After you select the security mode, EcoStruxure Machine SCADA Expert 2020 checks whether an appropriate client certificate exists in the project folder. If it does not, you are asked if you want to create it and its associated key. Click **Yes**.
   The *Certificate Creation* dialog box is displayed.
5. In the *Certificate Creation* dialog box, complete the certificate information (e.g., Application Name, Organization, etc.) as needed, and then click **Generate**.
   A new client certificate is created for your project, and the certificate and key files are saved in your project folder at: `<project name>\Config\`
6. If your project will run on Windows or Windows Embedded Standard, do the following:

a) Click **Trust server certificate**.
EcoStruxure Machine SCADA Expert 2020 attempts to connect to the specified OPC UA server, and if it is successful, it imports the certificate into the client's trust list. A warning message is displayed, asking you to confirm that you trust the certificate.

b) Click **OK** to confirm.
The certificate file is copied to your project folder at: `<project name>\Config\TrustList\Certs\<connection name>.der`

> **Note:** In most cases, you do not need to change the **Trust List** or **Issuer Certificate List** settings. These are the default locations in your project folder where certificate files are stored. You may change the locations if, for example, you have a single folder where you store certificates for several different projects, but we do not recommend it.

The server certificate file is now part of your project files, so it will be copied with the rest of the project files whenever you download your project to a target device.

7. As an alternative to clicking **Trust server certificate**, you may select the **Automatically add server certificate to certificate store on next connection** option in order to have the project to automatically get the certificate when it runs and then connects to the OPC UA server.
   The advantage of doing this is that your project will always have the latest server certificate, if that certificate is ever changed or renewed. The limitation of doing this is that it is less secure than manually adding the certificate, especially when you do not control the server and/or the network. In most cases, we do not recommend selecting this option.
8. Click **OK** to close the *Security Settings* dialog box and return to the *Server Configuration* dialog box.
9. In your OPC UA server, import the client certificate into the server's trust list.
   Example using UA Sample Server and UA Configuration Tool:
   a) In the UA Configuration Tool, click the **Manage Security** tab.
   b) Click **Import Certificate to Trust**.
   A standard *Open File* dialog box is displayed.
   c) Use the dialog box to locate and select the client certificate file.
   For projects running on Windows and Windows Embedded Standard, the file is located in yout project folder at: `<project name>\Config\UAClientCertificate.der`
   d) Click **Open**.
   You will be asked to confirm the import.
   e) Click **Yes**.
   The selected file is imported into the server's UA Applications certificate store.

When the certificates have been exchanged — that is, when the server certificate is in the client's (i.e., your project's) trust list and the client certificate is in the server's trust list — the OPC UA connection should be properly configured for secure, two-way communication.

## CONFIGURE AN OPC UA CONNECTION TO USE CA-SIGNED CERTIFICATES

You can configure an OPC UA connection to communicate securely using certificates signed by a certificate authority (CA).

This task is a supplement to another task, Create a new OPC UA connection on page 576. It assumes you have already created a new connection and are now configuring the security settings for that connection.

A certificate is used to identify a program that is trying to communicate securely. If another program is configured to trust the certificate, it accepts that the first program is what it claims to be and agrees to communicate with it. Therefore, when two programs trust each other's certificates, they can establish secure, two-way communication with each other.

There are two ways to configure a program to trust a given certificate. First, you can manually add the certificate to the program's trust list, so that when the certificate is presented during communication, the program can check it against the list. Second, you can instruct the program to trust the certificate authority (CA) that signed the certificate, so that when the certificate is presented during communication, the program automatically trusts it.

You can buy CA-signed certificates from commercial certificate authorities such as VeriSign, DigiCert, and GeoTrust. We do not recommend that, however, because such certificates are intended for public-facing applications in which you control one program but not the other.

In this application, as long as you control both programs — that is, both the OPC UA client in your project and the OPC UA server itself — you can create your own certificate authority to sign both programs' certificates and then instruct both programs to trust the same certificate authority. It is more complicated than using self-signed certificates, but it helps you to avoid other potential issues with using self-signed certificates. For more information, see Configure an OPC UA connection to use self-signed certificates on page 580.

OPC UA uses a file-based certificate store, which means that the certificates are saved as files in folders rather than as entries in a database or system registry. This is important to know because this task involves copying certificate files from one folder to another. For more information, see "Certificate Management" in *OPC Unified Architecture Specification, Part 2: Security Model*. You can download that document from the OPC Foundation website at: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-2-security-model/

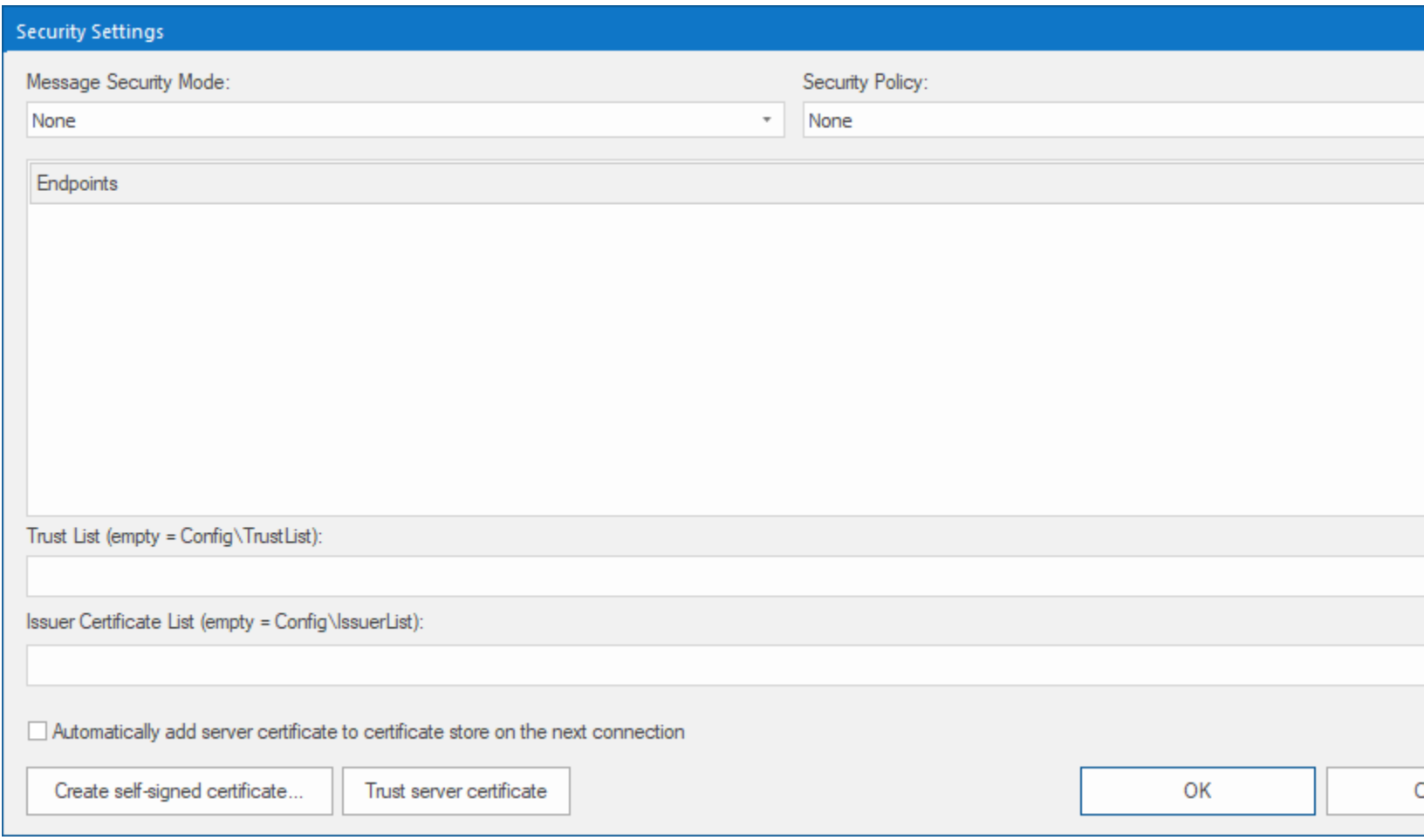
In this task, "OPC UA server" is a generic reference to any of the available hardware or software products that can run as an OPC UA server. For more information about how to complete certain steps, consult the manufacturer's documentation for your specific server. We have provided examples based on the free UA Sample Server and UA Configuration Tool that are offered by the OPC Foundation (opcfoundation.org), but these examples should be used only as a guide. The UA Configuration Tool is sometimes redistributed by other manufacturers with their own products, so these examples might appear to apply to your specific server, but you should still review each example before you proceed.

> **Note:** OPC UA security is supported in projects running on Windows and Windows Embedded Standard. For more information, see About target platforms, product types, and target systems on page 100.

To configure an OPC UA connection to use CA-signed certificates:

1. In EcoStruxure Machine SCADA Expert 2020, in the *Server Configuration* dialog box, click **Security**.

   The *Security Settings* dialog box is displayed.




2. If the OPC UA server is configured to broadcast the security modes and policies that it supports, a list of those will be displayed in the **Endpoints** box and you can select the server configuration that you want to use. Otherwise, you need to manually select the security mode and policy in the **Message Security Mode** and **Security Policy** boxes, respectively. These settings must match the server configuration.
   In a typical server configuration, the security mode might be **Sign and Encrypt** and the security policy might be **Basic128Rsa15**.
3. After you select the security mode, EcoStruxure Machine SCADA Expert 2020 checks whether an appropriate client certificate exists in the project folder. If it does not, you are asked if you want to create it and its associated key. Click **Yes**.
   The *Certificate Creation* dialog box is displayed.
4. In the *Certificate Creation* dialog box, complete the certificate information (e.g., Application Name, Organization, etc.) as needed, and then click **Generate**.

   > **Note:** Although you are creating a self-signed certificate here, you will reissue it as a CA-signed certificate later in this procedure.

   A new client certificate is created for your project, and the certificate and key files are saved in your project folder at: `<project name>\Config\`
5. Click **OK** to close the *Security Settings* dialog box and return to the *Server Configuration* dialog box.
6. In your OPC UA server, create your certificate authority.
   Example using UA Sample Server and UA Configuration Tool:
   a) In the UA Configuration Tool, click the **Manage Certificates** tab.
   b) Click **Create Certificate Authority**.

The *Create Certificate Authority* dialog box is displayed.

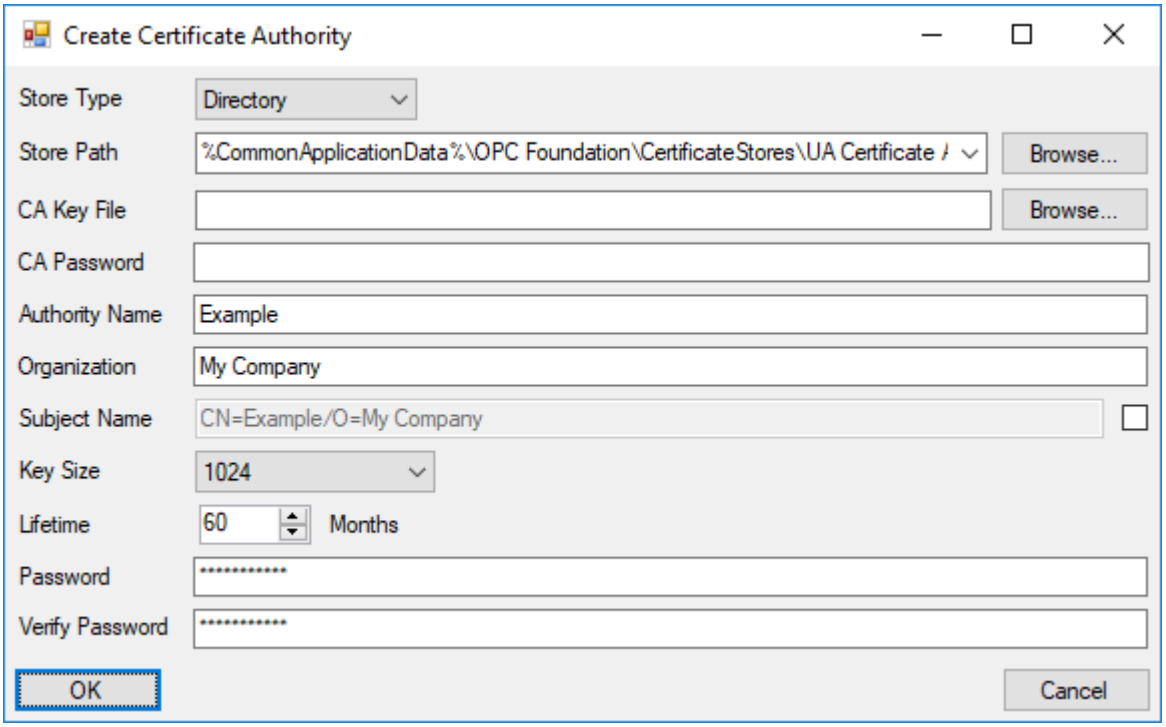
c) In the **Store Type** list, make sure **Directory** is selected.

d) In the **Store Path** box, make sure **UA Certificate Authorities** is selected.

e) Complete the certificate authority information (e.g., Authority Name, Organization, etc.) as needed, but leave the **CA Key File** and **CA Password** boxes empty.

That is what will make the certificate authority self-created. You can use one certificate authority to sign another certificate authority and thus give it the same credentials, but that is not necessary in this situation.

f) In the **Password** and **Verify Password** boxes, type a password that you will remember.




***Example of certificate authority settings for UA Sample Server***

g) Click **OK**.
The certificate authority is created, and the certificate authority's certificate and key files are saved in the UA Certificate Authorities store.

7. In your OPC UA server, add the certificate authority to your server's trust list.

Even though you just used the server to create the certificate authority, the server will not trust it unless you specifically instruct it to.

Example using UA Sample Server and UA Configuration Tool:

a) In the UA Configuration Tool, click the **Manage Security** tab.

b) In the **Application To Manage** list, make sure **Opc.Ua.SampleServer** is selected, and then click **Select Certificate to Trust**.
The *Manage Certificates in Certificate Store* dialog box is displayed.

c) In the **Store Type** list, make sure **Directory** is selected.

d) In the **Store Path** box, make sure **UA Certificate Authorities** is selected.

e) In the list of certificates, select the certificate authority that you created in the previous step, and then click **OK**.
The selected certificate is added to the server's trust list. Or more accurately, the certificate authority's certificate file is copied to the UA Applications store.

8. In your OPC UA server, create a new, CA-signed certificate for the server.

Example using UA Sample Server and UA Configuration Tool:

a) In the UA Configuration Tool, click the **Manage Application** tab.

b) In the **Application To Manage** list, make sure **Opc.Ua.SampleServer** is selected, and then click **Create Application Certificate**.
The *Create Certificate* dialog box is displayed.

c) In the **Store Type** list, make sure **Directory** is selected.

d) In the **Store Path** box, make sure **UA Applications** is selected.

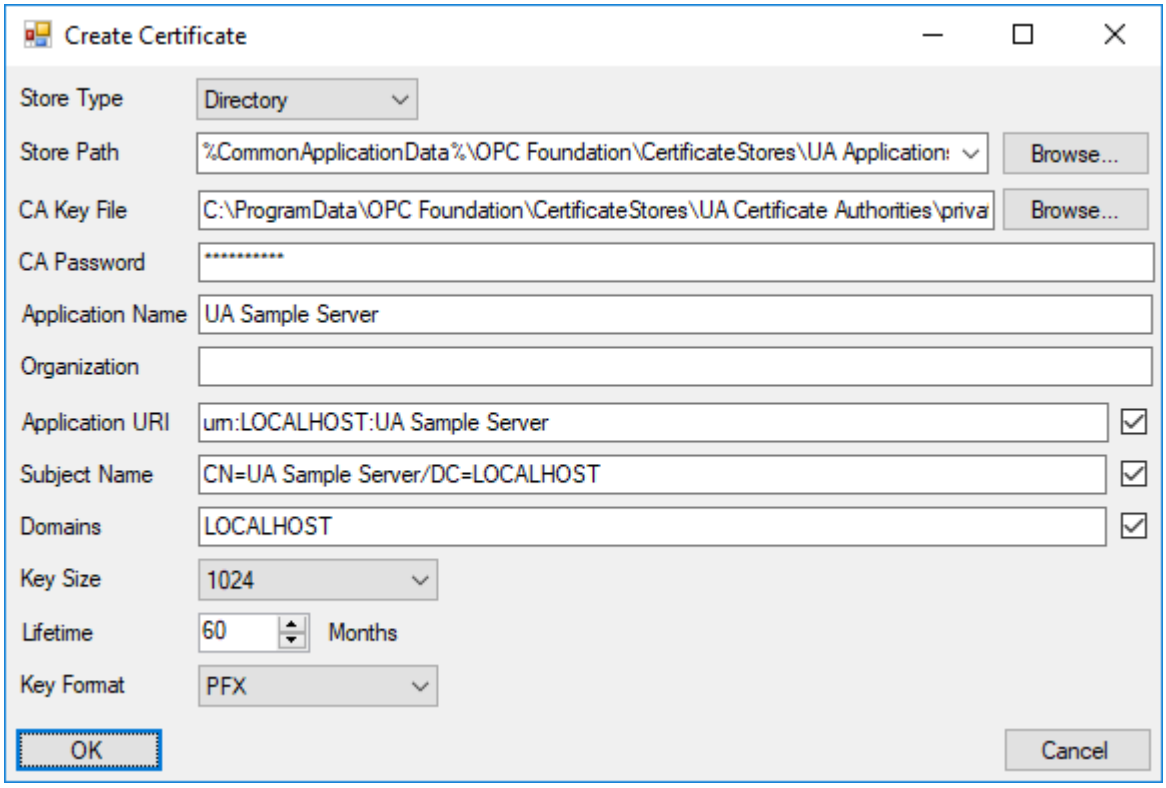
e) In the **CA Key File** box, make sure the certificate authority's key file is selected.

If it is not selected, you can browse for it. It should be located at:

```
C:\ProgramData\OPC Foundation\CertificateStores\UA Certificate Authorities\Private
\<authority name> [<ID string>].pfx
```

f) In the **CA Password** box, type the password for the certificate authority.

g) Complete the remaining certificate information (e.g., Application Name, Organization, etc.) as needed.




***Example of CA-signed certificate settings for UA Sample Server running on localhost***

h) Click **OK**.
The server certificate is created, and the certificate file is saved in the UA Applications store. If there is an old certificate file in the store, you might be prompted to overwrite or delete it. It should be safe to do so as long as there are no clients connected to the server.

i) Restart the UA Sample Server to make sure it uses the new certificate.

9. In your OPC UA server, import the client's self-signed certificate into the server's certificate store.
Example using UA Sample Server and UA Configuration Tool:

a) In the UA Configuration Tool, click the **Manage Certificates** tab.

b) In the **Store Type** list, make sure **Directory** is selected.

c) In the **Store Path** box, make sure **UA Applications** is selected.

d) Click **Import Certificate to Store**.
A standard *Open File* dialog box is displayed.

e) Use the dialog box to locate and select the client's certificate file.

For projects running on Windows and Windows Embedded Standard, the file is located in your project folder at: *<project name>*\Config\UAClientCertificate.der

f) Click **Open**.
You will be asked to confirm the import.

g) Click **Yes**.
The selected file is imported into the UA Applications store.

10. In your OPC UA server, reissue the client's self-signed certificate as a CA-signed certificate.
Example using UA Sample Server and UA Configuration Tool:

a) In the UA Configuration Tool, click the **Manage Certificates** tab.

b) Click **Select and Issue Certificate**.
The *Manage Certificates in Certificate Store* dialog box is displayed.

c) In the **Store Type** list, make sure **Directory** is selected.

d) In the **Store Path** box, make sure **UA Applications** is selected.

e) In the list of certificates, select the client certificate, and then click **OK**.
The *Create Certificate* dialog box is displayed with the client certificate's existing settings.

f) In the **Store Type** list, make sure **Directory** is selected.

g) In the **Store Path** box, make sure **UA Applications** is selected.

h) In the **CA Key File** box, make sure the certificate authority's key file is selected.

If it is not selected, you can browse for it. It should be located at:

**C:\ProgramData\OPC Foundation\CertificateStores\UA Certificate Authorities\Private \<*authority name*> [<*ID string*>].pfx**

i) In the **CA Password** box, type the password for the certificate authority.

j) In the **Key Format** list, select the format for the client's key file.

For projects running on Windows and Windows Embedded Standard, select **PEM**.

k) Click **OK**.
You are asked if it is okay to delete the original certificate.

l) Click **Yes**.
The reissued, CA-signed certificate and key files are saved in the UA Applications store.

11. Copy the client's CA-signed certificate and key files from the server's certificate store to the client's certificate store in your project folder.

These files should replace the existing, self-signed certificate and key files that you created earlier.

a) Locate the client's CA-signed certificate file in the server's certificate store.

For UA Sample Server and UA Configuration Tool, the file should be located at:

**C:\ProgramData\OPC Foundation\CertificateStores\UA Applications\Certs\<*application name*> [<*ID string*>].der**

Please note the certificate file has a new ID string that was generated by the OPC UA server when it signed and reissued the certificate. A certificate store uses these ID strings to manage its own certificate files, and the same file may have a different ID string in each store (e.g., server store versus client store).

b) Copy (not move!) the CA-signed certificate file to the client's certificate store in your project folder, and then rename it so that it replaces the existing, self-signed certificate file there.

For projects running on Windows and Windows Embedded Standard, copy the file to: *<project name>*\Config\UAClientCertificate.der

The name of the CA-signed certificate file must match the name of the self-signed certificate file that it is replacing. If it does not, the client will not be able to see the file. You might need to copy the file name and then delete the existing file before you can rename the new file.

c) Locate the client's CA-signed key file in the server's certificate store.

For UA Sample Server and UA Configuration Tool, the file should be located at either…

**C:\ProgramData\OPC Foundation\CertificateStores\UA Applications\Private \<*application name*> [<*ID string*>].pem**

…or…

**C:\ProgramData\OPC Foundation\CertificateStores\UA Applications\Private \<*application name*> [<*ID string*>].pfx**

…depending on the key format you selected in the previous step.

Again, please note the key file has a new ID string that was generated by the OPC UA server when it signed and reissued the key.

d) Copy (not move!) the CA-signed key file to the client's certificate store in your project folder, and then rename it so that it replaces the existing, self-signed key file there.

For projects running on Windows and Windows Embedded Standard, copy the file to: *<project name>*\Config\UAClientCertificatePrivateKey.pem

The name of the CA-signed key file must match the name of the self-signed key file that it is replacing. If it does not, the client will not be able to see the file. You might need to copy the file name and then delete the existing file before you can rename the new file.

12. If your project will run on Windows or Windows Embedded Standard, add the certificate authority to your project's issuer and trust lists:

a) Locate the certificate authority's certificate file.

For UA Sample Server and UA Configuration Tool, the file should be located at:

**C:\ProgramData\OPC Foundation\CertificateStores\UA Certificate Authorities\Certs \<*authority name*> [<*ID string*>].der**

b) Copy (not rename or move) the certificate file to: *<project name>*\Config\IssuerList\Certs\

c) Copy (not rename or move) the certificate file to: *<project name>*\Config\TrustList\Certs\

> **Note:** These are the default locations in your project folder where certificate files are stored. You can change these locations by changing the **Trust List** or **Issuer Certificate List** settings in the *Security Settings* dialog box, as shown above, and you may do this if, for example, you have a single folder where you store certificates for several different projects. In most cases, however, we do not recommend it.

When the client and server both have CA-signed certificates, and when the certificate authority that signed the certificates has been added to both programs' trust lists, the OPC UA connection should be properly configured for secure, two-way communication.

## CREATE A GROUP OF REDUNDANT OPC CONNECTIONS

Create a group of redundant connections that your OPC UA or OPC XML/DA client worksheet can use instead of an individual connection.

Before you begin this task, you should have two or more OPC connections that you can group together. You can create a group of only one connection, but it would serve no purpose other than to get the status of the connection.

A redundancy group consists of individual connections arranged in order. If the first connection in the group fails (due to BAD status) or times out (due to inactivity), the OPC client worksheet will use the second connection instead. Then, if the second connection fails or times out, the worksheet will use the third connection, and so on until the worksheet reaches the last connection in the group. If the last connection also fails or times out, the client will start over with the first connection in the group.

A properly configured redundancy group also provides continuously updated status information about each connection in the group. That allows you to monitor the connections during project run time.

After you create a redundancy group, you can select it in your OPC client worksheet in the same way that you would select an individual connection.

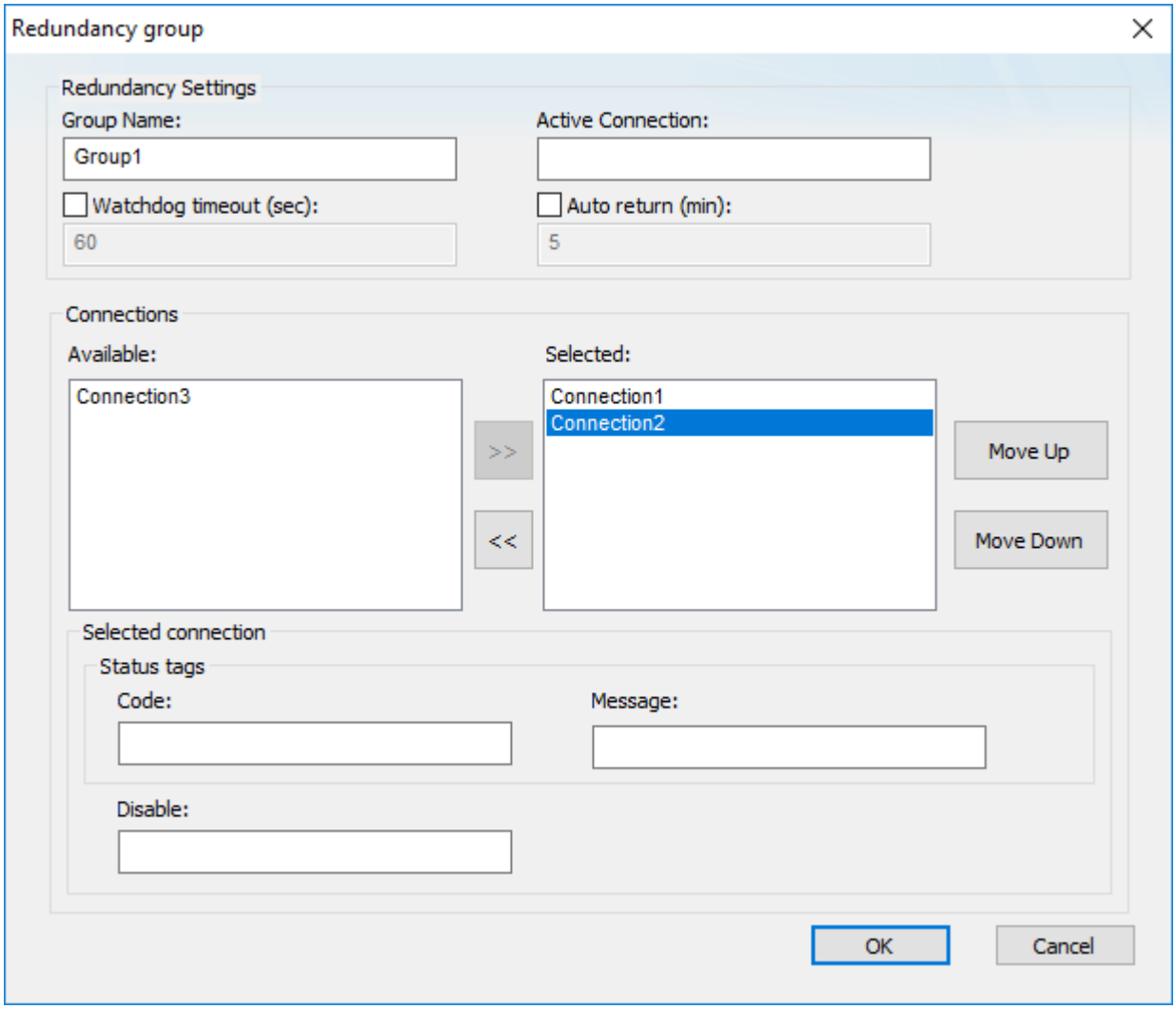
Keep in mind that if you configure several different worksheets to use the same redundancy group, they will all use the same connection at the same time during project run time. That might put too much load on a single connection and cause it to fail when it would not otherwise. To avoid that, create multiple groups — each with the connections arranged in a different order (e.g., ABC, CAB, BCA) — and then select a different group for each worksheet. This will provide rudimentary load balancing.

> **Note:** The OPC DA 2.05 client in EcoStruxure Machine SCADA Expert 2020 does not support redundancy groups at this time.

To create a redundancy group:

1. In the **Comm** tab of the Project Explorer, expand the folder for the type of OPC that you are using, either **OPC UA** or **OPC XML/DA**.
2. In that folder, right-click the **Redundancy Group** folder, and then on the shortcut menu, click **Insert**.

The *Redundancy Group* dialog box is displayed.




***Example of redundancy group settings***

3. In the **Group Name** box, type a unique name for the group.
   This name will be displayed in the list of available connections in your OPC client worksheet.
4. In the **Active Connection** box, type the name of a project tag of String type.
   The specified tag will receive the name of the connection that is currently being used during project run time. In other words, when the connection changes for any reason, the tag value changes to match. You can also change the tag value yourself, to control which connection is used. In both cases, the tag value is the full name of the connection (e.g., `Connection 1`). This setting is optional.
5. If you want a connection to automatically time out after a period of inactivity, rather than wait for it to actually fail (with BAD status), select the **Watchdog timeout** option.
   The default timeout is 60 seconds after the last change reported by the OPC Server. If the field device has less frequent changes, you need to either increase the timeout or set up a "heartbeat" on the field device to keep the connection active.

> **Note:** This watchdog can only watch the connection between the OPC Client (i.e., your project) and the OPC Server. It cannot watch the connection between the OPC Server and the field device. The OPC Server itself is responsible for monitoring that connection.

6. If you want the client to automatically return to the first connection in the group when it is possible to do so, rather than continue through all of the connections in the group, select the **Auto return** option.

   The default return time is 5 minutes, which means the client will try to return to the first connection every 5 minutes after the connection was lost. If the client cannot reestablish the first connection, it will continue with its current connection and try again later.

7. In the **Connections** area, in the **Available** list on the left, select a connection that you want to include in the group, and then click **>>**.

   There is no limit on the number of connections that you can include in a group.

   The connection is added to the **Selected** list on the right.

8. Repeat the previous step for each connection that you want to include in the group.

9. Select a connection in the **Selected** list on the right, and then do the following:

   a) Use the **Move Up** and **Move Down** buttons to move the selected connection to the desired position in the list.

   b) In the **Code** box, type the name of a project tag of Integer or String type.

   The specified tag will receive a continuously updated status code for the selected connection during project run time.

   c) In the **Message** box, type the name of a project tag of String type.

   The specified tag will receive a continuously updated status message for the selected connection during project run time.

   d) In the **Disable** box, type a tag name, expression, or literal value.

   When it evaluates as TRUE (i.e., non-zero) during project run time, the selected connection will be disabled and therefore skipped.

> **Note:** The **Code**, **Message**, and **Disable** settings are all optional and unique for each connection.

10. Repeat the previous step for each connection in the group.

11. Click **OK** to save your changes and close the dialog box.

The new group is saved in the appropriate **Redundancy Group** folder in the Project Explorer, and it becomes available for selection in the corresponding client worksheets.

During project run time, each connection in the redundancy group can have one of the following statuses:

| Code | Message |
|---|---|
| 1 | The connection to the OPC Server is established, and it is running in normal mode. |
| 1 | The connection to the OPC Server has been reestablished, and the watchdog has been reset. |
| -1 | [Any of the BAD messages that are described in List of read/write status codes and messages for OPC UA on page 599.] |
| -2 | The OPC Server stopped reporting changes from the field device (e.g., the PLC), and the watchdog timeout has elapsed. |
| -3 | Trying to establish connection to the OPC Server. |

## CREATE A NEW OPC UA CLIENT WORKSHEET

Create and configure an OPC UA Client worksheet to associate project tags with OPC server items.

Before you begin this task, you must have created at least one OPC UA server connection that the client worksheet can use. For more information, see Create a new OPC UA connection on page 576.

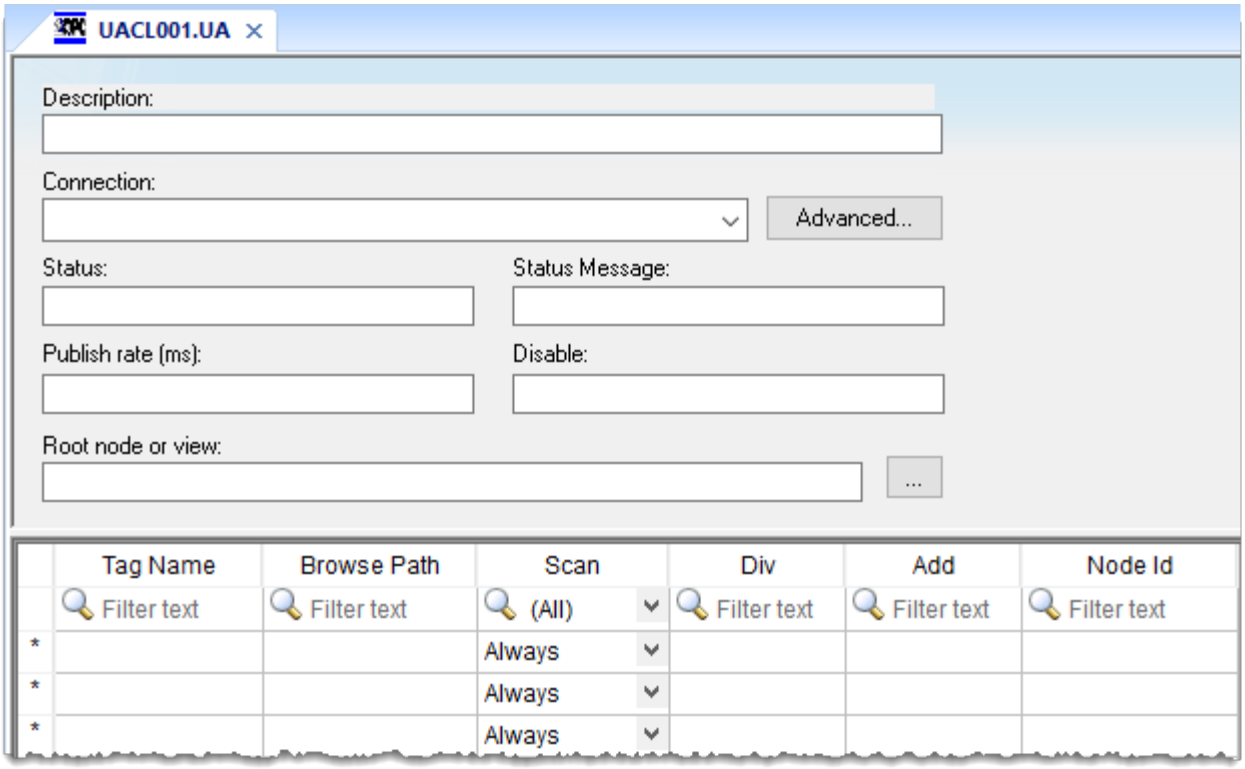
You should also be familiar with how to edit worksheets in the project development environment.

> **Tip:** You can create multiple worksheets and then configure them to have different settings, so you do not need to make any single worksheet unnecessarily large or complicated.

To configure a new OPC UA Client worksheet:

1. Do one of the following:
   - On the **Insert** tab of the ribbon, in the **Communication** group, click **OPC Client**, and then select **OPC UA Client** from the drop-down list; or
   - In the **Comm** tab of the Project Explorer, right-click the **OPC UA** folder, and then on the shortcut menu, click **Insert**.

   A new OPC UA Client worksheet is displayed.




2. In the **Description** box, type a description of the worksheet.
   This is for documentation purposes only and does not affect the execution of the worksheet.
3. In the **Connection** list, select the connection or redundancy group that you created earlier.
4. In the **Status** box, type the name of a project tag (Integer type) that will receive connection status codes during project run time, and then in the **Status Message** box, type the name of a project tag (String type) that will receive the corresponding status messages.
5. In the **Publish Rate** box, type the frequency (in milliseconds) at which the client will request updates from the server.
6. In the **Disable** box, type a tag/expression.
   While this tag/expression evaluates as TRUE (non-zero), the worksheet will not be executed and it will not use the specified connection. If any other OPC UA Client worksheets share the same connection, however, the connection will remain open and active unless those worksheets are also disabled.
7. In the **Root node or view** box, specify a server node that will serve as the root for all browse paths in the worksheet body.

Specifying a root node makes it easier to find items and improves run-time performance.

8. For each project tag that you want to associate with an OPC server item/node, complete a row in the worksheet body:
   a) In the **Tag Name** field, type the name of a project tag.
      You can type a string expression in this field (e.g., `{MyTagName}`), but if you do, make sure you have also configured the **Reload trigger** setting in the advanced settings.
   b) In the **Browse Path** field, do one of the following: type the full path of the server item; or right-click in this field, and then on the shortcut menu, click **Browse** in order to browse the server's list of items.

      > **Note:** If you have selected the **Enable bit notation** option in the advanced settings, you can select a specific bit of a server node value by appending the bit number to the item name (e.g., *<node path>.<bit>*). This is supported only for unsigned variables and 32-bit signed variables; if you try to do it with 16-bit or 8-bit signed variables, the most significant bit (MSB) will not work properly.

   c) In the **Scan** field, select either **Always** to have the row continuously scanned (i.e., processed) during project run time or **Screen** to have the row scanned only when a project screen that uses the specified tag is open.
   d) In the **Div** field, type a number to be used for scaling during project run time. This is optional.
      When a value is read from the server, it is divided by this number. When a value is written to the server, it is multiplied by this number.
   e) In the **Add** field, type a number to be used for scaling during project run time. This is optional.
      When a value is read from the server, this number is added to it. When a value is written to the server, this number is subtracted from it.
   f) In the **Node Id** field, a node ID is automatically generated from the browse path.
      The node ID is the actual value that your project will use to subscribe to the item during run time. If your project runtime log contains errors indicating that
      You can type a string expression in this field (e.g., `{MyNodeId}`), but if you do, make sure you have also configured the **Accept tag name in the Node Id column** and **Reload trigger** settings in the advanced settings.

   Both tag expansion and array distribution are enabled by default for all OPC UA Client worksheets. You can disable tag expansion for this worksheet by clearing the **Enable Tag Expansion** option in the advanced settings. You cannot disable array distribution.
9. When you are done, save and close the worksheet.
   The worksheet is saved in the **OPC UA** folder in the Project Explorer.

In order for your OPC UA Client worksheet(s) to be scanned during project run time, the OPC UA Client Runtime task in your project must be started. As such, the first time you configure and save a worksheet, the task's startup mode is set to **Automatic**. This is done for your convenience, and it means that when you run your project, the task will be started and the worksheet(s) will be scanned. You can set the task's startup mode back to **Manual**, however, if you want more control over how your project runs. For more information, see Runtime Tasks on page 136.

The worksheet is continuously scanned during project run time, so that the configured project tags and server items are updated as needed. Also, the project tags that you specified for the **Status** and **Status Message** settings in the worksheet header will receive the following possible values:

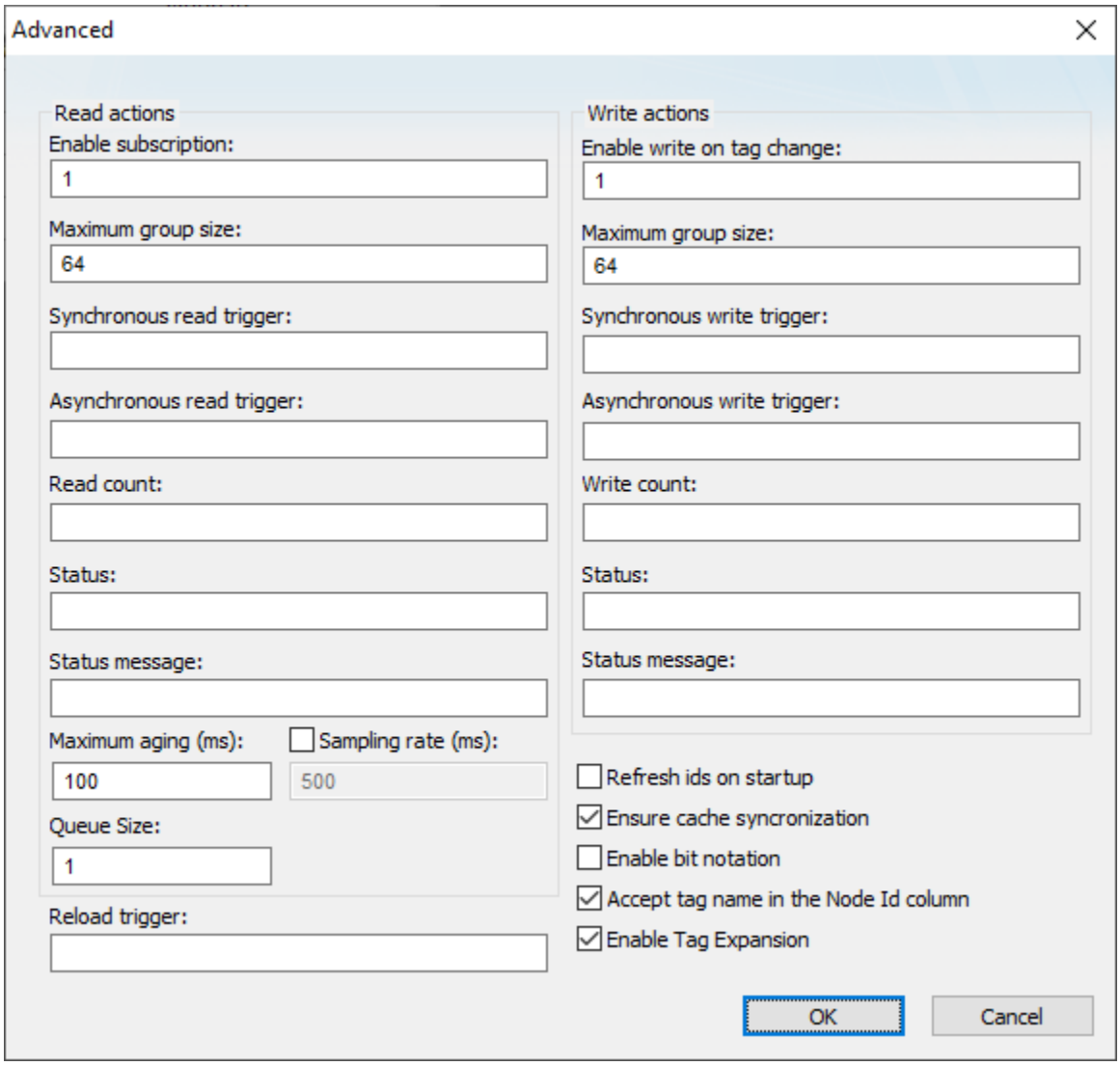
| Status | Status Message |
|---|---|
| 0 | The connection to the server is deactivated by the user of the client API. |
| 1 | The connection to the server is established and is working in normal mode. |
| 2 | The monitoring of the connection to the server indicated a potential connection problem. |
| 3 | The monitoring of the connection to the server detected an error and is trying to reconnect to the server. |
| 4 | The server sent a shutdown event and the client API tries a reconnect. |

| Status | Status Message |
|---|---|
| 5 | The client was not able to reuse the old session and created a new session during reconnect. This requires to redo register nodes for the new session. |
| 6 | The server time is two hours or more different from the client time. |
| 7 | Timeout connecting to the server. |
| 8 | Connecting to the server. |
| 9 | Host Unknonwn |
| 10 | The requested protocol is not supported, please check your connection URL. |

**Tip:** As long as the client is communicating normally with the server, the read/write status codes and messages (as configured in the advanced settings) will provide more information about each operation.

In certain situations, if the worksheet does not behave as expected during project run time — and especially if you see OPC communication errors in the runtime log — you might need to adjust the worksheet's advanced

settings: in the worksheet header, click **Advanced**, and then in the *Advanced* dialog box, review and configure the settings.




***Advanced settings for the OPC UA Client worksheet***

### Read actions

Advanced settings that control how values are read from the OPC UA server.

#### Enable subscription

While this value is TRUE (non-zero), the client will subscribe to the server nodes so that it can receive notifications when the node values change.

This setting has a default value of 1, which indicates it is always enabled. If it is disabled, or if you type a tag/expression that will cause it to be disabled during project run time, you might need to use read triggers instead (see below).

#### Maximum group size

The maximum number of server nodes that may be read in a single read operation.

For example, if you have 1000 items/rows configured in the worksheet and **Maximum group size** is set to 100, 10 read operations will be performed when a read is triggered.

#### Synchronous read trigger

When the value of this tag/expression changes, the client reads all of the node values from the server. The read operation is performed synchronously, which means all other client operations are blocked until the read operation is finished. When the read operation is finished, the client increments the tag configured in **Read count**.

#### Asynchronous read trigger

The same as **Synchronous read trigger** except that the read operation is performed asynchronously, which means that other client operations may continue while the read operation is being performed.

#### Read count

The name of a project tag (Integer type) that will receive a count of the number of read operations performed since the project was run.

#### Status

The name of a project tag (Integer type) that will receive a status code for the last read operation performed by a trigger. If the status code is `0`, the read operation finished successfully. For all other status codes, see List of read/write status codes and messages for OPC UA on page 599.

#### Status message

The name of a project tag (String type) that will receive the corresponding status message.

#### Maximum aging

The maximum age (in milliseconds) of values that will be accepted from the server's cache. If a value is older than this, the server will be forced to get the latest value from the field device.

#### Sampling rate

When this option is selected, you can change the rate (in milliseconds) at which the OPC server reads from field devices. By default, the sampling rate is half the publishing rate.

#### Queue size

### Write actions

Advanced settings that control how values are written to the OPC UA server.

#### Enable write on tag change

While this value is TRUE (non-zero), an asynchronous write operation will be performed automatically whenever the value of a project tag changes. All tag changes that occured since the last scan of the worksheet will be written in a single write operation, or in multiple write operations if the number of tag changes exceeds the maximum group size.

This setting has a default value of 1, which indicates it is always enabled. If it is disabled, or if you type a tag/expression that will cause it to be disabled during project run time, you might need to use write triggers instead (see below).

#### Maximum group size

The maximum number of tag changes that may be written in a single write operation.

For example, if you have 1000 items/rows configured in the worksheet and **Maximum group size** is set to 100, 10 write operations will be performed during each scan of the worksheet.

#### Synchronous write trigger

When the value of this tag/expression changes, the client writes all of the tag values to the server. The write operation is performed synchronously, which means all other client operations are blocked until the write operation is finished. When the write operation is finished, the client increments the tag configured in **Write count**.

#### Asynchronous write trigger

The same as **Synchronous write trigger** except that the write operation is performed asynchronously, which means that other client operations may continue while the write operation is being performed.

#### Write count

The name of a project tag (Integer type) that will receive a count of the number of write operations performed since the project was run.

#### Status

The name of a project tag (Integer type) that will receive a status code for the last write operation performed by a trigger. If the status code is `0`, the write operation finished successfully. For all other status codes, see List of read/write status codes and messages for OPC UA on page 599.

#### Status message

The name of a project tag (String type) that will receive the corresponding status message.

### Other

Other settings that control how this worksheet communicates with the OPC UA server.

#### Reload trigger

The name of a project tag (Boolean, Integer, or Real type) that can be used as a trigger. When the value of this tag changes, the worksheet is reloaded. String expressions configured in the body of the worksheet are reevaluated only when the worksheet is reloaded. After the worksheet is reloaded, the value of this tag will be reset to 0.

#### Refresh IDs on startup

When this option is selected, the node IDs in the worksheet will be automatically refreshed from the specified browse paths every time the project is run. This might resolve any node IDs that are missing or invalid.

Refreshing IDs like this can cause the project to take longer to run, however, so if you select this option, you should also specify a root node (i.e., in the **Root node or view** box in the worksheet header) to restrict the list of server items that must scanned.

The OPC server software might not support this option, depending on how it has implemented the OPC standard. For more information, see the documentation for that software.

#### Ensure cache synchronization

When this option is selected, the client will wait after each write operation for confirmation from the server that the node values actually changed. If the client does not receive confirmation, it will restore the previous tag values.

#### Enable bit notation

When this option is selected, bit notation is allowed in the **Item** column of the worksheet body.

#### Accept tag name in the Node Id column

When this option is selected, you can type string expressions in the **Node Id** column of the worksheet, but if you do, make sure you have also configured the **Reload trigger** setting (see above).

#### Enable Tag Expansion

When this option is selected, tag expansion is enabled for this worksheet. You can disable it for this worksheet without affecting any other OPC client worksheets, and you may choose to do so if you want more control over how OPC server items are associated with your project tags.

> **Note:** This software does not normally use the Triggering Mode that is defined the OPC standard. Instead, it allows any change in any tag/expression to be used as a trigger. If you want to use Triggering Mode, configure one worksheet to read the trigger values, and then configure another worksheet that specifies the read values as triggers.

## LIST OF READ/WRITE STATUS CODES AND MESSAGES FOR OPC UA

This is a list of the possible status codes and messages that might be generated by read/write operations in OPC UA.

| Status Code | Status Message |
|---|---|
| -2159476736 | Bad – Max Connections Reached |
| -2159411200 | Bad – Syntax Error |
| -2159345664 | Bad – Would Block |
| -2159280128 | Bad – Expected Stream To Block |
| -2159214592 | Bad – Operation Abandoned |
| -2159149056 | Bad – Waiting For Response |
| -2159083520 | Bad – No Data Available |
| -2159017984 | Bad – End Of Stream |
| -2147352576 | Bad – Internal Error |
| -2147287040 | Bad – Out Of Memory |
| -2147221504 | Bad – Resource Unavailable |
| -2147155968 | Bad – Communication Error |
| -2147090432 | Bad – Encoding Error |
| -2147024896 | Bad – Decoding Error |
| -2146959360 | Bad – Encoding Limits Exceeded |
| -2146893824 | Bad – Unknown Response |
| -2146828288 | Bad – Timeout |
| -2146762752 | Bad – Service Unsupported |
| -2146697216 | Bad – Shutdown |
| -2146631680 | Bad – Server Not Connected |
| -2146566144 | Bad – Server Halted |
| -2146500608 | Bad – Nothing To Do |
| -2146435072 | Bad – Too Many Operations |
| -2146369536 | Bad – Data Type Id Unknown |
| -2146304000 | Bad – Certificate Invalid |
| -2146238464 | Bad – Security Checks Failed |
| -2146172928 | Bad – Certificate Time Invalid |
| -2146107392 | Bad – Certificate Issuer Time Invalid |
| -2146041856 | Bad – Certificate Host Name Invalid |
| -2145976320 | Bad – Certificate Uri Invalid |
| -2145910784 | Bad – Certificate Use Not Allowed |
| -2145845248 | Bad – Certificate Issuer Use Not Allowed |
| -2145779712 | Bad – Certificate Untrusted |
| -2145714176 | Bad – Certificate Revocation Unknown |
| -2145648640 | Bad – Certificate Issuer Revocation Unknown |
| -2145583104 | Bad – Certificate Revoked |

| Status Code | Status Message |
|---|---|
| -2145517568 | Bad – Certificate Issuer Revoked |
| -2145452032 | Bad – User Access Denied |
| -2145386496 | Bad – Identity Token Invalid |
| -2145320960 | Bad – Identity Token Rejected |
| -2145255424 | Bad – Secure Channel Id Invalid |
| -2145189888 | Bad – Invalid Timestamp |
| -2145124352 | Bad – Nonce Invalid |
| -2145058816 | Bad – Session Id Invalid |
| -2144993280 | Bad – Session Closed |
| -2144927744 | Bad – Session Not Activated |
| -2144862208 | Bad – Subscription Id Invalid |
| -2144731136 | Bad – Request Header Invalid |
| -2144665600 | Bad – Timestamps To Return Invalid |
| -2144600064 | Bad – Request Cancelled By Client |
| -2144272384 | Bad – No Communication |
| -2144206848 | Bad – Waiting For Initial Data |
| -2144141312 | Bad – Node Id Invalid |
| -2144075776 | Bad – Node Id Unknown |
| -2144010240 | Bad – Attribute Id Invalid |
| -2143944704 | Bad – Index Range Invalid |
| -2143879168 | Bad – Index Range No Data |
| -2143813632 | Bad – Data Encoding Invalid |
| -2143748096 | Bad – Data Encoding Unsupported |
| -2143682560 | Bad – Not Readable |
| -2143617024 | Bad – Not Writable |
| -2143551488 | Bad – Out Of Range |
| -2143485952 | Bad – Not Supported |
| -2143420416 | Bad – Not Found |
| -2143354880 | Bad – Object Deleted |
| -2143289344 | Bad – Not Implemented |
| -2143223808 | Bad – Monitoring Mode Invalid |
| -2143158272 | Bad – Monitored Item Id Invalid |
| -2143092736 | Bad – Monitored Item Filter Invalid |
| -2143027200 | Bad – Monitored Item Filter Unsupported |
| -2142961664 | Bad – Filter Not Allowed |
| -2142896128 | Bad – Structure Missing |
| -2142830592 | Bad – Event Filter Invalid |
| -2142765056 | Bad – Content Filter Invalid |
| -2142699520 | Bad – Filter Operand Invalid |

| Status Code | Status Message |
|---|---|
| -2142633984 | Bad – Continuation Point Invalid |
| -2142568448 | Bad – No Continuation Points |
| -2142502912 | Bad – Reference Type Id Invalid |
| -2142437376 | Bad – Browse Direction Invalid |
| -2142371840 | Bad – Node Not In View |
| -2142306304 | Bad – Server Uri Invalid |
| -2142240768 | Bad – Server Name Missing |
| -2142175232 | Bad – Discovery Url Missing |
| -2142109696 | Bad – Sempahore File Missing |
| -2142044160 | Bad – Request Type Invalid |
| -2141978624 | Bad – Security Mode Rejected |
| -2141913088 | Bad – Security Policy Rejected |
| -2141847552 | Bad – Too Many Sessions |
| -2141782016 | Bad – User Signature Invalid |
| -2141716480 | Bad – Application Signature Invalid |
| -2141650944 | Bad – No Valid Certificates |
| -2141585408 | Bad – Request Cancelled By Request |
| -2141519872 | Bad – Parent Node Id Invalid |
| -2141454336 | Bad – Reference Not Allowed |
| -2141388800 | Bad – Node Id Rejected |
| -2141323264 | Bad – Node Id Exists |
| -2141257728 | Bad – Node Class Invalid |
| -2141192192 | Bad – Browse Name Invalid |
| -2141126656 | Bad – Browse Name Duplicated |
| -2141061120 | Bad – Node Attributes Invalid |
| -2140995584 | Bad – Type Definition Invalid |
| -2140930048 | Bad – Source Node Id Invalid |
| -2140864512 | Bad – Target Node Id Invalid |
| -2140798976 | Bad – Duplicate Reference Not Allowed |
| -2140733440 | Bad – Invalid Self Reference |
| -2140667904 | Bad – Reference Local Only |
| -2140602368 | Bad – No Delete Rights |
| -2140536832 | Bad – Server Index Invalid |
| -2140471296 | Bad – View Id Unknown |
| -2140340224 | Bad – Too Many Matches |
| -2140274688 | Bad – Query Too Complex |
| -2140209152 | Bad – No Match |
| -2140143616 | Bad – Max Age Invalid |
| -2140078080 | Bad – History Operation Invalid |

| Status Code | Status Message |
|---|---|
| -2140012544 | Bad – History Operation Unsupported |
| -2139947008 | Bad – Write Not Supported |
| -2139881472 | Bad – Type Mismatch |
| -2139815936 | Bad – Method Invalid |
| -2139750400 | Bad – Arguments Missing |
| -2139684864 | Bad – Too Many Subscriptions |
| -2139619328 | Bad – Too Many Publish Requests |
| -2139553792 | Bad – No Subscription |
| -2139488256 | Bad – Sequence Number Unknown |
| -2139422720 | Bad – Message Not Available |
| -2139357184 | Bad – Insufficient Client Profile |
| -2139291648 | Bad – Tcp Server Too Busy |
| -2139226112 | Bad – Tcp Message Type Invalid |
| -2139160576 | Bad – Tcp Secure Channel Unknown |
| -2139095040 | Bad – Tcp Message Too Large |
| -2139029504 | Bad – Tcp Not Enough Resources |
| -2138963968 | Bad – Tcp Internal Error |
| -2138898432 | Bad – Tcp Endpoint Url Invalid |
| -2138832896 | Bad – Request Interrupted |
| -2138767360 | Bad – Request Timeout |
| -2138701824 | Bad – Secure Channel Closed |
| -2138636288 | Bad – Secure Channel Token Unknown |
| -2138570752 | Bad – Sequence Number Invalid |
| -2138505216 | Bad – Configuration Error |
| -2138439680 | Bad – Not Connected |
| -2138374144 | Bad – Device Failure |
| -2138308608 | Bad – Sensor Failure |
| -2138243072 | Bad – Out Of Service |
| -2138177536 | Bad – Deadband Filter Invalid |
| -2137587712 | Bad – Refresh In Progress |
| -2137522176 | Bad – Condition Already Disabled |
| -2137456640 | Bad – Condition Disabled |
| -2137391104 | Bad – Event Id Unknown |
| -2137325568 | Bad – No Data |
| -2137194496 | Bad – Data Lost |
| -2137128960 | Bad – Data Unavailable |
| -2137063424 | Bad – Entry Exists |
| -2136997888 | Bad – No Entry Exists |
| -2136932352 | Bad – Timestamp Not Supported |

| Status Code | Status Message |
|---|---|
| -2136276992 | Bad – Invalid Argument |
| -2136211456 | Bad – Connection Rejected |
| -2136145920 | Bad – Disconnect |
| -2136080384 | Bad – Connection Closed |
| -2136014848 | Bad – Invalid State |
| -2135425024 | Bad – Request Too Large |
| -2135359488 | Bad – Response Too Large |
| -2135228416 | Bad – Event Not Acknowledgeable |
| -2135097344 | Bad – Invalid Timestamp Argument |
| -2135031808 | Bad – Protocol Version Unsupported |
| -2134966272 | Bad – State Not Active |
| -2134835200 | Bad – Filter Operator Invalid |
| -2134769664 | Bad – Filter Operator Unsupported |
| -2134704128 | Bad – Filter Operand Count Mismatch |
| -2134638592 | Bad – Filter Element Invalid |
| -2134573056 | Bad – Filter Literal Invalid |
| -2134507520 | Bad – Identity Change Not Supported |
| -2134376448 | Bad – Not Type Definition |
| -2134310912 | Bad – View Timestamp Invalid |
| -2134245376 | Bad – View Parameter Mismatch |
| -2134179840 | Bad – View Version Invalid |
| -2134114304 | Bad – Condition Already Enabled |
| -2134048768 | Bad – Dialog Not Active |
| -2133983232 | Bad – Dialog Response Invalid |
| -2133917696 | Bad – Condition Branch Already Acked |
| -2133852160 | Bad – Condition Branch Already Confirmed |
| -2133786624 | Bad – Condition Already Shelved |
| -2133721088 | Bad – Condition Not Shelved |
| -2133655552 | Bad – Shelving Time Out Of Range |
| -2133590016 | Bad – Aggregate List Mismatch |
| -2133524480 | Bad – Aggregate Not Supported |
| -2133458944 | Bad – Aggregate Invalid Inputs |
| -2133393408 | Bad – Bound Not Found |
| -2133327872 | Bad – Bound Not Supported |
| -2133196800 | Bad – Aggregate Configuration Rejected |
| 2949120 | Good – Subscription Transferred |
| 3014656 | Good – Completes Asynchronously |
| 3080192 | Good – Overload |
| 3145728 | Good – Clamped |

| Status Code | Status Message |
|---|---|
| 9830400 | Good – Local Override |
| 10616832 | Good – Entry Inserted |
| 10682368 | Good – Entry Replaced |
| 10813440 | Good – No Data |
| 10878976 | Good – More Data |
| 10944512 | Good – Communication Event |
| 11010048 | Good – Shutdown Event |
| 11075584 | Good – Call Again |
| 11141120 | Good – Non Critical Timeout |
| 12189696 | Good – Results May Be Incomplete |
| 14221312 | Good – Data Ignored |
| 1080819712 | Uncertain – Reference Out Of Server |
| 1083113472 | Uncertain – No Communication Last Usable Value |
| 1083179008 | Uncertain – Last Usable Value |
| 1083244544 | Uncertain – Substitute Value |
| 1083310080 | Uncertain – Initial Value |
| 1083375616 | Uncertain – Sensor Not Accurate |
| 1083441152 | Uncertain – Engineering Units Exceeded |
| 1083506688 | Uncertain – Sub Normal |
| 1084489728 | Uncertain – Data Sub Normal |
| 1086062592 | Uncertain – Reference Not Deleted |
| 1086324736 | Uncertain – Not All Nodes Available |

## ENABLE THE OPC UA TRACE LOG

If you are having problems with OPC UA communications during project run time, you can have the project runtime server generate an OPC UA trace log.

This trace log is in addition to the project runtime log that is displayed in the Output window and LogWin tool. It contains much of the same information as the project runtime log (assuming the project runtime log is configured to include OPC UA messages), but it is saved to an external file and it can be configured to capture even more detailed information.

To enable the OPC UA trace log:

1. Stop your project if it is running, and then exit the software.
2. Use a text editor (e.g., Notepad) to open your project file (*<project name>*.APP) and add the following properties:

```
[OPC]
UaLogPath=<file path and name>
UaTraceLevel=<level>
```

   For **UaLogPath**, the file path is relative to your project folder. As such, if you specify only a file name (e.g., opcualog.txt), the file will be saved in your project folder. Keep in mind that the file path should be appropriate for the computer or device that hosts the project runtime server, not necessarily for the computer that you are using to develop your project.

   For **UaTraceLevel**, select one of the following values:

| Level | Description |
|---|---|
| 0 | NoTrace – Disables the trace.<br>This is the default if **UaLogPath** is not configured. |
| 1 | Errors – Internal system errors that require bug fixing. |
| 2 | Warnings – Internal system warnings and external errors.<br>This is the default if **UaLogPath** is configured. |
| 3 | Info – More detailed information about system events. |
| 4 | InterfaceCall – Information needed for debugging. |
| 5 | CtorDtor – Information needed for debugging. |
| 6 | Program – All message content. |
| 7 | FlowData – All messages. |

> **Note:** The trace levels are cumulative, which means Level 2 includes Level 1, Level 3 includes Levels 1 and 2, and so on.

3. Save and close your project file.

The next time you run your project, the log file is saved at the specified location. The file will continue to grow as long as the project runs and the trace log is enabled, and the higher the trace level, the more quickly the file will grow.

## *OPC XML/DA Client*

Use the OPC XML/DA Client worksheet and runtime task to establish communication between your project and a data exchange server that supports the OPC DA (a.k.a. OPC Classic) or OPC XML-DA interoperability standard.

OPC XML-DA is an improvement on OPC DA because it uses cross-platform technologies like XML and SOAP for web services. It also standardizes the SOAP messages exchanged between clients and servers, which allows the OPC standard to be implemented on different operating systems.

In this software, the OPC XML/DA Client feature uses an updated toolkit that can communicate with both OPC DA servers and OPC XML-DA servers. It can automatically detect which standard is supported by the connected server. If you are creating a new project and you need to configure it to communicate with an existing OPC DA server, we recommend you use OPC XML/DA Client instead of OPC DA 2.05 Client.

### CREATE A NEW OPC XML/DA CONNECTION

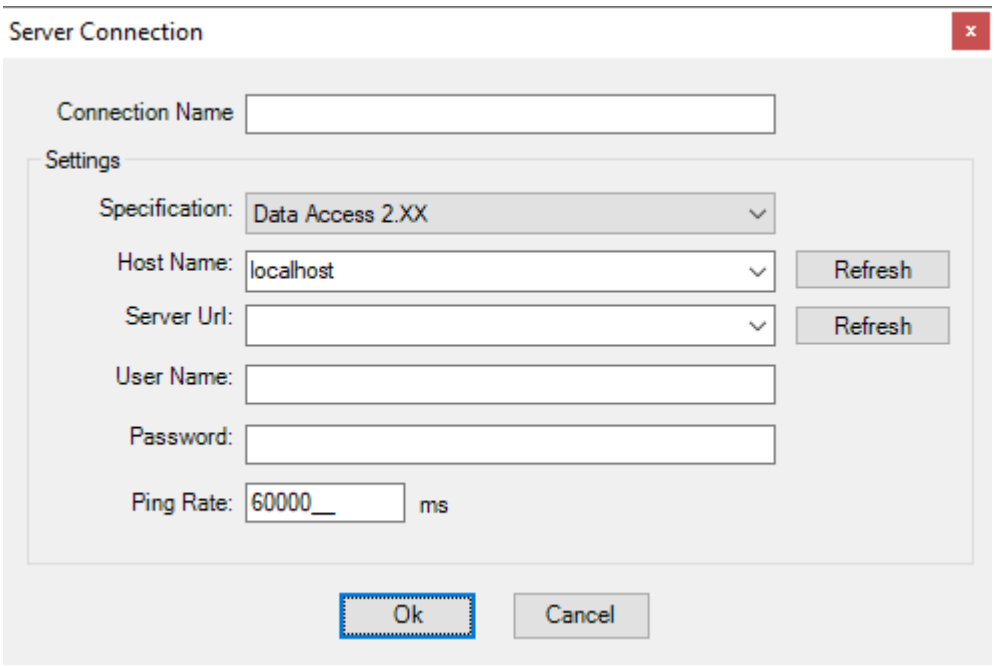
When you configure an OPC XML/DA Client worksheet, you must select the connection that the client will use. This task describes how to create that connection.

Before you begin this task, you should know the communication and security settings for the OPC XML/DA server to which you want to connect. If you do not, contact the server administrator.

To create a new connection to an OPC XML/DA server:

1. Do one of the following:
   - On the **Insert** tab of the ribbon, in the **Communication** group, click **OPC Client**, and then select **OPC XML/DA Connection** from the drop-down list; or
   - In the **Comm** tab of the Project Explorer, expand the **OPC XML/DA** folder, right-click the **Connections** folder, and then click **Insert** on the shortcut menu.

   The *Server Connection* dialog box is displayed.




2. In the **Connection Name** box, type a name for the connection.
   This name will be displayed in the **OPC XML/DA** > **Connections** folder in the Project Explorer, and it is the name you will look for when you configure the OPC XML/DA Client worksheet.
3. In the **Specification** list, select the specific OPC specificiation that is used by the OPC Server:
   - **Data Access 2.XX**
   - **Data Access 3.00**
   - **XML Data Access 1.00**
4. In the **Host Name** box, do one of the following:
   - Select the name or address of the computer that hosts the OPC Server. Hosts should broadcast their availability on the network. If the host you want does not appear in the list, click **Refresh** to update the list.
   - If you want to be able to change the host name during project run time, type an appropriate string expression (e.g., `{MyEndpointUrl}`). When the OPC XML/DA Client Runtime task is started, it will get the value of the string expression and then connect to that host. Please note this happens only when the task is started, which typically happens when the project itself is run. If the value of the string expression changes after the task is started, the task must be restarted in order to get the new value and then connect to the new host. To restart the task while the project is running, either use the *Runtime Tasks* dialog box or call the `EndTask` and `StartTask` functions.
5. In the **Server Url** list, select a specific OPC Server process on the selected host. If the process you want does not appear in the list, click **Refresh** to update the list from the host.
6. In the **User Name** and **Password** boxes, type your login credentials for the selected OPC Server.
7. In the **Ping Rate** box, type the frequency (in milliseconds) at which the client should ping the server to make sure the connection is still active.
   By default, the client pings the server once per minute.
8. Click **OK** to save your changes and close the *Server Configuration* dialog box.
   The connection is saved in the **OPC XML/DA** > **Connections** folder in the Project Explorer.

## CREATE A GROUP OF REDUNDANT OPC CONNECTIONS

Create a group of redundant connections that your OPC UA or OPC XML/DA client worksheet can use instead of an individual connection.

Before you begin this task, you should have two or more OPC connections that you can group together. You can create a group of only one connection, but it would serve no purpose other than to get the status of the connection.

A redundancy group consists of individual connections arranged in order. If the first connection in the group fails (due to BAD status) or times out (due to inactivity), the OPC client worksheet will use the second connection instead. Then, if the second connection fails or times out, the worksheet will use the third connection, and so on until the worksheet reaches the last connection in the group. If the last connection also fails or times out, the client will start over with the first connection in the group.

A properly configured redundancy group also provides continuously updated status information about each connection in the group. That allows you to monitor the connections during project run time.

After you create a redundancy group, you can select it in your OPC client worksheet in the same way that you would select an individual connection.

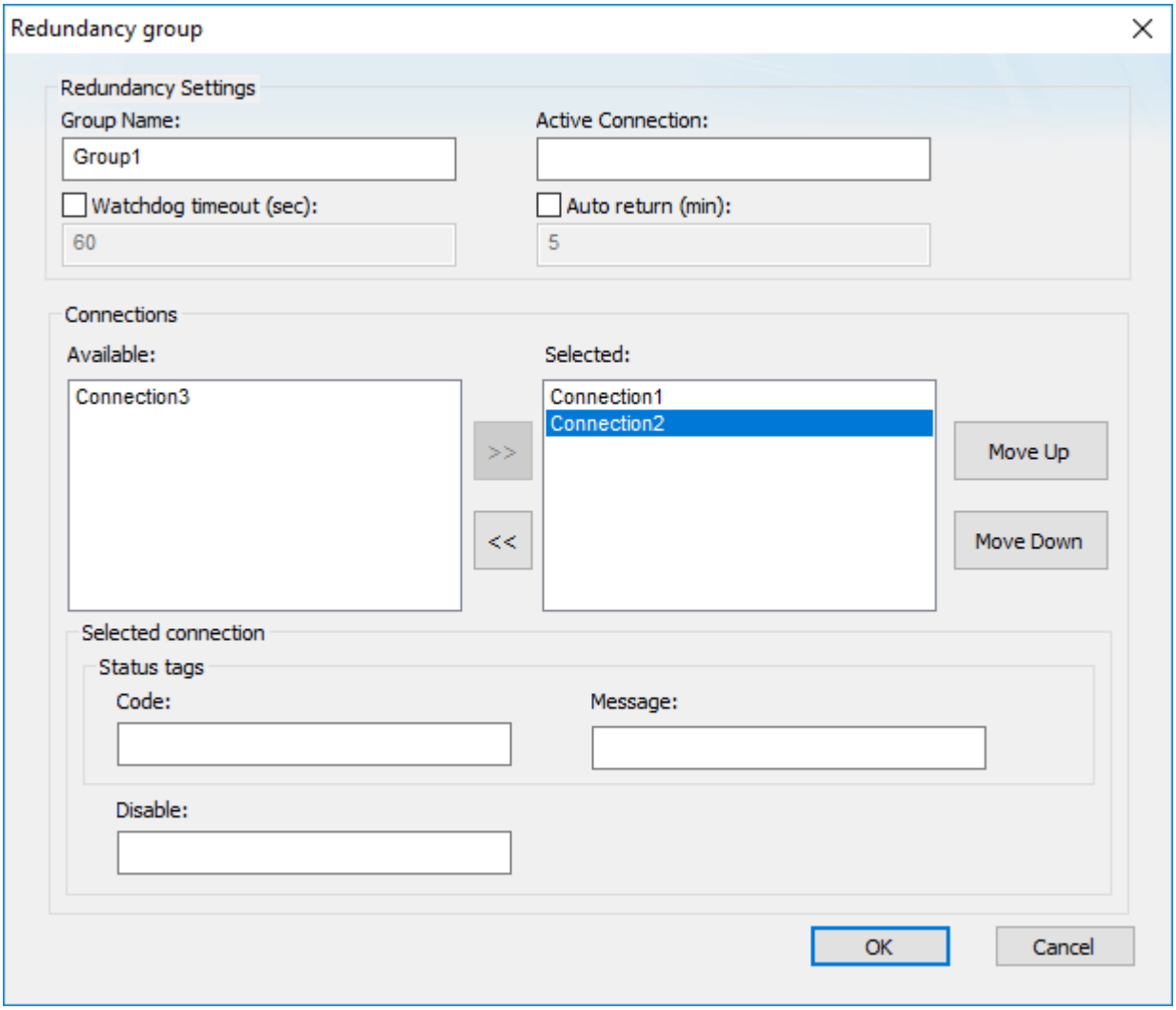
Keep in mind that if you configure several different worksheets to use the same redundancy group, they will all use the same connection at the same time during project run time. That might put too much load on a single connection and cause it to fail when it would not otherwise. To avoid that, create multiple groups — each with the connections arranged in a different order (e.g., ABC, CAB, BCA) — and then select a different group for each worksheet. This will provide rudimentary load balancing.

> **Note:** The OPC DA 2.05 client in EcoStruxure Machine SCADA Expert 2020 does not support redundancy groups at this time.

To create a redundancy group:

1. In the **Comm** tab of the Project Explorer, expand the folder for the type of OPC that you are using, either **OPC UA** or **OPC XML/DA**.
2. In that folder, right-click the **Redundancy Group** folder, and then on the shortcut menu, click **Insert**.

The *Redundancy Group* dialog box is displayed.




***Example of redundancy group settings***

3. In the **Group Name** box, type a unique name for the group.
   This name will be displayed in the list of available connections in your OPC client worksheet.
4. In the **Active Connection** box, type the name of a project tag of String type.
   The specified tag will receive the name of the connection that is currently being used during project run time. In other words, when the connection changes for any reason, the tag value changes to match. You can also change the tag value yourself, to control which connection is used. In both cases, the tag value is the full name of the connection (e.g., `Connection 1`). This setting is optional.
5. If you want a connection to automatically time out after a period of inactivity, rather than wait for it to actually fail (with BAD status), select the **Watchdog timeout** option.
   The default timeout is 60 seconds after the last change reported by the OPC Server. If the field device has less frequent changes, you need to either increase the timeout or set up a "heartbeat" on the field device to keep the connection active.

> **Note:** This watchdog can only watch the connection between the OPC Client (i.e., your project) and the OPC Server. It cannot watch the connection between the OPC Server and the field device. The OPC Server itself is responsible for monitoring that connection.

6. If you want the client to automatically return to the first connection in the group when it is possible to do so, rather than continue through all of the connections in the group, select the **Auto return** option.

   The default return time is 5 minutes, which means the client will try to return to the first connection every 5 minutes after the connection was lost. If the client cannot reestablish the first connection, it will continue with its current connection and try again later.

7. In the **Connections** area, in the **Available** list on the left, select a connection that you want to include in the group, and then click **>>**.

   There is no limit on the number of connections that you can include in a group.

   The connection is added to the **Selected** list on the right.

8. Repeat the previous step for each connection that you want to include in the group.

9. Select a connection in the **Selected** list on the right, and then do the following:

   a) Use the **Move Up** and **Move Down** buttons to move the selected connection to the desired position in the list.

   b) In the **Code** box, type the name of a project tag of Integer or String type.

   The specified tag will receive a continuously updated status code for the selected connection during project run time.

   c) In the **Message** box, type the name of a project tag of String type.

   The specified tag will receive a continuously updated status message for the selected connection during project run time.

   d) In the **Disable** box, type a tag name, expression, or literal value.

   When it evaluates as TRUE (i.e., non-zero) during project run time, the selected connection will be disabled and therefore skipped.

> **Note:** The **Code**, **Message**, and **Disable** settings are all optional and unique for each connection.

10. Repeat the previous step for each connection in the group.

11. Click **OK** to save your changes and close the dialog box.

The new group is saved in the appropriate **Redundancy Group** folder in the Project Explorer, and it becomes available for selection in the corresponding client worksheets.

During project run time, each connection in the redundancy group can have one of the following statuses:

| Code | Message |
|---|---|
| 1 | The connection to the OPC Server is established, and it is running in normal mode. |
| 1 | The connection to the OPC Server has been reestablished, and the watchdog has been reset. |
| -1 | [Any of the BAD messages that are described in List of read/write status codes and messages for OPC UA on page 599.] |
| -2 | The OPC Server stopped reporting changes from the field device (e.g., the PLC), and the watchdog timeout has elapsed. |
| -3 | Trying to establish connection to the OPC Server. |

## CREATE A NEW OPC XML/DA CLIENT WORKSHEET

Create and configure an OPC XML/DA Client worksheet to associate project tags with OPC server items.

Before you begin this task, you must have created at least one OPC XML/DA server connection that the client worksheet can use. For more information, see Create a new OPC XML/DA connection on page 605.

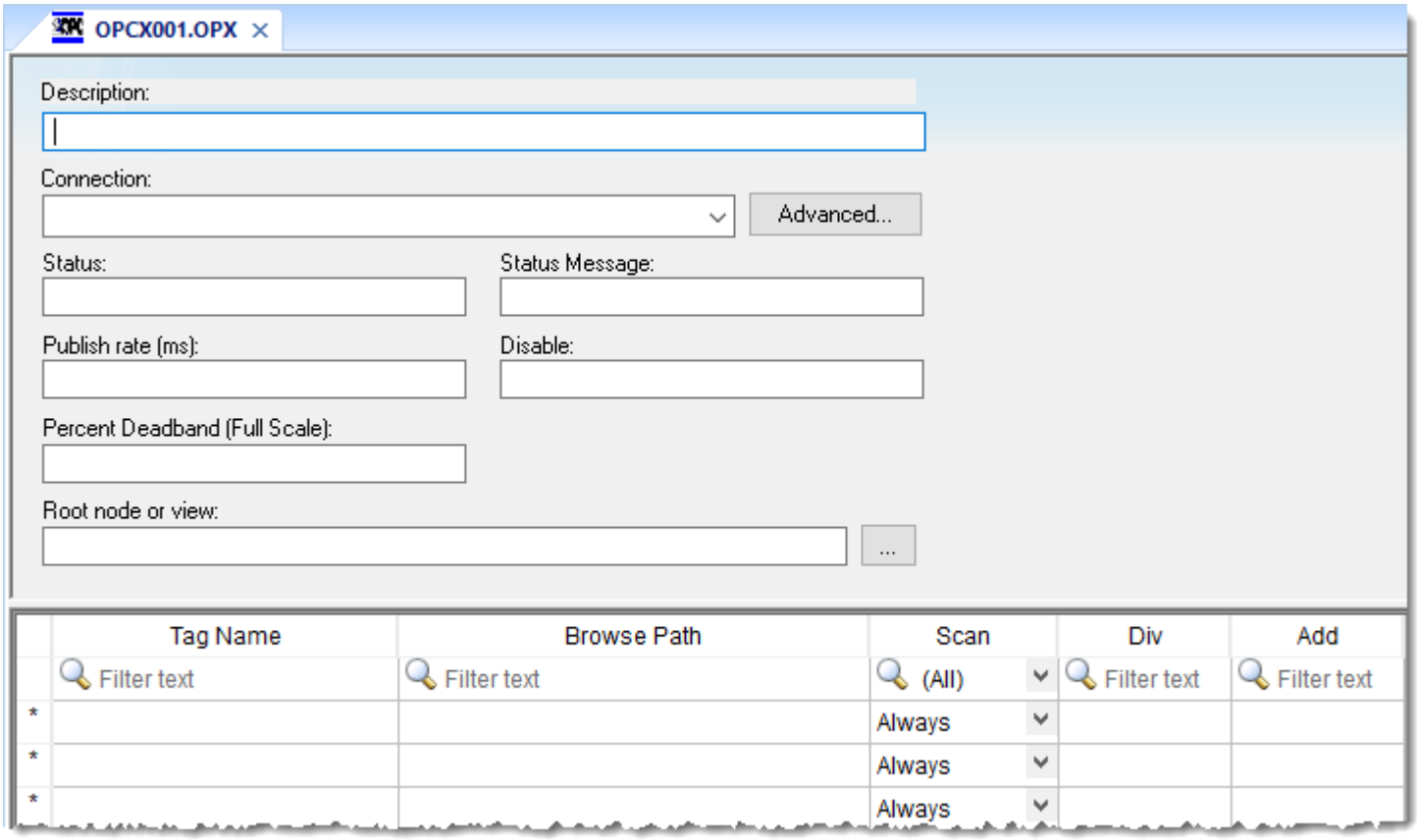
You should also be familiar with how to edit worksheets in the project development environment.

> **Tip:** You can create multiple worksheets and then configure them to have different settings, so you do not need to make any single worksheet unnecessarily large or complicated.

To configure a new OPC XML/DA Client worksheet:

1. Do one of the following:
   - On the **Insert** tab of the ribbon, in the **Communication** group, click **OPC Client**, and then select **OPC XML/DA Client** from the drop-down list; or
   - In the **Comm** tab of the Project Explorer, right-click the **OPC XML/DA** folder, and then on the shortcut menu, click **Insert**.

   A new OPC XML/DA Client worksheet is displayed.




2. In the **Description** box, type a description of the worksheet.
   This is for documentation purposes only and does not affect the execution of the worksheet.
3. In the **Connection** list, select the connection or redundancy group that you created earlier.
4. In the **Status** box, type the name of a project tag (Integer type) that will receive connection status codes during project run time.
5. In the **Status Message** box, type the name of a project tag (String type) that will receive the corresponding status messages.
6. In the **Publish Rate** box, type the frequency (in milliseconds) at which the client will request updates from the server.
7. In the **Disable** box, type a tag/expression.

While this tag/expression evaluates as TRUE (non-zero), the worksheet will not be executed and it will not use the specified connection. If any other OPC XML/DA Client worksheets share the same connection, however, the connection will remain open and active unless those worksheets are also disabled.

8. In the **Percent Deadband** box, type a value between `0.0` and `100.0`, as a percentage of the full engineering units scale. (The scale is calculated using the specified minimum and maximum values of the server item.)

   This tells the server to publish only changes in item values that are greater than the specified deadband. For example, if the minimum value is `1000`, the maximum value is `5000`, and the deadband is `0.1`, only changes greater than 4 (i.e., 0.1% of 4000) will be published by the server.

   If you do not specify a deadband, the default is `0.0`, which means the server will publish all changes in item values.

   **Percent Deadband** only applies to server items that have the Engineering Units Type attribute (**dwEUType**) set to Analog (**1**). For more information about this attribute, see either the OPC XML-DA specification or the documentation for your OPC server.

   If **Ensure cache synchronization** (in the advanced settings) is selected, **Percent Deadband** should not be used.

9. In the **Root node or view** box, specify a server node that will serve as the root for all browse paths in the worksheet body.

   Specifying a root node makes it easier to find items and improves run-time performance.

10. For each project tag that you want to associate with an OPC server item/node, complete a row in the worksheet body:

    a) In the **Tag Name** field, type the name of a project tag.

       You can type a string expression in this field (e.g., `{MyTagName}`), but if you do, make sure you have also configured the **Reload trigger** setting in the advanced settings.

    b) In the **Browse Path** field, do one of the following: for DA, type *<item name>*; or for XML, type *<item path>//<item name>*. To browse the server's list of items, right-click in this field, and then on the shortcut menu, click **Browse**.

    > **Note:** If you selected the **Enable bit notation** option in the advanced settings, you can select a specific bit of a server node value by appending the bit number to the item name (e.g., *<browse path>.<bit>*). Please note that this is supported only for unsigned variables and 32-bit signed variables; if you try to do it with 16-bit or 8-bit signed variables, the most significant bit (MSB) will not work properly.

    c) In the **Scan** field, select either **Always** to have the row continuously scanned (i.e., processed) during project run time or **Screen** to have the row scanned only when a project screen that uses the specified tag is open.

    d) In the **Div** field, type a number to be used for scaling during project run time. This is optional.

       When a value is read from the server, it is divided by this number. When a value is written to the server, it is multiplied by this number.

    e) In the **Add** field, type a number to be used for scaling during project run time. This is optional.

       When a value is read from the server, this number is added to it. When a value is written to the server, this number is subtracted from it.

    Both tag expansion and array distribution are enabled by default for all OPC XML/DA Client worksheets. You can disable tag expansion for this worksheet by clearing the **Enable Tag Expansion** option in the advanced settings. You cannot disable array distribution.

    If your project was created with a previous version of this software and then upgraded to the latest version, the worksheet body might include an additional **Array Index** field. That field has been deprecated in favor of array distribution.

11. When you are done, save and close the worksheet.
    The worksheet is saved in the **OPC XML/DA** folder in the Project Explorer.

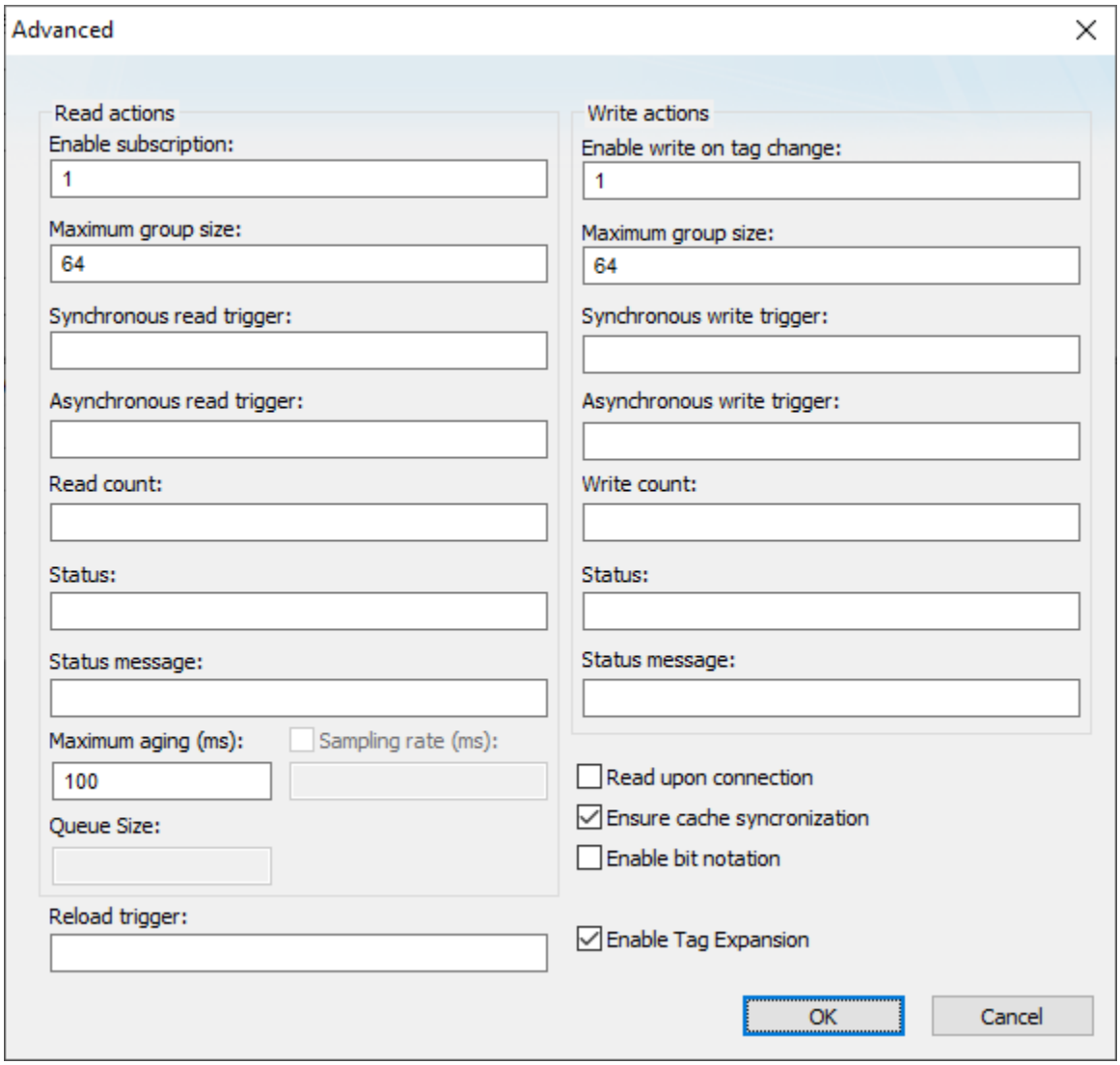
In order for your OPC XML/DA Client worksheet(s) to be scanned during project run time, the OPC XML/DA Client Runtime task in your project must be started. As such, the first time you configure and save a worksheet, the task's startup mode is set to **Automatic**. This is done for your convenience, and it means that when you run your project, the task will be started and the worksheet(s) will be scanned. You can set the

task's startup mode back to **Manual**, however, if you want more control over how your project runs. For more information, see Runtime Tasks on page 136.

The worksheet is continuously scanned during project run time, so that the configured project tags and server items are updated as needed. Also, the project tags that you specified for the **Status** and **Status Message** settings in the worksheet header will receive appropriate values.

> **Tip:** As long as the client is communicating normally with the server, the read/write status codes and messages (as configured in the advanced settings) will provide more information about each operation.

In certain situations, if the worksheet does not behave as expected during project run time — and especially if you see OPC communication errors in the runtime log — you might need to adjust the worksheet's advanced settings: in the worksheet header, click **Advanced**, and then in the *Advanced* dialog box, review and configure the settings.




***Advanced settings for the OPC XML/DA Client worksheet***

**Read actions**

Advanced settings that control how values are read from the OPC UA server.

#### Enable subscription

While this value is TRUE (non-zero), the client will subscribe to the server nodes so that it can receive notifications when the node values change.

This setting has a default value of 1, which indicates it is always enabled. If it is disabled, or if you type a tag/expression that will cause it to be disabled during project run time, you might need to use read triggers instead (see below).

#### Maximum group size

The maximum number of server nodes that may be read in a single read operation.

For example, if you have 1000 items/rows configured in the worksheet and **Maximum group size** is set to 100, 10 read operations will be performed when a read is triggered.

#### Synchronous read trigger

When the value of this tag/expression changes, the client reads all of the node values from the server. The read operation is performed synchronously, which means all other client operations are blocked until the read operation is finished. When the read operation is finished, the client increments the tag configured in **Read count**.

#### Asynchronous read trigger

The same as **Synchronous read trigger** except that the read operation is performed asynchronously, which means that other client operations may continue while the read operation is being performed.

#### Read count

The name of a project tag (Integer type) that will receive a count of the number of read operations performed since the project was run.

#### Status

The name of a project tag (Integer type) that will receive a status code for the last read operation performed by a trigger:

| Status Code | Description |
|---|---|
| 0 | Bad |
| 1 | Good |

#### Status message

The name of a project tag (String type) that will receive a status message for the last read operation performed by a trigger. For more information, see List of read/write status messages for OPC XML/DA on page 615.

#### Maximum aging

The maximum age (in milliseconds) of values that will be accepted from the server's cache. If a value is older than this, the server will be forced to get the latest value from the field device.

#### Sampling rate

When this option is selected, you can change the rate (in milliseconds) at which the OPC server reads from field devices. By default, the sampling rate is half the publishing rate.

#### Queue size

### Write actions

Advanced settings that control how values are written to the OPC UA server.

#### Enable write on tag change

While this value is TRUE (non-zero), an asynchronous write operation will be performed automatically whenever the value of a project tag changes. All tag changes that occured since

the last scan of the worksheet will be written in a single write operation, or in multiple write operations if the number of tag changes exceeds the maximum group size.

This setting has a default value of 1, which indicates it is always enabled. If it is disabled, or if you type a tag/expression that will cause it to be disabled during project run time, you might need to use write triggers instead (see below).

#### Maximum group size

The maximum number of tag changes that may be written in a single write operation.

For example, if you have 1000 items/rows configured in the worksheet and **Maximum group size** is set to 100, 10 write operations will be performed during each scan of the worksheet.

#### Synchronous write trigger

When the value of this tag/expression changes, the client writes all of the tag values to the server. The write operation is performed synchronously, which means all other client operations are blocked until the write operation is finished. When the write operation is finished, the client increments the tag configured in **Write count**.

#### Asynchronous write trigger

The same as **Synchronous write trigger** except that the write operation is performed asynchronously, which means that other client operations may continue while the write operation is being performed.

#### Write count

The name of a project tag (Integer type) that will receive a count of the number of write operations performed since the project was run.

#### Status

The name of a project tag (Integer type) that will receive a status code for the last write operation performed by a trigger:

| Status Code | Description |
|---|---|
| 0 | Bad |
| 1 | Good |

#### Status message

The name of a project tag (String type) that will receive a status message for the last write operation performed by a trigger. For more information, see List of read/write status messages for OPC XML/DA on page 615.

### Other

Other settings that control how this worksheet communicates with the OPC UA server.

#### Reload trigger

The name of a project tag (Boolean, Integer, or Real type) that can be used as a trigger. When the value of this tag changes, the worksheet is reloaded. String expressions configured in the body of the worksheet are reevaluated only when the worksheet is reloaded. After the worksheet is reloaded, the value of this tag will be reset to 0.

#### Read upon connection

When this option is selected, the worksheet will read the current values of all configured items immediately after the client connects to the server, regardless of any other settings that control read actions.

#### Ensure cache synchronization

When this option is selected, the client will wait after each write operation for confirmation from the server that the node values actually changed. If the client does not receive confirmation, it will restore the previous tag values.

**Enable bit notation**

When this option is selected, bit notation is allowed in the **Browse Path** column of the worksheet body.

**Enable Tag Expansion**

When this option is selected, tag expansion is enabled for this worksheet. You can disable it for this worksheet without affecting any other OPC client worksheets, and you may choose to do so if you want more control over how OPC server items are associated with your project tags

> **Note:** This software does not normally use the Triggering Mode that is defined the OPC standard. Instead, it allows any change in any tag/expression to be used as a trigger. If you want to use Triggering Mode, configure one worksheet to read the trigger values, and then configure another worksheet that specifies the read values as triggers.

## LIST OF READ/WRITE STATUS MESSAGES FOR OPC XML/DA

This is a list of the possible status messages that might be generated by read/write operations in OPC XML/DA.

| Status Message | Hexadecimal | Description |
|---|---|---|
| S_OK | 0x00000000 | Success |
| S_FALSE | 0x00000001 | Failure |
| OPC_E_INVALIDHANDLE | 0xC0040001 | The value of the handle is invalid. |
| OPC_E_BADTYPE | 0xC0040004 | The server cannot convert the data between the specified format and/or requested data type and the canonical data type. |
| OPC_E_PUBLIC | 0xC0040005 | The requested operation cannot be done on a public group. |
| OPC_E_BADRIGHTS | 0xC0040006 | The item's access rights do not allow the operation. |
| OPC_E_UNKNOWNITEMID | 0xC0040007 | The item ID is not defined in the server address space or no longer exists in the server address space. |
| OPC_E_INVALIDITEMID | 0xC0040008 | The item ID does not conform to the server's syntax. |
| OPC_E_INVALIDFILTER | 0xC0040009 | The filter string was not valid. |
| OPC_E_UNKNOWNPATH | 0xC004000A | The item's access path is not known to the server. |
| OPC_E_RANGE | 0xC004000B | The value was out of range. |
| OPC_E_DUPLICATENAME | 0xC004000C | Duplicate name not allowed. |
| OPC_S_UNSUPPORTEDRATE | 0x0004000D | The server does not support the requested data rate but will use the closest available rate. |
| OPC_S_CLAMP | 0x0004000E | A value passed to write was accepted but the output was clamped. |
| OPC_S_INUSE | 0x0004000F | The operation cannot be performed because the object is bering referenced. |
| OPC_E_INVALIDCONFIGFILE | 0xC0040010 | The server's configuration file is an invalid format. |
| OPC_E_NOTFOUND | 0xC0040011 | The requested object was not found. |
| OPC_E_INVALID_PID | 0xC0040203 | The specified property ID is not valid for the item. |

| Status Message | Hexadecimal | Description |
|---|---|---|
| OPC_E_READONLY | 0xC0048006 | The item is read only and cannot be written to. |
| OPC_E_INVALIDCONTINUATIONPOINT | 0xC0040403 | The continuation point is not valid. |
| OPC_E_WRITEONLY | 0xC0048007 | The item is write only and cannot be read or returned in a Write response. |
| E_NOTIMPL | 0x80004001 | Not implemented. |
| E_NOINTERFACE | 0x80004002 | No such interface supported. |
| E_ABORT | 0x80004004 | Operation aborted. |
| E_FAIL | 0x80004005 | Unspecified error. |
| E_OUTOFMEMORY | 0x8007000E | Out of memory. |
| E_INVALIDARG | 0x80070057 | One or more arguments are invalid. |
| CONNECT_E_NOCONNECTION | 0x80040200 | Advise cannot find Connection point or Unable to impersonate DCOM client. |
| CONNECT_E_ADVISELIMIT | 0x80040201 | Unable to obtain server's security context. |

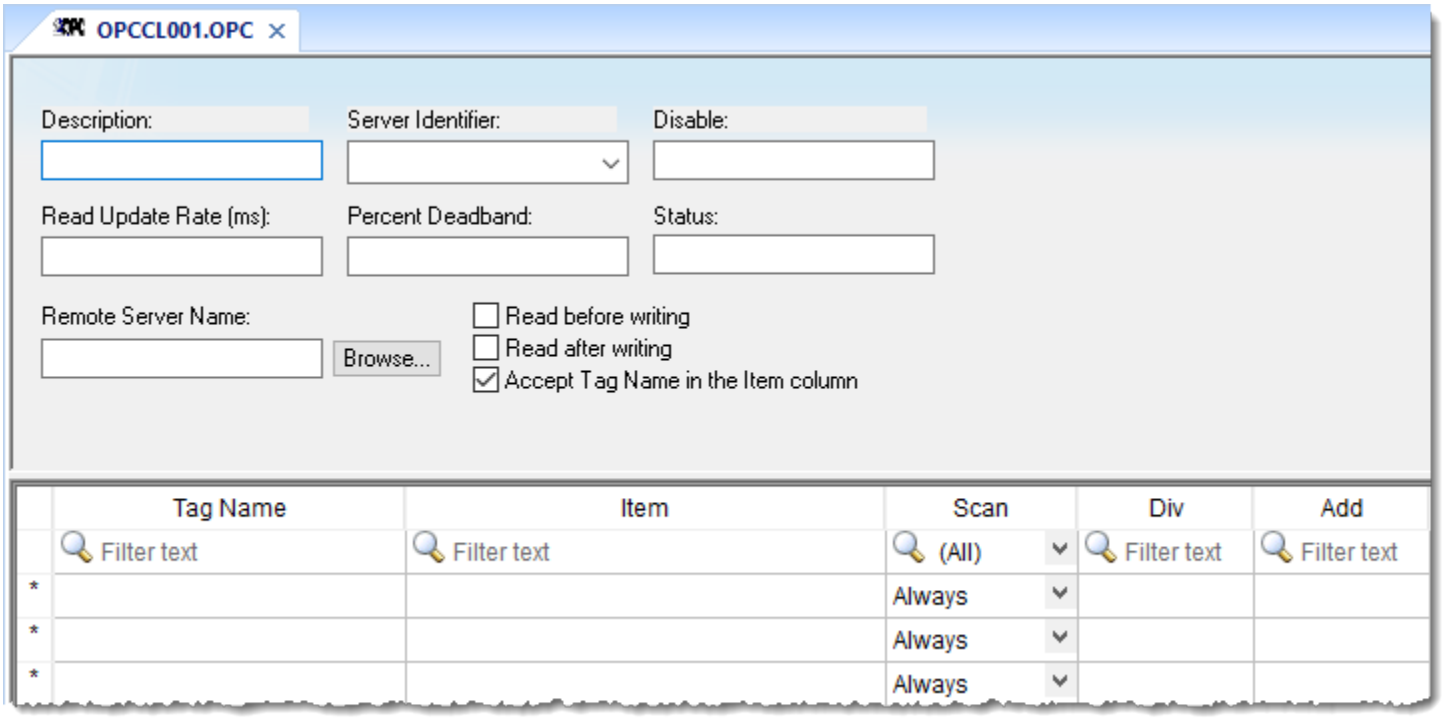
### *OPC DA 2.05 Client*

Use the OPC DA 2.05 Client worksheet and runtime task to establish communication between your project and a data exchange server that supports the OPC DA (a.k.a. OPC Classic) interoperability standard.

> **Note:** OPC DA 2.05 Client is a legacy feature that is included in this release of EcoStruxure Machine SCADA Expert 2020 only to maintain backward compatibility with existing projects. OPC DA depends on Microsoft's proprietary DCOM and OLE technologies, which have been superseded by cross-platform technologies like XML and SOAP for web services. If you are creating a new project, we strongly recommend you use either OPC UA Client or OPC XML/DA Client, depending on the configuration of your data exchange server.

Before you begin this task, make sure the OPC DA server software is properly installed and configured on the computer to which you want to connect.

To configure a new connection, insert a new OPC DA 2.05 Client worksheet on the **Comm** tab of the Project Explorer.




***Sample OPC DA 2.05 Client worksheet***

Configure the following settings for the worksheet:

- **Description** text box: Type a description of the worksheet for documentation purposes only. (The OPC DA 2.05 Client Runtime task ignores this information.)
- **Server Identifier**: Type the name of the server you want to connect. If the server is already installed on the computer, you can select the server name from the list.
- **Disable**: Type a tag/expression. While it evaluates as TRUE (non-zero), the subscription of the items configured in the OPC DA 2.05 Client worksheet will be disabled, which means the server will no longer send messages to update the values of those items. However, disabling the worksheet will not disconnect the client from the server; the client will still be able to write values to the server.
- **Read Update Rate**: Specify how often the server should update this group (in milliseconds). Specify `0` to indicate the server should use the fastest practical rate.
- **Percent Deadband** (valid for analog items only): Specify how much percent change in an item value should cause a notification by the server.
- **Status**: Type the name of a tag to receive the status of the connection. Good status is 1.
- **Remote Server Name**: Node name or IP address of server on node network.
- **Read before writing** checkbox: Check this option to force your project to read the original values of items on the OPC server just before writing new values to the server. The project does this by first buffering the new values to be written and then reading the original values from the server. Only after the project is synchronized with the server are the new values written from the buffer to the server.
- **Read after writing** checkbox: Check this option to force your project to read back the new values of items on the OPC server just after the project has written those values.

> **Note:**
>
> The **Read before writing** and **Read after writing** options are offered because the OPC Client/Server specification says that the value of an item on the client — in this case, your project — should

not change unless the server sends the change. That way, the client always stays in sync with the server.

Your project, however, may be designed to change those values according to runtime processes or user input. Therefore, the best way to change the values while staying in sync with the server is to make it seem like the changes originate on the server. With both options enabled, the following sequence of events happens on every scan of the OPC worksheet:

1. The new values on the client are buffered.
2. The original values on the server are read to the client — that is, the client is synchronized with the server.
3. The new values are written from the buffer to the server.
4. The new values on the server are read to the client — that is, the client is again synchronized with the server.

At the end of each scan, the values reflect what's happening in your project even though, technically speaking, the project is merely staying in sync with the server.

Both options should be enabled in most projects. In some projects, however, this may cause items to bounce between the original values and the new values. If this is a problem, try moving those items to another OPC worksheet where the **Read before writing** and **Read after writing** options are disabled.

- **Accept Tag Name in the Item column** checkbox: When this option is checked, the text configured between curly brackets in the Item field is resolved as a Tag Name (string tag). In this case, the value of this tag is used as the name of the item from the server, allowing the user to point to different item names during runtime, by changing the value of the tag(s) configured in the worksheet (Item column).

  When the **Accept Tag Name in the Item column** option is unchecked, all characters configured in the Item column are considered part of the Item name (including the curly brackets).

- **Tag Name**: Type the names of tags linked to the server items.

  > **Note:** Both tag expansion and array distribution are enabled for all OPC DA 2.05 Client worksheets.

- **Item**: Enter the name of the server's items. After selecting a server, you can select items from that server using the OPC Browser. Right-click in the **Item** field and select the **OPC Browser** option.

  > **Tip:** You can configure a tag name between curly brackets (e.g., `{TagName}` ) in this field, allowing the user to change the item names dynamically, during runtime.

- **Scan** field: Specify the condition under which the tag value is read from the remote device or server and then updated in the project database, using one of the following options:

  - **Always** means the tag is read and updated during every scan of the communication worksheet, regardless of whether the tag is used in any other project screens, scripts, or worksheets.

    This option is recommended for tags that must be continuously monitored in the background, such as tags that trigger alarms, tags used in recipes, tags that are recorded in the historical database, and so on.

  - **Screen** means the tag is read and updated only if it is being used in at least one open project screen, either locally or on another client station.

    This option is recommended for tags that are used in screen objects, because the project may not need to update tags that are not being visualized anywhere. Selecting this option can improve project performance.

  - **Auto** means the project will automatically choose either **Always** or **Screen**, depending on where the tag is used in your project. If the tag is only used in a screen object on a project screen, then the scan will default to **Screen**. But if the tag is configured in any other interface (e.g., Script, Math, Alarm, Trend, Recipe, Report, Scheduler), then the scan will switch to **Always** and remain there until the project is stopped.

  If you are not sure of which option to select, select **Always**. This will guarantee the tag is read and updated.

- **Div** field: Specify the division constant when scale adjustment is required. This value is a division factor in a read operation and a multiplication factor in a write operation.
- **Add** field: Specify the addition constant when scale adjustment is required. This value is a addition factor in a read operation and a subtraction factor in a write operation.

> **Note:** The OPC DA specification supports custom item qualities using the high byte of the two-byte quality field. However, such qualities are often vendor-specific or even hand-coded, so it is not possible for EcoStruxure Machine SCADA Expert 2020 to interpret them. All item qualities other than GOOD (192) will be ignored.

To run the OPC DA 2.05 Client Runtime task, you can choose to run it automatically on start up, or run the task manually by clicking **Tasks** (either local or remote) on the Home tab of the ribbon. After running this program, a small icon displays in your system tray.

To close the OPC DA 2.05 Client Runtime task, right-click the icon in the system tray, and click **Exit**.

### Troubleshooting

When you are using an OPC DA 2.05 Client worksheet and have problems establishing communication, you should first verify the messages in the LogWin.

If you are running the project on a Windows Embedded target system, there are two ways to check the log:

1. Remote LogWin
2. Local Log

For information about using these logs, see About the LogWin tool on page 712.

If you find error messages in the log, look them up in this manual/help system, and follow the documented steps for solving the problems. (Use `Ctrl+F` to find them in the manual; use the Index to find them in the context sensitive help system.)

If you feel that you need to contact your distributor for technical support, make sure that you provide them with the following information:

1. Log file
2. Software vendor and product name of the OPC Server/Client that you are using
3. If possible, a copy or an evaluation version of the OPC Server for testing purposes
4. The contact information for your OPC Server/Client technical support

Three possible errors and their resolutions are listed below...

**Security**

Error Code: **`0x80070005 or -2147024891`**

Reason for error: When the client tries to connect to the server, the DCOM layer usually requires authentication. The computer that is running the server needs to recognize the user logged on to the client computer, and such a user needs to have privileges to access the server.

Solution: The first step is to create a single user on both computers that has Administrator privileges and the same password. Log on with this user to both ends, and then try to establish the connection.

**Name Resolution**

Error: **`Couldn't create connection with advise sink, error: -2147022986 (0x80070776)`**

Reason for error: There is a problem resolving the computer name.

Solution: This problem can be solved by specifying the IP address of the server instead of specifying the computer name.

**Proxy for Windows CE**

Error: **`OPCServer: IIndCP::Advise - Could not query callback interface: 0x80040155`**

Reason for error: Your Windows Embedded device is missing the **`OPCCOMN_PS.dll`**.

Solution: You should download the .dll file to the device and register it. The file should be available in your program folder at: `Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Redist\WinCE 5.0\<device processor>\OPCComn_ps.dll`

If you do not find the **`.dll`** in the folder for your processor, contact your software distributor.

### *Tag Expansion for OPC Clients*

Tag expansion is the method by which a group of items on the OPC server are automatically associated with similarly named class members in your project.

> **Note:** This feature is not supported in projects that are configured to run on the Embedded target platform. For more information, see About target platforms, product types, and target systems on page 100.

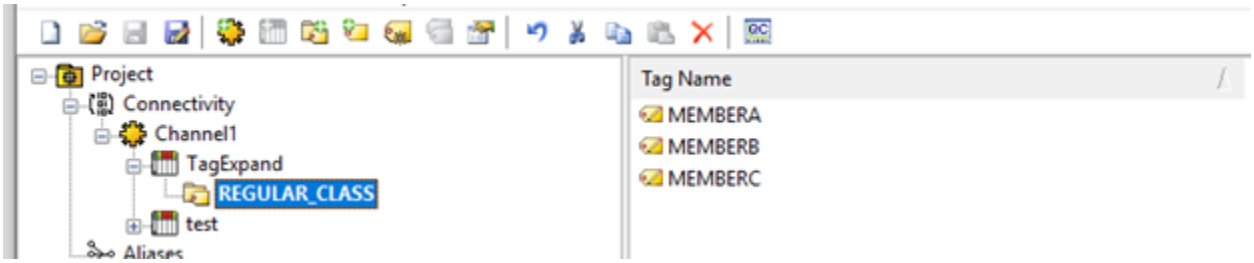
An OPC server acts as a container for OPC groups, and in turn, each OPC group acts as a container for OPC items. The OPC group is roughly equivalent to a class tag in your project, and the OPC items are roughly equivalent to class members.

If you create a class tag in your project that has class members with the same names as the OPC items, it is possible to automatically associate those class members with the OPC items. You need to configure only a single row in the OPC client worksheet that associates the class tag with the OPC group, and then tag expansion will manage the associations between the individual class members and OPC items during project run time.

You can disable tag expansion for a single OPC client worksheet without affecting any other worksheets, and you may choose to do so if you want more control over how your project tags are associated with OPC items. To disable tag expansion, clear the **Enable Tag Expansion** option in the worksheet's advanced settings.

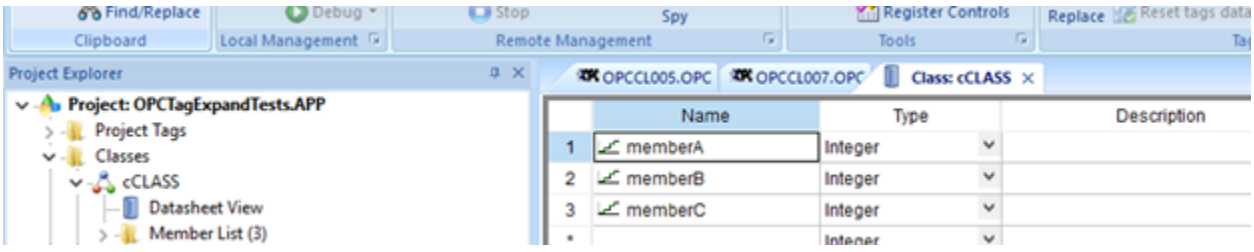
The following illustrations provide an example of how tag expansion works. (This is only an example; you do not need to use these exact names nor follow these exact steps.)

First, on the OPC server, you can create a group of three items.



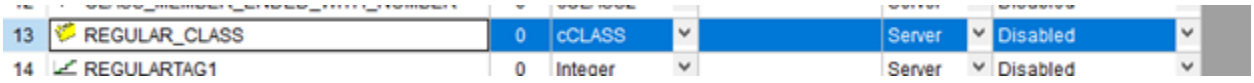

***Creating a group of three items on the OPC server***

Then, in your project, you can create a class with similarly named members.




***Creating a class with three members in your project***

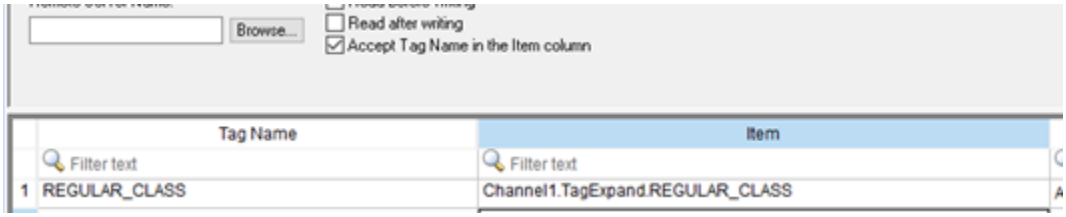
The class serves only as a sort of template for project tags, so you need to create a new project tag and then set its type to the appropriate class, thereby making it a class tag.




*Creating a project tag of the appropriate class*

> **Note:** Multi-dimensional or "nested" classes are not supported.

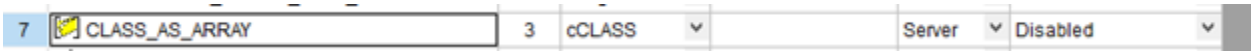
Finally, when you create your OPC client worksheet, you can configure a single row of the worksheet to associate the class tag with the OPC group.




*Associating the class tag with the OPC group*

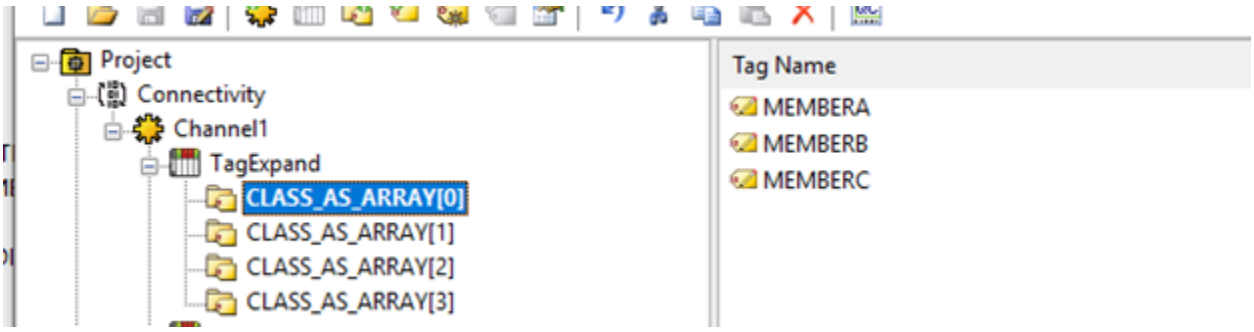
You do not need to configure any additional rows, because the one-to-one associations between the class members and the OPC items will be managed automatically during project run time, thanks to tag expansion.

Taking this a step further, you can create an array of classes in your project by increasing the Array property of the class tag. In this example, increasing the Array property to 3 creates an array of four classes in positions 0 through 3.



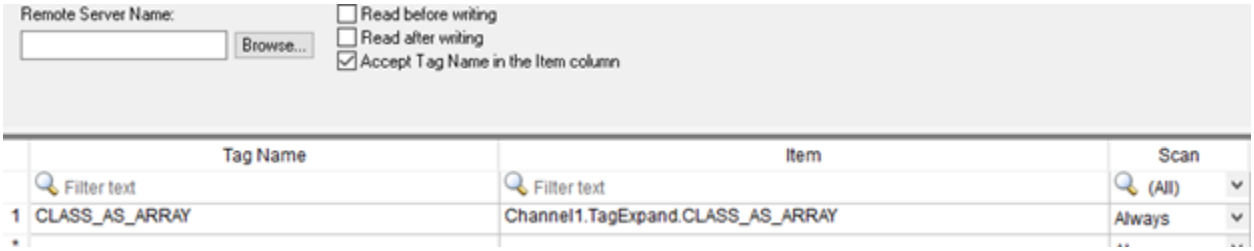

*Creating an array of classes by increasing the size of the tag*

It is technically not possible to create an array of OPC groups on the server, but it is possible to create a series of OPC groups that are sequentially numbered like array elements, using the standard syntax for array indices (`[n]`).




*Creating sequentially numbered groups on the OPC server*

Tag expansion will detect the numbering of the OPC groups, so that when you create your OPC client worksheet, you can configure a single row of the worksheet to associate the array of classes with the series of OPC groups.




***Associating the OPC groups with the array of classes***

Again, you do not need to configure any additional rows, because all twelve associations (i.e., four classes/groups, each with three members/items) will be managed automatically during project run time, thanks to tag expansion.

> **Note:** Make sure the size of the array in your project is equal the number of OPC groups on the server. If it is not equal, you might see unexpected behavior during project run time as tag expansion tries to make invalid associations between class members and OPC items.

> **Note:** The **Check node type** option, in the advanced settings for OPC UA connections, can interfere with tag expansion. Therefore, if you are using tag expansion in an OPC UA Client worksheet, you might need to clear the option for the connection used by that worksheet.

## *Array Distribution for OPC Clients*

Array distribution is the method by which the elements of an array item on the OPC server are automatically associated with various types of project tags.

> **Note:** This feature is not supported in projects that are configured to run on the Embedded target platform. For more information, see About target platforms, product types, and target systems on page 100.

The OPC clients in this software are capable of reading array items on the OPC server, but they cannot read only certain elements of those array items. Each array item is read in its entirety during every scan of an OPC client worksheet. In other words, it is not possible to associate a single project tag with a single element of an array item.

Instead, when an array item on the OPC server is read, the elements of that array item are automatically distributed to a group of project tags. You need to configure only a single row in the OPC client worksheet that associates the array item with the "first" tag in the group. What qualifies as the "first" tag, however, depends on how you have created and organized your project tags.

The following scenarios describe how array distribution works with different types of project tags.

### Elements of an Array Item → Elements of an Array Tag

In this scenario, you can create an array tag in your project by increasing the Array property of a regular project tag, and then you can associate the array tag with the array item on the OPC server. The elements of the array item are distributed one-for-one to the elements of the array tag. If you specify an array index (e.g., `MyArrayTag[3]`), the distribution will begin at that position. If you do not specify an array index, the distribution will begin at position 0 by default.

#### Examples of array tags

| When Tag Name is... | ...the elements of the OPC array item are distributed to: |
|---|---|
| `MyArrayTag` | **`MyArrayTag[0]`**, **`MyArrayTag[1]`**, **`MyArrayTag[2]`** |

| When Tag Name is... | ...the elements of the OPC array item are distributed to: |
|---|---|
| MyArrayTag[3] | **MyArrayTag[3]**, **MyArrayTag[4]**, **MyArrayTag[5]** |

Make sure the array tag in your project is large enough to accomodate the elements of the array item on the OPC server. If it is not large enough, elements will be lost.

### Elements of an Array Item → Sequentially Numbered Project Tags

In this scenario, you can create a series of sequentially numbered project tags, and then you can associate just the first tag in the series with the array item on the OPC server. The elements of the array item are distributed one-for-one to the project tags in the series.

#### Examples of sequentially numbered project tags

| When Tag Name is... | ...the elements of the OPC array item are distributed to: |
|---|---|
| MyTag1 | **MyTag1**, **MyTag2**, **MyTag3** |
| MyTag3 | **MyTag3**, **MyTag4**, **MyTag5** |
| MyTag001 | **MyTag001**, **MyTag002**, **MyTag003** |

Make sure you create enough project tags to accomodate the elements of the array item on the OPC server. If there are not enough tags, elements will be lost.

### Elements of an Array Item → Sequentially Numbered Members of a Class Tag

This scenario is similar to the previous scenario: you can create a class tag that consists of sequentially numbered members, and then you can associate just the member in the series with the array item on the OPC server. The elements of the array item are distributed one-for-one to the members in the series.

#### Examples of sequentially numbered class members

| When Tag Name is... | ...the elements of the OPC array item are distributed to: |
|---|---|
| MyClass.Member1 | **MyClass.Member1**, **MyClass.Member2**, **MyClass.Member3** |
| MyClass.Member3 | **MyClass.Member3**, **MyClass.Member4**, **MyClass.Member5** |
| MyClass.Member001 | **MyClass.Member001**, **MyClass.Member002**, **MyClass.Member003** |

Make sure you create enough members to accomodate the elements of the array item on the OPC server. If there are not enough members, elements will be lost.

### Elements of an Array Item → An Array of Class Tags

This scenario is a combination of the previous scenarios in that you can create an array of class tags, and then the elements of the array item on the OPC server will be distributed amongst all of the members depending on which one you associate with the array item.

#### Examples of members in an array of class tags

| When Tag Name is... | ...the elements of the OPC array item are distributed to: |
|---|---|
| MyArrayClass.Member | **MyArrayClass[0].Member**, **MyArrayClass[1].Member**, **MyArrayClass[2].Member** |
| MyArrayClass[3].Member | **MyArrayClass[3].Member**, **MyArrayClass[4].Member**, **MyArrayClass[5].Member** |
| MyArrayClass[3].Member1 | **MyArrayClass[3].Member1**, **MyArrayClass[3].Member2**, **MyArrayClass[3].Member3** |
| MyArrayClass[3].Member3 | **MyArrayClass[3].Member3**, **MyArrayClass[3].Member4**, **MyArrayClass[3].Member5** |

| When Tag Name is... | ...the elements of the OPC array item are distributed to: |
|---|---|
| MyArrayClass[3].Member001 | **MyArrayClass[3].Member001**, **MyArrayClass[3].Member002**, **MyArrayClass[3].Member003** |

## *OPC UA Server*

OPC Unified Architecture (OPC UA) is an interoperability standard for exchanging real-time data between clients and servers. This software includes an OPC UA Server module that you can use to make your project tags available to OPC UA clients on the network.

The OPC UA Server module is turned off by default. In order to use the server, you need to configure its communication settings, select the project tags that you want to make available to clients, and then start the OPC UA Server Runtime task. You might also need to manage program certificates so that the server and its clients can communicate securely with each other. This section describes how to do all of these things.

For more information about OPC UA, go to: opcfoundation.org/about/opc-technologies/opc-ua/

### CONFIGURE THE COMMUNICATION SETTINGS FOR OPC UA SERVER

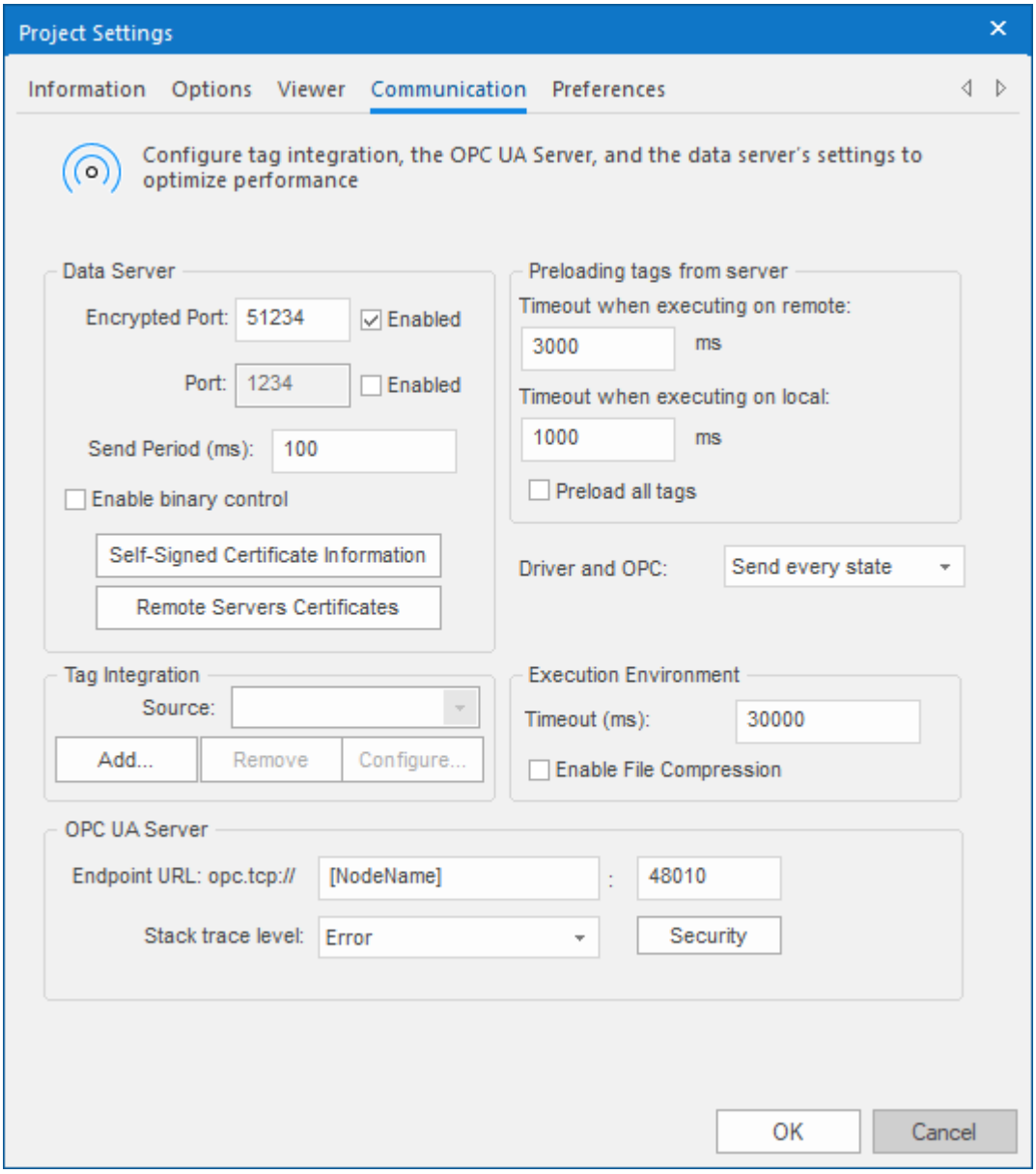
The communication settings for the OPC UA Server module are located in the *Project Settings* dialog box. These settings determine how the server will appear on the network, as well as how OPC UA clients may communicate with it. You need to review and configure these settings before project run time.

Before you begin this task, you should be familiar with the OPC United Architecture (OPC UA) specification and how OPC UA servers and clients communicate with each other. In particular, you should understand how servers and clients use certificates to identify each other and communicate securely.

To configure the communication settings for the OPC UA Server module:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.

The Communication tab of the *Project Settings* dialog box is displayed.




***Project Settings > Communication***

2. In the **Endpoint URL** boxes, type the node name and port number for the OPC UA server endpoint.

   The default node name is **`[NodeName]`**. This is a special string that automatically gets the host name of the actual computer or device that hosts the project, so that you do not need to change the node name each time you download your project to another computer.

   The default port number is 48010, which is the standard port number for OPC UA communication.

   In most cases, you should keep the default node name and port number. If you think you need to change either of them, ask your network administrator.

3. In the **Identity** area, select the identity types that will be allowed to log on to the OPC UA server.

   You must select at least one of the available options. If you do not, the OPC UA Server Runtime task will not be able to start during project run time. The options are not exclusive; you can select more than one option.

| Option | Description |
|---|---|
| **Enable anonymous login** | Allow clients to log on to the server without entering a username or password. |
| **Enable Username/Password** | Require clients to enter a username and password that a match a user in your project's security system. This option is available only if the security system has been enabled. |

4. In the **Security Policies** area, select the policies that the OPC UA server may use to communicate with OPC UA clients.

   In order for a server and client to communicate with each other, they must have at least one security policy in common.

   First, select the encryption types. You must select at least one of the available options. If you do not, the OPC UA Server Runtime task will not be able to start during project run time. The options are not exclusive; you can select more than one option.

| Option | Description |
|---|---|
| **None** | Communication between server and client does not need to be encrypted. |
| **Basic256** | The server will use and recognize 256-bit AES encryption. (This option is selected by default in new projects.) |
| **Basic128Rsa15** | The server will use and recognize 128-bit AES encryption. |

   Then, for each encryption type you have selected, select its messaging mode.

| Option | Description |
|---|---|
| **Sign** | Messages between server and client must be signed. |
| **Sign and Encrypt** | Messages between server and client must be signed and encrypted. |
| **Sign, Sign and Encrypt** | Messages between server and client may be either signed, or signed and encrypted. (This option is selected by default in new projects.) |

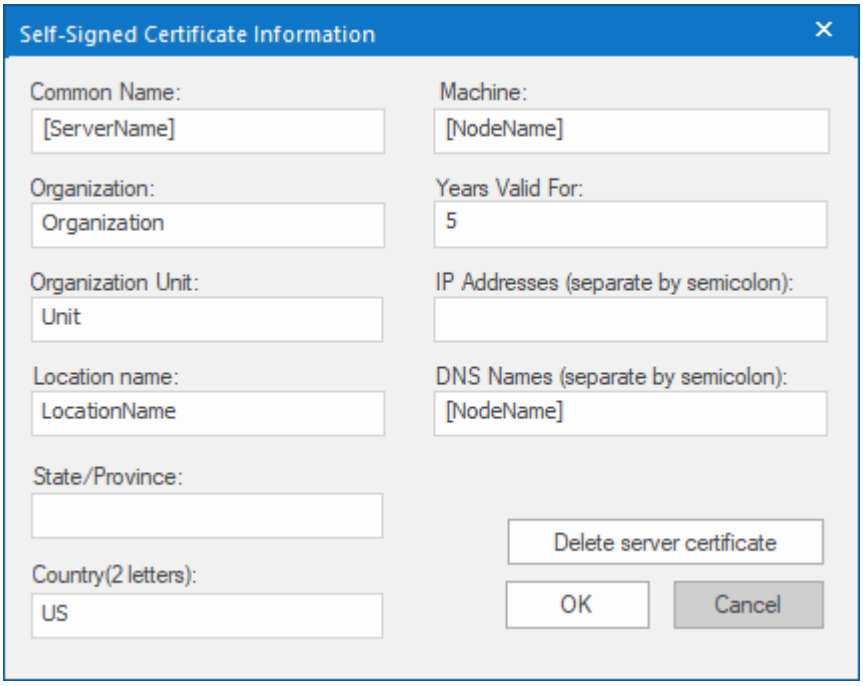
   For more information about these options and how the OPC UA clients might be configured, ask your network administrator.

5. In the **Certificates** area, review the information that will be included in the OPC UA server's self-signed certificate.

   This is the certificate that the server will present to clients in order to identify itself. It is signed by the software itself, as opposed to being signed by a Certificate Authority (CA).

   a) Click **Self-Signed Certificate Information**.

The *OPC UA Server Self-Signed Certificate Information* dialog box is displayed.




***OPC UA Server Self-Signed Certificate Information***

b) In the **Common Name** box, type the common name of the OPC UA server itself.

This is the name that is broadcast to the discovery server and other OPC UA clients on the network. The default common name is **`[ServerName]`**. This is a special string that includes **`[NodeName]`** (see above), which automatically gets the host name of the actual computer or device that hosts the project. The value of **`[ServerName]`** is:

```
StudioOpcUaServer@[NodeName]
```

c) In the **Machine** box, type the name of the computer or device that will host the project runtime server.

The default machine name is **`[NodeName]`** (see above).

d) In the **Organization**, **Organization Unit**, **Location Name**, **State/Province**, and **Country** boxes, type the appropriate information for your project.

e) In the **Years Valid For** box, type the number of years for which the certificate will be valid, starting from the date it is issued.

The default number of years is 5. When a certificate expires, you must delete it and then issue a new one.

f) In the **IP Addresses** box, type all of the addresses that may be used by the actual computer or device that will host the project and present this certificate.

You may leave this box empty. Doing so will not prevent the server certificate from being issued or make it not valid.

g) In the **DNS Names** box, type the names of the domain name servers that will administer the project runtime server.

The default DNS name is **`[NodeName]`** (see above).

h) Click **Delete server certificate**.

This is to make sure the existing certificate (if any) is deleted from the project files, so that a new certificate can be issued with the updated information.

i) Click **OK** to close the *OPC UA Server Self-Signed Certificate Information* dialog box.

> **Note:** The values of **[ServerName]** and **[NodeName]** are determined when the server certificate is issued, and the server certificate is issued when the project is run and the OPC UA Server Runtime task is started. As such, if you test your project on your development workstation, the certificate will be issued using the host name of that computer. You should delete the certificate — by clicking **Delete server certificate** — before you download your project to another computer. If you do not, you will need to manually delete the certificate files on that computer so that a new certificate can be issued with the correct information. For more information, see How to manage OPC UA Server during project run time on page 631.

6. If you want to accept the certificates of all OPC UA clients that try to connect to the OPC UA server, select **Automatically trust client certificates**.

> **Note:** The **Automatically trust client certificates** option is cleared by default, for security reasons. If you do not select this option, you will need to manually add client certificates to the server's "trusted" list. For more information, see How to manage OPC UA Server during project run time on page 631.

7. In the **Stack trace level** list, select the level of trace messages that you want to be saved in the OPC UA communication log.

| Option | Description |
|---|---|
| **Disabled** | Logging is disabled for OPC UA communication; no messages will be saved in the communication log. |
| **Error** | Critical issues that have caused OPC UA communication to fail. These issues must be resolved before communication can resume. (This level is selected by default in new projects.) |
| **Warning** | Non-critical issues that affect run-time performance or might cause OPC UA communication to fail under other conditions. These issues should be resolved as soon as possible.<br>This level includes **Error** (see above). |
| **Info** | Informational messages generated by normal OPC UA communication.<br>This level includes **Error** and **Warning** (see above). |
| **Verbose** | All trace messages generated by the OPC UA communication stack. This can be extremely verbose, and it will generate a very large log file over time. Select this level only if you are troubleshooting serious issues. |

8. Click **OK** to close the *Project Settings* dialog box.

## MAKE PROJECT TAGS AVAILABLE IN OPC UA SERVER

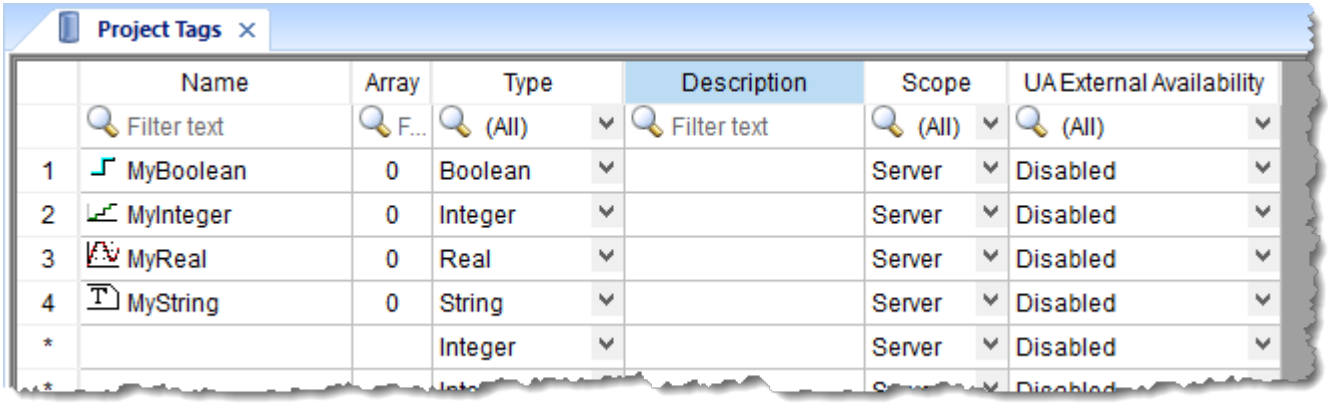
By default and for security reasons, project tags are not made available to OPC UA clients until you make them so, regardless of how you have configured the OPC UA server to run in your project. You must specifically select the tags in the *Project Tags* datasheet and then set the necessary tag property.

You can perform this task at any time — for example, as you create your project tags — but keep in mind that you also need to configure the communication settings and runtime task for OPC UA Server before you actually run your project.

To make project tags available in OPC UA Server:

1. In the **Global** tab of the *Project Explorer*, expand the **Project Tags** folder and then double-click **Datasheet View**.

The *Project Tags* datasheet is displayed.




***Project Tags datasheet***

2. For each project tag that you want to make available to OPC UA clients, select the appropriate option in the **UA External Availability** column.

| Option | Description |
|---|---|
| **Disabled** | The project tag is not available to OPC UA clients. (This option is selected by default for all new tags.) |
| **Read Only** | OPC UA clients can subscribe to and read the value of the project tag. |
| **Read/Write** | OPC UA clients can subscribe to, read, and write the value of the project tag. |

Changes to the project tags are saved immediately; you do not need to close the *Project Tags* datasheet.

For more information, see Set tag properties using the Project Tags datasheet on page 147.

## CONFIGURE THE OPC UA SERVER RUNTIME TASK TO START AUTOMATICALLY

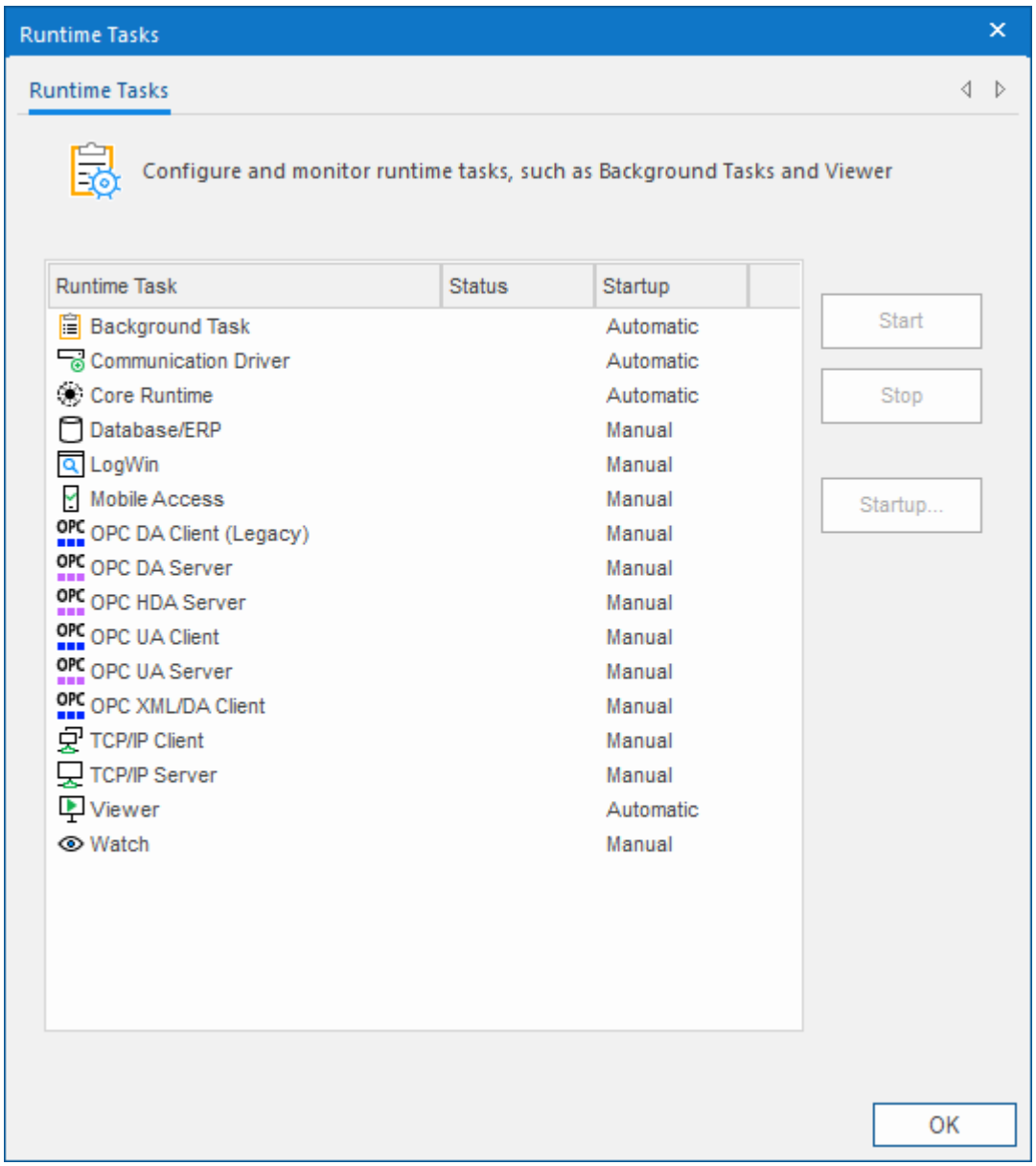
The OPC UA Server Runtime task is one part of the project runtime server. You must configure the task to start automatically with the rest of the project runtime server, when your project is run.

Before you begin this task, you should have configured the communication settings for OPC UA Server and selected the project tags that you want to make available to OPC UA clients.

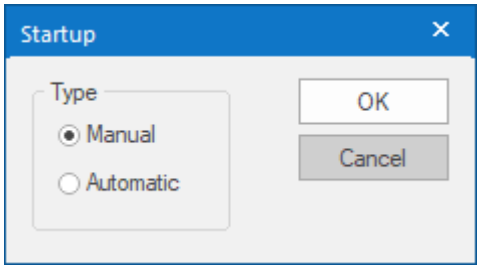
To configure the OPC UA Server Runtime task:

1. On the **Home** tab of the ribbon, in the **Local Management** group, click **Tasks**.

   The *Runtime Tasks* dialog box is displayed.




2. Select **OPC UA Server Runtime** in the list of tasks, and then click **Startup**.
   The *Startup* dialog box is displayed.




***Startup***

3. Select **Automatic**, and then click **OK** to close the *Startup* dialog box.
   The startup mode for the OPC UA Server Runtime task is changed to Automatic.

4. Click **OK** to close the *Runtime Tasks* dialog box.

For more information, see Runtime Tasks on page 136.

## HOW TO MANAGE OPC UA SERVER DURING PROJECT RUN TIME

These are tips and tricks for managing the OPC UA Server module during project run time.

### Deleting and recreating the server certificate files

When the project is run and the OPC UA Server Runtime task is started, it checks whether the server certificate files exist and are located in the project folder at:

```
<project name>\Config\uaserver\own\studio.der
<project name>\Config\uaserver\own\studio.pem
```

(The .der file is the certificate itself, and the .pem file is the associated key. Both files must be present for the certificate to be valid.)

If the files do exist, the project uses them as is. If the files do not exist, the OPC UA server creates them using the certificate information you entered when you configured the OPC UA Server communication settings.

It is important to remember the certificate information can include the special strings **[ServerName]** and **[NodeName]**, which automatically get the host name of the actual computer or device that hosts the project. As such, if you test your project on your development workstation, the certificate files will be created using the host name of that computer, and then those files may be included with the rest of the project files when you download your project to another computer. When the OPC UA server tries to use those files, the certificate information may be incorrect for that computer and the server may not be able to establish secure communication with clients.

You should delete the existing certificate files — by clicking **Delete server certificate**, in the OPC UA Server communication settings — before you download your project from your development workstation to another computer. If you do not, you will need to manually delete the files on that computer so that the OPC UA server can recreate them with the correct certificate information.

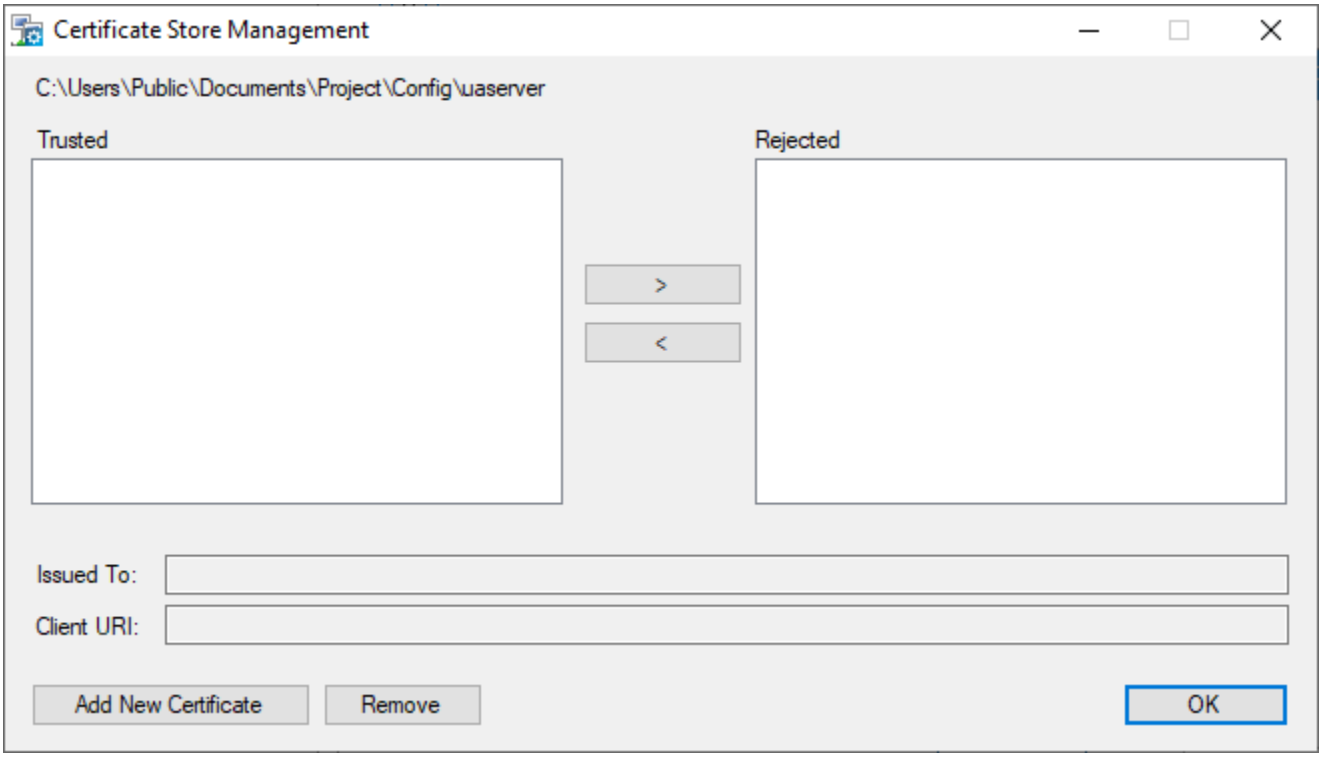
### Trusting and rejecting client certificates

Each OPC UA client that attempts to connect to and communicate with the OPC UA server will present its own client certificate. Your project will automatically trust or reject those certificates depending on whether you selected or cleared the **Automatically trust client certificate** option, in the OPC UA Server communication settings. The certificate files are saved in the appropriate folders in your project's certificate store at:

```
<project name>\Config\uaserver\rejected\
<project name>\Config\uaserver\trusted\
```

In many cases, you will want to manage the certificates on a client-by-client basis, trusting some and rejecting others. This software includes a small utility program called OPC UA Server Certificate Store Management (`OPCUAServerCertStore.exe`), which helps you to view client certificates and move them between folders in the certificate store. If you have the full SCADA software installed, you can open the

program by clicking **Certificate Store Management** in the OPC UA Server communication settings. The program will automatically open the certificate store for the current project:




***OPC UA Server Certificate Store Management***

**Trusted, Rejected**

These lists display the respective contents of the `trusted` and `rejected` folders in your project's certificate store. To move a certificate from one list to the other, select it and then click **>** or **<** as needed.

**Issued To, Client URI**

Certificate information extracted from the selected certificate. This information cannot be edited.

**Add New Certificate**

Click to add a new certificate to the **Trusted** list. You will be asked to locate and open the certificate file. You should be familiar with certificate file names and extensions, as defined by the X.509 standard, so that you can locate the correct file. The file will be copied to the `trusted` folder in your project's certificate store.

**Remove**

Click to remove the selected certificate; the certificate file will be deleted from your project's certificate store, regardless of which list/folder it is in. Be aware that removing a certificate does not prevent a client from presenting the same certificate again in the future. If you want to reject the certificate, move it to the **Rejected** list.

> **Tip:** It is often useful to add client certificates to your project's certificate store while you are still developing your project, so that they can be downloaded with the rest of the project files. However, this assumes you know in advance which clients will try to connect to the server.

If the SCADA software is licensed for Runtime only — that is, if it is installed on another computer and running only as a project runtime server — you might not be able to access the OPC UA Server

communication settings in order to open the Certificate Store Management program. In this case, you can manually run the program by locating it in the SCADA application folder and then double-clicking it. The program file should be located at:

```
C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin
\OPCUAServerCertStore.exe
```

When you open the Certificate Store Management program in this way, it will first ask you to locate and open your project file (*<project name>*.app). Otherwise, the program behaves the same as if you opened it from the OPC UA Server communication settings.

Regardless of how you move the certificate files, you should restart the OPC UA Server Runtime task after you make any changes. This is to make sure rejected clients that were previously trusted are properly disconnected from the server. You do not need to restart the entire project, thanks to the project runtime server's task-based architecture.

### Connecting to the server and browsing for tags

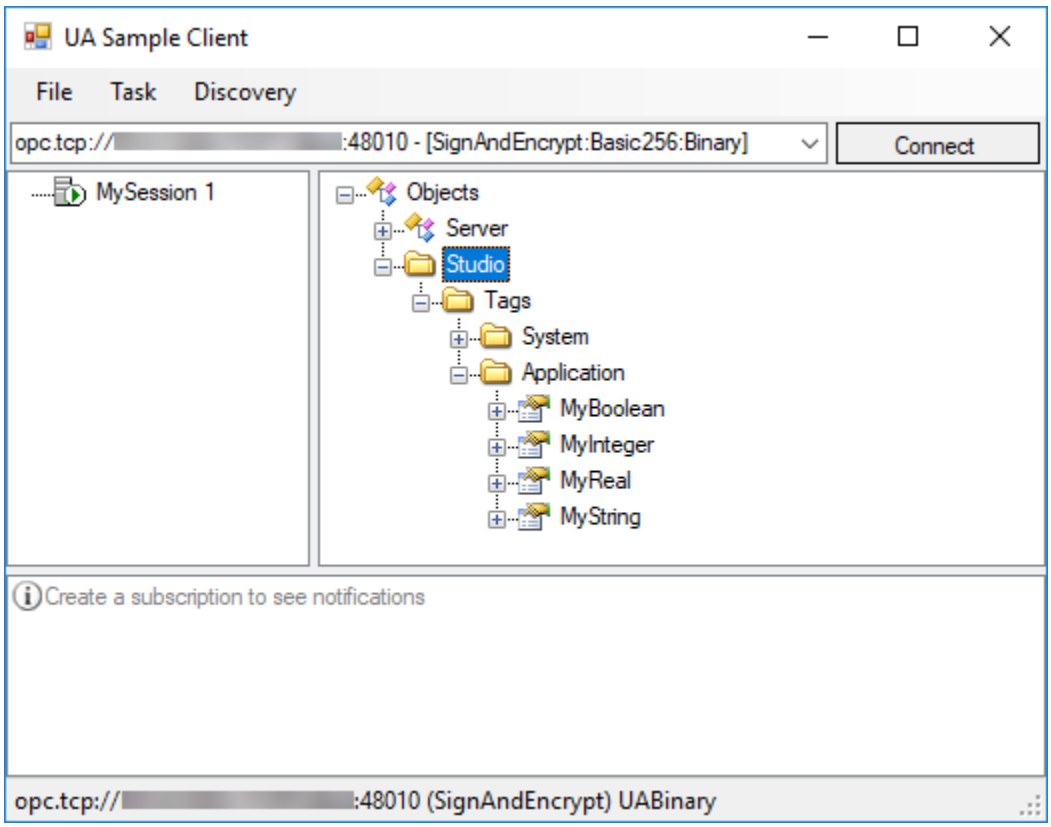
When a project is running and the OPC UA Server Runtime task is started, OPC UA clients should be able to find the server on the network at the endpoint URL that is configured in the OPC UA Server communication settings. If you kept the default node name and port number, that URL should be:

```
opc.tcp://<host name>:48010
```

Clients should also be able to find the server at its IP address instead of its node name. For example:

```
opc.tcp://192.168.0.15:48010
```

Assuming the endpoint URL is valid, the server and client have a security policy in common, and the server trusts the client certificate, the client should be able to connect to the server and then browse tags in the project tags database.




***Example of an OPC UA client connected to your project's OPC UA server***

For more information, see the documentation for your OPC UA client software.

### Communication and server logs

The OPC UA communication stack can generate trace messages, which are saved in a log file in the project folder at:

```
<project name>\Config\uaserver\uaserver.log
```

These trace messages comprise errors, warnings, and other information generated during normal OPC UA communication. The level of verbosity is determined by the **Stack trace level** option in the OPC UA Server communication settings. Be aware that the log file can become very large over time, depending on the level you select.

The OPC UA Server Runtime task can also generate its own messages, separate from the OPC UA communication stack. These messages comprise errors, warnings, and other information about the performance of the task itself as a part of the project runtime server. The messages are included with the rest of the project runtime server's log messages, in the Output/LogWin module. For more information, see Configure the log settings for the Output window on page 703.

## *OPC DA 2.05 Server*

OPC Data Access (OPC DA) is an interoperability standard for exchanging real-time data between clients and servers. The project runtime software includes a built-in OPC DA server that you can use to make your project tags accessible to OPC DA clients on the network.

Before you try to use the OPC DA server, you should be familiar with the OPC DA specification (a.k.a. OPC Classic). For more information, go to: opcfoundation.org/about/opc-technologies/opc-classic/

There are no user-configurable settings for the OPC DA server itself, but to enable the server, you must ensure the **Studio Scada OPC Server** task is started during project run time. You can configure the task to start automatically when the project is run, or you can manually start the task after the project is running. For more information, see Runtime Tasks on page 136.

Once your project is running and the task is started, you should be able to use any compatible OPC DA client program to access your project tags. The OPC DA server address is the same as your project's data server address, its port number is 135 (DCOM), and it should appear to the client as "Studio.Scada.OPC.4".

## *Array Distribution for OPC Clients*

Array distribution is the method by which the elements of an array item on the OPC server are automatically associated with various types of project tags.

> **Note:** This feature is not supported in projects that are configured to run on the Embedded target platform. For more information, see About target platforms, product types, and target systems on page 100.

The OPC clients in this software are capable of reading array items on the OPC server, but they cannot read only certain elements of those array items. Each array item is read in its entirety during every scan of an OPC client worksheet. In other words, it is not possible to associate a single project tag with a single element of an array item.

Instead, when an array item on the OPC server is read, the elements of that array item are automatically distributed to a group of project tags. You need to configure only a single row in the OPC client worksheet that associates the array item with the "first" tag in the group. What qualifies as the "first" tag, however, depends on how you have created and organized your project tags.

The following scenarios describe how array distribution works with different types of project tags.

### Elements of an Array Item → Elements of an Array Tag

In this scenario, you can create an array tag in your project by increasing the Array property of a regular project tag, and then you can associate the array tag with the array item on the OPC server. The elements of the array item are distributed one-for-one to the elements of the array tag. If you specify an array index (e.g., `MyArrayTag[3]`), the distribution will begin at that position. If you do not specify an array index, the distribution will begin at position 0 by default.

#### Examples of array tags

| When Tag Name is... | ...the elements of the OPC array item are distributed to: |
|---|---|
| `MyArrayTag` | **`MyArrayTag[0]`**, **`MyArrayTag[1]`**, **`MyArrayTag[2]`** |
| `MyArrayTag[3]` | **`MyArrayTag[3]`**, **`MyArrayTag[4]`**, **`MyArrayTag[5]`** |

Make sure the array tag in your project is large enough to accomodate the elements of the array item on the OPC server. If it is not large enough, elements will be lost.

### Elements of an Array Item → Sequentially Numbered Project Tags

In this scenario, you can create a series of sequentially numbered project tags, and then you can associate just the first tag in the series with the array item on the OPC server. The elements of the array item are distributed one-for-one to the project tags in the series.

#### Examples of sequentially numbered project tags

| When Tag Name is... | ...the elements of the OPC array item are distributed to: |
|---|---|
| MyTag1 | **MyTag1**, **MyTag2**, **MyTag3** |
| MyTag3 | **MyTag3**, **MyTag4**, **MyTag5** |
| MyTag001 | **MyTag001**, **MyTag002**, **MyTag003** |

Make sure you create enough project tags to accomodate the elements of the array item on the OPC server. If there are not enough tags, elements will be lost.

### Elements of an Array Item → Sequentially Numbered Members of a Class Tag

This scenario is similar to the previous scenario: you can create a class tag that consists of sequentially numbered members, and then you can associate just the member in the series with the array item on the OPC server. The elements of the array item are distributed one-for-one to the members in the series.

#### Examples of sequentially numbered class members

| When Tag Name is... | ...the elements of the OPC array item are distributed to: |
|---|---|
| MyClass.Member1 | **MyClass.Member1**, **MyClass.Member2**, **MyClass.Member3** |
| MyClass.Member3 | **MyClass.Member3**, **MyClass.Member4**, **MyClass.Member5** |
| MyClass.Member001 | **MyClass.Member001**, **MyClass.Member002**, **MyClass.Member003** |

Make sure you create enough members to accomodate the elements of the array item on the OPC server. If there are not enough members, elements will be lost.

### Elements of an Array Item → An Array of Class Tags

This scenario is a combination of the previous scenarios in that you can create an array of class tags, and then the elements of the array item on the OPC server will be distributed amongst all of the members depending on which one you associate with the array item.

#### Examples of members in an array of class tags

| When Tag Name is... | ...the elements of the OPC array item are distributed to: |
|---|---|
| MyArrayClass.Member | **MyArrayClass[0].Member**, **MyArrayClass[1].Member**, **MyArrayClass[2].Member** |
| MyArrayClass[3].Member | **MyArrayClass[3].Member**, **MyArrayClass[4].Member**, **MyArrayClass[5].Member** |
| MyArrayClass[3].Member1 | **MyArrayClass[3].Member1**, **MyArrayClass[3].Member2**, **MyArrayClass[3].Member3** |
| MyArrayClass[3].Member3 | **MyArrayClass[3].Member3**, **MyArrayClass[3].Member4**, **MyArrayClass[3].Member5** |
| MyArrayClass[3].Member001 | **MyArrayClass[3].Member001**, **MyArrayClass[3].Member002**, **MyArrayClass[3].Member003** |

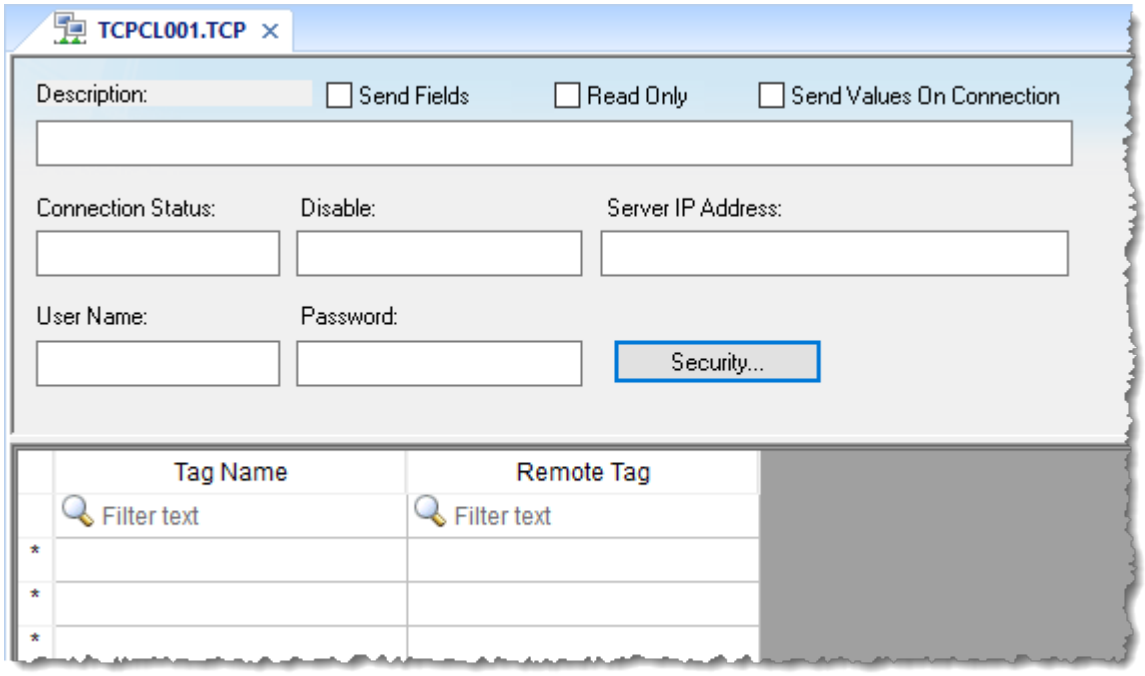
## Communicate with another project runtime server

You can configure a TCP/IP Client worksheet to communicate and exchange data with another project runtime server.

The TCP/IP Client Runtime and TCP/IP Server Runtime tasks enable two or more projects to keep their databases synchronized. These tasks use the TCP/IP protocol to provide communication between projects. Before using the tasks, you must make sure that TCP/IP (Ethernet) communication is properly configured and running on both servers.

- To configure the server: You do not have to configure anything on the server itself. You just have to run the TCP/IP Server Runtime task. You can choose to run it automatically on start up, or run the task manually by clicking **Tasks** (either local or remote) on the **Home** tab of the ribbon. After you start the task, a small icon displays in your system tray.
- To stop the TCP/IP Server Runtime task: Right-click the **TCP/IP Server** icon in the system tray, and then click **Exit** on the shortcut menu.
- To configure the client: You must use the TCP/IP Client worksheet to specify the server IP address and the tags you want to share with the server.

  The TCP/IP Client worksheet is located on the **Comm** tab of the Project Explorer, and it uses the same commands as the Driver worksheet.




***TCP/IP Client worksheet***

Use the following parameters to complete the TCP/IP Client configuration:

- **Description** box: Type a description of the TCP/IP Client worksheet, for documentation purposes only. The TCP/IP Client task ignores this information.
- **Send Fields** option: When this option is selected, the tag properties/fields (i.e., Min, Max, Ack, Unit, LoLoLimit, LoLimit, HiLimit, HiHiLimit, RateLimit, DevSetPoint, DevPLimit, and DevMLimit) are sent with the tag values to the specified server. When this option is cleared, only the tag values (including TimeStamp and Quality, which are required) are sent.

> **Note:** It is possible to add or remove fields in the list of fields sent. For more information, contact Support.

- **Read Only** option: When this option is selected, all communication is one-way and no tag values are written back to the specified server. This is useful when you only need to use the TCP/IP Client to retrieve data from other projects, and it can improve runtime safety and stability.
- **Send Values On Connection** box: When this option is selected and the project is run, the client will ignore the first tag values that it receives from the specified server and instead send its own tag values to the server.
- **Connection Status** box: Type a tag name and the TCP/IP Client Configuration task will update this tag according to its connection status. A tag value of zero indicates the connection is okay. Any other tag value indicates an error code returned by the *Windows Socket* library.
- **Disable** box: Type a tag name in this field. When this tag has any value other than 0, this TCP/IP worksheet will be disabled. Using this field, you can enable/disable the TCP/IP Client worksheet during runtime.
- **Server IP Address** box: Type the IP address and Port (optional) of the target server — for example, `169.254.182.158:123`. The Port should be the same on both the Client and Server stations.

  You can also specify a String tag enclosed in curly brackets (e.g., `{tagname}`) if you want to dynamically change this address during runtime.
- **User Name** and **Password** boxes: Type the credentials for the user account that will be used to log on to the server. That user account must be created in the other project, and it must belong to a group that has the **Enable Remote Security System and Remote Debugging Tools** option selected. For more information, see Group Account dialog on page 664.
- **Security** button: Opens the *TCP/IP Client Security Settings* dialog box, which you can use to enable encrypted communication with the server:
  - **Enable Encrypted Channel** option: Force the client to connect to the server using encrypted communication instead of standard, unencrypted communication. This option is selected by default. For more information about the Encrypted Channel feature, see Communication tab on page 122.
  - **Automatically trust server certificate** option: Have the client automatically trust and save the certificiate presented by the server. For more information, see Managing your project's certificate store on page 126.
- **Tag Name** field: Type the tags you want to share with the server.

  If the tag is an array or a class (or both), the project automatically enables every array position and class member for TCP/IP communication by default.

  To configure a specific array position and/or a specific class member, type the array position and/or class member in square brackets following the tag name. For example,`level[3].member`.
- **Remote Tag** field (*optional*): Type the name of a tag to be linked with the tag you specified in the **Tag Name** field. If you leave this field blank, the project uses the same tag name used in the client and in the server.

> **Note:** If you need to share an array, the tag in the server should contain the same number of elements as the tag in the client. If the tag is a class, the class definition should be the same in both server and client programs. If you do not follow these rules, unpredictable results can occur.

> **Note:** If your project acting as a client cannot connect to the specified server, there might be an incompatibility between the client's project runtime software and the server's project runtime software. Check the version number of the project runtime software on each station. (For software running on the local computer, click **About** on the **Help** tab of the ribbon. For software running on a remote station, use the Remote Management tool to connect to that station and then check the version displayed in the **Status** box.) If the versions are not the same, you might need to upgrade the software on one or both of the stations. For more information, contact your EcoStruxure Machine SCADA Expert 2020 software distributor.

# Project Security

EcoStruxure Machine SCADA Expert 2020 includes a project security system that manages how users and user groups can access a project, during both development and runtime.

## About security modes

In addition to managing users and groups locally, entirely within a single project, you can also get pre-defined users and groups from other projects or from an LDAP-compliant domain server.

Studio supports four security modes:

**Local Only**

This is the standard mode for most projects: users and groups are created in the project development environment, and they apply only to the project for which they are created.

**Distributed – Server**

This is similar to **Local Only**, except that the project's security system configuration is also made available to other projects (that are set to **Distributed – Client**) on the same network. Furthermore, if the project loses its security system configuration for some reason, it can reimport the configuration from one of its client projects.

**Distributed – Client**

When this mode is selected, the project gets its entire security system configuration from another project (that is set to **Distributed – Server**) on the same network. The project caches this configuration and can continue to run even if it loses communication with the server project.

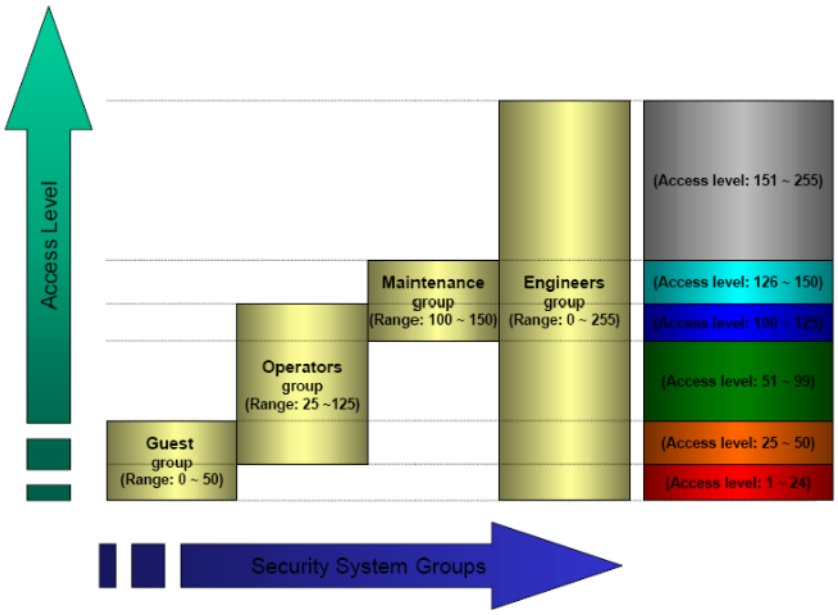
**Domain (LDAP)**

The Lightweight Directory Access Protocol (LDAP) is a recognized standard for managing users and groups across many different applications on a network. When this mode is selected, the project gets its users and groups from an LDAP-compliant domain server, such as Microsoft Active Directory for Windows or OpenLDAP for Linux. However, only the user names, passwords, and group memberships are taken from the domain; specific rights for each group must still be configured within the project.

## About security access levels

Almost every item in a project — screen object, object animation, project screen, task worksheet — can be assigned a security access level. That access level determines which user groups can edit the item during development and/or use the item during runtime.

There are 255 possible access levels, allowing a large amount of granularity. Each user group is configured with ranges of levels for both development and runtime, and the groups' ranges may overlap.




***Example of security access levels***

This means that for a user to be able to edit and/or use an item, the item's access level must fall within the range specified for that user's group.

For example, **UserA** of **GroupA** has a security access level range of `1-10` and **UserB** of **GroupB** has a security access level range of `5-15`. To continue the example:

- Item #1 has Access Level = 1
- Item #2 has Access Level = 7
- Item #3 has Access Level = 12
- Item #4 has Access Level = 20

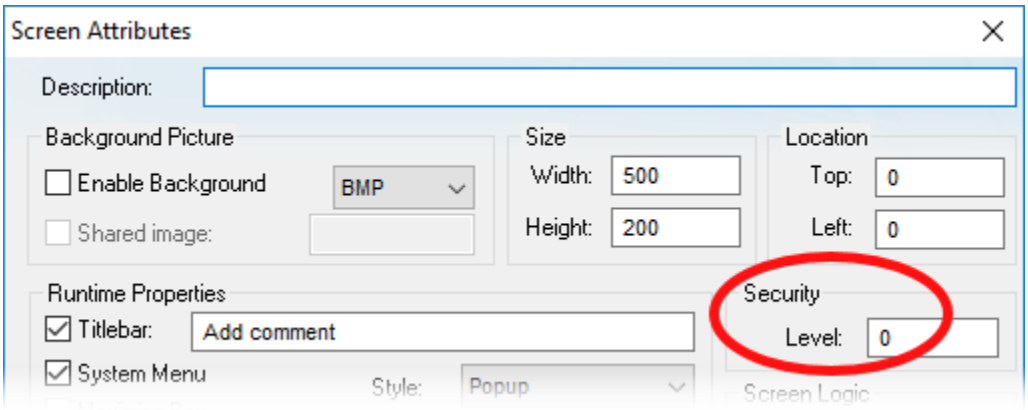
Consequently,

- Only **UserA** can access Item #1
- Both users can access Item #2
- Only **UserB** can access Item #3
- Neither user can access Item #4

> **Note:** The default access level for all items is 0, and all users can use all items at that level.

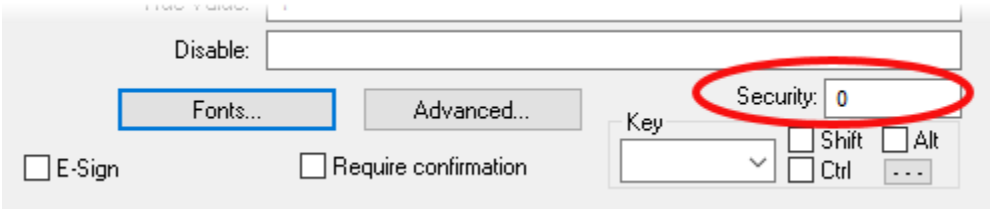
### Setting access levels for different types of items

For project screens, the access level can be set in the *Screen Attributes* dialog.




***Security (access level) setting in Screen Attributes dialog***

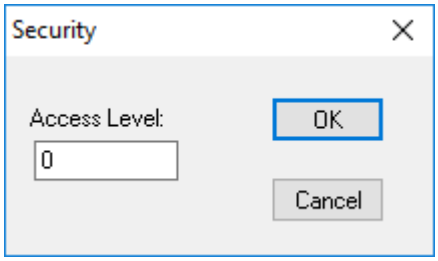
For screen objects and object animations, the access level can be set in the individual *Object Properties* dialog. (If the option is not available in the main dialog, then it should be available in one of the sub-dialogs.)




***Security (access level) setting in Object Properties dialog***

For task worksheets, do the following:

1. Open the worksheet for editing.
2. Click anywhere in the body of the worksheet. The **Access Level** control, on the Project tab of the ribbon, is enabled.
3. Click **Access Level**. The *Security* dialog is displayed.




***Security dialog***

4. In the **Access Level** box, type an access level for editing the worksheet.
5. Click **OK**.

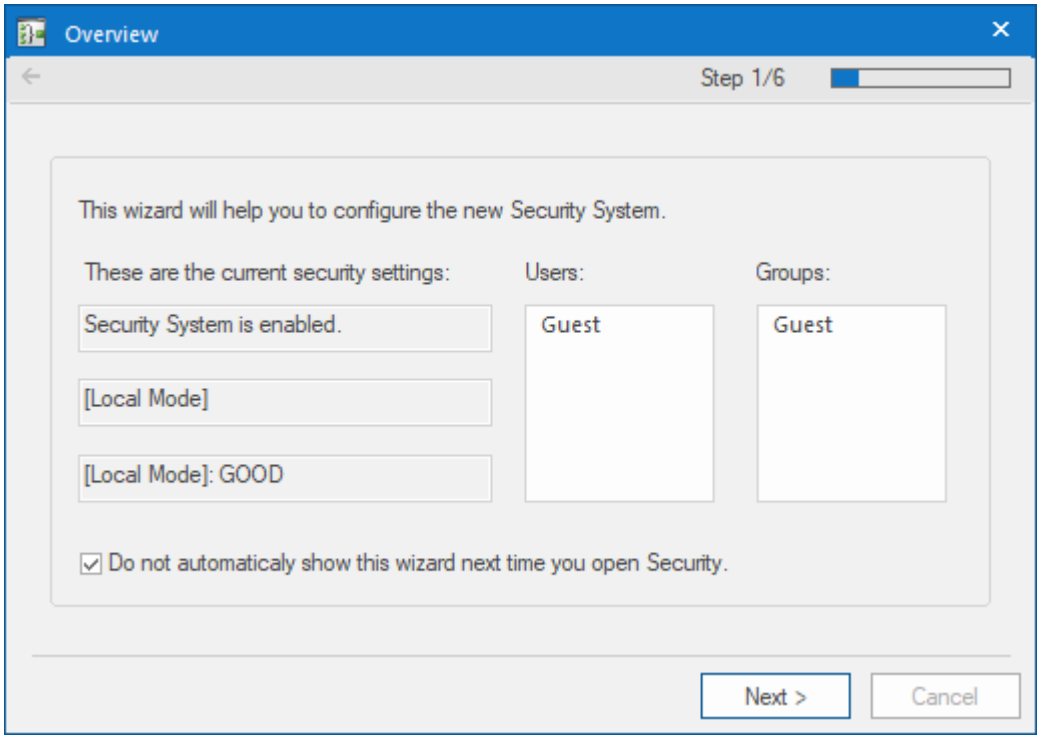
## Using the Security System Configuration Wizard

The *Security System Configuration Wizard* helps you through the steps to configure the security system for your project.

1. Do one of the following in order to start the wizard:
   - Create a new project. The wizard starts automatically as part of that procedure, because the security system is enabled by default for new projects.
   - If you are configuring the security system for the first time in an existing project, do one of the following:
     - On the **Project** tab of the ribbon, in the **Security System** group, click **Configure**; or
     - In the **Global** tab of the *Project Explorer*, right-click **Security**, and then click **Settings** on the shortcut menu.

     After the first time, doing either of these will open the *Security System* dialog box instead.
   - If you have already configured the security system for your project, open the *Security System* dialog box as described above and then click **Run Wizard**.

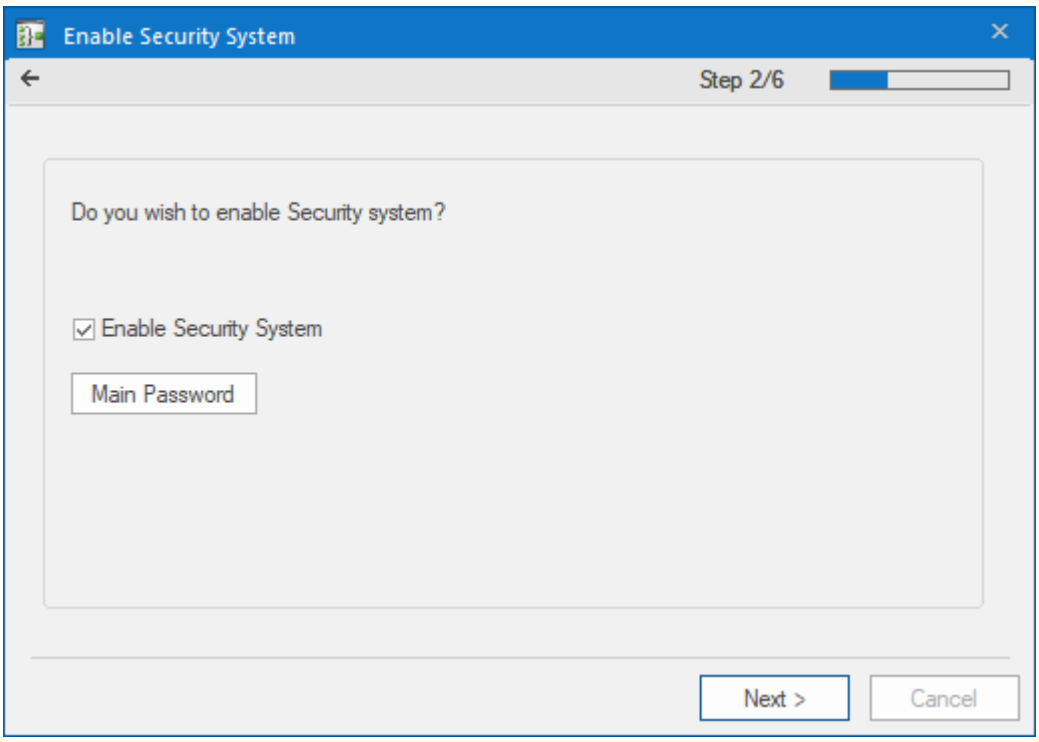
   The first page of the wizard is displayed.




   This page always shows how the security system is currently configured.

2. Click **Next**.

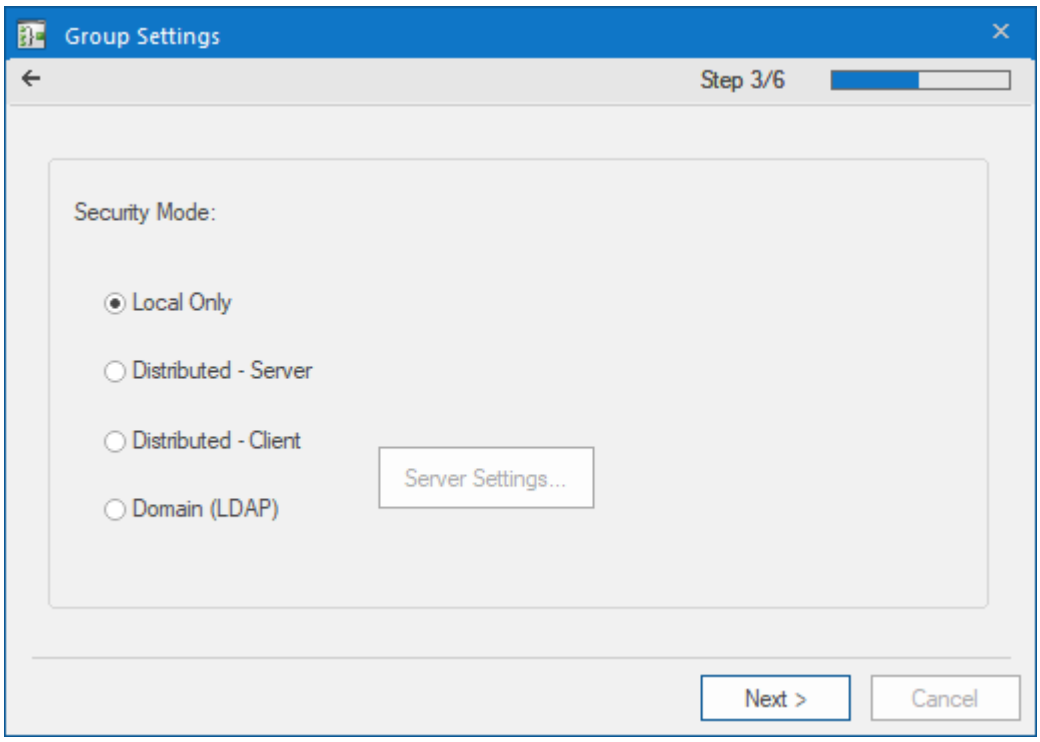
   The second page of the wizard is displayed.



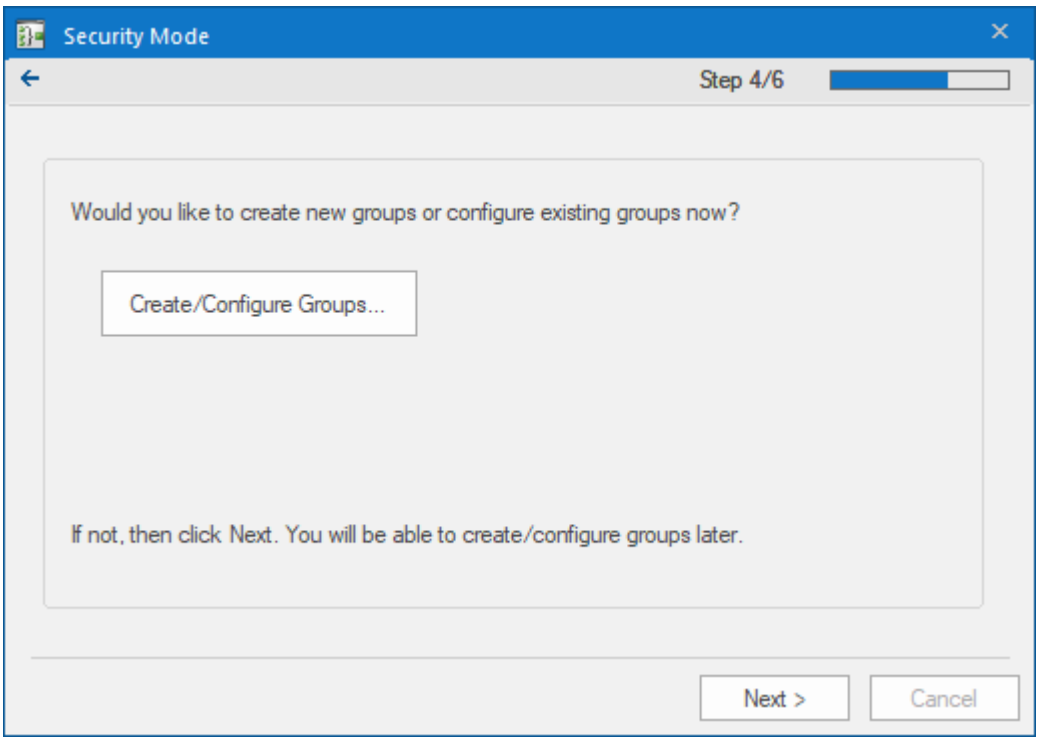

3. Select the **Enable Security System** option, if it is not already selected.
   The security system is enabled by default for new projects.
4. Set the Main Password for your project:
   a) Click **Main Password**.
      The *Security System Main Password* dialog box is displayed.
   b) In the **New Password** box, type a password for the security system itself.
   c) In the **Confirm Password** box, type the password again.
   d) Click **OK** to save the password and close the dialog box.

   The Main Password is separate from the passwords for individual user accounts, including any accounts that you create for yourself and then use to develop and test your project. Record the Main Password somewhere safe, because you will need it in order to make further changes to the security system.
5. Click **Next**.

   The third page of the wizard is displayed.




6. Select the security mode for your project.
   For more information about security modes, see About security modes on page 640.
7. If you selected either **Distributed – Client** or **Domain (LDAP)**, click **Server Settings** and then configure the settings as needed.
   For more information, see Configuring server settings for security modes on page 649.
   When you are done, it will return you to this page of the wizard. Keep in mind that you can make further changes later.
8. Click **Next**.
   The fourth page of the wizard is displayed.



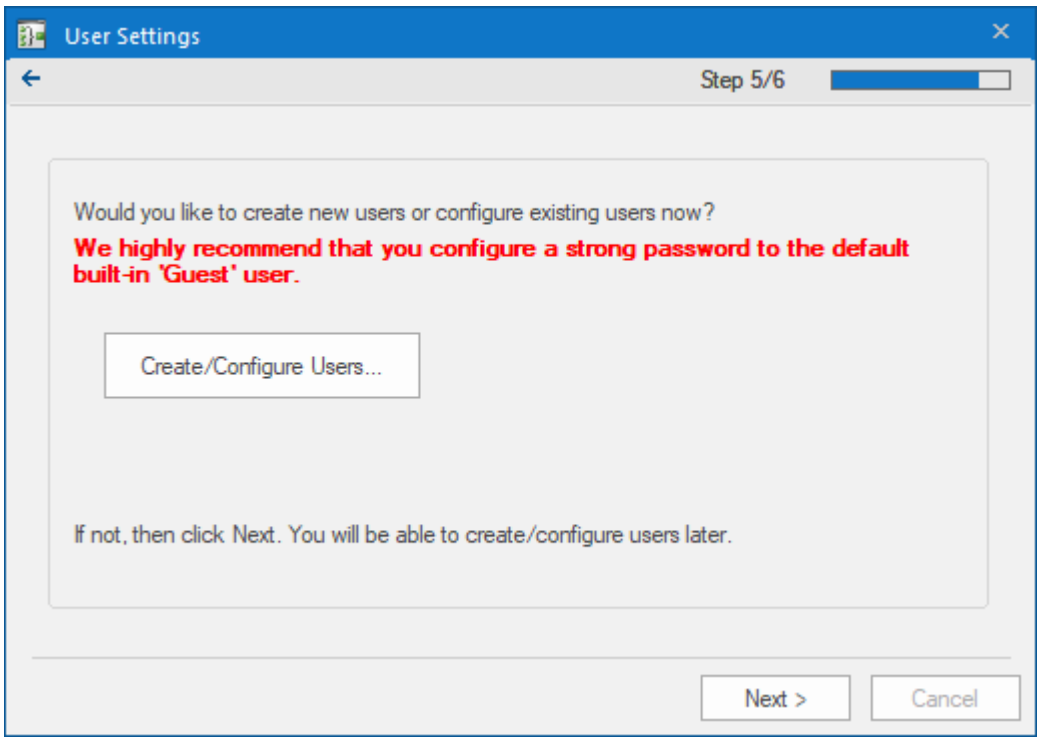

9. If you want to create or configure groups, click **Create/Configure Groups** and then configure the settings as needed.

   For more information about group accounts, see Group Account dialog on page 664.

   When you are done, it will return you to this page of the wizard. Keep in mind that you can make further changes later.

10. Click **Next**.

    The fifth page of the wizard is displayed.



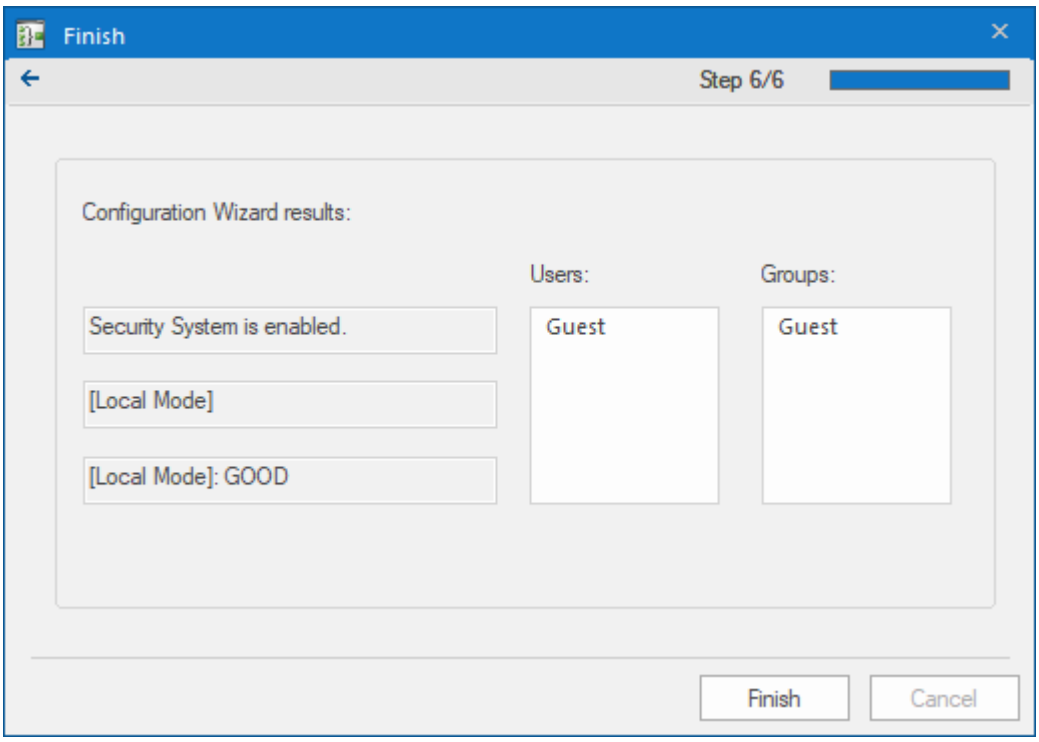

11. If you want to create or configure users, click **Create/Configure Users** and then configure the settings as needed. At a minimum, you should either set a password for the default user Guest or limit its access privileges.

    For more information about user accounts, see Creating and configuring users on page 671.

    When you are done, it will return you to this page of the wizard. Keep in mind that you can make further changes later.

12. Click **Next**.

    The sixth page of the wizard is displayed.



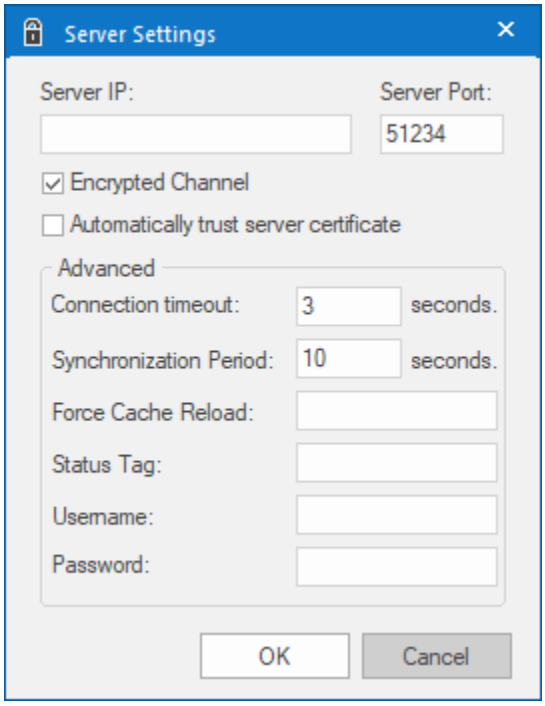

13. Review your configuration, and then click **Finish** to close the wizard.

## Configuring server settings for security modes

If you set your project's security mode to either **Distributed – Client** or **Domain (LDAP)**, you must also configure the server settings for that mode.

When you click **Server Settings** in either the security system configuration wizard or the main *Security System* dialog box, the appropriate *Server Settings* dialog box is displayed.

### Security Mode: Distributed – Client




To configure the server settings:

1. In the **Server IP** and **Server Port** boxes, enter the IP address and port number of the project runtime server that is configured as **Distributed – Server**.
2. If the Encrypted Channel feature has been enabled in that server's communication settings, you can select the **Encrypted Channel** option here to force the client to connect to the server using encrypted communication instead of standard, unencrypted communication. Selecting this option will automatically change the port number (in the **Server Port** box) to the default port number for the Encrypted Channel feature, so if that is not the correct port number for that server, correct it now. For more information about the Encrypted Channel feature, see Communication tab on page 122.
3. If you want the client (i.e., your project) to automatically trust and store the certificiate presented by the server, select the **Automatically trust server certificate** option. For more information, see Managing your project's certificate store on page 126.
4. Review the advanced settings.

**Connection timeout**

The duration (in seconds) after which the client will stop trying to connect to the server. The default is 3 seconds. A connection can time out if the server is slow to respond or not available at the specified address.

**Synchronization Period**

The period (in seconds) at which the client will try to synchronize with the server and update the local cache. The default is 10 seconds.

**Force Cache Reload**

If you want to force the local cache to reload, enter the name of a project tag (Integer or Boolean type) in this box. When the tag value is TRUE (non-zero), the client will reload the local cache with the server's current information regardless of whether the server cache is outdated (i.e., the timestamp on the server cache is older than the timestamp on the local cache).

**Status Tag**

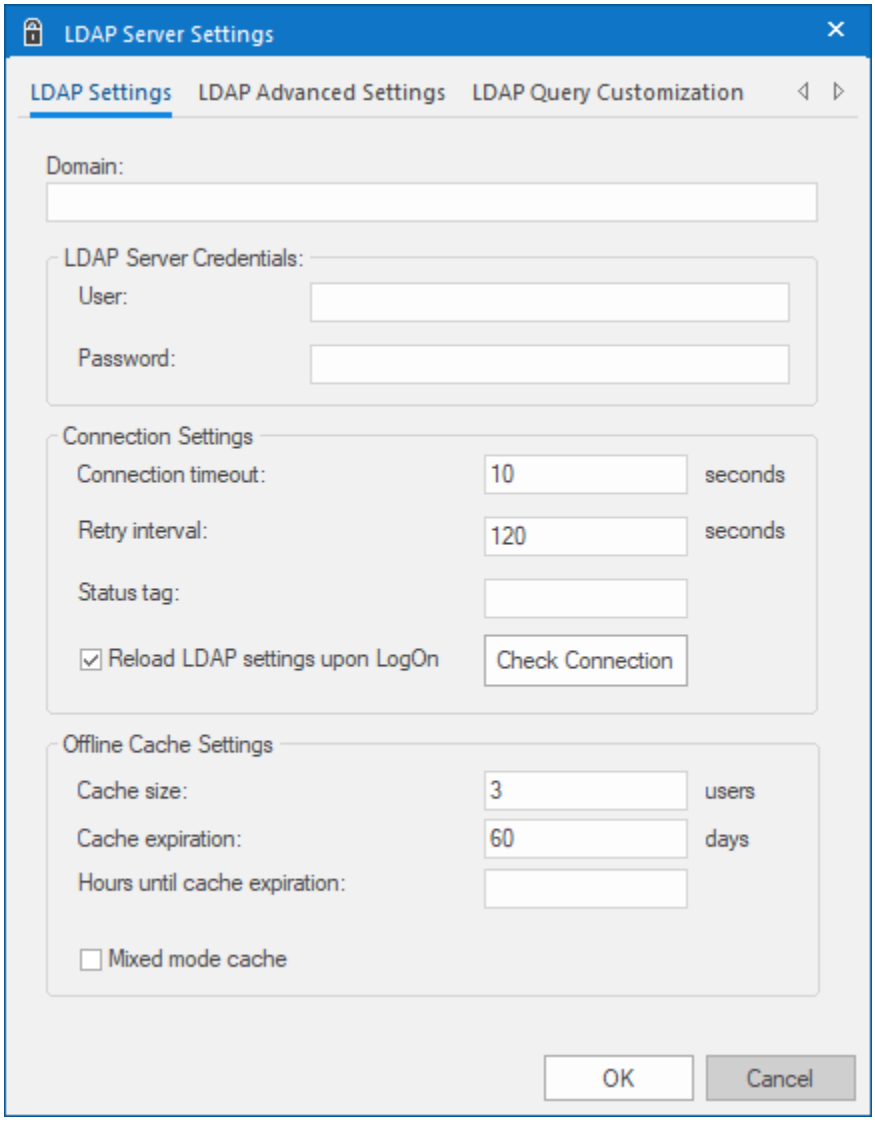
If you want to monitor the status of the security system during project run time, enter the name of a project tag (Integer type) in this box. The specified tag will receive continuously updated values that indicate the status of the client's connection to the server. The possible values are:

| Value | Description |
|---|---|
| 0 | No cache |
| 1 | Updated cache |
| 2 | Outdated local cache |
| 3 | Outdated server cache |
| 4 | Disconnected from server |

For more information, see GetSecuritySystemStatus on page 1093.

5. In the **User Name** and **Password** boxes, enter the credentials for the user account that will be used to log on to the server. That user account must be created in the server's project, and it must belong to a group that has the **Enable Remote Security System and Remote Debugging Tools** option selected. For more information, see Group Account dialog on page 664.
6. Click **OK**.

## Security Mode: Domain (LDAP)




To configure the server settings:

1. In the **Domain** box, enter the domain or host name of the LDAP server.
2. Under **LDAP Server Credentials**, in the **User** and **Password** boxes, enter your credentials for the LDAP server. You must have sufficient privileges to get the server's list of users and groups, because your project will use that list to validate other users' attempts to log on.

   The format of your user name is determined by the server configuration (e.g., `username`, `domain\username`, `username@domain`). If you are not sure, ask your LDAP server administrator.

   You can enter string expressions for these settings (e.g., `{MyUser}`, `{MyPassword}`). When the values of the expressions change during project run time, the credentials are changed to match and will be used the next time your project tries to get the server's list of users and groups.
3. Under **Connection Settings**, review the following settings:

   **Connection timeout**

   The duration (in seconds) after which the project runtime will stop trying to connect to the LDAP server. The default is 10 seconds. A connection can time out if the server is slow to respond or not available at the specified address.

   **Retry interval**

The interval (in seconds) between attempts to connect to the LDAP server, after the connection has timed out or failed. The default is 120 seconds. The project runtime will keep trying to connect until either the connection is established or the project is stopped.

**Status tag**

If you want to monitor the status of the security system during project run time, enter the name of a project tag (Integer type) in this box. The specified tag will receive continuously updated values that indicate the status of the client's connection to the LDAP server. The possible values are:

| Value | Description |
|---|---|
| 0 | Connection timeout |
| 1 | Bind timeout |
| 2 | Query timeout |
| 3 | Disconnected |
| 4 | Connected |
| 5 | No users or groups returned by query |
| 6 | Invalid user or group |

For more information, see GetSecuritySystemStatus on page 1093.

**Reload LDAP settings upon LogOn**

This option is selected by default. It means your project will reload the list of users and groups from the LDAP server each time a user attempts to log on to the project, and that can make it easier for the user to select their name and then log on.

Reloading the LDAP settings might take some time, however, especially if the list is long or the server is slow to respond. That can affect your project's run-time performance, so you may choose to clear this option and have more control over when your project reloads the settings.

If this option is cleared, your project will load the LDAP settings once when the project is run, and then it will reload the settings only when the `GetSecuritySystemStatus` function is executed.

4. Click **Check Connection** to confirm that your project can connect to the LDAP server using the specified settings. If it cannot, review and correct the settings.
5. Under **Offline Cache Settings**, review the settings for the offline cache of recent users. Users are cached when they successfully log on to the project, and cached users can log on even when the LDAP server is not available. The cache is a "first in, first out" (FIFO) list, which means old users are removed when new users are added. There are two exceptions to this rule: the project's default user (e.g., "Guest") and the user that you specified under **LDAP Server Credentials** are always kept in the cache.

**Cache size**

The number of most recent users that are kept in the offline cache. The default is 3 users. To keep an unlimited number of users in the cache, enter `0` for this setting.

**Cache expiration**

The number of days after which the offline cache will expire, starting from the most recent user logon. The default is 60 days. To make the cache never expire, enter `0` for this setting, but we do not recommended this.

**Hours until cache expiration**

If you want to monitor the offline cache during project run time, enter the name of a project tag in this box. The specified tag will receive a value equal to the number of hours remaining until the cache expires.

**Mixed mode cache**

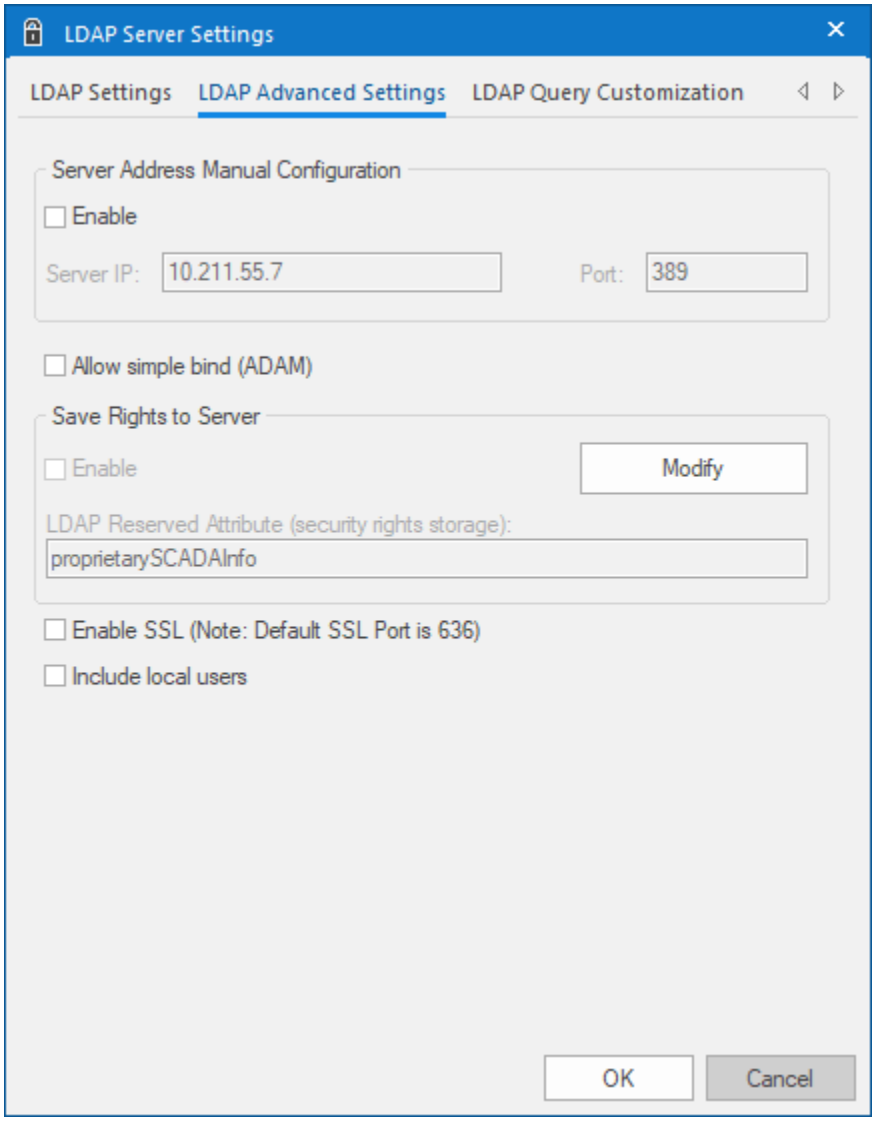
If you want to allow users to log on as local users when the LDAP server is not available, select this option. Local users can be created while **Security Mode** is set to **Local Only**, but they typically cannot be selected while **Security Mode** is set to **Domain (LDAP)** and the LDAP server is available. When the LDAP server is not available, however, users can select from both the previously

created local users and the cached users from the LDAP server. In short, the **Mixed mode cache** option serves as a sort of backup when the LDAP server is not available, in case someone needs to log on as a user that was not cached.

6. Click **OK**.

> **Tip:**
>
> If you want the project security system to generate a log message each time it updates the offline cache, manually edit your project file (*<project name>*.APP) to add the following setting:
>
> ```
> [Options]
> EnableSecuritySystemTraceLdapOfflineCache=1
> ```

There are additional settings in the *LDAP Advanced Settings* tab of the dialog box. These settings should be configured only by experienced LDAP server administrators.




**Server Address Manual Configuration**

If for some reason the LDAP server cannot be accessed using its domain or host name, you can manually configure the server address: select **Enable**, and then enter the IP address and port number of the LDAP server. The default port for LDAP is 389, but it can be changed on the server so you should confirm it before you configure these settings.

**Allow simple bind (ADAM)**

LDAP normally requires secure binding for authentication, but Active Directory Application Mode (ADAM) in Windows Server 2003 does not fully support secure binding. To allow simple binding with an ADAM server, select this option. Simple binding means that user credentials are sent in clear text, so you should secure the connection by other means such as VPN, TLS/SSL, or proxies.

**Save Rights to Server**

By default, group and user rights are saved entirely within your project. You can save those rights back to the LDAP server, however, either to make them available to other projects that use the same LDAP server or for simple redundancy.

To make this option work, you must first extend the server's LDAP schema to contain additional information about the project security system. See Extending the LDAP schema to allow saving of security rights on page 656.

Once that is done, click **Modify** to provide your LDAP server credentials, and then select **Enable**.

**Enable SSL**

To enable a Secure Sockets Layer (SSL) connection between the the project runtime server and the LDAP server, select this option. An SSL connection is required for users to be able to change their passwords on the LDAP server during project run time. Otherwise, changes can be made only outside the project through other LDAP clients.

If you select this option and you have also manually configured the server address, make sure the address you configured matches the address on the server's certificate.

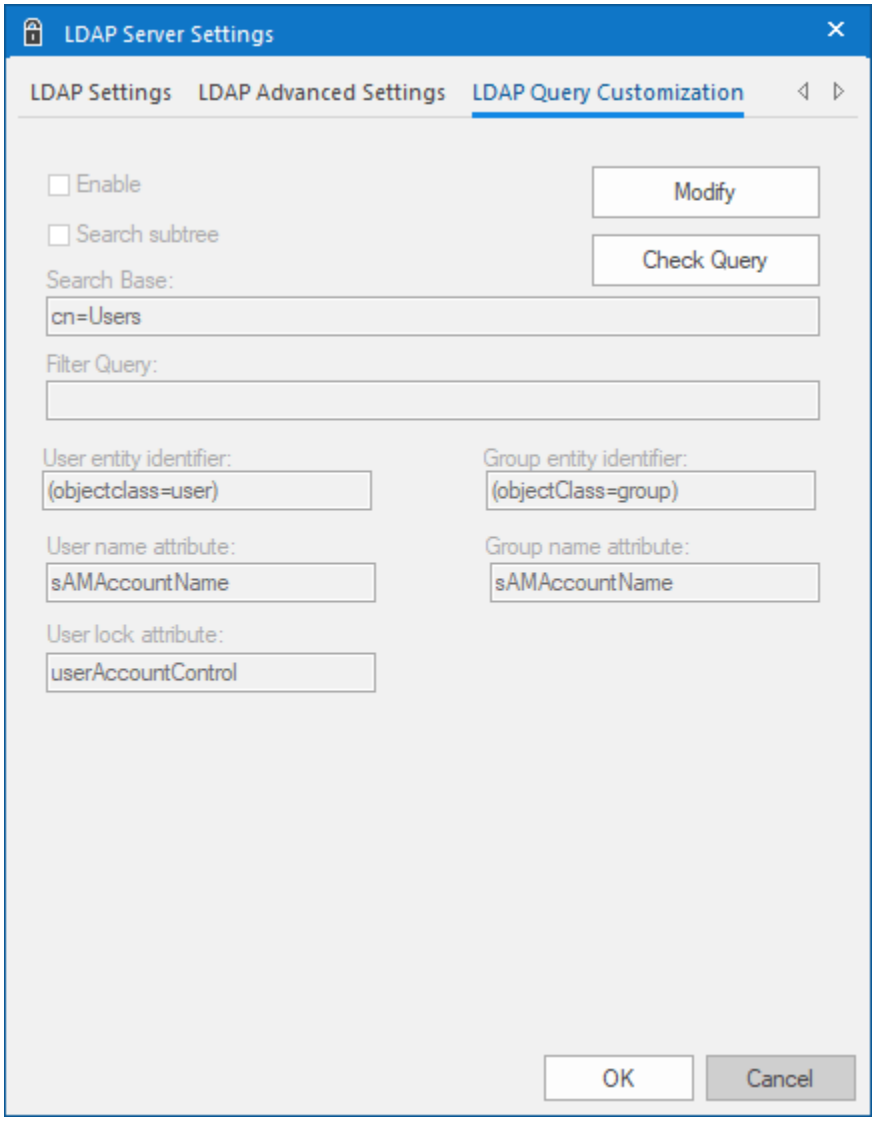
The default port for LDAP via SSL is 636, but it can be changed on the server so you should confirm it before you select this option.

**Include local users**

Local users can be created while **Security Mode** is set to **Local Only**, but they typically cannot be selected for logon while **Security Mode** is set to **Domain (LDAP)** and the LDAP server is available. Select this option to allow users to select from both the local users and the server accounts when they log on.

This option was introduced after the **Mixed mode cache** option (in the basic settings; see above), and it is being kept separate in order to maintain reverse compatibility with existing projects that already have the **Mixed mode cache** option selected. Selecting this option should not affect the performance of your project, and if this option is selected, the **Mixed mode cache** option is also selected automatically and cannot be cleared.

In the *LDAP Query Customization* tab of the dialog box, you can customize your project's query to the LDAP server.




By default, the LDAP server provides a list of all registered users and groups, so in a large or complex network environment, this can result in an impractically long list to manage when you are configuring your project security system. To shorten the list, customize the query to filter any users who should never have access to your project: click **Modify** to provide your LDAP server credentials, select **Enable**, and then configure the **Search Base** and **Filter Query** settings. For the proper syntax, consult the LDAP server documentation.

> **Note:** The **Filter Query** string is limited to 2048 characters.

Also, some non-standard LDAP implementations — such as Linux-based LDAP servers and Active Directory Application Mode (ADAM) in Windows Server 2003 — use different entity identifiers and attributes. Those can be customized in this dialog box, but again, it should be done only by an experienced LDAP administrator. For example:

| LDAP Server | User name attribute | Group name attribute | User lock attribute |
|---|---|---|---|
| Active Directory | `sAMAccountName` | `sAMAccountName` | `userAccountControl` |
| Active Directory Application Mode (ADAM) | `Name` | `Name` | `userAccountControl` |

### *Extending the LDAP schema to allow saving of security rights*

In order to save EcoStruxure Machine SCADA Expert project security rights back to a Domain (LDAP) server, the server's LDAP schema must be extended to contain the additional information.

The server must already be configured and running on your network, and you must have sufficient privileges to make changes to the server configuration.

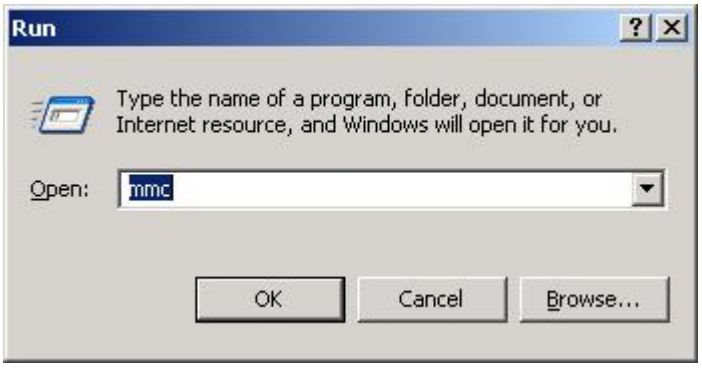
In this procedure, you will create a new attribute called "proprietarySCADAInfo" to contain the EcoStruxure Machine SCADA Expert project security rights, and then you will add the attribute to the "person" and "group" classes in the server configuration. These classes correspond to users and groups in the project security system.

Please note this procedure only shows how to extend the schema in Microsoft Active Directory running on Windows Server 2003. The exact procedure is different for other LDAP servers and operating systems, but the basic steps should be essentially the same. Please consult your LDAP server documentation.

> **Note:** Extending a server's LDAP schema cannot be undone.

1. Register the schema management DLL.
   a) Click **Start** > **Windows System** > **Command Prompt**.
   A *Command Prompt* window is displayed.
   b) At the prompt, type `cd %SystemRoot%\System32` and then press Return.
   The working directory is changed.
   c) Type `regsvr32 schmmgmt.dll` and then press Return.
   If the DLL is successfully registered, then a confirmation message is displayed.

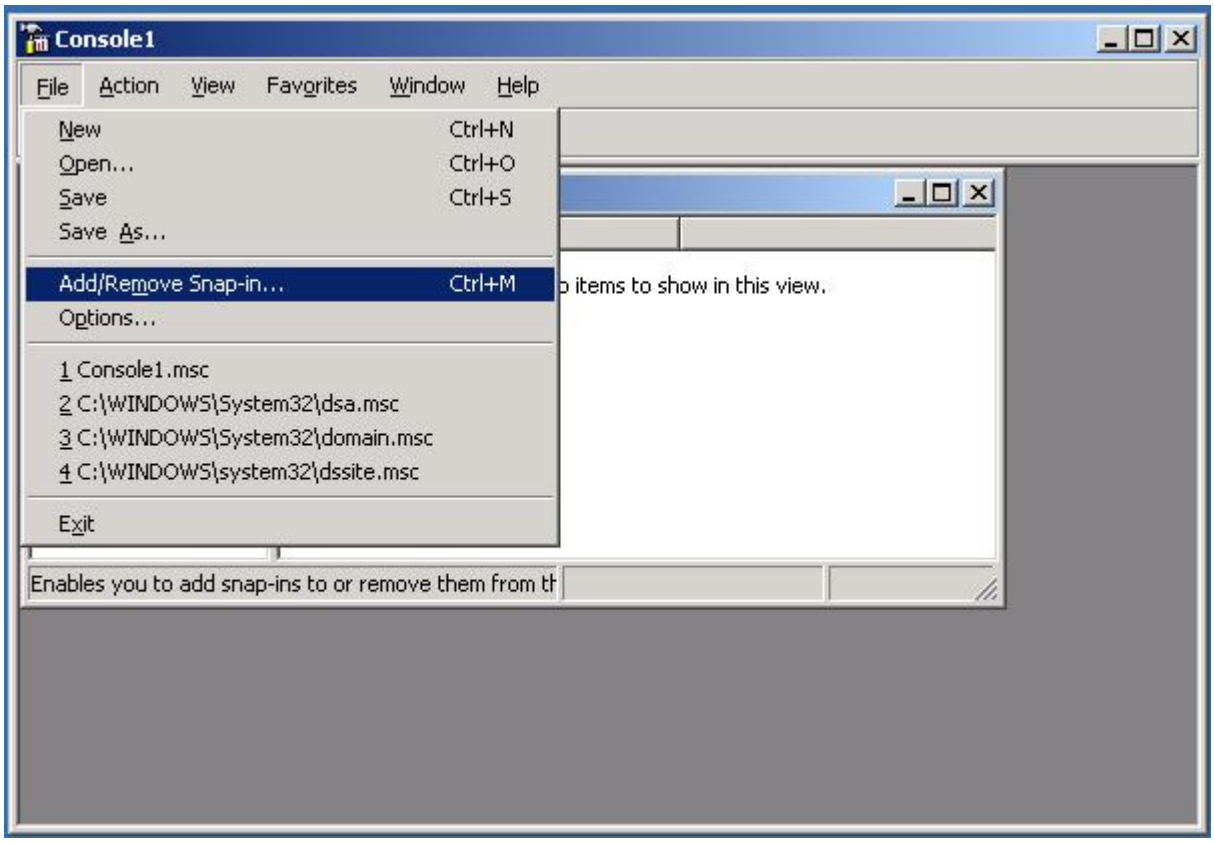
   d) Click **OK** to dismiss the message.
   e) Close the *Command Prompt* window.
2. Add the Active Directory Schema snap-in to the console root.
   a) Click **Start** > **Windows System** > **Run**.
   A *Run* dialog is displayed.
   b) In the **Open** box, type `mmc`, and then click **OK**.




   (If you have User Access Control (UAC) enabled, then you will be asked if you want to allow Microsoft Management Console to make changes. Click **Yes**.)

A console window is displayed.

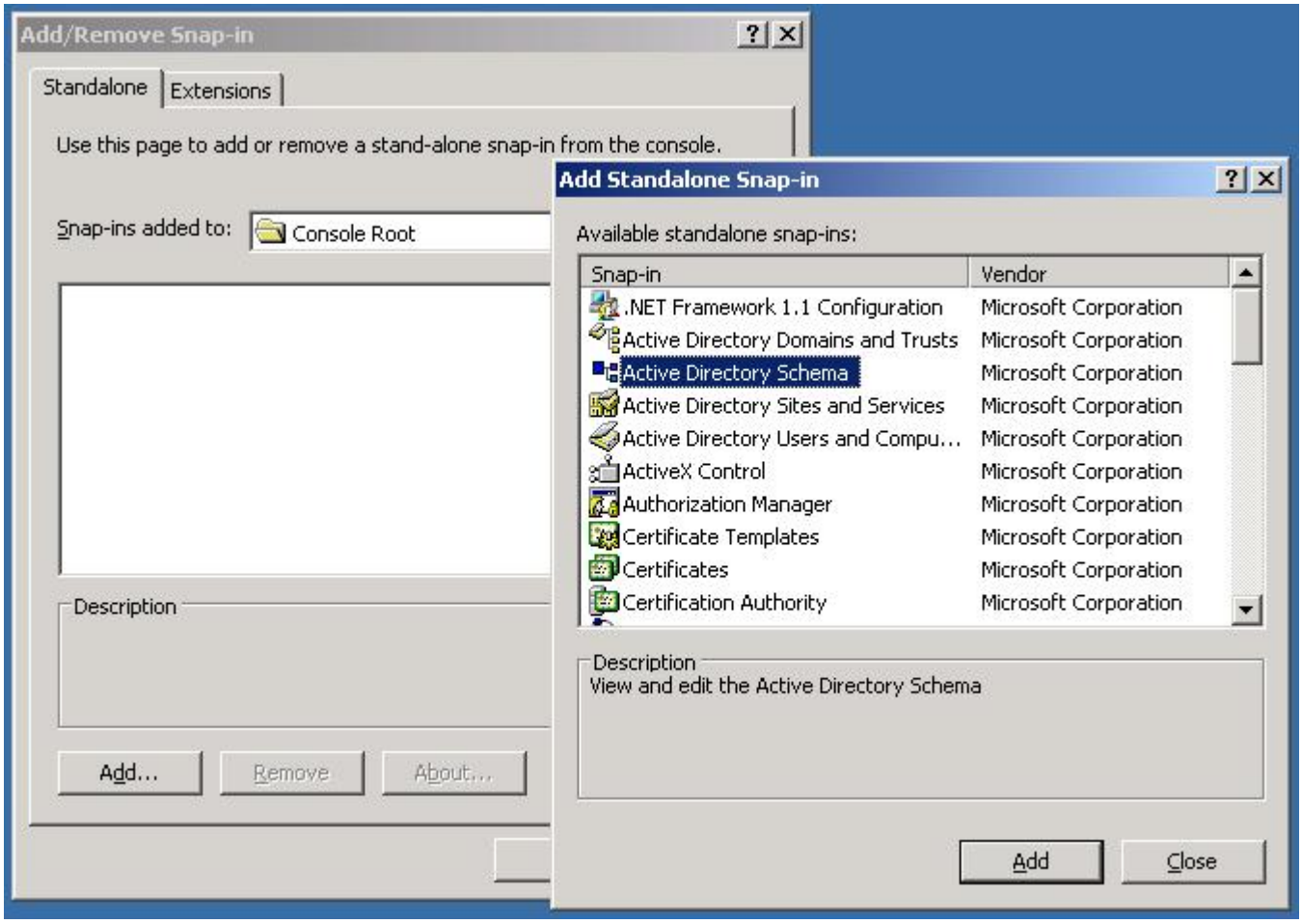
c) In the console window, click **File** > **Add/Remove Snap-in**.




The *Add/Remove Snap-in* dialog is displayed.

d) In the **Snap-ins added to** list, select **Console Root**, and then click **Add**.
The *Add Standalone Snap-in* dialog is displayed.

e) In the list of available snap-ins, select **Active Directory Schema**, and then and click **Add**.

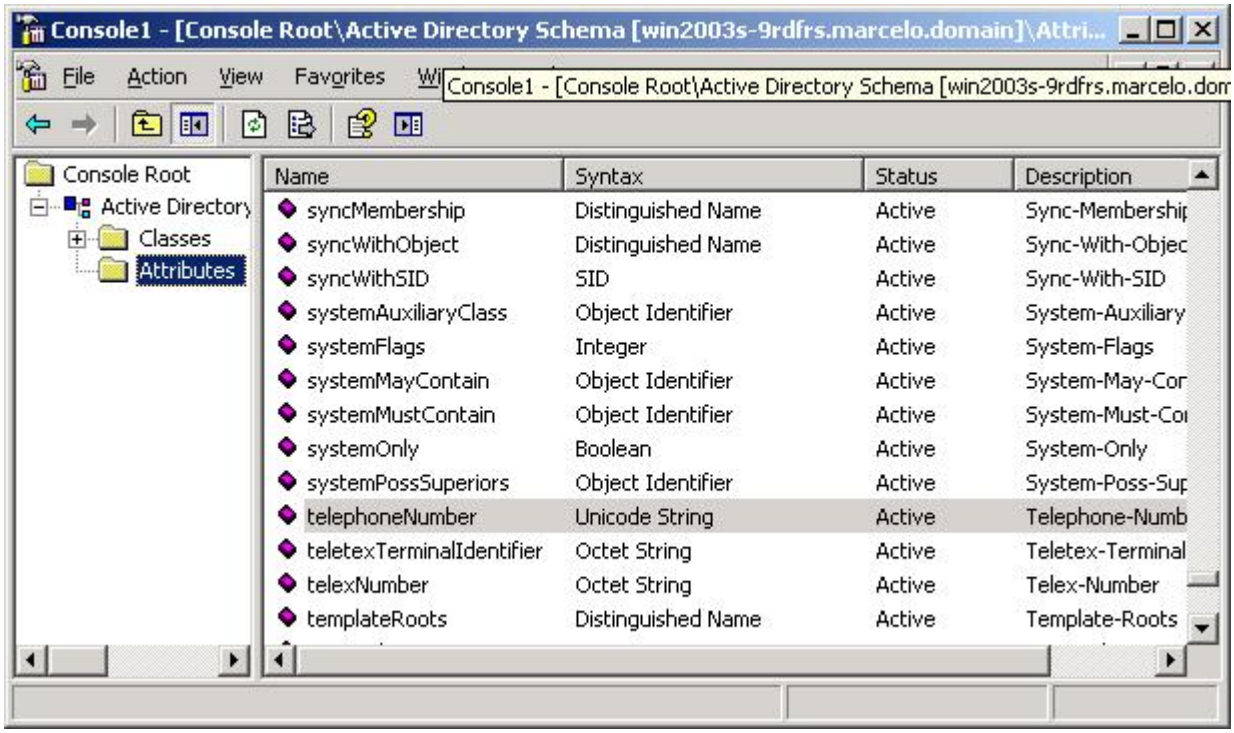



The snap-in is added to Console Root.

f) Click **OK** to close the *Add/Remove Snap-in* dialog.

3. Create the proprietarySCADAInfo attribute in the Active Directory Schema snap-in.

a) In the **Console Root** tree-view, expand **Active Directory Schema**.




b) Right-click **Active Directory Schema** > **Attributes**, and then click **Create Attribute** on the shortcut menu.
A message is displayed explaining that your schema will be permanent changed.

c) Click **Continue**.
A *Create New Attribute* dialog is displayed.

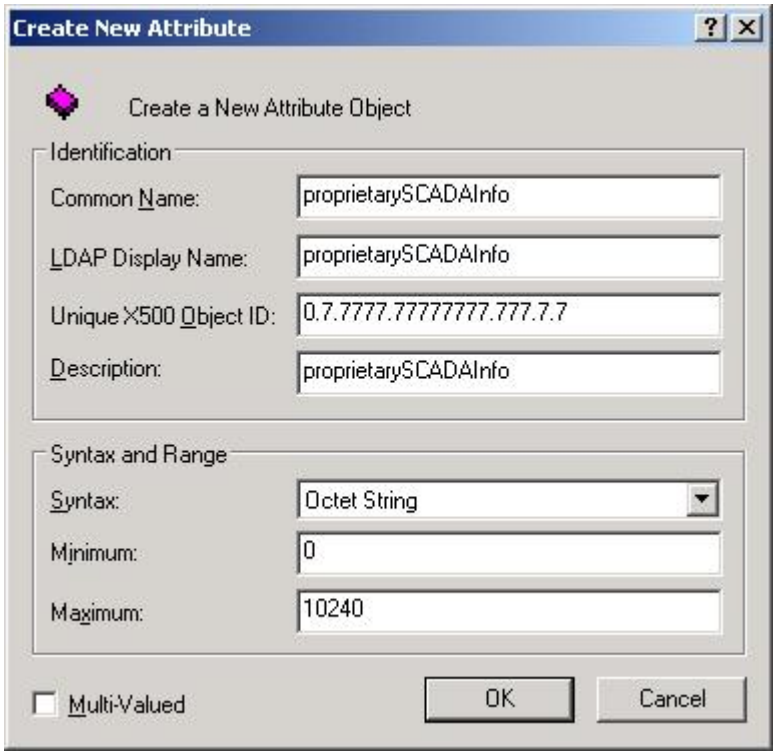
d) In the dialog, complete the fields as follows.

- **Common Name:** `proprietarySCADAInfo`
- **LDAP Display Name:** `proprietarySCADAInfo`
- **Unique X500 Object ID:** `0.7.7777.77777777.777.7.7`

> **Note:** An unique Object ID should be used.

- **Description:** `proprietarySCADAInfo`
- **Syntax:** `Octect String`
- **Minimum:** `0`

- **Maximum:** 10240

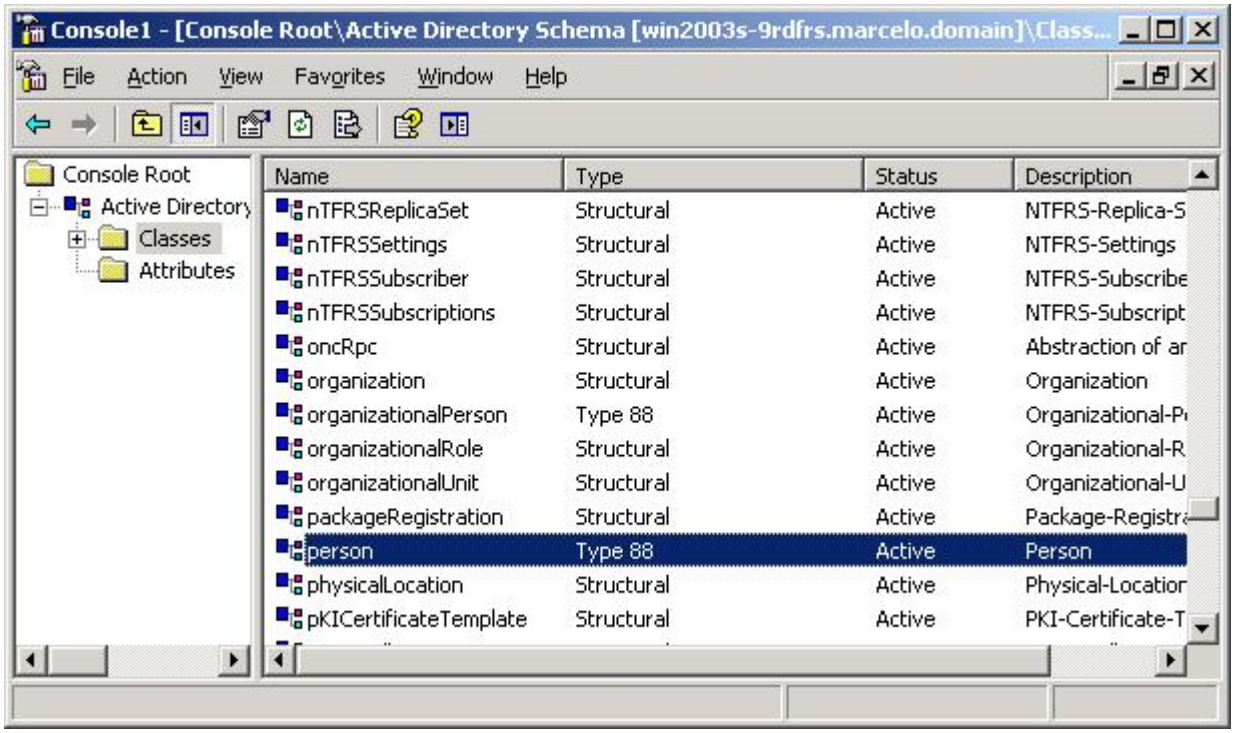



e) Click **OK** to close the dialog.

The proprietarySCADAInfo attribute is added to the list.

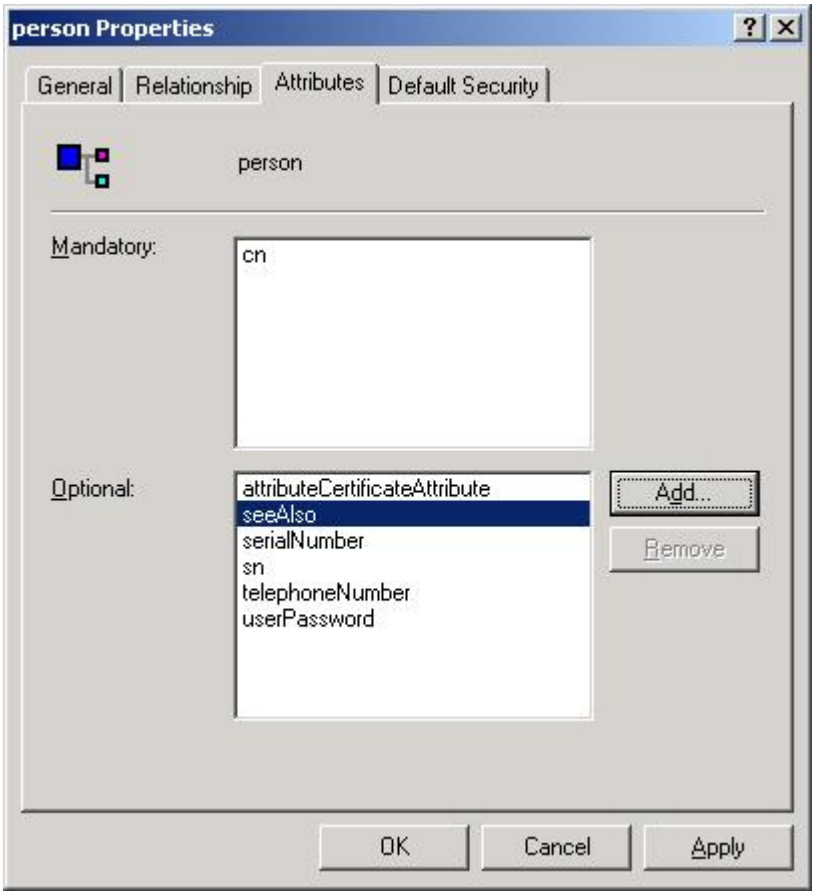
4. Add the proprietarySCADAInfo attribute to the **person** and **group** classes.

a) In the **Console Root** tree-view, select **Active Directory Schema** > **Classes**



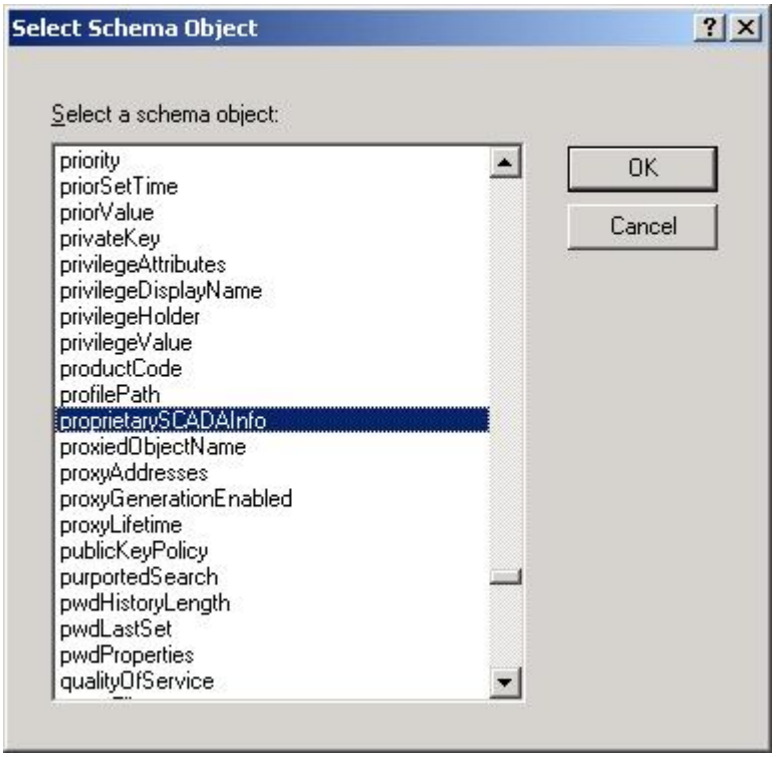

b) In the list of classes, right-click **person**, and then click **Properties** on the shortcut menu.
The *Properties* dialog is displayed.

c) In the dialog, click the **Attributes** tab.




d) Click **Add**.
The *Select Schema Object* dialog is displayed.

e) In the list of schema objects, select **proprietarySCADAInfo**, and then click **OK**.




The attribute is added to the class properties.

f) Click **OK** to close the *Properties* dialog.

g) Repeat steps b through f for the **group** class.

5. In the **Console Root** tree-view, right-click **Active Directory Schema**, and then click **Refresh** on the shortcut menu.
6. Click **File** > **Exit** to close the console window.
7. Restart the server.

## Group Account dialog

The *Group Account* dialog is used to create and delete user groups, as well as to configure the access privileges for a selected group.

### Accessing the dialog

Assuming the project security system has already been enabled (i.e., you have used the security system configuration wizard at least once), you can access this dialog by doing one of the following:

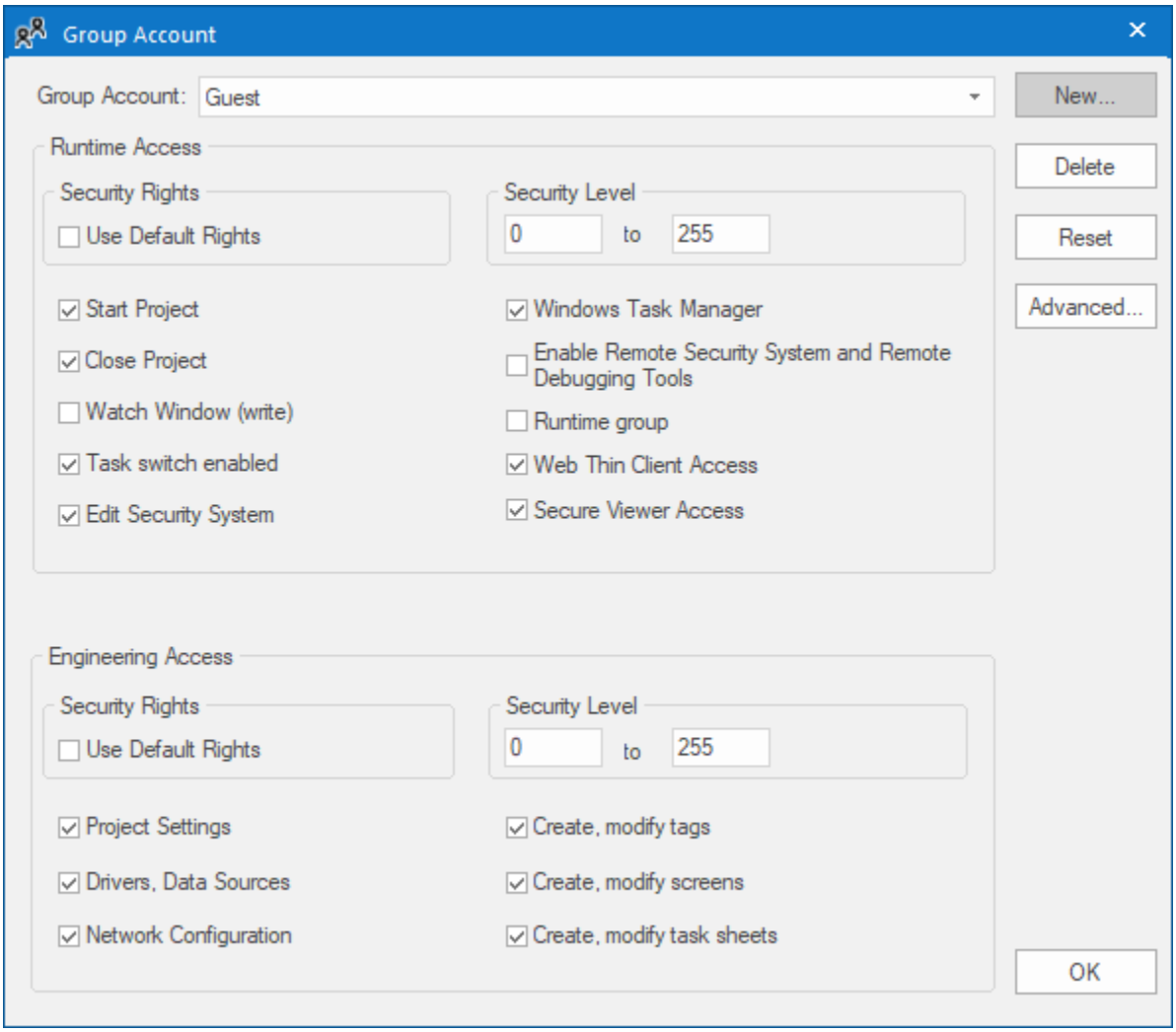
- Open the *Security System* dialog, and then click **Groups**; or
- In the Global tab of the Project Explorer, right-click **Groups** and then click **Groups properties** on the shortcut menu.

### The dialog in detail

Please note that if a user is assigned to more than one group (see Creating and configuring users), the groups' settings may conflict with each other. How the settings are resolved depends on which settings they are:

- The settings in the *Group Account* dialog below are permissive, which means the most permissive setting from all of a user's groups applies to the user. For example, if any of the groups can create and modify tags, the user can create and modify tags.
- The settings in the *Group Account Advanced* dialog (both tabs) are restrictive, which means the most restrictive setting from all of a user's groups applies to the user. For example, if one group has a minimum

password size of 8 and another group has a minimum password size of 12, the user's minimum password size is 12. (For Auto Log Off in particular, **Counting from logon** overrides **Counting from user's last action**.)




***Group Account dialog***

**Group Account**

The user group that you are currently configuring.

> **Note:** There are two default groups for all projects: **Guest** and **(Default Rights)**.

> **Note:** If the security mode is set to **Domain (LDAP)**, please note that the built-in groups in Microsoft Active Directory will not appear in this list of groups and cannot be added to the project.

**New**

Creates a new group. In the *New Group Account* dialog, type the name of the new group and then click **OK**.

**Delete**

Deletes the currently selected group.

**Reset**

Resets the privileges of the currently selected group to match the **(Default Rights)** group.

This does not lock the group to the default; you can make further changes. To lock the group, see **Use Default Rights** below.

**Advanced**

Opens the *Group Account Advanced* dialog (see below).

**Runtime Access**

The specific rights that a member of the group has when they use a project thin client to access your project during run time:

**Security Rights**

Locks the run-time privileges of the currently selected group to those configured for the **(Default Rights)** group. If changes are made to the **(Default Rights)** group, they also apply to this group.

**Security Level**

The range of access levels that this group may access in the project.

**Start Project**

Members of the group may run the project.

**Close Project**

Members of the group may stop the project.

**Watch Window (write)**

Members of the group may write values to the project database using the *Watch* window.

> **Note:** This only applies to projects running locally. For projects running remotely, see **Enable Remote Debugging Tools** below.

**Task switch enabled**

Members of the group may switch away from the project runtime client to another Windows task.

**Edit Security System**

Members of the group may make changes to the project security system during run time.

> **Note:** Be careful not to clear this option for your own group, or you may not be able to undo your own changes.

**Windows Task Manager**

Members of the group may open the Windows Task Manager.

> **Note:** Clearing this option means disabling the Task Manager during run time, which requires Administrator privileges. You will need to run the client as an administrator.

**Enable Remote Security System and Remote Debugging Tools**

Members of the group may:

- Configure the security system in another project (running in **Distributed - Client** mode) in order to use the settings in the current project (running in **Distributed - Server** mode);

- Configure a TCP/IP Client worksheet in another project in order to connect it to the current project; and
- Use Watch and LogWin tools to debug the project while it is running on remote station.

**Runtime group**

If this option is selected, the group will be available for new users created during run time. For example, if the `CreateUser` function is called during run time in order to create a new user, the new user can be assigned only to groups that have this option selected. This can prevent an existing user with limited privileges from creating a new user and then assigning it to a group with more privileges.

**Secure Viewer Access**

Members of the group may use Secure Viewer to connect to the project runtime server.

This option does not affect the user's ability to use Mobile Access.

**Engineering Access**

The specific rights that a member of the group has when they use the project development software to open and edit your project:

**Security Rights**

Locks the development privileges of the currently selected group to those configured for the **(Default Rights)** group. If changes are made to the **(Default Rights)** group, they also apply to this group.

**Security Level**

The range of access levels that this group may access in the development application.

**Project Settings**

Members of the group may modify the project settings and the Mobile Access configuration .

**Drivers, Data Sources**

Members of the group may create, modify device drivers and external data sources.

**Network Configuration**

Members of the group may create, modify TCP/IP Client worksheets.

**Create, modify tags**

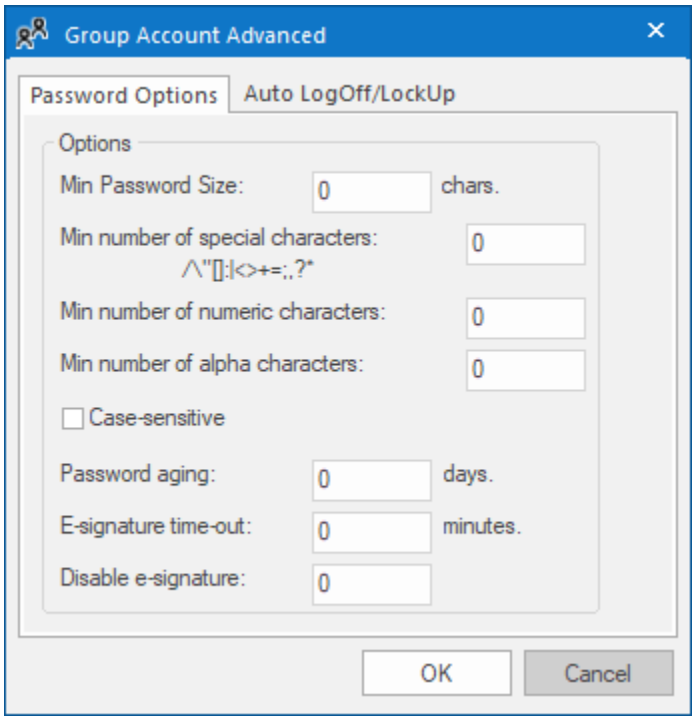
Members of the group may create, modify project tags.

**Create, modify screens**

Members of the group may create, modify project screens.

**Create, modify task sheets**

Members of the group may create, modify task worksheets.




***Advanced dialog – Password Options***

**Min password size**
**Min number of special characters**
**Min number of numeric characters**
**Min number of alpha characters**

To make user passwords more complex and therefore more secure, you can require that they contain a certain number of alpha (A-Z, a-z), numeric (0-9), and special (punctuation) characters. When the user is prompted to change their password — for example, when their old password expires (see **Password aging** below) — the new password will not be accepted unless it meets these requirements.

**Case-sensitive**

If this option is selected, passwords are case sensitive — that is, passwords created with both upper and lowercase characters must be entered the same way by the user.

**Password aging**

The number of days that a password can be used before it expires. When a user's password expires, that user will be forced to change it: when they try to log on to the project, the *Change Password* dialog box will be automatically displayed and the user cannot complete the logon process until they provide a new password.

This setting applies to all users in the group, although the actual aging is counted separately for each user. The aging is restarted after the password is changed, either by the *Change Password* dialog box as described above or by the `SetPassword` function.

By default, the user must choose a new password that is different from the old password. To disable this requirement, so that users can re-use the same passwords, use a text editor to manually edit your project file (`<project name>.app`) to include the following setting:

```
[Security]
```

```
ChangePasswordMode=1
```

To make passwords never expire, set **Password aging** to `0`.

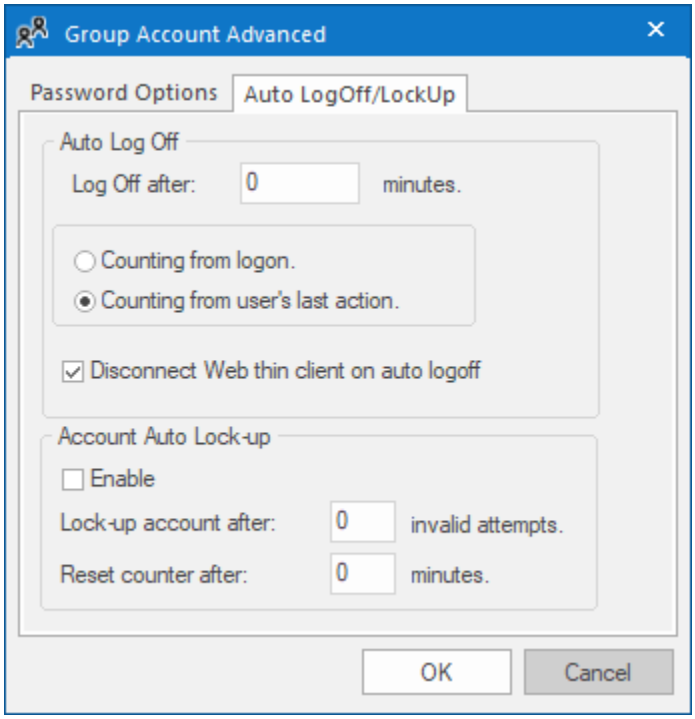
#### E-signature time-out

Timeout period (in minutes) of the *E-Sign* dialog box for all users in this group. The user must enter their user name and password before the specified timeout to use project features that require an e-signature.

#### Disable e-signature

When the value in this box is TRUE (non-zero), users in this group can ignore the e-signature requirement on any screen objects that have the **E-Sign** option selected and on any scripts that call the `CheckESign` function. Using the object or triggering the script still sends an event to the Event Log, but it is automatically signed on behalf of the user.

To change this setting during run time, type the name of a project tag (e.g., `DisableESign`). The value of the tag will be used.




***Advanced dialog – Auto LogOff/LockUp***

#### Auto Log Off

##### Log Off after

Number of minutes after which the current user must be logged off automatically. If this field is left in blank (or with the value 0), the current user is never logged off automatically.

##### Counting from logon

When this option is selected, the current user is automatically logged off after the period of time configured in the **Log Off after** field elapsed since when the current user was logged on the system.

##### Counting from user's last action

When this option is selected, the current user is automatically logged off after the period of time configured in the **Log Off after** field elapsed since the last action (mouse, touchscreen, or keyboard action) was performed by the current user.

**Auto Lock-up**

**Enable**

Enables the auto lock-up features described below.

**Lock up account after**

Maximum number of times a user can try to log on to an account. If the user exceeds the specified maximum number of attempts (provides an invalid password) within the period of time specified in the **Reset counter after** field, the project will automatically block the user.

**Reset counter after**

Defines how long after an invalid log-on attempt the project will wait (in minutes) until it resets the log-on attempts counter.

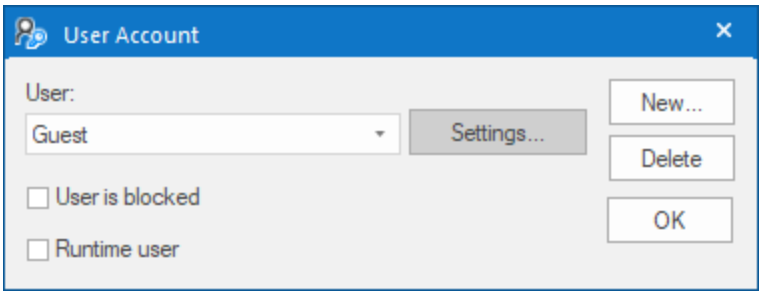


**Note:** If VBScript debugging is enabled, the **Auto Log Off** feature cannot be used; the normal execution cycle is suspended during debugging, so it is not possible to accurately measure the time elapsed without user input. For more information, see Debugging VBScript on page 1246.

## Creating and configuring users

To create and maintain accounts for project users, click the **Users** button on the *Security System* dialog. (Alternately, to configure a user, open the *Users* folder located in the *Security* folder.)

The *User Account* dialog displays.




***User Account dialog***

After the project initializes, if no users log on (or when the current user logs off), then the project automatically logs on the default user (**Guest**). In addition to the default **Guest** user, there is a **Guest** group, which has default privileges that enable all tasks. We recommend that you evaluate and edit the **Guest** group's privileges to specify a minimal amount of privileges for the start up procedure.

To create a new user, click **New** to open *the New User Account* dialog.

To delete a user, click the **User** combo-box button, select the user name from list, and then click **Delete**.

To configure a user, use the following procedure:

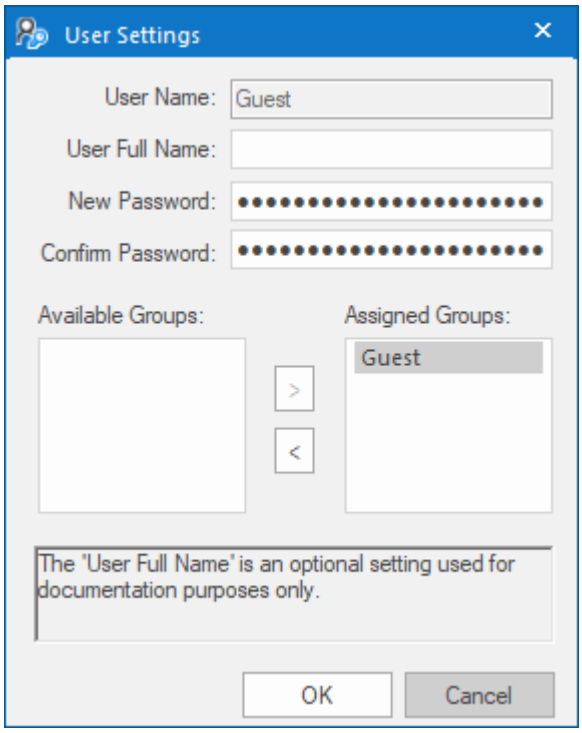
1. Click the **User** combo-box button and then select a user from the list.
2. To block the user from logging onto the project at all, select **User is blocked**. This allows you to disable a user account without deleting it.

   > **Tip:** To select and clear this option during run time, use the `BlockUser` and `UnblockUser` functions, respectively.

3. To flag the user so that it can be affected during run time by any of the Security functions, select **Runtime user**. If this option is not selected, the user cannot be changed, blocked, or removed except through this dialog. This allows you to protect certain users.

   Users that are actually created during run time — for example, by calling the `CreateUser` function — have this option automatically selected.

4. Click the new **Settings** button to open the *User Settings* dialog:




***User Settings dialog***

5. Configure the parameters on this dialog as follows:
   - **User Full Name** text box (*optional*): Type the user's full name.
   - **New Password** text box: Type the user's password.
   - **Confirm Password** text box: Re-type the user's password.

> **Tip:** In most cases, user names and passwords can include spaces. However, if you plan to enable Mobile Access for your project, make sure the user names and passwords do not include spaces. For more information, see Link directly to a project screen or screen group on page 801.

6. In the **Available Groups** list, select the group(s) to which the user should be assigned, and then click **>** to move those group(s) to the **Assigned Groups** list.
7. When you are finished, click **OK** to apply the changes and close the *Settings* dialog.

## Security System dialog

The main *Security System* dialog is used to manage the project security system after it has been initially configured.

### Accessing the dialog

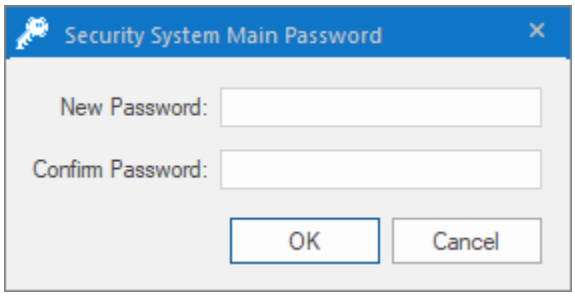
Assuming the project security system has already been enabled (i.e., you have used the security system configuration wizard at least once), then you can access this dialog by doing one of the following:

- On the Project tab of the ribbon, in the Security group, click **Configure**; or
- In the Global tab of the Project Explorer, right-click **Security** and then click **Settings** on the shortcut menu.

If you do either of these *before* the security system has been enabled, then the security system configuration wizard will open automatically.

If you've already configured the security system and set a main password, then you'll be prompted to enter it.

### The dialog in detail




***Security System dialog***

| Area / Element | | Description |
|---|---|---|
| **Enable Security System** | | Indicates whether the project security system is currently enabled. If it is, then the users and groups' specified access privileges are enforced. |
| **Main Password** | | Opens a dialog where you can specify a main administrative password for the entire project. |
| **Security Mode** | Mode | The current security mode of the project. |
| | **Server Settings** | Opens the *Server Settings* dialog, where you can configure the server settings for **Distributed – Client** or **Domain (LDAP)**. |
| **Run Wizard** | | Opens the security system configuration wizard. |
| **Backup** | | Opens the *Import/Export* dialog, where you can export or import the security system configuration. |
| **Accounts Management** | **Groups** | Opens the *Group Account* dialog, where you can create and configure groups. |
| | **Users** | Opens the *User Account* dialog, where you can create and configure users. |
| | **Display list of users at logon** | Displays a list of available users (in the *Log On* dialog) when a user is prompted to log on. The user may select from this list rather than type his user name.<br>If **Security Mode** is Domain (LDAP) and the offline cache is enabled, only the currently cached users will be displayed. |
| | **Log On on E-Signature** | Forces a user to log on with their own user account when they're prompted to e-sign an event. If this is not selected, then the current user account remains logged on regardless of who e-signs the event. |
| | **Default User** | This user is automatically logged on when no other user is logged on, such as when the previous user times out or manually logs off. |

| Area / Element | Description |
| --- | --- |
| | **Note:** This user's privileges should be heavily restricted, to prevent your project from being left vulnerable. |
| **Virtual Keyboard** | The type of virtual keyboard that is displayed on the client when the user is prompted to log on. |

## Backing up the security system configuration

You can back up your project's security system configuration by exporting it to a file. You can also import a configuration either from a file or from another runtime project.

### Exporting the configuration to a file

To export the security system configuration:

1. In the main *Security System* dialog, click **Backup**. The *Import/Export* dialog is displayed.
2. Click **Export to file**. A standard *Save As* dialog is displayed.
3. Specify a file name and location for the file, and then click **OK**.

The exported file is encrypted, using the main password configured in the *Security System* dialog.

> **Tip:** You can also export the configuration during runtime by calling the `ExportSeruritySystem` function.

### Importing the configuration from a file

If your project's security mode is set to **Local Only**, then you can import a configuration from a previously exported file.

To import the security system configuration:

1. In the main *Security System* dialog, click **Backup**. The *Import/Export* dialog is displayed.
2. Click **Import from file**. A standard *Open* dialog is displayed.
3. Locate the configuration file (`*.dat`) that you want to import, and then click **OK**. You will be prompted for the configuration's main password.
4. Type the password, and then click **OK**. The *Import from File* dialog is displayed.
5. Select an import method:
   - **Import only settings that do not conflict**: Merge the imported settings with the current project settings. In the case of conflicts, keep the current settings.
   - **Import all settings and replace conflicts**: Merge the imported settings with the project settings. In the case of conflicts, use the imported settings.
   - **Replace the current settings**: Completely replace the current project settings with the settings imported from the file.
6. Click **OK**.

> **Tip:** You can also import the configuration during runtime by calling the `ImportSecuritySystem` function.

### Importing the configuration from another project

If your project's security mode is set to **Distributed – Server**, then you can import a configuration from another project if:

- The other project's security mode is set to **Distributed – Client**, and its server settings are configured to use your project as the server; and
- The other project is currently running on the same network.

To import the security system configuration:

1. In the main *Security System* dialog, click **Backup**. The *Import/Export* dialog is displayed.
2. Click **Import from client station**. The **Import Security from Client Station** dialog is displayed.

The dialog shows a list of runtime projects that are using your project as their security system server. Each project/client listing includes a time stamp that shows when it last cached the security system configuration.

3. Select a client station, and then click **Import from client**. You will be prompted for the configuration's main password.
4. Type the password, and then click **OK**.

## Logging on/off

If the project security system has been enabled and the default "Guest" user's privileges have been restricted, then you must log on to fully use the development application and/or the runtime project.

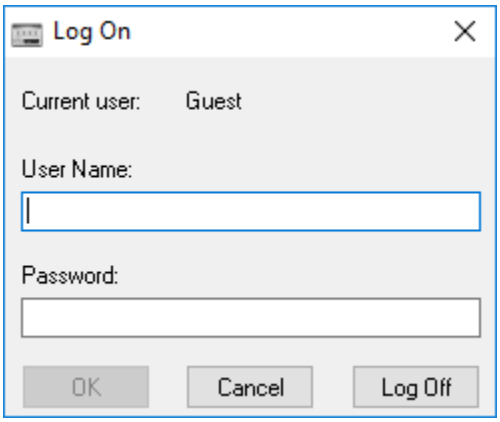
> **Note:** The project security system must be enabled before you can use this feature.

To log on to the development application, click **Log On** on the Project tab of the ribbon.

To prompt a user to log on to the runtime project, do one of the following:

- Call the `LogOn` function somewhere that an expression can be configured — for example, draw a Button object in a screen and then apply the Command animation to it, so that pressing the button shows a logon prompt; or
- Select the **Log On on E-Signature** option (in the main *Security System* dialog), which forces the user to log on whenever he performs some action that requires an e-signature.

In either the development application or the runtime project, the Log On dialog is displayed:




***Log On dialog***

Use this dialog as follows:

- To log on as yourself, type your user name and password in the appropriate boxes and then click **OK**.
- To log on as the default "Guest" user, type `guest` in the **User Name** box and then click **OK**.

  > **Note:** By default, "Guest" has no password, so you can leave the **Password** box empty. However, if you've changed the password or you're getting your security settings from a server (either Distributed or Domain), then you will need to enter a password for "Guest."

- To log off, simply click **Log Off**. The default user (typically "Guest," but this may be changed in the main *Security System* dialog) is automatically logged on to replace you.

> **Note:** If the security mode is set to **Domain (LDAP)** and a user created on the LDAP server is required to change his password the first time he logs onto the domain, then he must do that before he will be able to log onto the EcoStruxure Machine SCADA Expert project.

## Blocking or unblocking a user

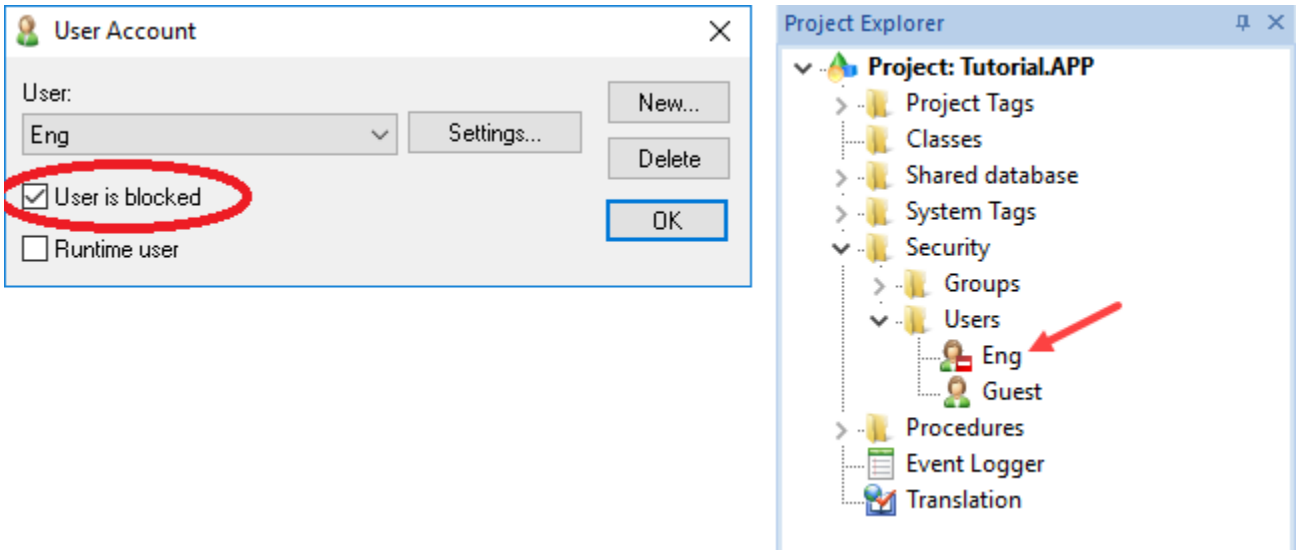
An individual user in the project security system may be completely blocked from accessing the project, and a blocked may subsequently be unblocked.

A user may be blocked in the following ways:

- By manually selecting the **User is blocked** option in the *User Account* dialog;
- By calling the `BlockUser` function during runtime; or
- Automatically if the user enters the wrong password too many times. (The number of attempts allowed is configured in the *Group Account* dialog.)

To check whether a user is blocked, do one of the following:

- Look at their user icon in the Project Explorer, which will be marked with a red circle; or
- Call the `GetUserState` function during runtime.




***User is blocked***

To unblock a blocked user, do one of the following:

- Clear the **User is blocked** option in the *User Account dialog*; or
- Call the `UnblockUser` function during runtime.

## Password-protecting screens, symbols, and worksheets

Screens, symbols, and worksheets in the Project Explorer can be password-protected. You can assign individual passwords to each file, or you can assign a single password to all files in the project

Almost all project files are encrypted as a matter of course, to prevent unauthorized analysis by third-party tools. (Screen files are not encrypted, because decrypting them during runtime would decrease performance.) However, you can take the extra step of password-protecting your files to prevent unauthorized changes or re-use by other EcoStruxure Machine SCADA Expert project developers.

> **Note:** These passwords are always case sensitive.

### Assigning a password to a single file

To assign a password to a single project file:

1. In the Project Explorer, find and right-click the desired file, and then click **Password Protection** on the shortcut menu. The *Edit Protection* dialog is displayed.
2. Type the new password, and then type it again to confirm.
3. Click **OK** to close the dialog.

The file is now protected. The next time you try to open it, you will be prompted for the password.

### Clearing the password from a single file

To clear a password from a single project file:

1. In the Project Explorer, find and double-click the desired file to open it. You will be prompted for the password.
2. With the file open for editing, right-click the file in the Project Explorer and then click **Password Protection** on the shortcut menu. The *Edit Protection* dialog is displayed.
3. Leave the **New password** and **Confirm password** boxes empty.
4. Click **OK** to close the dialog.

The file is no longer protected. You can open the file without being prompted for the password.

### Assigning a password to all files

To assign a single password to all files in your project:

1. On the Home tab of the ribbon, in the Tools group, click **Verify**. The *Verify Project* dialog is displayed.
2. Click **Set password for all files**. The *Edit Protection* dialog is displayed.
3. Type the current password for your project, if any.
4. Type the new password, and then type it again to confirm.
5. Click **OK**. The verification routine proceeds.
6. Click **Close** to close the *Verify Project* dialog.

All files in your project are now protected. The next time you try to open one, you will be prompted for the password.

### Clearing the password from all files

To assign a single password to all files in your project:

1. On the Home tab of the ribbon, in the Tools group, click **Verify**. The *Verify Project* dialog is displayed.
2. Click **Set password for all files**. The *Edit Protection* dialog is displayed.
3. Type the current password for your project.
4. Leave the **New password** and **Confirm password** boxes empty.

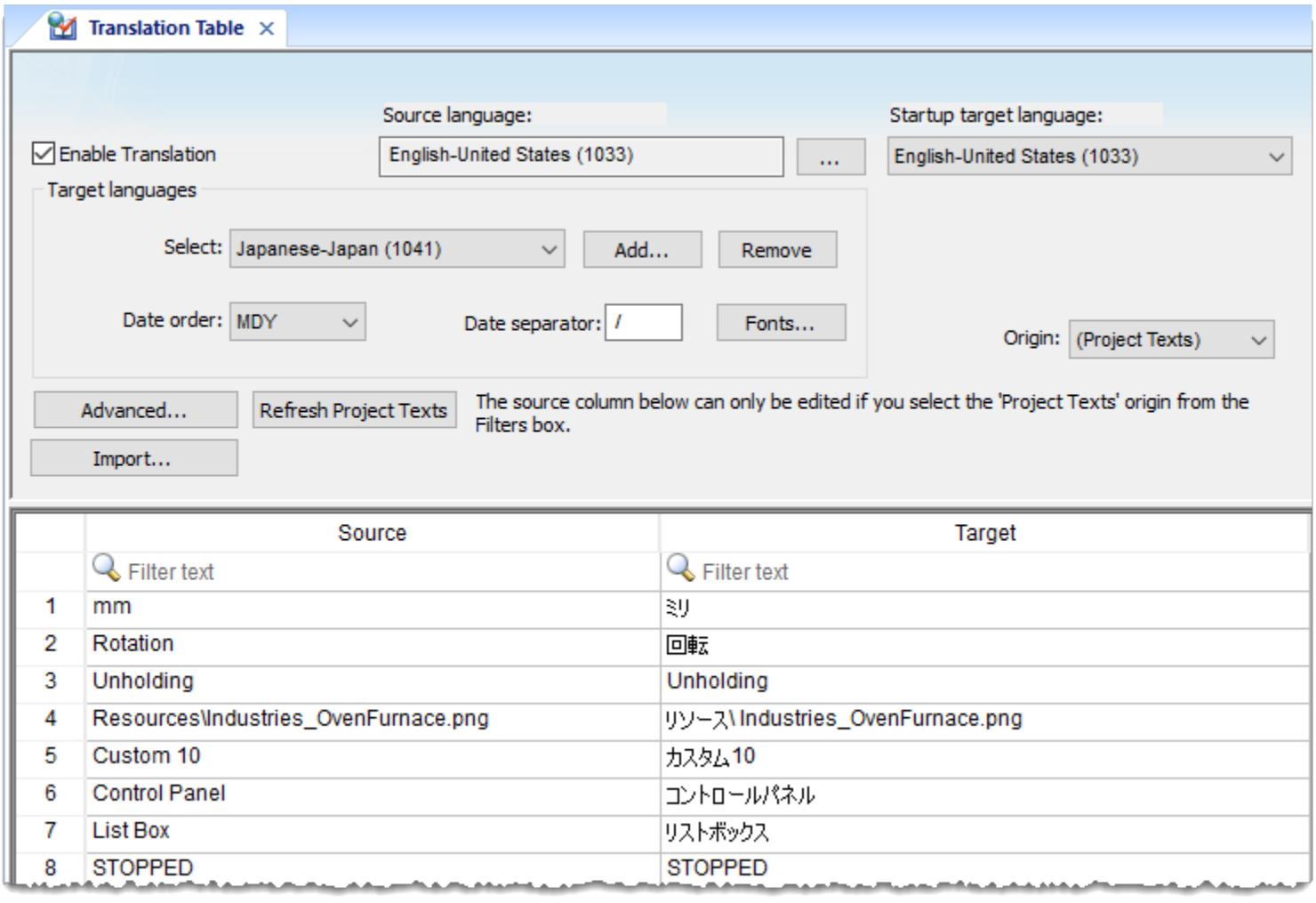
5. Click **OK**. The verification routine proceeds.
6. Click **Close** to close the *Verify Project* dialog.

Your project files are no longer protected.

# Project Localization

You can quickly translate your project's user interface to multiple languages, using either machine translation (e.g., Google Translate) or a human translator, and then you can switch your project's language during runtime with a simple function call.

The Translation Table is a worksheet that you can use to create a multilingual user interface (MUI) for your project. (This is different from changing the language of the development environment itself; that is done with the **Language** command on the **View** tab of the ribbon.) The worksheet is divided into a Source column, which contains original pieces of text from your project screens, and a Target column, which contains the translated equivalents of the items in the Source column.




**Note:**

Google has made the Google Translate API a paid service, so automatic translation of project texts is not available. You can still use the Google Translate website to translate project texts, but doing so requires additional steps.

It is our goal to ensure that the functionality of EcoStruxure Machine SCADA Expert 2020 continues to evolve and grow around emerging technology. We are pursuing alternatives for automatic translation, and we hope to offer this feature again in the near future.

## Add a target language to the Translation Table

The Translation Table is used to manage the languages into which you want to translate your project. Adding a language to the table can be as simple as selecting it from a list and then automatically translating your project texts.

> **Note:** By default, the source language of a project is the language of the development environment itself. In other words, it is assumed that when you create a new project, it is for the same language that you work in. For more information about changing the language of the development environment, see #unique_450.
>
> In some cases, however, you might work in one language but develop your project for another — for example, you might work in Portuguese but develop your project for English. If that is the case, you must remember to set the source language in Step 3 below, to associate the correct language with the **Source** column of the worksheet.

To add a target language to the Translation Table:

1. Open the Translation Table worksheet by doing one of the following:
   - On the **Insert** tab of the ribbon, in the **Global** group, click **Translation**; or
   - In the **Global** tab of the Project Explorer, double-click **Translation**.

   The Translation Table worksheet is opened for editing.
2. Make sure the **Enable Translation** option is selected.
   If this option is cleared, the entire Translation Table worksheet is disabled and the language cannot be changed during run time.
3. If the source language of your project is other than the language of the development environment itself, set the source language:
   a) To the right of the **Source language** box, click the browse button.
   The *Languages* dialog box is displayed.
   b) In the *Languages* dialog box, select the language for which your project was originally developed, and then click **OK**.

   The selected language is set in the **Source language** box.
4. Set the target language:
   a) In the **Target languages** group, click **Add**.
   The *Languages* dialog box is displayed.
   b) In the *Languages* dialog box, select the language to which you want to translate your project, and then click **OK**.

   The selected language is added to the **Select** list, and a new worksheet is created for the language. The **Source** column of the worksheet is automatically populated with all of the translatable text strings in your project, and the **Target** column is blank.
5. Configure **Date order** and **Date separator** as needed for the target language.
   For example, for `English-United States` (i.e., American English), **Date order** is typically `MDY` and **Date separator** is typically `/`, resulting in a date format of `MM/DD/YYYY`. In many European languages, however, **Date order** is typically `YMD` and **Date separator** is typically `.`, resulting in a date format of `YYYY.MM.DD`.
   For more information, see About the date format and how to change it on page 694.
   The language is added to the **Select** list, and a new worksheet is created for the language. The **Source** column of the worksheet is automatically populated with all of the translatable text strings in your project, and the **Target** column is blank.
6. Use the **Filters** to search the worksheet for specific text items; as you type a few characters, the list is dynamically filtered to show only the items that match.

> **Note:** The ampersand character (&) is ignored when filtering rows. This is to improve the handling of text in program dialogs, where ampersands are used to indicate keyboard accelerators.

7. For each text item in the **Source** column, enter the translation in the **Target** column.

   You can manually translate the items one by one, or you can use a translation service such as Google Translate to automatically translate multiple items.

   a) Highlight the cells in the **Source** column to select them, and then press **Ctrl+C** to copy those items to the clipboard.

      You can also click the column header to select the entire column.

   b) In your web browser, go to: translate.google.com

   c) Select the **From** and **To** languages.

   d) Click in the text box on the left, and then press **Ctrl+V** to paste the items from the clipboard.
   The items are pasted as separate lines.

   e) Click **Translate**.
   The translation appears on the right. Again, the items are displayed as separate lines.

   f) Highlight the translated items to select them, and then press **Ctrl+C** to copy the items to the clipboard.

   g) In EcoStruxure Machine SCADA Expert 2020, highlight the empty cells in the **Target** column to select them.

      You can also click the column header to select the entire column.

   h) Press **Ctrl+V** to paste the translated items from the clipboard into the **Target** column.

      Because the translated items were copied as separate lines, they should be correctly pasted into the rows of the worksheet.

8. Review the translated items in the **Target** column in order to confirm the translations and make sure they corespond with the untranslated items in the **Source** column.

   If you want to keep the original, untranslated text for a specific item, leave the **Target** column blank.

9. Save and close the worksheet.

The translation table is saved as a tab-separated text file in your project folder at: `<project name>\Web\Translation.trn`

You can use a spreadsheet application such as Microsoft Excel to open and directly edit this file, if you want to do so.

Changes made to the translation table will not take effect until you either call the `SetLanguage` function during run time or restart the project runtime server itself.

## Configure fonts for a target language

Configure the fonts for a target language so that translated text items are displayed in another font, size, and/or style that is more appropriate to the language.

Before you begin this task, you should have already added one target language to your project's Translation Table.

By default, when you change your project's runtime appearance to one of the target languages in the Translation Table, the translated text items are still displayed in their original font, size, and style. This can cause many different problems, depending on the target language. For example, if the target language uses a non-Roman character set, or it typically has longer or shorter words, or it is read right-to-left instead of left-to-right, then the screen objects with translated text items may not be displayed correctly.

You can use the Font Configuration tool on the Translation Table worksheet to configure new fonts, sizes, and styles for the translated text items. When you set your project's interface to the target language, the translated text items will be changed to the new configuration. The configuration can be either very general or very specific — just add more rows for more specific changes.

If you have two or more rows that conflict with each other — for example, if one row applies to all objects and another row applies only to Button objects — then the row that most specifically matches a given text item is the one that is applied.

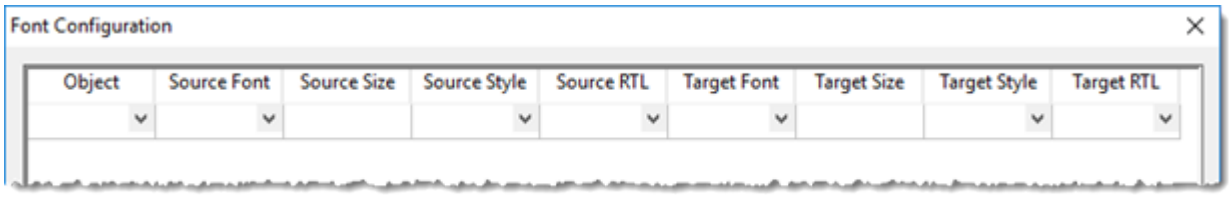
For more information, see "Examples" below.

To configure fonts:

1. Open the Translation Table worksheet by doing one of the following.
   - On the Insert tab of the ribbon, in the Global group, click **Translation**; or
   - In the Global tab of the Project Explorer, double-click **Translation**.

   The Translation Table worksheet is opened for editing.
2. In the **Target languages** area, select the language for which you want to configure fonts, and then click **Fonts**.
   This font configuration applies only to the selected target language.

   The *Font Configuration* dialog box is displayed.




3. In the **Object** column, select the type of screen objects that the configuration will apply to. To have it apply to all types, select `(All)`.
4. In the **Source Font** column, select the font that the configuration will apply to. To have it apply to all fonts, select `(All)`.
   The list of selectable fonts includes all fonts that are installed on your computer, not just the fonts that are used in your project.
5. In the **Source Size** column, type the font size (in points) that the configuration will apply to. To have it apply to all sizes, type `(All)`.
6. In the **Source Style** column, select the font styles that the configuration will apply to. To have it apply to all styles, select `(All)`.
7. In the **Source RTL** column, select the reading direction (left-to-right or right-to-left) that the configuration will apply to. To have it apply to both, select `(All)`.
8. In the **Target Font** column, select the font to which the **Source Font** should be changed. If you want to leave the **Source Font** unchanged, select `*`.

The list of selectable fonts includes all fonts that are installed on your computer.

Make sure the font you select is also installed on any computers or devices that will run as project runtime clients. If you cannot do this — in other words, if you do not control all of the potential clients — then select a "standard" font that is commonly installed on all platforms.

9. In the **Target Size** column, type the font size (in either percentage or points) to which the **Source Size** should be changed. If you want to leave the **Source Size** unchanged, type `*`.

   Keep in mind that percentage size is relative while point size is absolute. If you repeatedly change the same text item to new percentage sizes, those changes will be cumulative. For example, if you change the font size of a text item to 80%, then to 100%, then to 80% again, the end result will be 64% of the original size. Therefore, if you plan to switch back and forth between languages, you should specify point sizes instead of percentage sizes.

10. In the **Target Style** column, select the font style to which the **Source Style** should be changed. If you want to leave the **Source Style** unchanged, select `*`.

11. In the **Target RTL** column, select the reading direction (left-to-right or right-to-left) to which the **Source RTL** should be changed. If you want to leave the **Source RTL** unchanged, select `*`.

12. Repeat this procedure for additional rows, if necessary.

13. Click **OK** to save the configuration and close the dialog box.

When you set your project's interface to the target language, the translated text items will be changed to the new configuration.

## *Examples of font configuration*

The Font Configuration tool included in the Translation Table worksheet

### Example #1

You are using ten different fonts in your project and they all have different font sizes. You have added Chinese as a target language in your Translation Table, so when you set it as the language for your project interface, you also want to convert all text items to a font that supports the Chinese character set.

You would create the following configuration using the Font Configuration tool:

| Object | Source Font | Source Size | Source Style | Source RTL | Target Font | Target Size | Target Style | Target RTL |
|---|---|---|---|---|---|---|---|---|
| (All) | (All) | (All) | (All) | (All) | Arial Unicode MS | * | * | * |

The `*` configured in the **Target** columns means that the original settings should be kept. In other words, if you have two texts with different sizes, then the font type will be changed to Arial Unicode MS (a general-purpose font that includes all Unicode character sets, including Chinese) but the font sizes will not be changed.

### Example #2

You are using a single font type in your project, but it has different font sizes depending on where and how it is used. You have added German as a target language in your Translation Table; words tend to be much longer in German, so you when you set it as the language for your project interface, you also want to decrease the font sizes to make sure the translated text items fit within their screen objects.

You would create the following configuration using the Font Configuration tool:

| Object | Source Font | Source Size | Source Style | Source RTL | Target Font | Target Size | Target Style | Target RTL |
|---|---|---|---|---|---|---|---|---|
| (All) | (All) | (All) | (All) | (All) | * | 80% | * | * |

Under this configuration, it does not matter how many fonts you are using or what their specific sizes are. They will all be proportionately decreased to 80% of their original sizes.

### Example #3

You have added Arabic as a target language in your Translation Table, but you have also decided not to translate your button labels. Therefore, you want to convert all objects except buttons to a font that supports the Arabic character set, and you also want to change the reading direction from left-to-right (default) to right-to-left.

You would create the following configuration using the Font Configuration tool:

| Object | Source Font | Source Size | Source Style | Source RTL | Target Font | Target Size | Target Style | Target RTL |
|---|---|---|---|---|---|---|---|---|
| (All) | (All) | (All) | (All) | (All) | Arial Unicode MS | * | * | Enabled |
| Button | (All) | (All) | (All) | (All) | * | * | * | * |

Even though it would seem like both rows of the configuration would apply to Button objects, only the second row is actually applied because it most specifically matches.

## Set the project's language at startup

Even when you have multiple languages configured for your project, you must still specify which language you want your project to start in at runtime.

This procedure assumes that you have already added at least one target language to the Translation Table. For more information, see Add a target language to the Translation Table on page 682.

To set the language at startup:

1. Open the Translation Table worksheet by doing one of the following.
   - On the **Insert** tab of the ribbon, in the **Global** group, click **Translation**; or
   - In the **Global** tab of the Project Explorer, double-click **Translation**.

   The Translation Table worksheet is opened for editing.
2. In the **Startup target language** list, select the language in which you want your project to start.
   The list of available languages includes the source language in which you developed the project and any target languages that you have added.
3. Save and close the worksheet.

> **Note:** Setting your project's language will also automatically set the language of any Virtual Keyboards displayed in project screens, as long as there is a VK initialization file for the selected language. For more information, see Data input in screens on Thin Clients on page 345.

## Set the project's language during run time

You can set your project's language during run time by using the `SetLanguage` function anywhere that an expression can be configured.

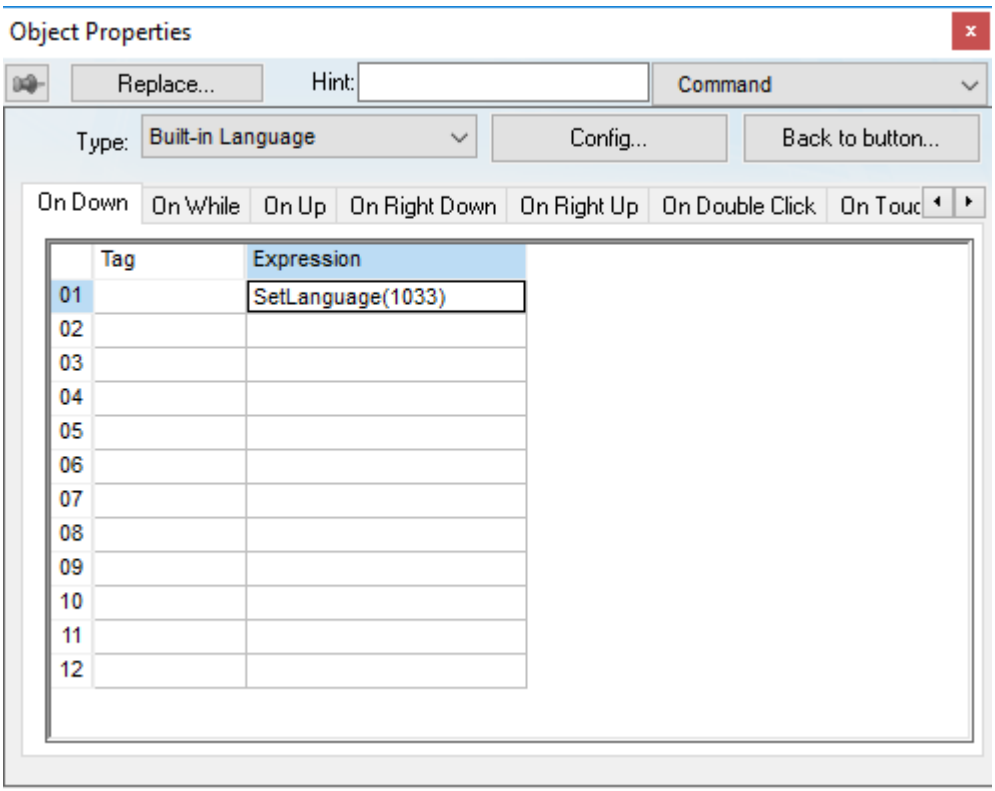
Before you begin this task, you must have already added at least one target language to the Translation Table worksheet. For more information, see Add a target language to the Translation Table on page 682.

The `SetLanguage` function takes one parameter: the ID number of the the target language. Each language's ID number is shown in parentheses next to that language in the Translation Table worksheet — for example, "English-United States (1033)".

> **Note:** LCID values were the proprietary language/region codes used in Microsoft Windows up until Windows 8. Microsoft changed how it handles language identifiers in Windows 10, which effectively deprecated LCID in the operating system, but we hardcoded the values into this software and we continue to use them in order to maintain compatibility across all platforms and with existing projects.

The following example shows how to draw two Button objects that switch the project's language between English and French:

1. In the **Graphics** tab of the Project Explorer, double-click a project screen to open it for editing.
2. On the **Draw** tab of the ribbon, in the **Active Objects** group, click **Button**.
3. Draw a Button object in the project screen.
4. Double-click the Button object.
   The *Object Properties* dialog is displayed.
5. In the **Caption** box, type `English`.
6. Click **Command**.
   The Command animation properties are displayed in the dialog.
7. In the first row of the **On Down** tab, in the **Expression** field, type `SetLanguage(1033)`.




8. Close the *Object Properties* dialog.
9. Duplicate the Button object, either by copy-and-paste or by Ctrl-click.
10. Repeat steps 4 through 8, replacing the caption with `French` and the expression with `SetLanguage(1036)`.

11. Save and close the project screen.

During project run time, clicking each button will set the language of the entire project to that language, using the translated text from the Translation Table.

> **Note:** Setting your project's language will also automatically set the language of any Virtual Keyboards displayed in project screens, as long as there is a VK initialization file for the selected language. For more information, see Data input in screens on Thin Clients on page 345.

## Disable translation of selected screen objects

By default, translation is enabled for all screen objects that have text to be translated. However, you can disable translation of selected objects if you need to preserve their original text.

1. Double-click a screen object to open its *Object Properties* dialog.
2. In the dialog, look for the **Enable translation** option.
   If the option is not available in the object's basic properties, then click **Advanced** to access the advanced properties
3. Clear the **Enable translation** option.
4. Close the *Object Properties* dialog.

Once the option is cleared on an object, its text will no longer be translated during project runtime. The text is still added to the translation table and may be processed though the automatic translation engine, along with all other project texts, but the resulting translation will not actually be applied to the object during runtime.

## Configure the advanced translation settings

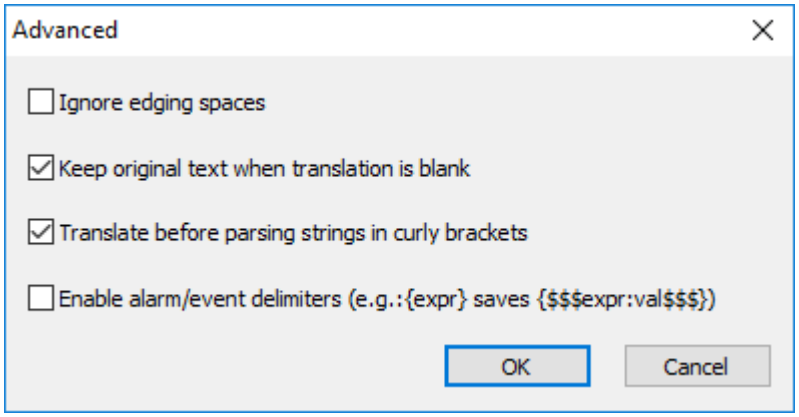
Configure the advanced translation settings as needed.

Before you begin this task, you should have already enabled translation and added at least one target language to your project's Translation Table.

To configure the advanced translation settings:

1. Open the Translation Table worksheet by doing one of the following:
   - On the **Insert** tab of the ribbon, in the **Global** group, click **Translation**; or
   - In the **Global** tab of the Project Explorer, double-click **Translation**.

   The Translation Table worksheet is opened for editing.
2. Click **Advanced**.
   The *Advanced* dialog is displayed.




***Advanced translation settings***

3. Select or clear the options as needed.

| Option | Description |
|---|---|
| **Ignore edging spaces** | Ignore any leading or trailing spaces in strings to be translated. If this option is not selected, then strings must match exactly. |
| **Keep original text when translation is blank** | Display the string in the **Source** column when the **Target** column is empty. If this option is not selected, then the original text may be replaced with blank space. |
| **Translate before parsing strings in curly brackets** | Translate the string before evaluating any tag/expression configured in curly brackets. For example, if the original text is "The current user is {UserName}" and the Portuguese translation is "O usuário atual é {UserName}", then the string is correctly translated before getting the value of **`UserName`**.<br><br>If this option is not selected, then the tag/expression is evaluated before the string is translated. Therefore, if you have not configured a translation that includes the exact value of the tag/expression, then the string will not be translated. |
| **Enable alarm/event delimiters** | Enable the use of delimiters in alarm/event messages, so that when the messages are generated and saved, they are formatted in such a way that they can be correctly translated when retrieved. |

| Option | Description |
| --- | --- |
| | If this option is not selected, then any tag/expression configured in curly brackets is evaluated when the message is generated and saved, before translation. In other words, the message is saved as a literal string, and if you do not configure a translation that includes the exact value of the tag/expression, then the message will not be translated.<br><br>Selecting this option, however, will cause the message to be saved with both the tag/expression and its value at the time the message was generated. For example, if you configure an event message to be "The current user is {UserName}", then it will be saved as "The current user is {$$$UserName:Michael$$$}", which means that the value of **`UserName`** was **`Michael`** at the time the message was generated. Then, when the message is retrieved, it is translated as described in **Translate before parsing strings in curly brackets** above, with the saved value inserted as needed. |

4. Click **OK** to close the dialog, and then save and close the Translation Table.

## Import a legacy translation file into the Translation Table

Due to changes in how the Translation Table works, legacy translation files are not compatible with the latest version of this software. Use the Translation Table worksheet to import a legacy translation file into your project, rather than manually reenter the information from that file.

Before you begin this task, you must have a legacy translation file (i.e., a .tra or .csv file) that was created by an earlier version of this software. You should be able to find the file in the old project folder.

To import a legacy translation file and add it to the current translation:

1. Open the Translation Table worksheet by doing one of the following:
   - On the **Insert** tab of the ribbon, in the **Global** group, click **Translation**; or
   - In the **Global** tab of the Project Explorer, double-click **Translation**.

   The Translation Table worksheet is opened for editing.
2. Make sure the **Enable Translation** option is selected.
   If this option is cleared, then the entire Translation Table worksheet is disabled and the language cannot be changed during run time.
3. In the Translation Table worksheet, click **Import**.
   A standard *Open* dialog box is displayed.
4. Locate and select the legacy translation file (.tra or .csv format), and then click **Open**.

The legacy translation file is imported and added to the current translation.

## About the date format and how to change it

The date format determines how dates are displayed throughout the EcoStruxure Machine SCADA Expert 2020 software, as well as how date strings are handled by the project during run time. You can change the date format, either as part of localizing your EcoStruxure Machine SCADA Expert project or simply to suit your personal preference.

Actual dates and times are stored as complete timestamps, the format of which is hardcoded into the software. Such timestamps can be difficult to read, however, and they often contain additional information that is not immediately relevant to the user. Therefore, when it is necessary to display a date and/or time, the software parses the timestamp, gets the relevant information, and displays it in a user-friendly format. The date format determines what exactly that format is for dates.

Generally speaking, the date format has two configurable settings: the order and the separator. First, the order setting determines the order in which the day (D), month (M), and year (Y) are displayed. Any combination of the three can be used, but some combinations like MDY, YMD, and DMY are more commonly used. Second, the separator setting determines the character that separates the three parts of the date. Again, any single character can be used, but some characters like the forward slash (/), the hyphen (-), and the dot/period (.) are more commonly used.

The order and separator settings together determine the overall date format. For example, EcoStruxure Machine SCADA Expert 2020 uses the MDY order and the forward slash (/) separator by default, and together they produce the American-style format MM/DD/YYYY (e.g, 02/13/2015). Another common date format is the one recommended by ISO 8601; it uses the YMD order and the hyphen (-) separator to produce the format YYYY-MM-DD (e.g., 2015-02-13).

> **Note:** EcoStruxure Machine SCADA Expert 2020 always uses two-digit days and months (i.e., DD, MM) and four-digit years (i.e., YYYY), regardless of the order.

To check the date format at any time, simply reference the **`Date`** system tag. The value of this tag is always the current system date on the local computer, formatted as a string using the current date format. The "local computer" is the computer where the tag is referenced, regardless of whether it is referenced in a background task on the project runtime server or in screen on a project thin client. The server and each client can have its own date format, because the date format can be changed from the default (see "Changing the date format" below).

The easiest way to reference the **`Date`** system tag is to type it into the Watch window on page 71, because the value of the tag is immediately displayed there even when your project is not running. You can use the tag anywhere in your project that accepts a tag or expression, however. For more information about system tags, see System Tags Folder on page 151.

It is important for you to know what the date format is when you are both developing and running your project, because whenever you specify a date as a string, you must do so in the format that the software expects. If you do not, the software will not be able to parse the string correctly and you will see unexpected behavior during project run time. For example, if the software is currently using the format MM/DD/YYYY and you try pass the date string **`"2015-02-13"`** as an argument to a function, the function will return an error code indicating that the specified date is invalid.

### Changing the date format

The default date format for all EcoStruxure Machine SCADA Expert 2020 software components — the project development environment, the project runtime server, and each project thin client — is MM/DD/YYYY. You cannot change this default. Every time you run the software, it automatically uses this default at least to start.

We can and often do change the default date format for customers in other regions around the world that use different formats, but there are special considerations in doing so, such as maintaining backward compatibility with existing projects and not interfering with future software upgrades. That is why we have not made the default a user-configurable setting. If you need to have your default changed, please contact your EcoStruxure Machine SCADA Expert 2020 software distributor.

You can change the date format at any time after you run a project, however, and the project will keep using the specified format until you either change it again or stop the project. When you stop the project, the date format is reset to the default.

There are a few different ways to change the date format.

First, you can call the function `SetDateFormat`. This is the most straightforward way, because you can call it anywhere that you normally use the built-in functions. It is most commonly called in a project's Startup Script, to set the date format on the project runtime server, as well as in the Graphics Script, in the **`Graphics_OnStart`** sub-routine, to set the date format on each project thin client. For more information, see SetDateFormat on page 1148.

Second, if you install Secure Viewer on a computer or device in order to use it as a project thin client, you can specify the date format for that client in Secure Viewer's advanced settings. Doing so should eliminate the need to call the function `SetDateFormat` on that client, as described above, but it will not override the function if it is called. For more information, see Install the Thin Client software on page 41.

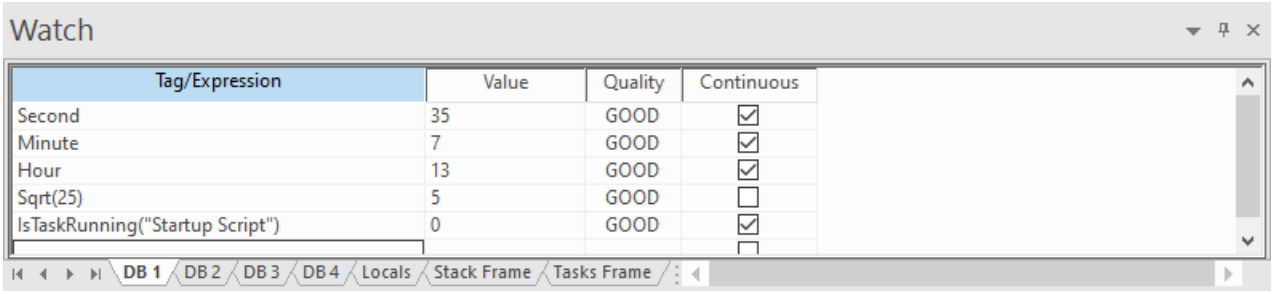
Third, when you use the Translation Table to localize your project for other languages or regions, you can also specify the date format as part of the localization. Then, when you set the language during project run time, the date format will be changed with everything else. For more information, see Project Localization on page 681.

Please keep in mind that the project runtime server and each project thin client — even the client on the same computer as the server, because it runs in a separate processing thread — can have its own date format, depending on how you design and run your project. This can be useful in some cases, such as when your server is located in one region while your clients are located in another. You might even allow users to change the date format on their clients just to suit their personal preferences. But whatever you do, you must thoroughly test your project to ensure that changing the date format does not cause other issues.

# Debugging Tools

## Watch window

The *Watch* window is a debugging tool that lets you: watch and force values to project tags; execute and test functions; and execute and test math expressions.




***Example of the Watch window***

The *Watch* window contains the following elements:

- For each item that you want to watch during project run time:
  - **Tag/Expression**: Specify a project tag, system tag, or expression that you want to watch.
  - **Value**: Displays the value returned by the tag/expression.
  - **Quality**: Displays the quality (GOOD or BAD) of the value returned by the tag/expression.
  - **Continuous**: Select this option to have the project continuously evaluate the tag/expression.
- **DB** tabs: You can use these tabs to organize the items you are watching, so that you do not need to scroll through one long list of items.
- **Locals**, **Stack Frame**, and **Tasks Frame** tabs: These tabs are used to debug VBScript.
- Scroll bars: Use to view areas of the *Watch* window that are obscured from view because of the window size or the size of the current sheet.

> **Tip:** The *Watch* window is dockable, which means you can move it to another location in the project development environment. Click on the titlebar and drag it to a new location. Release the mouse button to attach or dock the window to its new location.

### *Using the Watch tool*

This software includes a standalone Watch tool, separate from the project development environment, that you can use to connect to a running project and then monitor that project's database.

The Watch tool is functionally similar to the Watch window on page 71 in the project development environment, but it differs in several important ways:

- The Watch tool can connect to projects running on both the local computer and remote computers, while the Watch window only shows the project that is currently open in the project development environment;
- You can open multiple instances of the Watch tool in order to connect to different projects at the same time; and
- The Watch tool runs separate from the project development environment, and it can be left open when the project development environment is closed.

Before you try to use the Watch tool to connect to a project, confirm the project is running and the TCP/IP Server Runtime task is started. The TCP/IP Server (a.k.a. the Data Server) is what makes the project database accessible to debugging tools like Watch. Also, if the project runtime is hosted on a remote computer, note the host name or IP address of that computer. For more information, see Runtime Tasks on page 136.

Any time you try to access a project, if that project's security system is enabled then you may be asked to log on. This includes when you use the Watch tool to connect to a project. As such, your user account must

belong to a group that has the **Enable Remote Debugging Tools** option selected, in the security system settings. If you also want to be able to write values to the database, the **Watch (write)** option should also be selected, but that option is not required simply to use the Watch tool. For more information, see Project Security on page 639.

### Open the tool from the Local Management tools

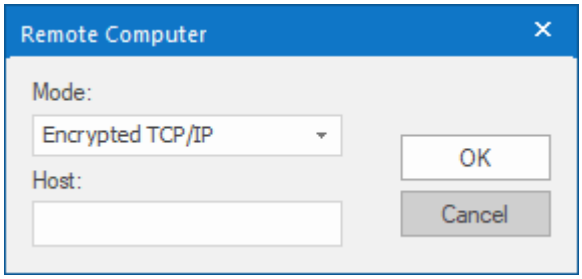
To open the Watch tool from the Local Management tools:

1. On the **Home** tab of the ribbon, in the **Local Management** group, click **Tasks**. The *Runtime Tasks* dialog box is displayed.
2. In the list of execution tasks, select the **Watch** task and then click **Start**. The Watch tool is displayed in a separate window, and it automatically connects to the project running on the local computer (a.k.a. LOCAL).
3. Click **OK** to close the *Runtime Tasks* dialog box.

### Open the tool from the Remote Management tools

To open the Watch tool from the Remote Management tools:

1. On the **Home** tab of the ribbon, in the **Remote Management** group, click **Watch**. The *Remote Computer* dialog box is displayed.




2. In the **Mode** box, select one of the following:

| Mode | Description |
|---|---|
| **TCP/IP** | Connect to the project's TCP/IP Server using unencrypted communication. |
| **Encrypted TCP/IP** | Connect to the project's TCP/IP Server using encrypted communication. |

   If you want to use **TCP/IP** mode, the standard **Port** option must be selected in the project settings. If you want to use **Encrypted TCP/IP** mode, the **Encrypted Port** option must be selected in the project settings. For more information about both options, see Communication tab on page 122.

3. In the **Host** box, type the host name or IP address of the computer that is hosting the project runtime. You do not need to include the port number as long as it has not been changed from the default port for the selected mode (e.g., port 51234 for **Encrypted TCP/IP**). If the port number has been changed, however, include the correct port number with the specified host.

> **Tip:** Even though you are opening the tool from the Remote Management tools, you can use it to connect to the project running on the local computer. To do this, leave the **Host** box empty.

4. Click **OK**. The Watch tool is displayed in a separate window, and it automatically connects to the specified host.

Remember that you can open multiple instances of the tool as long as you specify a different host and/or port for each instance.

### Run the tool from the command prompt

The Watch tool is actually a small utility program that is included with EcoStruxure Machine SCADA Expert 2020, and you can run it from the Windows command prompt with appropriate parameters. Assuming

the EcoStruxure Machine SCADA Expert 2020 software has been installed in its default location on your computer, the Watch tool should be located at:

```
C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin
\DatabaseSpyExt.exe
```

To run the program from the command prompt, make sure you are in the correct directory and then type the following:

```
DatabaseSpyExt /DSIP:<host name or IP address> /DSPORT:<port number>
```

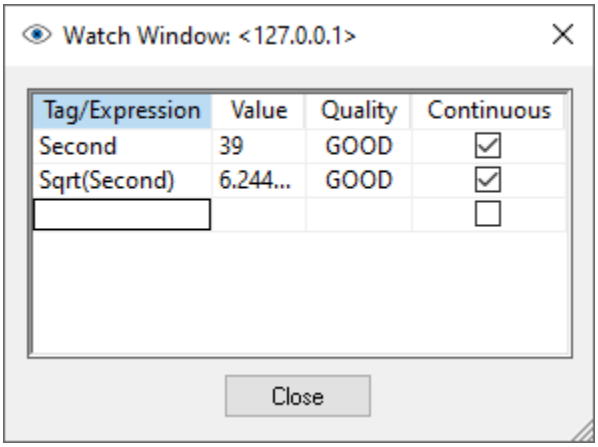
For the project running on the local computer, you can specify the host as `localhost` or `127.0.0.1`.

Remember that you can open multiple instances of the tool as long as you specify a different host and/or port for each instance.

### Use the tool to monitor the project database

If the Watch tool successfully connects to the specified host, it is displayed in its own window separate from the project development environment.

As stated at the beginning of this topic, the Watch tool is functionally similar to the Watch window on page 71 in the project development environment. It provides the same basic interface, and in most cases it can be used in the same way.




*An example of the Watch tool in use*

When you use the tool to connect to a project running on a remote computer, however, there are some limitations on accessing tags in the Shared Database part of the project database. For example, you might not be able to access class members that were imported through Tag Integration.

When you are done, click **Close** to disconnect from the specified host and close the tool.

## *Opening the Watch page for Mobile Access*

Use the Watch tool to open the *Watch* page for a project that is running on Mobile Access, or use your web browser to go directly to that page.

The *Watch* page for Mobile Access is functionally similar to the Watch window on page 71 in the project development environment, but it differs in several important ways:

- You can open the *Watch* page for projects running on both the local computer and remote computers, while the *Watch* window only shows the project that is currently open in the project development environment;
- You can open multiple instances of the *Watch* page, each in a separate browser tab, in order to access different projects at the same time; and

- The *Watch* page is displayed entirely in your web browser, as a part of the Mobile Access web interface.

Of course, the project must be configured to run on Mobile Access, and that requires some additional work that is not required for projects that run only on Windows or Windows Embedded. For more information, see Mobile Access on page 756.

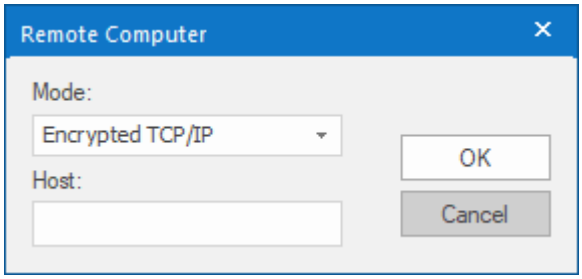
Before you try to open the *Watch* page for a project, confirm the project is running and the Mobile Access Runtime task is started. The Mobile Access Runtime is what generates the Mobile Access web interface for a project, and the *Watch* page is part of that web interface. Also, if the project runtime is hosted on a remote computer, note the host name or IP address of that computer. For more information, see Runtime Tasks on page 136.

Any time you try to access a project, if that project's security system is enabled then you may be asked to log on. This includes when you open the *Watch* page for a project. As such, your user account must belong to a group that has the **Enable Remote Debugging Tools** option selected, in the security system settings. If you also want to be able to write values to the database, the **Watch Window (write)** option should also be selected, but that option is not required simply to use the *Watch* page. For more information, see Project Security on page 639.

### Use the Watch tool to open the page

To use the Watch tool to open the *Watch* page:

1. On the **Home** tab of the ribbon, in the **Remote Management** group, click **Watch**. The *Remote Computer* dialog box is displayed.




2. In the **Mode** box, select one of the following:

| Mode | Description |
|---|---|
| **HTTP (Web)** | Connect to the Mobile Access web interface using unencrypted communication. |
| **HTTPS (Web)** | Connect to the Mobile Access web interface using encrypted communication. |

   If you want to use **HTTP (Web)** mode, you should not need to do anything else because that is the default configuration for most web servers. If you want to use **HTTPS (Web)** mode, the web server must be configured to use Secure Sockets Layer (SSL) to serve web pages. For more information, see Mobile Access web server add-on on page 768.

3. In the **Host** box, type the host name or IP address of the computer that is hosting the project runtime. You do not need to include the port number as long as it has not been changed from the default port for the web server. If the port number has been changed, however, include the correct port number with the specified host.

> **Tip:** Even though you are opening the page from the Remote Management tools, you can use it to open the *Watch* page for a project running on the local computer. To do this, leave the **Host** box empty.

4. Click **OK**. Your default web browser is opened, and it opens the *Watch* page for the specified host.

Remember that you can open multiple instances of the page in separate browser tabs as long as you specify a different host for each tab.

### Go directly to the page in your web browser

The *Watch* page is part of the Mobile Access web interface for a project, so you can go directly to the page without opening the project development environment. In the address bar of your web browser, type the following URL:

```
https://<host name or IP address>/EMSE2020/index.html?watch=1
```

If you are not already logged on to the project, you are prompted to log on at this time.

You can automatically log on by including your user name and password. To include only your user name, type the following URL:

```
https://<host name or IP address>/EMSE2020/index.html?watch=1&user=<user name>
```

The Logon page is displayed with your user name automatically entered. You are still prompted for your password.

To include both your user name and your password, type the following URL:

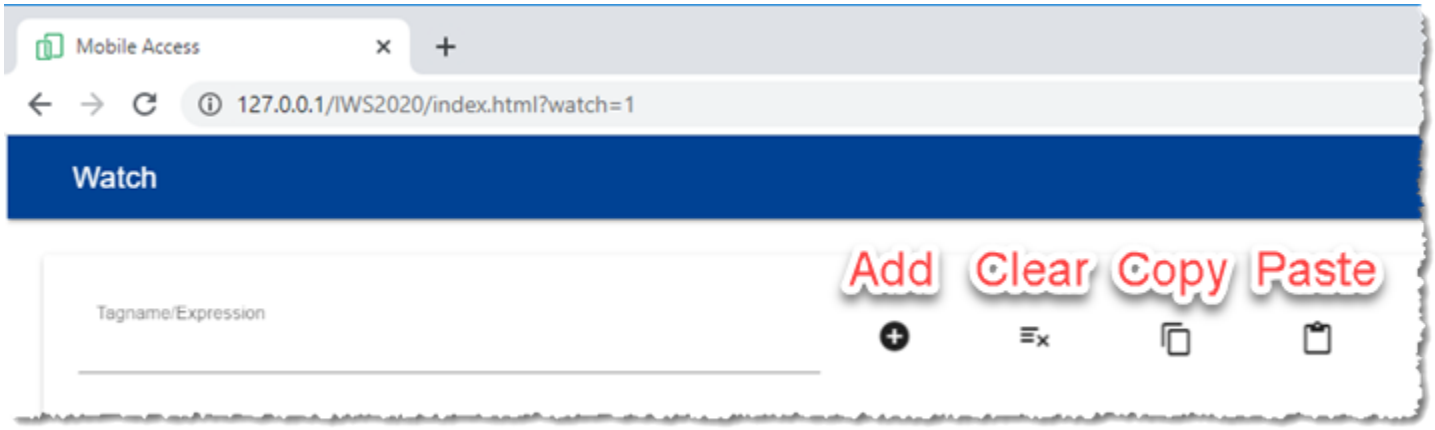
```
https://<host name or IP address>/EMSE2020/index.html?watch=1&user=<user
 name>&password=<password>
```

The Logon page is not displayed. Instead, you are automatically logged on, and the *Watch* page is displayed immediately. This option is not secure, however, especially if you save this URL as a favorite or bookmark.

Remember that you can open multiple instances of the page in separate browser tabs as long as you specify a different host for each tab.

### Use the page to monitor the project database

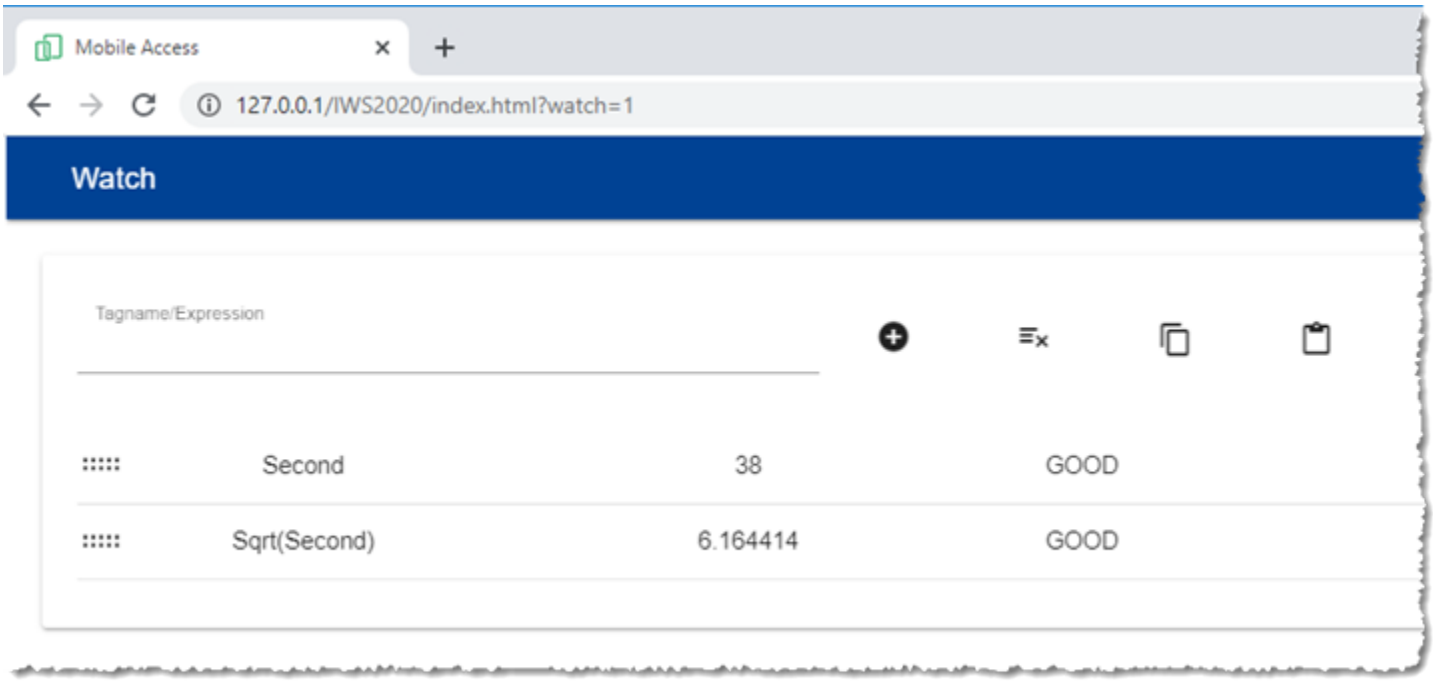
As stated at the beginning of this topic, the Watch tool is functionally similar to the Watch window on page 71 in the project development environment. It provides the same basic interface, and in most cases you can use it in the same way. However, the tools to manage the list of tags/expressions are somewhat different.




***Tools on the Watch page***

To add a project tag or expression to the list, type it in the **Tagname/Expression** box and then click/tap the **Add** tool. Each tag value will be updated whenever it changes. Each expression will be evaluated when it is added to the list, and thereafter it will be updated only if it includes at least one tag value that changes. Only expressions that are supported on Mobile Access can be evaluated with GOOD quality. Expressions that

are not supported cannot be evaluated and will always have BAD quality. For more information, see List of available functions on page 904.




***An example of tagnames and expressions added to the list***

To edit a tagname/expression in the list, double-click/tap it and then make your change. It will be updated after you make your change, as if you just added it.

To write a new value to a tag, double-click/tap the value (not the tagname) and then make your change. Remember that you must be logged on as a user who belongs to a group that has the **Watch Window (write)** option selected.

To remove a single tagname/expression from the list, click/tap it and then slide right.

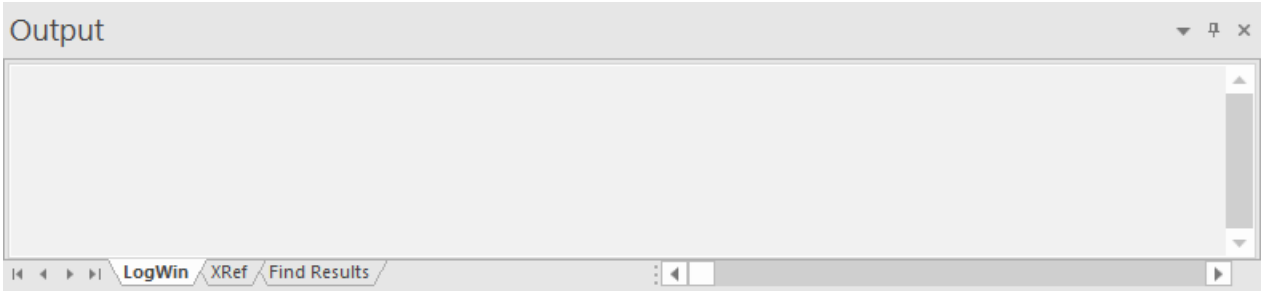
To clear the entire list, click/tap the **Clear** tool.

To copy the entire list to your computer's clipboard, click/tap the **Copy** tool. You can then paste the copied list into another application such as a text editor or spreadsheet.

To paste a list from your computer's clipboard, click/tap the **Paste** tool. The pasted list will be appended to the existing list.

## Output window

Use the *Output* window to view additional information about your project. By default, the window is located in the bottom-right corner of the project development environment.




***Output window***

The *Output* window has three tabs:

- The **LogWin** tab displays the log messages that are generated by your project. You can select exactly which types of messages are displayed, but generally speaking, the log includes run-time messages from the tags database, the communication drivers, the background tasks, the project security system, and so on, as well as certain "housekeeping" messages generated by the project development environment itself. You can use these messages to test and debug your project.

  > **Note:** The *Output* window cannot display the log for a project running on a remote computer. It also cannot print or save log messages. If you want to do either of those things, use the **LogWin** command instead.

- The **XRef** tab displays the results of using the **Cross Reference** command to find where a specific tag is used in your project. The results include the file path and name of the worksheet in which the tag is used, as well as the column and row in the worksheet. So, if something changes in the tag and produces unexpected or unsuccessful results, you can locate all instances of the tag for debugging purposes.
- The **Find Results** tab displays the results of using the **Global Find** command.

The *Output* window is dockable, which means you can drag it to another location in the project development environment.

### *Configure the log settings for the Output window*

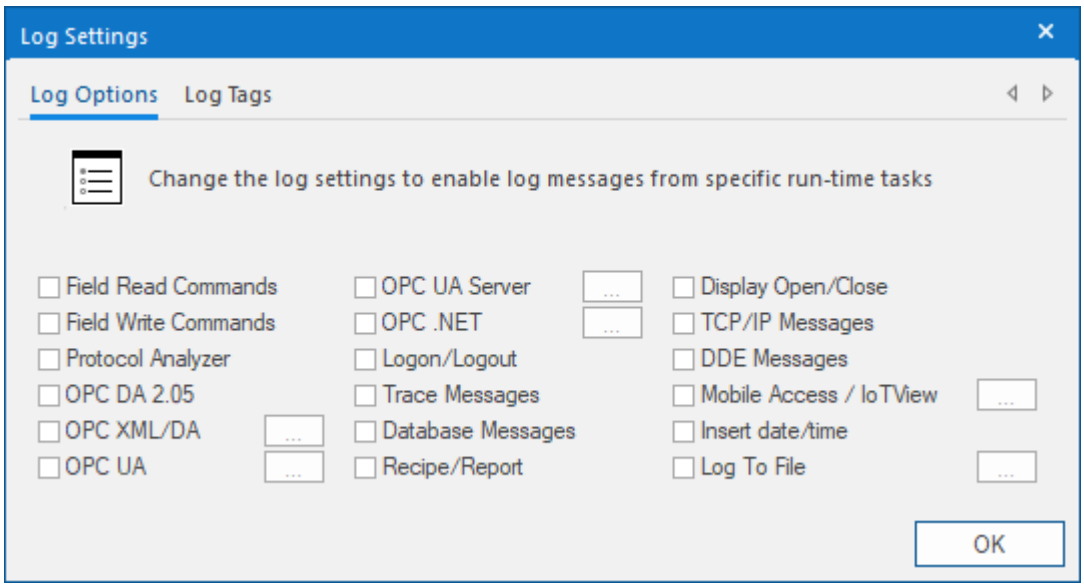
Configure the log settings for the *Output* window to select exactly which types of messages are included in the log.

By default, the log shows only debugging and error messages — that is, messages which indicate your project is not running properly. If the log showed all possible messages generated during project run time, it would quickly overflow and become unusable. Configuring these log settings allows you to select the types of log messages that you want to include in the log.

To configure the log settings for the *Output* window:

1. In the *Output* window, make sure the **Log** tab is selected.
2. Right-click anywhere in the window, and then click **Settings** on the shortcut menu.

   The *Log Settings* dialog box is displayed with the **Log Options** tab selected.



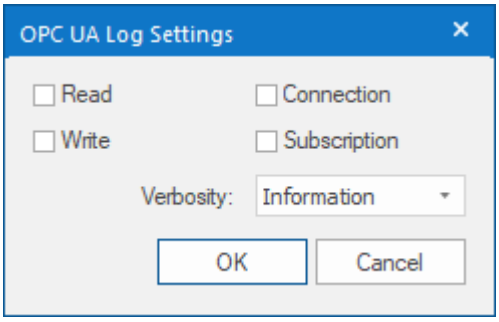

3. In the *Log Options* tab of the dialog box, select the types of log messages that you want to include in the log.

| Option | Description |
|---|---|
| **Field Read Commands** | Show any read commands that are sent to connected devices. |
| **Field Write Commands** | Show any write commands that are sent to connected devices. |
| **Protocol Analyzer** | Show messages generated by configured device drivers. |
| **OPC DA 2.05** | Show messages generated by the OPC DA 2.05 Client Runtime task. |
| **OPC XML/DA** | Show messages generated by the OPC XML/DA Client Runtime task. |
| **OPC UA** | Show messages generated by the OPC UA Client Runtime task. |
| **OPC UA Server** | Show messages generated by the OPC UA Server Runtime task. |
| **Logon/Logout** | Display a message whenever a user logs on or logs out. (For more information, see Security.) |
| **Trace Messages** | Show messages generated by the `Trace` function. This function is used to generate customized messages from within your project. |
| **Database Messages** | Show messages generated by communication with external databases through the Database Gateway. (Error messages only.) |
| **Recipe/Report** | Show messages generated by the execution of Recipe and Report worksheets. |
| **Display Open/Close** | Display detailed information whenever a screen is opened or closed:<br>• Disk Load Time: Time to load the screen file from the disk into memory. |

| Option | Description |
|---|---|
| | • Open Time: Time to open the screen, including initializing tags used in the screen and running any "OnOpen" scripts or functions.<br>• Total Load Time: Total time to load the screen (includes Disk Load Time and Open Time above).<br>• First Draw Time: Time to first drawing of screen objects.<br>• First OnWhile Time: Time to first running of any "OnWhile" scripts or functions.<br>• Total Open Time: Total time to open the screen (includes First Draw Time and First OnWhile Time above).<br>• Close Time: Time to close the screen, including finalizing tags used in the screen and running any "OnClose" scripts or functions.<br>• Total Close Time: Total time to close the screen, including the time to close the screen file on the disk.<br>This information can be used to analyze runtime performance on low-end target systems. If a particular step of opening or closing takes an unusually long time, then it can be identified and redesigned. |
| **TCP/IP Messages** | Show messages generated by TCP/IP communications. |
| **Mobile Access** | Show messages generated by Mobile Access. |
| **Insert date/time** | Timestamp each message. |

4. If you selected **OPC XML/DA** or **OPC UA**, you need to specify exactly which types of messages from the OPC communication stack should be included in the log. Click the browse button to the right of the selected option in order to open the *OPC Log Settings* dialog box, and then use it to select the types of messages you want to include.

   Each OPC communication stack has its own settings, but the descriptions of the settings are the same for all. The example messages below are for OPC UA.




***Log settings for OPC UA Client***

| Option | Description |
|---|---|
| **Read** | Enable trace messages on read operations. Examples of messages:<br>`OPC UA: Read Group 1 Started – OK`<br>`OPC UA: Read Group 1 Completed – OK`<br>`OPC UA: Read Group 25 Started – Error, asynchronous reading pending for the current group` |
| **Write** | Enable trace messages on write operations. Examples of messages:<br>`OPC UA: Write Group 1 Started [Line 1 = 10.25, Line 42 = 20.45] – Status OK`<br>`OPC UA: Write Group 1 Completed [All Items] – Status OK` |
| **Connection** | Enable trace messages on changes in connection status between the OPC Client (i.e., your project) and the OPC Server. Examples of messages:<br>`OPC UA: Connection established with server "Connection1"`<br>`OPC UA: Error to connect to server "Connection2" – Time out waiting for server response` |
| **Subscription** | Enable trace messages on subscriptions to server items, such as the creation of new subscriptions and changes in data type. Examples of messages:<br>`OPC UA: Group 1 => Subscription created` |

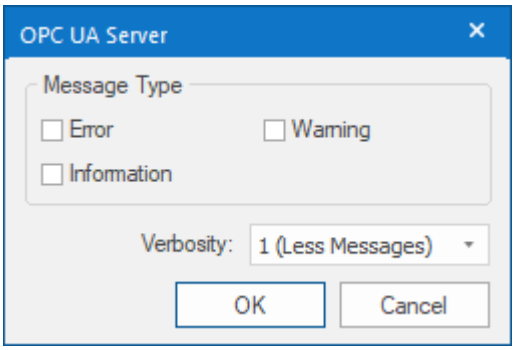
You also need to select the level of verbosity for OPC messages.

| Option | Description |
|---|---|
| **Error** | Critical issues that have caused the OPC communication to fail. These issues must be resolved before you can resume communication. |
| **Warning** | Non-critical issues that affect runtime performance or might cause the OPC communication to fail under other conditions. These issues should be resolved as soon as possible.<br>Includes **Error** above. |

| Option | Description |
|---|---|
| **Information** | All messages generated by the OPC communication. This is the default option.<br>Please note this can be extremely verbose, depending on which type(s) of messages you have selected to display and how many OPC item subscriptions you have created.<br>Includes **Error** and **Warning** above. |

When you are done, click **OK** to save the settings and close the dialog box.

5. If you selected **OPC UA Server**, you need to specify exactly which types of messages from the OPC UA Server Runtime task should be included in the log. Click the browse button to the right of the selected option in order to open the *OPC UA Server* dialog box, and then use it to select the types of messages you want to include.




***Log settings for OPC UA Server***

> **Note:** These settings are for the runtime task only. They are separate from the trace messages generated by the OPC UA communication stack. For more information, see How to manage OPC UA Server during project run time on page 631.

When you are done, click **OK** to save the settings and close the dialog box.

6. Use the *OPC UA Server* dialog box to select the types of messages that you want to include in the log.

Each OPC communication stack has its own settings, but the descriptions of the settings are the same for all. The example messages below are for OPC UA.

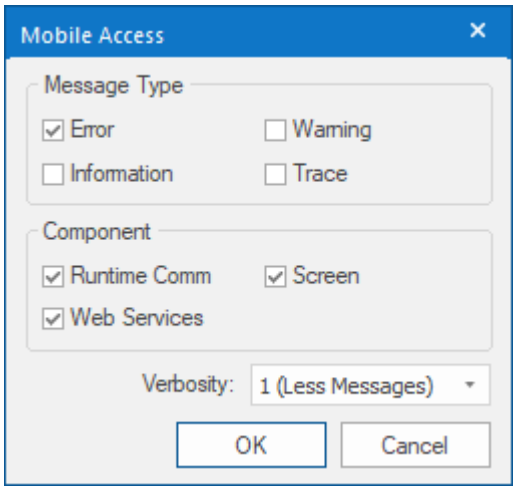
You also need to select the level of verbosity for OPC messages.

| Option | Description |
|---|---|
| **Error** | Critical issues that have caused the OPC communication to fail. These issues must be resolved before you can resume communication. |
| **Warning** | Non-critical issues that affect runtime performance or might cause the OPC communication to fail under other conditions. These issues should be resolved as soon as possible.<br>Includes **Error** above. |
| **Information** | All messages generated by the OPC communication. This is the default option.<br>Please note this can be extremely verbose, depending on which type(s) of messages you have selected to display and how many OPC item subscriptions you have created. |

| Option | Description |
| --- | --- |
| | Includes **Error** and **Warning** above. |

When you are done, click **OK** to save the settings for the selected OPC task and close the dialog box.

7. If you selected **Mobile Access**, you need to specify exactly which types of messages from the Mobile Access Runtime task should be included in the log. Click the browse button to the right of the selected option in order to open the *Mobile Access* dialog box, and then use it to select the types of messages you want to include.



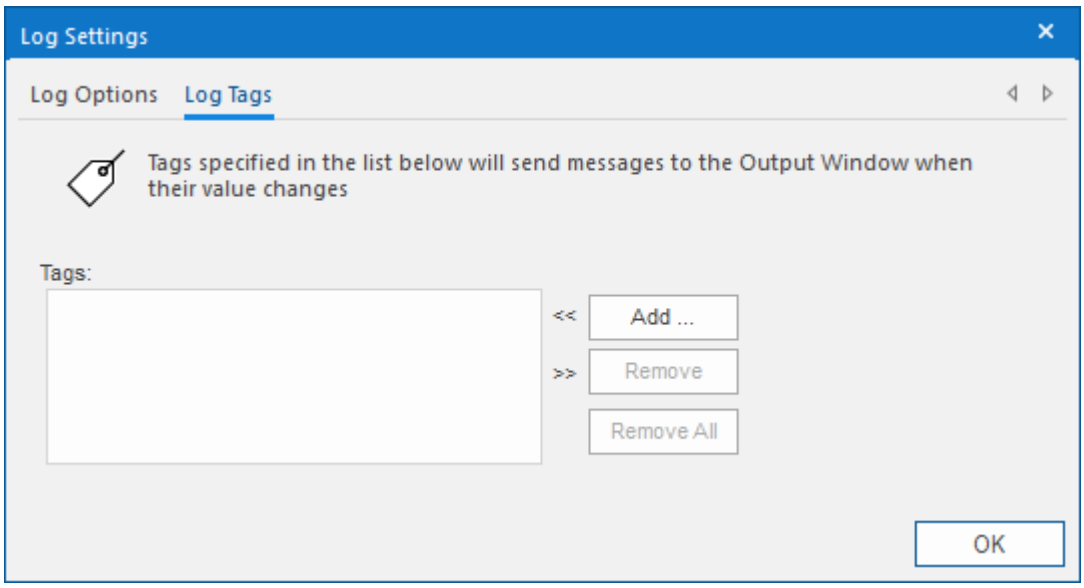

***Log settings for Mobile Access***

By default, only error messages — that is, messages about critical issues that have caused Mobile Access to fail — are included in the log. For more information about the available options, see Use the activity log to troubleshoot the Mobile Access web interface on page 804.

When you are done, click **OK** to save the settings and close the dialog box.

8. If you want to log every change in the values of specifc project tags, do the following:
   a) In the *Log Settings* dialog box, click the **Log Tags** tab.

      The tab is displayed.




b) Click **Add**.
The *Object Finder* is displayed.

c) Use the Object Finder to select the project tag that you want to log, and then click **OK**.
You can also log changes in tag properties. For more information, see Reference a tag property instead of a project tag on page 169.
The selected tag is added to the log.

d) Repeat as needed for each project tag that you want to log.

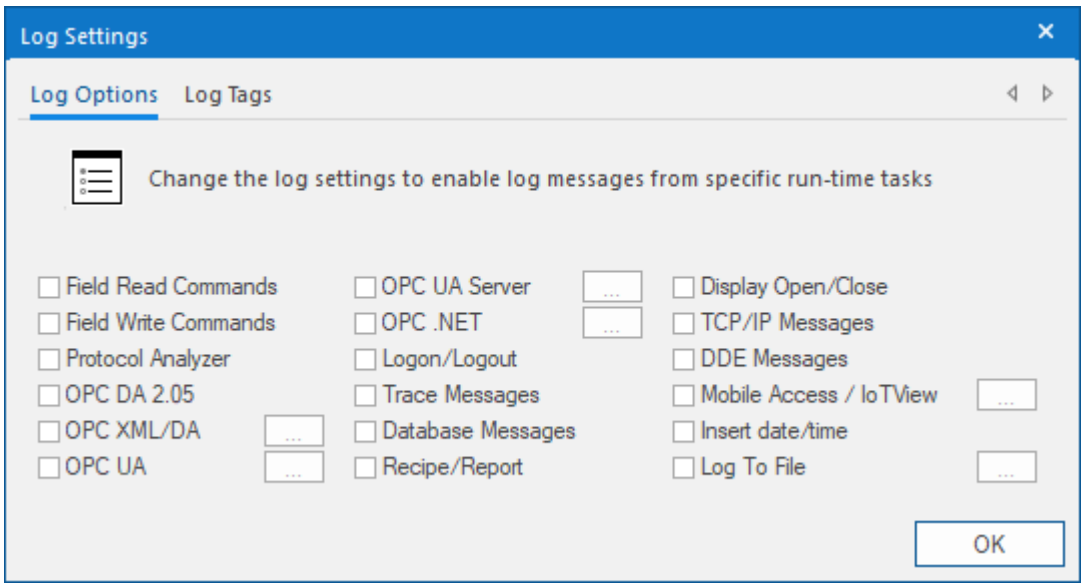
9. Click **OK** to save your settings and close the *Log Settings* dialog box.

### *Save log messages from the Output window to a file*

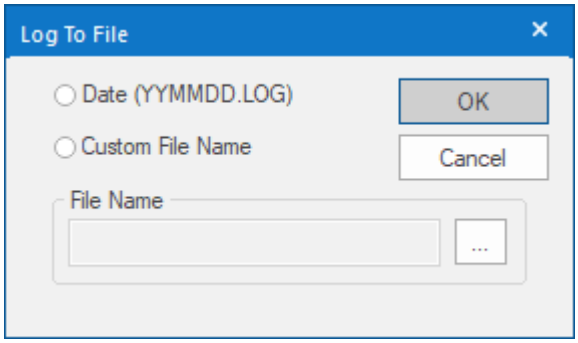
Configure the log settings for the Output window in order to save its output — that is, the log messages from the project running on the local computer — to a file.

To configure the settings for the Output window:

1. In the *Output* window, make sure the **LogWin** tab is selected.
2. Right-click anywhere in the *Output* window, and then click **Settings** on the shortcut menu.
The *Log Settings* dialog box is displayed with the **Log Options** tab selected.




3. Select the **Log To File** option, and then click the More button (...) to the right of the option.
   The *Log To File* dialog box is displayed.




4. Select how you want to save the output.

| Option | Description |
|---|---|
| **Date** | The log is saved in dated files in the project folder, using the file name format `YYMMDD.log` (e.g., `200110.log`). A new file is created for each day the project runs. |
| **Custom File Name** | The log is saved in a single file at a specified location. |

5. If you selected **Custom File Name**, use the **File Name** box to specify where the file should be saved:
   a) Click the browse button to the right of the **File Name** box.
   A standard *Save As* dialog box is displayed.
   b) Locate the folder in which you want to save the file, and then in the **File name** box, type the custom file name.
   c) Click **Save**.
   The *Save As* dialog box is closed, and the file path and name are displayed in the **File Name** box.
6. Click **OK** to close the *Log To File* dialog box.

If your project is already running, it will start saving the file(s) immediately. Otherwise, it will start saving the file(s) the next time it is run.

The saved log files can quickly fill the available storage space, depending on how active the project is and how the log settings are configured, so you should be careful about leaving the **Log To File** option selected for extended periods. This should be for testing and debugging purposes only. If you want to save a more permanent record of run-time behavior or performance, use Alarms, Events, Trends, Reports, and other such features in your project.

## About the LogWin tool

The LogWin tool provides additional tools for viewing, printing, and saving the project runtime log.

By default, the project runtime log is displayed in the Output window in the project development environment. That is good enough for most testing and debugging, but the *Output* window can only display the log for the project running on the local computer and it cannot print or save the log.

The LogWin tool displays essentially the same information as the *Output* window, but it provides some additional tools for doing so. It can view the log generated by either the project runtime on either the local computer or a remote computer, depending on how you open it, and it can print or save the log for future reference.

> **Note:**
>
> - To enable logging for Mobile Access, select the appropriate options in the *Mobile Access Configuration* worksheet. For more information, see Configure the global settings for all areas on page 790.

### *Open the LogWin tool*

This software includes a standalone LogWin tool that you can use to connect to a running project and then view that project's log messages.

The LogWin tool is functionally similar to the Output window in the project development environment, but it differs in several important ways:

- The LogWin tool can connect to projects running on both the local computer and remote computers, while the Output window only shows the project that is currently open in the project development environment;
- You can open multiple instances of the LogWin tool in order to connect to different projects at the same time; and
- The LogWin tool runs separate from the project development environment, and it can be left open when the project development environment is closed.

Before you try to use the LogWin tool to connect to a project, confirm the project is running and the TCP/IP Server Runtime task is started. The TCP/IP Server (a.k.a. the Data Server) is what makes the project database accessible to debugging tools like LogWin. Also, if project runtime is hosted on a remote computer, note the host name or IP address of that computer. For more information, see Runtime Tasks on page 136.

Any time you try to access a project, if that project's security system is enabled then you may be asked to log on. This includes when you use the LogWin tool to connect to a project. As such, your user account must belong to a group that has the **Enable Remote Debugging Tools** option selected, in the security system settings. For more information, see Project Security on page 639.

#### Open the tool from the Local Management tools

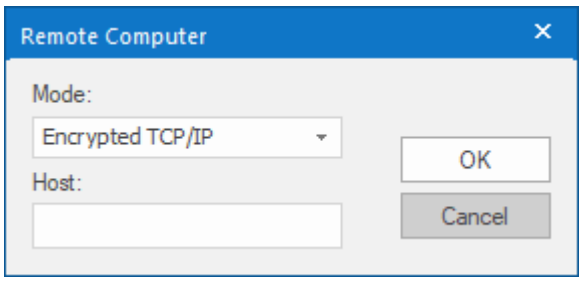
To open the LogWin tool from the Local Management tools:

1. On the **Home** tab of the ribbon, in the **Local Management** group, click **Tasks**. The *Runtime Tasks* dialog box is displayed.
2. In the list of execution tasks, select the **LogWin** task and then click **Start**. The LogWin tool is displayed in a separate window, and it automatically connects to the project running on the local computer (a.k.a. LOCAL).
3. Click **OK** to close the *Runtime Tasks* dialog box.

#### Open the tool from the Remote Management tools

To open the LogWin tool from the Remote Management tools:

1. On the **Home** tab of the ribbon, in the **Remote Management** group, click **LogWin**. The *Remote Computer* dialog box is displayed.




2. In the **Mode** box, select one of the following:

| Mode | Description |
|---|---|
| **TCP/IP** | Connect to the project's TCP/IP Server using unencrypted communication. |
| **Encrypted TCP/IP** | Connect to the project's TCP/IP Server using encrypted communication. |

   If you want to use **TCP/IP** mode, the standard **Port** option must be selected in the project settings. If you want to use **Encrypted TCP/IP** mode, the **Encrypted Port** option must be selected in the project settings. For more information about both options, see Communication tab on page 122.

3. In the **Host** box, type the host name or IP address of the computer that is hosting the project runtime. You do not need to include the port number as long as it has not been changed from the default port for the selected mode (e.g., port 51234 for **Encrypted TCP/IP**). If the port number has been changed, however, include the correct port number with the specified host.

   > **Tip:** Even though you are opening the tool from the Remote Management tools, you can use it to connect to the project running on the local computer. To do this, leave the **Host** box empty.

4. Click **OK**. The LogWin tool is displayed in a separate window, and it automatically connects to the specified host.

Remember that you can open multiple instances of the tool as long as you specify a different host and/or port for each instance.

### Run the tool from the command prompt

The LogWin tool is actually a small utility program that is included with EcoStruxure Machine SCADA Expert 2020, and you can run it from the Windows command prompt with appropriate parameters. Assuming the EcoStruxure Machine SCADA Expert 2020 software has been installed in its default location on your computer, the LogWin tool should be located at:

```
C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin
\LogWinExt.exe
```

To run the program from the command prompt, make sure you are in the correct directory and then type the following:

```
LogWinExt /DSIP:<host name or IP address> /DSPORT:<port number>
```

For the project running on the local computer, you can specify the host as `localhost` or `127.0.0.1`.

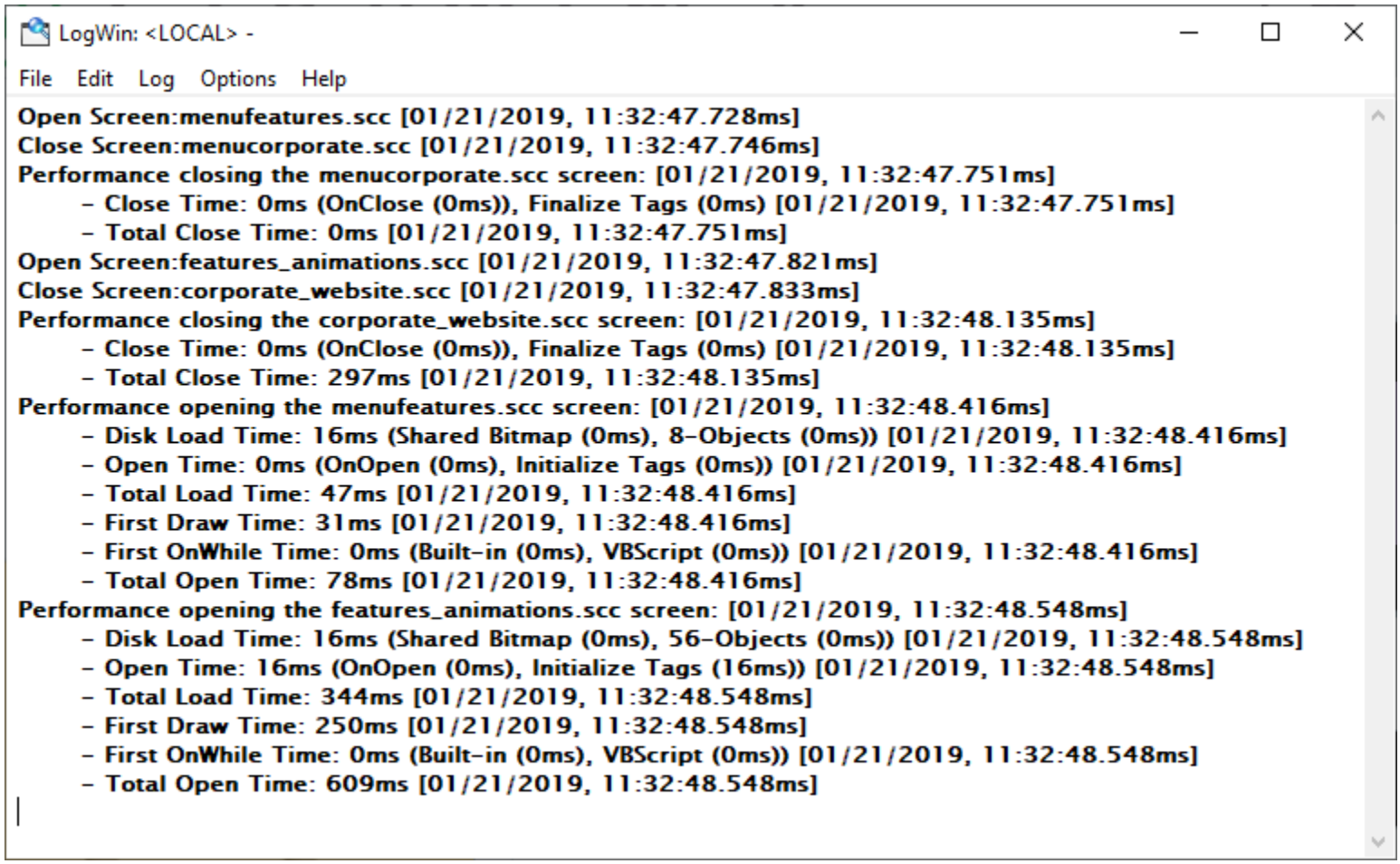
Remember that you can open multiple instances of the tool as long as you specify a different host and/or port for each instance.

### Use the tool to view log messages

If the LogWin tool successfully connects to the specified host, it is displayed in its own window separate from the project development environment.

As stated at the beginning of this topic, the LogWin tool is functionally similar to the Output window in the project development environment. It provides the same basic interface, and in most cases it can be used in

the same way. However, the LogWin tool offers even more options than the Output window, such as saving the output on disk. For more information about those options, see the other topics in this section.




*An example of the LogWin tool in use*

When you are done, click **File** > **Exit** to disconnect from the specified host and close the tool.

### *Configure the log settings for the LogWin module*

Configure the log settings for the LogWin module to select exactly which types of messages are included in the log.

Before you begin this task, verify that the project is running on either the local or remote computer and that the LogWin module is open for that computer.

By default, the log shows only debugging and error messages — that is, messages which indicate your project is not running properly. If the log showed all possible messages generated during project run time, it would quickly overflow and become unusable. Configuring these log settings allows you to select the types of log messages that you want to include in the log.

> **Note:** If you are using the LogWin module to view the log for the local computer, be aware that it shares its log settings with the *Output* window in the project development environment. If you change the settings for one, they will also be changed for the other. For more information, see Output window on page 72.

To configure the log settings for the LogWin module:

1. In the *LogWin* window, on the **Log** menu, select the types of log messages that you want to include in the log.

   You will need to open the menu to select each additional option, because there is not a single dialog box for the options as there is in the *Output* window.

| Option | Description |
|---|---|
| **Field Read Commands** | Show any read commands that are sent to connected devices. |
| **Field Write Commands** | Show any write commands that are sent to connected devices. |
| **Protocol Analyzer** | Show messages generated by configured device drivers. |
| **OPC DA 2.05** | Show messages generated by the OPC DA 2.05 Client Runtime task. |
| **OPC XML/DA** | Show messages generated by the OPC XML/DA Client Runtime task. |
| **OPC UA** | Show messages generated by the OPC UA Client Runtime task. |
| **OPC UA Server** | Show messages generated by the OPC UA Server Runtime task. |
| **Logon/Logout** | Display a message whenever a user logs on or logs out. (For more information, see Security.) |
| **Trace Messages** | Show messages generated by the `Trace` function. This function is used to generate customized messages from within your project. |
| **Database Messages** | Show messages generated by communication with external databases through the Database Gateway. (Error messages only.) |
| **Recipe/Report** | Show messages generated by the execution of Recipe and Report worksheets. |
| **Display Open/Close** | Display detailed information whenever a screen is opened or closed:<br>• Disk Load Time: Time to load the screen file from the disk into memory.<br>• Open Time: Time to open the screen, including initializing tags used in the screen and running any "OnOpen" scripts or functions.<br>• Total Load Time: Total time to load the screen (includes Disk Load Time and Open Time above).<br>• First Draw Time: Time to first drawing of screen objects.<br>• First OnWhile Time: Time to first running of any "OnWhile" scripts or functions.<br>• Total Open Time: Total time to open the screen (includes First Draw Time and First OnWhile Time above).<br>• Close Time: Time to close the screen, including finalizing tags used in the screen and running any "OnClose" scripts or functions.<br>• Total Close Time: Total time to close the screen, including the time to close the screen file on the disk. |

| Option | Description |
|---|---|
| | This information can be used to analyze runtime performance on low-end target systems. If a particular step of opening or closing takes an unusually long time, then it can be identified and redesigned. |
| **TCP/IP Messages** | Show messages generated by TCP/IP communications. |
| **Mobile Access** | Show messages generated by Mobile Access. |
| **Insert date/time** | Timestamp each message. |

After you select an option, a check mark is displayed next to it to indicate that it has been selected.

2. If you select **OPC XML/DA** or **OPC UA** in the **Log** menu, a dialog box is displayed. In this dialog box, select **Enable**, and then select the types of OPC messages that you want to include in the log.

   Each OPC task has its own settings, but the descriptions of the settings are the same for all OPC tasks. The example messages below are for OPC UA.

| Option | Description |
|---|---|
| **Read** | Enable trace messages on read operations. Examples of messages:<br>`OPC UA: Read Group 1 Started – OK`<br>`OPC UA: Read Group 1 Completed – OK`<br>`OPC UA: Read Group 25 Started – Error, asynchronous reading pending for the current group` |
| **Write** | Enable trace messages on write operations. Examples of messages:<br>`OPC UA: Write Group 1 Started [Line 1 = 10.25, Line 42 = 20.45] – Status OK`<br>`OPC UA: Write Group 1 Completed [All Items] – Status OK` |
| **Connection** | Enable trace messages on changes in connection status between the OPC Client (i.e., your project) and the OPC Server. Examples of messages:<br>`OPC UA: Connection established with server "Connection1"`<br>`OPC UA: Error to connect to server "Connection2" – Time out waiting for server response` |

| Option | Description |
|---|---|
| **Subscription** | Enable trace messages on subscriptions to server items, such as the creation of new subscriptions and changes in data type. Examples of messages:<br>`OPC UA: Group 1 => Subscription created` |

You also need to select the level of verbosity for OPC messages.

| Option | Description |
|---|---|
| **Error** | Critical issues that have caused the OPC communication to fail. These issues must be resolved before you can resume communication. |
| **Warning** | Non-critical issues that affect runtime performance or might cause the OPC communication to fail under other conditions. These issues should be resolved as soon as possible.<br>Includes **Error** above. |
| **Information** | All messages generated by the OPC communication. This is the default option.<br>Please note this can be extremely verbose, depending on which type(s) of messages you have selected to display and how many OPC item subscriptions you have created.<br>Includes **Error** and **Warning** above. |

3. If you select **Mobile Access** in the **Log** menu, the *Mobile Access* dialog box is displayed. In this dialog box, select **Enable**, and then select the types of Mobile Access messages that you want to include in the log.

   By default, only error messages — that is, messages about critical issues that have caused Mobile Access to fail — are included in the log. For more information about the available options, see Use the activity log to troubleshoot the Mobile Access web interface on page 804.

4. If you want to log every change in the values of specifc project tags, do the following:
   a) On the **Log** menu, click **Tags**.
   The *Log Tags* dialog box is displayed.
   b) In the dialog box, click **Add**.
   The *Object Finder* window is displayed.
   c) Use the Object Finder to select a project tag that you want to log, and then click **OK**.

   > **Tip:** You can also log changes in tag properties. For more information, see Reference a tag property instead of a project tag on page 169.

   The selected tag is added to the log.
   d) Repeat as needed for each project tag that you want to log.
   e) Click **Close** to close the *Log Tags* dialog box.
5. If you want to insert a time stamp in the log messages, click **Insert date/time** on the **Options** menu.

The new settings are saved in a file on the local computer. Every time you open the LogWin module on that computer, it will use the settings from the file, regardless of which project runtime the module connects to.

### *Save log messages from the LogWin tool to a file*

Configure the settings for the LogWin tool in order to save its output — that is, the log messages from a project running on a remote computer — to a file. The output is saved continuously as long as the LogWin tool is open and the project is running.

Before you begin, verify the project is running on the remote computer and the project's LogWin task is started.

To configure the settings for the LogWin tool:

1. In the *LogWin* window, go to **Options**, and then select **Log To File**.
   The *Log To File* dialog box is displayed.
2. Select how you want to save the output.

| Option | Description |
|---|---|
| **Disable** | Saving is disabled. This is the default when you open the LogWin tool. |
| **Date** | The output is saved in dated files, using the name format `YYMMDD.log` (e.g., `150513.log`). |
| **Custom File Name** | The output is saved in a single file at a specified location. |

3. If you selected **Custom File Name**, use the **File Name** box to specify where the file should be saved:
   a) Click the browse button to the right of the **File Name** box.
      A standard *Save As* dialog box is displayed.
   b) Locate the folder in which you want to save the file, and then in the **File name** box, type the custom file name.
   c) Click **Save**.
      The *Save As* dialog box is closed, and the file path and name are displayed in the **File Name** box.
4. Click **OK** to close the *Log To File* dialog box.

Assuming the project is running and generating log messages, the LogWin tool immediately starts saving the file(s). If you selected the **Date** option, the files are saved in the project folder of the project that was most recently opened in the project development environment, regardless of whether that is the project to which the LogWin tool is currently connected. If you selected the **Custom File Name** option, the file is save at the specified location.

The saved log files can quickly fill the available storage space, depending on how active the project runtime is and how the log settings are configured, so you should be careful about letting the LogWin tool run for extended periods. This should be for testing and debugging purposes only. If you want to save a more permanent record of run-time behavior or performance, use Alarms, Events, Trends, Reports, and other such features in your project.

# Remote Management

Use the Remote Management tool to download project files to a target device and then run/stop the project on that device.

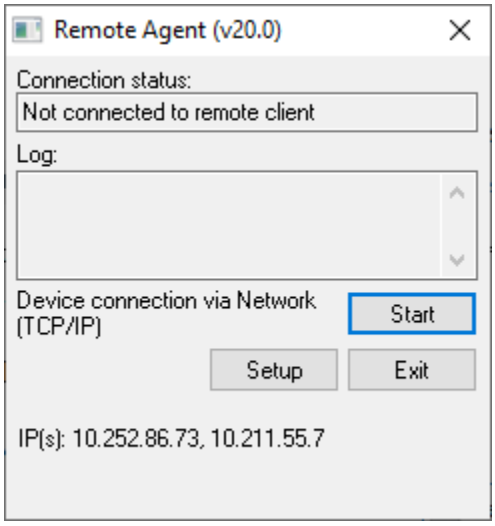
In this case, a "target device" is any computer that has the project runtime server software installed and running. For more information, see Installation Guide on page 33.

The actual connection is handled by a small program on the target device called Remote Agent (`CEServer.exe`).

## Enable security in Remote Agent and add users

Enable security in Remote Agent on a target device in order to allow only certain users to connect to the device and to encrypt communications between the device and the project development application.

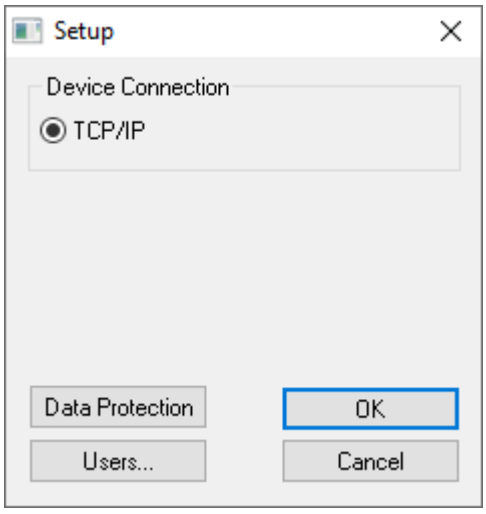
Before you begin this task, the project runtime software (EcoStruxure Machine SCADA Expert 2020) must be installed on the target device and Remote Agent must be running.




***Remote Agent***

To enable security and add users:

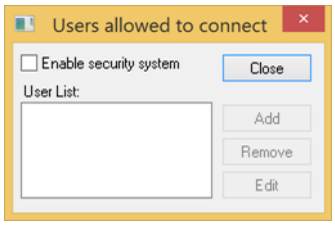
1. In Remote Agent on the target device, click **Setup**.
   The *Setup* dialog box is displayed.




***Setup dialog box***

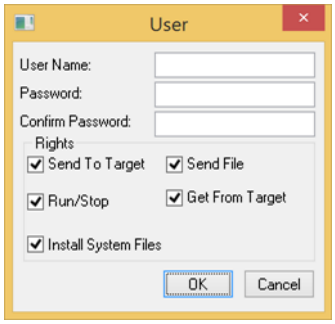
2. Click **Users**.

The *Users* dialog box is displayed.




***Users dialog box***

3. Select **Enable security system**.
4. Click **Add**.
   The *Add User* dialog box is displayed.




***Add User dialog box***

5. Type the **User Name** and **Password** for the user, and then select which rights that user should have.
   The available rights correspond to commands in the Remote Management tool in the project development application.

| Option | Description |
|---|---|
| **Send To Target** | Send an entire project to the device. |
| **Send File** | Send a specific file to the device. |
| **Run/Stop** | Run or stop the project runtime server on the device. |
| **Get From Target** | Get an entire project from the device. |
| **Install System Files** | Install the project runtime software on the device. |

6. Click **OK**.
   The *Add User* dialog box is closed and the user is added to the **User List**.
7. Repeat for all of the users that should be able to connect to the device.
8. Click **Close** to close the *Users* dialog box.
9. Click **OK** to close the *Setup* dialog box, but leave Remote Agent running on the device.

## *Customize Remote Agent's encryption key*

Customize Remote Agent's encryption key in order to increase the security of remote management.

Before you begin this task, Remote Agent must be installed and running on the target device and security must be enabled.

When you enable security in Remote Agent, connections between the project development application and the target device are automatically encrypted. This prevents third-party programs from intercepting projects or sending unauthorized Run/Stop commands to the target device

By default, Remote Agent uses a built-in encryption key that should be secure enough for most situations. You may want to customize the key, however, because the more unique the key is, the more secure the connection will be.

To customize the key:

1. Determine what you want the key to be.

   It does not matter what the key actually is, because it is automatically shared between Remote Agent and the project development application. As such, you can use an online GUID generator to generate a suitable key.

2. On the target device, exit Remote Agent.
3. Edit the Remote Agent initialization file.
   - For Remote Agent running on Windows, Windows Server, or Windows Embedded Standard, create a new text file named `RemoteAPI.ini` in the same directory as Remote Agent (`CEServer.exe`).
4. Insert the following lines in the initialization file.

   ```
   [Protection]
   InternalKey=<your custom key>
   ```

5. Save and close the initialization file.
6. Run Remote Agent.

Remote Agent will automatically share the new key with the project development application, when you use the Remote Management tool to connect to the target device and log on as an authorized user.

## Download your project to the target device

Use the Remote Management tool to download your project to a target device.

Before you begin this task, your project development workstation should be connected to the target device and Remote Agent should be installed and running on that device.

To download your project:

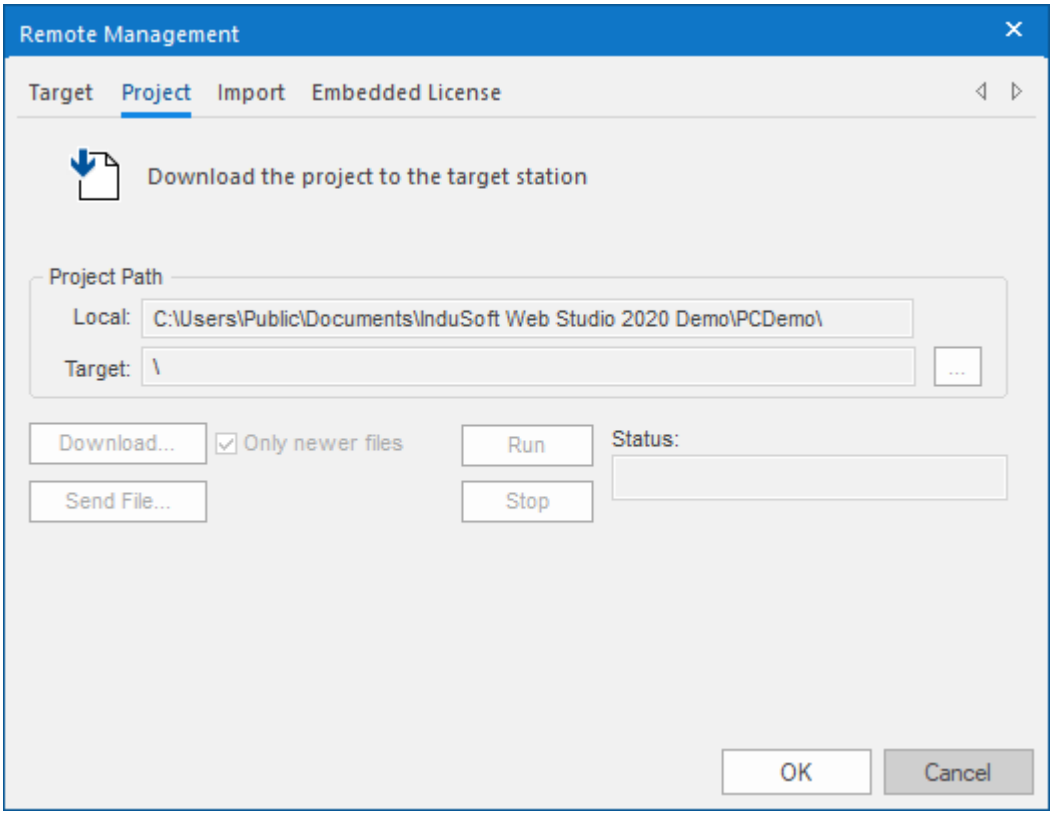
1. On the **Home** tab of the ribbon, in the **Remote Management** group, click **Connect**.
   The *Remote Management* dialog box is displayed with the **Target** tab selected.
2. On the **Target** tab of the dialog box, confirm that you are connected to the target device. If you are not, review the connection settings and then click **Connect**.

   The Remote Management tool connects to the target device and its status is displayed.

   > **Tip:** If the connection settings are correct but you cannot connect to the target device, make sure the Remote Agent program is running on the device, empty the device's `\Temp` directory, and then try again.

3. In the list of tabs on the left, click **Project**.

   The **Project** tab of the dialog box is displayed.




4. In the **Local** box, you should see the location of the project that is currently open in your development environment. This is the project that will be downloaded to the target device. If it is not the project that you want to download, cancel the *Remote Management* dialog box, open the correct project, and then restart this task.
5. In the **Target** box, confirm the location of the project folder on the target device. If the location is not correct, click the browse button to the right and then select a new location.

   Assuming you are properly connected to the device, you should be able to browse it like a network volume.

> **Note:** By default, you can download the project to any location on the target device, even to another location outside the folder that contains the project runtime software. You might consider this a security vulnerability, however, so if you want to restrict downloads only to the folder that contains the project runtime software, go to Remote Agent on the target device and select **Lock project download**.

6. Click **Download** to download the entire project to the target device, or click **Send File** to select a specific file to send.

   When you download your project to the device, new project files automatically and immediately replace old ones, even while the project is running. As such, you may choose to stop the project on the device (by clicking **Stop**) before you download files, to make sure the project stops as expected and does not cause a disruption. You are not required to stop the project, however; if it is robust enough to handle changes while running, you can download new files at any time.

   The **Only newer files** option controls which project files are downloaded:

   - If this option is selected, only newer files — that is, files that have changed since the last time the project was downloaded to the device — will be downloaded. This can reduce the total time needed to download.
   - If this option is cleared, all of the project files will be downloaded, overwriting the existing files on the device.

7. If you have enabled Data Protection in your project settings but not on the target device, you will be asked if you want to enable it on the target device. Click **Yes**, and then when prompted, enter the Data Protection password.

   For more information, see Enable Data Protection to encrypt sensitive information on page 113. If you do not enable Data Protection on the target device now, you must do so later in Remote Agent on the device itself before you run the project.

The project is downloaded to the target device. If the download is interrupted, you will be asked if you want to continue, and if you do, you will also be advised that the project might not run properly after it is downloaded.

Please note that once you have configured the Remote Management settings, you can click **Download** on the **Home** tab of the ribbon to send new project files at any time without opening this dialog box.

You can also compress the project files to make them download more quickly over a slow network connection. To do this, select the **Enable File Compression** check box in the Communication tab of the *Project Settings* dialog box.

## Run or stop your project on the target device

Once you have downloaded your project files to the target device, you can run or stop the project at will.

There are three ways to run or stop a project on a target device.

### Ribbon

Assuming you are currently connected to the target device and you have downloaded your project files to it, the easiest way to run and stop the project is to use the appropriate commands on the ribbon: on the **Home** tab of the ribbon, in the **Remote Management** group, click **Run** or **Stop**.

### Remote Management

If you are not currently connected to the target device:

1. On the **Home** tab of the ribbon, in the **Remote Management** group, click **Connect**. The *Remote Management* window is displayed.
2. Click the **Target** tab of the *Remote Management* window, check the connection settings, and then click **Connect**.
3. Click the **Project** tab of the *Remote Management* window, check the project settings, and then click **Run** or **Stop**.

## Configure Remote Agent to autorun a project

By default, you must manually run your finished project on the target device, either from your PC by using the **Project** tab of the *Remote Management* dialog (see above) or on the target device itself by clicking the **Start** button in the *Remote Agent* dialog.

However, you can configure the target device to automatically run a specified project. To do this, edit the file **`CEServer.ini`** on the target device to include the following setting:

```
[Setup]
AppName=Applicaion Path
```

Where ***<project name>*** is the location of the EcoStruxure Machine SCADA Expert project files on the target device. For example:

```
[Setup]
AppName=\Harddisk\Test\CEserverTest
```

The next time the target device boots up and opens the *Remote Agent* dialog (**`CEServer.exe`**), it will read this setting and automatically run the specified project.

There are three ways to edit the **`CEServer.ini`** file:

- Edit the file directly on the target device using an attached keyboard or the touchscreen keypad. The file should be located in the same directory as the Remote Agent program (`CEServer.exe`).
- Mount the target device as a shared volume on your PC and edit the file there.
- Edit the file in the `Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Redist` directory *before* you install the system files on the target device.

> **Note:** This last method changes the default copy of `CEServer.ini` that is included with EcoStruxure Machine SCADA Expert. Use this method only if:
>
> - You back up the file before editing it;
> - You are installing the same system files on multiple, identical devices; and
> - You already know the location (file path) of the EcoStruxure Machine SCADA Expert project files on the target device (perhaps by using the normal installation method on a test station).

# Thin Clients and Mobile Access

This section describes how to make your project accessible to thin clients and mobile devices.

EcoStruxure Machine SCADA Expert 2020 is built on a server/client architecture that can support both thick clients and thin clients. The choice of the type of client depends upon your system requirements:

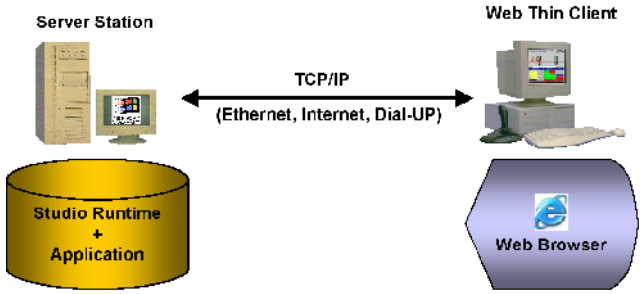
- A thick client is a computer that performs most, if not all, of the processing activity during project run time. It has sufficient processing power, memory, and graphics to run the complete project files, and it only exchanges data (i.e., function syncronization and tag value changes) with the project runtime server as needed. (A project runtime server can also be a thick client to another server.)

  In EcoStruxure Machine SCADA Expert, thick clients are handled through Remote Management.

- A thin client is a computer that depends primarily on the project runtime server for processing. It only needs to have a network connection to get screens and data from the server and a web browser to display the screens to the user.

### Thin Clients in EcoStruxure Machine SCADA Expert

EcoStruxure Machine SCADA Expert 2020 allows you to create screens that can be viewed on a remote station in a regular web browser. The station where the user can view the screens is called the thin client.




*Typical thin client architecture*

The actual EcoStruxure Machine SCADA Expert 2020 software is installed only on the server station. All project files — the tags database, screen files, and task worksheets — are stored on the server, and all background and communication tasks are executed on the server.

The thin client simply loads your project's graphical interface (i.e., the screens containing objects and animations) as needed and then uses that interface to represent data (i.e., tag values) on the server. You do not need to install the EcoStruxure Machine SCADA Expert development application or any of the project files on the thin client.

This solution provides a high level of flexibility because any computer that has a network connection to the server station (via TCP/IP) can access the project during run time.

> **Note:** Since screens and screen objects may contain scripting, using VBScript and/or the EcoStruxure Machine SCADA Expert Scripting Language, these scripts are executed on the thin client.

### Competitive Advantages of Thin Clients

EcoStruxure Machine SCADA Expert 2020 is built on a server/client architecture that supports true thin clients. This capability is built into EcoStruxure Machine SCADA Expert and is not an add-on. This means that:

- The project runtime server can support a large number of simultaneous thin client connections. Each thin client can view the same or different screens as another thin client.
- The server knows which screen each thin client is viewing and automatically "pushes" any tag value changes to the thin client, thereby eliminating the need for browser refreshes.

- The server can support run-time language switching for each thin client, which means that one thin client can display a screen in English while another thin client can display the same screen in Spanish.
- The project can be configured to support redundant data and web servers with automatic switch-over.

### Thin Client Licensing

The maximum number of simultaneous thin client connections depends on settings of the license installed on the project runtime server. The user does not need to install any license on the thin client.

# Thin Clients

## *The Underlying Technology*

In a EcoStruxure Machine SCADA Expert project, there are several components used to implement the Thin Client capability.

These components are:

**Data Server**

The Data Server is built-in to the EcoStruxure Machine SCADA Expert runtime. The Data Server has direct access to the EcoStruxure Machine SCADA Expert Project Tags Database (runtime) and is responsible for working with ISSymbol to make sure any Tag data being displayed on a Web page at any Thin Client is updated with the latest value(s).

EcoStruxure Machine SCADA Expert can support a backup or secondary Data Server that will be used should the Primary Data Server become unavailable. The Thin Client will automatically switch over to the Secondary without user intervention required.

**Web Server**

The Primary Web Server is responsible for providing Web pages on demand (i.e., when requested by the Client) through navigation to various project screens by the Thin Client. The Web Server communicates with the Thin Client via HTTP protocol over TCP/IP. SSL (Secure Socket Layer) encrypted communications can be enabled. The Web Server does not need to reside on the same PC as the EcoStruxure Machine SCADA Expert runtime project. In fact, the Web Server could be a non-Windows corporate Web Server. However, the Web Server needs to have access to the HTML files that are the project Web pages.

EcoStruxure Machine SCADA Expert supports a Secondary Web Server that will be automatically switched to (by the Client) in case the Primary Web Server becomes unavailable.

**Web Browser**

The Web Browser is located on the Thin Client PC and provides the graphical interface function with the user. Web pages (HTML) is passed to the browser via demand ("pull") and data is "pushed" to the browser by the Data Server whenever a Tag or Tags referenced on the Screen displayed on the Web Client is updated in the Tag Database.

**ISSymbol**

ISSymbol is a Schneider Electric-provided ActiveX Control that facilitates the interaction between the browser on the Web Client and the Web Server as well as the Data Server. There are ISSymbol ActiveX Control versions for Windows and all Windows Embedded platforms.

The ISSymbol ActiveX Control is used for both the Internet Explorer-based and Secure Viewer-based browsers.

**Web Tunneling Gateway**

The (Primary) Web Tunneling Gateway is a bridge between the Web Server and the Data Server that is used in one of two situations. The first is whenever data security is required (e.g., EcoStruxure Machine SCADA Expert data exchanged with the Thin Client needs to be encrypted). The second situation is when the Data Server is "hid" behind a corporate firewall, and only the Web Server IP address (or URL) is exposed.

EcoStruxure Machine SCADA Expert supports a backup (Secondary) Web Tunneling Gateway to be used if the Primary Web Tunneling Gateway becomes unavailable. The Thin Client will automatically switch over to the secondary Web Tunneling Gateway.

The Web Tunneling Gateway is automatically installed when EcoStruxure Machine SCADA Expert 2020 is installed on your PC if the installation program detects that IIS is present.

**Note:**

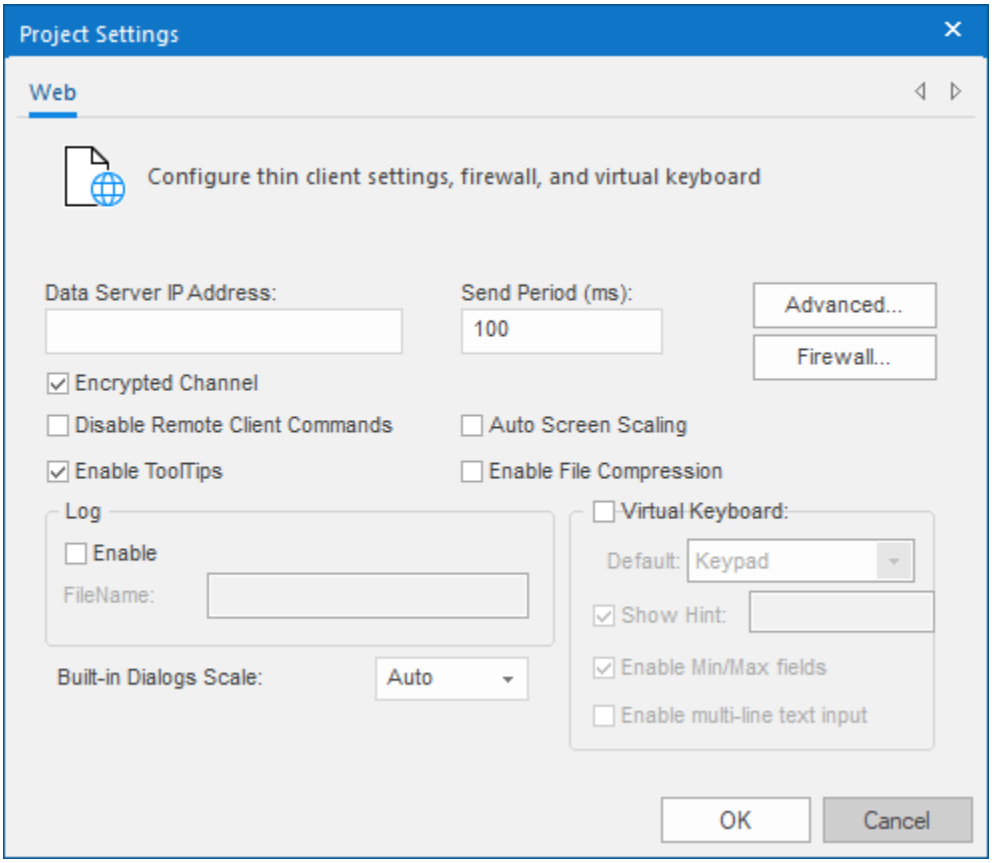
- The Web Tunneling Gateway is automatically installed if IIS is detected during the installation process. Otherwise, it must be manually installed.

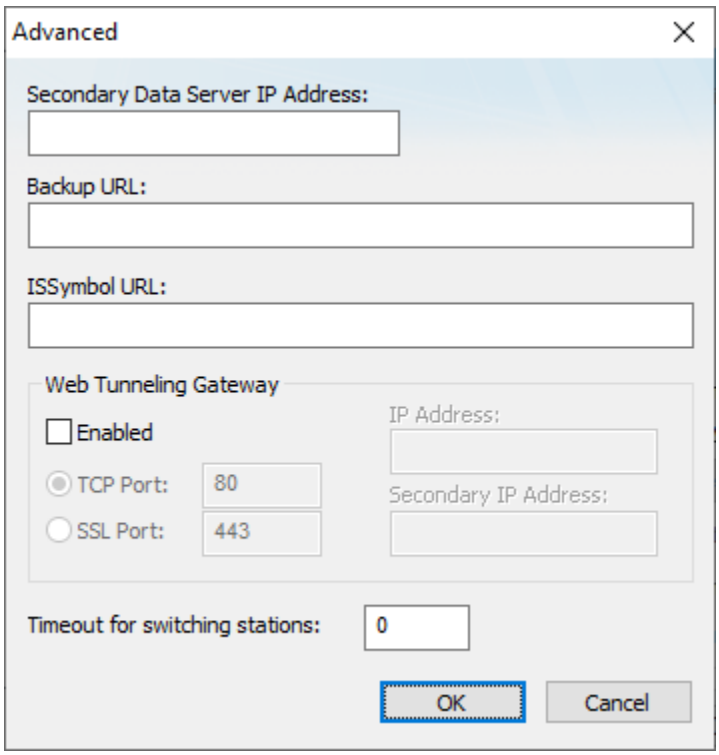
- The main function of the Web Tunneling gateway is to encapsulate data packets in HTTP or HTTPS for communication through a firewall.

### *Examples of Client/Server Architecture*

This section describes some example architectures applied for web-based solutions and provides information on how to configure the project for each architecture. This section does not describe all possible architectures, but it provides the concepts necessary to design and configure different scenarios based on the basic architectures illustrated below

The Web Settings are configured by the Web tab of the Project Settings dialog. To open this dialog: on the Project tab of the ribbon, in the Web group, click **Thin Client**. By pressing the Advanced button, you access additional settings. The following pictures illustrate these dialoges:






The following table describes the meaning of the main Web settings illustrated in the above dialoges:

| Setting | Description |
|---|---|
| Data Server IP Address | When the Web Tunneling Gateway is **disabled**: The Thin Client Control (ISSymbol) uses the Data Server IP Address to connect to the EcoStruxure Machine SCADA Expert TCP/IP Server Task.<br>When the Web Tunneling Gateway is **enabled**: The Web Tunneling Gateway uses the Data Server IP Address to connect to the EcoStruxure Machine SCADA Expert TCP/IP Server Task. |
| Secondary Data Server IP Address | Same as the Data Server IP Address. However, the Secondary IP Address is used only when the connection with the Data Server IP Address fails. |
| Web Tunneling Gateway IP Address | The Thin Client Control (ISSymbol) uses the Web Tunneling Gateway IP Address to connect to the Web Tunneling Gateway. |
| Web Tunneling Gateway Secondary IP Address | Same as the Web Tunneling Gateway IP Address. However, the Web Tunneling Gateway Secondary IP Address is used only when the connection with the Web Tunneling Gateway IP Address fails. |

The Secondary addresses can be used in the following scenarios:

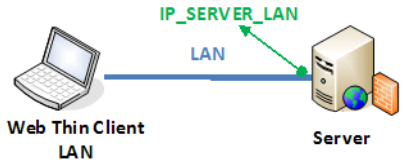
- When the Thin Clients can connect to either one of two redundant Servers (Web or Data); or
- When the Thin Clients can connect to the Server through the Intranet (LAN – Local Area Network) or through the Internet (WAN – Wide Area Network). In this case, the Primary addresses should be configured based on the network used more often by the Thin Clients. In the following examples, the LAN addresses are used as Primary and the WAN addresses are used as Secondary.

The following table describes the meaning from some terms used in the next examples:

| Term | Description |
|---|---|
| LAN | Local Area Network (for example, Intranet) |
| WAN | Wide Area Network (for example, Internet) |
| Server | Station where the following components are running: |

| Term | Description |
|---|---|
| | • EcoStruxure Machine SCADA Expert (TCP/IP Server task)<br>• Web Server (for example, Internet Information Services from Microsoft – IIS)<br>• Web Tunneling Gateway for IIS (if enabled)<br>Although EcoStruxure Machine SCADA Expert does not need to run in the same station where the other components are running, the following examples assume that it is. |
| Thin Client LAN | Thin Client station (Web Browser + ISSymbol control) that connects the Server via the LAN. |
| Thin Client WAN | Thin Client station (Web Browser + ISSymbol control) that connects the Server via the WAN. |
| IP_SERVER_LAN | IP Address of the Server on the LAN. |
| IP_SERVER_WAN | IP Address of the Server on the WAN. |
| IP_ROUTER_LAN | IP Address of the Router on the LAN. |
| IP_ ROUTER_WAN | IP Address of the Router on the WAN. |
| ScreenName | Name of the project screen, saved as HTML, that is open on the Thin Client station. |

### Example 1: Web Server and Thin Client in the same Intranet (LAN)




This is the very common architecture, as well as the simplest to configure. In this architecture, both the Web Server (e.g., Microsoft IIS) and the Data Server (i.e., the EcoStruxure Machine SCADA Expert TCP/IP Server module) are running on the same PC. The Thin Client connects to the Web Server to download the HTML screen file(s). Then it connects to the Data Server to exchange data with the EcoStruxure Machine SCADA Expert runtime project. Since both the Thin Client and the Server station are connected to the same network, the Thin Client can access the Server station directly through its IP address (or host name).

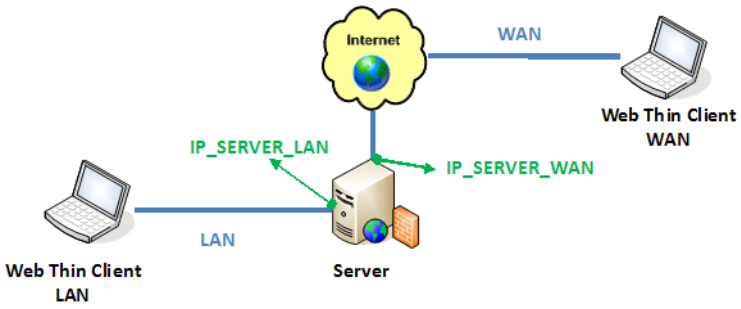
**Configuration**:

| Setting | WTG Enabled | Web Gateway Disabled |
|---|---|---|
| Data Server IP Address | IP_SERVER_LAN | IP_SERVER_LAN |
| Secondary Data Server IP Address | - | - |
| Web Tunneling Gateway IP Address | IP_SERVER_LAN | - |
| Web Tunneling Gateway Secondary IP Address | - | - |

**Note:**

- URL From Thin Client LAN: `http://IP_SERVER_LAN/ScreenName.html`
- Home directory of the web server (HTTP server) on the server station: `Web` sub-folder of the project

### Example 2: Web Server with Intranet (LAN) and Internet (WAN) Connections




This architecture has both the Web Server (e.g., Microsoft IIS) and the Data Server (i.e., the EcoStruxure Machine SCADA Expert TCP/IP Server module) running on the same PC. Thin Clients can connect to the Server through either an Intranet (LAN) connection to the Server or an Internet (WAN) connection to the Server (e.g., two different Ethernet ports).

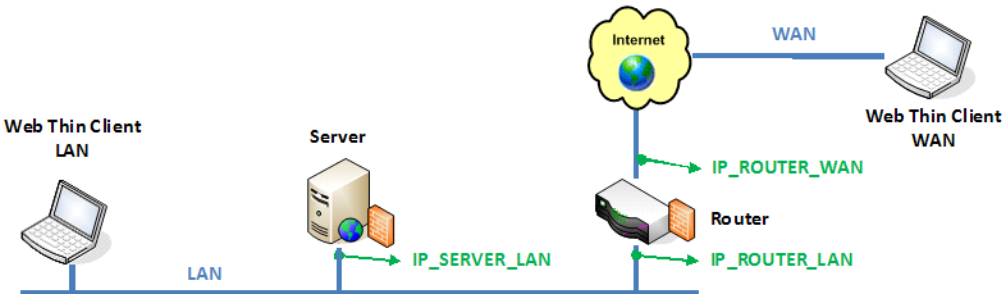
**Configuration**:

| Setting | Web Gateway Enabled | Web Gateway Disabled |
|---|---|---|
| Data Server IP Address | IP_SERVER_LAN | IP_SERVER_LAN |
| Secondary Data Server IP Address | IP_SERVER_LAN | IP_SERVER_WAN |
| Web Tunneling Gateway IP Address | IP_SERVER_LAN | - |
| Web Tunneling Gateway Secondary IP Address | IP_SERVER_WAN | - |

**Note:**

- URL From Thin Client LAN: `http://IP_SERVER_LAN/ScreenName.html`
- URL From Thin Client WAN: `http://IP_SERVER_WAN/ScreenName.html`
- Home directory of the Web Server (HTTP server) on the Server station: `Web` sub-folder of your project folder
- You must assign a Fixed IP address to the Web Server on the Internet (WAN), and the project must be running in this Server. Consult your ISP provider or IT department for further information about how to get a Fixed IP address for your Server.

### Example 3: Web Server with Intranet (LAN) and Router Internet (WAN) Connections




This architecture has both the Web Server (e.g., Microsoft IIS) and the Data Server (i.e., the EcoStruxure Machine SCADA Expert TCP/IP Server module) running in the same PC. Thin Clients can connect to the Server through either an Intranet (LAN) connection or an Internet (WAN) connection. There is a Router between the Intranet (LAN) and the Internet (WAN).

**Configuration**:

| Setting | Web Gateway Enabled | Web Gateway Disabled |
|---|---|---|
| Data Server IP Address | IP_SERVER_LAN | IP_SERVER_LAN |
| Secondary Data Server IP Address | IP_SERVER_LAN | IP_ROUTER_WAN |
| Web Tunneling Gateway IP Address | IP_SERVER_LAN | - |
| Web Tunneling Gateway Secondary IP Address | IP_ROUTER_WAN | - |

**Note:**

- URL From Thin Client LAN: **http://IP_SERVER_LAN/ScreenName.html**
- URL From Thin Client WAN: **http://IP_ROUTERR_WAN/ScreenName.html**
- The Router must be configured to forward the TCP Port(s) from its public IP (IP_ROUTER_WAN) to the Server private IP (IP_SERVER_LAN).

  If the Web Gateway is **enabled**, only the HTTP Port (80, by default) or the HTTPS Port (SSL Port 443, by default) must be forwarded from IP_ROUTER_WAN to the IP_SERVER_LAN.

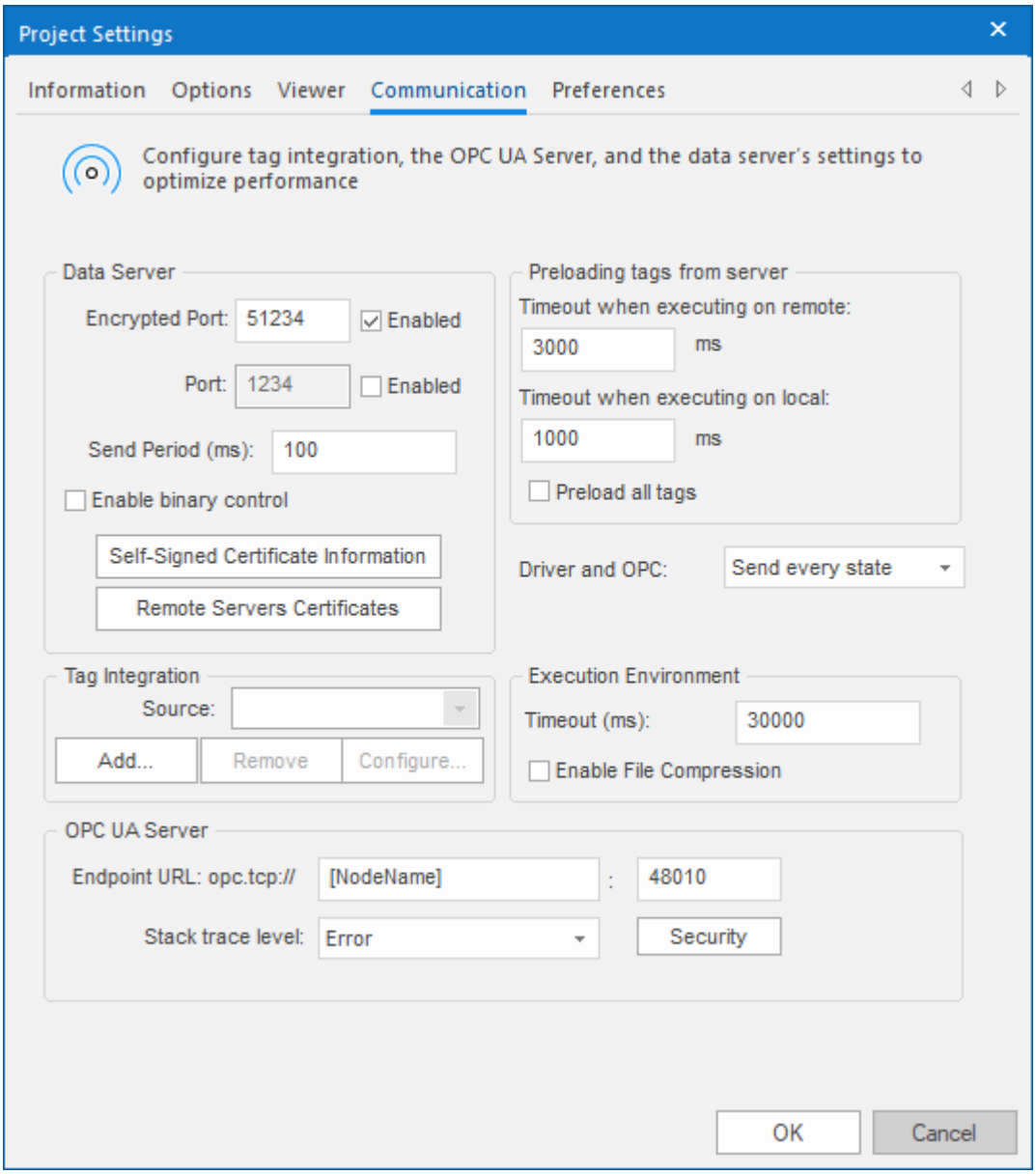
  If the Web Gateway is **disabled**, both the HTTP Port (80, by default) and the Studio TCP/IP Server Port (1234, by default) must be forwarded from IP_ROUTER_WAN to the IP_SERVER_LAN. Consult the Router documentation for further information about how to configure Port Forwarding on it.
- Home directory of the Web Server (HTTP server) on the Server station: `Web` sub-folder of your project folder
- You must assign a Fixed IP address to the Router on the Internet (WAN), and the project must be running in this Server. Consult your ISP provider or IT department for further information about how to get a Fixed IP address for your Server.

## *Configuring the Data Server*

EcoStruxure Machine SCADA Expert has a couple dialoges that are used for configuration of the Data Server and the Web Server configuration to be used. The Data Server is part of the project runtime and uses the TCP/IP Server module.

### Communication Settings dialog configuration

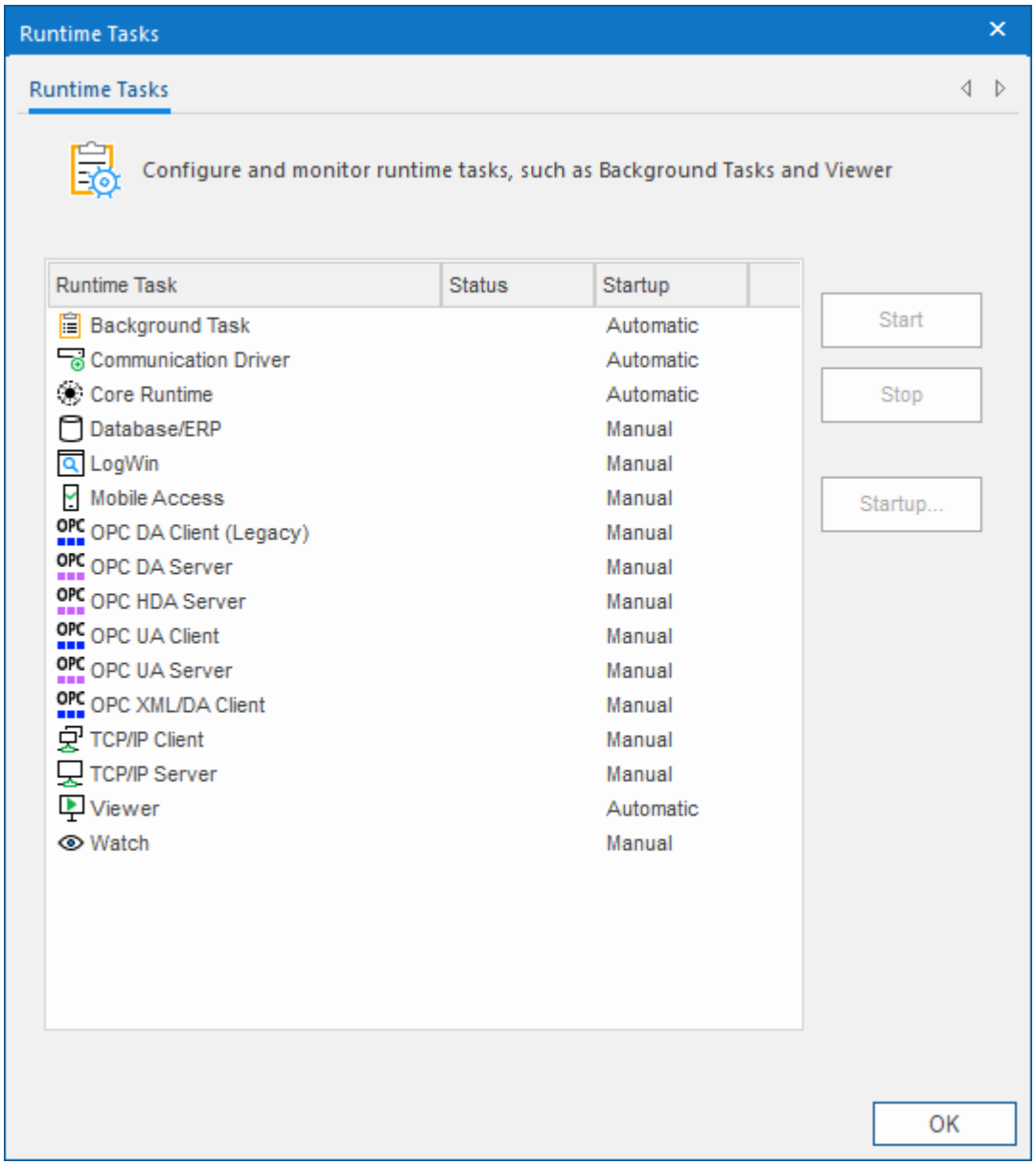
1. On the Project tab of the ribbon, in the settings group, click **Communication**.




2. Enter the Port number (1234 is the default) for the Data Server. You can also define the Data Send Period (i.e., time period for updated communication of data values to the Web Client).
3. Optionally enable **Binary Control** of the data. It is more secure, but is slower. The default is disabled.

### Enable the TCP/IP task

1. On the Home tab of the ribbon, click **Tasks** (local or remote, depending on the project's target system).




2. Be sure the TCP/IP Server is set to Automatic. This should be the default state, but can be manually configured by selecting the **Startup** button.
3. Be sure the TCP Port number is properly set (see Communication Settings above), otherwise the TCP/IP Server will start then stop.

## *Configuring a web server to host your project pages*

As part of deploying your project over the Web, you must configure a web server to host your project screens.

You are not required to use a Windows PC or a Windows Embedded device to host your project pages. The pages are essentially static files waiting to be downloaded; all runtime processing is handled by the project viewer (i.e., Internet Explorer with ISSymbol installed, or Secure Viewer) on the Thin Client. As such, you can use any standards-compliant web server on any computer platform to host your pages.

For example, if you already have a Unix-based intranet server, then you can copy your project's `Web` sub-folder (or whatever folder in which you've saved your project pages) to the server and have your Thin Clients point to that server's address.

Please note, however, that the web server you choose may not be robust enough to serve your project in a production environment and/or it may not support all features of EcoStruxure Machine SCADA Expert 2020. If you want to use these features, then in most cases you should use Microsoft IIS as described below.

Before you install and configure any software, please review its documentation thoroughly.

> **Tip:**
>
> A web server typically runs on, or "listens to", a computer's TCP/IP port 80. Only one process can run on a given port, however, so if another process on your computer — for example, some third-party SCADA software — is already running on port 80, then it and the web server process might conflict with each other. You must either configure one of the processes to run on a different port or use Task Manager to end the conflicting process. If you cannot identify the conflicting process, open Command Prompt and enter the following command to get a list of all networking processes:
>
> ```
> netstat -a -o
> ```

### NTWebServer and CEWebServer

NTWebServer and CEWebServer are lightweight, zero-configuration web servers that are included free with EcoStruxure Machine SCADA Expert 2020. You can use them to demonstrate your project and run basic tests without making the financial and technical investment in a full-featured web server. They are not robust, however, and they do not support Mobile Access or Web Tunneling Gateway. For real-world applications, we recommend you use Microsoft IIS as described below.

NTWebServer can be run on any supported Windows computer, and it can be found in your EcoStruxure Machine SCADA Expert 2020 program folder at:

```
C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin
\NTWebServer.exe
```

Copy NTWebServer to your project's Web sub-folder, and then run it. It must be located in the same folder as the pages it will serve, so that after you run it, that folder becomes the "home" directory of the website.

> **Note:**
>
> By default, NTWebServer runs on port 80. To run it on another port, open Command Prompt and enter the following command:
>
> ```
> NTWebserver.exe p <port number>
> ```
>
> You can run several instances of NTWebServer on the same computer, as long as each instance runs on a different port and does not conflict with any other processes for control of that port.

CEWebServer can be run on any supported Windows Embedded device, and it can be found in your EcoStruxure Machine SCADA Expert 2020 program folder at:

```
C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Redist
\<Windows Embedded version>\<device processor>\CEWebServer.exe
```

If you copy CEWebServer to your project's Web sub-folder before your download your project to the device, it should be downloaded with the rest of the project files. If not, you can manually copy it to the device after the fact. Either way, make sure it is located in your project's Web sub-folder, and then run it. Again, that folder becomes the "home" directory of the website.

Both NTWebServer and CEWebServer must run as normal programs; they cannot run as Windows services.

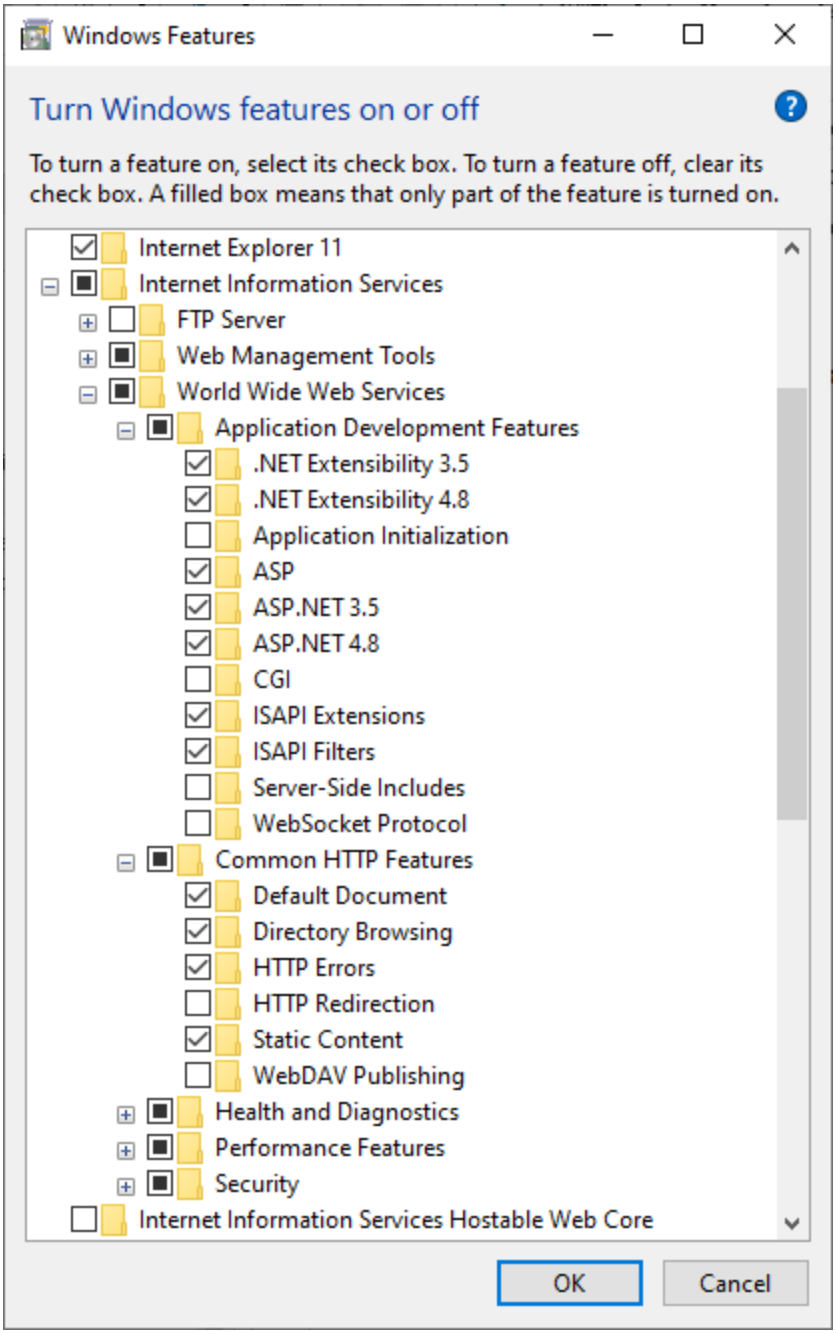
### Microsoft IIS

Internet Information Services (IIS) is the full-featured server software that is included in most versions of Windows and Windows Server. It supports all features of EcoStruxure Machine SCADA Expert 2020, and it is robust enough to serve almost any project in a production environment. It is the web server that we recommend for most users. To properly install and configure it, however, you should be experienced with administering Windows on a network.

Please note that these instructions apply only to the following versions of IIS:

| Version | ...on Operating System |
|---|---|
| IIS 8.5 | • Windows 8.1<br>• Windows Server 2012 R2 |
| IIS 10 | • Windows 10<br>• Windows Server 2016<br>• Windows Server 2019 |

For the sake of system security, IIS is turned off by default when the operating system is installed. Use the *Windows Features* control panel to turn it on. If you want to use Mobile Access and/or WTG in your project, make sure ASP, ASP.NET, and ISAPI Extensions are also turned on.




***An example of the features selected in Windows 10***

Once IIS is turned on, you can use Administrative Tools to configure it. For more information, please refer to Microsoft's extensive documentation.

### Apache for Windows

If IIS is not available to you or if you choose not to use it, then the second most popular web server for Windows is the open-source Apache. However, it requires even more expertise than IIS to properly install and configure, so please review the documentation thoroughly before you attempt it.

## *Install the Thin Client software*

Install the Thin Client software on a client station in order to let users view your project.

> **Note:** We recommend that you use Mobile Access instead of our traditional Thin Client software whenever possible. Thin Client depends on legacy, Windows-only technologies, while Mobile Access allows you to use any HTML5-compatible browser running on any platform as a project viewer. Mobile Access does not yet support all of the features that Thin Client does, but we are continuing to improve Mobile Access with every new release.

If you have already installed either the full EcoStruxure Machine SCADA Expert 2020 software or one of the runtime editions on the computer or device that you want to use as a client station, you may skip this procedure because you do not need to install the Thin Client software on the same computer or device. The full software and the runtime editions include the same components as the Thin Client software, except that they are preconfigured to view the project that is running locally.

Before you begin this procedure, you should install the full EcoStruxure Machine SCADA Expert 2020 software on at least one Windows computer — typically, on your project development workstation — because doing so also unpacks the Thin Client software installer.

To run the Thin Client software installer, you must have a computer or device with a network connection and one of the following operating systems:

- Windows:
  - Windows 10, version 1803 or later (including LTSC/LTSB versions)
  - Windows 8.1
- Windows Server:
  - Windows Server 2019
  - Windows Server 2016
  - Windows Server 2012 R2

You must also have Administrator privileges on the computer or device in order to install any software.

The Thin Client software is based on ISSymbol, which is an ActiveX control that we developed to open project screens and exchange data (e.g., tag values) with the project runtime. It acts as a control layer between the client and the server, similar to the Java Virtual Machine for Java-based applications, and it provides a high level of security because it does not allow the project to access the operating system on the client station.

To install the Thin Client software:

1. Locate the Thin Client software installer in your EcoStruxure Machine SCADA Expert 2020 program folder.

   If EcoStruxure Machine SCADA Expert 2020 was installed at its default location on your computer, the Thin Client software installer should be located at:

   ```
   C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
    2020\Redist\WebAddOn\ThinClient\ThinClientSetup.exe
   ```

2. Copy the Thin Client software installer to the computer or device on which you want to install the software. Assuming the computer or device has a network connection — which it should, if you plan to use it as a project viewer — you can simply copy the installer across the network. Otherwise, copy the installer to removable media (e.g., a USB flash drive) and then carry it to the computer or device.
3. On that computer or device, run the Thin Client software installer (`ThinClientSetup.exe`). The first page of the installation wizard is displayed.
4. Click **Next**.

The next page of the wizard is displayed.

5. On the **Customer Information** page, type your name and your company name, and then click **Next**.
The next page of the wizard is displayed.
6. On the **Choose Destination Location** page, select the folder where the software should be installed, and then click **Next**.

   By default, the software will be installed at:

   ```
   C:\Program Files (x86)\Schneider Electric\Thin Client\<ID string>\
   ```

   The next page of the wizard is displayed.
7. On the **Select Features** page, select the specific features and components that you want to install, and then click **Next**.

| Feature | Description |
|---|---|
| **Program Files** | The main program files for the thin clients. This feature cannot be deselected. |
| **Secure Viewer** | Creates shortcuts in the Start menu and on the desktop. If you deselect this feature, the program files will still be installed but the shortcuts will not be created. You will need to locate the Secure Viewer program (`Viewer.exe`) and then manually run it. |
| **PDF Printing** | Additional software that allows the project to save run-time reports as PDF files. |
| **Security System Device Driver** | An additional keyboard driver that enforces project security during run time by controlling user input. |

   The next page of the wizard is displayed.
8. On the **Ready to Install the Program** page, click **Install**.
The software is installed, and then when the installation is finished, the last page of the wizard is displayed.
9. Click **Finish** to close the installation wizard.

Once the Thin Client software is installed, you may choose which type of thin client to use:

- If you choose to use Secure Viewer as a standalone program, you must configure it before you can run it. For more information, see Configure and run Secure Viewer on page 742.

The Thin Client software itself does not need to be licensed on any computer or device. The license for the project runtime determines the number of thin clients that are allowed to connect to it at the same time. For more information, see License Settings on page 45.

## INSTALL THE CUSTOM WIDGET FRAMEWORK ON A CLIENT STATION

If your project screens include custom widgets, you might need to install Custom Widget Framework on some client stations to enable them to properly display the widgets.

This task applies only to stations on which you have already installed either Thin Client or Embedded HMI — in other words, stations that are using the Thin Client software to view your project.

Stations that are viewing your project through Mobile Access do not need to have Custom Widget Framework installed, because custom widgets are HTML5-based screen objects that can be displayed normally in the web browser.

Before you begin this task, you must have installed the full SCADA software on at least one Windows computer — typically, on your project development workstation — because doing so also unpacks the Custom Widget Framework installer. (Custom Widget Framework is not included in the Thin Client installer because it would greatly increase the file size of that installer, for a feature that not all projects use.)

You must have Administrator privileges on a computer or device in order to install any software.

To install the Custom Widget Framework on a client station:

1. Locate the Custom Widget Framework installer (`CustomWidgetFrameworkSetup.exe`) in your SCADA program folder.

   If SCADA was installed in its default location, the Custom Widget Framework installer should be located at:

   ```
   C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
    2020\Redist\CustomWidgetFramework\CustomWidgetFrameworkSetup.exe
   ```

2. Copy the installer to the client station, either over the network or on a portable hard drive.
3. Run the installer. You might need to do this as a user with Administrator privileges: right-click the installer, and then on the shortcut menu, click **Run as Administrator**.
4. Follow the installer's instructions. On the *Choose Destination Location* page of the installer, make sure the Bin folder of either Thin Client or Embedded HMI is selected. If it is not, click **Browse** and then use the file browser to locate and select the folder.

### *Configure and run Secure Viewer*

After you have installed Secure Viewer on a Windows computer or Windows Embedded device, you can configure and run it as a project thin client.

Before you begin this task, you must have already installed Secure Viewer on the computer or device as part of the full EcoStruxure Machine SCADA Expert 2020 software, the thin client software, or the project runtime software. For more information, see Installation Guide on page 33.

Also, you must have Administrator privileges on the computer or device in order to run the Secure Viewer configuration utility.

Finally, this task assumes that you have properly developed and deployed your project for network access, that the project itself is running, and that you have the information you will need to configure Secure Viewer (e.g., the IP address or hostname of the project runtime server). For more information, see Thin Clients on page 729.

To configure and run Secure Viewer:

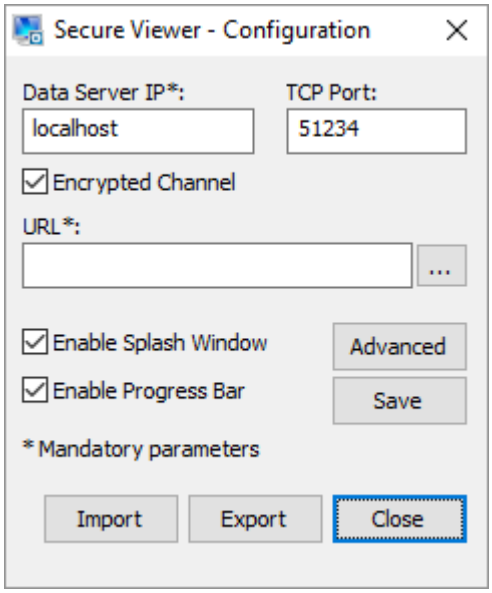
1. Locate and run the Secure Viewer configuration utility (`ViewerCfg.exe`).

   If you installed the full EcoStruxure Machine SCADA Expert 2020 software at the default location on a Windows computer or Windows Embedded device, the Secure Viewer configuration utility should be located in the EcoStruxure Machine SCADA Expert 2020 program folder at: `C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin\ViewerCfg.exe`

   If you installed the thin client software at the default location on a Windows computer or Windows Embedded device, the Secure Viewer configuration utility should be located in the Thin Client program folder at: `C:\Program Files (x86)\Schneider Electric\Thin Client\`*`ID string`*`\ViewerCfg.exe`

   > **Note:** To run properly, the Secure Viewer configuration utility must be run with Administrator privileges. It should be installed with those privileges by default, but if you have problems, check the file properties for `ViewerCfg.exe` and make sure the **Run as Administrator** option is selected.

The configuration utility window is displayed.




***Secure Viewer configuration utility***

2. In the **Data Server IP** and **TCP Port** boxes, type the host name or IP address and port number of the data server (a.k.a. the project runtime server).

   By default, the **Encrypted Channel** option is selected and the port number is 51234. Make sure these settings match the corresponding project settings. For more information, see Communication tab on page 122.

3. In the **URL** box, type the URL of the project file (`<project name>.APP`) on the project runtime server.

   The project file must be accessible via either the local file system, network file sharing, or a properly configured web server, and the syntax of the URL depends on which it is.

4. Configure the other settings as needed.

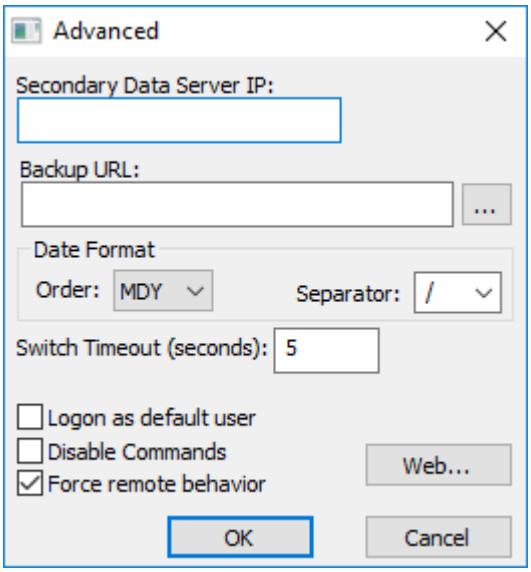
   > **Note:** Configuring these settings is optional. In most cases, the default settings should suffice.

   a) Select the **Enable Splash Window** check box to display a splash window when Secure Viewer is run.

   b) Select the **Enable Progress Bar** check box to display a progress bar while Secure Viewer loads the project file.

   c) Click **Advanced**.

The *Advanced* dialog box is displayed.




***Secure Viewer advanced settings***

d) In the **Secondary Data Server IP** box, type the IP address (or hostname) of the secondary data server, if any.

The secondary data server is another computer that is hosting the same project in parallel with the primary data server. If Secure Viewer loses its connection to the primary data server, it will automatically attempt to connect to the secondary data server.

e) In the **Backup URL** box, type the URL of the backup project file.

f) In the **Date Format** area, select the order and separator for the date format.

For more information, see About the date format and how to change it on page 694.

g) In the **Switch Timeout** box, type the number of seconds that Secure Viewer should wait before it attempts to connect to the secondary data server, in the event that it becomes disconnected from the primary data server.

h) Select the **Log on as default user** check box to have Secure Viewer automatically log on as the default user "Guest", if it is enabled.

This will eliminate the need to enter a specific username and password when Secure Viewer is run, although the user can choose to log off and then log on again with another username. For more information, see Project Security on page 639.

> **Tip:**
>
> You can also change the name of the default user, from "Guest" to something else. To do this, use a text editor to open the Secure Viewer initialization file (`Viewer.ini`) and edit the following setting:
>
> ```
> [Options]
> user=<default user name>
> ```

i) Select the **Disable Commands** check box to prevent Secure Viewer from sending commands (i.e., user input) to the project runtime server.

When this option is selected, Secure Viewer will only display current run-time information received from the server, effectively making the client station a simple, non-interactive viewscreen.

j) Select the **Force remote behavior** check box to force Secure Viewer to behave as if it is running on a remote station, which means that it will keep virtual copies of project tags with Local scope.

This option is relevant only if Secure Viewer is running on the same computer as the project runtime server. If it is, and if this option is not selected, Secure Viewer will synchronize all project tags with the server, regardless of scope. For more information, see Choosing the Tag Scope on page 156.

k) Click **OK** to close the *Advanced* dialog box.

5. In the Secure Viewer configuration utility, review the settings and then click **Save**.

   The configuration is saved as a Secure Viewer initialization file (`Viewer.ini`). This file should always be in the same folder as the Secure Viewer program.

   > **Tip:** Once you have a properly configured initialization file, you can reuse it with other installations of Secure Viewer.

6. Click **Close** to close the Secure Viewer configuration utility.
7. Run the Secure Viewer program by doing one of the following:
   - Double-click the Secure Viewer program itself (`Viewer.exe`), which should be located in the same folder as the Secure Viewer configuration utility; or
   - Double-click the **Secure Viewer** shortcut on your desktop, if you installed the Thin Client software.

   The Secure Viewer program window is displayed, and if the program has been properly configured, it connects to the project runtime server and loads the project file.
8. If the Secure Viewer program is connecting to the project runtime server via encrypted channel, you might be prompted to verify the server certificate — in the *Verify Certificate* dialog box, do one of the following:
   - To verify the certificate for the current session only, click **Yes**.
   - To verify the certificate for all sessions, click **View Certificate** and then follow the instructions to install the certificate in the Windows certificate store.

   If you choose to install the certificate and then accept the default settings, it will be installed in the following store:

   ```
   Current User\Intermediate Certification Authorities\Certificates
   ```

   For more information about installing and managing certificates, see Windows help.

When you are done with this task, you can use Secure Viewer to access the project as you normally would.

As part of securing the project thin client, you can configure the computer or device to automatically run Secure Viewer on startup and then not allow the user to exit the program or switch to other programs. For more information, contact your system administrator.

## CUSTOMIZE THE VIEWER PROGRAM ICON IN THE TASKBAR

You can customize the Viewer program icon that appears in the Windows taskbar simply by copying your custom icon into the program folder.

This customization works both for the local Viewer module that is part of the project runtime software and for the standalone Secure Viewer program that is part of the thin client software. It is primarily intended for the latter, however, because that is the program most users see.

You might want to customize the Viewer program icon if, for example, you are a system integrator and you want to provide a branded, turnkey solution to your customers. Customizing the icon can help to maintain a common look and feel.

This task assumes you already have the custom icon that you want to use. It should be a standard Windows ICO file (`*.ico`).

To customize the Viewer program icon, copy the ICO file into the same folder that contains the Viewer program (`Viewer.exe`), and then rename the ICO file to `Viewer.ico`.

The location of the Viewer program varies:

- For the standalone Secure Viewer program running in Windows, it is typically located at:

  **C:\Program Files (x86)\Schneider Electric\Thin Client\BBBE2E0F-084D-484b-AFDF-EA12BF0E52FF\Viewer.exe**

- For SCADA running in Windows, it is typically located at:

  **C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin\Viewer.exe**

The next time the Viewer program is run, it will automatically get the custom icon and display it in the taskbar.

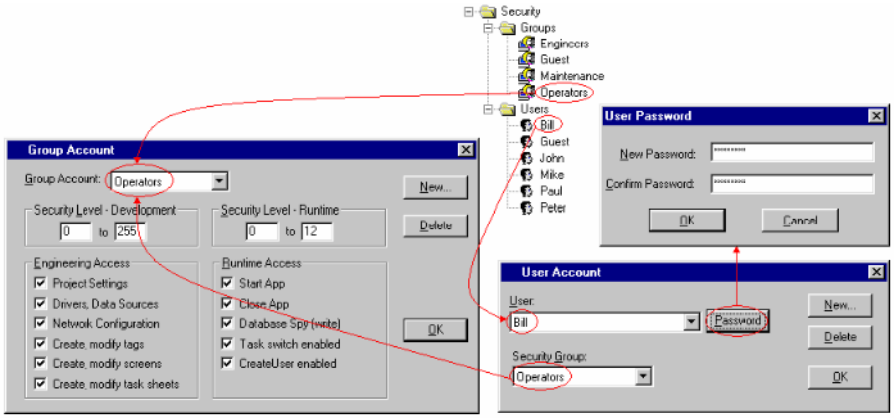
*Customizing the icon: before and after*

## *Implementing Security for Web-based Applications*

There are various methods for implementing security of Web-based applications. The approach that you require can depend on a number of factors, and may involve one or more methods of implementing Security.

### Method 1: Password Protection

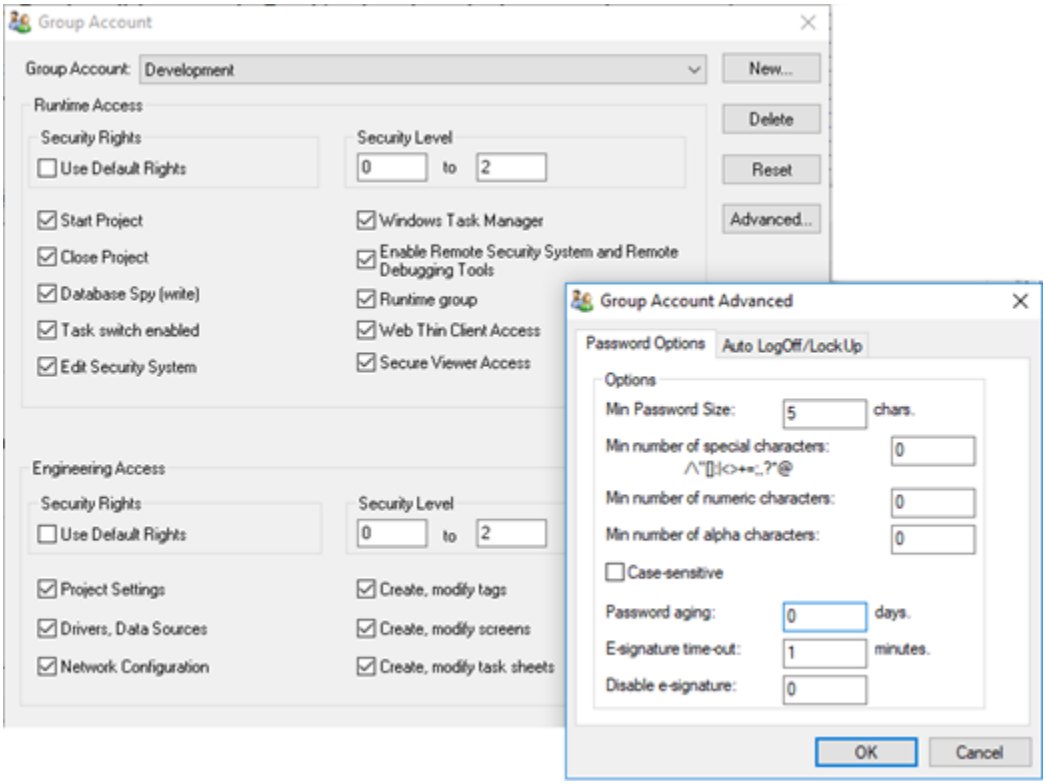
EcoStruxure Machine SCADA Expert provides the ability to create Groups of Users and individual Users within a Group. Each Group (e.g., Operators, Supervisors, Maintenance) can have different security levels and access different levels of functionality. Individual passwords can be configured for each User.



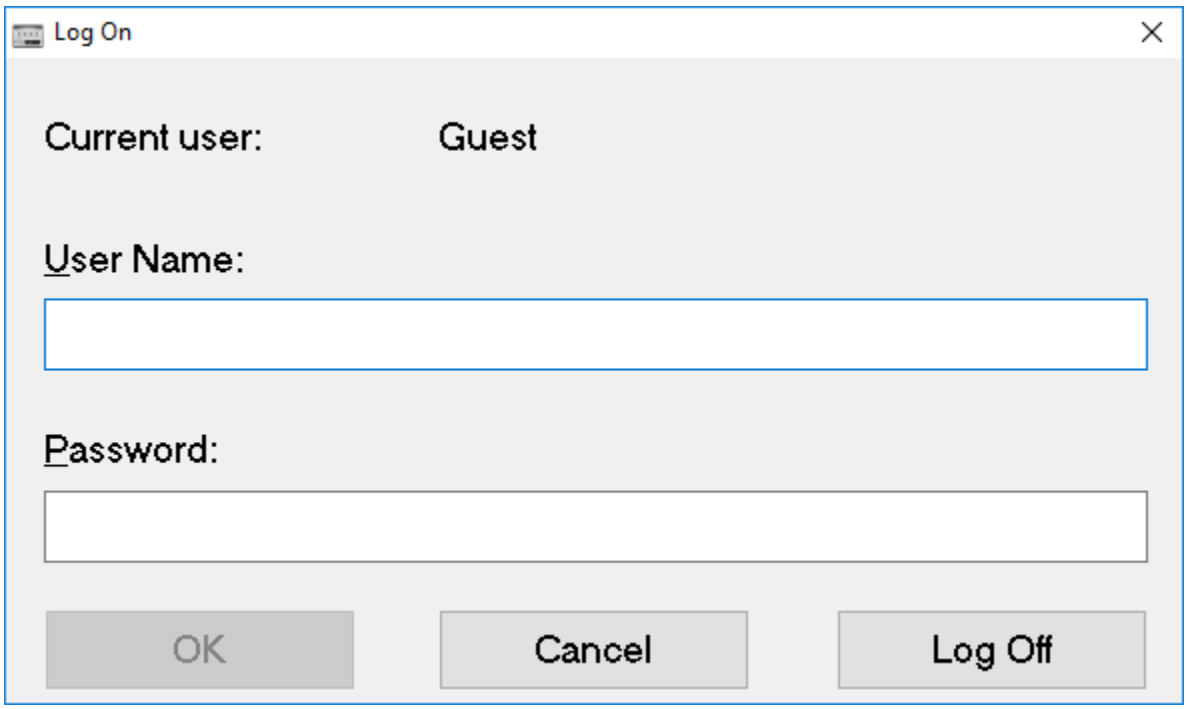

*Security Groups and Users*

In addition, Groups can have advanced settings, allowing features like minimum password size, password aging, e-signature on Objects with Command animations, Account Auto-lockup (e.g., lock up after a number

of invalid attempts to access), and User Account blocking (temporarily disable – e.g., when employee is on vacation).

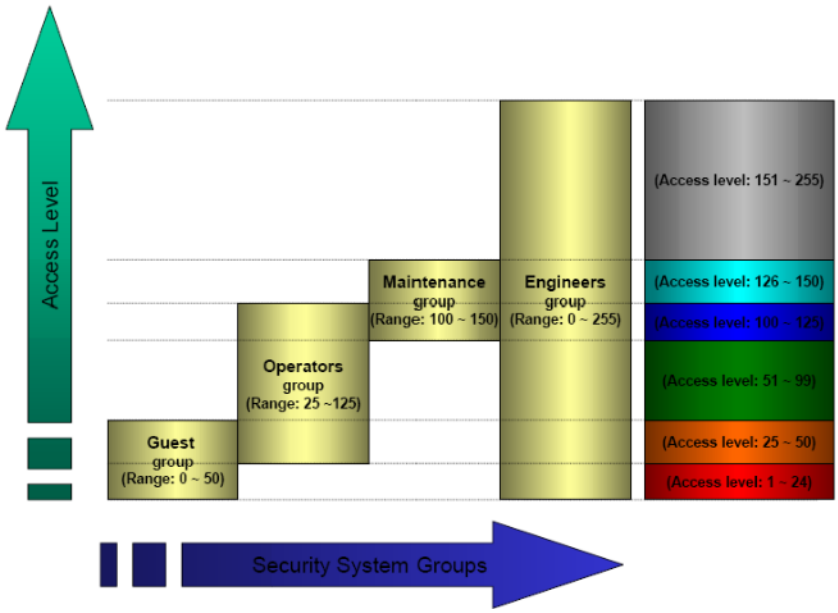


If System Security is enabled, these Password Protection features are also available at the Thin Client station. When a User at a Thin Client station attempts to connect to the Web Server, they will be prompted for a User Name and a Password. If either is invalid, the User will not be let on to the system.




***Log On dialog***

Within a project, the various screen objects and their animations, and Screen access can have a security level assigned to it. The current User logged on must have a access level range which matches the desired Object or Screen. The following is a representative method of assigning security access levels by Group.

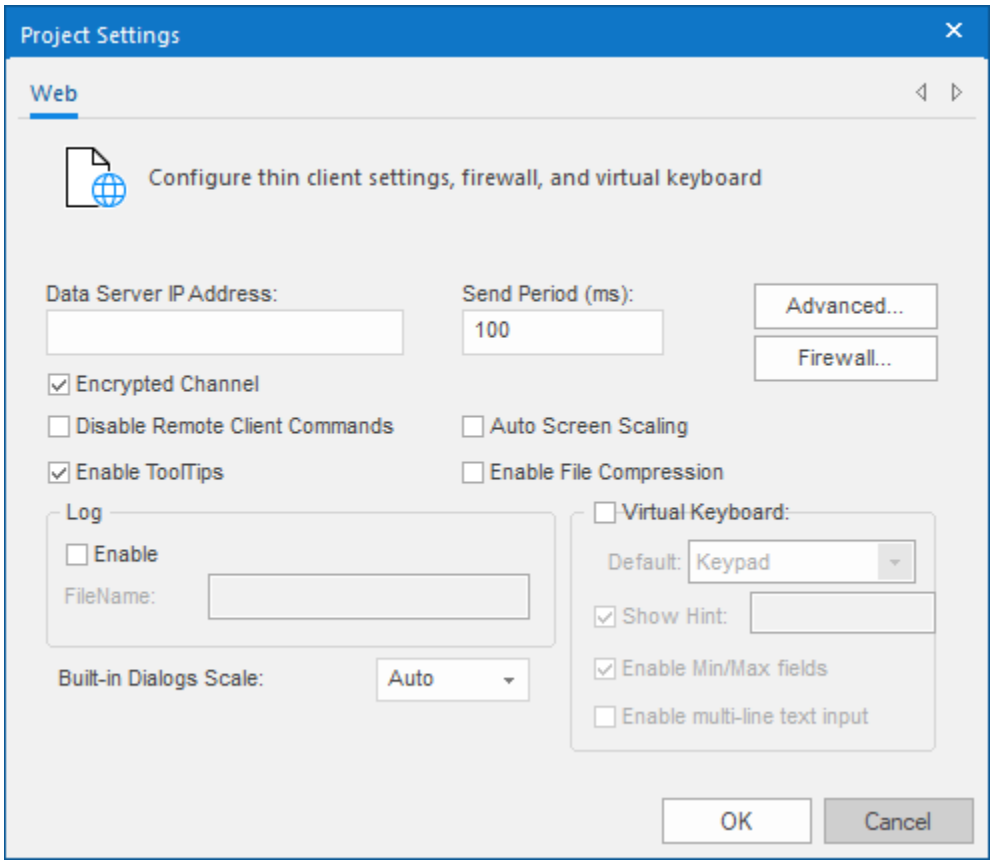



For more information, see Security.

### Method 2: Disabling Thin Client Commands

EcoStruxure Machine SCADA Expert allows bi-directional data exchange between the Thin Client and the Data Server. However, for security reasons it may be advantageous to only allow the Thin Client to view the process or machine data, and not send any data back to the Data Server.

Selecting (checking) the **Disable Remote Client Commands** option in the project settings (**Thin Client** on the Project tab of the ribbon) ensures that all commands coming from a Thin Client station are blocked. The communication becomes unidirectional (from the Server to the Thin Clients):

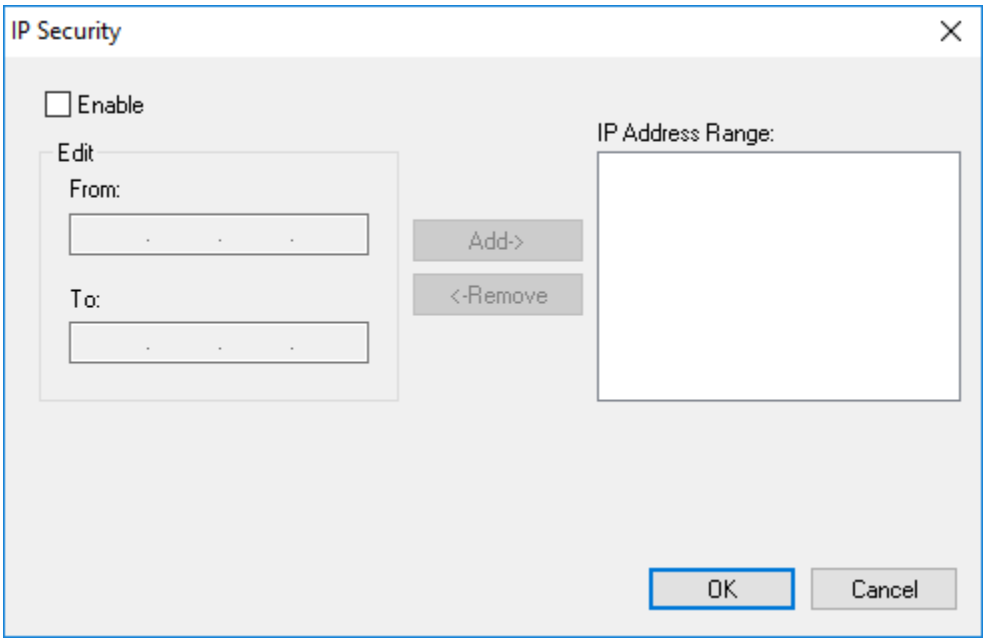



***Project Settings — Web tab***

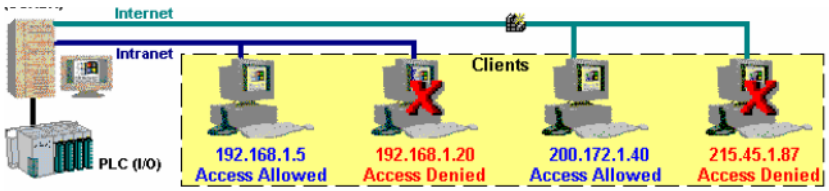
### Method 3: Embedded Firewall

This feature allows the user to filter access to the project based on the Thin Clients IP Address. When a Thin Client attempts to connect to the Server station, the Server checks if the IP Address of the Thin Client station

is authorized to access the project. The ranges of authorized IP Addresses can be configured in the Server station by clicking **IP Security** in the project settings (**Thin Client** on the Project tab of the ribbon):




*IP Security dialog*

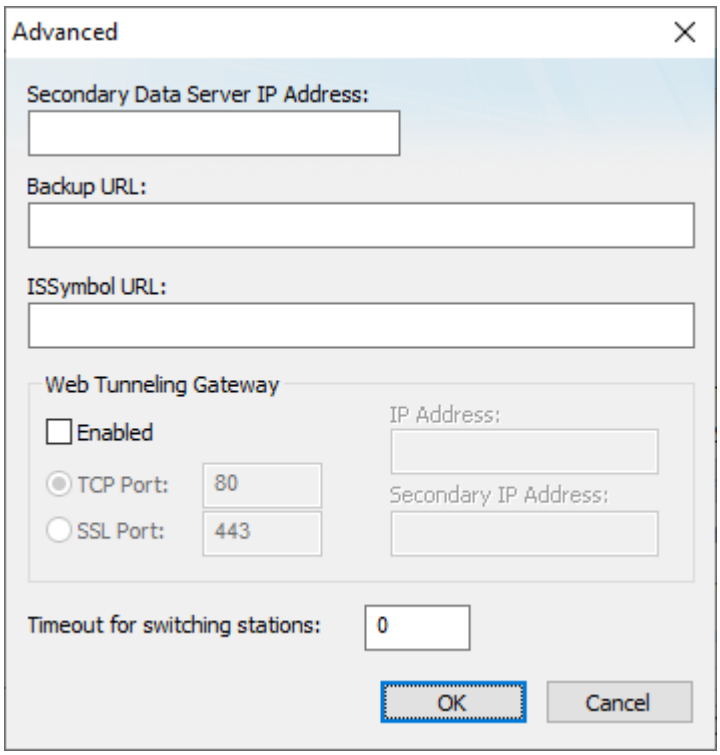



*Access allowed by IP address*

### Method 4: Encrypted Communications (SSL)

By enabling the Web Tunneling Gateway (WTG), you can enable all communications between the Data Server + Web Server and the Thin Client to be encrypted using RC6, a highly-secure 128-bit encryption standard. To use SSL, you must do the following:

1. Click **Advanced** in the project settings (**Thin Client** on the Project tab of the ribbon). Select (check) the **Web Tunneling Gateway Enabled** option. Click on the **SSL** radio button and be sure the SSL port is set to 443. Click **OK**.




***Project Settings — Web — Advanced dialog***

2. In your Web Server, be sure SSL capabilities are enabled and that a SSL Certificate of Authentication is present.
3. Be sure SSL is enabled in the Web Client
4. Set up all other Web configurations to support the WTG.

### Method 5: VPN

A VPN is a Virtual Private Network. It is called virtual since it really uses the public Internet to transport data from one computer to another. But since this network is encrypted and uses other security mechanisms enabled by the ISP, is it a very secure Private Network. While VPN's are inherently secure, they are more costly that a simple public Internet connection.

## *List of network ports used by this software*

This is a list of the various network ports that are used by the SCADA project runtime, its subordinate drivers and utilities, and other related programs.

| Port | Feature or Protocol |
|---|---|
| 20 | FTP Server (Data) |
| 21 | FTP Server (Command) |
| 25 | Simple Mail Transfer Protocol (SMTP), for outgoing mail |
| 80 | Web Server (HTTP), unencrypted |
| 102 | Siemens SIMATIC S7 Protocol, for the SIETH and SITIA drivers |
| 110 | Post Office Protocol version 3 (POP3), for incoming mail |

| Port | Feature or Protocol |
|---|---|
| 118 | Microsoft SQL Server Services |
| 135 | Distributed Component Object Model (DCOM), for OPC DA and OPC HDA |
| 161 | Simple Network Management Protocol (SNMP) |
| 162 | Simple Network Management Protocol (SNMP) Trap |
| 389 | Lightweight Directory Access Protocol (LDAP), unencrypted |
| 443 | Web Server (HTTPS), encrypted using SSL |
| 502 | Modbus TCP/IP, for the MOTCP driver |
| 636 | Lightweight Directory Access Protocol (LDAP), encrypted using SSL |
| 1028 | FTP Client (Command) |
| 1029 | FTP Client (Data) |
| 1234 | SCADA project runtime server (a.k.a. Data Server or TCP/IP Server), unencrypted<br>**Note:** You can change this port number in your project settings. For more information, see Communication tab on page 122. |
| 1433 | Microsoft SQL Server |
| 1434 | Microsoft SQL Server Dedicated Administrator Connection (DAC) |
| 1521 | Oracle |
| 1526 | Oracle |
| 2030 | Oracle |
| 3306 | MySQL (can be configured to use 3306–3309) |
| 3872 | Oracle Management Remote Agent |
| 3997 | Studio Database Gateway (`StADOSvr.exe`)<br>**Note:** You can change this port number in the gateway settings, particularly if you need to run multiple instances of the gateway. For more information, see Manually running Studio Database Gateway on page 828. |
| 4322 | Remote Agent (`CEServer.exe`) |
| 4448 | Mobile Access Runtime (`MobileAccessTask.exe`) |
| 5432 | PostgreSQL |
| 44818 | Allen-Bradley CIP (default), for the ABCIP driver<br>**Note:** You can change this port number in the ABCIP driver settings. |
| 47808 | BACNet UDP (default), for the BACNE driver<br>**Note:** You can change this port number in the BACNE driver settings. |
| 48010 | OPC UA Server (default)<br>**Note:** You can change this port number in your project settings. For more information, see Configure the communication settings for OPC UA Server on page 624. |

| Port | Feature or Protocol |
|---|---|
| 51234 | SCADA project runtime server (a.k.a. Data Server or TCP/IP Server), encrypted via TLS/SSL<br><br>**Note:** You can change this port number in your project settings. For more information, see Communication tab on page 122. |

### Notes

Depending on which features you use in your project, you might need to update your network firewall settings in order to open the corresponding ports and allow the traffic to pass through. For example, if you develop your project to send email alerts, you need to open port 25 for SMTP.

Many of the port numbers listed above are only the defaults for those ports. You can change them, but if you do so, you need to do so on both the server side and the client side, and that is in addition to updating the firewall settings. Keep in mind that the SCADA project runtime acts as the server in some cases and as the client in others, depending on the feature or protocol.

The communication drivers (e.g., MOTCP, ABCIP, BACNE) included in the list above are only some of the most commonly used drivers. For more information about the port used by a specific driver, see the documentation for that driver.

## *View or disconnect client sessions*

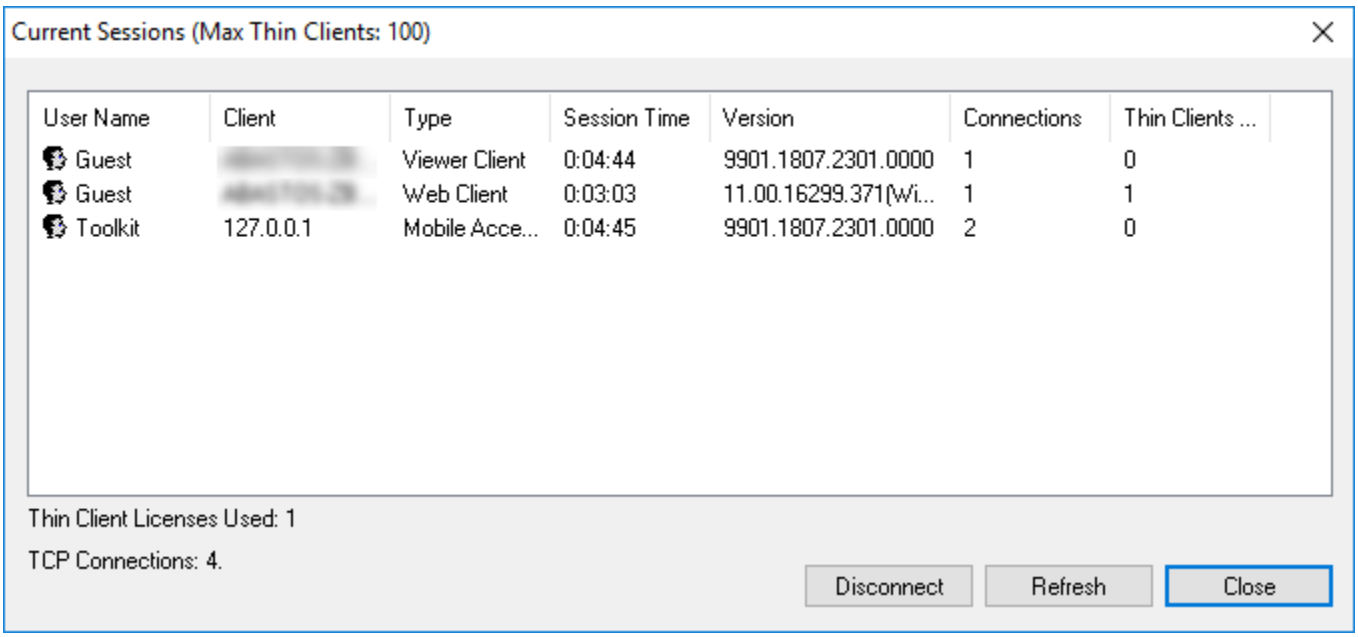
Use the *Current Sessions* dialog box to view or disconnect clients that are currently connected to your project runtime server.

Before you begin, your project must be running on Windows using the full EcoStruxure Machine SCADA Expert 2020 software with an appropriate runtime license. The Current Sessions feature is not available in our other runtime editions. Also, you must have access to the computer that hosts the project runtime, either directly or through screen sharing. You cannot access the Current Sessions feature through Remote Management.

When a client connects to your project runtime server, a client session is initiated. Each session counts against the maximum number of clients allowed by your runtime license. A session ends only when the user either logs off from the project or closes the client program, so if the current number of client sessions approaches the maximum number of clients allowed, you might need to disconnect old or idle sessions in order to ensure that your project runtime server remains accessible.

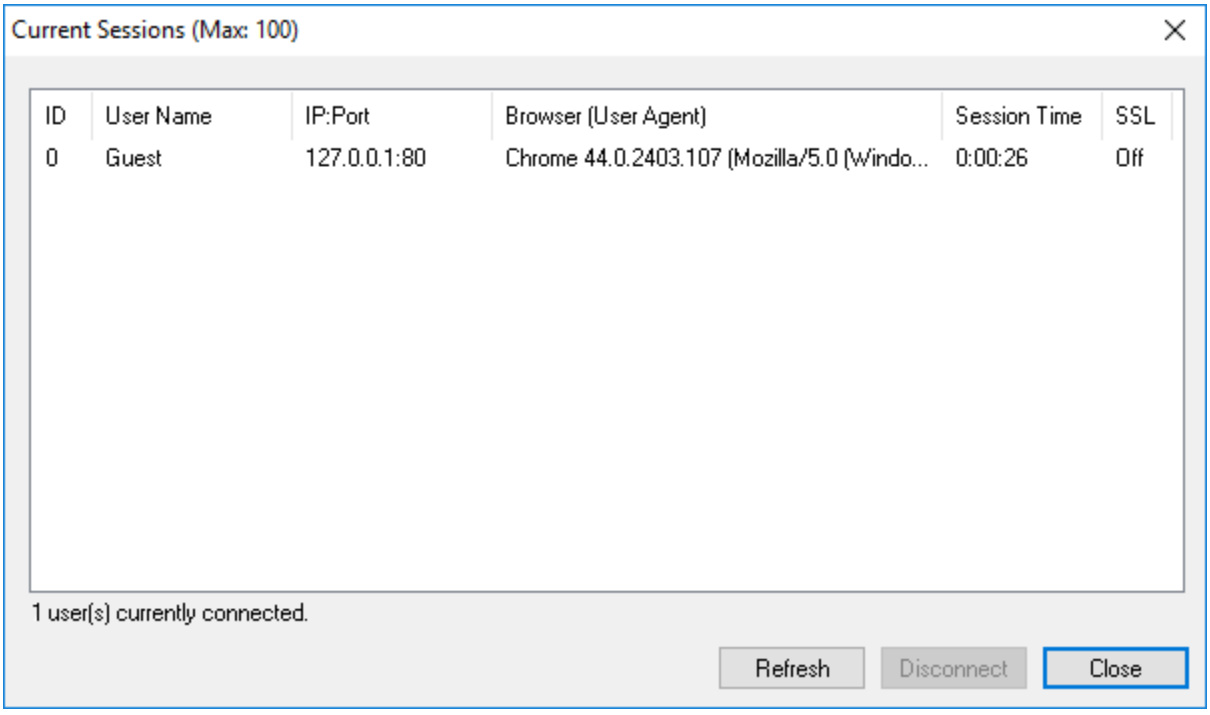
**Note:** For Mobile Access only: due to technical differences between web browsers, the exact moment when the thin client is considered "closed" — and therefore the session ends — varies somewhat. In Chrome for Android, the session ends when the user goes to a new website but not when the user closes the browser tab. In Safari for iOS, it is the opposite: the session ends when the user closes the browser tab but not when the user goes to a new website.

Thin Client sessions are handled by the TCP/IP Server runtime task in your project. When the project is running and the task is started, a **TCP/IP Server** icon is displayed in the notification area of the Windows taskbar. You can use this icon to open the *Current Sessions* dialog box for the Thin Client sessions.




***Example of the Current Sessions dialog box for TCP/IP Server***

Mobile Access sessions are handled by the Mobile Access runtime task, and similar to the above, when the project is running and the task is started, a **Mobile Access Task** icon is displayed in the notification area of the Windows taskbar. You can use this icon to open the *Current Sessions* dialog box for the Mobile Access sessions.




***Example of the Current Sessions dialog box for Mobile Access***

For more information about the TCP/IP Server and Mobile Access runtime tasks, see Runtime Tasks on page 136.

Also, please note how the title bar shows the maximum number of clients allowed. That information is gotten from your runtime license settings. For more information, see License Settings on page 45.

Finally, the **User Name** column will show individual user names only if you enabled the security system in your project. If you did not, all users will be logged on and shown as "Guest". For more information, see Project Security on page 639.

> **Note:** The Mobile Access runtime task has a memory limit of 1.5 GB. If this limit reached during project run time — typically due to trying to run an extremely large project, but also sometimes due to managing a large number of client sessions — additional clients will not be allowed, regardless of the runtime license settings. When this happens, an alert message will be displayed to users who try to log on.

To view or disconnect thin client sessions:

1. In the notification area of the Windows taskbar, right-click either the **TCP/IP Server** icon or the **Mobile Access Task** icon, and then on the shortcut menu, click **Current Sessions**.

   You might need to expand the notification area to show hidden icons.

   The *Current Sessions* dialog box is displayed.

2. To refresh the list of sessions, click **Refresh**.

   In most cases, the list will automatically refresh itself as thin clients connect and disconnect, but you can also manually refresh it make sure you have the latest information.

3. To disconnect a specific session, select that session in the list, and then click **Disconnect**.
   The selected session is disconnected and the session's user is logged off. A new session is automatically initiated, as if the user restarted or reloaded the thin client, but it will expire after a specfied period if no one logs on. For more information about session expiration, see Configure the global settings for all areas on page 790.

4. When you are done, either close the window or click **Close**.

# Mobile Access

You can use Mobile Access (sometimes also called Studio Mobile Access or SMA) to deploy an HTML5-enhanced web interface that presents alarms, trends, process values, and even project screens in a unified, easy-to-use "dashboard".

This web interface is designed for smartphones and tablets, such as Android and iOS devices, but it can be accessed from almost any computer using a modern web browser.

It is important to remember that although Mobile Access is part of the same project runtime and may be hosted on the same server that hosts the screens published for Thin Clients, it is a distinct interface based on platform-agnostic technology. Our traditional Thin Client software is based on ActiveX technology, and as such, it can run only on Windows computers. In contrast, the Mobile Access web interface is based on HTML5, so it can run in most web browsers on most computers and devices.

To use Mobile Access, you must have a web server running on the same computer that hosts your project runtime server, and you need to install and configure some additional software that creates the actual Mobile Access web interface and allows the web server to communicate with the project runtime server. This software is available both for Microsoft IIS and for other, CGI-enabled web servers (e.g., Apache).

Also, your software license must include enough Thin Clients to accommodate all of the users that you expect to access the Mobile Access web interface at the same time. Please contact your vendor to review your software license. For more information, see License Settings on page 45.

The rest of this section describes how to set up your web server for Mobile Access, how to configure the Mobile Access web interface during project development, and how to log on to and navigate the web interface during project run time. You should already be familiar with how to locate and open worksheets in the Project Explorer, how to edit a worksheet, and how to save and close a worksheet.

## *Supported features in Mobile Access*

This is a list of the features that are currently supported in project screens when they are viewed in the Mobile Access web interface.

### Screen objects and animations

The following table shows exactly which screen attributes, objects, and animations are supported in project screens, as well as the specific properties that are supported on each one:

| Group | Type | Properties |
|---|---|---|
| Screen Attributes | Background Picture | You can select any image format, but support for certain formats varies from browser to browser. If you use an unsupported image in your project screen, the browser will not be able to render that image when you view the screen in the Mobile Access web interface. For the best performance across all browsers, try to use "web-compatible" image formats such as GIF, JPG, and PNG wherever possible. |
| | Size | Width, Height |
| | Location | Top, Left |
| | Titlebar | Text only — you cannot configure a tag in curly brackets (e.g., `{MyTitle}`). |
| | System Menu | Close button only<br>**Note:** If you select the **System Menu** option, the user will be able to drag the screen within the browser window. |
| | Style | Dialog, Popup, Replace (Partial), Replace (Complete), Overlapped |
| | Border | None, Thin |
| | Screen Logic | On Open, While Open, On Close |
| | Multi Touch Settings | Inner Zoom only — it is automatically enabled and cannot be changed. |
| Shapes | Line | Solid Line, Dashed Line, No Line, Color, Weight |

| Group | Type | Properties |
|---|---|---|
| | Open Polygon | Solid Line, Dashed Line, No Line, Color, Weight |
| | Closed Polygon | All Border Types, Border Color, Border Weight, Fill Color, Fill Effects (Horizontal, Vertical, Diagonal Up, Diagonal Down) |
| | Rectangle | All Border Types, Border Weight, Border Color, Fill Color, Fill Effects (Horizontal, Vertical, Diagonal Up, Diagonal Down), Caption, Fonts, Multiline, Wrap Text |
| | Rounded Rectangle | All Border Types, Border Weight, Border Color, Fill Color, Fill Effects (Horizontal, Vertical, Diagonal Up, Diagonal Down) |
| | Ellipse | Type - Ellipse only, Border Weight, Border Color, Fill Color, Fill Effects (Horizontal, Vertical, Diagonal Up, Diagonal Down) |
| Active Objects | Text | Caption, Align, Fonts, Background, Hint (for Data Input only)<br>Horizontal scaling of text — which you can normally achieve by horizontally resizing the Text object — is not supported. Text will always appear at its full width for the specified font size. If you need "narrow" text, use an appropriate font like Arial Narrow. |
| | Text Box | Hint (for Data Input only), Format (all choices), Input Enabled, Fonts, Mask/Count, Minimum Value, Maximum Value, Disable, Multi-line, Password, Scroll Bar, Word Wrap, RTL |
| | Button | Styles (3D Sharp, 3D Soft, OS Like), Fonts, Align, Multiline, Wraptext<br>Images, including the Size and Position properties, are supported. The Transparent Color property is not supported, however. Images that have transparency encoded in the image file itself (i.e., in the so-called "alpha channel") will be displayed as intended in the browser. This includes .png files and some .gif files. Other image formats — most notably .bmp files — cannot be displayed with transparency in the browser. We recommend using .png files whenever possible. |
| | Pushbutton | All Types, States, and Styles<br>Animations applied to the Pushbutton object are not supported, regardless of whether they are listed below as being supported in general. |
| | Check Box | Tag, True Value, Caption, Fonts, 2 states |
| | Radio Button | Tag, True Value, Caption, Fonts, 2 states |
| | Combo Box | Label, Position, Disable, Security, Sort, Data Sources (Static Labels or Database; see note), Advanced (Color, Decimal Points), Fonts (except Strikeout and Underline)<br>If you select Database as the data source for a Combo Box object, the object must use the default database (primary or secondary) that is configured in the project settings. However, you can select a different table and/or field for each instance of the object. |
| | List Box | All properties are supported except for the Font properties Strikeout, Underline, and Script<br>Animations applied to the List Box object are not supported, regardless of whether they are listed below as being supported in general. |
| | Smart Message | Message Display only, Integer Value only, Static Data Source only, Align, Read Tag/Expression, all Message properties, and all Font properties except for Strikeout, Underline, and Script<br>Animations applied to the Smart Message object are not supported, regardless of whether they are listed below as being supported in general. |
| Data Objects | Alarm/Event | see "Alarm/Event Control object" below |
| | Trend | see "Trend Control object" below |
| | Grid | see "Grid object" below |
| Libraries | Linked Symbol | see "Custom properties" below |
| | Linked Picture | Link File, Transparent (Color Code, Tracker)<br>**Note:** BMP, JPG, and PNG files only. |

| Group | Type | Properties |
|---|---|---|
| | Custom Widget | fully supported |
| Animations | Command | Events: On Down, On Up, On Right Down, On Right Up<br>Types: VBScript, Open Screen, Close Screen, Set Tag, Reset Tag, Toggle Tag |
| | Hyperlink | Type, URL |
| | Bargraph | Minimum Value, Maximum Value, Foreground Color, Direction (Vertical, Horizontal), Orientation (Up, Down) |
| | Text Data Link | Format (all choices), Input Enabled, Minimum Value, Maximum Value, Disable, Password, RTL |
| | Color | Type (By Limit, By Color), Change Limit, Color, Blink (Slow, Fast)<br>This animation is not supported when it is applied to a group of shapes (e.g., Line, Rectangle, Ellipse). Instead, apply the animation separately to each shape before you group them. |
| | Visibility/Position | Visibility, Horizontal (Tag/Expression, Value Range, Position, Reference), Vertical (Tag/Expression, Value Range, Position, Reference), Slider<br>**Note:** The tags configured for Horizontal Tag/Expression and Vertical Tag/Expression are not updated until the user releases the object. |
| | Resize | Height (Tag/Expression, Value Range, Size Range, Reference), Width (Tag/Expression, Value Range, Size Range, Reference) |
| | Rotation | Range Minimum, Range Maximum, Degrees Start, Degrees End, Reference, Offset (x, y), Counter Clockwise |

The Windows-based Virtual Keyboard (VK) is not used for data input on any screen objects or animations. Instead, if input is required from the user, a customized Data Input dialog box is displayed and the client station's own keyboard — on-screen for tablets and smartphones, physical for other computers — is used. For more information, see Data Input on page 345.

The **Hint**, **E-Sign**, **Auto Format**, **Security**, and **Key** properties — which are common to most screen objects and animations — are not supported unless otherwise noted. The **Enable Translation** option is supported with some limitations; see "Translation" below.

***Examples of supported screen objects and animations***

### Alarm/Event Control object

The Alarm/Event Control object is supported in project screens on Mobile Access, and it appears and behaves essentially the same as it does in the other, Windows-based thin clients. Nevertheless, you should thoroughly test your project on both types of clients in order to confirm that your Alarm/Event Control object(s) behave as expected during project run time.

The following list describes the specific object properties and features that are supported on Mobile Access:

- Type: All types and history formats are supported.
- Filters: All options are supported, except **Filter Expression** for **Alarm Online**.
- Columns:
  - All available columns are supported, except **Delete Message**.
  - Properties: **Label**, **Width**, **Align**.
  - As noted above, the **Key** feature is not supported in any screen object on Mobile Access.
- Advanced:
  - Date & Time Format: All options are supported.
  - Delete Message: Not supported.
  - Acknowledgement: All properties are supported, except the **Security** feature which is not supported in any screen object on Mobile Access.

- Run-time Returned Values: All properties are supported, except **Summary Changes**. Please note that you must specify arrays for **First Row Text** and **Selected Row Text**.
- Run-time Dialog Triggers: Not supported.
- Save/Print: Not supported.
- Navigation Triggers: Not supported.
- As noted above, the **Auto Format** feature is not supported in any screen object on Mobile Access.

- Fonts: All options are supported, except **Strikeout**, **Underline**, and **Script**.
- Format: All options are supported, except background color fill effects which are not supported in any screen object on Mobile Access.

### Trend Control object

The Trend Control object is supported in project screens on Mobile Access, but it appears and behaves somewhat differently than it does in the other, Windows-based thin clients, moreso than can be described in this documentation. You should thoroughly test your project on both types of clients in order to become familiar with the differences. In particular, note the differences in behavior of the cursor and toolbar.

You can use multi-touch gestures to manipulate the trend control, assuming your Mobile Access device has touchscreen input. For example, you can "pinch" and "stretch" to zoom (i.e., change the X-axis scale) and you can "slide" to pan (i.e., change the X-axis period).

The following list describes the specific object properties and features that are supported on Mobile Access:

- Points:
  - Supported properties: **Label**, **Color**, **Tag/Field**, **Hide**.
  - Data Source: **Tag** only. All tags must be specified in Trend worksheets, but both online (i.e., current) and historical values can be displayed.
- Data Sources: No other data sources are supported. All tags must be specified in Trend worksheets.
- Axes:
  - X Axis: For the Date/Time data type, **Period** must be **Auto**. For the Numeric data type, only **Min** and **Max** are supported at this time.
  - Y Axis: Only **Min** and **Max** are supported at this time.
- Legend:
  - The entire legend can be shown or hidden, by selecting or clearing the **Show legend** option. If no fields are selected to be visible, however, then the legend will be automatically hidden.
  - Only the **Label** property is supported at this time, and it displays the label that is configured in the Points section above. Additional properties will be supported in future releases of this software.
- Toolbar:
  - The entire toolbar can be shown or hidden, by selecting or clearing the **Show toolbar** option.
  - Supported commands: **Run**, **Stop**, **Zoom In**, **Zoom Out**, **Cancel Zoom**, **Cursor**, **Auto Scale**.
  - If the toolbar is hidden, you can use activation tags to trigger the supported commands.
  - You can specify tooltips for the supported commands.
- Advanced:
  - Update trigger is supported.
  - "Move to current time on run" is the default behavior of the Trend Control object on Mobile Access. If the option is cleared, it will be ignored.
  - Decimation is the default behavior of the Trend Control object on Mobile Access. It cannot be disabled, and any changes in the configuration will be ignored.

### Grid object

The Grid object is supported in project screens on Mobile Access, and it appears and behaves essentially the same as it does in the other, Windows-based thin clients. Nevertheless, you should thoroughly test your project on both types of clients in order to confirm that your Grid object(s) behave as expected during project run time.

The most significant limitation is that only the Database source type is supported; the Text File and Class Tag source types are not supported at this time. Also, only the default database connection (either Primary or Secondary) is supported; if you configure a Grid object to use a database connection other than than the default, it will not be supported on Mobile Access. You can use any table in the default database, however.

The following list describes the other, specific object properties and features that are supported on Mobile Access:

- Columns:
  - Supported properties: **Label**, **Field**, **Type** (all except **Picture**), **Width**, **Input**.
  - You can configure a project tag for **Label** (e.g., **`{MyLabel}`**) in order to change a column label during project run time, but the value of the tag is gotten only when the project screen is opened. Using **Reload** to reload the contents of the Grid object does not also reload the column label.
- Advanced:
  - Supported properties: **Selected Values**, **Number of Rows**, **Row Number**, **Condition** (see below), **Reload**, **Save Trigger**, **Insert Trigger**, **Inserted Values**, **Save on data change**.
  - When using the **Condition** property to filter the grid rows, Date/Time data types are not supported. In other words, if a grid column is configured so that its Type is one of the Date, Time, or Date/Time options, then you cannot enter a condition expression that filters the grid rows according to the values in that column.
  - For the **Selected Values** and **Inserted Values** properties, you can include an array index in order to specify the starting position (offset), but that index must be a literal value and not a project tag. For example, `MyArray[3]` is valid but `MyArray[GridRowStartPos]` is not.
- Fonts: All configurations are supported.
- Colors: All configurations are supported.

### Translation

The Translation Table feature, the **Enable Translation** option on most screen objects, and the associated Translation functions are all supported on Mobile Access, but with the following limitations.

First, customized fonts and date formats for target languages are not supported. Second, only project texts — that is, the text you add to screens and worksheets as you develop your project — can be translated; the other options in the **Origin** menu in the Translation Table worksheet are not supported, which means user interface elements like menus and dialog boxes cannot be translated. And third, the **Translate before parsing strings in curly brackets** and **Enable alarm/event delimiters** options in the advanced settings are not supported. For more information, see Project Localization on page 681.

### Custom properties

Custom properties (formerly known as "mnemonics") are supported in project screens on Mobile Access, but only for object properties that are themselves supported, as described in the table above. For more information, see Use custom properties to set property values when screens are opened on page 330.

### Multiple screens and screen groups

You can open multiple screens and screen groups in the Mobile Access web interface, just as you normally would in other thin clients.

It is not possible at this time, however, to make screen groups available through the Screens control. (The Screens control is a part of the Mobile Access web interface that allows the user to view selected project screens.) Instead, to open a screen group, you must do one of the following:

- In another screen, configure a screen object or script to call the `Open` function to open the screen group, and then have the user view that other screen first; or

- Link directly to the screen group file. For more information, see Link directly to a project screen or screen group on page 801.

### Multi-touch gestures

You can use certain multi-touch gestures in project screens — specifically, you can use the "pinch" and "stretch" gestures to zoom a screen, and you can use the "slide" gesture to pan a screen that has been zoomed. No other gestures are supported at this time, however. For more information, see Using multi-touch gestures in project screens on page 358.

### Built-in functions

Many but not all of Studio's built-in functions are supported in project screens. To see if a specific function is supported, please refer to the documentation for that function. The function will be marked either "Supported", "Not Supported", or "Executed on Server", and there might be additional notes describing how the function is executed in Mobile Access.

Functions that are marked "Executed on Server" are executed via remote procedure call (RPC) on the project runtime server. While these functions are supported, you should avoid using a large number of them in any project screens that you include in Mobile Access. The extra communication required between server and client can affect run-time performance.

Also, please note that because Database/ERP functions are executed on the server, they affect server tags (i.e., tags with Server scope) rather than local tags (i.e., tags with Local scope), and that might result in unexpected behavior when multiple clients try to execute the functions at the same time. As such, you should avoid specifying optional parameters that take tag names — for example, if you call the function `DBCursorOpen`, do not specify the parameters *optStrTags* and *optStrTagError*.

For more information, see Appendix: Built-in Language on page 896. More functions will be supported in future releases of this software.

### VBScript

Most VBScript interfaces — including the Graphics Script, the Screen Script, and Command animations on screen objects — are supported in project screens in Mobile Access. Most VBScript and built-in functions, commands, and syntax can be used the same as they are outside of Mobile Access, with the following exceptions and limitations:

**VBScript functions that return as Date**

VBScript functions that return values as the Date type are not yet supported in Mobile Access. These functions include the following:

- `Date`
- `Time`
- `Now`
- `DateAdd`
- `DateSerial`
- `DateValue`
- `TimeSerial`
- `TimeValue`

**Functions that open project screens and display dialog boxes**

Functions that open project screens and display dialog boxes can be called from the Screen Script and Command animations, but they cannot be called from the Graphics Script. These functions include the built-in functions `Open`, `LogOn`, and `ShowMessageBox`, as well as the VBScript function `MsgBox`.

Dialog boxes are displayed as modal windows in the Mobile Access web interface. In other words, when a function displays a dialog box, project screens in the background will not be updated until the user closes that dialog box.

**Calling variables and procedures declared in the Graphics Script**

Normally, variables and procedures that have been defined in the Graphics Script can be called from the other VBScript interfaces using the syntax **Graphics.*variable_name*** or **Graphics.*procedure_name***, respectively. This feature is not supported on Mobile Access, however, because in the Mobile Access web interface, the Graphics Script and each project screen runs in its own thread separate from the others.

For variables that you want to be global, use project tags instead. For procedures that you want to be global, define them in Global Procedures and then run them using the function RunGlobalProcedureOnServer on page 1064.

### System tags GroupCNFLoLevel and GroupCNFHiLevel

The pre-defined system tags GroupCNFLoLevel and GroupCNFHiLevel are not supported in Mobile Access. In fact, these tags have been deprecated; if you want to check the security levels to which the user has access, use the function `CheckSecurityLevel` instead.

### Setting a project tag to an empty value

When a project tag is given an empty value — for example, when it is set to equal the VBScript keyword **Empty** — an error message is sent to the log and the actual value of the tag is not changed. In other words, the project tag retains its existing value.

### Tag fields

The following tag fields are supported on each type of project tag in Mobile Access:

| Tag Field | Supported on Type… | | | |
|---|---|---|---|---|
| | Boolean | Integer | Real | String |
| Name | # | # | # | # |
| MemberName | | | | |
| Size | # | # | # | # |
| Index | # | # | # | # |
| Description | # | # | # | # |
| Value | # | # | # | # |
| TimeStamp | # | # | # | # |
| Quality | # | # | # | # |
| Blocked | | | | |
| Min | | # | # | |
| Max | | # | # | |
| Unit | | # | # | |
| UnitDiv | | | | |
| UnitAdd | | | | |
| DisplayValue | | | | |
| DisplayMin | | | | |
| DisplayMax | | | | |
| DisplayUnit | | | | |
| Hi | # | # | # | |
| Lo | # | # | # | |
| HiHi | | # | # | |
| LoLo | | # | # | |
| Rate | | # | # | |

| Tag Field | Supported on Type... | | | |
|---|---|---|---|---|
| | Boolean | Integer | Real | String |
| DevP | | # | # | |
| DevM | | # | # | |
| HiLimit | # | # | # | |
| LoLimit | # | # | # | |
| HiHiLimit | | # | # | |
| LoLoLimit | | # | # | |
| RateLimit | | # | # | |
| DevPLimit | | # | # | |
| DevMLimit | | # | # | |
| DevSetPoint | | # | # | |
| AlrStatus | # | # | # | |
| AlrDisable | # | # | # | |
| Ack | # | # | # | |
| UnAck | # | # | # | |
| AlrOffValue | # | | | |
| AlrOnValue | # | | | |
| AlrAckValue | # | | | |
| B0 ... B31 | | # | | |

**Note:**

If you reference a simple class tag without specifying a member (e.g., **`MyClass->fieldname`**), only the Size and Description fields are supported. If you reference a complex, array-based class tag without specifying a member (e.g., **`MyClass[0]->fieldname`**), none of the fields are supported. To access all of the supported fields listed above, you must specify the full tag and member names, as well as the array index if applicable. For example, **`MyClass[0].Member->fieldname`**. For more information, see About classes on page 149.

Also, the script/expression compiler used in Mobile Access is stricter than the one used elsewhere in Studio. It will not accept references to unsupported tag fields. For example, if you try to reference **`MyString->B0`** anywhere else in Studio, the compiler will accept the reference and then simply return **`0`** or some other invalid value. In Mobile Access , however, a run-time error will be generated because B0 is not supported on String tags. You can check for such errors in the log.

For more information about tag fields, see Reference a tag property instead of a project tag on page 169.

### Tag changes in the Event Logger

When a project tag is changed using VBScript in Mobile Access, that change will be logged in the Event Logger with the client's IP address instead of its host name.

### Date formats and time zones

While the server and clients may have their respective system times, Mobile Access always uses the server's date format and time zone settings when it opens project screens. In the current release, it is not possible for the server and clients to have different settings, so you should not try to view project screens on clients with different settings as that might result in unexpected behavior during run time. (It is okay to use the Alarm, Process Values, and Trend controls in

the web interface, because they do not include any VBScript that might be affected by this limitation.)

As a workaround, you can change the time zone setting on your client to match the server, but if that is not practical and you must view your project screens while in a different time zone, you should use our traditional Thin Client software instead of Mobile Access.

**Boolean tags**

Mobile Access does not support the legacy method for handling Boolean tags (i.e., project tags of Boolean type) in VBScript. Boolean tags are always handled as if they have a numerical value of -1 for TRUE, to ensure compatibility with Boolean variables in VBScript. Editing your project file to change the property **VBBoolean** will not override this. For more information, see How Boolean tags are handled in VBScript on page 1238.

**Statement continuation with comments**

In VBScript, you can use an underscore character to indicate that a statement is continued to the next line. Programmers often do this to make a long statement easier to read. For example:

```
MyArray = Array("FIRST_NAME", _
                "LAST_NAME", _
                "ADDRESS")
```

Without the underscore character, the end of the line would also be the end of the statement.

The VBScript compiler in Mobile Access supports continuing a statement like this, except for the following limitation: in `MsgBox` statements and function calls, you cannot insert a comment after an underscore character. For example:

```
MyVar = MsgBox("Hello World!", _ 'This is a comment
               65, _ 'This is another comment
               "MsgBox Example")
```

This code would not be accepted by the VBScript compiler in Mobile Access, and the resulting compiler error could prevent a project screen from opening at all.

For more information about using VBScript in your project, see Overview of VBScript on page 1204.

### Important features not supported

Mobile Access supports only the features listed above and with the limitations mentioned. Among the features not supported, the following ones are most commonly used:

**Selecting from a list of users to log on**

The Mobile Access Logon screen does not support selecting from a list of users. The user must know and type their user name. For more information, see Log on to the Mobile Access web interface on page 792.

**Tag updates while built-in dialog boxes are displayed**

Certain built-in dialog boxes (e.g., MsgBox, LogOn) act like modal windows when they are displayed on the Mobile Access client, even though they are displayed within the web browser. As such, open project screens will stop receiving tag updates from the project runtime server — and consequently, animations in the project screens might appear to freeze — while one of those dialog boxes is displayed. The tag updates will resume as soon as the user closes the dialog box. Please note that this limitation does not apply to Popup-style project screens, which can appear similar to those dialog boxes.

**Tabbing through screen objects**

Using a keyboard to tab through and activate screen objects is not supported, due to how project screens and screen groups are composed and displayed in Mobile Access. In most cases, it is better to click or tap on a screen object in order to activate it.

**Embedded bitmaps**

Embedded bitmaps are not supported in project screens on Mobile Access. If you paste bitmaps into your screens, make sure that they are saved in separate files. For more information, see Paste a bitmap image into a screen on page 248.

**Image formats**

Support for certain image formats varies from browser to browser. If you use an unsupported image in your project screen, the browser will not be able to render that image when you view the screen in the Mobile Access web interface. For the best performance across all browsers, try to use "web-compatible" image formats such as GIF, JPG, and PNG wherever possible.

**Background color fill effects**

Background color fill effects are not supported in project screens in the Mobile Access web interface. You can select any solid color for the background, but if you use fill effects to create a color gradient, only the gradient's "start" color will be displayed. If the screen's background must be a color gradient, create it as a background image instead. For more information, see Modifying a screen's background color or image on page 235.

**Additional options for Driver and OPC communication**

Some additional options for Driver and OPC communication are not supported in Mobile Access. First, in the project settings, the **Send last state** option is not supported. Mobile Access automatically uses the **Send every state** option, with a fixed buffer size of 5. For more information, see Communication tab on page 122.

Second, in all Driver and OPC client worksheets, the **Scan** field in the worksheet body is not supported. More specifically, while the field can normally be set to either **Screen** (scan the tag only while a screen that uses the tag is open) or **Always** (always scan the tag while the worksheet is enabled), the **Screen** option cannot be supported due to how project screens are presented by the Mobile Access web interface. All tags in all Driver and OPC client worksheets should have the **Always** option selected. As an alternative, you can configure a worksheet's **Disable** setting to disable the entire worksheet unless a screen is open.

**Compressing files for faster downloads**

In projects that are accessed by our traditional Thin Client software, you can choose to compress the screen files to make them faster to download over slow connections. (The **Enable File Compression** option is located on the **Web** tab of the project settings.) This feature is not supported in Mobile Access. If you try to view a project screen that has been compressed like this, the screen might behave unexpectedly and you might see messages in the activity log that say tags or objects do not exist.

If your project requires a feature that is currently not supported in Mobile Access, consider using our traditional Thin Client software instead. For more information, see Thin Clients on page 729.

More features will be supported in future releases of this software.

## *Tips for Mobile Access development and run time*

These are general tips for developing projects for Mobile Access, as well as for using the Mobile Access web interface during run time.

### Do not use unsupported features in your project

Mobile Access currently supports many but not all features of EcoStruxure Machine SCADA Expert 2020. If you use an unsupported feature in a project screen, you might see unexpected behavior when you view that screen in the Mobile Access web interface. Such behavior can range from incorrect tag changes and function calls to objects, animations, or scripts that do not work at all.

Make sure that all of the screen objects, animations, background tasks, VBScript, and built-in functions that you use are included in the list of supported features. For more information, see Supported features in Mobile Access on page 756.

If you do use unsupported features, they will be reported in the Output window in the development environment when you either verify your project or publish your screens as HTML.

### Use an HTML5-compatible browser to view your project

The Mobile Access web interface uses HTML5 (including CSS3 and AJAX) to create animated graphics and perform real-time data exchange. That means you must use an HTML5-compatible browser to view your project in Mobile Access. We recommend Google Chrome — it is available for most platforms and operating systems, and we have found that it provides the best overall performance and compatibility.

For more information, see Log on to the Mobile Access web interface on page 792.

### Make sure your computer or device has enough resources

Viewing large or complex project screens in the Mobile Access web interface can be resource-intensive, and your computer or device might not have enough resources (e.g., processor, memory, bandwidth) to do the job regardless of how new it is, which operating system it runs, or which browser you use. If either the web interface in general or a specific project screen seems to perform unsatisfactorily, try viewing it on another computer or device. Also, particularly on mobile devices like smartphones and tablets, check to see if your device's battery is low or if you have other apps open. Any or all of these factors can affect the performance of the web interface and give you a false impression of Mobile Access itself.

If, after checking these things, you still see unsatisfactory performance, you may need to "lighten" your project screens — that is, modify your screens to decrease the amount of resources that they require. Here are a few ways to do that:

- Try not to use large, high-resolution pictures in a screen, especially if you resize them in the screen editor after you place them. Replace them wherever possible with pictures that are properly scaled and resampled. Also, try decreasing the image resolution (e.g., from 300 DPI to 72 DPI) if full resolution is not necessary.
- Do not to paste a picture or use a group of objects more than once in a screen, because each instance requires its own resources. Replace these pictures and groups with Linked Pictures and Linked Symbols, respectively.
- Make sure the project screen itself is properly sized for the computers or devices on which you plan to view it. It is a waste of resources to create, for example, a 2560-by-1440 screen for a 750-by-1334 smartphone (i.e., the iPhone 6). If the **Auto Screen Scaling** option is selected in the project settings (see Viewer tab on page 117), the screen is automatically downscaled to fit the browser in which it is viewed, so nothing will be inadvertently cropped or hidden and the user can zoom in to see the smaller details, if necessary. (Please note that zooming works somewhat differently in mobile browsers versus desktop browsers.) An oversized screen, however, always takes more resources and is more difficult to use.

  You can change the screen size in the Screen Attributes.

In the end, however, please keep in mind that the performance of the Mobile Access web interface is not an indicator of the performance of the EcoStruxure Machine SCADA Expert 2020 project runtime itself. Regardless of what you see in your browser, your project should be running well everywhere else.

### Other tips for developing for Mobile Access

Here are some other tips for project and screen development:

**Do not enable file compression**

In projects that are accessed by our traditional Thin Client software, you can choose to compress the screen files to make them faster to download over slow connections. (The **Enable File Compression** option is located on the **Web** tab of the project settings.) This feature is not supported in Mobile Access. If you try to view a project screen that has been compressed like this, the screen might behave unexpectedly and you might see messages in the activity log that say tags or objects do not exist.

**Avoid tag synchronization when opening screens**

If you use a large number of project tags in the VBScript sub-routines `Screen_OnOpen` or `Graphics_OnOpen`, your screens might take a long time to open or update. This is because the tag values — even for project tags with Local scope — must be synchronized between server and

client when the scripts are executed, and that can take a long time if you have a slow client/server connection.

The solution is to use VBScript variables instead of project tags wherever possible. Variables exist only within the scripts where they are declared and used, so no synchronization is required. Otherwise, you can check the activity log to see which tags are being synchronized and when.

**Do not use "Executed on Server" functions in FOR loops**

Similar to the issue of tag synchronization that is described above, if you use a large number of functions marked "Executed on Server", your project screens might take a long time to update. This is because the function calls must be sent from the client to the server, and then the returned values must be sent from the server to the client.

This is especially true of functions called from within a FOR loop. The loop itself can be executed relatively quickly on the client, but the function calls might "stack up" as the client waits for the server to execute them.

If you experience any of the issues described above, you can use the activity log to troubleshoot your project screens. For more information, see Troubleshooting project screens in Mobile Access on page 803.

## *Mobile Access web server add-on*

The Mobile Access add-on is an extension to your web server that allows it to work with your project runtime server, and it is required to make the Mobile Access web interface accesible to remote users.

More specifically, this add-on establishes a connection between your web server and the Mobile Access task in your project runtime. It is the task that actually manages the Mobile Access features of your project, and it communicates with clients through the web server.

This add-on also provides the webpages, scripts, and images that make up the Mobile Access web interface. The web server serves these files to clients as needed.

The add-on must be installed with the web server on the same computer that hosts the project runtime server. As such, if you plan to have remote users access your project over the Internet, the computer itself must be connected to the Internet.

Add-ons are available for several different web server platforms:

| Operating System | Microsoft IIS | CGI / Apache |
|---|---|---|
| Windows / Windows Server | Supported (More info…) | Supported (More info…) |

### MOBILE ACCESS WEB SERVER ADD-ON FOR IIS

The Mobile Access web server add-on for IIS connects your project runtime server with Microsoft's Internet Information Services (IIS), as long as they are both running on the same computer.

#### Turn on IIS for thin client access

Turn on Microsoft's Internet Information Services (IIS) and configure it with the correct settings to make your project accessible to thin clients over the network.

Please note that these instructions apply only to the following versions of IIS:

| Version | ...on Operating System |
|---|---|
| IIS 8.5 | • Windows 8.1<br>• Windows Server 2012 R2 |
| IIS 10 | • Windows 10<br>• Windows Server 2016<br>• Windows Server 2019 |

> **Note:** For Windows Embedded Standard, IIS must be included in the device's operating system image by the device manufacturer. It cannot be manually installed after the fact.

IIS supports all features of EcoStruxure Machine SCADA Expert 2020, and it is robust enough to serve almost any EcoStruxure Machine SCADA Expert project in a production environment. It is the web server software that we recommend for most users, and to achieve the best performance during project run time, we strongly recommend that you use one of the versions listed above.

You only need to turn on IIS on the computer that will be your project runtime server. This might be the same computer that you are using to develop your project, especially if you plan to test your project locally, but it does not need to be.

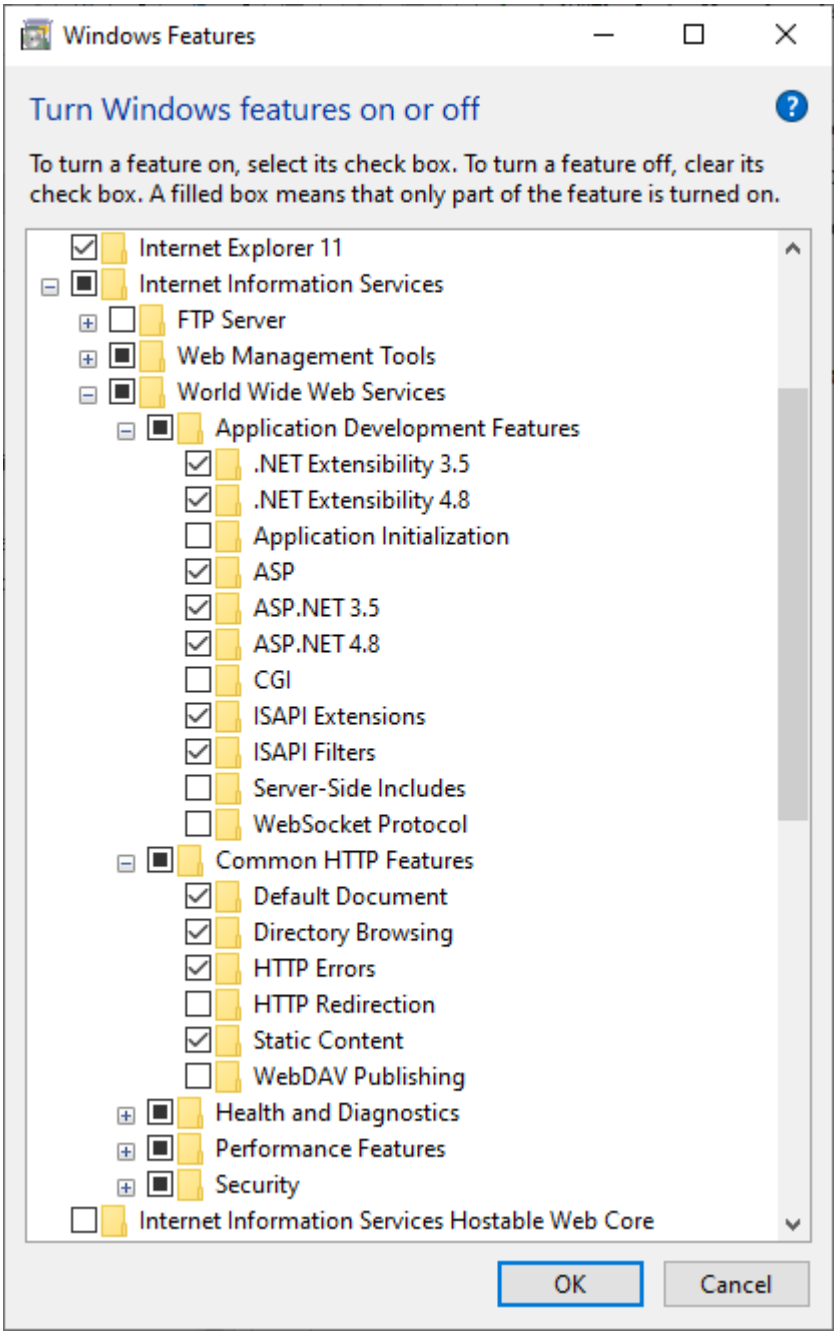
You must have Administrator privileges on the computer in order to turn on and configure IIS, and you should be familiar with administering Microsoft Windows on a network.

For the sake of system security, IIS is turned off by default when the operating system is installed. To turn on and configure IIS in Windows:

1. Do one of the following:
   - In Windows 8.1, go to **Start** and then click **Control Panel**; or
   - In Windows 10, go to **Start** and then click **Settings**.

   The *Control Panel* (Windows 8.1) or *Windows Settings* (Windows 10) window is displayed.
2. Use the search box to search for **Turn Windows features on or off**.
   The *Windows Features* dialog box is displayed.
3. In the *Windows Features* dialog box, select **Internet Information Services**.
   IIS is selected with its default features, but you need to make sure that all of the features required by EcoStruxure Machine SCADA Expert 2020 are also selected.
4. Expand **Internet Information Services** > **World Wide Web Services** > **Application Development Features**, and then make sure all of the following features are selected:
   - **Application Development Features**
     - **.NET Extensibility 3.5**
     - **.NET Extensibility 4.x**
     - **ASP**
     - **ASP.NET 3.5**
     - **ASP.NET 4.x**
     - **ISAPI Extensions**
     - **ISAPI Filters**
   - **Common HTTP Features**
     - **Static Content**

Some additional features may be selected by default, but you do not need to clear them. You only need to make sure the features listed above are selected.




***An example of the features selected in Windows 10***

5. Click **OK**.
   IIS is turned on with the selected features, but you might be prompted to restart Windows to apply the changes.
6. After you have turned on IIS, you can use Internet Information Services (IIS) Manager to configure it. To open IIS Manager:
   a) In the *Control Panel* window, click **System and Security**, and then click **Administrative Tools**.
      The *Administrative Tools* window is displayed.

b) In the *Administrative Tools* window, double-click **Internet Information Services (IIS) Manager**.

Please note that turning on IIS as a Windows feature does not mean the web server is actually running. You will need to use IIS Manager to start Default Web Site, after you have finished configuring IIS and installing any other software.

> **Tip:**
>
> A web server typically operates on, or "listens to," a computer's TCP/IP port 80. Only one running process can listen to a given port, so if another process on your computer — for example, third-party SCADA software — is already listening to port 80, it and the web server process may conflict with each other. You must either configure one of the processes to listen to a different port or use Task Manager to end the conflicting process. If you cannot identify the conflicting process, run Command Prompt and then enter the following command to get a list of all networking processes:
>
> ```
> netstat -a -o
> ```

## Enable SSL encryption in Microsoft IIS

Enable Secure Socket Layer (SSL) encryption in Microsoft Internet Information Services (IIS) in order to secure communications between the web server and your thin clients.

Please note that these instructions apply only to the following versions of IIS:

| Version | ...on Operating System |
|---|---|
| IIS 8.5 | • Windows 8.1<br>• Windows Server 2012 R2 |
| IIS 10 | • Windows 10<br>• Windows Server 2016<br>• Windows Server 2019 |

For information about enabling SSL on earlier versions of IIS, go to: support.microsoft.com/kb/299875

Also, before you begin this task, you should know whether you are going to use a signed or a self-signed certificate. Both types of certificates are explained below, but since this is not intended to be a complete discussion of Windows server administration, instructions are provided only for creating a self-signed certificate so that you can continue developing and testing your EcoStruxure Machine SCADA Expert project. For information about requesting a signed certificate, go to: technet.microsoft.com/library/cc732230

Your EcoStruxure Machine SCADA Expert project has a built-in security system that you can use to control who logs on and what access they have. It does nothing to secure the connection between the server and the client, however, so if your local network is insecure and/or you connect to your server over the Internet, then your communications can be intercepted and possibly compromised.

One way to secure the connection is to use Secure Socket Layer (SSL) encryption to encrypt the packets that are sent between the server and the client. When SSL is enabled on the server, the server offers a certificate that includes proof of the identity of the server and an encryption key. The client — in this case, your web browser — can either accept or reject the certificate, depending on whether it trusts the certificate. If the certificate is trusted, then it is automatically accepted and SSL is turned on; in many web browsers, this is indicated by a padlock icon. If the certificate is not trusted, then an alert message is displayed and the user must choose whether to accept it or reject it.

The criteria for trusting a certificate is typically whether the certificate is signed by a known certificate authority (CA) and is unexpired. However, a signed certificate must be requested and purchased from a CA, so there is also an option to create a free, self-signed certificate. A self-signed certificate is a certificate signed by the server that is offering it, and as long as it is used only on a secure local network where you know and trust all of the other computers, it is sufficient for project development. (Again, for information about requesting a signed certificate, go to: technet.microsoft.com/library/cc732230)

> **Note:** You should not use a self-signed certificate in a production environment.

To create a self-signed certificate and enable SSL encryption in IIS:

1. Do one of the following:
   - In Windows 8.1, go to **Start** and then click **Control Panel**; or
   - In Windows 10, go to **Start** and then click **Settings**.

   The *Control Panel* (Windows 8.1) or *Windows Settings* (Windows 10) window is displayed.
2. Use the search box to search for **Administrative Tools**.
   The *Administrative Tools* dialog box is displayed.
3. In the *Administrative Tools* window, double-click **Internet Information Services (IIS) Manager**.
   The *IIS Manager* window is displayed.
4. Create the self-signed certificate:
   a) In the *IIS Manager* window, in the **Connections** list on the left, select your server (typically your own computer).
   b) In **Features** view, double-click **Server Certificates**.
   c) In the **Actions** pane, click **Create Self-Signed Certificate**.
   d) On the *Create Self-Signed Certificate* page, in the **Specify a friendly name for the certificate** box, type a friendly name for the certificate (e.g., EcoStruxure Machine SCADA Expert), and then click **OK**.

   Your self-signed certificate is added to the list of server certificates.
5. Enable SSL for your web site:
   a) In the **Connections** list on the left, open your server, open **Sites**, and then select **Default Web Site**.
   b) In the **Actions** pane, click **Bindings**.
   c) On the *Site Bindings* page, click **Add**.
   d) On the *Add Site Binding* page, in the **Type** list, select **https**.
   e) In the **SSL certificate** list, select the self-signed certificate that you created.
   f) Click **OK** to close the *Add Site Binding* page, and then click **Close** to close the *Site Bindings* page.
6. Require clients to connect with SSL:
   a) In the *IIS Manager* window, in **Features** view, double-click **SSL Settings**.
   b) Select **Require SSL**.

   This step is optional. If you have problems connecting to the web site, then you may clear this option and try connecting without SSL.
7. Restart your web site with the new settings:
   a) In the **Connections** list on the left, select **Default Web Site** again.
   b) In the **Actions** pane, click **Restart**.
8. Close IIS Manager.

When you want to deploy your EcoStruxure Machine SCADA Expert project in a production environment, you should request a signed certificate and reconfigure IIS to use that certificate.

### Install the Mobile Access web server add-on for IIS

Use the standalone Mobile Access Runtime software to install the Mobile Access web server add-on for IIS.

> **Note:** You must have administrator privileges in order to install any software in Windows or Windows Embedded.

Before you begin this task, you must have already installed the full EcoStruxure Machine SCADA Expert 2020 software on at least one computer, even if you only use it for project development, because doing so also unpacks the standalone Mobile Access Runtime software installer that you need for this task.

The Mobile Access add-on requires .NET Framework 4.5 or later, and if it is not present, the Mobile Access Runtime software installer will try to install it for you.

In order to install the Mobile Access add-on on the computer or device that hosts your project runtime server, IIS must be turned on and the ASP.NET features required for Mobile Access should be selected. The Mobile Access Runtime software installer will try to confirm that IIS is turned on, and if it is not, the installer will abort the installation. The installer cannot also confirm that the ASP.NET features are selected, however, so you should confirm that yourself before you begin this task. For more information, see Turn on IIS for thin client access on page 768.

You should also enable Secure Socket Layer (SSL) encryption in IIS, especially if you plan to access your project over a public network. For more information, see Enable SSL encryption in Microsoft IIS on page 771.

There are two ways to install the Mobile Access web server add-on for IIS. The first way is to select **Mobile Access Runtime** as an installable feature when you install the full EcoStruxure Machine SCADA Expert 2020 software. For more information, see Install the full EcoStruxure Machine SCADA Expert 2020 software on page 35. If you already did that, however, you do not need to do anything more and you may skip the rest of this task.

The second way is to manually install the software after the fact using the standalone Mobile Access Runtime software installer that is included in your EcoStruxure Machine SCADA Expert 2020 program folder. You must do this if you did not select the installable feature, as described above.

To install the Mobile Access web server add-on for IIS:

1. On the computer or device where you want to install the software, make sure IIS is turned on, but if the default web site is started, use IIS Manager to stop it.
2. Locate the standalone Mobile Access Runtime software installer (`MobileAccessSetup.exe`) in your EcoStruxure Machine SCADA Expert 2020 program folder.

   If EcoStruxure Machine SCADA Expert 2020 was installed at its default location on your computer, the Mobile Access Runtime software installer should be located at:

   ```
   C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
    2020\Redist\WebAddOn\IIS\MobileAccessSetup.exe
   ```

3. Run the installer, and then follow its instructions (i.e., proceed through the installation wizard).

   > **Note:** You might need to run the installer as an administrator, if your own user privileges are not sufficient — right-click the installer, and then on the shortcut menu, click **Run as Administrator**. You will be prompted for an administrator's user name and password.

4. When the installation is finished, use IIS Manager to start the default web site.

To confirm that the add-on was successfully installed, run your project and then try to log on to the Mobile Access web interface. (This assumes you have already enabled and configured Mobile Access in your project, of course.) For more information, see Log on to the Mobile Access web interface on page 792.

If the web interface fails to load — that is, if the Mobile Access Logon screen is not displayed at all — use IIS Manager to confirm that your web server has started and the Mobile Access application pool (StudioMobileAccessPool) has been configured to use .NET Framework 4.x.

Finally, you should repeat this task whenever you upgrade the full EcoStruxure Machine SCADA Expert 2020 software, to ensure that you are using the latest version of Mobile Access.

### MOBILE ACCESS WEB SERVER ADD-ON FOR CGI

The Mobile Access web server add-on for CGI connects your project runtime server with any web server that supports the Common Gateway Interface (CGI), as long as they are both running on the same computer.

### Install and configure Apache for Windows

This is an example of how to install and configure Apache for Windows (including Windows Server), so that you can subsequently install the Mobile Access web server add-on for CGI.

You must have Administrator privileges on a Windows computer in order to install software and run network applications.

Apache (a.k.a. httpd) is a free and widely used web server package that supports the Common Gateway Interface (CGI). It is available for many different platforms, including both Windows and Linux. Once you have it installed and configured, you can also install the Mobile Access web server add-on for CGI.

If Apache is already installed and you are familiar with it, you may skip to the end of this task.

> **Note:** This is for testing and demonstration purposes only. Before you set up a "live" web server, you should consider all of the administrative and security issues that are involved in doing so. Please consult your network administrator.

To install and configure Apache for Windows:

1. On the computer or device that will host your project runtime server, in the web browser, go to: www.apachehaus.com/cgi-bin/download.plx
2. Download the latest version of Apache for Windows.
   At the time of this writing, the latest version is 2.4.16, and that is reflected in the remaining steps of this procedure. Also, make sure that you get correct build for your version of Windows: **x86** for Windows 32-bit, **x64** for Windows 64-bit.
   The compressed folder (e.g., `httpd-2.4.16-x64-r2.zip`) is saved in the **Downloads** folder.
3. Right-click the compressed folder, and then click **Extract All** on the shortcut menu.
   The *Extract* dialog box is displayed, asking you to confirm where the files will be extracted.
4. Click **Extract**.
   The files are extracted to the specified location.
5. Open the uncompressed folder, and then in that folder, find the **Apache24** folder.
6. Copy or move the **Apache24** folder to the top level of the C drive (i.e., the computer's root directory).
7. Click the **Start** button, and then on the **Start** menu, point to **Accessories** > **Command Prompt**.
8. Right-click **Command Prompt**, and then click **Run as administrator** on the shortcut menu.
   A *User Account Control* dialog box is displayed, asking you to allow Command Prompt to make changes to the computer.
9. Click **Yes**.
   The *Command Prompt* window is displayed.
10. At the prompt, type `cd C:\Apache24\bin`, and then press **Return**.
    The prompt is changed to the specified directory.
11. At the prompt, type `httpd -k install`, and then press **Return**.
    Apache is installed as a Windows service, so that it can run in the background (similar to a Unix/Linux daemon).
12. At the prompt, type `httpd -k start`, and then press **Return**.
    The Apache service is started.
13. In the web browser, in the address bar, type `localhost`, and then press **Return**.
    Apache is preconfigured for localhost access, which means you should be able to go to the localhost address (i.e., `http://localhost/`) to access the default website.
    If Apache is running correctly, the website's default page — typically, the readme file — is displayed.

    > **Tip:** You should review the readme file at this time.

The following table shows the most commonly used Apache commands:

| Command | Description |
|---|---|
| `httpd -k install` | Install Apache as a Windows service. |
| `httpd -k config` | Configure the startup options of the Apache service. |
| `httpd -k uninstall` | Uninstall the Apache service. |
| `httpd -k start` | Start the Apache service. |
| `httpd -k restart` | Restart the Apache service while it is running. |
| `httpd -k stop` | Stop the Apache service. |
| `httpd -t` | Test the Apache configuration syntax. |
| `httpd -v` | Show the Apache version number. |
| `httpd -h` | List all of the available Apache commands. |

You can also use Apache Monitor, a desktop tray application, to start and stop Apache services. The Apache Monitor program file is located at `C:\Apache24\bin\ApacheMonitor.exe`. Either double-click the program file to run it, or copy it to your **Startup** folder so that it starts automatically when the computer is turned on.

The default TCP/IP port for most web servers is port 80. If you already have another web server running — or "listening" — on port 80, it will conflict with Apache. Either stop or disable the other web server, or reconfigure Apache to listen on an alternative port. To do the latter, stop Apache, open the configuration file (`C:\Apache24\conf\httpd.conf`), and edit the following settings:

```
Listen <alternative port>
```

```
ServerName localhost:<alternative port>
```

For the complete Apache documentation, go to: httpd.apache.org/docs/2.4/

Once you have Apache installed and configured, you can proceed with installing the Mobile Access web server add-on for CGI. For more information, see Install the Mobile Access web server add-on for CGI on page 775.

### Install the Mobile Access web server add-on for CGI

Install the Mobile Access web server add-on for any web server that supports the Common Gateway Interface (CGI).

Before you begin this task, you must have already installed the full EcoStruxure Machine SCADA Expert 2020 software on your computer, even if you only use it for project development, because it includes the redistributable Mobile Access web server add-on files.

Also, you must properly install, configure, and run a CGI-enabled web server on the same computer that hosts your project runtime server. There are many such web servers available for many different platforms, so it is beyond the scope of this documentation to cover all of the possible installation and configuration procedures.

The most widely used, CGI-enabled web server is Apache, which can be installed on Windows (including Windows Server) as an alternative to Windows' built-in web server, Internet Information Services (IIS). For more information, see Install and configure Apache for Windows on page 774.

IIS also supports CGI, but you must use IIS Manager to enable and configure it, and that is beyond the scope of this documentation. For more information, consult the documentation for IIS.

When you configure the web server, note the locations of its cgi-bin and document root directories. If you followed the preceding instructions to install and configure Apache for Windows, you should have these directories on your target device:

| Directory | Location |
|---|---|
| cgi-bin | `C:\Apache24\cgi-bin\` |
| DocumentRoot | `C:\Apache24\htdocs\` |

The cgi-bin directory contains supplemental scripts and programs that enable certain features of the website. The Mobile Access web server add-on is such a program.

The document root directory is the "top level" of the website, which means that when a user goes to the website (e.g., `http://www.mywebsite.com/`), they actually go to that directory on the computer. You can reconfigure the web server to change the directory, but in most cases, that should not be necessary.

Keep in mind that while the following steps use Apache for Windows as an example, they should apply to any CGI-enabled web server that has cgi-bin and document root directories.

To install the Mobile Access web server add-on for CGI:

1. Locate the web server add-on files in your SCADA program folder, and then copy all of them to the web server's cgi-bin directory.

   If SCADA is installed at the default location on your computer, the add-on files should be located at:

   ```
   C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
    2020\Redist\WebAddOn\CGI\*.*
   ```

   The add-on files include the CGI process itself and several associated libraries. Make sure you copy all of them to the cgi-bin directory.

   For example, using Apache for Windows:

   ```
   C:\Apache24\cgi-bin\WebCGIProc.exe
   ```

2. Locate the Mobile Access web files in your EcoStruxure Machine SCADA Expert 2020 program folder, and then copy the entire `MA` folder to the web server's document root directory.

   If SCADA is installed at the default location on your computer, the `MA` folder should be located at:

   ```
   C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
    2020\Redist\MobileAccess\MA
   ```

   After you have done this, the `MA` folder should be a sub-directory of the document root directory. For example, using Apache for Windows:

   ```
   C:\Apache24\htdocs\MA\
   ```

3. Edit the Mobile Access configuration file (`config.js`) to point to the CGI process:
   a) Locate the Mobile Access website configuration file.

      For example, using Apache for Windows, the file should be located at:

      ```
      C:\Apache24\htdocs\MA\sma\config.js
      ```

   b) Open the configuration file in a text editor, and then find the **`servicesUrl`** setting:

      ```
      window.sma.configSettings = {
          "servicesUrl": "service"
      };
      ```

   c) Replace **`"service"`** with the URL of the CGI process (`WebCGIProc.exe`) that you previously copied to the cgi-bin directory.

      Using Apache for Windows as an example:

      ```
      window.sma.configSettings = {
          "servicesUrl": "/cgi-bin/WebCGIProc.exe"
      };
      ```

Please note the URL is relative to the "top level" of the website, and it is not the same thing as the file path on the computer.

d) Save and close the configuration file.

You might need to restart the web server for these changes to take effect. For example, assuming that you are running Apache as a Windows service (as described in the preceding topic), open a Command Prompt window and then enter the following:

```
httpd -k restart
```

To confirm the files were successfully installed, run your project and then try to log on to the Mobile Access web interface. (This assumes you have configured Mobile Access in your project and your project is running, of course.) For more information, see Log on to the Mobile Access web interface on page 792.

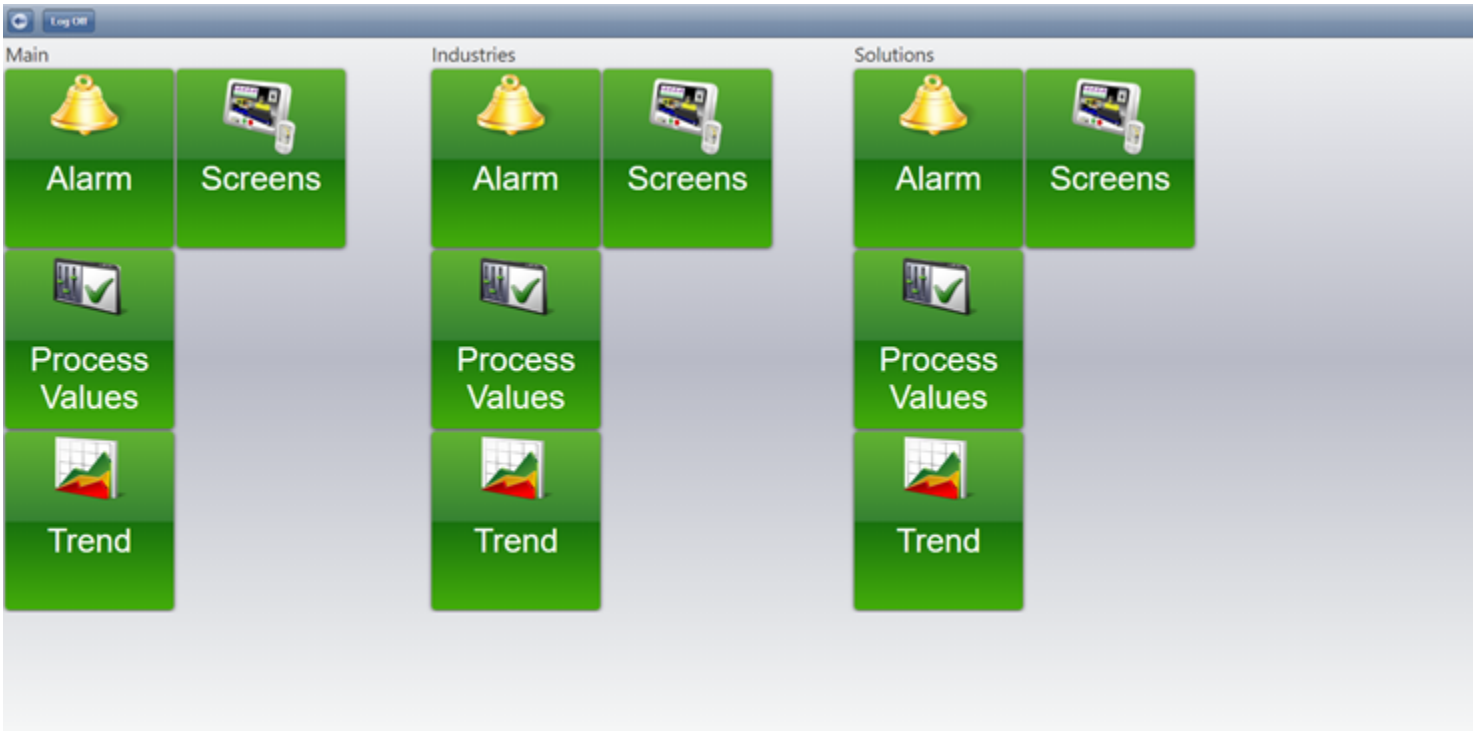
If it does not work, make sure you have selected the **Enable CGI** option in the *Mobile Access Configuration* worksheet in your project. For more information, see Configure the global settings for all areas on page 790.

Finally, you should repeat this task whenever you upgrade the full EcoStruxure Machine SCADA Expert 2020 software, to ensure you are using the latest version of Mobile Access.

### *Configuring the Mobile Access web interface*

Use the Mobile Access Configuration worksheet to configure the Mobile Access web interface for your project.

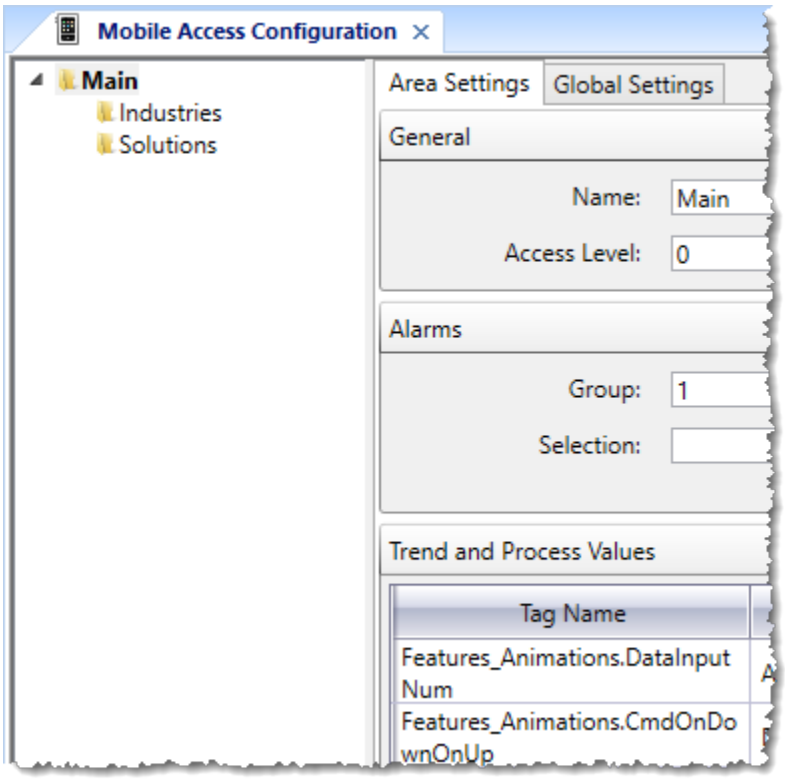
Your project's web interface consists of one or more "areas", which are typically organized by location, system, or machine. Each area has an Alarm control, a Process Values control, a Trend control, and a Screens control. These controls appear in the web interface as green tiles, and when you click/tap one of these tiles, it opens a new page for that control.




***An example of the Mobile Access web interface***

Each area can also have one or more "sub-areas", which appear in the web interface as orange tiles. When you click/tap one of these tiles, it opens a new page for that sub-area.

You can insert as many areas and sub-areas as you want; the web interface is automatically expanded to accomodate them. You use the Mobile Access Configuration worksheet to insert and configure areas.




***Configuring the Mobile Access web interface for your project***

The Alarm, Process Values, Trend, and Screens controls are functionally similar to their corresponding screen objects:

**Alarm**

The Alarm control is similar to the Alarm/Event Control screen object. It displays active alarms and allows the user to acknowledge them.

**Process Values**

The Process Values control is similar to the Symbols library, in that it uses various pre-made widgets (e.g., gauges and switches) to graphically represent project tag values. It can also allow the user to change the values during project run time, depending on how you configure the widgets.

**Trend**

The Trend control is similar to the Trend Control screen object. It graphs the changes in process values during project run time, and it can also display trend history when available.

**Screens**

The Screens control presents project screens that you have selected to include in the web interface. In most cases, selected screens function the same as screens published for Thin Clients, but there are some limitations. For more information, see Supported features in Mobile Access on page 756.

> **Note:** The first time you open and edit the Mobile Access Configuration worksheet, the startup mode of your project's Mobile Access Runtime task is automatically changed to **Automatic**. This is to ensure that Mobile Access will behave as expected when you run your project. You can change the startup mode back to **Manual**, if you wish. For more information, see Runtime Tasks on page 136.

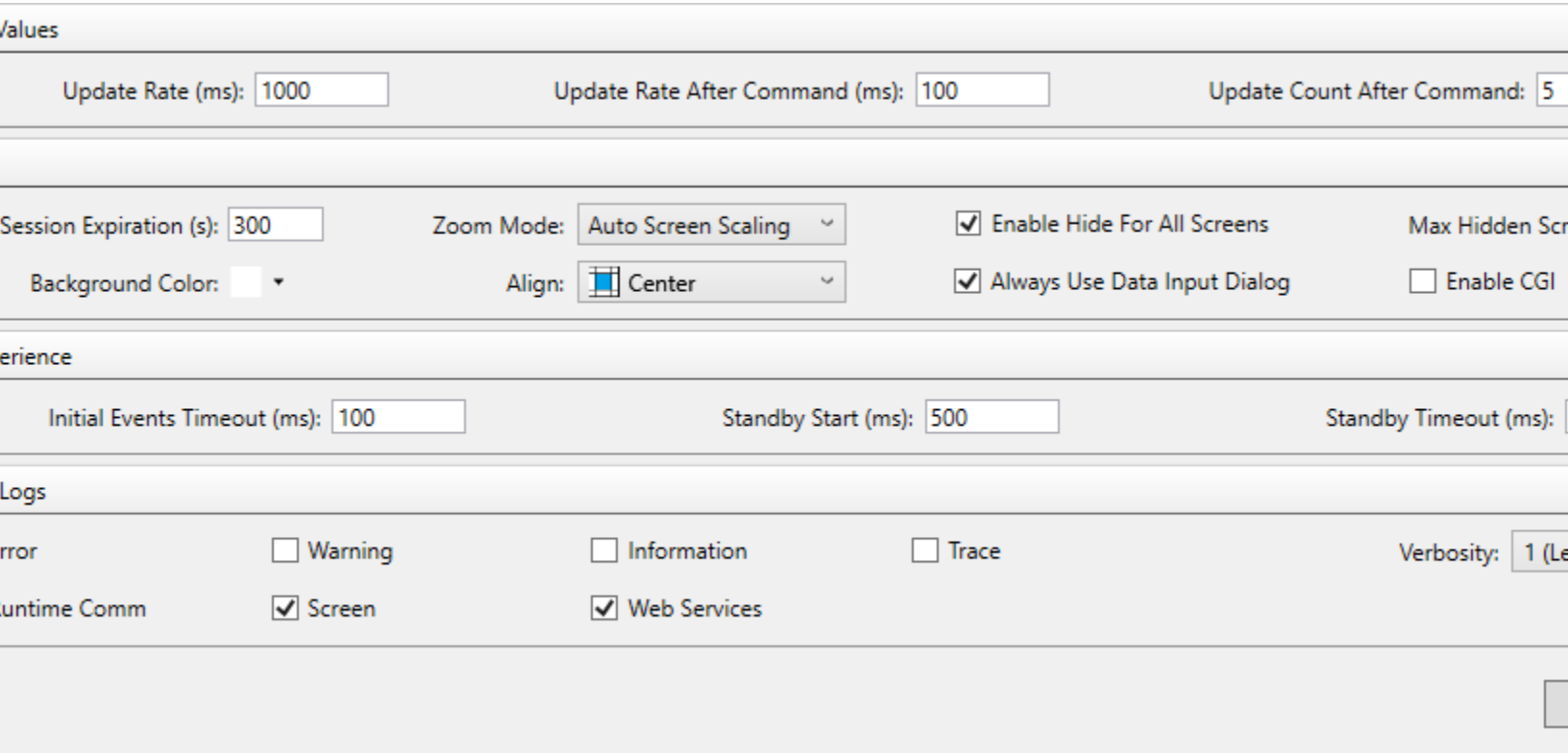
## CONFIGURE THE WEB SETTINGS FOR MOBILE ACCESS

Configure these general settings to determine the overall user experience for viewing project screens through Mobile Access.

To configure the web settings for Mobile Access:

1. If the *Mobile Access Configuration* worksheet is not already open, do one of the following:
   - On the **Project** tab of the ribbon, in the **Web** group, click **Mobile Access**.
   - On the **Graphics** tab of the Project Explorer, double-click **Thin Clients** > **Mobile Access**.

   The *Mobile Access Configuration* worksheet is opened for editing.




2. Under **Process Values**, review the settings that determine how process values are updated during run time.

   **Update Rate**

   The normal rate (in milliseconds) at which process values in screens are updated from the project runtime server. The default rate is 1000 milliseconds (i.e., 1 second). In other words, project screens are normally updated once per second.

   **Update Rate After Command**

   The increased rate (in milliseconds) at which process values are updated after any command is executed in a project screen. This lets the user see immediate changes resulting from the command.

   **Update Count After Command**

   The number of times that process values are updated at the increased rate, after which they return to the normal rate.

   These settings make project screens appear more responsive while users are actually using them. You can increase the rate and/or count, but doing so will put more of a load on your network and the project runtime server. In most cases, you should accept the default values.

3. Under **Web**, in the **Session Expiration** box, type the number of seconds of inactivity that will be allowed before a client session expires.

The default period is 300 seconds (or 5 minutes). A session is considered active as long as the client is connected to the server and a user is logged on, even if there is no user input. The session becomes inactive when the user logs off or the client loses its connection to the server. Then, after the specified period of inactivity, the session expires and the user must reload the Mobile Access web interface in order to reconnect to the server.

4. In the **Zoom Mode** list, select how you want your project screens to be displayed in the browser window.
   The following options are available:

| Option | Description |
|---|---|
| **Disabled** | In this mode, screen zoom is disabled entirely and the project is displayed at full resolution, regardless of the size of the browser window. All of the screens are displayed in their specified sizes and positions (as configured in the Screen Attributes for each screen), relative to the top-left corner of the browser window. Resizing the browser window does not affect the screens in any way. If a screen is configured so that some or all of its area will be displayed outside the available area of the browser window, such screen area will not be visible. The browser window does not include scroll bars, and the user cannot use "pinch" and "stretch" gestures to zoom the screens. |
| **Auto Screen Scaling** | In this mode, all of the screens are scaled proportionally according to the ratio between the project's display resolution and the size of the browser window. As long as a screen's specified size and position (as configured in Screen Attributes) do not exceed the project's display resolution, the screen will be fully visible within the browser window. Resizing the browser window also resizes all of the screens displayed within the browser window. The browser window does not include scroll bars, and the user cannot use "pinch" and "stretch" gestures to zoom the screens.<br><br>When the system calculates the ratio between the project's display resolution and the size of the browser window, it keeps the original width/height ratio of the display resolution in order to avoid distorting the contents of the screens. For example, if the specified display resolution is 1000x500 (2:1) and the browser window is 500x100 (5:1), the maximum display resolution that will actually fit within the browser window is 200x100 (2:1). Therefore, each screen's size/position will be divided by 5 (1000/200 or 500/100) before it is displayed in the browser window.<br><br>This is the default option for new projects. |
| **Custom Zoom** | This mode is the same as **Disabled** mode (see above), except that the user can use "pinch" and "stretch" gestures to zoom the screens. The zoom is applied to all open screens at the same time. If the project's display resolution is smaller than the size of the browser window, the user can zoom in so that the screens fill the window. Conversely, if the project's display resolution is larger than the size of the browser window, the user can zoom out so that the screens fit within the window. When zooming, the system keeps the original width/height ratio of each |

| Option | Description |
| --- | --- |
| | screen in order to avoid distorting the contents of the screen.<br><br>Once the user zooms in so that the screens fill the browser window, that becomes the minimum level of zoom. |
| **Single Screen Scaling** | In this mode, each screen is scaled proportionally according to the ratio between its own specified size (as configured in Screen Attributes) and the size of the browser window. The screen's specified position is ignored, because the screen is automatically centered in the browser window. The project's display resolution is ignored as well. Since each screen is resized and centered separately, this mode is not useful for displaying more than one screen at the same time, such as screen groups. Only the last screen opened is visible; previously opened screens are automatically closed. The browser window does not include scroll bars, and the user cannot use multi-touch gestures (i.e., "pinch and stretch") to zoom screens.<br><br>**Note:** When the system calculates the ratio between the screen's specified size and the size of the browser window, it keeps the original width/height ratio of the screen size in order to avoid distorting the contents of the screen. For example, if the specified screen size is 1000x500 (2:1) and the browser window is 500x100 (5:1), the maximum screen size that will actually fit in the browser window is 200x100 (2:1). Therefore, the screen's size will be divided by 5 (1000/200 or 500/100) before it is displayed in the browser window. |

5. Review the **Enable Hide for All Screens** and **Max Hidden Screens** settings.

   Project screens can be hidden instead of actually closed, so that they appear to reopen quickly. This will increase the responsiveness of your project screens, but it will also use more resources on client devices.

   **Enable Hide for All Screens**

   This option enables hiding for all project screens viewed through Mobile Access.

   **Max Hidden Screens**

   The maximum number of project screens that can be hidden and then cached in memory. Screens are cached on a First In First Out (FIFO) basis. You might need to adjust the number depending on the available resources on client devices.

   These settings are similar to the **Performance Optimization** settings in the Screen Attributes, except that they apply equally to all project screens viewed through Mobile Access rather than on a screen-by-screen basis.

6. Review the **Background Color** and **Align** settings.

   Even when zooming is enabled, your project screens often will not perfectly fill the browser window; the browser window's background color will be visible around the sides of your project screens. To minimize the contrast, align the project screens within the browser window and then adjust the background color to match.

   **Background Color**

   The color that fills the browser window around the project screen. Click the color picker and then select a new color, if you wish.

**Align**

The alignment of the project screen within the browser window: Top Left, Top Center, Center Left, Center (default).

> **Note:** The **Align** setting applies only when **Zoom Mode** is **Auto Screen Scaling**. Each of the other modes overrides the alignment in some way.

7. Select **Always Use Data Input Dialog** if you want to display the Data Input dialog box for all screen objects that take user input.

   The Data Input dialog box in Mobile Access provides an easy-to-use interface with an on-screen keyboard that allows the user to type their input on touchscreen devices. This option is selected by default in order to ensure that your project will be usable on small screens, where some screen objects might be too small to type into without zooming. If this option is cleared, the Data Input dialog box will be displayed only for certain screen objects that do not have input boxes at all, such as a Text object with a Text Data Link animation applied to it. For more information, see Data input in screens on Mobile Access on page 349.

8. Select **Enable CGI** if you have a set up a CGI-enabled web server (e.g., Apache) to serve the Mobile Access web interface to users.

9. Under **User Experience**, review the settings that determine how feedback is given to the user when the project runtime server is slow to respond.

   **Initial Events Timeout**

   When the user tries to open a project screen in Mobile Access, some requests are sent to the server in order to load and initialize that screen. If the server does not respond within the period specified by this setting, an appropriate message is sent to the activity log. For example:

   ```
   [Warning, 1]:Mobile Access([05E95210] HTML5 Server => Timeout waiting for
    graphic initialization [03/11/2020 15:56:28.966]
   ```

   **Standby Start**

   When the user tries to click/tap a screen object (e.g., a button), the command associated with that object is sent as a request to the server. If the server does not respond within the period specified by this setting, a "spinning wheel" animation is displayed in front of the object.

   **Standby Timeout**

   After the "spinning wheel" animation is displayed in front of the object, if the server still does not respond within the period specified by this setting, the request is cancelled and the animation is removed.

   If the project runtime server is often slow to respond, your project's user experience can be significantly affected. Check your network activity, the number of thin clients connected to server, and the status/health of the server itself.

10. Under **Browser Logs**, select the types of log messages that you want to send to Mobile Access clients.

    Mobile Access generates its own activity log. It is similar to the project runtime log displayed in the Output window and LogWin tool, but it comprises only messages about Mobile Access itself and the performance of the web interface. Users can view the activity log in their browser consoles. For more information about the available options, see Types of Mobile Access log messages on page 809.

> **Note:** The settings in this worksheet apply to all Mobile Access clients; i.e., except for messages about the specific project screens that are currently open on each client, the same log messages are sent to all clients and can be viewed by all users. If an individual user wants to filter their own view of the log, they should use the tools in their own browser console.

When you are done, you can click *Area Settings* to configure the specific areas of the Mobile Access web interface, or you can close the worksheet altogether. The entire Mobile Access configuration is saved when you close the worksheet.

## INSERT A NEW AREA IN THE WEB INTERFACE

Insert a new node, or "area", in the Mobile Access tree view in order to add a new page to the Mobile Access web interface.

The site map of the Mobile Access web interface is determined by the hierarchical tree on the left side of the Mobile Access Configuration worksheet. The root node of the tree, named **Main**, corresponds to the home page of the web interface. If you wish, you can configure the settings for **Main** so that it includes everything you want to display in the web interface. You are not required to add to the tree, and if you choose not to, you can skip this task and proceed to configuring the settings for **Main**.

However, if you want organize your project tags in some way — for example, by machine, by process, or by facility — you can insert additional nodes, or "areas", into the tree. Each area has its own settings and is represented by its own page in the Mobile Access web interface. Areas can have as many sub-areas as you want, as many levels deep as you want. The structure of the tree, and therefore the site map of the web interface, is entirely up to you.

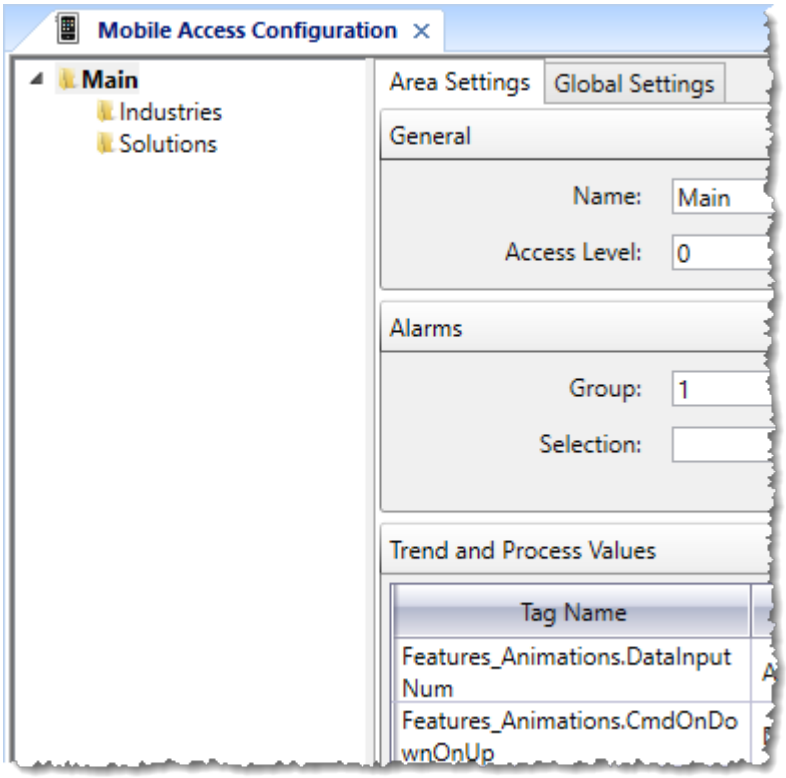
Keep in mind, however, that the structure of the tree determines how the user must navigate the Mobile Access web interface during run time. You may choose to have all of the areas together on the same level, but if you do, then the user must pan/scroll a lot to move between the areas. Alternatively, you may choose to create many levels of areas and sub-areas, but if you do, then the user must click/tap down through those levels and then back up again. In short, it is important to keep your web interface logically organized and easy to navigate, and as such you should manually outline your tree before you begin to insert areas.

To insert an area in the Mobile Access tree view:

1. If the *Mobile Access Configuration* worksheet is not already open, do one of the following:
   - On the **Project** tab of the ribbon, in the **Web** group, click **Mobile Access**; or
   - On the **Graphics** tab of the Project Explorer, double-click **Thin Clients** > **Mobile Access**.

   The *Mobile Access Configuration* worksheet is opened for editing, and the *Web Settings* page of the worksheet is displayed.
2. At the bottom of the *Web Settings* page, click **Area Settings**.

The *Area Settings* page of the worksheet is displayed, with the tree view displayed on the left. It replaces the *Web Settings* page.




***An example of the Mobile Access tree view / site map***

3. Carefully determine where in the tree view you want to insert the new area, because once you have inserted it, it is not possible to move it using the Mobile Access Configuration worksheet.
4. Right-click on the existing area in which you want to insert the new area, and then on the shortcut menu, click **Insert Area**.
   A *New Area* dialog box is displayed.
5. In the **Area name** box, type the name of the new area, and then click **Add**.
   The new area is inserted in the tree view.

After you have inserted the new area, you can configure the settings for that area.

To delete an area that you have inserted, right-click the area, and then on the shortcut menu, click **Delete Area**. Please note that when you delete an area, you also delete all of the sub-areas that it contains. You cannot delete **Main**.

> **Note:**
>
> Your Mobile Access configuration is saved as an XML file in your project folder at:
>
> ***<project name>*\Web\MobileAccess.sma**
>
> If necessary, you can manually edit this file to make changes that cannot be made in the Mobile Access Configuration worksheet. For more information, please contact your software distributor.

## CONFIGURE THE SETTINGS FOR A SELECTED AREA

Each area in the Mobile Access tree / site map has its own area settings. Configure these settings to determine which alarms, trends, process values, and project screens to display on that area's corresponding page in the Mobile Access web interface.

The default area, at the root of the Mobile Access tree view, is "Main". Every project that has Mobile Access enabled has a "Main" area. You can rename if from "Main" to something else that is more appropriate to your project, but it is always the default area and it is always displayed first in the Mobile Access web interface.

The first time you open the *Area Settings* worksheet, it automatically displays the settings for the "Main" area. This task shows how to edit those settings, but the same procedure applies to editing the settings for any selected area.

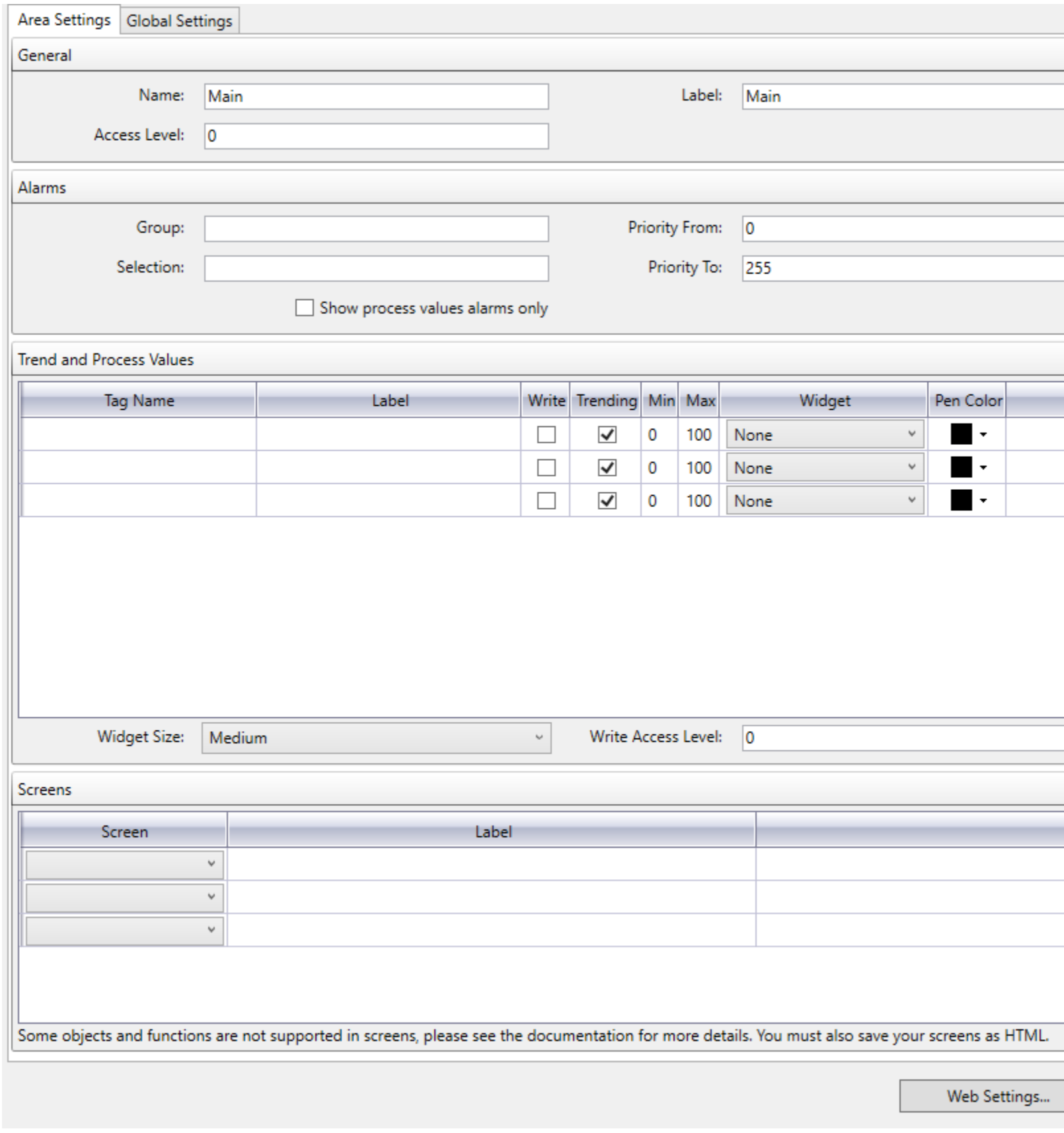
For more information about the Mobile Access tree view, how it determines the site map of the Mobile Access web interface, and how to insert additional areas, see Insert a new area in the web interface on page 783.

To configure the settings for a selected area:

1. If the *Mobile Access Configuration* worksheet is not already open, do one of the following:
   - On the **Project** tab of the ribbon, in the **Web** group, click **Mobile Access**.
   - On the **Graphics** tab of the Project Explorer, double-click **Thin Clients** > **Mobile Access**.

   The *Mobile Access Configuration* worksheet is opened for editing, and the *Web Settings* page of the worksheet is displayed.
2. At the bottom of the *Web Settings* page, click **Area Settings**.
   The *Area Settings* page of the worksheet is displayed. It replaces the *Web Settings* page.
3. In the tree view, select the area for which you want to configure settings.

   The *Area Settings* worksheet for that area is displayed.


Area Settings
Global Settings
General
Name:
Main
Label:
Main
Access Level:
0
Alarms
Group:
Priority From:
0
Selection:
Priority To:
255
Show process values alarms only
Trend and Process Values
Tag Name
Label
Write
Trending
Min
Max
Widget
Pen Color
0
100
None
Widget Size:
Medium
Write Access Level:
0
Screens
Screen
Label
Some objects and functions are not supported in screens, please see the documentation for more details. You must also save your screens as HTML.
Web Settings...

4. Under **General**, configure the general settings for this area.
   a) In the **Name** box, type the name of the area as it should appear in the tree view.
   b) In the **Label** box, type the area label that should be displayed in the Mobile Access web interface during run time. Typically, the label is the same as the full name, but you may choose to abbreviate it if the full name cannot be displayed correctly in the Mobile Access web interface.
   c) In the **Access Level** box, type the minimum security level that the user must have in order to access the area during run time.
5. Under **Alarms**, specify which alarms should be displayed in this area's Alarm window during run time.
   By default, all areas of the Mobile Access web interface will display all alarms that are configured in your project's Alarm worksheets. However, you can filter the alarms by group, by selection, or by priority in order to display only the alarms that are relevant to this area. The easiest way to do this is to configure a separate Alarm worksheet/group for each area and then filter by those group numbers, but if you configured your project's alarms long before you configured the areas of your Mobile Access web interface, then they may not correspond. If that is the case, you can use the other settings. For more information about group, selection, and priority, see Alarm worksheet on page 377.
   a) In the **Group** box, type the number of the Alarm group(s) that you want to display in this area.
      You can specify more than one group by using commas and hyphens. For example, if you type…

      ```
      5-10,60,80-90
      ```

      …you will display groups 5 through 10, group 60, and groups 80 through 90 in this control.
      The **Group** box will validate as you type; if you type an invalid group or groups, then the box will be bordered in red.
      If you leave this box empty, no filtering will be done by Alarm group.
   b) In the **Selection** box, type the selection alias(es) of the specific alarms that you want to display in this area.
      You can specify more than one selection alias by using commas. For example…

      ```
      AliasA,AliasB,AliasD
      ```

      If you leave this box empty, no filtering will be done by selection alias.
   c) In the **Priority From** and **Priority To** box, type the priority range of the specific alarms that you want to display in this area.
      If you leave the default values of `0` and `255` (i.e., the maximum range), no filtering will be done by priority.
   d) Select **Show process values alarms only** if you only want to show alarms for the process values that are actually configured for this area (see below).
      If you select this option, it will override all of the other alarm filter settings.
6. Under **Trend and Process Values**, select the values that you want to display in this area's Trend and Process windows during run time. For each row, do the following:
   a) In the **Tag Name** column, type the name of a project tag that you want to display as a process value.
      You can also double-click in the box to open the Object Finder.
   b) In the **Label** column, type a simple label for the project tag.
   c) Select **Write** if you want to let the user write new values to the tag by manipulating the corresponding widget (e.g., push the button, slide the gauge, toggle the switch).
   d) Select **Trending** if you want the process value to be graphed as a trend in the area's Trend window, rather than displayed as a widget in the area's Process Values window.
   e) In the **Min** and **Max** columns, type the minimum and maximum for the process value.
      Please note that these are not hard limits on the process value. The actual value of the project tag can exceed both the minimum and maximum, depending on how the value is calculated during run time.

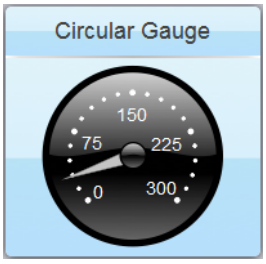
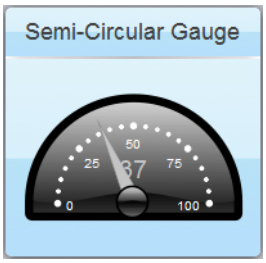
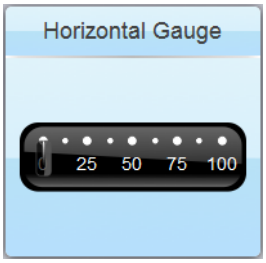
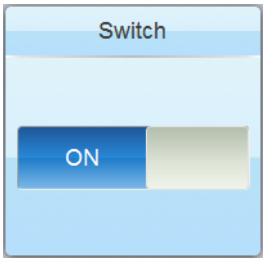
The actual value simply will not be displayed, because it is literally off the scale. Instead, when the actual value is less than the minimum, the minimum is displayed, and when the actual value is greater than the maximum, the maximum is displayed.

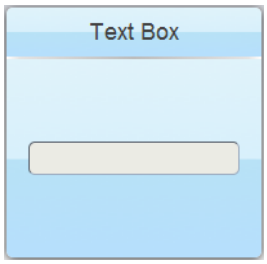
In the Process Values window, **Min** and **Max** are used to determine the scale of the selected widget.

In the Trend window, **Min** and **Max** are used to calculate the percentage. (For example, if **Min** is `10`, **Max** is `20`, and the actual value is 16, then the percentage is 60%.)

f) In the **Widget** column, select the type of widget that should be used to represent the process value in the Process Values control.

The following widgets are available.

| Option | Tag Types | Appearance |
|---|---|---|
| **Circular Gauge** | Integer, Real |  |
| **Semi-Circular Gauge** | Integer, Real |  |
| **Horizontal Gauge** | Integer, Real |  |
| **Switch** | Boolean |  |

| Option | Tag Types | Appearance |
|---|---|---|
| **Text Box** | any |   |

If you select **None**, the process value will not be displayed in the Process Values window.

This setting has no effect if the **Trending** option is selected.

g) In the **Pen Color** column, use the color picker to pick a pen color for the process value.
This setting has no effect if the **Trending** option is cleared.

h) In the **Widget Size** list, select how large the widgets should be displayed.
Larger widgets are clearer and easier to use on mobile devices, but they also take up more screen space.

i) In the **Write Access Level** box, type the minimum security level that the user must have in order to write new values to project tags.

7. In the **Screens** area, specify which project screens should be made available through this area's Screens window. For each row, do the following:

a) In the **Screen** column, click the list menu, and then select a project screen.

> **Tip:**
>
> The list menu should include all project screens that have been saved as HTML. If you do not see the screen(s) that you want to select, do one of the following:
>
> - Verify your project (i.e., on the **Home** tab of the ribbon, in the **Tools** group, click **Verify**); or
> - Re-publish your project screens for the web (i.e., go to **File**, and then click **Save All As HTML**).
>
> Screen names should not include spaces. If a screen name does include a space, save that project screen with a new file name and then re-publish it for the web.

b) In the **Label** column, type a label for the project screen that you selected.
This label is displayed only in the Mobile Access web interface; it is not saved with the original screen file.

Any project screen that has been saved as HTML can be selected for Mobile Access. However, not all screen objects and animations are fully supported in Mobile Access at this time, so make sure that you test the selected screens before you deploy your project. Also, even though you can open multiple screens and screen groups in Mobile Access, you cannot make screen groups available through the Screens control. For more information, see Supported features in Mobile Access on page 756.

Also, with regards to project security, the **Disable** and **Security** settings are enforced on all screen objects and animations, but the **E-sign** setting is not.

When you are done, you can either return to the *Web Settings* page of the worksheet or close the worksheet altogether. The entire Mobile Access configuration is saved when you close the worksheet.

## CONFIGURE THE GLOBAL SETTINGS FOR ALL AREAS

Configure the Mobile Access global settings in order to set default values — such as alarm columns and colors, trend duration, and update rates — for the Alarm and Trend windows in all of the configurable areas in the Mobile Access web interface.

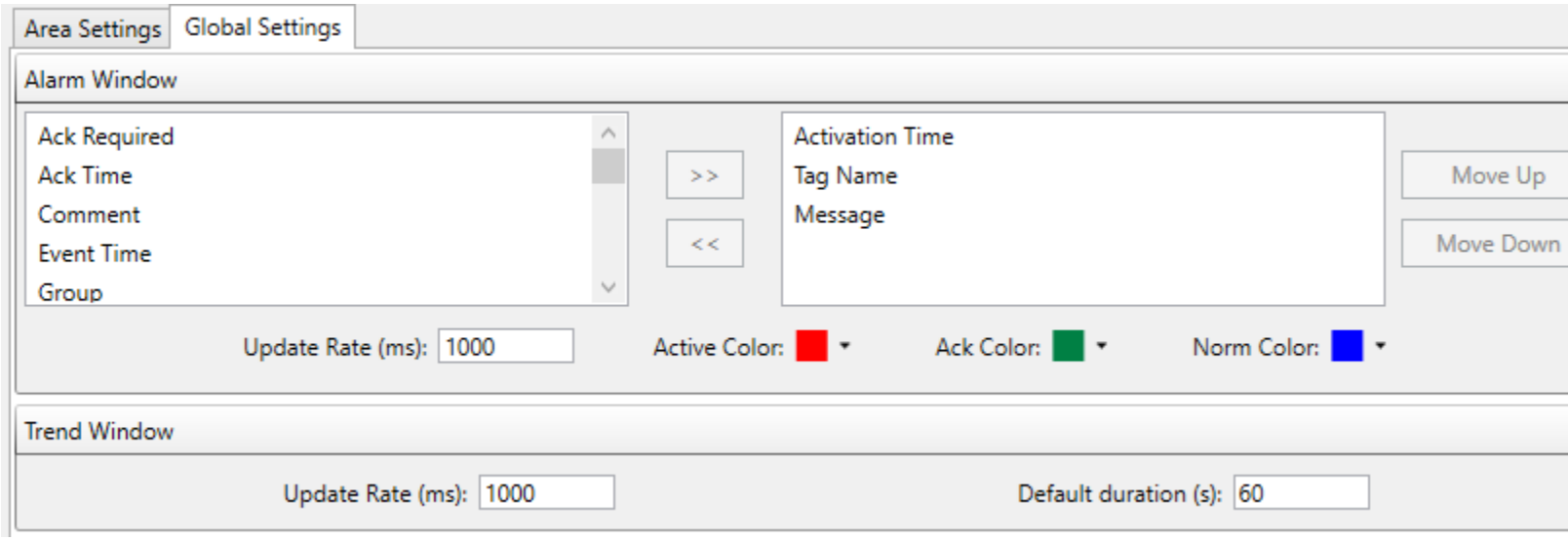
> **Note:** These settings do not apply to Alarm Control and Trend Control objects in project screens that are viewed through Mobile Access. Those objects have their own settings that you configure when you develop your project screens.

To configure the global settings for all areas:

1. If the *Mobile Access Configuration* worksheet is not already open, do one of the following:
   - On the **Project** tab of the ribbon, in the **Web** group, click **Mobile Access**.
   - On the **Graphics** tab of the Project Explorer, double-click **Thin Clients** > **Mobile Access**.

   The *Mobile Access Configuration* worksheet is opened for editing, and the *Web Settings* page of the worksheet is displayed.
2. At the bottom of the *Web Settings* page, click **Area Settings**.
   The *Area Settings* page of the worksheet is displayed. It replaces the *Web Settings* page.
3. At the top of the *Area Settings* page, click the **Global Settings** tab .

   The **Global Settings** tab is displayed.




4. Under **Alarm Window**, select and arrange the columns that the Alarm windows should display by default.

   > **Note:** These settings can be overriden by the user in the client-side settings.

   The available columns are listed on the left. The displayed columns are listed on the right. For more information about what each column means, see Alarm/Event Control object on page 401.

   a) To display a column, select it in the list of available columns and then click the **>>** button.

   b) To hide a column, select it in the list of displayed columns and then click the **<<** button.

   c) To arrange the displayed columns, select a column and then click **Move Up** or **Move Down**.
   The order in which the columns are listed here is the order in which they will be displayed, from left to right, in the alarm control.

5. In the **Update Rate** box, type the rate (in milliseconds) at which Alarm windows should be updated during run time.

   The default rate is 1000 milliseconds (i.e., 1 second), which means each Alarm window is updated once per second. You can increase the rate, but doing so will put more of a load on your network and the project runtime server. In most cases, you should accept the default rate.

6. Select the alarm colors: for each alarm state (Active, Acknowledged, Normalized), click the color picker and then select a new color.

7. Under **Trend Window**, in the **Update Rate** box, type the rate (in milliseconds) at which Trend windows should be updated during run time.

   The default rate is 1000 milliseconds (i.e., 1 second), which means each Trend window is updated once per second. You can increase the rate, but doing so will put more of a load on your network and the project runtime server. In most cases, you should accept the default rate.

8. In the **Default duration** box, type the number of seconds of trend history that each Trend window should display by default.

   The default value is 60 seconds (i.e., 1 minute). However, this value can be overriden by the user in the client-side settings.

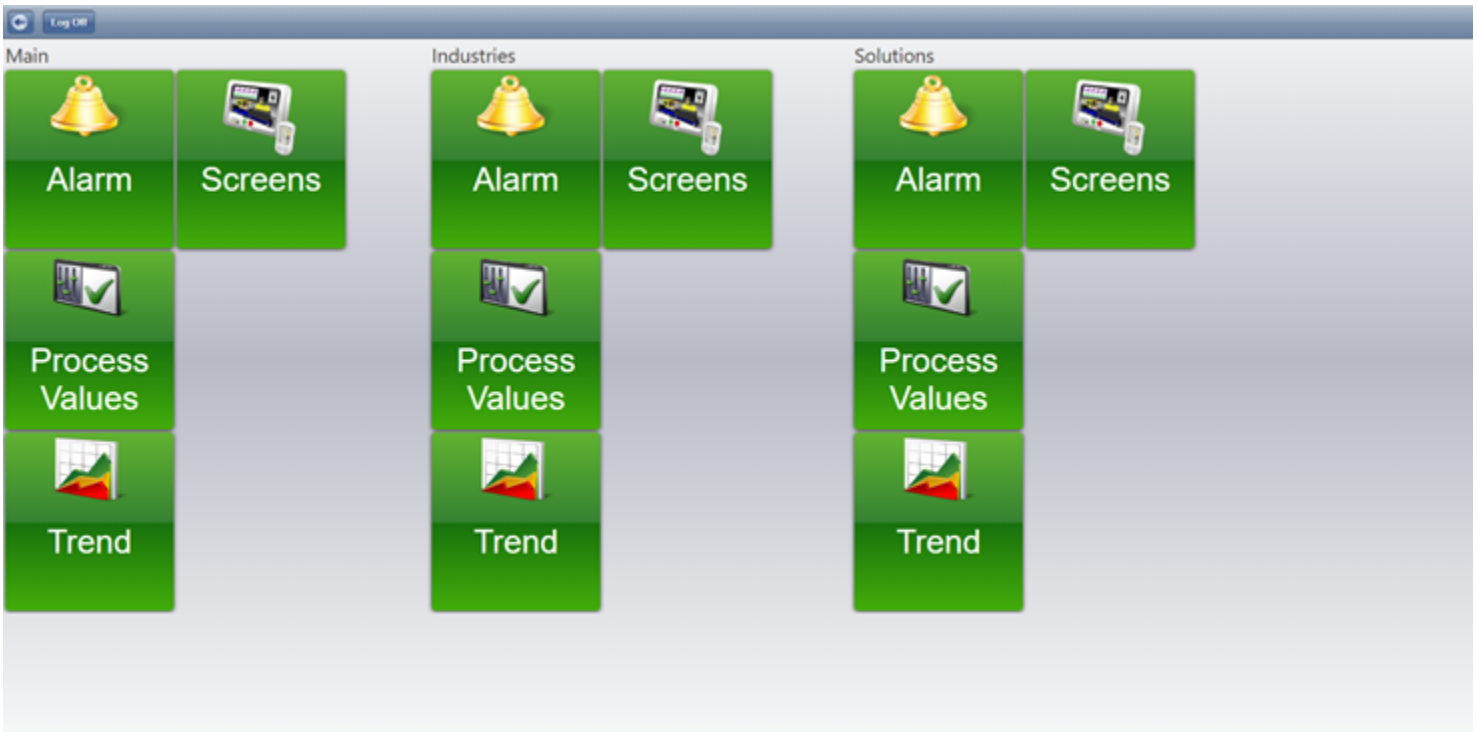
When you are done, you can either return to the *Area Settings* tab of the worksheet or close the worksheet altogether. The entire Mobile Access configuration is saved when you close the worksheet.

### *Navigating the Mobile Access web interface*

Navigate the Mobile Access web interface as you configured it for your EcoStruxure Machine SCADA Expert project.

Before you can use the web interface, the Mobile Access Runtime software must be properly installed and configured on Microsoft IIS your EcoStruxure Machine SCADA Expert project must be running.

The home page of the web interface shows the top-level "Main" area and the other second-level areas that you configured in the Mobile Access Configuration worksheet. The page is automatically created as wide as it needs to be to accomodate all of the areas, but that means some of the page may be out of view depending on the size of your browser window or mobile device.



Each area has an Alarm control, a Process Values control, a Trend control, and a Screens control, represented by the green buttons. Some areas may also have sub-areas, if that is how you structured your web interface in the Mobile Access Configuration worksheet. Sub-areas are represented by orange buttons.

To navigate the Mobile Access web interface:

1. Pan/scroll left or right to find the area that you want.
2. Click/tap a green control button to access that control, or click/tap an orange sub-area button to access that sub-area.
   The button spins to show that it was clicked/tapped, and then the page for the selected control or sub-area is displayed.
3. On any page other than the home page, click/tap the **Return** button to return to the previous page.

## LOG ON TO THE MOBILE ACCESS WEB INTERFACE

Log on to the Mobile Access web interface to view the configured alarms, process values, trends, and project screens during run time.

The Mobile Access web interface uses HTML5 (including CSS3 and AJAX) to create animated graphics and perform real-time data exchange. That means you must use an HTML5-compatible browser to access the web interface. Most modern browsers are HTML5-compatible, but browsers are updated frequently and the HTML5 specification itself is still being revised, so we cannot provide a comprehensive list of browsers that can be used with Mobile Access.

That being said, we recommend you use either Microsoft Edge or Google Chrome. They are available for most platforms and operating systems, and we have found that they provide the best overall performance and compatibility.

We do not support any version of Internet Explorer for use with Mobile Access.

To log on to your project's Mobile Access web interface:

1. On your computer or mobile device, use the web browser to go to the *Mobile Access Logon* page:

   **`https://<server address>/ma/`**

   You should always include the HTTPS protocol (**`https://`**)before the server address in order to connect securely via SSL. If you do not, your connection might not be secure or the server might not allow you to connect at all, depending on how it is configured. For more information, see Enable SSL encryption in Microsoft IIS on page 771.

   For *<server address>*, do one of the following:

   - If the server is a computer on your local network, type the host name (or IP address, if you know it) of that computer. For example:

     **`https://station01/ma/`**

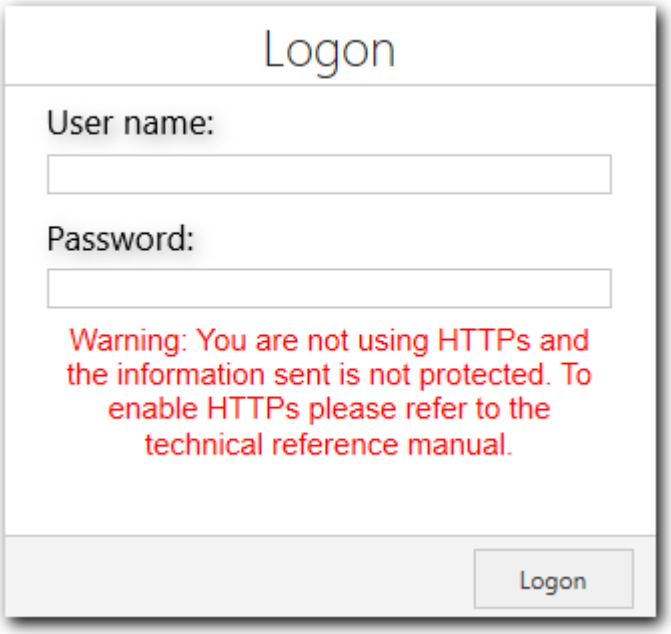
   - If the server is a computer on the Internet, type is the full domain name (or IP address, if you know it) of that computer. For example:

     **`https://scada.ourcompany.com/ma/`**

   - If the server is located behind a network router that has been configured to do port forwarding, type the public IP address of that router followed by the port number that has been assigned to the server. For example:

     **`https://200.128.128.0:3040/ma/`**

   The *Mobile Access Logon* page is displayed in the browser.

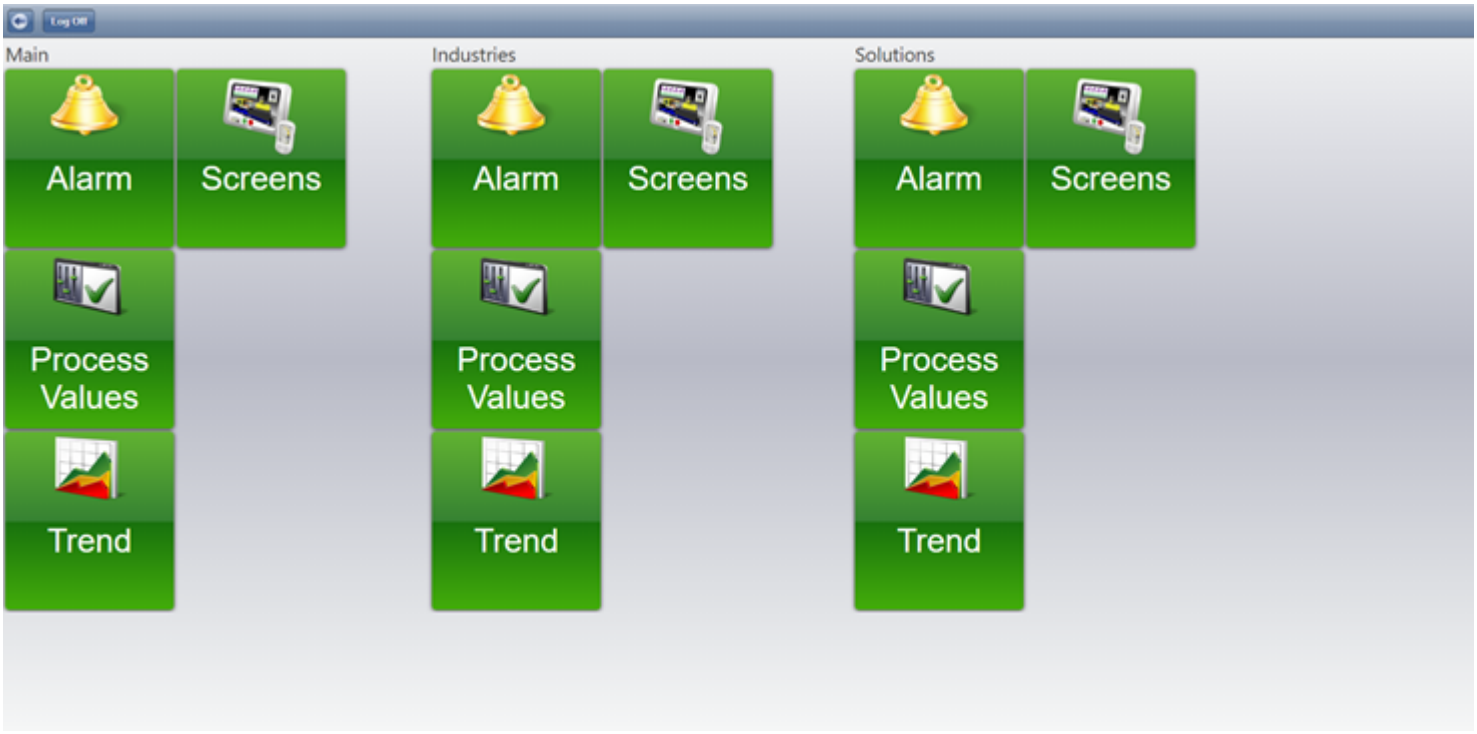



If you are not connecting via SSL, a message is also displayed to remind you that your connection is not secure.

If you are connecting via SSL for the first time, you may be prompted to accept the host's SSL certificate. This is done automatically for most certificates issued by trusted Certificate Authorities, but if your web server is using a self-signed certificate, you may need to manually install that certificate on your computer or mobile device. The procedure to do this varies by operating system, so for more information, consult the documentation for your computer or mobile device.

2. Type your user name and password in the respective boxes in the *Mobile Access Logon* page, and then either press **Return** on your keyboard or click/tap the green arrow.

You are logged on to the Mobile Access web interface, and the home screen is displayed.




***Example of the Mobile Access home screen***

Alternatively, an alert message might be displayed when you try to log on. Remember the project runtime server and the web server are two separate processes; if the project is not running, you may be able to go to the *Mobile Access Logon* page but you will not be able to log on to the project. Make sure the project is running, and then try again.

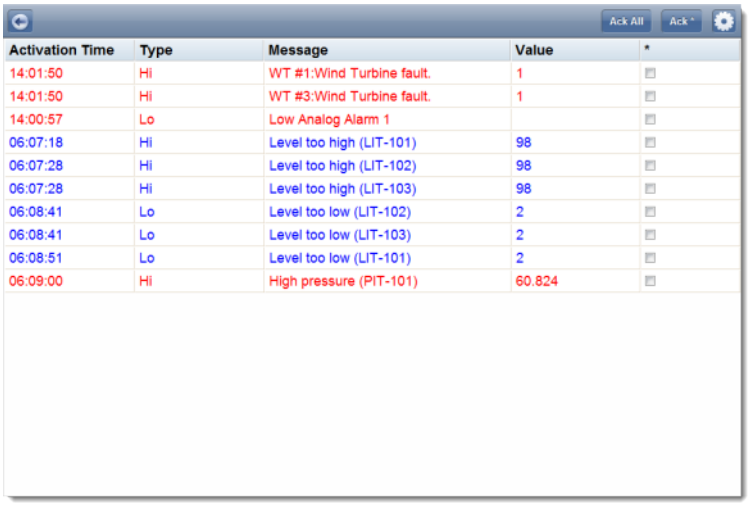
You can open only one session of the Mobile Access web interface per browser. You cannot open new sessions in additional tabs or even in additional instances of the same browser. So, for example, if you want to open two sessions of the web interface on the same computer, you must open the first in Google Chrome and the second in Microsoft Edge.

To log off from the Mobile Access web interface, click/tap the **Return** button to return to the home screen, and then on the home screen, click/tap **Log Off**.

## USE THE ALARM WINDOW

Use the Alarm window in the Mobile Access web interface to view and acknowledge alarms.

The Alarm window is similar to the Alarm/Event Control screen object. It displays online alarms and allows the user to acknowledge them.




| Activation Time | Type | Message | Value | * |
|---|---|---|---|---|
| 14:01:50 | Hi | WT #1:Wind Turbine fault. | 1 | |
| 14:01:50 | Hi | WT #3:Wind Turbine fault. | 1 | |
| 14:00:57 | Lo | Low Analog Alarm 1 | | |
| 06:07:18 | Hi | Level too high (LIT-101) | 98 | |
| 06:07:28 | Hi | Level too high (LIT-102) | 98 | |
| 06:07:28 | Hi | Level too high (LIT-103) | 98 | |
| 06:08:41 | Lo | Level too low (LIT-102) | 2 | |
| 06:08:41 | Lo | Level too low (LIT-103) | 2 | |
| 06:08:51 | Lo | Level too low (LIT-101) | 2 | |
| 06:09:00 | Hi | High pressure (PIT-101) | 60.824 | |

***An example of the Alarm window***

By default, active alarms are written in red, acknowledged alarms are written in green, and normalized alarms are written in blue. However, you can change these colors in the Global Settings tab of the Mobile Access Configuration worksheet.

The grid columns are also similar to those in the Alarm/Event Control screen object. The following columns are included by default:

**Activation Time**

The time when the alarm became active.

**Type**

The type of alarm (e.g., HiHi, Hi, Lo, LoLo).

**Message**

The message that was displayed when the alarm became active.

**Value**

The current value of the affected project tag.

Like the default alarm colors, you can change these default columns in the Global Settings tab of the Mobile Access Configuration worksheet. However, those changes will apply to the entire Mobile Access website. If you only want to change the columns for a specific Alarm window, during a specific user session, see below.

To use the Alarm window:

1. To acknowledge all currently active alarms:
   a) Click/tap **Ack All**.
   The *Ack* screen is displayed.
   b) Type a comment that will be saved with the acknowledged alarms, and then click/tap **Confirm**.
   The alarms are acknowledged.
2. To acknowledge only selected alarms:
   a) In the * column of the grid, select the alarms that you want to acknowledge.

   > **Note:** The Alarm window will not be updated while you have alarms selected.

   b) Click/tap **Ack ***.

The *Ack* screen is displayed.

c) Type a comment that will be saved with the acknowledged alarms, and then click/tap **Confirm**.

The alarms are acknowledged.

3. To change the grid columns for a specific Alarm window:

a) Click/tap the **Settings** button .
The *Settings* screen is displayed with a list of all of the available grid columns and a toggle switches for each column.

b) Click/tap the switches to turn the columns on or off.

c) Click **Alarm** to apply your changes and return to the previous screen.

In most cases, these columns are the same as — and display the same information as — the columns in the Alarm/Event Control object. The one exception is the **Group** column, which displays the group number (e.g., `Group 1`) instead of the group name for the specific alarm.

4. To return to the home screen, click/tap the **Return** button .

## USE THE PROCESS VALUES WINDOW

Use the Process Values window to view tag values as graphical widgets and also to update selected tags.

The Process Values window is similar to the Symbols library. It uses various pre-made widgets (i.e., gauges and switches) to graphically represent project tag values. It can also allow the user to change the values during run time, depending on how you configure the widgets.

***An example of the Process Values window***

All widgets are continuously updated to show the current values of their associated project tags.

If a widget is highlighted blue, then its associated project tag is writable. That means the user can use the widget to set a new value for the tag.

To use the Process Values window:

1. To change a value:

a) Click/tap the widget.
The widget must be highlighted blue, to indicate that the tag is writable.
A new screen with an enlarged version of the widget is displayed.

b) Either manipulate the widget (i.e., toggle the switch, move the gauge) to set the new value, or type the new value in the text box below the widget.

c) Click/tap **Write**.

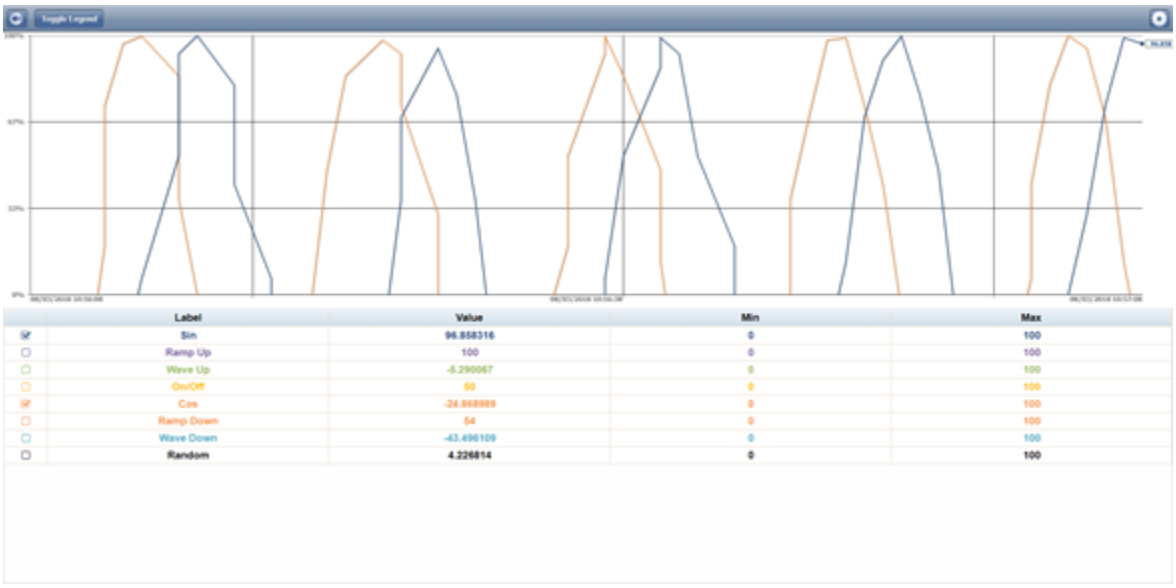
The new value is written to the tags database.

2. To return to the home screen, click/tap the **Return** button .

## USE THE TREND WINDOW

Use the Trend window to view trend graphs of selected process values.

The Trend window is similar to the Trend Control screen object. It graphs the changes in process values during project run time, and it can also display trend history when available.




*An example of the Trend window*

The X-axis of the graph is time, and the Y-axis is the value of the tags. The legend below the trend graph includes the following columns:

**Check Box**

Select to show the process value on the trend graph, or clear to hide it.

**Label**

The name of the process value. Please note that this may be different from the original name of the project tag, depending on how you configure it in the Mobile Access Configuration worksheet.

**Value**

The current process value.

**Min and Max**

The **Min** and **Max** settings from the area settings. These are used to calculate the percentage on the Y-axis. For example, if **Min** is `10`, **Max** is `20`, and the actual value is 16, then the percentage is 60%.

By default, the Trend window runs in real-time (or "play") mode with a duration of 60 seconds. That means the graph is continuously updated with the current process values, and only the last 60 seconds are actually shown on the graph. However, you can change all of this in the *Settings* window. For more information, see below.

To use the Trend window:

1. To hide or show the legend below the trend graph, click/tap **Toggle Legend**.
2. To hide or show a specific process value on the trend graph, click/tap the check box in the first column.
3. To add translucent fills below the trend lines:

a) Click/tap the **Settings** button.
The *Settings* screen is displayed.

b) Toggle the **Fill** switch to **ON**.

c) Click/tap **Trend** to apply your changes and return to the Trend window.

4. To change the duration (i.e., the X-axis) of the trend graph:

a) Click/tap the **Settings** button.
The *Settings* screen is displayed.

b) In the **Duration** text box, type the new duration in seconds.

c) Click/tap **Trend** to apply your changes and return to the Trend window.

5. To show the trend lines as actual values rather than as percentages:

a) Click/tap the **Settings** button.
The *Settings* screen is displayed.

b) Toggle the **Percentage Mode** switch to **OFF**.

c) In the **Min** and **Max** boxes, type the minimum and maximum values for the Y-axis of the trend graph.
These apply to all of the process values, overriding the values' individual **Min** and **Max** settings that are used to calculate the percentages.

d) Click/tap **Trend** to apply your changes and return to the Trend window.

6. To pause the Trend window and switch to history mode:

a) Click/tap the **Settings** button.
The *Settings* screen is displayed.

b) Toggle the **Play** switch to **OFF**.
The **Duration** setting changes to **Period**.

c) Click/tap **Period**.
The *Period* screen is displayed.

d) Use the date and time controls to set the **From** (start) and **To** (end) points of the graph's X-axis.

e) Click/tap **Settings** to apply your changes and return to the *Settings* screen.

f) Click/tap **Trend** to apply your changes and return to the Trend window.

g) Click/tap **Toggle Cursor** to turn on a vertical cursor that you can slide left and right on the trend graph, in order to see the process values at a specific time.

> **Note:** The **Toggle Cursor** button is hidden when the Trend window is not in history mode.

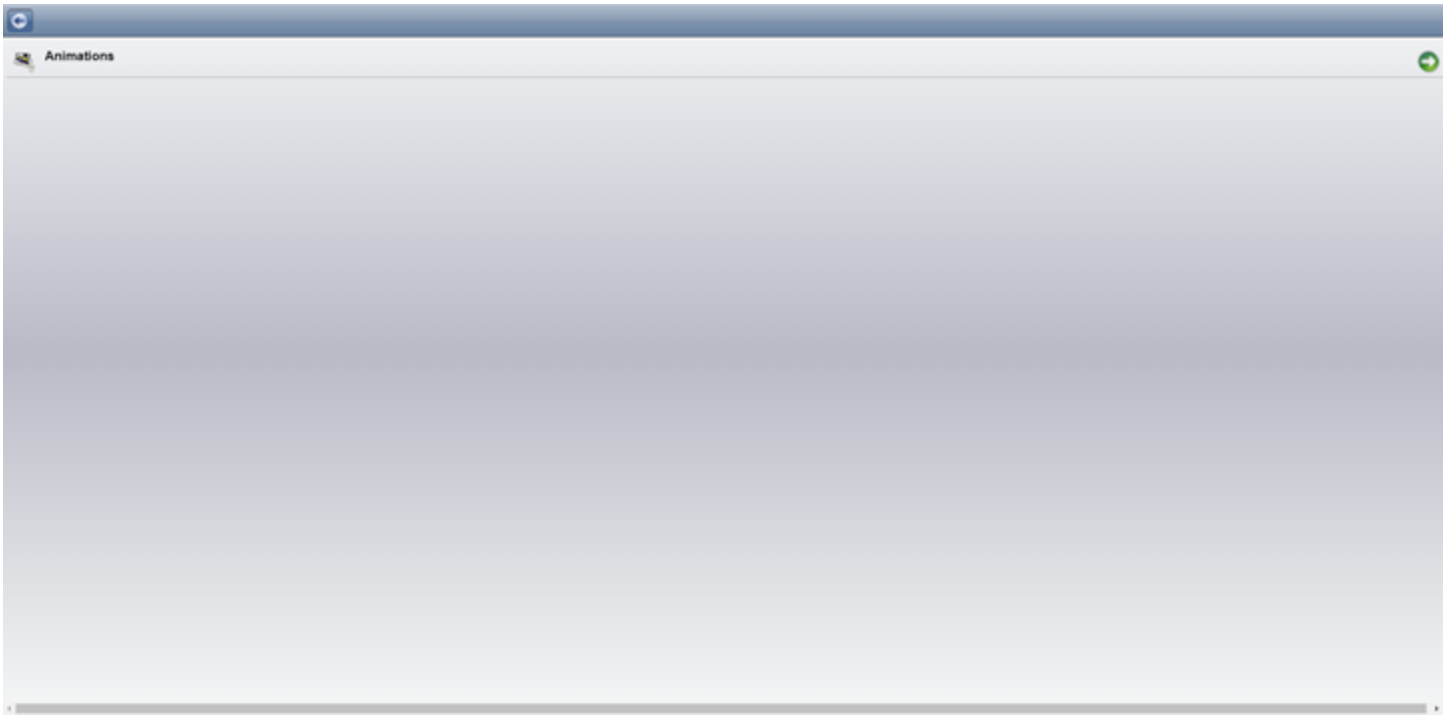
In order to display trend history, your project must include a properly configured Trend worksheet that saves the historical data for the selected project tags. The Trend window itself cannot save historical data. For more information, see Trend worksheet on page 413.

7. To return to the home screen, click/tap the **Return** button.

### USE THE SCREENS WINDOW

Use the Screens window to view selected project screens.

The Screens window shows a list of project screens that you have selected to include in Mobile Access.




*An example of the Screens window*

*An example of a project screen*

All screen objects and animations are continuously updated to show the current values of their associated project tags. Also, the **Disable** and **Security** settings are enforced on all screen objects and animations, but the **E-sign** setting is not.

To use the Screens window:

1. To go to a particular project screen, click/tap the screen name in the list.
2. To return to the list of project screens, click/tap the **Screens** button in the top-left corner.
3. To return to the home screen, click/tap the **Return** button .

## LINK DIRECTLY TO A PROJECT SCREEN OR SCREEN GROUP

Use a custom URL to bypass the Mobile Access web interface and link directly to a specific project screen or screen group.

In most cases, you would use the Mobile Access web interface to log onto your project runtime server and then navigate through your project. If you often visit a specific project screen or screen group, however, you might want to bookmark it and skip the rest of the navigation. To do that, you can compose a custom URL that includes the server address, the specific version of the Mobile Access Runtime software, the name of the project screen or screen group, and if you wish, your user credentials.

When you compose and use this custom URL, your primary concern should be project security. If the project security system has been enabled, you always need to log onto the project runtime server before you can view any project screens. Therefore, you must decide whether to include your user credentials in the URL. If you do, you will be automatically logged onto the server and then taken to the project screen. If you do not, you will be prompted to log on before proceeding.

The key here is that if you save the custom URL with your user credentials, anyone who has an opportunity to examine the URL — for example, by copying your bookmark or by watching over your shoulder while you use it — might learn your credentials. You must consider the possibilites and weigh your convenience against project security.

In some cases, the project security system is configured to log users off after a specified period. (For more information about the Auto LogOff settings, see Group Account dialog on page 664.) When it does, the browser will be redirected to the Mobile Access Logon page and the custom information will be stripped from the URL in the browser's address bar. This is to prevent other people from using an unattended computer to reload the URL and view the project screen.

> **Note:** URLs might be case-sensitive, depending on the host's operating system and/or web server.

There are four ways to compose the custom URL:

**Screen only**

This URL will take you to the Mobile Access Logon page, where you will be prompted for your user name and password. After you log on, you will be redirected to the specified project screen or screen group.

```
https://<host name or IP address>/<version>/index.html?screen=<screen name>
```

Example for default version:

```
https://scada.ourcompany.com/MA/index.html?screen=animations
```

Example for specific version:

```
https://scada.ourcompany.com/EMSE2020/index.html?screen=Animations
```

**Screen, with guest logon**

This URL will log you on as a guest user and take you directly to the specified project screen or screen group.

```
https://<host name or IP address>/<version>/index.html?screen=<screen
 name>&guestuser=1
```

Example for default version:

```
https://scada.ourcompany.com/MA/index.html?screen=animations&guestuser=1
```

Example for specific version:

```
https://scada.ourcompany.com/EMSE2020/index.html?
screen=Animations&guestuser=1
```

**Screen, with user name**

This URL will take you to the Mobile Access Logon page and automatically enter the specified user name. You will be prompted for the corresponding password. After you log on, you will be redirected to the specified project screen or screen group.

```
https://<host name or IP address>/<version>/index.html?screen=<screen
 name>&user=<user name>
```

Example for default version:

```
https://scada.ourcompany.com/MA/index.html?
screen=animations&user=Operator112
```

Example for specific version:

```
https://scada.ourcompany.com/EMSE2020/index.html?
screen=Animations&user=Operator112
```

**Screen, with user name and password**

This URL will automatically log you on as the specified user and then take you directly to the specified project screen or screen group.

```
https://<host name or IP address>/<version>/index.html?screen=<screen
 name>&user=<user name>&password=<password>
```

Example for default version:

```
https://scada.ourcompany.com/MA/index.html?
screen=animations&user=Operator112&password=eWi28fb2
```

Example for specific version:

```
https://scada.ourcompany.com/EMSE2020/index.html?
screen=Animations&user=Operator112&password=eWi28fb2
```

This is the least secure option, because the user name and password are both included in the URL.

### Notes

This procedure only works for the project screens and screen groups that have been saved as HTML. It does not work for any other screens, nor does it work for the Alarm, Process Values, or Trend controls in the web interface.

If you are linking to a screen group instead of a project screen, you must include the screen group file extension in the custom URL (e.g., `index.html?screen=group.sg`).

Mobile Access supports character encoding — that is, substituting character codes for special, non-ASCII characters — in the custom URL. Character encoding is most often used to allow for spaces in screen and user names; you can substitute the character code **`%20`** for each space in the custom URL (e.g., `index.html?screen=main%20screen`). It can be used for any special characters, however, as long as you know the appropriate character codes. For more information, go to: http://www.w3schools.com/tags/ref_urlencode.asp

Some web browsers automatically encode whatever you type in the address bar, which means you can type the custom URL with spaces (or other special characters) as you normally would. Not all web browsers do this, however, so you should test your custom URL in the web browser(s) that you use and be prepared to manually encode, if necessary.

## *Troubleshooting project screens in Mobile Access*

Use the activity log to troubleshoot project screens in the Mobile Access web interface.

Mobile Access currently supports many but not all features of EcoStruxure Machine SCADA Expert 2020. If you use an unsupported feature in a project screen, you might see unexpected behavior when you view that screen in the Mobile Access web interface. Such behavior can range from incorrect tag changes and function calls to objects, animations, or scripts that do not work at all.

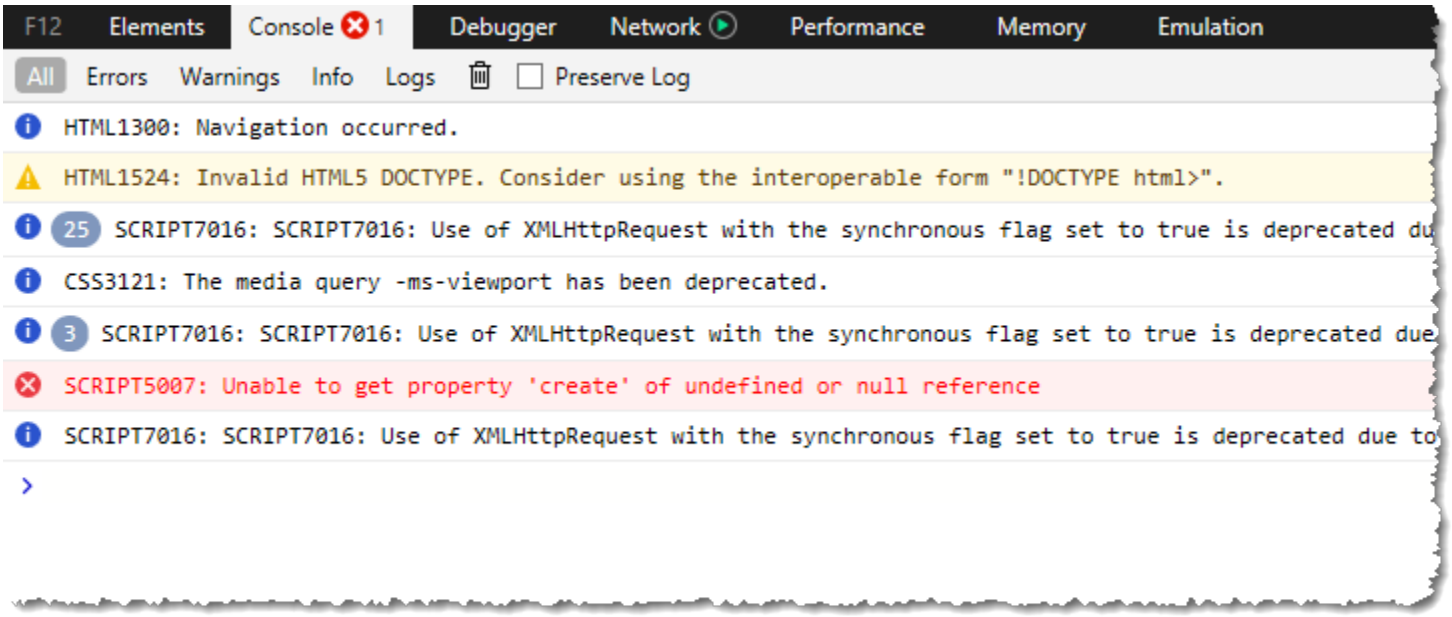
Mobile Access automatically generates an activity log for the web interface, and you can use the browser console to get the log for the project screen that you are currently viewing. Then, using the information provided by the log, you can identify and resolve most issues that you might have.

There are also some common issues that can be resolved without using the activity log. Those issues are addressed at the end of this section.

## USE THE ACTIVITY LOG TO TROUBLESHOOT THE MOBILE ACCESS WEB INTERFACE

Use the Mobile Access activity log, which can be viewed either in the browser console or as part the full project runtime log, to troubleshoot the Mobile Access web interface.

This activity log is similar to the project runtime log that is displayed in the *Output* window and/or LogWin module, except that it comprises only messages about Mobile Access itself and the performance of the web interface, rather than about the entire project runtime.




*An example of the activity log in the console*

### Where to view the activity log

When Mobile Access is configured and your project is running, you can view the Mobile Access activity log in three different places.

First, you can view it in the console of the web browser that you use to access the Mobile Access web interface. This is the most limited view of the activity log, because it displays only the log messages for the project screen(s) that you are currently viewing in that web browser. It does not display the log messages for any other client sessions, neither on other computers nor in other client applications (e.g., Secure Viewer) on the same computer. However, it is the most immediately available when you are actually using the web interface, and it is somewhat easier to read because it is separated from the full project runtime log. For more information, see Use the browser console to view the Mobile Access activity log on page 807.

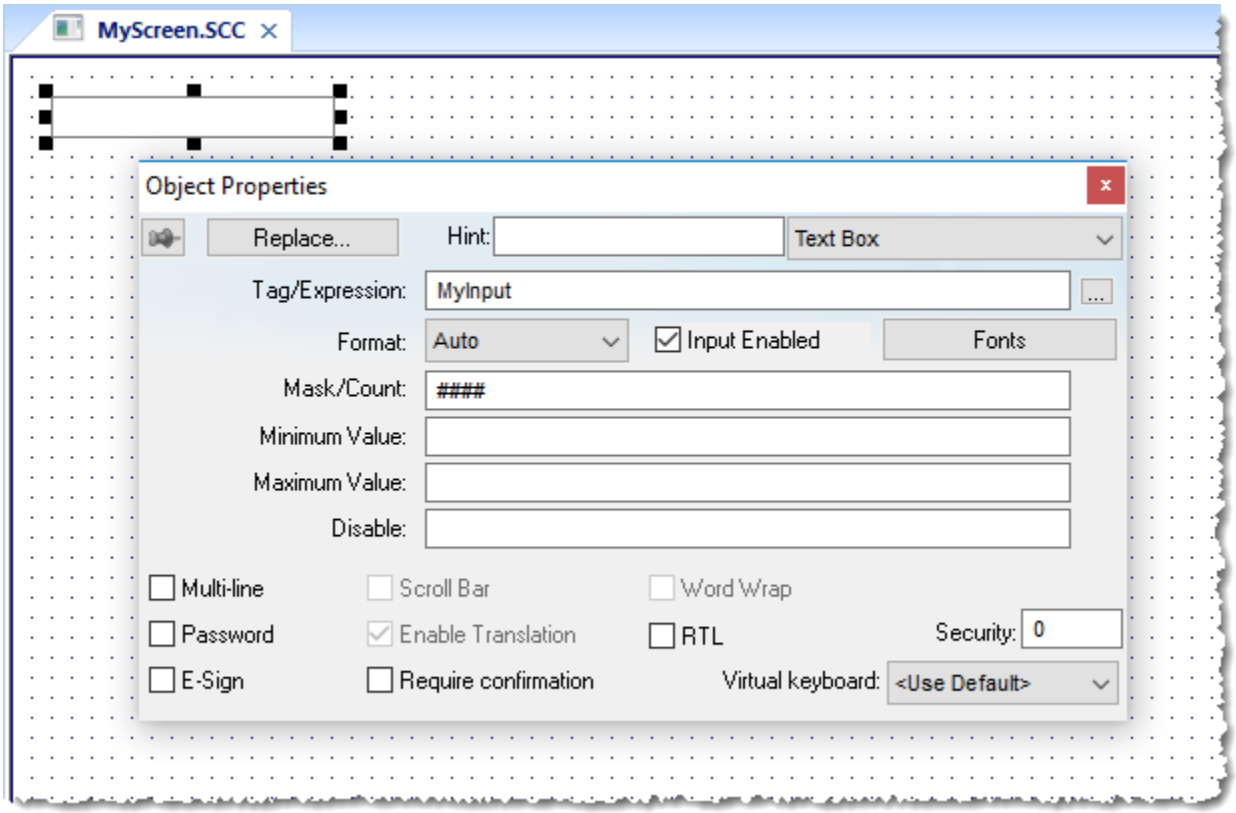
Second, you can view the Mobile Access activity log as part of the full project runtime log, in the *Output* window in the project development environment, at least for the project runtime on the local computer. (The full EcoStruxure Machine SCADA Expert 2020 software can function as both project development environment and project runtime, depending on how it is licensed.) For more information, see Output window on page 72.

Third, you can view Mobile Access activity log as part of the full project runtime log, in the LogWin module, for a project runtime on either the local computer or a remote computer. For more information, see About the LogWin tool on page 712.

### Use the activity log to troubleshoot an issue

Here is an example of how to use the activity log to troubleshoot an issue in one of your project screens.

The screen MyScreen contains a Text Box object that has been configured with the tag MyInput:

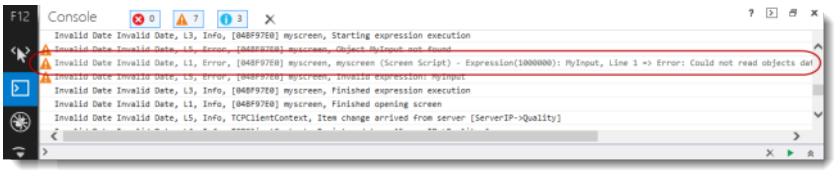



When you normally view the screen in the Mobile Access web interface, the text box shows a value of 0 as it waits for user input:

If the tag does not exist, however — for example, if you accidentally deleted it from the tags database — then the text box shows its mask instead and rejects all input:

Verifying the project will catch things like missing tags, but in order to verify the project, you must be able to use the project development environment to open and edit the project. The user typically will not be able to do that during run time.

This is where you can use the activity log to troubleshoot the issue. Open the browser console, and then look for a message like this in the activity log:




```
Invalid Date Invalid Date, L1, Error, [04BF97E0] myscreen, myscreen (Screen Script) -
 Expression(1000000): MyInput, Line 1 => Error: Could not read objects database. Object
 MyInput not found [Error Code: 0x80070057]
```

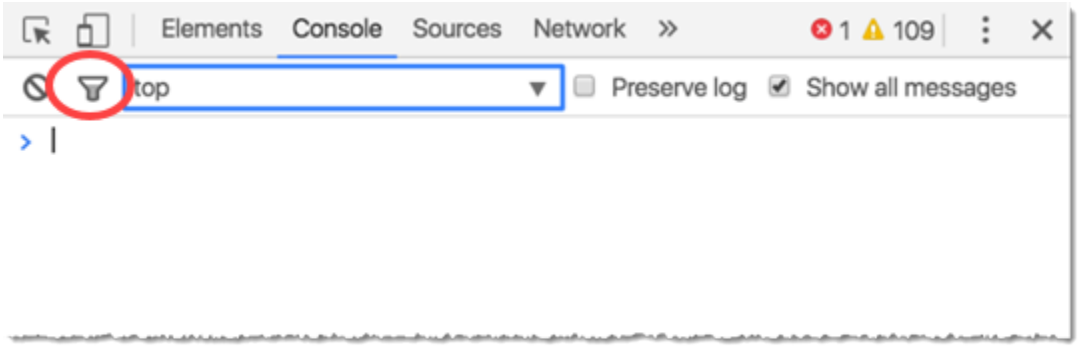
> **Tip:** The browser console cannot receive log messages unless it is open, so if the console is closed when you first notice an issue, open it and then access the screen again.

The message provides the name of the screen, the name of the affected object or tag, and the exact nature of the error. With this information, you should get some idea of how to resolve the issue in your project. This is a very simple example, of course, and an issue like this typically will be included in a Level 1 log; more complex issues might require increasing the log level and searching through more messages. But this essentially is the procedure for using the activity log to troubleshoot a project screen in Mobile Access.

### Filter the activity log in the browser console

You can filter the Mobile Access activity log in the browser console in order to reduce the number of log messages that you must look through to find the information that you need. The tools for filtering the log vary by browser, so please consult the documentation for the browser that you are using.

In Google Chrome, click the **Filter** tool on the console tool bar to reveal additional tools:




***Additional filter tools in Google Chrome***

If you filter by message type, remember that Mobile Access errors are displayed as "Warnings", Mobile Access warnings are displayed as "Info", and Mobile Access info messages are displayed as "Logs". ("Errors" are reserved for critical errors in the web browser itself.) Otherwise, you can try to filter by specific text.

### Configure the log settings to show more information

Filtering works only if you are receiving too much information, of course. If you are not receiving enough information, you might need to configure the log settings to increase the verbosity of the log and/or change the types of messages that are included in the log.

The log settings for the browser console are actually in your project's Mobile Access Configuration worksheet. For more information, see Configure the global settings for all areas on page 790. Configuring those settings will change the log messages that are sent to all of your Mobile Access clients, however, and any change in your project will require rensending it to its target device(s). As such, this option should reserved for troubleshooting during project development. If your project is already running on a target device, consider using the LogWin module instead.

The log settings for both the *Output* window and the LogWin module are in those respective tools. You can change those settings at any time without changing the project itself.

## USE THE BROWSER CONSOLE TO VIEW THE MOBILE ACCESS ACTIVITY LOG

Use the browser console in Google Chrome, Microsoft Edge, or Safari in order to view the Mobile Access activity log.

The browser console is a part of the web browser's developer tools, and it is typically used to debug code in a web page. When you are using the Mobile Access web interface, the browser console also displays log messages sent by project screens. For more information, see Use the activity log to troubleshoot the Mobile Access web interface on page 804.

You can open the browser console at any time, but it will not display any log messages unless both the project runtime server and the web server are running and you have successfully logged on to the Mobile Access web interface. For more information, see Log on to the Mobile Access web interface on page 792.

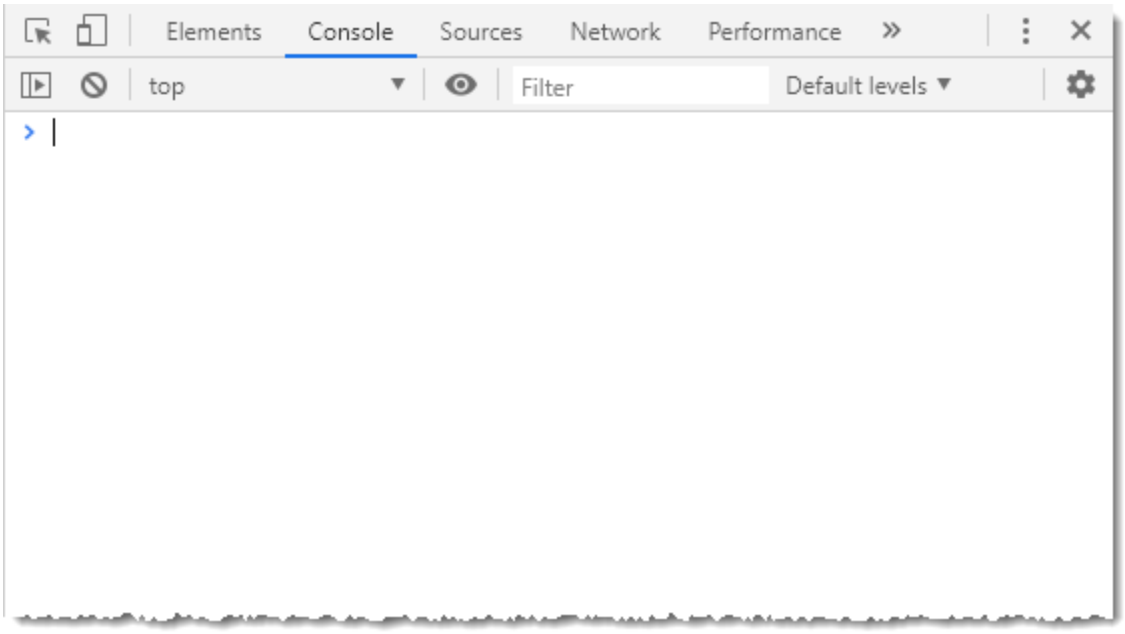
> **Note:** These instructions apply only to the desktop versions of these browsers. For more information about how to access the browser console or developer tools in a mobile browser, please consult the documentation for that browser.

### Google Chrome (Windows, macOS, Linux)

To open the browser console in Google Chrome, do one of the following:

- Go to **More** (i.e., the dots icon) on the toolbar, and then select **More Tools** > **Developer Tools**; or
- Press **Ctrl+Shift+J** (or **Option+Cmd+J** in macOS) on the keyboard.

The console is displayed at the right side of the browser window.




***Browser console in Google Chrome***

### Microsoft Edge (Windows, macOS)

To open the browser console in Microsoft Edge, do one of the following:

- Go to **Settings** (i.e., the dots icon) on the toolbar, and then select **More tools** > **Developer Tools**; or
- Press **F12** on the keyboard.

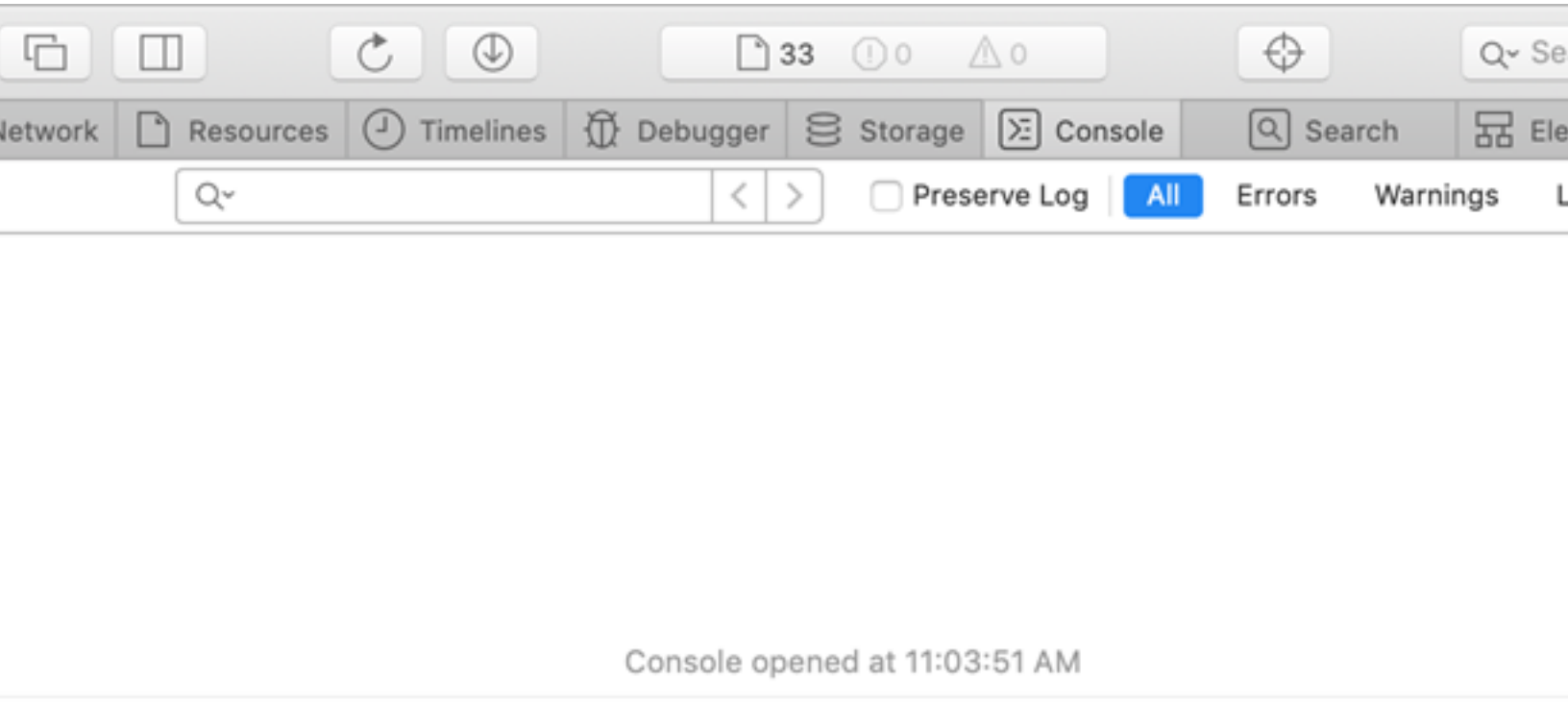
The console is displayed at the right side of the browser window.

### Safari (macOS only)

To open the browser console in Safari:

1. Go to **Safari** (i.e., the application menu), and then select **Preferences**.
2. In the *Preferences* dialog box, go to the **Advanced** tab, and then make sure the **Show Develop menu in menu bar** option is selected.
3. Close the *Preferences* dialog box.
4. Do one of the following:
   - Go to **Develop**, and then select **Show Javascript Console**; or
   - Press **Option+Cmd+C** on the keyboard.

The console is displayed at the bottom of the browser window.




***Browser console in Safari***

## TYPES OF MOBILE ACCESS LOG MESSAGES

The Mobile Access activity log can include several different types of log messages. You can configure the log settings to change which types of messages are displayed, depending on what information you want to see during project run time.

### Where to configure the log settings

Each tool that you can use to view the log has its own log settings.

The log settings for the browser console are actually in your project's Mobile Access Configuration worksheet. For more information, see Configure the global settings for all areas on page 790. Configuring those settings will change the log messages that are sent to all of your Mobile Access clients, however, and any change in your project will require rensending it to its target device(s). As such, this option should reserved for troubleshooting during project development. If your project is already running on a target device, consider using the LogWin module instead to view that device's log.

The log settings for both the Output window and the LogWin tool are in those respective tools. You can change those settings at any time without changing the project itself.

### Types of log messages

The first three types of log messages are according to severity:

**Error**

These are messages about serious errors encountered during project run time, like unsupported features used in a project screen and run-time errors in the Mobile Access task. (Errors are marked by the yellow "Warning" icon in most browser consoles.)

**Warning**

These are messages about issues that can affect run-time performance, such as excessive tag synchronization and function calls that must be executed on the server. (Warnings are marked by the blue "Message" or "Info" icon in most browser consoles.)

**Information (Info)**

These are messages that report individual function calls, tag value changes, screen openings and closings, and so on. (Info messages are not marked by icons in browser consoles.)

> **Note:** It is not a mistake that we use the "Warning" icon to mark Mobile Access errors and the "Message"/"Info" icon to mark Mobile Access warnings. The browser console's red "Error" icon is reserved for critical errors in the browser itself.

The second three types of log messages are according to which run-time component generated the message:

**Runtime Comm**

These are messages about communication between the project runtime server and the Mobile Access task.

**Screen**

These are messages about the project screens — including objects and scripts — that are currently being viewed by Mobile Access users.

When you view the activity log in either the *Output* window or the LogWin module, you can see messages about all project screens being viewed by all users. When a user views the activity log in their browser console, they can only see messages about the project screens they are viewing in their own browser.

**Web Services**

These are messages about communication between the Mobile Access task and the client sessions, through whatever web server you have set up to serve the Mobile Access web interface.

The first three types work in combination with the second three types to determine exactly which messages are included in the activity log. If you want to see everything, simply select all of the options in the log settings. If you only want to see messages about serious errors in the web services, for example, select only **Error** and **Web Services**. You might need to experiment to get the exact information that you need.

Finally, select the **Trace** option to display the messages that are generated whenever the `Trace` function is called in a project screen.

### Increase the verbosity to get more information

By default, the activity log is set to verbosity level 1, which provides the least amount of information. If you have issues while using the Mobile Access web interface — especially while trying to view selected project screens — and the log does not help, you can increase the verbosity to get more information.

Be careful not to set the verbosity too high, however, because the higher it is, the longer and more detailed the log will be, which might make it hard for you to find the information that will actually help you to resolve your issues. Try increasing the verbosity in steps, first from level 1 to level 2, then from level 2 to level 3, and so on until you get the information you need.

Here is a basic guide to which levels you should try:

- If objects, animations, or scripts are not working at all, try level 1. This level should be enough to identify missing tags, unsupported features, and other such things. For more information, see Supported features in Mobile Access on page 756.
- If a project screen is unusually slow to open or update, try level 3 (which includes levels 1–2). This level should help you to diagnose issues that affect run-time performance. For more information, see Tips for Mobile Access development and run time on page 766.
- If you simply do not see the behavior that you expect during run time, try level 5 (which includes levels 1–4). This level reports every tag update and function call.

Please note that these are only general suggestions; you will probably need to change the verbosity more than once to get the right amount of detail on your specific issues.

## USE THE PROBEHEALTH SERVICE TO TEST MOBILE ACCESS

Mobile Access includes a ProbeHealth service that you can use to test the status and availability of the server.

Each instance of Mobile Access includes its own ProbeHealth service that returns HTTP status codes when queried.

To access the ProbeHealth service for a given instance, configure the load balancer to go to the following URL:

**https://*<host name or IP address>*/*<version>*/ProbeHealth**

Example for the default version of Mobile Access:

**https://scadaserver01/MA/ProbeHealth**

Example for a specific version of Mobile Access:

**https://scadaserver01/EMSE2020/ProbeHealth**

For more information about linking directly to specific versions of Mobile Access, see Link directly to a project screen or screen group on page 801. However, the ProbeHealth service is a recent addition to Mobile Access and is not included in some earlier versions, so make sure you test the configuration of the server before you allow it to accept connections from users.

The ProbeHealth service returns one of the following HTTP status codes when queried:

| HTTP Status Code | Description |
|---|---|
| 200 OK | This instance of Mobile Access is running as expected. |
| 500 Internal Server Error | This instance of Mobile Access is not running as expected. |

Additional information is provided as a JSON-formatted message, which can be used for logging or troubleshooting purposes. That message should include one of the following result codes:

| Result Code | Description |
|---|---|
| 0 | Ok |
| 61 | LicenseLimit |
| 201 | InitializationFailed |
| 205 | ErrorConnect |
| 305 | ProbeHealthNotOk |

Examples of messages:

```
{"id":0,"resultCode":0,"message":"","data":{"probeHealth":"Ok!"}}
```

```
{"id":0,"resultCode":305,"message":"WebProxy: Error to create connection, please make
 sure the runtime is running and that the Mobile Access Task is started (see Project-
>Tasks).  [status: -13, message: Fail to establish connection, data: Error to establish
 connection [OS Error Code: 10061]]","data":{}}
```

For more information about the ProbeHealth service, please contact your EcoStruxure Machine SCADA Expert 2020 software distributor.

### *View or disconnect client sessions*

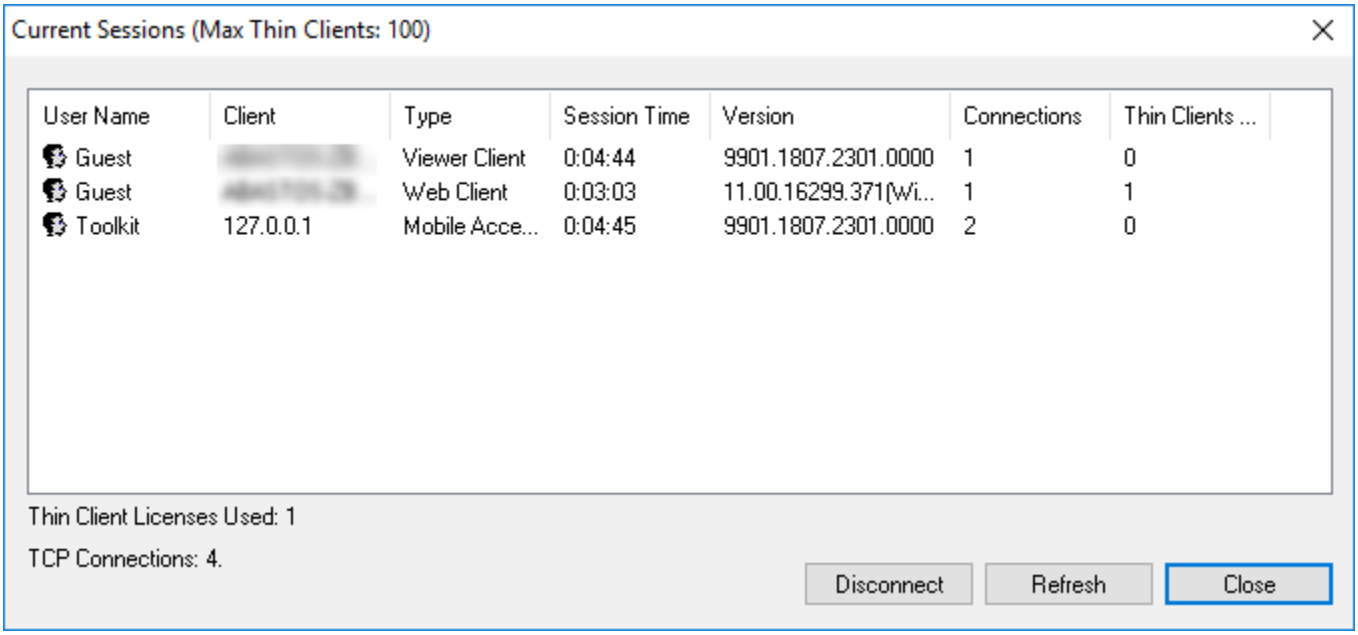
Use the *Current Sessions* dialog box to view or disconnect clients that are currently connected to your project runtime server.

Before you begin, your project must be running on Windows using the full EcoStruxure Machine SCADA Expert 2020 software with an appropriate runtime license. The Current Sessions feature is not available in our other runtime editions. Also, you must have access to the computer that hosts the project runtime, either directly or through screen sharing. You cannot access the Current Sessions feature through Remote Management.

When a client connects to your project runtime server, a client session is initiated. Each session counts against the maximum number of clients allowed by your runtime license. A session ends only when the user either logs off from the project or closes the client program, so if the current number of client sessions approaches the maximum number of clients allowed, you might need to disconnect old or idle sessions in order to ensure that your project runtime server remains accessible.

> **Note:** For Mobile Access only: due to technical differences between web browsers, the exact moment when the thin client is considered "closed" — and therefore the session ends — varies somewhat. In Chrome for Android, the session ends when the user goes to a new website but not when the user closes the browser tab. In Safari for iOS, it is the opposite: the session ends when the user closes the browser tab but not when the user goes to a new website.

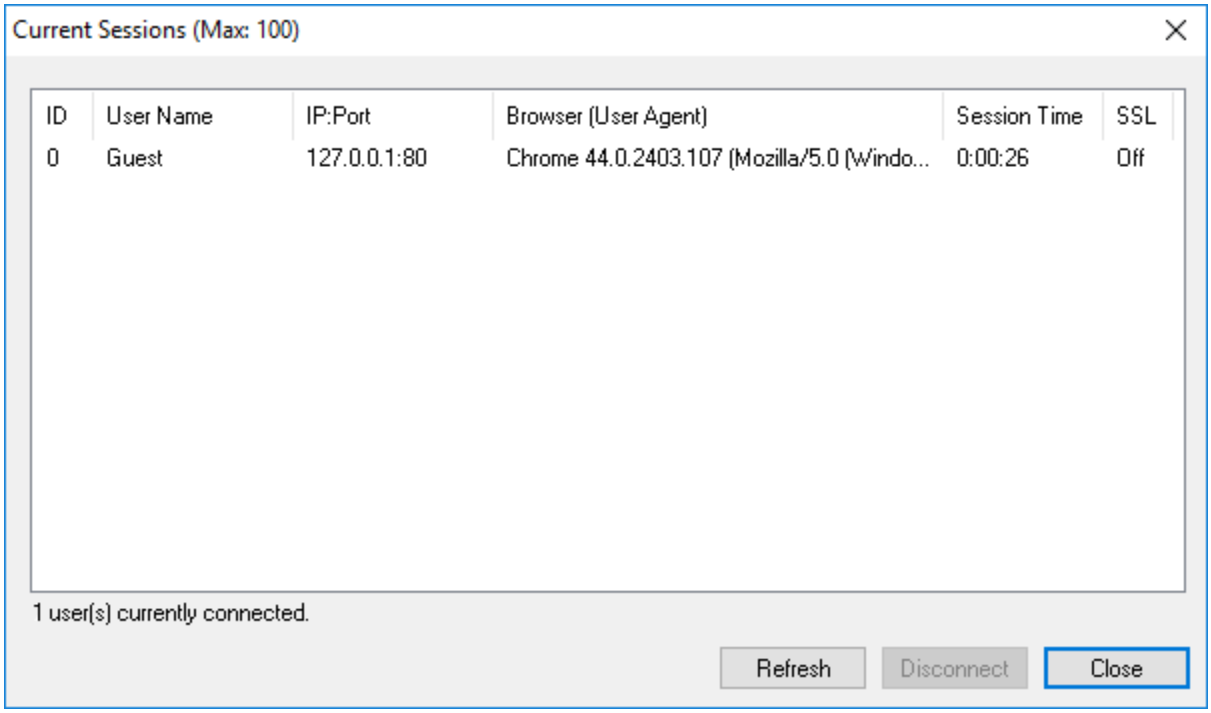
Thin Client sessions are handled by the TCP/IP Server runtime task in your project. When the project is running and the task is started, a **TCP/IP Server** icon is displayed in the notification area of the Windows taskbar. You can use this icon to open the *Current Sessions* dialog box for the Thin Client sessions.




***Example of the Current Sessions dialog box for TCP/IP Server***

Mobile Access sessions are handled by the Mobile Access runtime task, and similar to the above, when the project is running and the task is started, a **Mobile Access Task** icon is displayed in the notification area of

the Windows taskbar. You can use this icon to open the *Current Sessions* dialog box for the Mobile Access sessions.




***Example of the Current Sessions dialog box for Mobile Access***

For more information about the TCP/IP Server and Mobile Access runtime tasks, see Runtime Tasks on page 136.

Also, please note how the title bar shows the maximum number of clients allowed. That information is gotten from your runtime license settings. For more information, see License Settings on page 45.

Finally, the **User Name** column will show individual user names only if you enabled the security system in your project. If you did not, all users will be logged on and shown as "Guest". For more information, see Project Security on page 639.

> **Note:** The Mobile Access runtime task has a memory limit of 1.5 GB. If this limit reached during project run time — typically due to trying to run an extremely large project, but also sometimes due to managing a large number of client sessions — additional clients will not be allowed, regardless of the runtime license settings. When this happens, an alert message will be displayed to users who try to log on.

To view or disconnect thin client sessions:

1. In the notification area of the Windows taskbar, right-click either the **TCP/IP Server** icon or the **Mobile Access Task** icon, and then on the shortcut menu, click **Current Sessions**.
   You might need to expand the notification area to show hidden icons.
   The *Current Sessions* dialog box is displayed.
2. To refresh the list of sessions, click **Refresh**.
   In most cases, the list will automatically refresh itself as thin clients connect and disconnect, but you can also manually refresh it make sure you have the latest information.
3. To disconnect a specific session, select that session in the list, and then click **Disconnect**.
   The selected session is disconnected and the session's user is logged off. A new session is automatically initiated, as if the user restarted or reloaded the thin client, but it will expire after a specfied period if no

one logs on. For more information about session expiration, see Configure the global settings for all areas on page 790.

4. When you are done, either close the window or click **Close**.

# Database Interface

Configuring a database interface with EcoStruxure Machine SCADA Expert is basically linking tasks from EcoStruxure Machine SCADA Expert (Alarms, Events or Trends) to tables of external databases via a specific Database Provider that supports the database you have chosen.

Each history task (Alarm, Events or Trend) can be configured to save data either to files with the proprietary format from EcoStruxure Machine SCADA Expert or to external SQL Relational Databases. Use the Options tab to configure the database to save Alarm and Event history. (See the Trend Folder for instructions for saving history for the trend tasks.)

EcoStruxure Machine SCADA Expert supports ADO.NET to provide an intuitive, simple, flexible and powerful interface with standard technologies from MDAC (Microsoft Data Access Components) such as OLE-DB (Object Linking Embedded — Database) and ODBC (Open Database Connectivity). By using this capability, you can connect to any database that is MDAC compatible (please see the Conformance Table for the list of databases already tested)

The following tasks support the database interface:

- **Alarms**: The project can save and/or retrieve the alarm history messages in a relational database.
- **Events**: The project can save and/or retrieve the event messages in a relational database.
- **Trends**: The project can save and/or retrieve the Trend history values in a relational database.
- **Viewer**: Database information can be displayed both in table format (Alarm/Event Control and Grid objects) or in a graphical format (Trend Control object).
- **Web**: Because the items listed below are already available in EcoStruxure Machine SCADA Expert Web interface, you can deploy a project that stores/saves data in a relational database and have it working over the Web.

Using its embedded database interface, EcoStruxure Machine SCADA Expert can easily provide data from the plant floor to third-party systems (e.g., ERP) or get data from them.

EcoStruxure Machine SCADA Expert can interface with any relational database supported by a valid ADO.NET Provider, OLE DB provider or ODBC driver. However, the conformance tests were executed with the following databases:

### Conformance Test Table

| Database | Database Version | ADO.NET Provider | Assembly Version |
|---|---|---|---|
| Microsoft SQL Server 2000 | 8.0 | System.Data.SqlClient | 1.0.5000.0 |
| Microsoft Access 2000 | 9.0.3821 SR-1 | System.Data.OleDbClient | 1.0.5000.0 |
| Microsoft Excel 2000 | 9.0.3821 SR-1 | System.Data.OleDbClient | 1.0.5000.0 |
| Oracle | 10g Release 1 for Windows | System.Data.OracleClient | 1.0.5000.0 |
| Sybase | Anywhere 9.0.1.1751 | iAnywhere.Data.AsaClient | 9.0.1.1751 |
| MySQL | 4.0.20a | ByteFX.MySqlClient | 0.7.6.15073 |

**Note:**

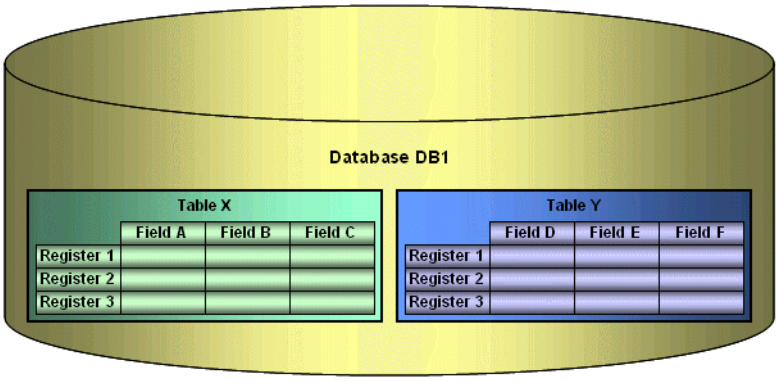
For information about how to configure a specific database, please refer to the following:


- Using ODBC Databases on page 848
- Using Microsoft SQL Server on page 849
- Using Oracle Databases on page 852
- Using Microsoft Access or Microsoft Excel on page 853
- Using Sybase on page 855
- Using MySQL on page 855

## SQL Relational Databases

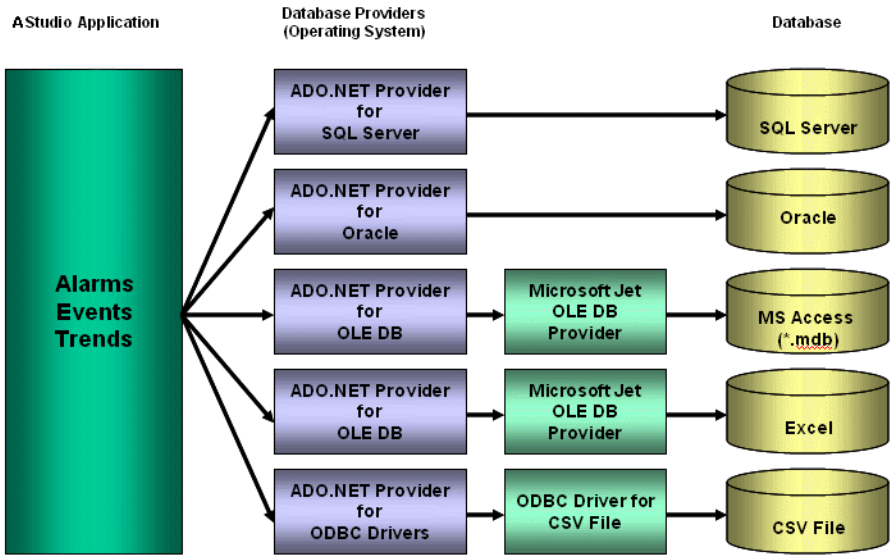
A SQL Relational Database is a set of information stored in tables with fields and registers, which support SQL commands.

Each database can have one or more tables. Each table is composed of fields (columns) and registers (rows). Typically, the fields are pre-defined and the project adds or reads one or more registers, according to the query condition.




EcoStruxure Machine SCADA Expert uses Database Providers (ADO.NET) to interface with SQL Relational Databases. Database Providers are libraries developed to access data from different databases through SQL commands. The ADO.NET Provider for a specific database can be supplied by the operating system or by the database manufacturer.

The following picture illustrates how EcoStruxure Machine SCADA Expert can interface with different databases using a different Database Provider for each database.




The previous picture shows some of the most popular ADO.NET Providers for databases. Notice that the *Microsoft ADO.NET Provider for ODBC Drivers* allows you to access the database through an ODBC driver. See Using ODBC Databases on page 848 for information about how to use this provider. It is also possible that you do not have an ADO.NET provider, but an OLE DB provider is available. By using the *Microsoft ADO.NET Provider for OLE DB* you can get access to the database; the Microsoft Jet OLE DB provider gives access to applications in the Microsoft Office package by using this approach.

> **Note:** It is important to note that EcoStruxure Machine SCADA Expert provides the interface for ADO.NET Providers. However, the ADO.NET Providers and/or the ODBC Driver/OLE DB

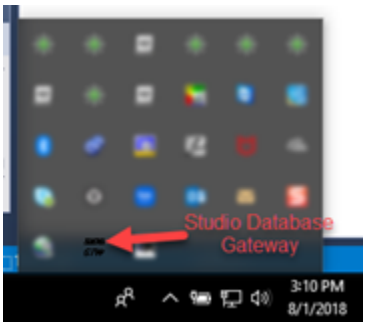
Provider must be supplied either by the operating system or by the database manufacturer. If your Connection String does not refer to a valid ADO.NET Provider, the OLE.DB Provider will be used.

Although most projects typically link to only one type of database, EcoStruxure Machine SCADA Expert gives you the flexibility to link each task to a specific database supported by a Database Provider. Furthermore, by using this architecture, you do not need to worry about the specific characteristics of each database (it is mostly handled by the Database Provider for each database or by the EcoStruxure Machine SCADA Expert Database Gateway interface). Therefore, the project settings are mostly uniform, regardless of the specific database chosen by you.

## Studio Database Gateway

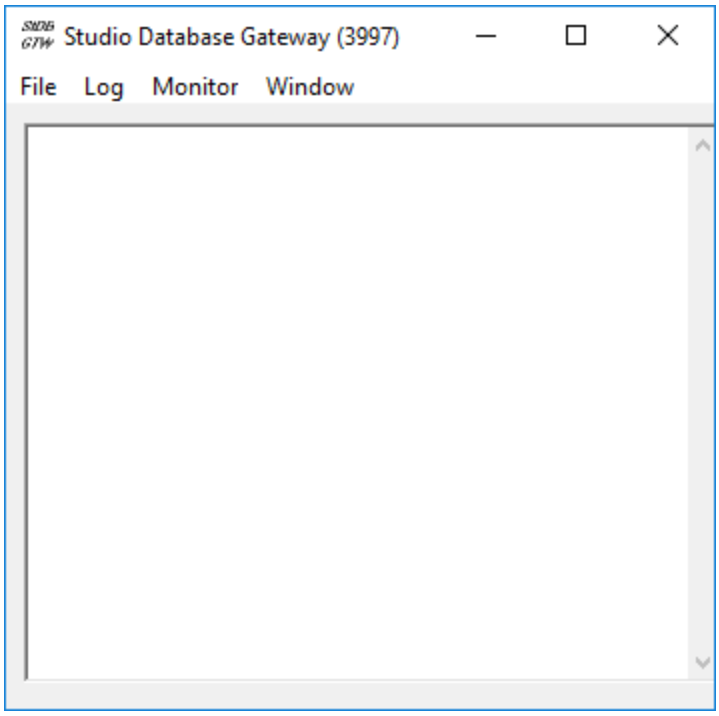
The Studio Database Gateway is a TCP/IP server that interacts with databases using the Microsoft .Net Framework. It can run on the same computer that is running the EcoStruxure Machine SCADA Expert project, or on a different computer. The Database Gateway Host in the Advanced Settings (see Database Configuration on page 108) specifies whether the gateway will be running on the local computer or not. If you are using the local computer you should enter either **`localhost`** or **`127.0.0.1`** in the Host name. You do not need to worry about starting or stopping the gateway because it will be done automatically by EcoStruxure Machine SCADA Expert tasks. On the other hand, when running the gateway remotely, you need to start the gateway manually. To do so, copy the files **`StADOSvr.exe`** and **`StADOSvr.ini`** from the `\BIN` folder to the remote computer, and then execute the **`StADOSvr.exe`**.

The gateway can be started multiple times for different TCP/IP port numbers. The default port number is 3997, and it is changed by specifying the desired port number in the command prompt (e.g., **`StADOSvr 1111`** ). When running the StADOSvr, it will add the following icon to the system tray:




When you right-click on the icon, the following shortcut menu is displayed:

The **Hide** option controls whether the *Studio Database Gateway* window is displayed on the desktop. (The gateway software runs continuously after you launch it, regardless of whether the window is displayed.) If you disable the **Hide** option, the window is displayed:

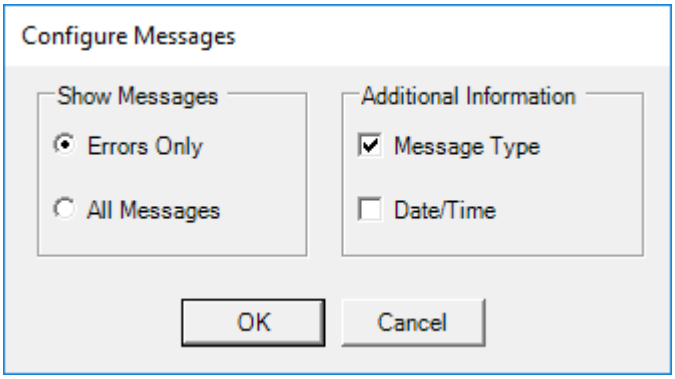



***Studio Database Gateway***

Any failure that occurs during operations with databases will be displayed both in this window and also in the *LogWin* window. The messages are reported by exceptions generated by the ADO.NET Provider. (Please refer to Database Troubleshooting for more information about error messages in the gateway module.)

You can configure the output in this window by using the **Log** menu:

- **Show Log** menu option: Shows the EcoStruxure Machine SCADA Expert Database Gateway log files.
- **Options** menu option: Open the *Configure Messages* dialog.



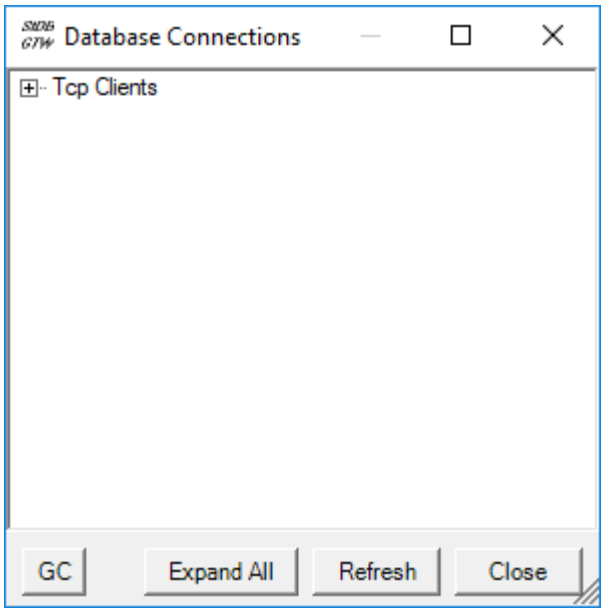

***Studio Database Gateway: Configure Messages dialog***

- *Show Messages* pane: Select **Errors Only** to show only error messages in the log, or select **All Messages** to show all database messages.
- *Additional Information* pane: Configure to show additional information about each database message.

- **Message Type** checkbox: Click (check) this option to show the type of the message.
- **Date/Time** checkbox: Click (check) this option to show the timestamp of the message.

Also, you can directly monitor database connections using the **Monitor** menu:

- **Connections** menu option: Displays the *Database Connections* window.




***Studio Database Gateway: Database Connections window***

- **Log to File** menu option: Logs the monitor output to a file named `log`*`date`*`.txt` in the same folder as `StADOSvr.exe`.

### Data Protection

If you have enabled Data Protection in your project, you also need to enable Data Protection in the Studio Database Gateway so that it and the project runtime server can communicate securely with each other.

> **Note:** If more than one project runtime server will be using the same Studio Database Gateway to manage database connections, all of the projects should have the same Data Protection password.

To enable Data Protection in the Studio Database Gateway:

1. In the *Studio Database Gateway* window, on the **File** menu, click **Data Protection**. The *Data Protection* dialog box is displayed.
2. Select the **Enable** check box. The **Password** and **Confirm Password** boxes become active.
3. In the **Password** box, type your password, and then in the **Confirm Password** box, type it again. Be sure to type the same password that you used in your project.
4. Click **OK**.

For more information about Data Protection, see Enable Data Protection to encrypt sensitive information on page 113.

### Advanced Settings

The Studio Database Gateway has Advanced Settings that are configured in the **`StADOSvr.ini`** file. If you are having problems interfacing with a specific database, you will probably need to change some of these settings or add new providers to the file. The following settings are available:

| Section of .INI File | Setting | Accepted Values | Description |
|---|---|---|---|
| **Providers** | **SaveMSec** | 1 : Disable<br>2 : Enable<br>3 : Separate Column | This setting specifies the default behavior for the provider when saving milliseconds. The default can be changed on the Advanced Settings in the Database Configuration Dialogs. |
| | **Assembly** | Any string that contains a .NET Framework assembly | Assembly option for all providers. The assembly has all the classes required to interface with the database. Most of the providers are inside the System.Data assembly. |
| | **ConnectionClass** | Any connection class inside the assembly | The Connection Class is the one that implements the System.Data.IDbConnection interface. |
| | **DateAdapterClass** | Any data adapter class inside the assembly | The Data Adapter class is used on operations where updates to the database are necessary. It must be compatible with the connection class specified and it should implement IDbDataAdapter. |
| | **CommandBuilderClass** | Any command builder class inside the assembly | The Command Builder class is also responsible for updates on databases. It must be compatible with the connection class. |
| | **Provider** | Name of the provider | One of the parameters in the connection string is the "Provider". The Studio ADO Gateway compares the value on the connection string with the value for this parameter in each provider and defines the proper one to be used. |
| | **ColumnDelimiterPrefix** | Any character or group of characters | Specify a character that will be placed before column names on SQL statements |
| | **ColumnDelimiterSuffix** | Any character or group of characters | Specify a character that will be placed after column names on SQL statements |
| | **TableDelimiterPrefix** | Any character or group of characters | Specify a character that will be placed before table names on SQL statements |
| | **TableDelimiterSuffix** | Any character or group of characters | Specify a character that will be placed after table names on SQL statements |
| | **ValueString** | Any string | This value indicates how constant values are identified on SQL statements. For Microsoft SQL databases for instance, the value should be @Value, for ODBC question mark (?) |
| | **ValueStringPrefix** | Any string | This value indicates a prefix to be used before the values. Oracle values, for instance, require the prefix. The SQL statements use value identifiers by using their prefixes, but the parameters in the Connection class do not use the prefix. |
| | **ValueAddNumber** | 0 or 1 | Indicates whether a sequential number should be added to the ValueString to identify the parameter or not. For Microsoft SQL database, this setting should have the value 1, because parameters are identified by using @Value1, @Value2, …, @ValueN. For ODBC, this setting should be 0. |
| | **BoolType** | Any string representing a valid data type for the database | When trying to create columns to store boolean values, the data type specified on this setting will be used. You need to make sure that the data type specified is able to save boolean values. |
| | **IntegerType** | Any string representing a valid data type for the database | When trying to create columns to store integer values, the data type specified on this setting will be used. You need to make sure that the data type specified here is able to store 32 bit values. |
| | **RealType** | Any string representing a valid data type for the database | When trying to create columns to store real values, the data type specified on this setting will be used. You |

| Section of .INI File | Setting | Accepted Values | Description |
|---|---|---|---|
| | | | need to make sure that the data type specified here is able to store 64 real values. |
| | **StringType** | Any string representing a valid data type for the database | When trying to create columns to store string values, the data type specified on this setting will be used. You need to make sure that the data type specified is able to save the number of characters that you are willing to save on your project. |
| | **TimeStampType** | Any string representing a valid data type for the database | When trying to create columns to store TimeStamp values, the data type specified on this setting will be used. |
| | **EnableTop** | 0 or 1 | When this field is set to 1, the ADO will place the TOP in the SQL statement to limit the amount of registers required. |
| | **SingleConnection** | 0 or 1 | When this field is set to 1, the ADO will open only one connection with the database, regardless of how many tasks or computers are requesting services from it. The synchronization between the tasks will be performed by the gateway, and they will not be able to be executed simultaneously if this option is enabled. |
| **Communication** | **TimeOut** | Any integer | Timeout (in seconds) to perform insert and update operations. If no value is specified, then the default of 2 seconds is used. |
| | **LongTimeOut** | Any integer | Timeout (in seconds) to perform connection and query updates. If no value is specified, then the default of 5 seconds is used. |
| | **SyncTimeOut** | Any integer | Timeout (in seconds) to perform synchronization. If no value is specified, then the default of 60 seconds is used. |
| | **OpenNonQueryTimeOut** | Any Integer | Timeout (in seconds) a request will wait for the connection used for Non-Query operations. Non-Query operations should be fast, but we experienced some issues where the provider would lock up and the gateway would have too many threads waiting with a high memory usage. If no value is specified, then the default of 3000 miliseconds is used. |
| **Connection** | **RegBufSize** | Any integer | Size (in number of registers) of the internal buffer created for each database worksheet. If no value is specified, the default value is 128. The total amount of memory used depends on the number of database worksheets and the data types of the registers. |
| | **InsertBufferSize** | Any integer | Size (in number of registers) of the buffer for all data to be inserted into the database. This is to prevent alarms/events from individually timing out, stacking up, and causing the project to freeze. If no value is specified, the default value is 1024. |
| **Options** | **CultureInfo** | Any standard language-country code (e.g., **en-US**), which is a combination of an ISO 639-1 language code and an ISO 3166-1 country code | The language or culture that should be used to format values in SQL statements. For example, this determines whether the decimal mark in numeric values is a point (**###.##**) or a comma (**###,##**).<br>This is important when the option **Disable SQL variables** is selected in the Database Configuration settings. For more information, see Database Configuration on page 108.<br>The default value of this paramter is **en-US** (i.e., "English – United States"). |
| | **DisableCloseQuestion** | 0 or 1 | When this is set to 1, no user confirmation is required to close `StADOSvr.exe`. This is important |

| Section of .INI File | Setting | Accepted Values | Description |
|---|---|---|---|
| | | | for devices that have alternative methods for exiting applications and restarting. |
| | **EnableWatchDog** | 0 or 1 | When this is set to 1, if we have a OpenNonQueryTimeOut, the gateway assumes there was a deadlock and quits. If the gateway is running locally, the runtime will restart it automatically. If it is running remotely, you should consider using the StudioProcessWatchDog. The default value of this setting is 0 |
| **Monitor** | **Monitor** | 0 or 1 | When this is set to 1, the **Monitor** menu is displayed in the Studio Database Gateway. This is enabled by default now, but it was not in previous versions of EcoStruxure Machine SCADA Expert 2020. |
| **Performance** | **GarbageCollectorConnectionEnabled** | 0 or 1 | This setting indicates whether garbage collection for database connections is enabled or disabled. The default value is 1, even if the setting itself does not exist in the .ini file, which means the garbage collector is enabled by default. You need to add this setting to the .ini file only if you want to disable the garbage collector. |
| | **GarbageCollectorConnectionPeriod** | Any integer greater than 0 | The period (in minutes) after which the garbage collector runs and old database connections are closed. The default value is 1, even if the setting itself does not exist in the .ini file, which means the garbage collector runs once per minute and all old connections (see **MaxOpenTime** below) are closed. You need to add this setting to the .ini file only if you want to increase the period. |
| | **GarbageCollectorCursorMaxOpenTime** | Any integer greater than 0 | The maximum amount of time (in minutes) that a database connection can be open before the garbage collector will close it. The default value is 1, even if the setting itself does not exist in the .ini file, which means if a connection has been open longer than 1 minute when the garbage collector runs (see **Period** above) then it will be closed. You need to add this setting to the .ini file only if you want to increase the maximum. |

The paramters are grouped into five sections — **Providers**, **Communication**, **Connection**, **Options**, and **Performance** — but all of the settings for configuring database providers are listed in the **Providers** section of the file. The default values are specified in the beginning of the file, using the prefix "Default" in each setting as shown below:

```
[Providers]
DefaultSaveMSec=3
DefaultAssembly=System.Data
DefaultConnectionClass=System.Data.OleDb.OleDbConnection
DefaultDataAdapterClass=System.Data.OleDb.OleDbDataAdapter
DefaultCommandBuilderClass=System.Data.OleDb.OleDbCommandBuilder
DefaultValueString=@Value
DefaultValueAddNumber=1 DefaultBoolType=INTEGER
DefaultIntegerType=INTEGER DefaultRealType=REAL
DefaultStringType=VARCHAR(255) DefaultTimeStampType=DATETIME
DefaultSingleConnection=0
```

The next item on the file lists the amount of providers:

```
Count=5
```

The providers are identified by the "Provider" setting followed by a number. When connecting to a database, the Provider setting in the connection string is compared to the provider's identification, in order to determine which provider will be used. If there is no provider with the value on the connection string, all the default

values are assumed. Besides its identification, each provider can have its own value per each setting. Again, if no value is specified, the default is used. Below is an example with seven providers:

```
Count=7

Provider1=MICROSOFT.JET.OLEDB
SaveMSec1=3
ColumnDelimiterPrefix1=[
ColumnDelimiterSuffix1=]
SingleConnection1=1

Provider2=SQLOLEDB
ConnectionClass2=System.Data.SqlClient.SqlConnection
DataAdapterClass2=System.Data.SqlClient.SqlDataAdapter
CommandBuilderClass2=System.Data.SqlClient.SqlCommandBuilder
ColumnDelimiterPrefix2=[
ColumnDelimiterSuffix2=]
TableDelimiterPrefix2=[
TableDelimiterSuffix2=]
RealType2=FLOAT

Provider3=MSDASQL
ConnectionClass3=System.Data.Odbc.OdbcConnection
DataAdapterClass3=System.Data.Odbc.OdbcDataAdapter
CommandBuilderClass3=System.Data.Odbc.OdbcCommandBuilder
ValueString3=?
ValueAddNumber3=0
StringType3=VARCHAR(128)
EnableTop3=0

Provider4=ORAOLEDB
Assembly4=System.Data.OracleClient
ConnectionClass4=System.Data.OracleClient.OracleConnection
DataAdapterClass4=System.Data.OracleClient.OracleDataAdapter
CommandBuilderClass4=System.Data.OracleClient.OracleCommandBuilder
ValueString4=Value
ValueAddNumber4=1
ValueStringPrefix4=:
BoolType4=Number(1)
IntegerType4=Number(10)
RealType4=Number
StringType4=VARCHAR(255)
TimeStampType4=TIMESTAMP(0)
EnableTop4=0

Provider5=ASAPROV
Assembly5=iAnywhere.Data.AsaClient
ConnectionClass5=iAnywhere.Data.AsaClient.AsaConnection
DataAdapterClass5=iAnywhere.Data.AsaClient.AsaDataAdapter
CommandBuilderClass5=iAnywhere.Data.AsaClient.AsaCommandBuilder
ValueString5=?
ValueAddNumber5=0
ColumnDelimiterPrefix5=[
ColumnDelimiterSuffix5=]
TableDelimiterPrefix5=[
TableDelimiterSuffix5=]

Provider6=MYSQLPROV
Assembly6=ByteFX.MySqlClient
ConnectionClass6=ByteFX.Data.MySqlClient.MySqlConnection
DataAdapterClass6=ByteFX.Data.MySqlClient.MySqlDataAdapter
CommandBuilderClass6=ByteFX.Data.MySqlClient.MySqlCommandBuilder
ValueString6=@Value
ValueAddNumber6=1
StringType6=VARCHAR(128)
EnableTop6=0
```

```
Provider7=MSDAORA
Assembly7=System.Data.OracleClient
ConnectionClass7=System.Data.OracleClient.OracleConnection
DataAdapterClass7=System.Data.OracleClient.OracleDataAdapter
CommandBuilderClass7=System.Data.OracleClient.OracleCommandBuilder ValueString7=Value
ValueAddNumber7=1
ValueStringPrefix7=:
BoolType7=Number(1)
IntegerType7=Number(10)
RealType7=Number
StringType7=VARCHAR(255)
TimeStampType7=TIMESTAMP(0)
EnableTop7=0
```

## Manually install Studio Database Gateway

You can manually install Studio Database Gateway on a Windows computer or Windows Embedded Standard device in order to have it relay connections between a project runtime and an external database.

Before you begin this task, you must have already installed the full EcoStruxure Machine SCADA Expert 2020 software on your computer, because doing so unpacks the standalone Studio Database Gateway software installer.

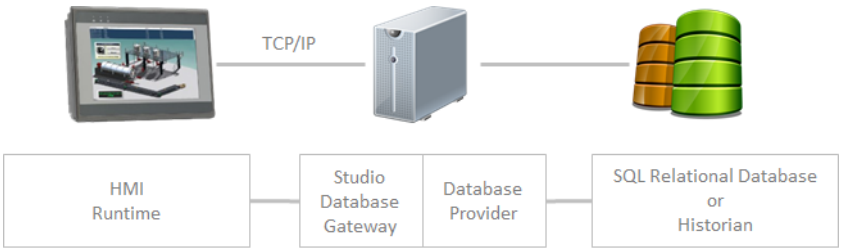
Also, you must have Administrator privileges on a computer or device in order to install any software.

Studio Database Gateway requires .NET Framework 4.7.2, which is supported by but not necessarily included in all of the versions of Windows and Windows Embedded Standard that also support this software. Before you try to install the Studio Database Gateway software on a computer or device, make sure .NET Framework 4.7.2 is installed and turned on. For more information, go to: docs.microsoft.com/en-us/dotnet/framework/

In most cases, the project runtime and the database gateway run on the same computer or device: the database gateway software is installed as part of the project runtime software, and the database gateway runs automatically when the project itself is run. If this is the case for your project, and if you are satisfied with the run-time performance of your project, you do not need to do anything more and you may skip this task.

In some cases, however, it is advantageous or even necessary to manually install and run Studio Database Gateway:

- When you want to optimize network traffic between the project runtime and the database server. For example, by running the database gateway on the same computer that hosts the database server itself and then taking advantage of the Decimation feature of the Trend Control object. The database gateway implements the decimation before it relays the data to the project runtime.
- When you want to limit the number of concurrent connections to the database server. You can have several projects access the same database server through a single database gateway. As far as the database server is concerned, the only connection is the database gateway.




***Studio Database Gateway relaying database connections***

You can install Studio Database Gateway on any Windows computer or Windows Embedded Standard device on your network, as long as that computer or device can communicate with both the project runtime and the database server.

To manually install Studio Database Gateway:

1. Locate the standalone Studio Database Gateway software installer (`GatewaySetup.exe`) in your SCADA program folder.

   If the full SCADA software was installed at its default location, the standalone Studio Database Gateway software installer should be located at:

   ```
   C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
    2020\Redist\DatabaseGateway\GatewaySetup.exe
   ```

2. Copy the installer to the Windows computer or Windows Embedded Standard device on which you want to install the database gateway.

   You can either copy the installer across the network or move it on a USB flash drive.

3. Run the installer, and then follow the installation instructions.

There are no installation options for you to select.

> **Note:** The installer will try to confirm that .NET Framework 4.5.1 is installed on the computer or device. If it cannot, it will cancel the installation.

When the installation is finished, if the software was installed at its default location, it should be located at:

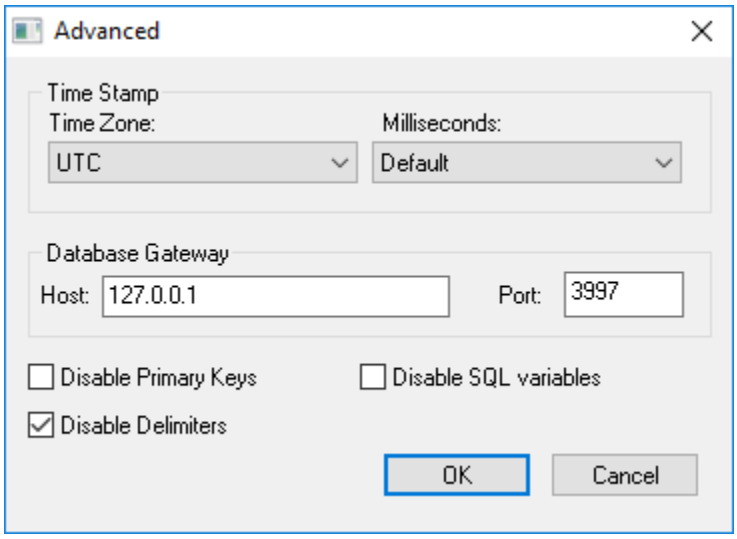
```
C:\Program Files\Studio Database Gateway v20.0\
```

Remember, the database gateway runs automatically (i.e., when the project itself is run) only if it is installed on the same computer or device as the project runtime. If the database gateway is installed on another computer or device, you must manually run it before you run your project, so that your project can find it and connect to it. For more information, see Manually running Studio Database Gateway on page 828.

## Manually running Studio Database Gateway

By default, Studio Database Gateway is run automatically when you run your project. But in some cases, you might need to manually run it.

Studio Database Gateway is installed as part of project runtime software. When your project is configured to connect to an external database and you use any of those runtimes to run your project, the project itself will try to automatically run the locally installed database gateway. This is reflected in the default database configuration in your project: the project looks for the database gateway at IP address 127.0.0.1 (i.e., localhost), port 3997, as shown in the screen shot below.



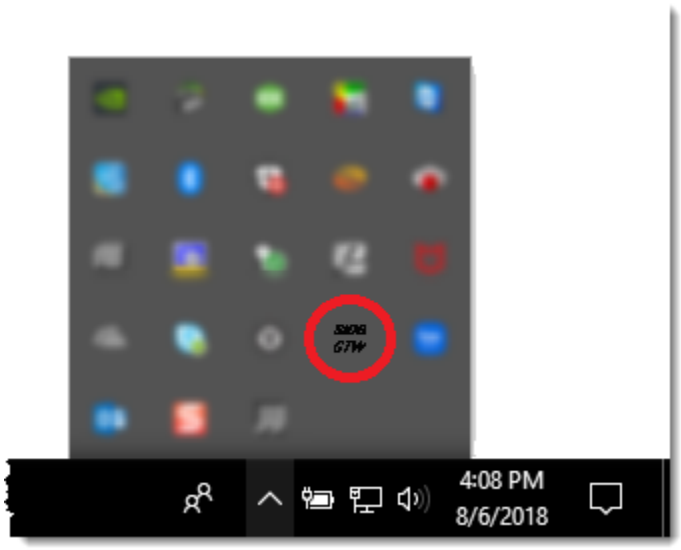

***Default settings for your project's database configuration***

If you do not need to change the configuration or access the database gateway's advanced features — that is, if you keep the default settings and simply run your project as is — you might never have occasion to manually run the database gateway. There are several reasons you might do so, however:

- If you have installed the Studio Database Gateway software on another computer, where the project runtime software cannot automatically run it;
- If you want to access the database gateway's advanced features while your project is not running;
- If you want to run the database gateway on a port other than the default port 3997; or
- If you want to run multiple instances of the database gateway on different ports.

When the Studio Database Gateway software is installed on another computer, it adds a shortcut to the Start menu, so you can use that to run it: click the **Start** button, and then, on the **Start** menu, click **All Programs** > **Studio Database Gateway** > **Studio Database Gateway**. (This option is not available for the version of Studio Database

Gateway that is installed as part of the project runtime software.) The Studio Database Gateway icon appears in the notificiation area, at the far right of the Windows taskbar, to show that the program is running.




*The Studio Database Gateway icon in the notification area*

Otherwise, to manually run the database gateway, you must know where the Studio Database Gateway program file (`StADOSvr.exe` or `StADOSvrCE.exe`) is actually located. When it is installed as part of the project runtime software, it is located in the same program folder. For example, for SCADA, the Studio Database Gateway program file should be located at:

```
C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin
\StADOSvr.exe
```

When the Studio Database Gateway software is installed on another computer, the program file should be located at:

```
C:\Program Files\Studio Database Gateway v20.0\StADOSvr.exe
```

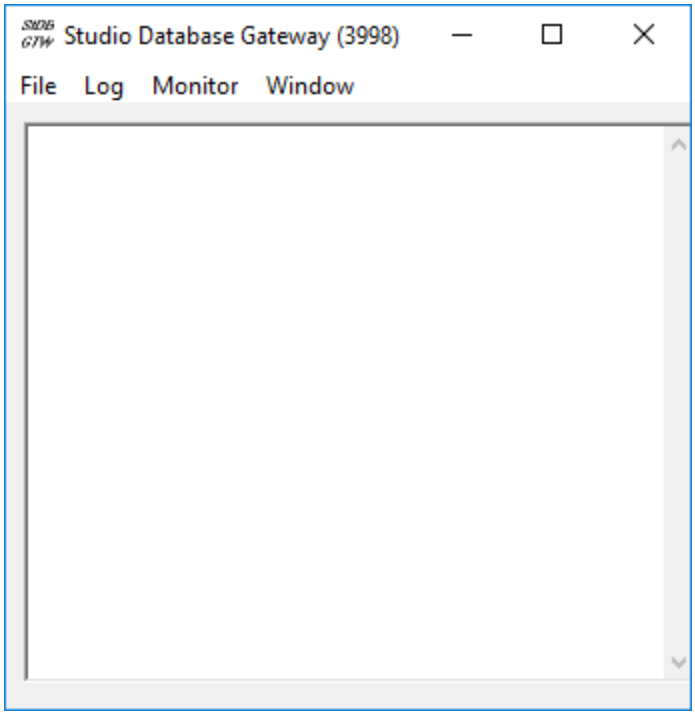
In both cases, if you want to run a single instance of the database gateway on the default port 3997, simply double-click the program file. If you want to run the database gateway on a port other than the default port 3997, or if you want to run multiple instances of the database gateway on different ports, open a *Command Prompt* window and then type the following:

```
StADOSvr.exe <port number>
```

For example:

```
StADOSvr.exe 3998
```

There is no limit on the number of instances that you can run, as long as you have the necessary system resources and unused ports. For each instance, its port number is displayed in the title bar of the program window, as shown in the screen shot below.

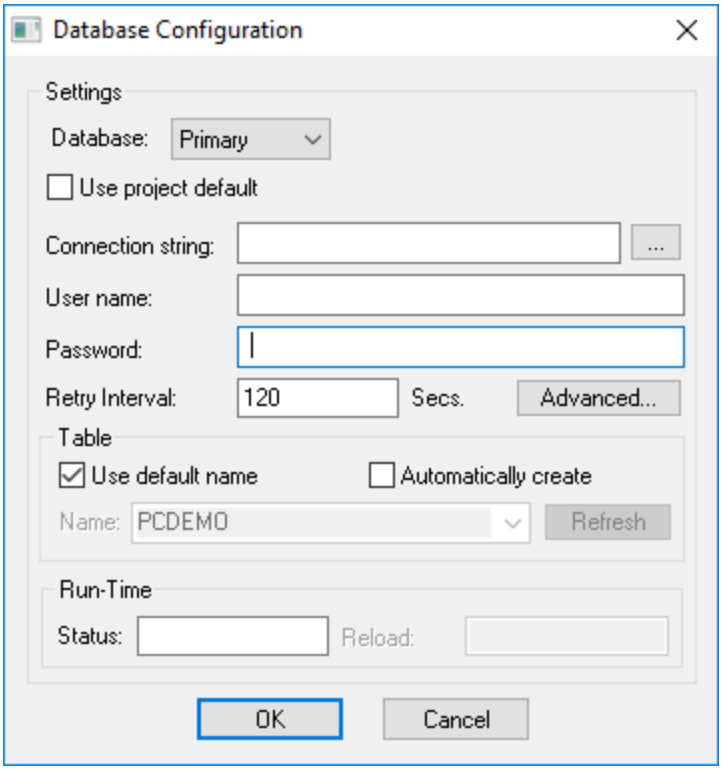



***Studio Database Gateway running on port 3998***

Whenever you run Studio Database Gateway, you should note the host name or IP address of the computer and the number of the port on which the database gateway is running. You will need this information in order to update the database configuration in your project.

## Database Configuration

The *Database Configuration* dialog allows you to configure the necessary settings to link EcoStruxure Machine SCADA Expert to an external database file.



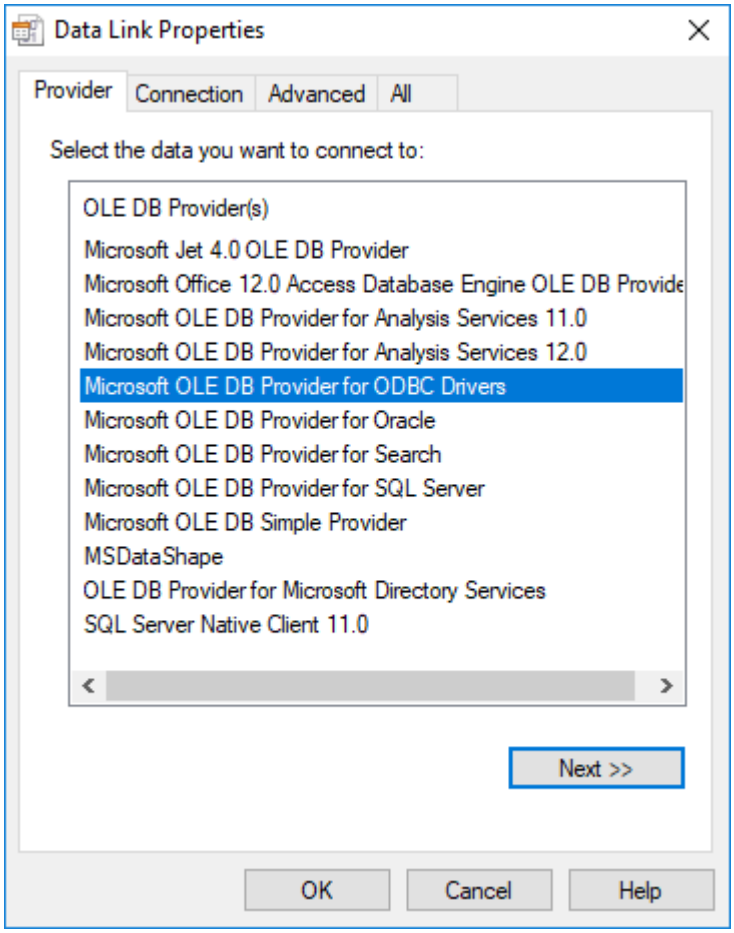

***Database Configuration dialog***

- **Database** combo-box: Allows you to select either Primary or Secondary. With Primary, all settings displayed in the Database Configuration window apply to the Primary Database interface. Otherwise, they apply to the Secondary Database interface. You can configure the Secondary database in the following modes:
  - **Disabled**: In this mode, EcoStruxure Machine SCADA Expert saves data in the Primary Database only. If the Primary Database is unavailable for any reason, the data is not saved anywhere else. This option may cause loss of data if the Primary Database is not available.
  - **Redundant**: In this mode, EcoStruxure Machine SCADA Expert saves data in both Primary and Secondary Databases. If one of these databases is unavailable, EcoStruxure Machine SCADA Expert keeps saving data only in the database that is available. When the database that was unavailable becomes available again, EcoStruxure Machine SCADA Expert synchronizes both databases automatically.
  - **Store and Forward**: In this mode, EcoStruxure Machine SCADA Expert saves data in the Primary Database only. If the Primary Database becomes unavailable, EcoStruxure Machine SCADA Expert saves the data in the Secondary Database. When the Primary Database becomes available again, EcoStruxure Machine SCADA Expert moves the data from the Secondary Database into the Primary Database.

> **Note:** The Primary and Secondary can be different types of databases. However, they must have the same fields.

Using the Secondary Database, you can increase the reliability of the system and use the Secondary Database as a backup when the Primary Database is not available. This architecture is particularly useful when the Primary Database is located in the remote station. In this case, you can configure a Secondary Database in the local station to save data temporarily if the Primary Database is not available (during a network failure, for instance).

- **Use project default** checkbox: When this option is checked, EcoStruxure Machine SCADA Expert uses the settings configured in the Default Database for the task that is being configured (Connection string, User name, Password, Retry Interval and Advanced Settings). When this option is not checked, you can configure these settings individually to the current task.
- **Connection string** field: This field defines the database where EcoStruxure Machine SCADA Expert will write and read values as well as the main parameters used when connecting to the database. Instead of writing the Connection string manually, you can press the browse button (...) and select the database type from the **Data Link Properties** window.




***Data Link Properties dialog***

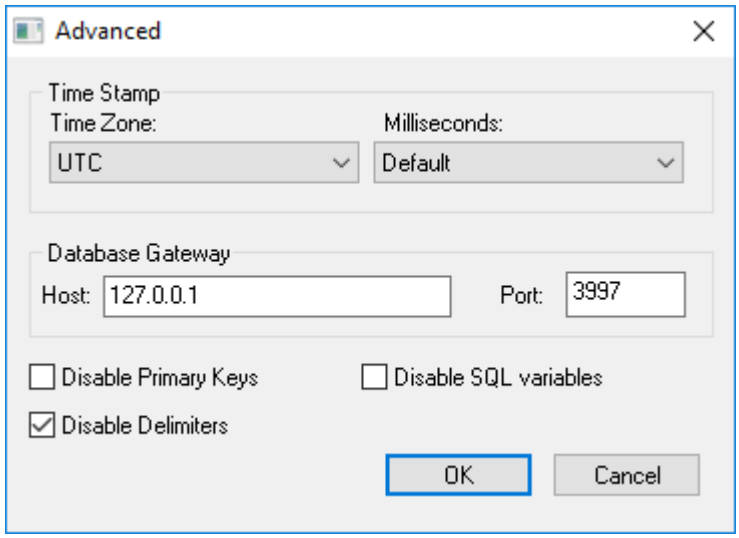
> **Note:** The list of Database Providers shown in the Data Link Properties window depends on the providers actually installed and available in the computer where you are running EcoStruxure Machine SCADA Expert. Consult the operating system documentation (or the database documentation) for further information regarding the settings of the Provider for the database that you are using.

- **User name** field: User name used to connect to the database. The user name configured in this field must match the user name configured in the database.
- **Password** field: Password used to connect to the database. The password configured in this field must match the password configured in the database.

> **Note:** In the **Connection string**, **User name**, and **Password** boxes, as in other boxes and fields that accept plain text, you can configure tag names in curly brackets (e.g., `{MyTag}`) in order to use the values of those tags. You can then change the tag values during run time and thereby change your database connection and credentials. You should be aware, however, that tag values are not

encrypted when they are sent between the EcoStruxure Machine SCADA Expert project runtime server and connected thin clients. Therefore, to ensure that your database credentials cannot be intercepted or compromised, you can configure only server tags — that is, tags that have **Scope** set to **Server**; for more information, see Choosing the Tag Scope on page 156 — in these boxes. The tags will be evaluated on the server only, and no tag values will be sent between the server and client.

- **Retry Interval** field: If EcoStruxure Machine SCADA Expert is unable to connect to the database for any reason, it retries automatically to connect to the database after the number of seconds configured in this field have passed.
- **Advanced** button: After pressing this button, you have access to customize some settings. For most projects, the default value of these settings do not need to be modified and should be kept.




***Database Configuration: Advanced dialog***

  - **Time Zone** combo box:
    - **Local Time + Time Difference**: Save the local time on the computer, plus the difference (bias) between the local time zone and Coordinated Universal Time (UTC).
    - **Local Time**: Save the local time only with no bias. This is not recommended.
    - **UTC**: Save the UTC time only. This is the default, and it is strongly recommended for most situations.
  - **Milliseconds** combo box: You can configure how the milliseconds will be saved when saving the date in the database. Each database saves the date in different formats; for example, some databases do not support milliseconds in a **Date** field. The following options are available:
    - **Default**: Uses the format pre-defined for the current database. The databases previously tested by Schneider Electric are previously configured with the most suitable option. When selecting Default, EcoStruxure Machine SCADA Expert uses the setting pre-configured for the current database type. If you are using a database that has not been previously configured, the Default option attempts to save the milliseconds in a separate field.

      **Tip:** The default option for each database is configured in the `StADOSvr.ini` file, stored in the `\BIN` sub-folder of EcoStruxure Machine SCADA Expert. See Studio Database Gateway on page 818 for information about how to configure the `StADOSvr.ini` file.

    - **Disable**: Does not save the milliseconds at all when saving the date in the database.
    - **Enable**: Saves the milliseconds in the same field where the date is saved.
    - **Separate Column**: Saves the milliseconds in a separated column. In this case, the date is saved in one field (without the milliseconds precision) and the number of milliseconds is saved in a

different column. This option is indicated where you want to save timestamps with the precision of milliseconds but the database that you are using does not support milliseconds for the **Date** fields.

- **Database Gateway**: Enter the Host Name/IP Address where the EcoStruxure Machine SCADA Expert Database Gateway will be running. The TCP Port number can also be specified, but if you are not using the default, you will have to configure the EcoStruxure Machine SCADA Expert Database Gateway with the same TCP Port. See Studio Database Gateway on page 818 for information about how to configure the advanced settings for the EcoStruxure Machine SCADA Expert ADO Gateway.
- **Disable Primary Keys**: For some modules, EcoStruxure Machine SCADA Expert will try to define a primary key to the table in order to speed up the queries. If you are using a database that does not support primary keys (e.g., Microsoft Excel), then you should select (check) this option.
- **Disable Delimiters**: Select this troubleshooting option to disable the delimiters that are used to format communications with the database. Delimiters can cause problems when a Trend Control or Grid builds a query that includes aggregates such as Min and Max.
- **Disable SQL variables**: Select this troubleshooting option to disable SQL variables, such as **`@Value1`** and **`?`**, that are often used in SQL statements and queries. Some specific database providers do not support these variables.

> **Note:** If you select this option, you might need to specify the language or culture that should be used to format values in SQL statements. For more information, see "Advanced Settings" in Studio Database Gateway on page 818.

### Table Pane

This area allows you to configure the settings of the Table where the data will be saved. All tasks can share the same database. However, each task (Alarm, Events, Trend worksheets) must be linked to its own Table. EcoStruxure Machine SCADA Expert does not check for invalid configurations on this field, therefore you should make sure that the configuration is suitable for the database that you are using.

- **Use default name** checkbox: When this option is checked (default), EcoStruxure Machine SCADA Expert saves and/or retrieves the data in the Table with the default name written in the **Name** field.
- **Automatically create** checkbox: When this option is checked (default), EcoStruxure Machine SCADA Expert creates a table with the name written in the **Name** field automatically. If this option is not checked, EcoStruxure Machine SCADA Expert does not create the table automatically. Therefore, it will not be able to save data in the database, unless you have configured a table with the name configured in the **Name** field manually in the database.
- **Name**: Specifies the name of the Table from the database where the history data will be saved.

> **Tip:** To specify a sheet in a Microsoft Excel spreadsheet file, use the following syntax:
>
> **`[`*`sheetname`*`$]`**

- **Refresh** button: If the database configured is currently available, you can press the **Refresh** button to populate the **Name** combo-box with the name of the tables currently available in the database. In this way, you can select the table where the history data should be saved instead of writing the Table name manually in the **Name** field.

### Run-Time Pane

This area allows you set runtime values. The following fields are available:

- **Status** (output) checkbox: The tag in this field will receive one of the following values:

| Value | Description |
|---|---|
| 0 | Disconnected from the database. The database is not available; your configuration is incorrect or it is an illegal operation. |
| 1 | The database is connected successfully. |
| 2 | The database is being synchronized. |

- **Reload** (output): Specify a reload tag if you are using curly brackets in any of the configuration fields. When you want to reconnect to the database using the updated values on your tags, set the tag on this field to 1. EcoStruxure Machine SCADA Expert will update the configuration when trying to perform an action in the database, setting the tag back to 0 when it is finished.

**See also**:

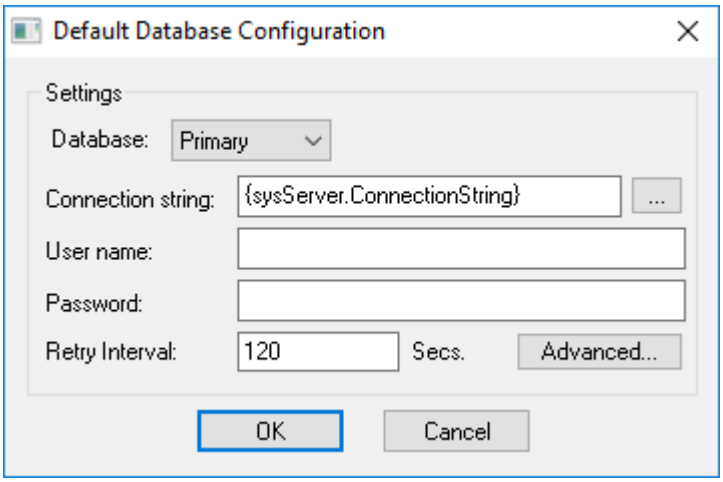
Configuring a Default Database for All Task History.

## Configuring a Default Database for All Task History

You can configure a Default Database that will save the historical data from all Tasks in a project. After you do, when you create a new Task worksheet, you can choose either to use the Default Database or to configure a new database for that specific worksheet.

To configure the connection settings for the Default Database:

1. On the Project tab of the ribbon, in the Settings group, click **Options**. The *Project Settings* dialog is displayed.
2. Click **Configure**. The *Default Database Configuration* dialog is displayed.




***Default Database Configuration dialog***

Please refer to *Database Configuration* dialog for help completing the fields in this window.

## Support for AVEVA Insight and Historian

This software includes support for saving historical data either to AVEVA Insight online or to a Historian database located on-premises.

Insight is a secure, managed solution for collecting, storing, visualising, and analysing industrial data for faster, smarter business decisions. It consolidates disparate data for complete visibility into how your business is performing, and enables users throughout the enterprise to access data and information from anywhere. For more information, go to: https://sw.aveva.com/monitor-and-control/industrial-information-management/insight

Historian is a high-performance process database capable of storing the huge volumes of data generated from today's industrial facilities. It combines advanced data storage and compression techniques with an industry-standard query interface to ensure open access to your process, alarm, and event data. For more information, go to: https://sw.aveva.com/monitor-and-control/industrial-information-management/historian

At this time, only Trendworksheets can save historical data to AVEVA Insight and Historian.

> **Note:** To use this feature, you must have selected the **Historian** option when you installed this software. For more information, see Install the full EcoStruxure Machine SCADA Expert 2020 software on page 35.

### *Connect to AVEVA Insight using AVEVA Insight Publisher*

Use Studio Database Gateway and AVEVA Insight Publisher to establish a connection between your project and AVEVA Insight, so that you can publish historical data from your project to your Insight account.

This task is a supplement to other tasks that describe how to save historical data from your project; it assumes you have an appropriate worksheet (e.g., Trend) open for editing, and it starts from that point.

This task also assumes you already have an Insight account. (It is possible to create an account during this task, while you are using AVEVA Insight Publisher, but doing so might interrupt this task.) If you do not have an account, or if you simply want more information about AVEVA Insight, go to: https://sw.aveva.com/monitor-and-control/industrial-information-management/insight

In most cases, the database gateway is automatically installed and run on the same computer that hosts the project runtime. Depending on your network architecture or the nature of your project, however, you may choose to manually install and run the database gateway on separate computer — either the computer that hosts the database server or an entirely different computer. If you do that, you should know the host name or IP address of that computer, as well as the TCP port number for the database gateway. For more information, see Manually install Studio Database Gateway on page 826.

Multiple project runtimes can use the same database gateway, but that database gateway can connect to only one Insight account at a time. If you need to connect to multiple accounts, you can run multiple instances of the database gateway on different TCP ports and then configure the worksheets in your project to use the appropriate instances. For more information, see Manually running Studio Database Gateway on page 828.

> **Note:** Your project can only save historical data to AVEVA Insight. It cannot retrieve historical data from AVEVA Insight.

To connect to AVEVA Insight using AVEVA Insight Publisher:

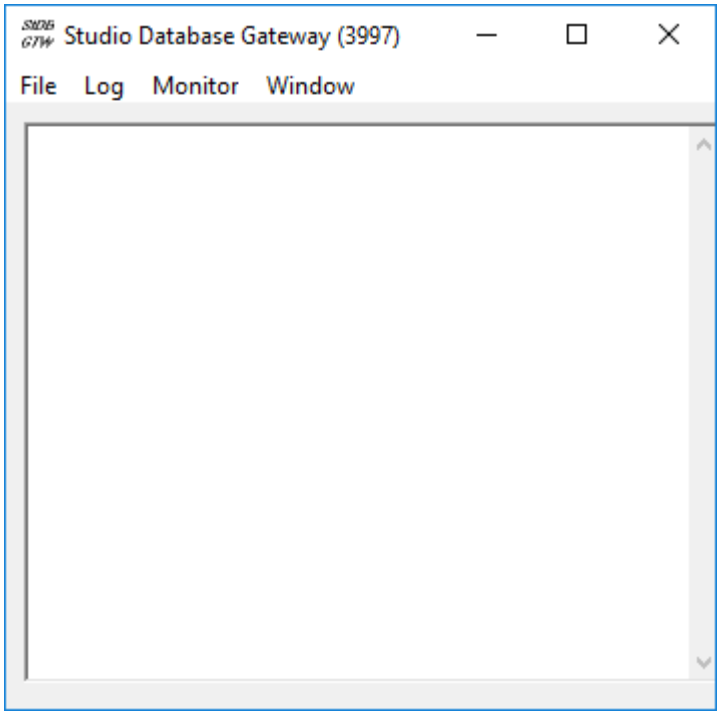
1. Create a new AVEVA Insight Connection in Studio Database Gateway:
   a) Run Studio Database Gateway, if it is not already running.

      The database gateway should run automatically when you run a project that requires it. Otherwise, double-click the Studio Database Gateway program file (`StADOSvr.exe`), which is located in the EcoStruxure Machine SCADA Expert 2020 program folder.

      The Studio Database Gateway program icon (**StDB GTW**) appears in the notificiation area, on the right side of the Windows taskbar, to show that the program is running. The program window is automatically hidden by default, however.

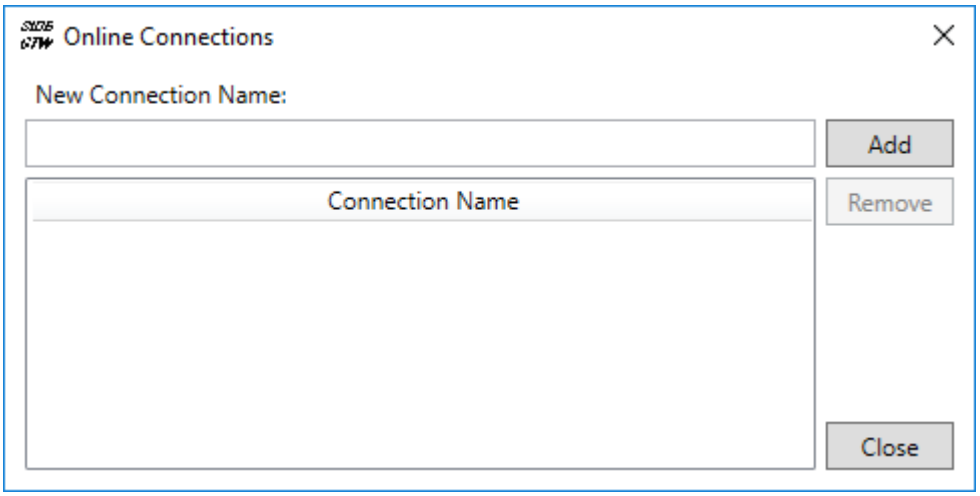
   b) Right-click the icon in the notification area, and then on the shortcut menu, clear the **Hide** option.

The *Studio Database Gateway* program window is displayed. The number in parentheses (e.g., 3997) is the TCP port on which this instance of the database gateway is running.




c) Go to **File**, and then click **AVEVA Insight Connections**.

The *Online Connections* dialog box is displayed.




d) In the **New Connection Name** box, type the name of the connection, and then click **Add**.
AVEVA Insight Publisher is opened, the **Register** option is automatically selected, and the first page of the registration wizard is displayed.

e) Use the registration wizard to sign in to your Insight account and then register the database gateway as a new data source.

For more information about how to do this, follow the instructions in the registration wizard or go to: https://sw.aveva.com/monitor-and-control/industrial-information-management/insight

> **Note:** If you are developing your project on a Windows Server computer, or if you plan to run your project on a Windows Server computer after you have finished developing it, you might need to turn off Internet Explorer Enhanced Security Configuration (IE ESC) in Server Manager. IE ESC can interfere with attempts to sign in to AVEVA Insight, and adding AVEVA Insight as a Trusted Site will not resolve the issue.

When you are done, the new connection is saved in the list of available connections in the *Online Connections* dialog box.

f) Click **Close** to close the *Online Connections* dialog box.

g) In the *Studio Database Gateway* program window, on the **File** menu, click **Exit**.
The program window is closed.

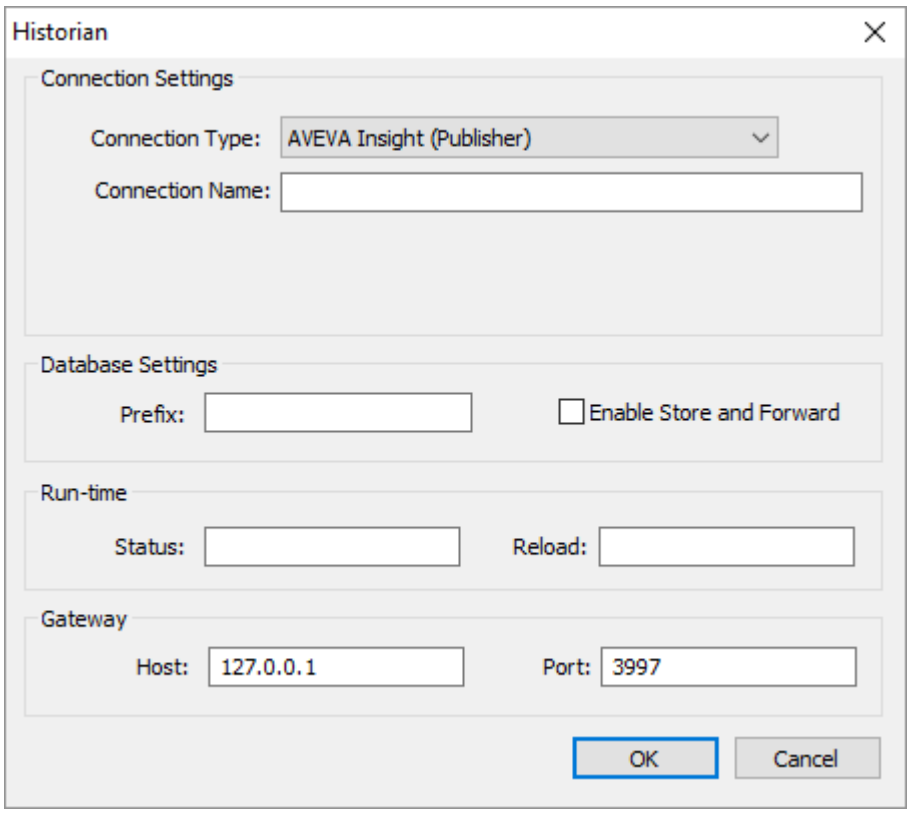
You can repeat this step to create a list of available connections in the database gateway, but the database gateway can use only one connection at a time (i.e., it can connect to only one Insight account at a time). If you run multiple instances of the database gateway on the same computer, however, all of the instances will share the same list of available connections and each instance can use a different connection.

Keep in mind that these connections are saved with the database gateway and not with your project files, so you might need to repeat this step for each station on which you install the project runtime and then run your project. You cannot copy connections from one station to another. As mentioned earlier, however, you may choose to manually install and run the database gateway on separate computer and then have multiple project runtimes use that database gateway. If you do that, then you need to create and save the connections only once, for that database gateway.

2. In the project development environment, in the header of the worksheet, make sure **Historian** is selected as the history format, and then click **Historian Configuration**.
The *Historian* dialog box is displayed.

3. In the **Connection Settings** area, in the **Connection Type** list, select **AVEVA Insight (Publisher)**.

   The settings for the selected connection type are displayed.




4. In the **Connection Name** box, type the name of a connection that you previously created and saved in Studio Database Gateway.

   You can type a string expression for this setting (e.g., `{MyConnection}`), but the connection must be reloaded whenever the connection settings change. See "Reload" below.

5. In the **Database Settings** area, in the **Prefix** box, type a prefix that will be added to the tags saved in the database in order to keep them grouped together.
   For example, if you are configuring the connection for Trend worksheet **`TREND001`**, you could make that the prefix as well.

   This setting is optional, but if you do not specify a prefix, the tags will be not be sorted together in the database and therefore might be difficult to find. You can type a string expression for this setting (e.g., `{MyPrefix}`), but the connection must be reloaded whenever the connection settings change. See "Reload" below.

6. If you want your project to store its historical data when it is not connected to the database and then forward the stored data when it becomes connected again, select the **Enable Store and Forward** option.

   The historical data will be saved in the project folder on the computer or device that hosts the project runtime. This can use a large amount of hard drive space (or non-volatile memory) if the connection is unavailable for a long time.

7. In the **Run-time** area, in the **Status** box, type the name of a project tag (Integer type) that will receive values indicating the status of the connection during project run time.

   This setting is optional. The specified tag can receive the following possible values:

| Value | Description |
|---|---|
| 1 | Connected to database server (AVEVA Insight or Historian). |
| 2 | Not connected to database server (AVEVA Insight or Historian). |

8. In the **Reload** box, type the name of a project tag that will trigger a reload of the connection during project run time.

When the value of this tag changes, all other connection settings that have been configured with string expressions (see the preceding steps) will be updated and the connection will be reloaded using those updated settings.

The connection is reloaded only after the settings have been updated, and the settings are updated only when the value of the specified **Reload** tag changes. This is to make sure all of the settings are updated at the same time, rather than when the value of any single string expression changes.

9. In the **Gateway** area, in the **Host** and **Port** boxes, type the host name or IP address and port number of the database gateway that your project will use.

   The default settings are for the database gateway running on localhost — that is, on the same computer or device that hosts the project runtime.

10. Click **OK** to save the settings and close the dialog box.

If your project is configured to use a database gateway hosted on another computer, make sure it is running before you try to run your project. If the database gateway is not running, your project will not be able to use it to connect to the specified database.

In contrast, if your project is configured to use the database gateway hosted on the same computer that hosts the project runtime, you do not need to do anything — when you run your project, it in turn will automatically run the database gateway.

### *Connect to a Historian database located on-premises*

Configure your project to connect to a Historian database located on-premises (i.e., a database server hosted on a Windows computer on your own network).

This task is a supplement to other tasks that describe how to save historical data from your project; it assumes you have an appropriate worksheet (e.g., Trend) open for editing, and it starts from that point.

Before you begin this task, you should know the host name or IP address of the Historian database server on your network, and you should have the necessary credentials (i.e., user name and password) to access that database.

Your project will use Studio Database Gateway to manage communications between itself and the Historian database server. In most cases, the database gateway is automatically installed and run on the same Windows computer that hosts the project runtime. Depending on your network architecture or the nature of your project, however, you may choose to manually install and run the database gateway on another Windows computer — either the computer that hosts the database server or an entirely different computer. If you do that, you should know the host name or IP address of that computer, as well as the TCP port number for the database gateway. For more information, see Manually install Studio Database Gateway on page 826.

Multiple project runtimes can use the same database gateway, but that database gateway can connect to only a single Historian database. If you need to connect to multiple databases, you can run multiple instances of the database gateway on different TCP ports and then configure the worksheets in your project(s) to use the appropriate instances. For more information, see Manually running Studio Database Gateway on page 828.

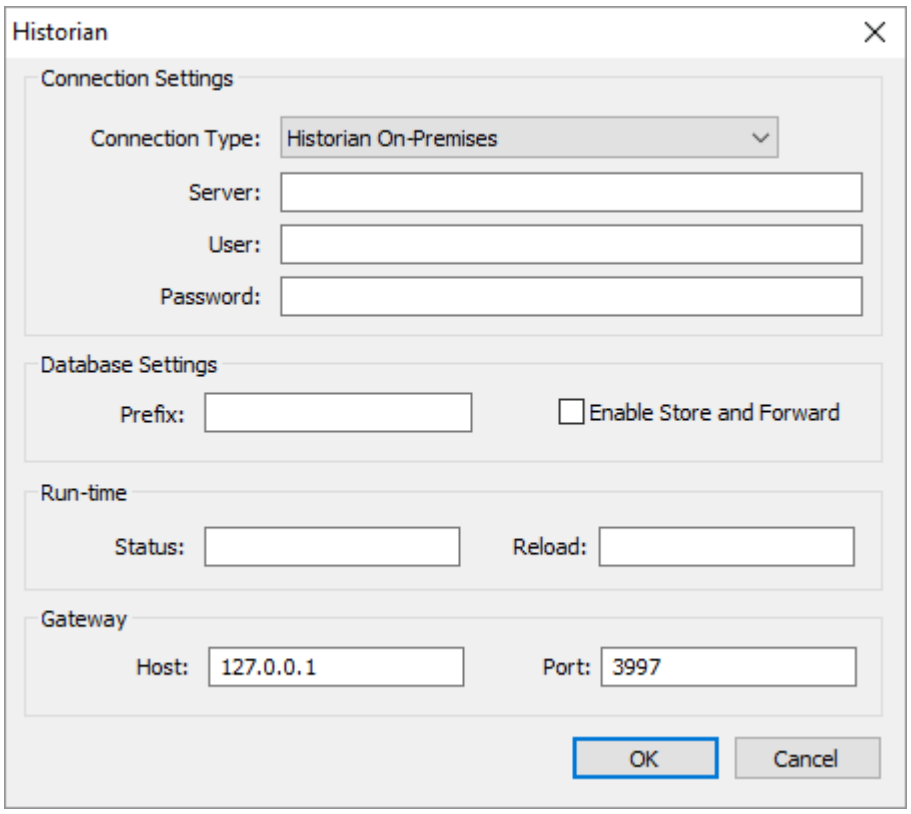
> **Note:** The latest version of Studio Database Gateway is compatible only with Historian 2017 (a.k.a. version 17.0) or later. If you want to connect to an earlier version of Historian, you must acquire and install an earlier version of the database gateway. For more information about how to do this, contact your software distributor.

To connect to a Historian database located on-premises:

1. In the project development environment, in the header of the worksheet, make sure **Historian** is selected as the history format, and then click **Historian Configuration**.
   The *Historian* dialog box is displayed.

2. In the **Connection Settings** area, in the **Connection Type** list, select **Historian On-Premises** if it is not already selected.

   The settings for the selected connection type are displayed.




3. In the **Server** box, type the host name or IP address and port number of the Historian database server. For example, `HistorianDBServer:123`.

   The port number is optional. If you did not change the port number in the Historian database settings, you can omit it here and the default (port 32568) will be used. Otherwise, the port number must match the one that is specified in the Historian database settings.

   You can type a string expression for this setting (e.g., `{MyServer}`), but the connection must be reloaded whenever the connection settings change. See "Reload" below.

4. In the **User** and **Password** boxes, type your credentials to access the Historian database.

   You can type a string expression for the **User** setting (e.g., `{MyUser}`), but the connection must be reloaded whenever the connection settings change. See "Reload" below.

5. In the **Database Settings** area, in the **Prefix** box, type a prefix that will be added to the tags saved in the database in order to keep them grouped together.
   For example, if you are configuring the connection for Trend worksheet **`TREND001`**, you could make that the prefix as well.

   This setting is optional, but if you do not specify a prefix, the tags will be not be sorted together in the database and therefore might be difficult to find. You can type a string expression for this setting (e.g., `{MyPrefix}`), but the connection must be reloaded whenever the connection settings change. See "Reload" below.

6. If you want your project to store its historical data when it is not connected to the database and then forward the stored data when it becomes connected again, select the **Enable Store and Forward** option.

   The historical data will be saved in the project folder on the computer or device that hosts the project runtime. This can use a large amount of hard drive space (or non-volatile memory) if the connection is unavailable for a long time.

7. In the **Run-time** area, in the **Status** box, type the name of a project tag (Integer type) that will receive values indicating the status of the connection during project run time.

   This setting is optional. The specified tag can receive the following possible values:

| Value | Description |
|---|---|
| 1 | Connected to database server (AVEVA Insight or Historian). |
| 2 | Not connected to database server (AVEVA Insight or Historian). |

8. In the **Reload** box, type the name of a project tag that will trigger a reload of the connection during project run time.

   When the value of this tag changes, all other connection settings that have been configured with string expressions (see the preceding steps) will be updated and the connection will be reloaded using those updated settings.

   The connection is reloaded only after the settings have been updated, and the settings are updated only when the value of the specified **Reload** tag changes. This is to make sure all of the settings are updated at the same time, rather than when the value of any single string expression changes.

9. In the **Gateway** area, in the **Host** and **Port** boxes, type the host name or IP address and port number of the database gateway that your project will use.

   The default settings are for the database gateway running on localhost — that is, on the same computer or device that hosts the project runtime.

10. Click **OK** to save the settings and close the dialog box.

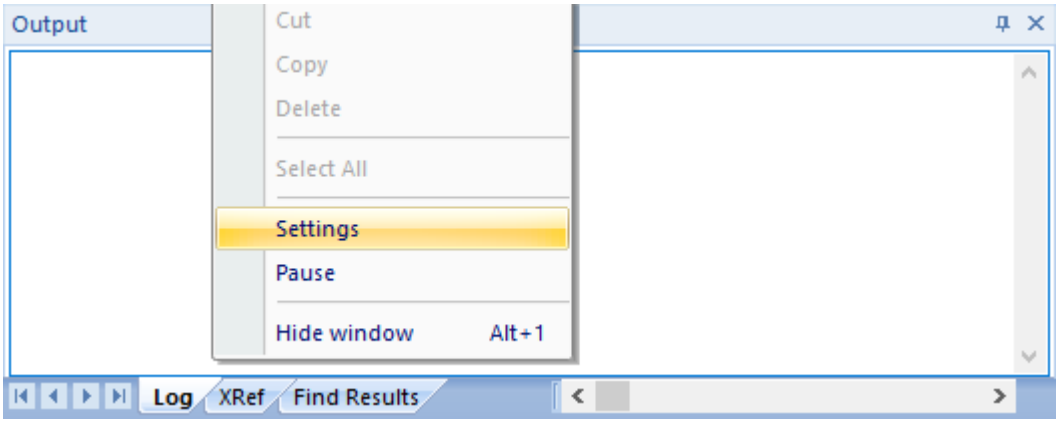
If your project is configured to use a database gateway hosted on another computer, make sure it is running before you try to run your project. If the database gateway is not running, your project will not be able to use it to connect to the specified database.

In contrast, if your project is configured to use the database gateway hosted on the same computer that hosts the project runtime, you do not need to do anything — when you run your project, it in turn will automatically run the database gateway.

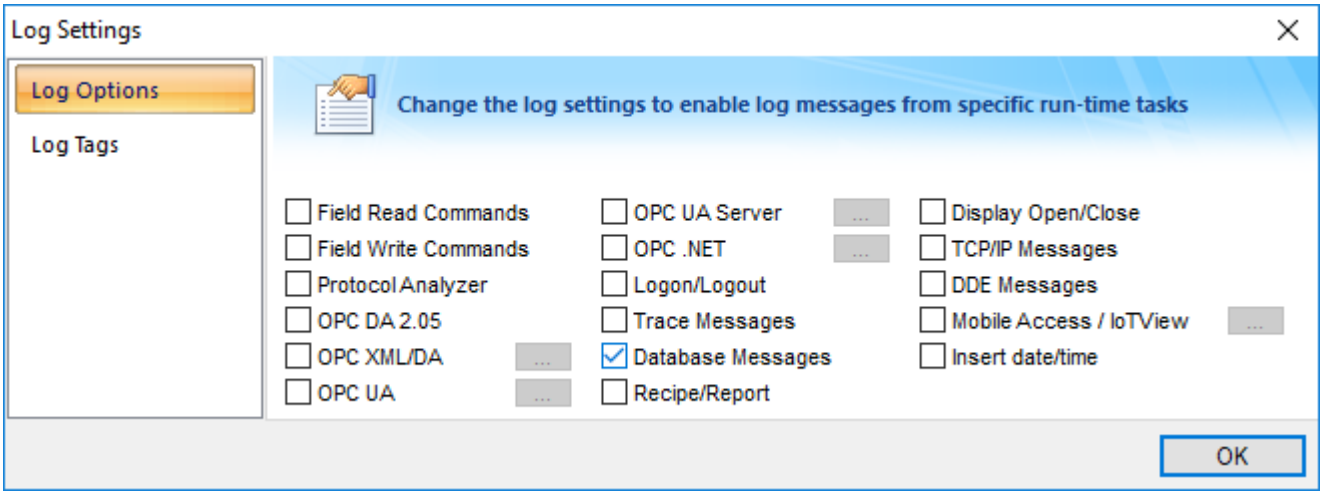
## Database Troubleshooting

EcoStruxure Machine SCADA Expert database interface provides powerful tools that will help you to identify configuration problems with databases. If you are having problems interfacing with a database, you should first enable the **Database Messages** in the *Log*window. You can do so by following the steps below:

1. In the EcoStruxure Machine SCADA Expert Development environment, make sure to show the *Output* window (**Output Window** check box on the View tab of the ribbon).
2. Right-click in the *Output*window (usually located in the lower-right corner of the development environment), and then click **Settings** on the shortcut menu:




3. In the *Log Settings* dialog, check the **Database Messages** option:




***enabling Database Messages***

After enabling this option, the *Output* window will display error messages related to the database. The FAQ section below lists some common errors that you can see in the *Output* window.

### GENERAL QUESTIONS

**Q**: I configured my database, but the runtime modules (Alarm, Trend, and Events) are not being saved to the database. I only see the following error message in the *Output* window:
Database: Error: Error to add new register[CMD_ADD].

**A**: Most of the database errors in the *Output* window will be followed by additional information such as the SQL command being executed, the Connection String and the Table name. Error messages such as the one described above, will usually happen after a more detailed message. For example, if your Trend task fails to add a register in the database because the cable is disconnected, you should first receive a network error;

if the task tries to add more registers before the time specified in the **Retry** field (see *Database Configuration* dialog), it will only display **`Database: Error: Error to add new register[CMD_ADD]`**. If you think that your configuration is correct, and you want to debug this type of problem, reduce the Retry. Then you should see more detailed information.

**Q**: When I try to access the MySQL database server, I get the following message:
Object is not set to an instance of an object.

**A**: This problem was detected under the following conditions:

- A known bug in MySQL Connector/Net v6.1.2 would not correctly specify the charset; and
- The database table you are trying to access doesn't exist.

To solve this problem, make sure you are using MySQL Connector/Net v6.2.0 and that the table you are accessing exists in the database.

**Q**: Why is the Database Interface automatically closing some connections?

**A**: By default, the Database Interface can have a maximum of 1000 connections. When this maximum is exceeded, the oldest connection is automatically closed to allow the new connection and the *Output* window displays an extended message describing which connection was closed and what was the last command executed.

To increase the maximum number of database connections, open the project file (`<project name>.app`) in a text editor and change the following setting:

```
[StDB]
MaxConnections=number_of_connections
```

Keep in mind that increasing the maximum number of connections may decrease project performance.

**Q**: I configured my Connection String using the browser and the Data Link Properties Window. When I click the Test button, it says "Test succeeded". However, when I run my project, the Database Interface displays error messages, and I am not able to save data.

**A**: The Data Link Properties Window uses OLE DB to interface with the Database. EcoStruxure Machine SCADA Expert Database Interface uses ADO.NET; therefore, you can have the OLE DB provider on your machine and be missing the ADO.NET provider. It is also possible that you are using an ADO.NET provider that is not listed in the StADOSvr.ini file. Please refer to Studio Database Gateway on page 818 for more information about adding ADO.NET providers to the **`StADOSvr.ini`** file.

**Q**: Why, when I update information in one line in the Grid object, is it updating more than one line in my database?

**A**: The grid object issues an update command in the database using the values in all the columns for the specific row **t**hat you are trying to update. If you have rows with duplicate values, you might see this problem. If your table has a primary key or any other unique field that you do not want to display in the *Grid* object, you can add it to the **Columns** but specify the **Width** 0. This will fix the problem.

**Q**: Why do I have to use a separate **Column** to store the milliseconds on my database?

**A**: Some databases do not support milliseconds in the **Time Stamp** field. EcoStruxure Machine SCADA Expert Database interface, by default, requires another column for the milliseconds. If your database can handle milliseconds, or if you do not want to record the milliseconds, you can change the default behavior in the Advanced settings. Note that some databases are able to store milliseconds, but they have lower precision. If you mix different databases with different precisions in redundant mode, you can get synchronization problems.

**Q**: My project works fine when I run in emulation mode. But when I send to the Windows Embedded device, it cannot communicate with my database.

**A**: It might be the case that your Windows Embedded device does not have the correct version of .NET Framework or it does not have the necessary data provider for your database. Try to use the database gateway remotely. For more information, see Manually install Studio Database Gateway on page 826.

**Q**: When I try to connect to the database, why do I receive the message, **`Error to create connection class`**?

**A**: The .Net Provider that you are trying to use is not installed on your machine. This error message is usually followed by the provider name; if you are using the Sybase database, for instance, the message is followed

by **[iAnywhere.Data.AsaClient.AsaConnection]**. The Provider is the iAnywhere.Data.AsaClient. You can check if the provider is installed on your machine by going to the **Control Panel** > **Administrative Tools** > **Microsoft .Net Framework x.x Configuration**. The provider should be listed in the *Assembly Cache*.

**Q**: What if I have the provider assembly (usually a .dll file) but it is not listed in the *AssemblyCache*?

**A**: If your assembly has a strong name, you can register it in the *Assembly Cache* using the gcautil program. Or it should work if you copy your assembly to the same folder as the `StADOSvr.exe` (usually the `Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin` folder).

**Q**: I am not able to access my table from the Grid when I use a specific condition. But if no condition is applied, it works fine. Why is that?

**A:** You should check for the following items:

1. Follow the Troubleshooting steps, and look for error messages in the log. An error message can tell you if you have made a mistake, such as entering with a wrong column name or specifying an invalid data format.
2. Some databases have problems when you use reserved words as column names. Therefore, you should avoid using column names such as Time, Date, Numeric, etc.
3. If your column name starts with AND or OR (e.g., ORange), enter the name surrounded by square brackets. For example, instead of ORange=10, enter [ORange]=10.
4. If you are using SQL Server CE, you might have some problems when querying string fields. It has been identified that filters do not work with NCHAR data types; however, they do work if you declare these fields as NVARCHAR(<Number>). You might try to recreate your table by using this data type. An example of a command that creates a table with strings that can be queried is displayed below:

```
CREATE TABLE Table1 (Name NVARCHAR(128), Age Numeric, Sex NVARCHAR(1))
```

### MYSQL

**Q**: When I try to access the database from my local machine it works fine, but when I move my project to a remote machine, it says **Access Denied**.

**A**: Each user on a MySQL database has a property associated with it that indicates the computer fromwhich it can get access to the database. By default, this property is set to **localhost**, so you will only be able to access the database if you are accessing from the local computer. You should read the MySQL manual for information about changing this setting.

**Q**: Sometimes when I try to synchronize a remote MySQL database with a local MySQL database, or if I try to use application redundancy, a connection to the ADO.NET interface is opened and never closed.

**A**: Go to the *Database Configuration* dialog and uncheck the **Automatically Create** option.

### SYBASE

**Q**: I configured my Sybase database using the Browse button. When I click the test button, the test succeeds, but when I try to run my project I get the following error: **Database: Error: Parse error: DSN 'MyDatabase' does not exist**. What am I doing wrong?

**A**: Please refer to Database Appendix F - Using Sybase for more information about this problem.

**Q**: Why, when I try to connect to the Sybase database, am I receiving the error **Error to create connection class [iAnywhere.Data.AsaClient.AsaConnection]**?

**A**: You do not have the ADO.NET Provider installed on your computer. The database setup program has an option to install the Provider. Rerun the setup program, and make sure to check that option.

### SQL SERVER CE

**Q**: Why does the gateway show **TypeLoad failure** when I try to access my SQL Server CE database?

**A**: This problem usually happens when you do not have the SQL Server CE .NET Provider installed on your CE Device.

**Q**: Why am I getting the error message, **There is a file sharing violation. A different process might be using the file**?

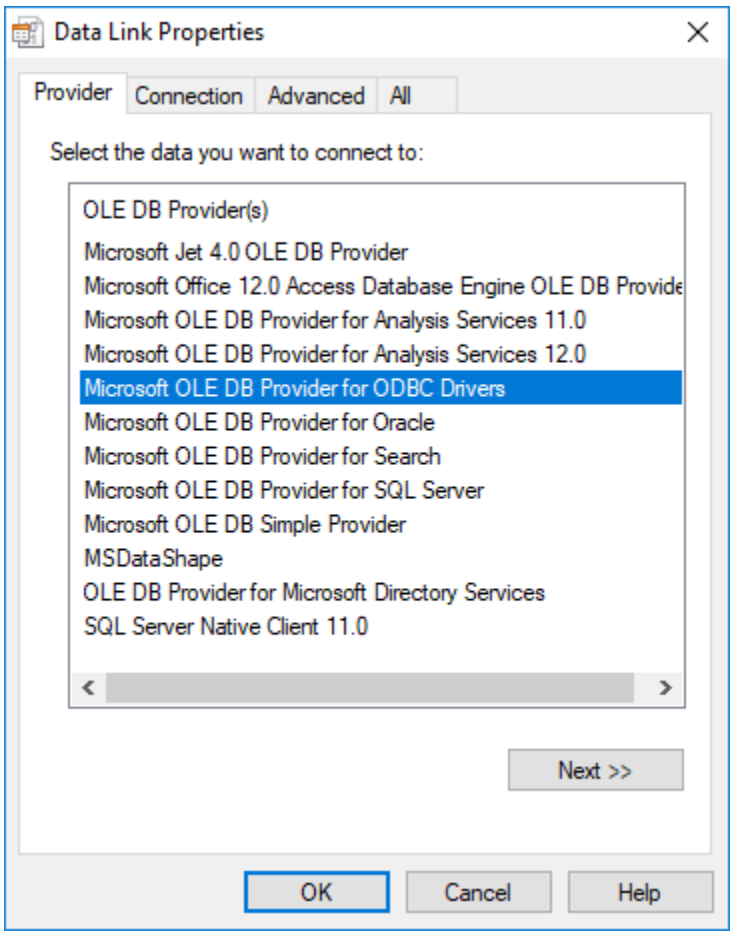
**A**: You have another progarm with the SQL Server CE database open. For instance, this will happen if you are using the SQL Server CE configuration software.

# Appendices

## *Using ODBC Databases*

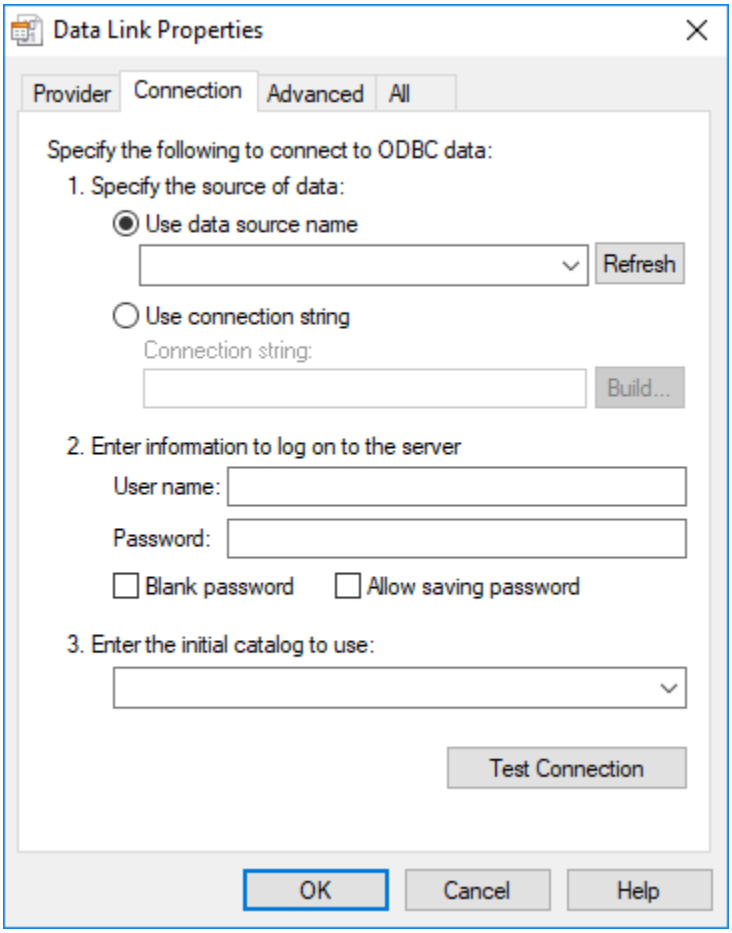
Almost every database provides an ODBC interface that can be used to interface with it. The database features provided by EcoStruxure Machine SCADA Expert can be used with ODBC drivers through the ADO.NET interface for ODBC. In order to use this capability, you must use Microsoft .NET Framework 1.1 or higher.

The Database Configuration dialog allows you to provide connection strings that will connect to an ODBC DSN. The connection string can be built automatically by clicking on the Browse button (...). When the *Data Link* window displays, you should select the option Microsoft OLE DB Provider for ODBC Drivers as shown below:




***Data Link Properties, Provider - ODBC***

When you click **Next**, the following window will display:



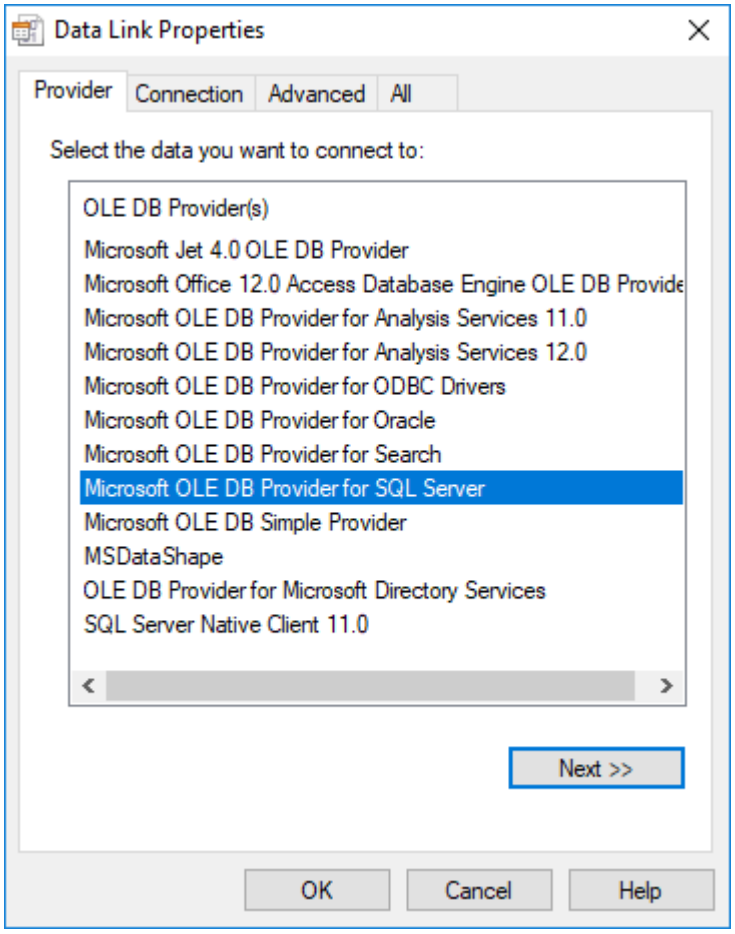

***Data Link Properties, Provider - Connection***

Select the DSN that you want to connect to and click **OK**. If you want to specify the user name and password on this window instead of specifying on the *Object Properties* dialog, remember to check the **Allow saving password** checkbox.

## *Using Microsoft SQL Server*

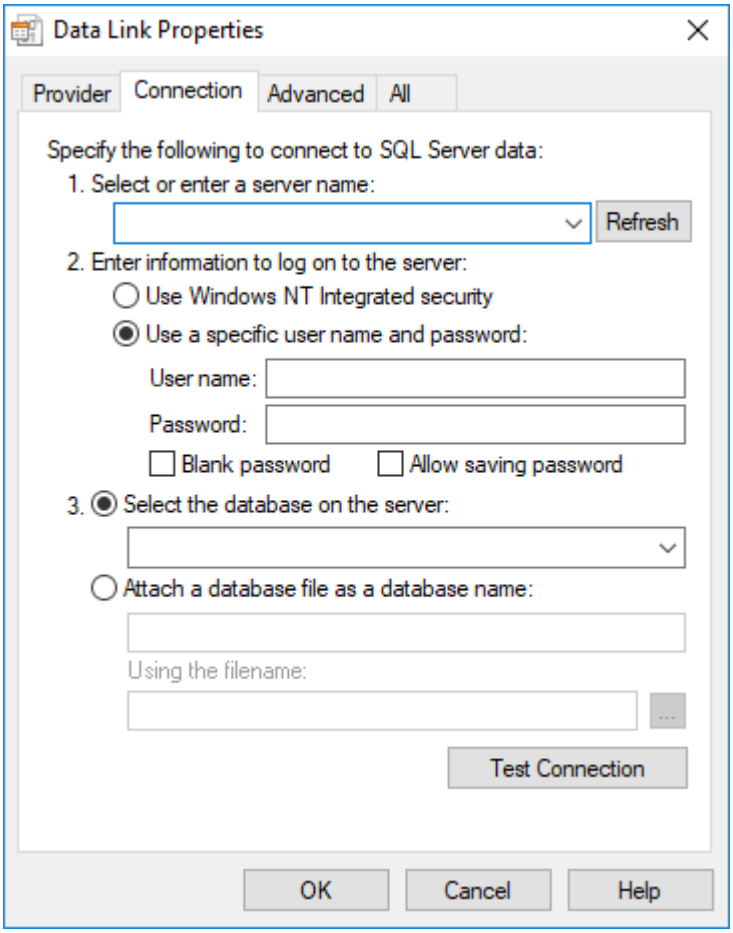
EcoStruxure Machine SCADA Expert Database Interface allows you to retrieve and store information on Microsoft SQL Server relational databases. You should follow the steps below in order to configure the SQL Server database:

1. Click on the **Browse** button in the Database Configuration Dialog window. The following window will display:




***Data Link Properties, Provider - SQL Server***

2. Select the Microsoft OLE Provider for SQL Server and click **Next**. The following window will display:




***Data Link Properties, Connection - SQL Server***

3. Fill out the fields on this window with your database information. If you are not using Windows NT Integrated security, remember to check the **Allow saving password** checkbox to save the password when the *Data Link Properties* window is closed.
4. Click **OK** to finish the *Connection String* configuration.

Your connection string should be very similar to this one:

```
Provider=SQLOLEDB.1; Integrated Security=SSPI; Initial Catalog=MyDatabase; Data
 Source=192.168.23.200
```

> **Note:** These procedures were tested using Microsoft SQL Server 2000.

> **Tip:** The Database Gateway (`StADOSvr.exe`) now uses an updated time stamp when saving data to Microsoft SQL Server databases, so that milliseconds do not need to be stored in a separate column. However, this only works with Microsoft SQL Server 2008 or later, so if you are using an earlier version of SQL Server, then you must edit the program settings to reverse the change:
>
> 1. If your project is running, stop it.
> 2. Locate the program settings file at: `C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin\StADOSvr.ini`
> 3. Open the file with a text editor, such as Notepad.

4. Delete the following line:

   ```
   TimeStampType2=DATETIME2
   ```

5. Save your changes, and then exit the text editor.

## Using Oracle Databases

Use the *Data Link Properties* dialog box to compose a connection string that will enable your project to connect to an Oracle database instance via OLE DB.

This task is a supplement to other tasks that describe how to configure your project to connect to external databases; it assumes you have already opened the *Data Link Properties* dialog box, which is a standard Windows system interface, and it starts from that point. The *Data Link Properties* dialog box is invoked by EcoStruxure Machine SCADA Expert 2020 — for example, from the *Database Configuration* dialog box — but it is not actually a part of EcoStruxure Machine SCADA Expert 2020. For a more comprehensive description of the *Data Link Properties* dialog box, go to: msdn.microsoft.com/en-us/library/79t8s5dk.aspx

Before you begin this task, you should download and install the latest version of 32-bit Oracle Data Access Components (ODAC) for Windows, on both your project development workstation and the computer(s) that will host the project runtime / database gateway. (EcoStruxure Machine SCADA Expert 2020 is a 32-bit application and therefore requires 32-bit providers.) To download the software, go to: www.oracle.com/technetwork/database/windows/downloads/utilsoft-087491.html

Most versions of Windows include **Microsoft OLE DB Provider for Oracle**, which is the provider that we recommended in the past. More recently, however, we have observed issues when we use that provider to try to connect to the latest version of Oracle, so we have determined it is better to use the providers that are made available by Oracle itself. The procedure below reflects that.

> **Note:** This task was last tested using Oracle 10g Release 1.

To compose a connection string for an Oracle database instance:

1. In the **Provider** tab of the *Data Link Properties* dialog box, in the **OLE DB Provider(s)** list, select **Oracle Provider for OLE DB**.
2. Click **Next**.
   The **Connection** tab of the *Data Link Properties* dialog box is displayed.
3. In the **Data Source** box, type the name of the data source using the following syntax:

   ```
   <hostname or IP address>/<Oracle TNS name>
   ```

   ***<hostname or IP address>***
   The hostname or IP address of the computer that hosts the Oracle database server.

   ***<Oracle TNS name>***
   The Transparent Network Substrate (TNS) name for the specific database instance. This name references an entry in the server's `tnsnames.ora` file, which contains additional information about the database instances.

   Example: `127.0.0.1/MyDatabase`
4. In the **User name** and **Password** boxes, type your credentials for the database instance.
5. Select the **Allow saving password** option.
6. Click **Test Connection**.
   A message is displayed to inform you if the test connection succeeded. If it did not, review your settings and then try again.
7. Click **OK** to close the *Data Link Properties* dialog box.

The connection string, user name, and password are automatically pasted into the database configuration. An example of the connection string is shown below:

```
Provider=OraOLEDB.Oracle.1; Data Source=127.0.0.1/MyDatabase
```

### *Using Microsoft Access or Microsoft Excel*

Use the *Data Link Properties* dialog box to compose a connection string that will enable your project to connect to a Microsoft Access database or Microsoft Excel workbook via OLE DB.

This task is a supplement to other tasks that describe how to configure your project to connect to external databases; it assumes you have already opened the *Data Link Properties* dialog box, which is a standard Windows system interface, and it starts from that point. The *Data Link Properties* dialog box is invoked by EcoStruxure Machine SCADA Expert 2020 — for example, from the *Database Configuration* dialog box — but it is not actually a part of EcoStruxure Machine SCADA Expert 2020. For a more comprehensive description of the *Data Link Properties* dialog box, go to: msdn.microsoft.com/en-us/library/79t8s5dk.aspx

You do not need to have Microsoft Office installed on your computer in order to use Microsoft Access and Microsoft Excel files with EcoStruxure Machine SCADA Expert 2020; your project can directly read from and write to existing files, as long as the correct OLE DB provider is installed on both your project development workstation and the computer(s) that will host the project runtime / database gateway.

You might need to install the 32-bit version of the OLE DB provider, however, because EcoStruxure Machine SCADA Expert 2020 is a 32-bit application and therefore requires 32-bit providers. Whether you need to do this depends on which versions of Microsoft Windows and Microsoft Office you already have installed, and there are too many possible combinations to cover in this documentation, so the only way for you to know for sure is if you can select the correct provider (i.e., Microsoft Office 12.0 Access Database Engine OLE DB Provider) as described in the first step of the procedure below. If you cannot, you need to install the provider.

In most cases, we recommend you download and install Microsoft Access Database Engine 2010 Redistributable (www.microsoft.com/en-US/download/details.aspx?id=13255). It is not the latest version of that software, but it includes the correct OLE DB provider and it can be installed on top of the latest version of Microsoft Office, if necessary. Make sure you download the 32-bit version (a.k.a. X86), not the 64-bit version (a.k.a. X64).

> **Note:** This task was last tested using Microsoft Office 2016.

To compose a connection string for a Microsoft Access database or Microsoft Excel workbook:

1. In the **Provider** tab of the *Data Link Properties* dialog box, in the **OLE DB Provider(s)** list, select **Microsoft Office 12.0 Access Database Engine OLE DB Provider**.
2. Click **Next**.
   The **Connection** tab of the *Data Link Properties* dialog box is displayed.
3. In the **Data Source** box, type the absolute file path and name of the Microsoft Access database (.accdb) or Microsoft Excel workbook (.xls or .xlsx).

   The data source file should be located somewhere that can be accessed by both your project development workstation and the computer(s) that will host the project runtime / database gateway. If the file is located in your project folder, you might need to correct the file path when you download your project folder to another computer.
4. If access to the data source file is restricted, so that you need a user name and/or password in order to use it, do the following:
   a) In the **User name** and **Password** boxes, type your credentials for the data source file.

      You might need to clear the **Blank password** option before you can type the password.

   b) Select the **Allow saving password** option.
5. If the data source file is a Microsoft Excel workbook (.xls or .xlsx), do the following:
   a) Click the **All** tab of the *Data Link Properties* dialog box.

   b) In the list of properties, double-click **Extended Properties**.
      An *Edit Property Value* dialog box is displayed.

c) In the **Property Value** box, type one of the following:

| File Extension | Property Value |
|---|---|
| .xls | `Excel 12.0; Hdr=Yes` |
| .xlsx | `Excel 12.0 Xml; Hdr=Yes` |

The first part of the property value indicates that the data source file is a Microsoft Excel workbook, as opposed to a Microsoft Access database. The second part of the property value indicates that the data source includes a header row with appropriate column names.

d) Click **OK** to save the property value and close the *Edit Property Value* dialog box.

e) Click the **Connection** tab of the *Data Link Properties* dialog box.

6. Click **Test Connection**.
A message is displayed to inform you if the test connection succeeded. If it did not, review your settings and then try again.

7. Click **OK** to close the *Data Link Properties* dialog box.

The connection string, user name, and password are automatically pasted into the database configuration. Examples of the connection string are shown below:

```
Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\MyDatabase.accdb
```

```
Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\MyWorkbook.xls; Extended
 Properties="Excel 12.0; Hdr=Yes"
```

```
Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\MyWorkbook.xlsx; Extended
 Properties="Excel 12.0 Xml; Hdr=Yes"
```

You can manually edit the connection string to replace any part of it with a tag/expression enclosed in curly brackets, so that part can be programmatically changed during project run time. This is typically done when the data source file is located in the project folder — if you replace the absolute file path with an expression that gets the file path of the project folder, you can ensure the connection string will work when you download the project to another computer or device. For example:

```
Provider=Microsoft.ACE.OLEDB.12.0; Data Source={GetAppPath()}\MyWorkbook.xlsx; Extended
 Properties="Excel 12.0 Xml; Hdr=Yes"
```

As you finish the database configuration, there are a few other things you might need to do.

First, in the database configuration's advanced settings, select the **Disable Primary Keys** option. If you do not, your project will not be able to connect to the data source. For more information, see Database Configuration on page 108.

Second, if the data source file is a Microsoft Excel workbook, be aware that the sheets of the workbook are equivalent to tables in a database, and when you need to specify a table in your database configuration, you actually need to specify a sheet. The name of the sheet should be formatted as follows:

```
[<sheet name>$]
```

Third, make sure the specified sheet includes a header row with appropriate column names, because you will need to reference those column names when you use the specified sheet with your project.

More generally, please keep in mind that Microsoft Access and Microsoft Excel are desktop office applications, and they cannot efficiently handle large amounts of data. If you try to save all of your project's historical data in a Microsoft Access database or Microsoft Excel workbook, the queries will become slower over time and you might get unexpected results. Therefore, we recommend you use such a data source only as a **Secondary Database**, with the **Store and Forward** option selected.

To handle large amounts of historical data, we recommend you use either EcoStruxure Machine SCADA Expert 2020's proprietary format or a dedicated relational database such as Microsoft SQL Server or Oracle.

### Using Sybase

You need to install the AsaClient provider on your computer; the tests with EcoStruxure Machine SCADA Expert were performed using the architecture explained in the topic Manually install Studio Database Gateway on page 826.

If you are using the browse button to automatically generate the connection string, the string returned will have the following format:

```
Provider=ASAProv.90; Data Source=Test
```

This format requires that you create an ODBC DSN with the same name as the **Data Source** (in this case, *Test*) in order to communicate with the database. If the DSN is not created, the following error will display in the LogWin when connecting to the database:

```
Database: Error: Parse error:  DSN 'Test' does not exist
```

To void an ODBC DSN, you can enter with the connection string manually as shown in the example below:

```
Provider=ASAProv.90; DBF=C:\ Test.db
```

> **Note:** These procedures were tested using Sybase Server Anywhere 9.0.1.1751.

### Using MySQL

EcoStruxure Machine SCADA Expert can interface with MySQL databases, but to do so, you must install an ADO.Net provider for MySQL.

The provider required by EcoStruxure Machine SCADA Expert is MySQL Connector/Net, and at the time of this writing, the necessary software can be downloaded from the official MySQL site. (Please note that the linked site is beyond our control and may change without notice.)

Once the provider is installed, you can use the *Database Configuration* property sheet to configure a MySQL database connection. However, unlike for other database types, you cannot use the *Data Link Properties* dialog (which is accessed by clicking ... to the right of the **Connection string** box) to form the connection string. Instead, you must directly enter the connection string using this basic format:

```
Provider=MYSQLCLIENT; Server=myServerAddress; Database=myDataBase; Uid=myUsername;
 Pwd=myPassword;
```

The following optional parameters can be appended to the connection string:

**Optional parameters for the MySQL Connector/Net connection string**

| Parameter | Description |
|---|---|
| `Port=number;` | Specifies what port to use for the connection. The default port is 3306, but any port can be specified as long as it matches the server configuration.<br>If a port of -1 is specified, then the connection will use the named pipes network protocol (see `Protocol` below). |
| `Server=myServerAddress1 & myServerAddress2 & … & myServerAddressN;` | Use any server in a replicated server configuration. |
| `Encryption=true;` | Enables SSL encryption for all data sent between the client and the server. The server must have a valid certificate installed. |
| `Encrypt=true;` | An alternative to `Encryption` above, in case there are errors. |

| Parameter | Description |
|---|---|
| `Default Command Timeout=`*`milliseconds`*`;` | Specifies a default command timeout for the connection. This does not supercede any timeout properties on individual commands. |
| `Connection Timeout=`*`seconds`*`;` | Specifies how long the client will wait for a server connection before terminating the attempt. |
| `Ignore Prepare=true;` | Instructs the database provider to ignore **`Command.Prepare`** statements, to prevent corruption from server-side prepared commands. |
| `Protocol=`*`myProtocol`*`;` | Specifies which network protocol to use. The default is `socket` or `tcp`, but you can specify `pipe` to use a named pipes connection or `memory` to use a shared memory object. |
| `Shared Memory Name=MySQL;` | Specifies the name of the shared memory object to be used for communication. (This parameter applies only if the `Protocol` parameter above is set to `memory`.) |
| `CharSet=UTF8;` | Specifies which character set to use to encode queries to the server.<br>Please note that query results are encoded in the same character set that the data itself is recorded. |

**Note:** These procedures were tested using MySQL v5.1.11 and MySQL Connector/Net v6.2.0.

# Troubleshooting

## General Troubleshooting

If you do find yourself in need of technical assistance, there are certain things that you will need to know before you contact technical support. Regardless of the problem, you will need to know the sequence of events that led to you discovering the problem. It must be explained in as much detail as possible and you should be careful not to ad-lib, as it may drastically affect troubleshooting time and procedures. It's also best to be in front of the computer you are having problems with, and to keep a pen and paper handy.

### Before Contacting Technical Support

Some things you should try before you contact technical support are:

- **Check out the documentation**

  The application help and release notes can be accessed on the Help tab of the ribbon, and more documentation is available on our website. You may find that your particular issue has already been documented.

- **Consider recent changes on your system**

  If something used to work, think about what may have changed. New software installation or general system changes can affect performance and general functionality of other software on your system.

- **Try reproducing the problem in a new file**

  If the problem can not be reproduced in a new test file, compare the new file with your original file to find and eliminate the differences. This will help narrow down the cause of the issue.

- **Try reproducing the problem on another machine**

  If the problem goes away on another machine, compare what is different between the two systems. If this is the case, there is most likely a system conflict.

### Verifying Your Project

If you change, reorder or delete any tags in the Tags database, or if you reconfigure any settings in the **Web** tab of the *Project Settings* dialogue, then you must verify your project to realign all of your screens and worksheets to the current state of your database. On the Home tab of the ribbon, in the Tools group, click **Verify**.

### Related Documentation

The *EcoStruxure Machine SCADA Expert 2020 Quick Start Guide* is designed for first-time users. This guide contains information about the basic functions of EcoStruxure Machine SCADA Expert 2020, and it is provided in the **Documentation** folder on the EcoStruxure Machine SCADA Expert 2020 installation CD.

The communication driver user guides explain how to configure the direct communication drivers, according to their individual specifications. One customized user guide is included with each driver. These guides are provided in the drivers sub-folder of the EcoStruxure Machine SCADA Expert 2020 program folder (`Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Drv`), or they can be opened from the **Help** tab of the ribbon. (On the **Help** tab of the ribbon, in the **Documentation** group, click **Communication Drivers**. When the list of drivers is displayed, select the driver that you are using and then click **Help**.)

### Contacting Technical Support

If you cannot find an answer to your technical question in the product documentation or help system, our Technical Support Specialists are available to assist any customer with current product maintenance. The telephone number is 1-877-342-5173.

Please try to define the problem before you contact Technical Support so that you can repeat the steps that led to the problem and specifically identify when and how the problem occurred. The support representative will need to know exactly what the problem is in order to provide help. These steps will help us pinpoint and solve your problem more quickly.

Please have the following information available:

- Hardware environment: available memory, processor type, output device
- Software environment: operating system, version of Windows®, network platform
- Product name, version number, and product registration number

- Amount of memory installed on your system
- Amount of free hard disk space on your system
- Screen resolution (screen size in pixels, for example, 1024 by 768)
- Screen color depth (number of colors or bits, for example, 256 colors or 8-bit color)
- Graphics card manufacturer, model name, and driver version number
- Sound card manufacturer and model name
- A list of external devices connected to the computer
- Brief description of the problem or error, and the specific text of any error messages
- Description of the steps you have taken to troubleshoot the issue, for example, how many machines you have tested on, and whether the issue is reproducible in a new file
- Steps to reproduce the issue, if it is reproducible. If the issue is not reproducible, it may be an development issue rather than an issue with the product.

If your project crashes completely during runtime, it will generate a debugger report and save it to:

**EcoStruxure Machine SCADA Expert 2020 Projects\<*project name*>\Web\Dump\WindDump.dmp**

Please have this file ready to send to Technical Support for analysis.

When you contact us, please have your system information ready. You can get this by using the **Support Information** command located in the **Help** menu.

If your problem or question is not urgent, please go to our website at: www.se.com

# Frequently Asked Questions

## Database & Security System

**What does the Shared Tags folder store?**

The Shared Tags folder stores the tags imported from the PC-based Control linked to the EcoStruxure Machine SCADA Expert project. The PC-based Control is linked to the project by the *New Project* wizard.

**How do I count how many tags are configured in the project database?**

The number of tags currently used in the project is displayed in the status bar at the bottom of the development environment. Individual array elements and class members are counted as tags.

## Graphics

**How do I insert and configure an ActiveX object in a project?**

To insert an ActiveX object in a project screen:

1. On the Graphics tab of the ribbon, in the Libraries group, click **ActiveX Control**.
2. Select the ActiveX control that you want to insert from the list, and then click **OK**. The ActiveX object will then appear on the screen. (Unregistered ActiveX objects will not be available in this list box.)
3. Double-click on the ActiveX object and assign a name to it (enter a value in the **Name** field). The animations and methods list can be viewed by selecting the **Methods** button. The static properties can be set by the **Properties** button (A detailed description about the objects properties can be found in the component documentation, provided by the component developer).

There are three functions to access the ActiveX component during runtime:

- **`XGet(strName,strProperties)`**: Returns the value of the properties ***`strProperties`*** from the object ***`strName`***. The list of properties which can be read from the object are listed in the *Object Properties* dialog from the object, with the syntax ***`Properties Name`*****`(PropGet)`** (for example, **`Color(PropGet)`** ).
- **`XSet(strName,strProperties,Value)`**: Writes the value ***`Value`*** to the properties ***`strProperties`*** of the object ***`strName`***. The list of properties which can be set to the object are listed in the *Object Properties* dialog from the object, with the syntax ***`Properties Name`*****`(PropPut)`** (for example, **`Color(PropPut)`** ).
- **`XRun(strName, strMethod, Parameter1, Parameter2, …, ParameterN)`**: Executes the method ***`strMethod`*** from the object ***`strName`***, according to the parameters ***`Parameter1`***, ***`Parameter2`***, …, ***`ParameterN`***. The list of methods available in the object is listed in the *Object Properties* dialog from the object, with the syntax ***`Method Name`*****`(Method)`** (for example, **`OpenFile(Method)`** ).

> **Tip:** Before inserting an ActiveX control (usually an OCX file) into the project, make sure it has been properly registered in the computer. To register an ActiveX control from with the development application, click **Register Controls** on the Home tab of the ribbon.

> **Note:** The amount of parameters set in the `XRun` function can vary from 0 up to 255 and it depends each the ActiveX component. It's possible to use tags to set the parameters; however, the tag type must match the component parameter type (Boolean, integer, string or real).

**How do I designate one screen that will open each time I start the project?**

On the Project tab of the ribbon, in the Settings group, click **Viewer**, and then in the *Project Settings* dialog, type the startup screen name in the **Startup screen** box.

**How do I insert a background picture on the screen?**

Right click on the screen and select the option **Screen Attributes** from the popup menu. Enable the checkbox **Enable Background** and choose the picture format in the combo-box besides this label. Copy the picture file to the `Screen` sub-folder of the project and rename it with the same name of the screen (***ScreenName*.scc** file). Using the **Shared image** option, it's possible to copy a bitmap file to the `Screen` sub-folder and share this picture with more than one screen. In this case, it's necessary to type the bitmap name in the **Shared image** field.

## Tasks

**How do I convert a trend history from the proprietary binary file to an ASCII text file?**

In Windows, select the history file (`*.HST`) that you want to convert, and then drag it to the HST2TXT utility in the program folder (`Schneider Electric\EcoStruxure Machine SCADA Expert 2020\Bin\HST2TXT.exe`). The converted file will be automatically created in that folder.

Alternatively, you can use the `HST2TXT` function in a Script or Math worksheet to convert files automatically during project run time.

**How do I set a DATE field for an ODBC interface with an Oracle package?**

Configure the "Column" cells in the ODBC worksheet with the syntax ***ColumnName*.ts** (for example: **MyDate.ts** ).

**How do I execute a Math worksheet during the startup and another Math worksheet during the project shutdown?**

- **Startup**: Execute a *Math* worksheet during the startup by creating a *Math* worksheet and filling in its **Execution** field with the expression **<TagName>=0** (for example, **StartTag=0** ). In the last line of the *Math* worksheet, set the value **1** to the **<TagName>** tag. The **<TagName>**tag type should be **Boolean**.
- **Shutdown**: Instead of executing the **ShutDown()** function directly, execute one *Math* worksheet and configure the **ShutDown()** function in the last line of this *Math* worksheet.

## Communication

**How do I set a "communication error" alarm?**

Configure a tag in the **Write Status** or **Read Status** field of the driver worksheets and configure an alarm whenever this tag is not **0** (zero).

**How do I start and stop communication drivers during project run time?**

There are three functions you can use to manage the execution of the communication drivers:

- Use the `StartTask` function to start the Driver task itself, which in turn starts all of the drivers that have been added to the project:

```
StartTask("Driver")
```

- Use the `Exec` function to start a single, specific driver that has been added to the project. The command itself should use the following syntax:

```
"<runtime program folder>\Bin\Studio Manager.exe" "<runtime program
 folder>\Bin\Driver.dll" <driver name>
```

> **Note:** When executing a command-line command, file paths that contain spaces must be enclosed in quotes. However, unpaired quotes can interfere with the concatenation of strings within a function call, so in the examples below, each quote character is replaced by **Asc2Str(34)**. The quote character

has ASCII code 34, and the `Asc` function converts that code into the actual character.

Then, call the `WinExec` function to compose and execute the command during project run time. For example, to start the MODBU driver:

```
WinExec(Asc2Str(34) + "C:\Program Files (x86)\Schneider
 Electric\EcoStruxure Machine SCADA Expert 2020\Bin\Studio Manager.exe"
 + Asc2Str(34) + " " + Asc2Str(34) + "C:\Program Files (x86)\Schneider
 Electric\EcoStruxure Machine SCADA Expert 2020\Bin\Driver.dll" +
 Asc2Str(34) + " " + "MODBU")
```

Alternatively, you can call the `GetProductPath` function to get the location of the runtime program folder and then reference a project tag for the driver name:

```
WinExec(Asc2Str(34) + GetProductPath() + "Bin\Studio Manager.exe" +
 Asc2Str(34) + " " + Asc2Str(34) + GetProductPath() + "Bin\Driver.dll" +
 Asc2Str(34) + " " + MyDriver)
```

- Use the `EndTask` function to stop a single, specific driver that has been added to the project:

```
EndTask("Driver<driver name>")
```

For example, to stop the MODBU driver:

```
EndTask("DriverMODBU")
```

Alternatively, you can reference a project tag for the driver name:

```
EndTask("Driver" + MyDriver)
```

**Is the EcoStruxure Machine SCADA Expert OPC interface compliant with OPC specification v1.0a or v2.0?**

The EcoStruxure Machine SCADA Expert OPC Client and OPC Server modules are compliant with both OPC specification v1.0a and v2.0.

**How do I get errors from Intellution / GE Fanuc iFIX applications?**

If your project is communicating via TCP/IP with an iFIX application, then you should add the following key to your project file (i.e., ***<project name>*.APP**):

```
[TCP]
SetQualityToBadOnError=1
```

After you do this, if the iFIX application generates an error during runtime, then the quality of the affected tags in your project will be set to BAD. You can get this information by reading the Quality tag field (i.e., ***tagname*->Quality**).

## General

**What operating systems are compatible with EcoStruxure Machine SCADA Expert 2020?**

See About the EcoStruxure Machine SCADA Expert 2020 software components on page 34.

**How do I automatically run my project when the computer or device is turned on?**

If you are using the full SCADA software as your project runtime, you can use the RunStartUp utility (`RunStartUp.exe`) that is included with the software. Assuming the software was installed at its default location, the utility should be located at:

```
C:\Program Files (x86)\Schneider Electric\EcoStruxure Machine SCADA Expert
 2020\Bin\RunStartUp.exe
```

When you run this utility, it starts the project runtime and then runs the last opened project. Therefore, to automatically run the project when the computer is turned on, create a shortcut to the utility and then add the shortcut to the Startup folder in Windows.

Alternatively, you can configure the project to run as a Windows service. For more information, see Run a project as a Windows service on page 138.

**What are the main steps to create a Web-based application?**

Follow the procedure below:

1. Develop the project locally. Don't use features that are not supported by Thin Clients for the screens which will be saved as HTML format.
2. After saving the screens in the standard format (**Save** in the File menu), save the screens that must be available for the Thin Client in HTML format (**Save as HTML** in the File menu).
3. On the Project tab of the ribbon, in the Web group, click **Thin Client**.
4. In the **Data Server IP** box, type the IP address of the Server station (i.e., the station where the project is running).
5. On the Home tab of the ribbon, click **Tasks**, and then set the TCP/IP Server module as **Startup=Automatic**.
6. Open the project tags database and set the option **Server** instead of **Local** in the **Scope** column for all tags that must exchange value between the Server and the Thin Client station.
7. Verify the project (**Verify** on the Home tab of the ribbon) to update the HTML files with these new settings.
8. If there is no Web Server running on the computer, copy the program **`NTWebServer.exe`** from the `Bin` sub-folder of the program directory into the Web directory (e.g., `\project name\Web`) and run it. The path where this Web Server program is executed will be the root directory of the server. The Web Server is necessary to export data (web files) in HTTP protocol to the Thin Clients.
9. Run the project on the Server station.
10. Using Internet Explorer on the client station, type the URL address to download the screen that had been saved in HTML format (for example, **`http://ServerIPAddress/ScreenName.html`**).

> **Note:** The Thin Client requires an ActiveX component ( **`ISSymbol.ocx`** ) to handle the screens on the browser. If the Thin Client is connected to the Internet, this component is downloaded and registered automatically. Otherwise, it's necessary to copy it to the **`\OSPath\System32`** directory of the Thin Client and register it by the command **`regsvr32 ISSymbol.ocx`**. This file can be found in the `\BIN` folder from the EcoStruxure Machine SCADA Expert installation directory.

**How do I maintain communication between a Thin Client connecting via proxy and a Web Gateway application running on Microsoft IIS?**

Microsoft Internet Information Services (IIS) has a configuration option to keep HTTP connections alive. When this option is enabled, it may conflict with Thin Clients that are connecting via proxy. To disable this option:

1. Start Internet Services Manager.

2. In the *Internet Information Services* window, open the local server ( **** server name*** ).
3. Right-click on **Default Web Sites** and select **Properties** from the shortcut menu. The *Default Web Site Properties* dialog is displayed.
4. Select the **Web Site** tab of the *Default Web Site Properties* dialog.
5. In the *Connections* pane of the **Web Site** tab, uncheck the **HTTP Keep-Alives Enabled** option.
6. Click **OK** to save the change and close the dialog.

**How do I send an email from the EcoStruxure Machine SCADA Expert project?**

Follow the procedure below:

- Execute the function **`CNFEMail(strSMTP,strFrom,strPOP3,strUser, strPassword,numTimeOut)`** to configure the overall parameters used to send emails. After executing this function once, the parameters set by it are kept in the system until the project is shut down. So, most projects execute this function just once, after starting the project;
- Execute the function **`SendEMail(strSubject,strMessage,strTO) and/ or SendEMailExt(strSubject,strMessage,strTO,strCC,strBCC, strFile1,...,strFileN)`** each time that an email message must be sent. The main difference between both functions are listed in the next table:

| Characteristic | SendEmail | SendEmailExt |
|---|---|---|
| Execution | Synchronous | Asynchronous |
| Supports Subject text | Y | Y |
| Supports Message text | Y | Y |
| Supports TO addresses | Y | Y |
| Supports CC addresses | N | Y |
| Supports BCC addresses | N | Y |
| Supports attached files | N | Y |

**The runtime task (e.g., TCP/IP, OPC) does not work.**

Make sure the runtime task is set to **Automatic** in the *Runtime Tasks* dialog (**Tasks** on the Home tab of the ribbon). Select a runtime task that must be executed (for example, TCP/IP Server), click **Startup**, and then set it as **Automatic**.

**The browser of the Thin Client launches an error message missing the `ISSymbol.ocx` and does not display the screens from the Server.**

**`ISSymbol.ocx`** is the ActiveX object used by the browser from the Thin Client to view the web pages. If the Thin Client is connected to the Internet, the **`ISSymbol.ocx`** control is automatically downloaded and registered in the Thin Client station. Otherwise, it's necessary to copy it to the **`\WinNT\System32`** folder of the Thin Client station and register it manually. Once it is registered your browser will be able to see the pages.

> **Note:** Use the command **`regsvr32 ISSymbol32.ocx`** to register the ActiveX component in the Thin Client.

**The screens are shown on the Thin Client (Browser); however, the data (tags values) are not read from the Server.**

Make sure the parameter in the column **Scope** from the project tags database is set as **Server** instead of **Local**. The tags set as **Server** keep the same value in the Server and in the Thin Client (Browser). The tags set as **Local** have independent values in the Server and in the Thin Client (Browser).

> **Note:** It's necessary to verify the project (**Verify** on the Home tab of the ribbon) after modifying the tags settings. Otherwise, the changes will not be updated in the Web pages.

**The "On Up" expressions configured in the Command animation are not executed.**

The "On Up" expressions from the Command animation are not executed if the mouse pointer is dragged out the object area before releasing it. If the checkbox **Release from the Command Object Properties** window is enabled, the On Up expression is executed even if the mouse pointer is dragged out the object area before releasing it.

**The Trend History does not work after adding or removing tags in the Trend worksheet.**

When a tag is inserted or removed FROM a *Trend* worksheet, the format of the history files ( **`*.hst`** ) is modified. The same **`.hst`** file cannot have two different formats; otherwise, the data will not be retrieved from it properly by the Trend object. If you need to add or remove tags for history files, there are two valid procedures: Create a new *Trend* worksheet or delete the old **`*.hst`** files.

## Proxy Settings

If your project can communicate with other stations on your local network (e.g., PLCs, thin clients, external databases) but it cannot access the Internet, you might need to configure proxy settings on the computer or device that hosts the project runtime.

Some features of the project runtime use the command-line interface (CLI) to activate third-party tools, and some of those tools connect to other services over the Internet, so the CLI itself often needs to be able to access the Internet. If your local network uses a network gateway or proxy server to control access, however, you might need to configure the proxy settings for the CLI — specifically, you need to set environment variables for the protocols used by the CLI to access the Internet.

Please note the CLI has its own proxy settings separate from other applications on the same computer. Even if you previously configured the proxy settings for the web browser, for example, you still need to configure the proxy settings for the CLI.

Of course, you should do this only if your project actually needs to access the Internet — in other words, if you have developed your project to take advantage of those features that access the Internet but it does not behave as expected during project run time. Otherwise, you should leave the proxy settings unchanged in order to maintain network security and privacy.

For more information, consult your network administrator.

### *Configure the proxy settings on a Windows computer or device*

Configure the proxy settings for the command-line interface (CLI) on a Windows computer or device — specifically, set the environment variables used by Command Prompt.

Before you begin this task, you should know the host name or IP address of your network gateway or proxy server, as well as the port number (if any) of the proxy service. If you do not know, ask your network administrator.

If the proxy service requires authentication, you should also know your user name and password for that service.

You do not need Administrator privileges on the Windows computer or device in order to perform this task, as long as you are logged on as the same user that runs the project runtime.

To configure the proxy settings:

1. Make sure you are logged on as the same user that runs the project runtime.
2. Open the *Environment Variables* control panel.
3. In the control panel, do the following:
   a) In the **User variables** area of the control panel, click **New**.
   A *New User Variable* dialog box is displayed.
   b) In the **Variable name** box, type `http_proxy`.
   c) In the **Variable value** box, type the URL of the proxy service. Remember to include the port number of the service and/or your user name and password for that service, if necessary.

      Examples of the URL:

      **`http://proxy-server`**

      **`http://proxy-server.mynetwork.com`**

      **`http://gateway.mynetwork.com:3128`**

      **`http://username:password@gateway.mynetwork.com:3128`**

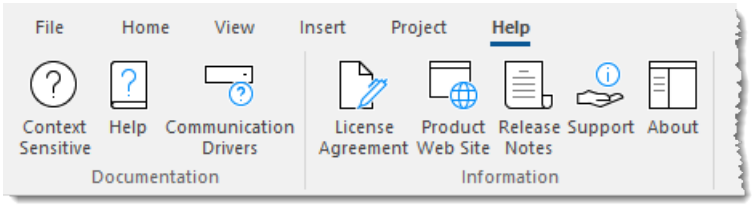
   d) Click **OK**.
   The new variable is saved.

4. Repeat the previous step for the variables `https_proxy` and `ftp_proxy`.
5. Click **OK** to close the *Environment Variables* control panel.

Once these environment variables are set, they will be available whenever a new *Command Prompt* window is opened.

## Help tab

The **Help** tab of the ribbon provides additional help with using the software.




The tools are organized into the following groups:

- **Documentation**: Access the documentation for the development application, including this help file / technical reference and notes for the individual communication drivers.
- **Information**: Access other information about EcoStruxure Machine SCADA Expert 2020, including the license agreement, product website, and release notes, as well as support details that make it easier for us to assist you.

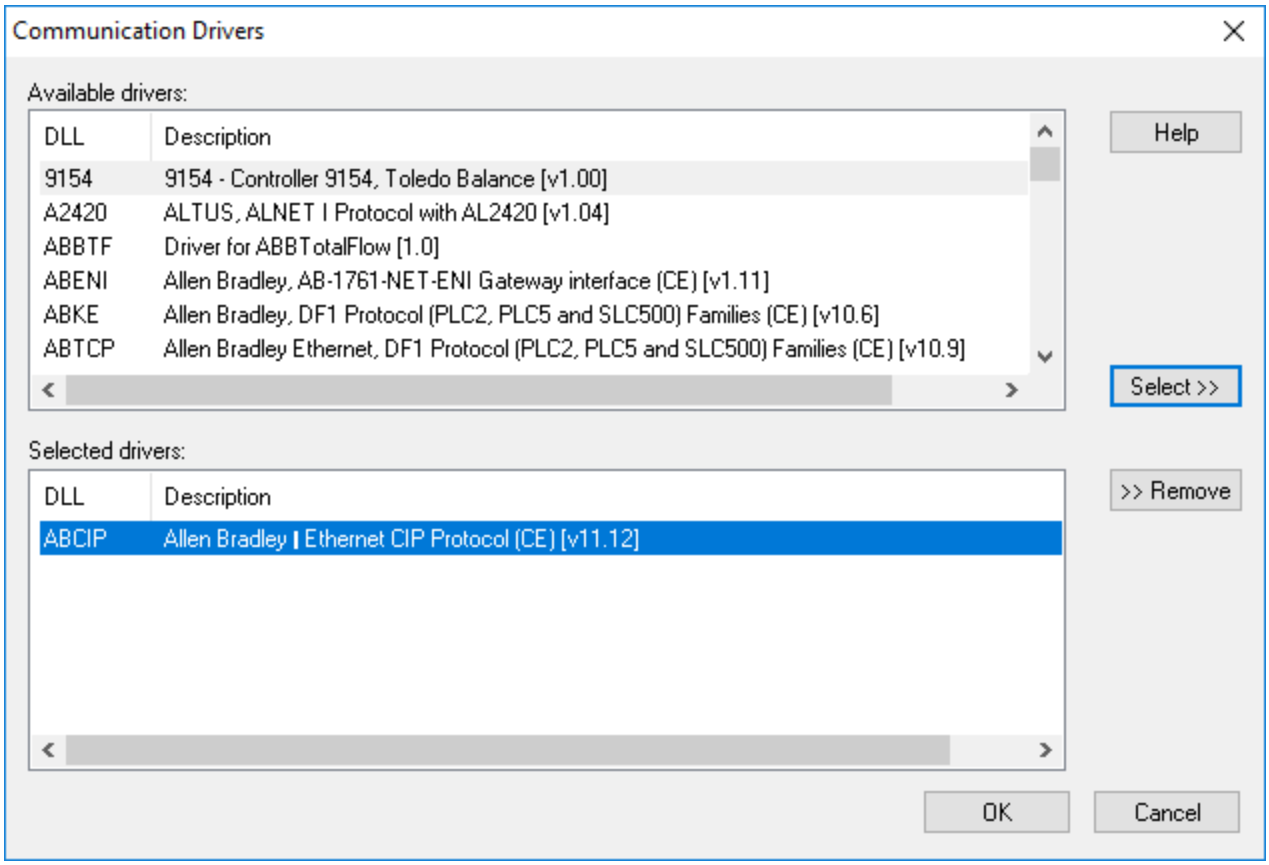
### *Help*

To open the help manual for the EcoStruxure Machine SCADA Expert 2020 software, click **Help** on the Help tab of the ribbon.

> **Tip:** This documentation is also available as a PDF.

### Communication Drivers

To see the available documentation for the communication drivers, click **Communication Drivers** on the Help tab of the ribbon.




***Communication Drivers dialog***

From this dialog, you can select an installed driver then click the **Help** button to open Adobe Acrobat® Reader™ and display a detailed document about that driver in PDF format.

### License Agreement

To display a PDF copy of the EcoStruxure Machine SCADA Expert 2020 software license, click **License Agreement** on the Help tab of the ribbon.

### Product Web Site

To go to the Schneider Electric company site, click **Product Web Site** on the Help tab of the ribbon.

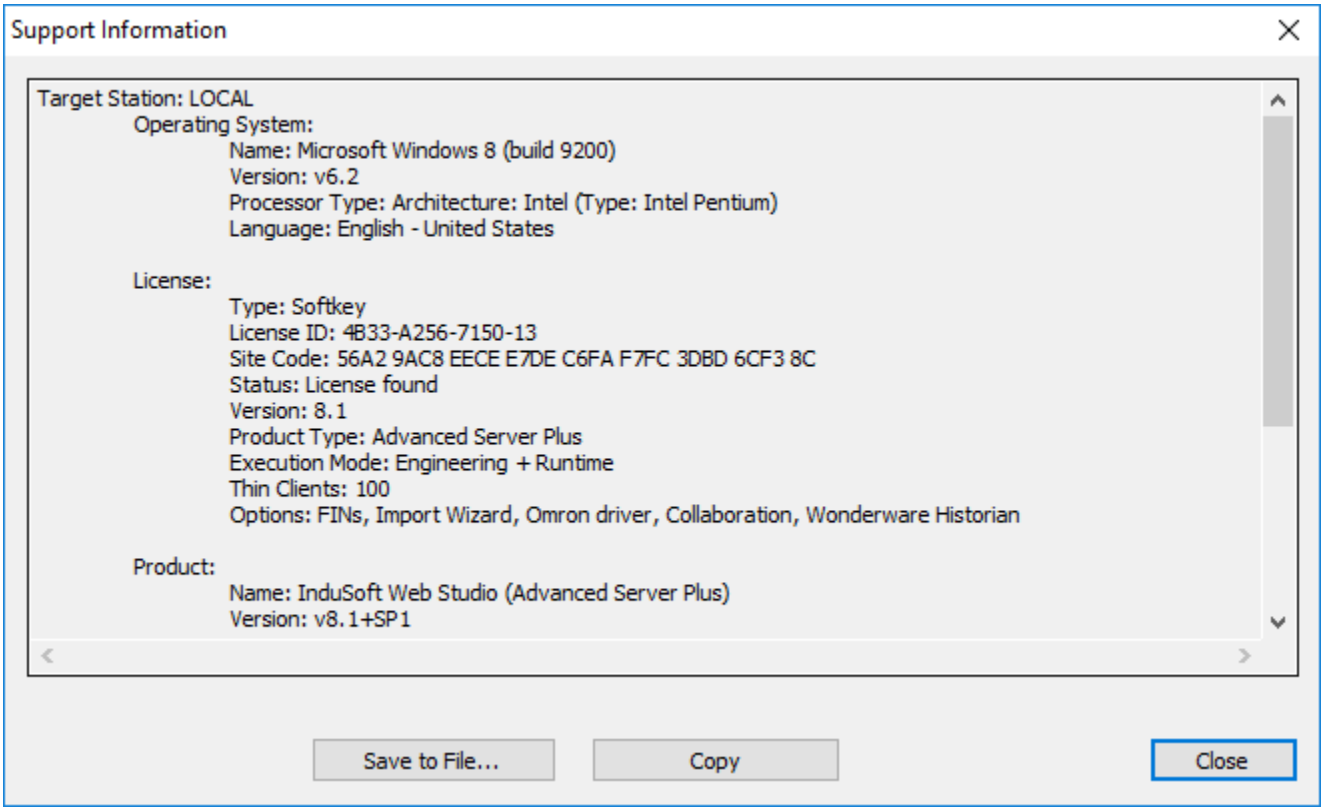
### Release Notes

To view the release notes for EcoStruxure Machine SCADA Expert 2020, click **Release Notes** on the Help tab of the ribbon. The document will be opened in your default web browser.

### Support

The *Support Information* dialog displays basic information about your computer's operating system, your EcoStruxure Machine SCADA Expert 2020 installation and license, and your project settings. If you need to contact Customer Support, then you should have this information ready to answer their questions.

To open the dialog, click **Support** on the Help tab of the ribbon. The dialog will be displayed:




***Support Information dialog***

To copy the information to the Clipboard, click **Copy**. You can then paste the information into another window or text field, such as the body of an email message.

To save the information to a file, click **Save to File**. A standard *Save As* dialog will be displayed.

When you are done, click **Close**.

## *About*

To get more information about the EcoStruxure Machine SCADA Expert 2020 software, click **About** on the Help tab of the browser.

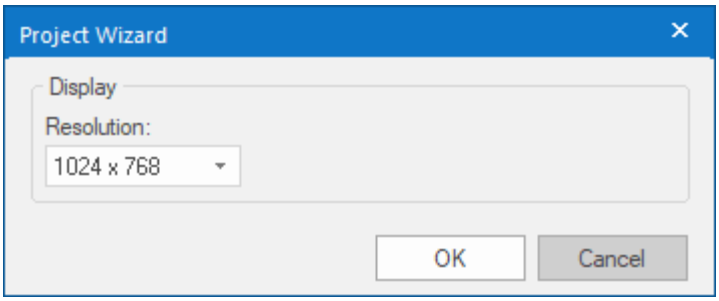
# Tutorial: Building a Simple Project

This section explains, using a step-by-step tutorial, how to build a simple project, as well as how to select and configure an I/O driver.

## Creating a new project

This part of the tutorial shows how to create a new project, including how to give it a name and then select the target platform and system.

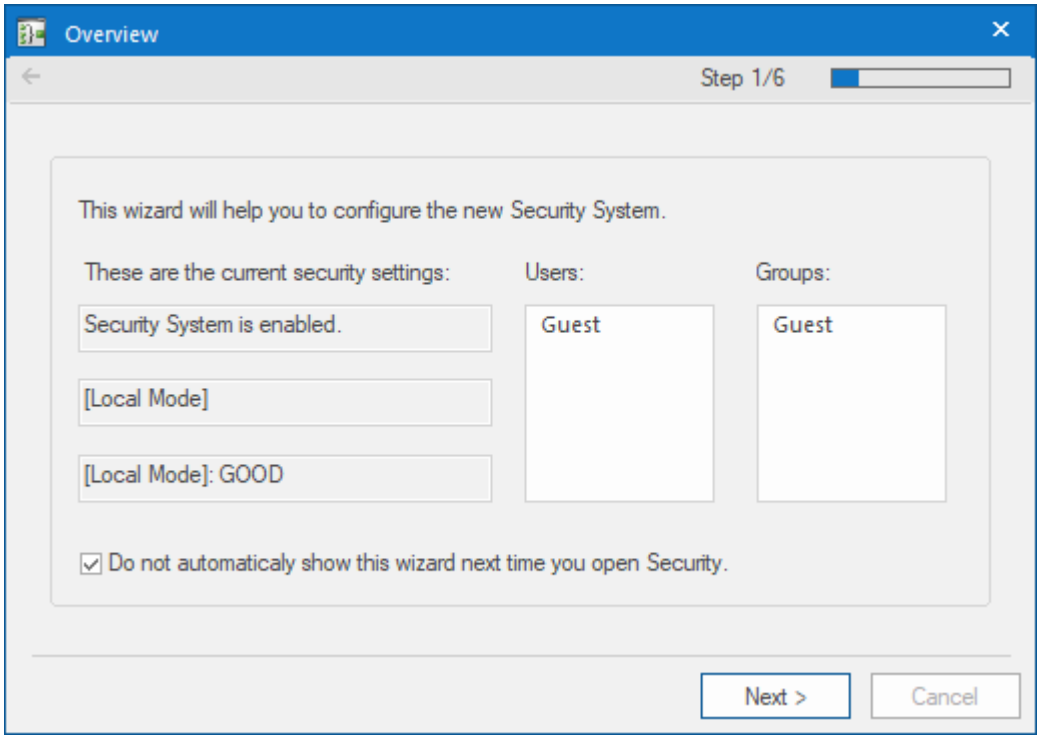
1. Go to **File**, and then click *New*.
   The *New* dialog is displayed.
2. Click the **Project** tab, if it is not already selected.
3. In the **Project name** box, type the name of your project.
   For this tutorial, type `Tutorial`.
   The development application automatically creates a new directory of the same name and assigns your project file to that directory. (Notice the **Configuration file** text box in the figure.) To put your project file somewhere other than in the default projects folder, click **Browse** and navigate to the preferred location.
4. In the **Product type** list, select the type of project that you want to build.
5. Click **OK**.
   The *New* dialog is closed and the *Project Wizard* dialog is displayed.
6. In the **Resolution** list, select **1024 x 768**.




***Specifying an empty Application with 1024x768 resolution***

7. Click **OK**.

The *Project Wizard* dialog is closed, the project is created in the development environment, and the *Security System Configuration Wizard* is displayed.




***Security System Configuration Wizard***

8. Use the *Security System Configuration Wizard* to set a Main Password for your project.
   The security system is enabled by default for all new projects.

When you finish the *Security System Configuration Wizard*, your new project is ready for development.

## Specifying the startup screen

This part of the tutorial shows how to open the project settings and then specify which screen should be displayed on startup.

- Use the **Information** tab to provide information that identifies the project (such as project description, revision number, Company name, Author's name, field equipment, and general notes).
- Use the **Options** tab to specify generic settings for the project, such as the Target System, Automatic Translation, Alarm history and Events, Default Database and Shared Tags.
- Use the **Viewer** tab to enable/disable the run-time desktop parameters.
- Use the **Communication** tab to specify communication parameters relating to the project in general.
- Use the **Preferences** tab to enable/disable warning messages when using the development application.

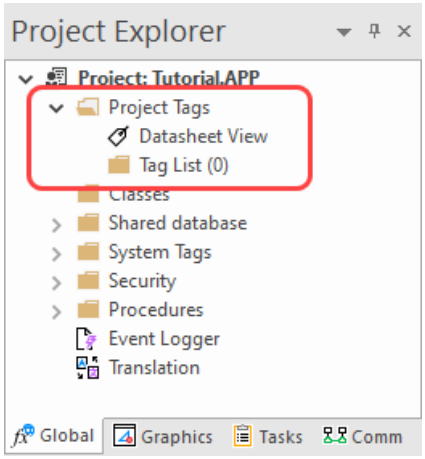
To specify the startup screen:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Viewer**.
   The *Project Settings* dialog is displayed with the **Viewer** tab selected.
2. In the **Startup screen** box, type `main.scc`.
   When you run the project, it will automatically display the main screen (or whichever screen you specify) first. You can specify a screen before you create it, but if the screen has been created, then you can also select it from the list.
3. Click **OK**.

## Creating tags

This part of the tutorial shows how to create new tags by adding them to the Project Tags datasheet.

A tag is any variable that holds a value. All tags created in a project are stored in the Project Tags folder, on the Global tab of the Project Explorer.




***Project Tags folder***

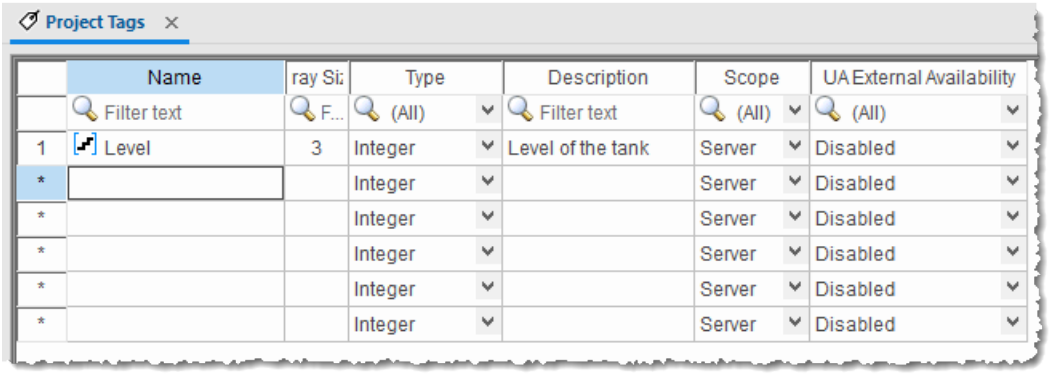
1. In the Project Explorer, click the **Global** tab.
2. Double-click **Project Tags** to expand the folder.
3. Double-click **Datasheet View** to open the *Project Tags* datasheet.
4. Use the following parameters to create a tag for the sample project.
   a) **Name**: Specify a unique tag name. For this tutorial, type `Level`.
   b) **Array Size**: Specify the top array index of the tag. (Simple tags have an **Array Size** of `0`.) For this tutorial, type `3`.

      Each array index corresponds to one of the three tanks:

      - **`Level[1]`** is the level of Tank #1
      - **`Level[2]`** is the level of Tank #2
      - **`Level[3]`** is the level of Tank #3

      You will not use **`Level[0]`** in this tutorial, even though it is a valid tag. It is easier to understand if the array indices match the tank numbers.
   c) **Type**: Specify the data type of the tag: Boolean, Integer, Real, String, or Class. For this tutorial, select **Integer**.
   d) **Description** (optional): Type a description of the tag for documentation purposes only.
   e) **Scope**: Specify how the tag is managed between the Server and the Thin Client stations.
      - Select **Local** if you want the tag to have independent values on the Server and Client stations.
      - Select **Server** if you want the tag to share the same value on the Server and Client stations.

For this tutorial, select **Server**.




***Creating the Level tag***

5. Save and close the *Project Tags* datasheet.

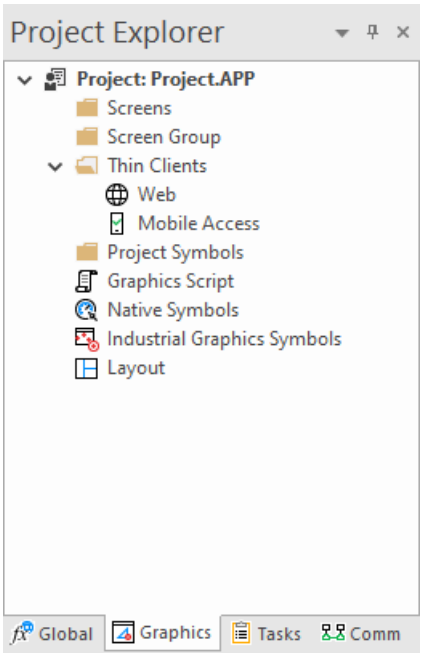
You will create additional tags as you build the project.

> **Tip:** You can sort the data in the *Project Tags* datasheet or insert/remove additional columns by right-clicking on it and then choosing the applicable option from the pop-up menu.

## Creating the main screen

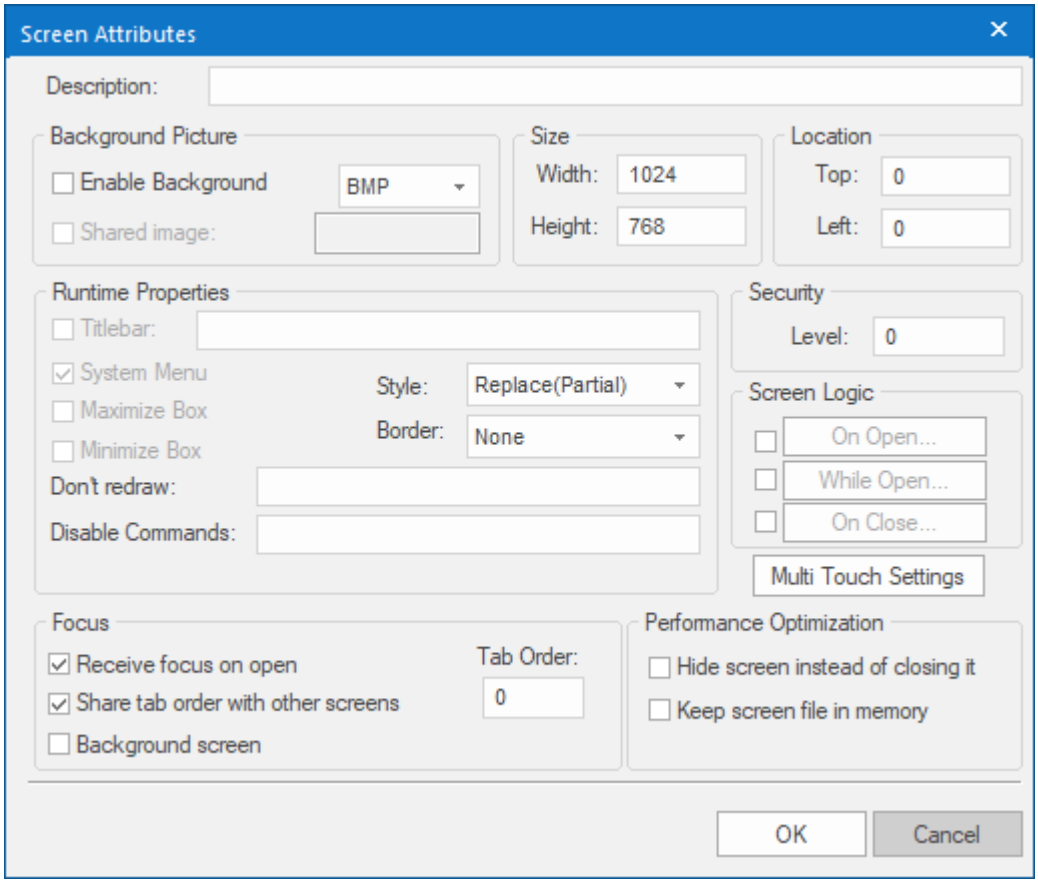
This part of the tutorial shows how to create your first screen, which will contain a single button that opens another screen.

1. In the *Project Explorer*, click the **Graphics** tab.



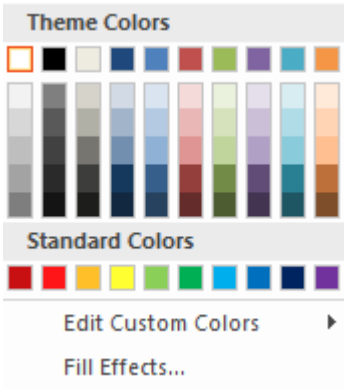

2. Right-click **Screens**, and then click **Insert** on the shortcut menu.
   The development application stores all screens created for a project in this **Screens** folder.

The *Screen Attributes* dialog is displayed.




***Screen Attributes dialog***

3. Use this dialog to set screen properties such as size and type.
   For this tutorial, click **OK** to accept the default settings.
   The *Screen Attributes* dialog is closed, and the new screen is opened in the workspace for editing.
4. On the **Draw** tab of the ribbon, in the **Screen** group, click **Background Color**.
   A standard color picker is displayed.
5. In the color picker, select a light gray color.



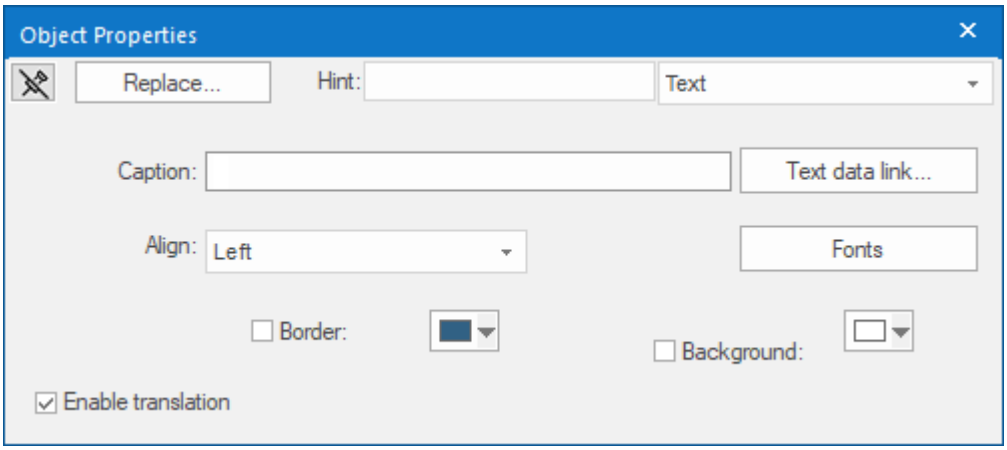

***Color picker***

That color is applied to the screen.

### *Drawing the main screen's title*

This part of the tutorial shows how to draw the main screen's title using a Text object.

1. On the **Draw** tab of the ribbon, in the **Active Objects** group, click **Text**.
   Your mouse cursor changes from an arrow to a crosshair.
2. Click on the screen, type `Welcome to the Tutorial Application`, and then press Return.
   This creates a new Text object with the specified text.
3. Double-click the object to open its *Object Properties* dialog.

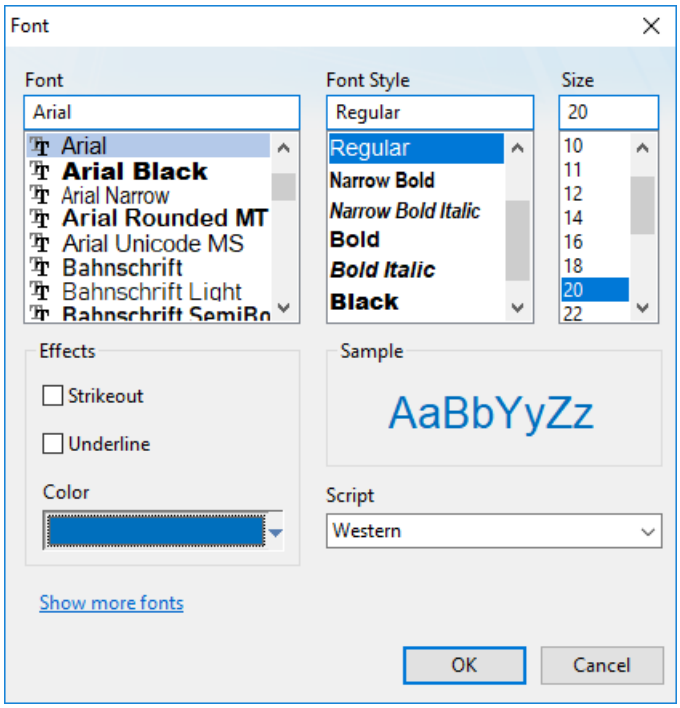



***Object Properties: Text dialog***

- Double-clicking on any screen object opens an *Object Properties* dialog containing the properties for that object. The properties shown in the dialog change depending on the type of object.
- The *Object Properties* dialog also contains a pin button that controls whether this dialog remains open. The button changes state (and function) each time you click on it, as follows:
  - When the pin button is released, the focus is passed to the object on the screen as soon as it is selected. It is recommended that this button is kept released when you want to manipulate the objects (Copy, Paste, Cut, or Delete). Although the *Object Properties* dialog is on the top, the keyboard commands (**Ctrl+C**, **Ctrl+V**, **Ctrl+X**, or **Del**) are sent directly to the objects.
  - When the pin button is pressed, the focus is kept on the *Object Properties* dialog, even when you click the objects on the screen. We recommend you keep this button pressed when you want to modify the settings of the objects. You can click an object and type the new property value directly in the *Object Properties* dialog (it is not necessary to click on the window to bring focus to it). Also, when the pin button is pressed, the *Object Properties* dialog does not automatically close when you click on the screen.

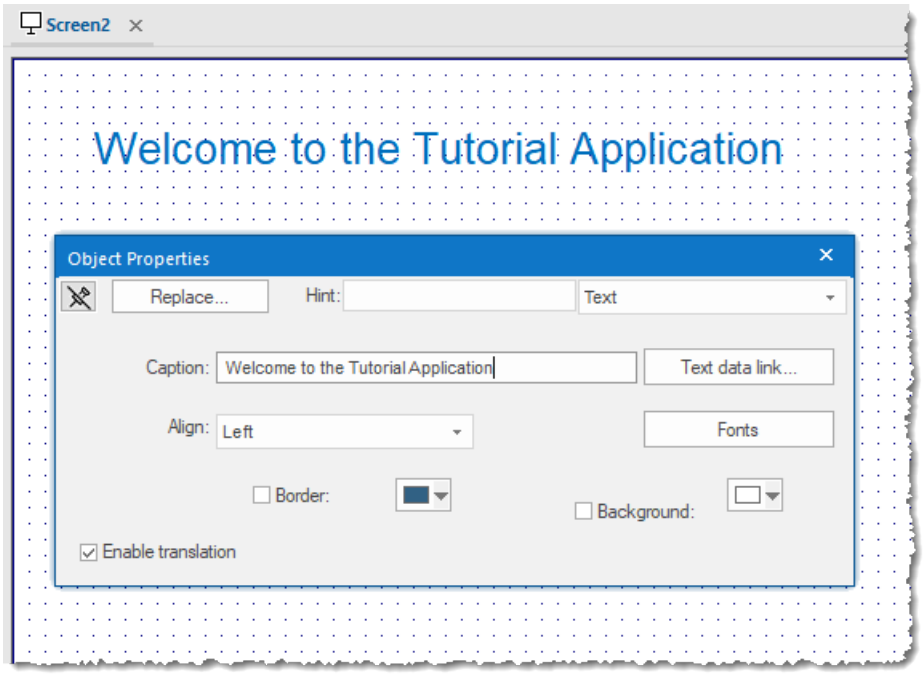
4. Click **Fonts** to open *Font* dialog, and then specify the font settings.
   For this tutorial...
   - Font is **Arial**
   - Font style is **Regular**
   - Size is **20**

- Color is **Blue**




***Specifying the font settings***

5. Click **OK** to close the *Font* dialog.
   The font settings are applied to the Text object.



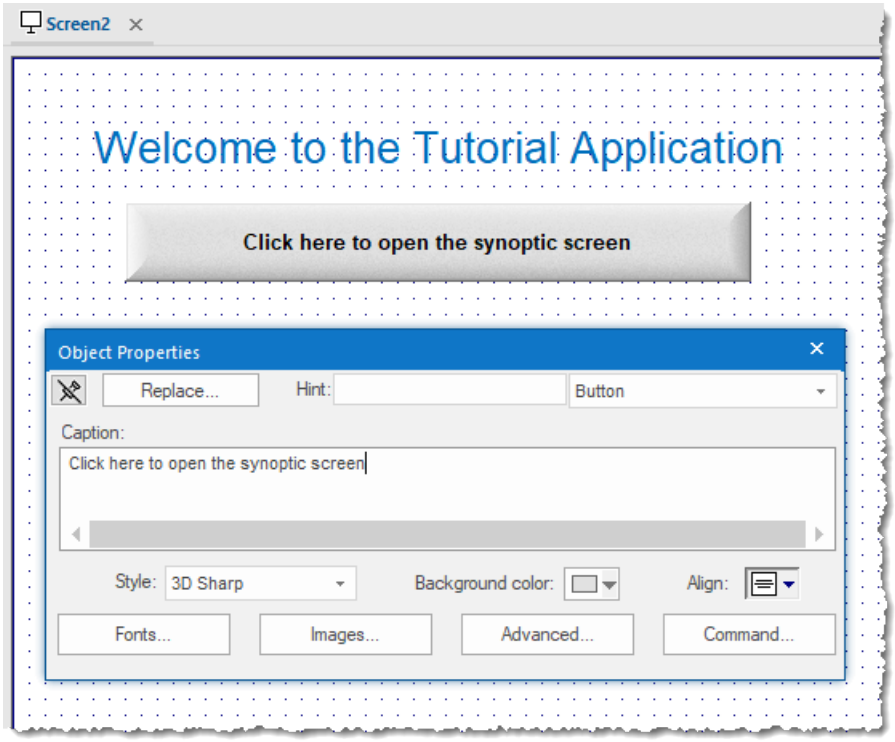

***Font settings applied to Text object***

6. Close the *Object Properties* dialog (i.e., click the Close button in the dialog box's top-right corner).

### *Drawing a button to open another screen*

This part of the tutorial shows how to draw and configure a button that will open another screen.

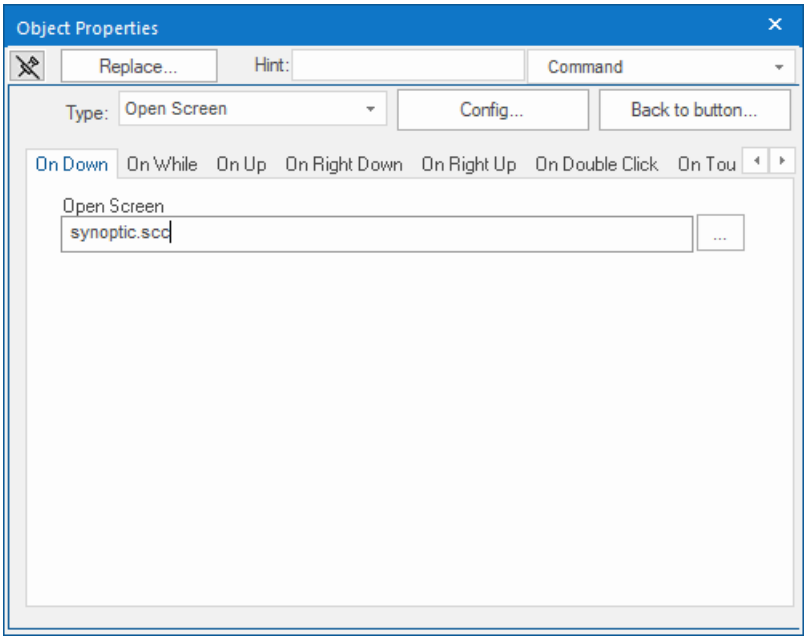
1. On the **Draw** tab of the ribbon, in the **Active Objects** group, click **Button**.
   Your mouse cursor changes from an arrow to a crosshair.
2. Click and hold on the screen, and then drag the cursor to draw the Button object.
3. Double-click the object to open its *Object Properties* dialog.
4. In the **Caption** box, type the following text: `Click here to open the synoptic screen`.




***Adding a caption to the button***

5. Click **Command**.
   The *Object Properties* dialog changes to show the properties for the Command animation.
6. In the **Type** list, select **Open Screen**.

7. In the **Open Screen** box, type `synoptic.scc`.




***Configuring an Open Screen command on the button***

You can specify a screen that you have not yet created.

8. Close the *Object Properties* dialog.

## Saving and closing the main screen

This part of the tutorial shows how to properly save and close a screen.

1. Go to **File**, and then select **Save**.
   A standard Windows *Save* dialog is displayed.
2. In the **File name** box, type `main`.
3. Click **Save**.
   The file is saved in your project folder (at `<project name>\Screen\main.scc`), and the *Save* dialog is closed.
4. Go to **File**, and then select **Close**.

## Creating the synoptic screen

This part of the tutorial show how to create your second screen, which will include an animated tank of liquid and some basic controls for that tank.

1. In the **Graphics** tab of the *Project Explorer*, right-click the **Screens** folder, and then click **Insert** on the shortcut menu.
   The *Screen Attributes* dialog is displayed.
2. Use this dialog to set attributes such as size and type.
   For this tutorial, click **OK** to accept the default settings.
3. Go to **File**, and then select **Save As**.
   A standard Windows *Save As* dialog is displayed.
4. In the **File name** box, type `synoptic`.
5. Click **Save**.
   The file is saved in your project folder (at `<project name>\Screen\synoptic.scc`), and the *Save* dialog is closed.

Do not close the screen like you did the main screen when you saved it. You still need to draw the synoptic screen.

### Drawing the synoptic screen's title

As in a previous part, this part of the tutorial shows how to draw the synoptic screen's title using a Text object.

1. On the **Draw** tab of the ribbon, in the **Active Objects** group, click **Text**.
2. Click on the screen, type `Synoptic Screen`, and then press Return.
3. Double-click the object to open its *Object Properties* dialog.
4. Click **Fonts** to open *Font* dialog, and then specify the font settings.
   For this tutorial...
   - Font is **Arial**
   - Font style is **Bold**
   - Size is **20**
   - Color is **Blue**
5. Click **OK** to save the font settings and close the dialog.
6. Close the *Object Properties* dialog.
7. Move the Text object to the top left corner of the screen.
8. Go to **File**, and then select **Save**.

This figure shows how your screen should look after you have drawn the screen title.

***Finished screen title***

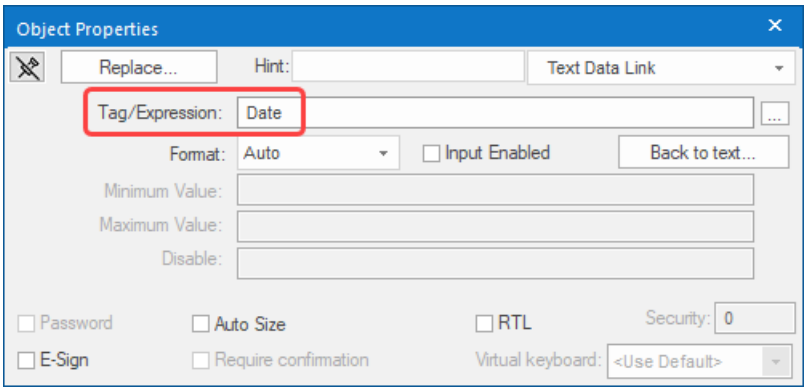
### Drawing "Date" and "Time" displays

This part of the tutorial shows how to draw "Date" and "Time" displays by linking Text objects to system tags.

**`Date`** and **`Time`** are system tags that hold the current date and time of the local station. These tags are available to any project.

1. On the **Draw** tab of the ribbon, in the **Active Objects** group, click **Text**.

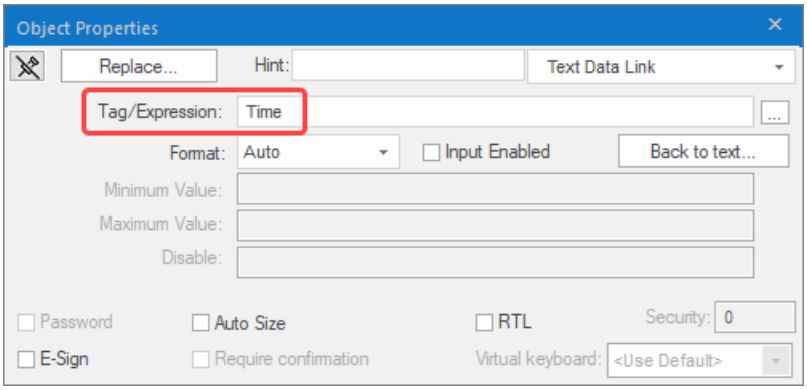
2. Click on the screen, type `Date: ##########`, and then press Return.
3. Double-click the object to open its *Object Properties* dialog.
4. Click **Text Data Link**.
   The *Object Properties* dialog changes to show the properties for the Text Data Link animation.
5. In the **Tag/Expression** box, type `Date`.




***Specifying the Date system tag***

During run time, the project replaces the **##########** characters of the Text object with the value of the system tag **Date**.

6. Close the *Object Properties* dialog.
7. On the **Draw** tab of the ribbon, in the **Active Objects** group, click **Text**.
8. Click on the screen, type `Time: ##########`, and then press Return.
9. Double-click the object to open its *Object Properties* dialog.
10. Click **Text Data Link**.
    The *Object Properties* dialog changes to show the properties for the Text Data Link animation.
11. In the **Tag/Expression** box, type `Time`.




***Specifying the Time system tag***

During run time, the project replaces the **##########** characters of the Text object with the value of the system tag **Time**.

12. Close the *Object Properties* dialog.
13. Go to **File**, and then select **Save**.

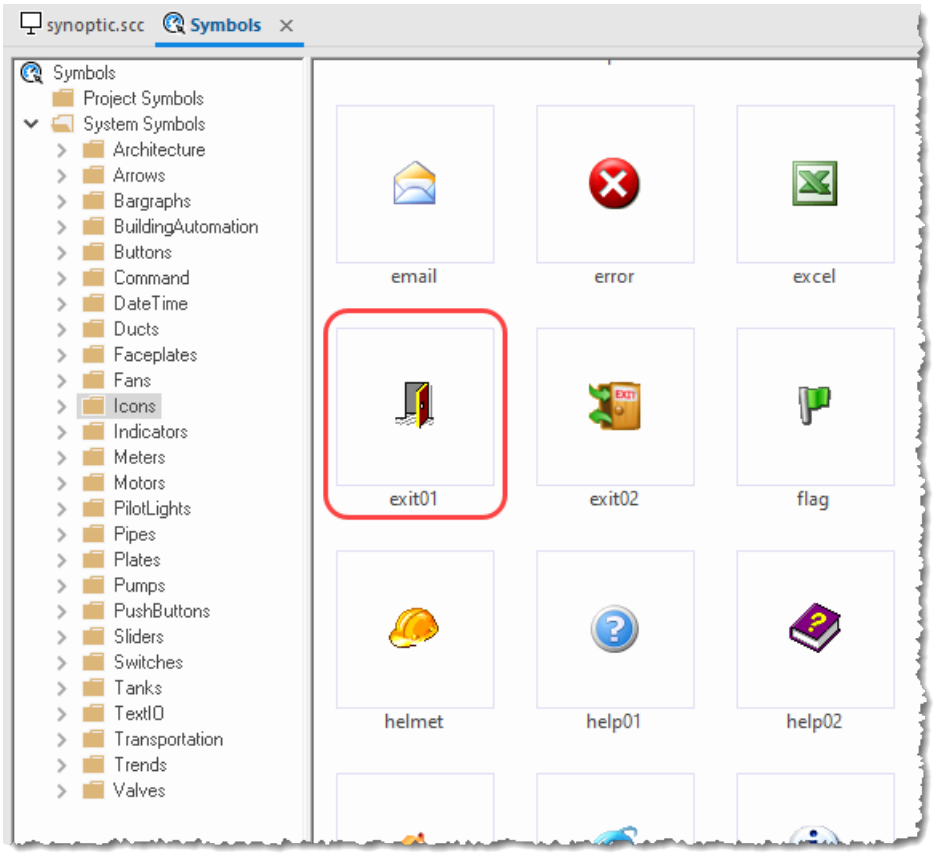
This figure shows how your screen should look after you have created the date and time objects.

***Finished date and time objects***

## *Placing an "Exit" icon*

This part of the tutorial shows how to place an icon (by selecting and configuring a Linked Symbol) that allows the user to exit the project, .

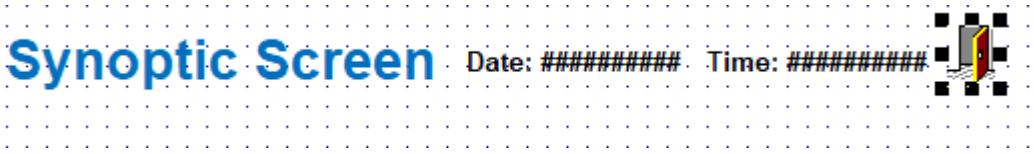
1. On the **Draw** tab of the ribbon, in the **Libraries** group, click **Symbols**.
   The symbols library is displayed.
2. In the Symbols menu tree, open the **System Symbols** folder and then open the **Icons** sub-folder.
3. In the Icons sub-folder, select **exit01**.
   The symbol will be displayed in the symbol viewer to the right of the menu tree.




***Selecting the "exit01" symbol***

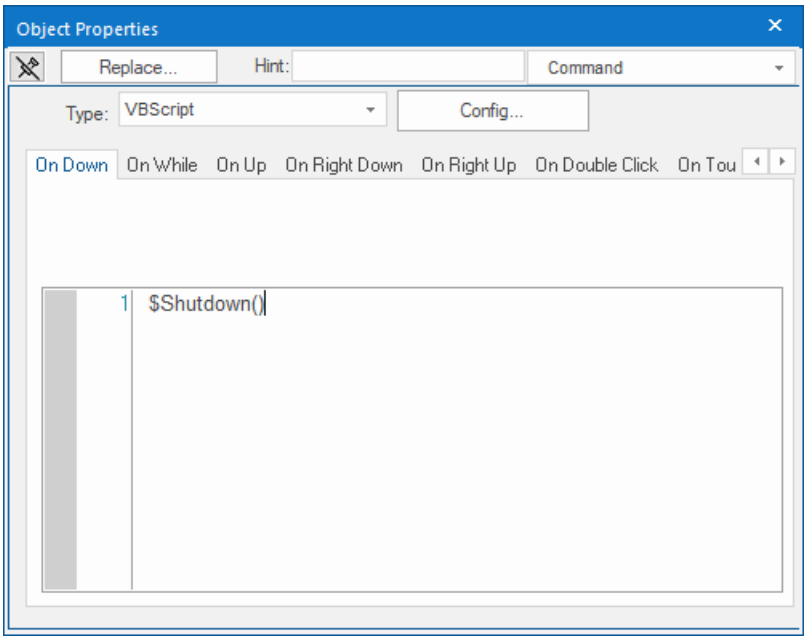
4. Click on the symbol.
   The mouse cursor will change to show that the symbol is ready to be placed in a screen.
5. Switch back to the screen where you want to place the symbol and then click in it.

The symbol is placed as a Linked Symbol object.




***Placing the Linked Symbol object***

6. With the object still selected, click **Command** (on the **Draw** tab of the ribbon, in the **Animations** group) to apply this animation to the object.
7. Double-click the object to open its *Object Properties* dialog.
8. In the **Type** list, select **VBScript**.
9. In the **On Down** box, type `$Shutdown()`.
   `Shutdown` is one of EcoStruxure Machine SCADA Expert 2020's built-in scripting functions, but it can be used within VBScript by prefacing it with a dollar sign (**$**).




***Specifying the Shutdown command on the symbol***

10. Close the *Object Properties* dialog.
11. Go to **File**, and then select **Save**.

Now, when a user clicks this icon during run time, the project will stop and exit to the station's desktop.

## *Testing the project*

This part of the tutorial shows how to test the project so far.

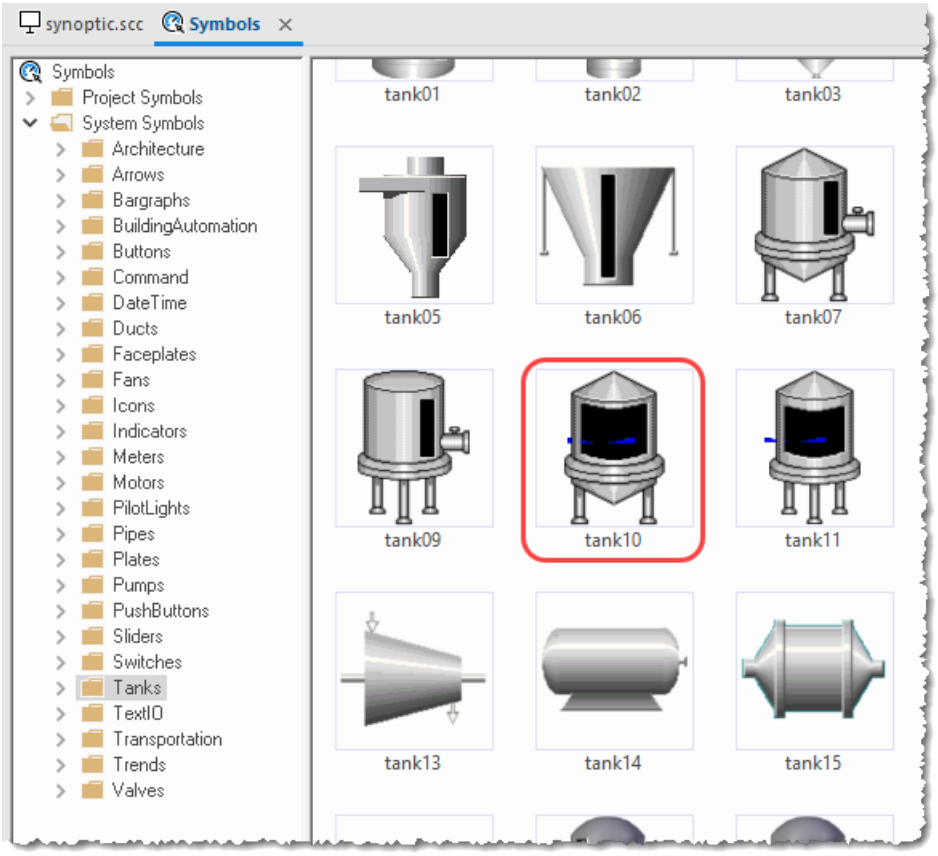
1. Go to **File**, and then select **Close All**.
   All open worksheets are closed.
2. On the **Home** tab of the ribbon, in the **Local Management** group, click **Run**.
   The project runs and the startup screen is displayed.
3. Click the button to open the synoptic screen.
   The synoptic screen is displayed.
4. Click the exit icon to shut down the project.

If any part of the project does not work as expected, switch back to the development application (**ALT+TAB**) and then click **Stop** on the Home tab of the ribbon.

### *Placing an animated tank*

This part of the tutorial shows how to select an animated tank from the Symbol Library and place it on the screen (similar to how you selected and placed the "Exit" icon), then associate some project tags with the tank's properties.

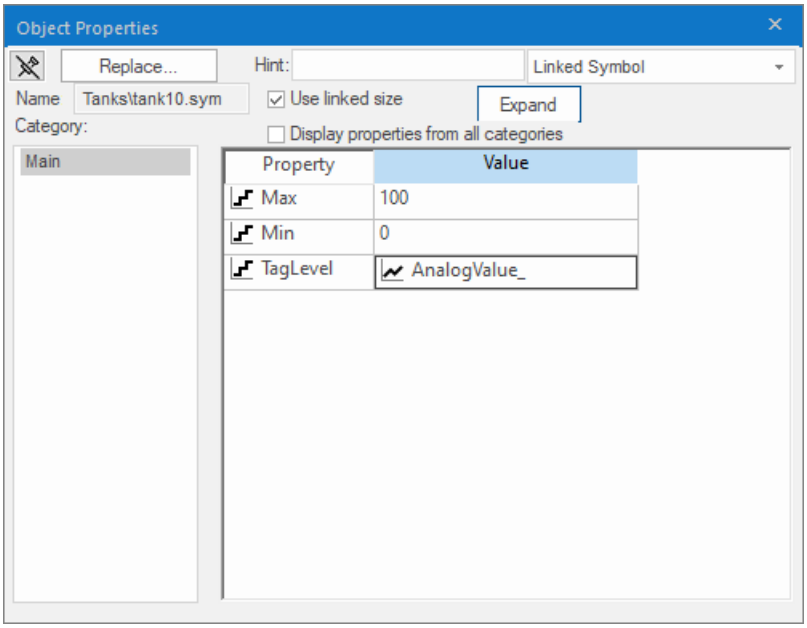
1. In the **Graphics** tab of the *Project Explorer*, expand the **Screens** folder.
2. Double-click **synoptic.scc**.
   The synoptic screen worksheet is reopened for editing.
3. On the **Draw** tab of the ribbon, in the **Libraries** group, click **Symbols**.
4. In the Symbols menu tree, open the **System Symbols** folder and then open the **Tanks** sub-folder.
5. Browse the tank symbols and choose one.
   You may choose any tank symbol that you like; they all function basically the same.




***Choosing a tank symbol***

6. Click the symbol.
   The mouse cursor will change to show that the symbol is ready to be placed in a screen.
7. Switch back to the screen where you want to place the symbol and click in it.
   The symbol is placed as a Linked Symbol object.

8. Double-click the object to open its *Object Properties* dialog.



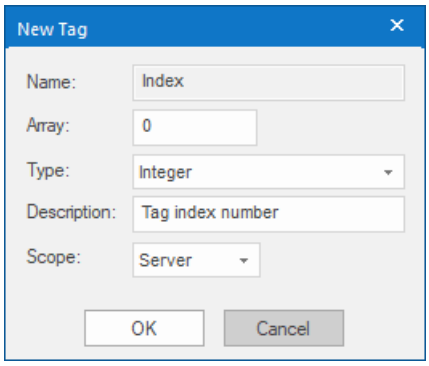

***The tank symbol's properties***

A tank is an arrangement of different objects and animations (for example a rectangle, a bar graph, etc.), all combined together as a Linked Symbol. You can modify the properties of this symbol by editing the properties list. For this tutorial, you will modify the tag associated with the tank level.

9. For the property **`TagLevel`**, delete the existing value and then type `Level[Index]`.
Note that you do not need to reopen the Project Tags datasheet to create tags as you develop the project. Because you have not previously created the tag **`Index`** in the Project Tags database, an alert message asks you if you would like to create it.

10. Click **Yes**.
A *New Tag* dialog is displayed.

11. Configure the new tag with **Array** as `0`, **Type** as `Integer`, and **Scope** as `Local`.




***Configuring a new tag***

12. Click **OK** to close the *New Tag* dialog.

You can use the tag **`Index`** to set the array position of the tag **`Level`**, and show the level for any of the three tanks in the same object:

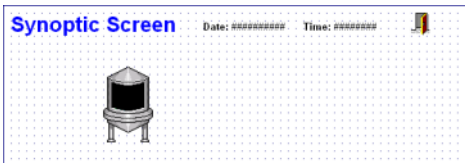
- When **`Index`** equals **`1`**, the tank object shows the level of Tank #1 (i.e., **`Level[1]`**);
- When **`Index`** equals **`2`**, the tank object shows the level of Tank #2 (i.e., **`Level[2]`**); and

- When **Index** equals **3**, the tank object shows the level of Tank #3 (i.e., **Level[3]**).

Also, because the tag scope is local, the tag can have different values for the Server and Client stations at the same time. Consequently, the local user (i.e., the Server station) can be monitoring the level of Tank #1 while the remote user (i.e., the Client station) is monitoring the level of Tank #2.

13. Close the *Object Properties* dialog.
14. Go to **File**, and then select **Save**.

This figure shows how your screen should look after you've created the tank object.

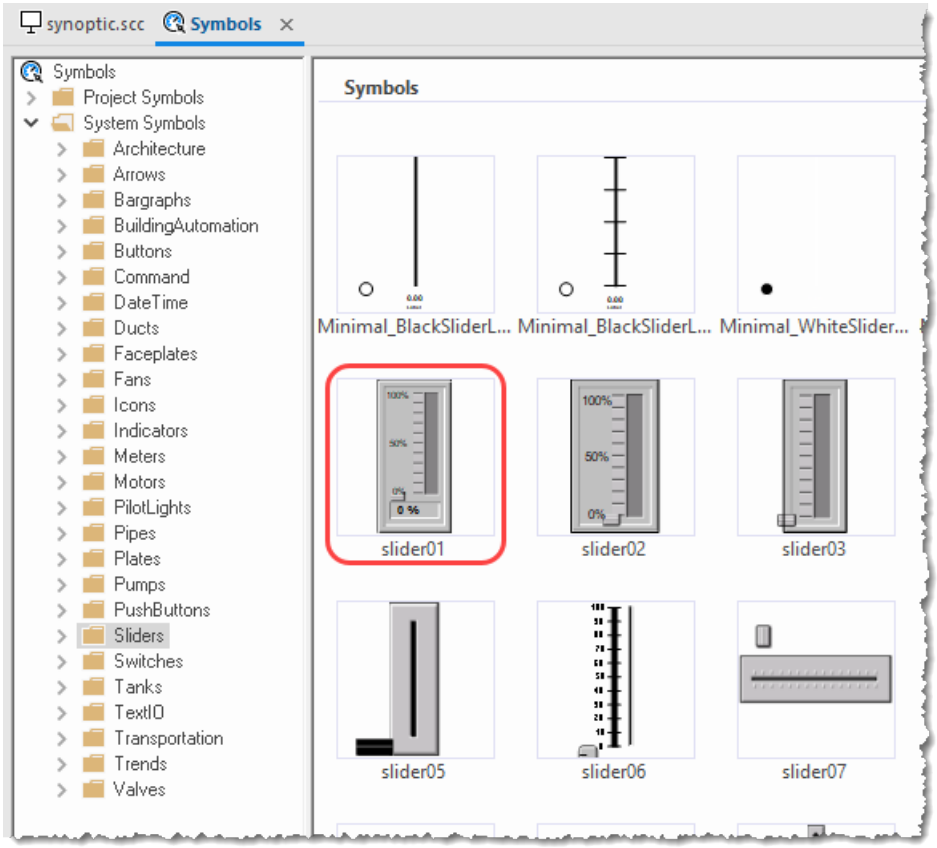



***Finished tank object***

## *Placing a level slider*

This part of the tutorial shows how to select a slider control from the Symbol Library and then connect it to the animated tank.

1. On the **Draw** tab of the ribbon, in the **Libraries** group, click **Symbols**.
2. In the Symbols menu tree, open the **System Symbols** folder and then open the **Sliders** sub-folder.



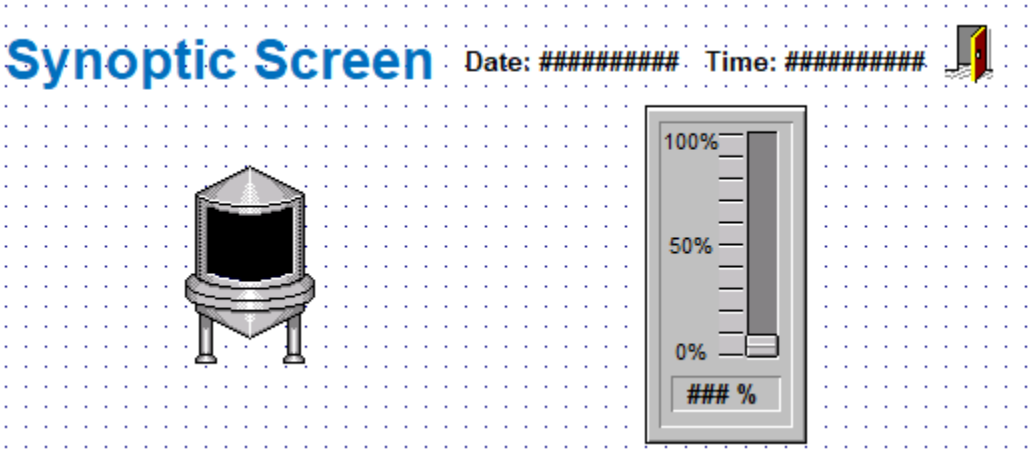

***Selecting a slider symbol***

3. In the Sliders sub-folder, select a slider control.

You may select any slider you like; they all function basically the same way.

4. Click on the symbol.
   The mouse cursor will change to show that the symbol is ready to be placed in a screen.
5. Switch back to the screen where you want to place the symbol and click in it.
   The symbol is placed as a Linked Symbol object.
6. Double-click the object to open its *Object Properties* dialog.
7. For the property **TagName**, delete the existing value and then type `Level[Index]`.
   Just as with the tank, you need to modify the symbol property associated with the slider level.
8. Close the *Object Properties* dialog.
9. Go to **File**, and then select **Save**.

This figure shows how your screen should look after you've created the level slider object.



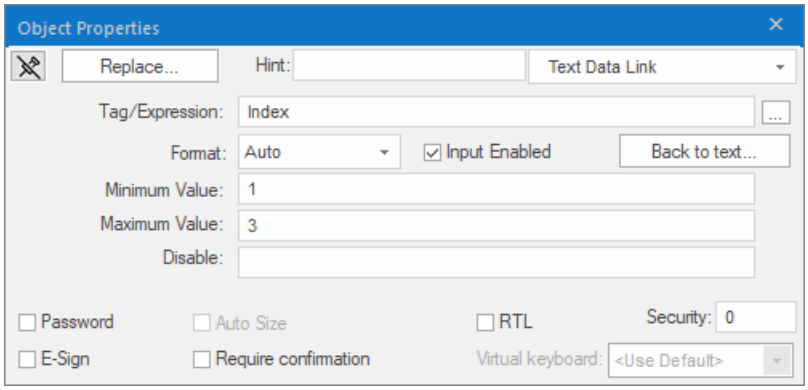

***Finished level slider object***

### *Drawing a tank selector*

This part of the tutorial shows how to draw a text input box that can be used to change which real-world tank is represented by the animated tank on the screen.

1. On the **Draw** tab of the ribbon, in the **Active Objects** group, click **Text**.
2. Click on the screen, type **`Tank: #`**, and then press Return.
3. Double-click the object to open its *Object Properties* dialog.
4. Click **Text Data Link**.
   The *Object Properties* dialog changes to show the properties for the Text Data Link animation.
5. In the **Tag/Expression** box, type `Index`.
6. Select the **Input Enabled** option.
   This lets the user enter a new value for the tag during run time.
7. In the **Minimum Value** box, type `1`.

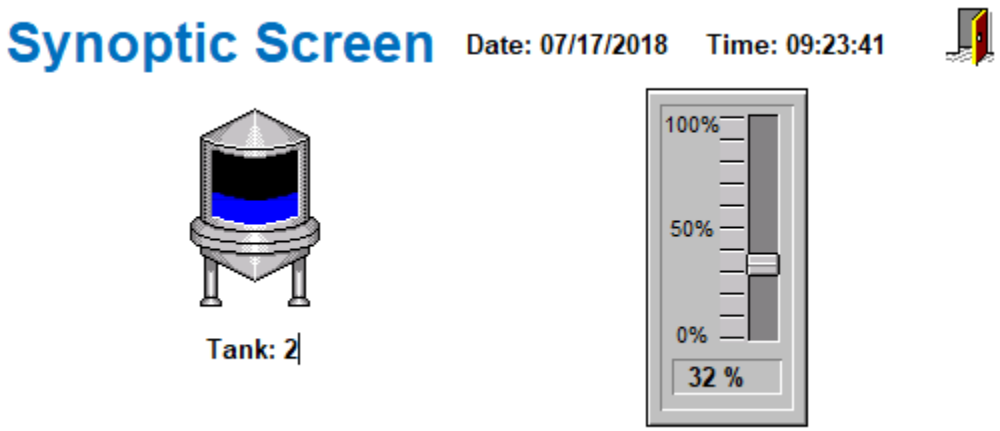
8. In the **Maximum Value** box, type 3.




***Configuring the "Tank" text input***

9. Close the *Object Properties* dialog.
10. Go to **File**, and then select **Save**.

This figure shows how your screen should look after you've created the tank selector object.




***Finished tank selector object during run time***

## *Testing the project*

This part of the tutorial shows how to test the project again with the animated tank, the level slider, and the tank selector.

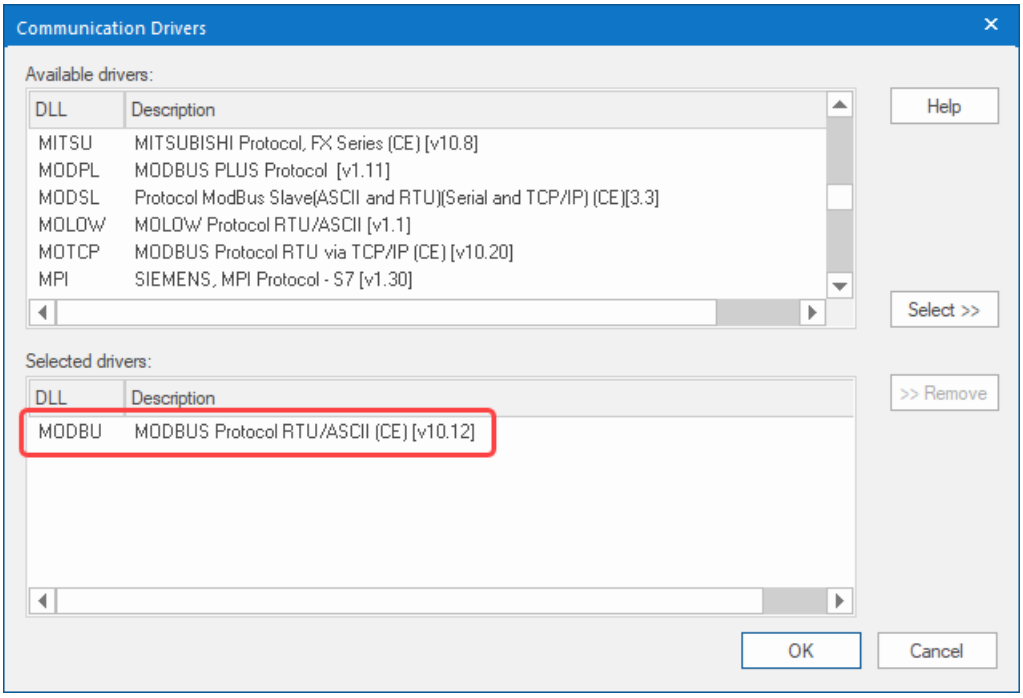
1. Go to **File**, and then select **Close All**.
   All open worksheets are closed.
2. On the **Home** tab of the ribbon, in the **Local Management** group, click **Run**.
   The project runs and the startup screen is displayed.
3. Click the button to open the synoptic screen.
   The synoptic screen is displayed.
4. Type the tank number (1, 2, or 3) in the Tank label, and then use the slider to adjust the tank level.
   Note that you can view/adjust the level of each tank independently.
5. Click the exit icon to shut down the project.

If any part of the project does not work as expected, switch back to the development application (**ALT+TAB**) and then click **Stop** on the Home tab of the ribbon.

## Configuring the communication driver

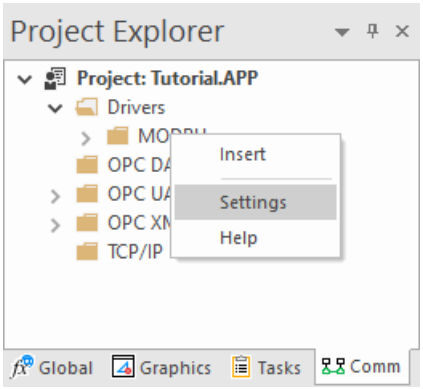
This part of the tutorial shows how to select and configure a driver to communicate with an external I/O device.

1. In the *Project Explorer*, click the **Comm** tab.
2. Right-click the **Drivers** folder, and the click **Add/Remove Drivers** on the shortcut menu.
   The *Communication Drivers* dialog is displayed.
3. Select a driver from the **Available drivers** list, and then click **Select**.
   For this tutorial, select MODBU.
   The driver is moved to the **Selected drivers** list.

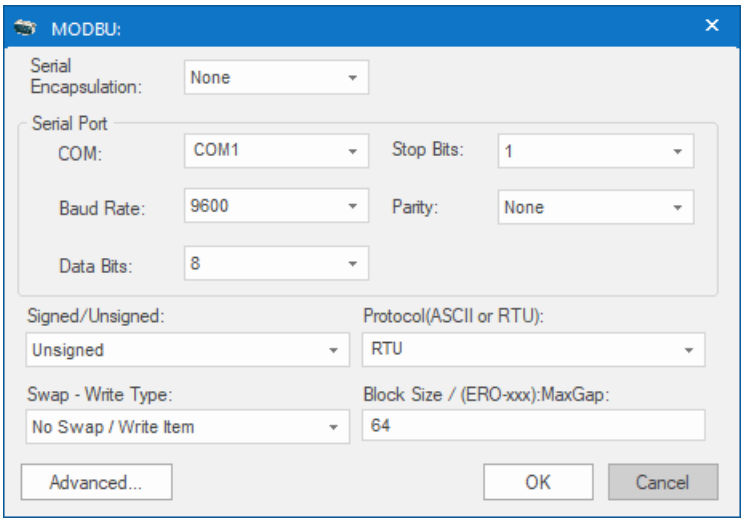



***MODBU driver selected***

4. Click **OK**.
   The *Communication Drivers* dialog is closed, and the driver is added to the Drivers folder in the *Project Explorer*.
5. In the Project Explorer, right-click the **MODBU** folder, and then click **Settings** on the shortcut menu.



The *Communication Settings* dialog is displayed.




***Communication Settings dialog for MODBU driver***

6. Configure the communication settings as needed for the target device.
For this tutorial, accept the default settings.

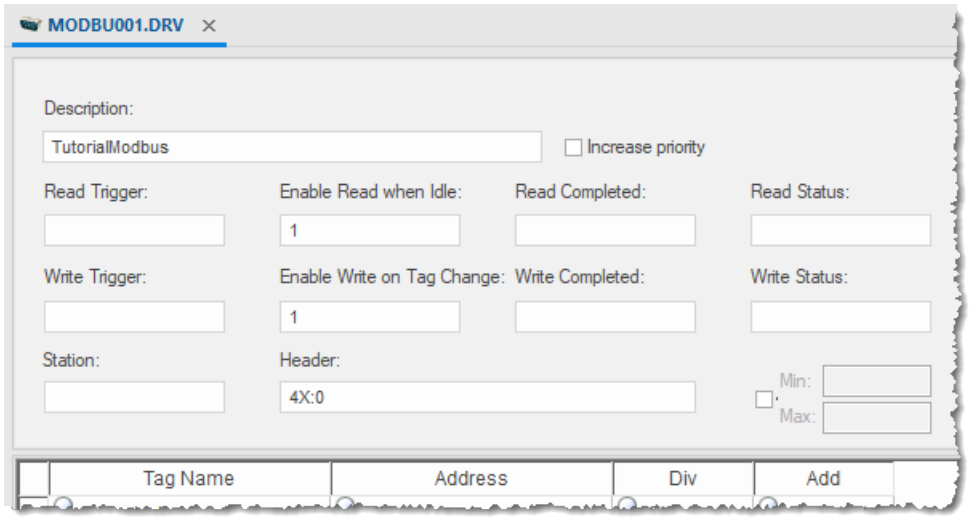
> **Note:** For more information about a specific driver, click **Communication Drivers** on the **Help** tab of the ribbon.

7. Click **OK** to close the dialog.
8. In the Project Explorer, right-click the **MODBU** folder and then click **Insert** on the shortcut menu.
A new driver worksheet named `MODBU001.drv` is created and opened for editing.
9. Configure the worksheet header:
   a) In the **Description** box, type `Tutorial Modbus`.
   This setting is for documentation only; it does not affect the project during run time.
   b) In the **Enable Read When Idle** box, type `1`.
   This setting is a trigger that takes a Boolean value. A value of 1 — either entered manually as above or evaluated from a tag/expression — forces your project to continue reading tag values from the target device even when there are no changes in value.
   c) In the **Enable Write On Tag Change** box, type `1`.
   This setting is also a trigger. A value of 1 forces your project to write tag values to the target device only when those values change, rather than continuously. This saves system resources and improves performance during rutime.
   d) In the **Station** box, type `1`.
   This indicates the I/O device number to be accessed by this driver. Typically, the PLC is specified as Device #1.
   e) In the **Header** box, type `4X:0`.
   You must use a driver-specific format. The format for the MODBU driver is:

   ***register_type*:*initial_offset***

| Register Type | Description |
|---|---|
| **0X** | Coil Status |
| **1X** | Input Status |
| **3X** | Input Register |

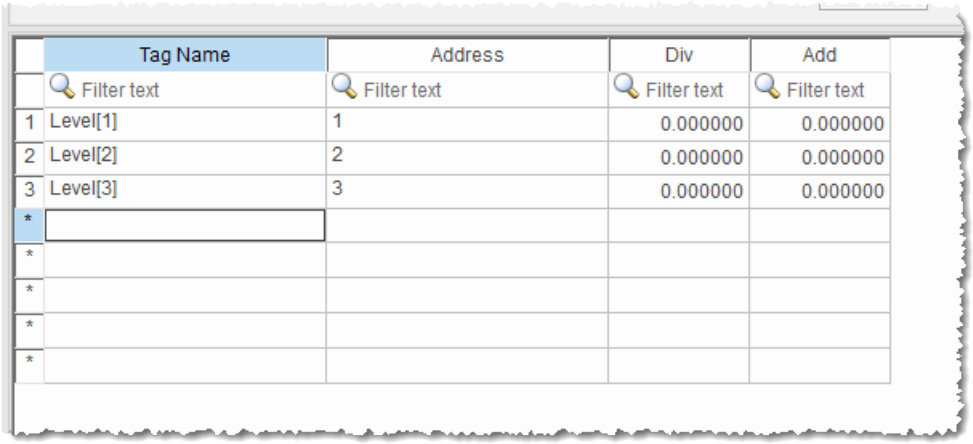
| Register Type | Description |
|---|---|
| **4X** | Holding Register |
| **ID** | Slave ID Number |




***Completed worksheet header***

10. In the worksheet body, enter the tags and their associated device addresses — for each tag:
    a) In the **Tag Name** field, type the name of the project tag.
    b) In the **Address** field, type the value to be added to the header to form the complete device address.

| Tag Name | Address | Complete Device Address |
|---|---|---|
| `Level[1]` | 1 | 4X:1 (Holding Register 1) |
| `Level[2]` | 2 | 4X:2 (Holding Register 2) |
| `Level[3]` | 3 | 4X:3 (Holding Register 3) |




***Completed worksheet body***

11. Go to **File**, and then select **Save**.
12. When prompted to choose the driver sheet number, type `1` and then click **OK**.

### *Monitoring device I/O during run time*

This part of the tutorial shows how to monitor device I/O during run time by using the *Log* window.

1. On the **Home** tab of the ribbon, in the **Local Management** group, click **Run**.
   The project runs and the startup screen is displayed.
2. Press **ALT+TAB** to switch back to the development application.
3. Right-click in the *Output* window, and then click **Settings**.
   The *Log Settings* dialog is displayed.
4. Select the **Field Read Commands**, **Field Write Commands**, and **Protocol Analyzer** options.
5. Click **OK** to close the *Log Settings* dialog.

You can now monitor the device I/O during run time.

# Appendix: Built-in Language

This section describes the functions that make up the Built-in Language. Each function description includes complete syntax, possible returned values, and examples of usage.

## Logic and arithmetic operators

The Built-in Language supports the following logic and arithmetic operators.

### Logic operators

| Operator | Usage | Description |
|---|---|---|
| **AND** | **A AND B** | TRUE if A and B are both TRUE |
| **OR** | **A OR B** | TRUE if A is TRUE, or B is TRUE, or both |
| **XOR** | **A XOR B** | TRUE if A is TRUE, or B is TRUE, but not both |
| **NOT** | **NOT A** | TRUE if A is FALSE |
| **=** | **X = Y** | TRUE if X is equal to Y |
| **>** | **X > Y** | TRUE if X is greater than Y |
| **>=** | **X >= Y** | TRUE if X is greater than or equal to Y |
| **<** | **X < Y** | TRUE if X is less than Y |
| **<=** | **X <= Y** | TRUE if X is less than or equal to Y |
| **<>** | **X <> Y** | TRUE if X is not equal to Y |
| **&** | **X & Y** | Bitwise AND:<br>**0101 (decimal 5)**<br>**AND 0011 (decimal 3)**<br>**= 0001 (decimal 1)** |
| **\|** | **X \| Y** | Bitwise OR:<br>**0101 (decimal 5)**<br>**OR 0011 (decimal 3)**<br>**= 0111 (decimal 7)** |
| **^** | **X ^ Y** | Bitwise XOR:<br>**0101 (decimal 5)**<br>**XOR 0011 (decimal 3)**<br>**= 0110 (decimal 6)** |
| **~** | **~ X** | Bitwise NOT:<br>**NOT 0101 (decimal 5)**<br>**= 1010 (decimal 10)** |
| **>> *n*** | **X >> Y** | Rotate ***n*** bits to right:<br>**0110 (decimal 6)**<br>**ROTATE RIGHT**<br>**= 0011 (decimal 3)** |
| **<< *n*** | **X << Y** | Rotate ***n*** bits to left:<br>**0110 (decimal 6)**<br>**ROTATE LEFT** |

| Operator | Usage | Description |
|---|---|---|
| | | `= 1100 (decimal 12)` |

> **Tip:** For more complex logic, try the Logical and Loop functions.

### Arithmetic operators

| Operator | Usage | Description |
|---|---|---|
| `+` | `X + Y` | Add (plus) |
| `-` | `X - Y` | Subtract (minus) |
| `*` | `X * Y` | Multiply by |
| `/` | `X / Y` | Divide by |

Arithemtic operators are resolved according to the standard order of operations. To change the order, enclose in parentheses the part of the expression to be resolved first. For example, the following expression gives a result of 11 because multiplication is resolved before addition:

```
5+2*3
```

If you enclose the addition in parentheses, so that it is resolved before the multiplication, the expression gives a result of 21:

```
(5+2)*3
```

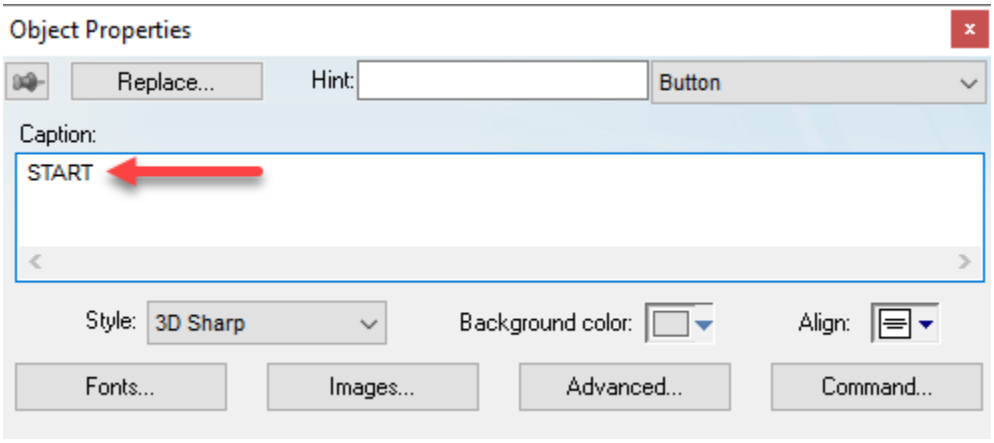
For more complex math, try the Arithmetic, Statistical, Logarithmic and Trigonometric functions.

## String expressions

A string expression is a way of specifying a dynamic value (i.e., a value that can change during project run time) instead of a static value for a setting or property that requires a string.

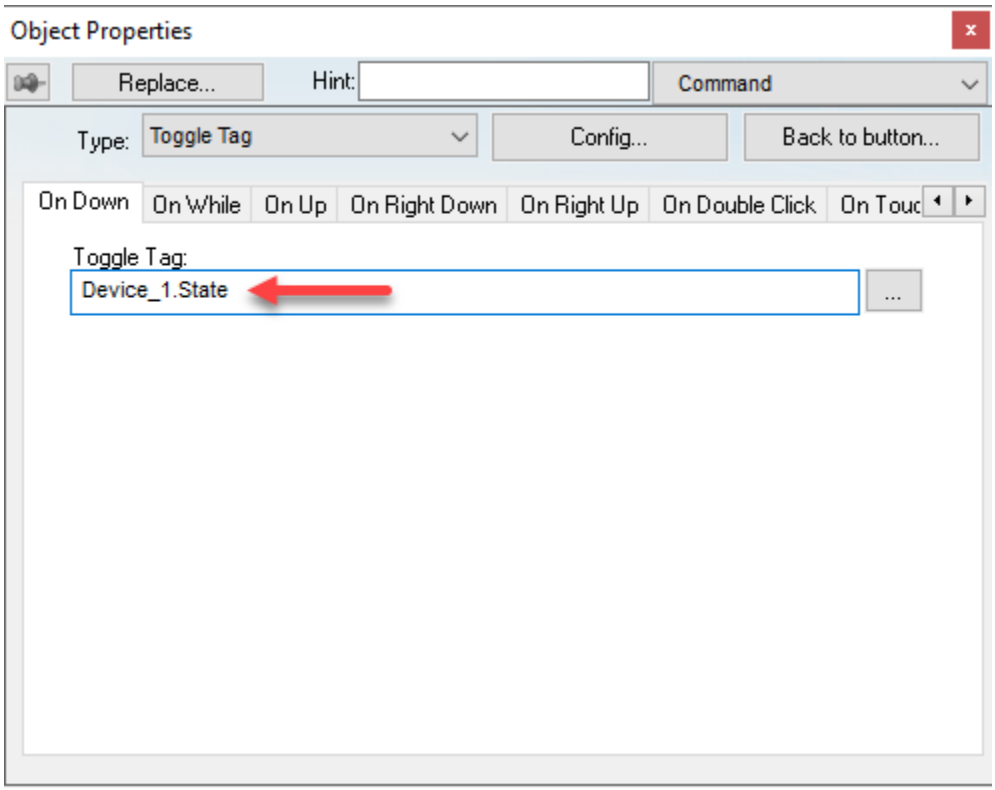
There are many settings and properties throughout a project that require strings. One of the most common is the caption on a Button object. When you insert a Button object in a project screen, you can edit its properties in order to add a caption, and that caption must be a string of some kind. Even if the caption consists entirely of numerals, those numerals are handled and displayed as a string rather than as an integer or real number.

An example of a static value for the caption might be **START**, on a button that starts a device:




This works well enough if you have separate buttons for **START** and **STOP**, but you might want to have a single button with a caption that changes depending on the current state of the device: **START** when the device is stopped, and **STOP** when the device is started. In other words, you might want to specify a dynamic value for the caption.

To do this, you must first select a tag or compose an expression that determines the value. In this example, the actual purpose of the button is to start and stop the device, so the object is probably configured to toggle the value of a tag (e.g., **Device_1.State**) when the button is clicked/tapped:

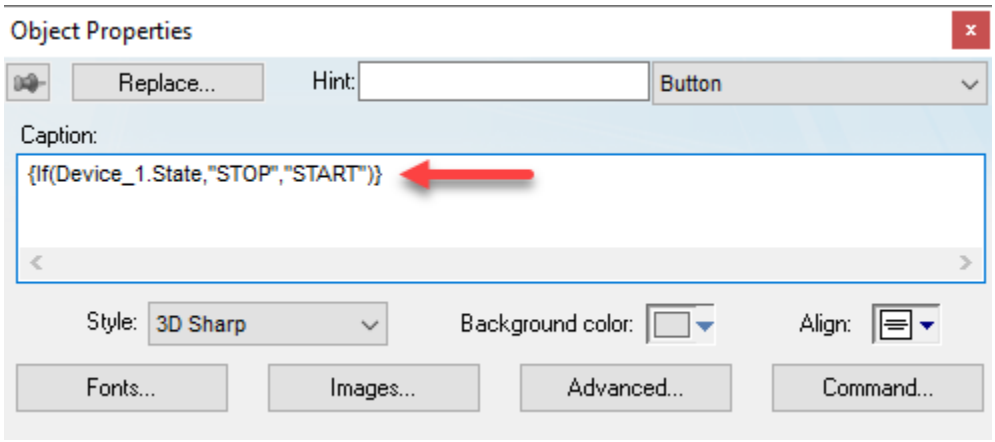



The value toggles between two possible values: 0 (FALSE) for stopped, and 1 (TRUE) for started. Therefore, you need to compose an expression that gets the value and then determines the appropriate caption. There are many ways to do this using the built-in language, so here is just one example that uses the `If` function:

```
If(Device_1.State,"STOP","START")
```

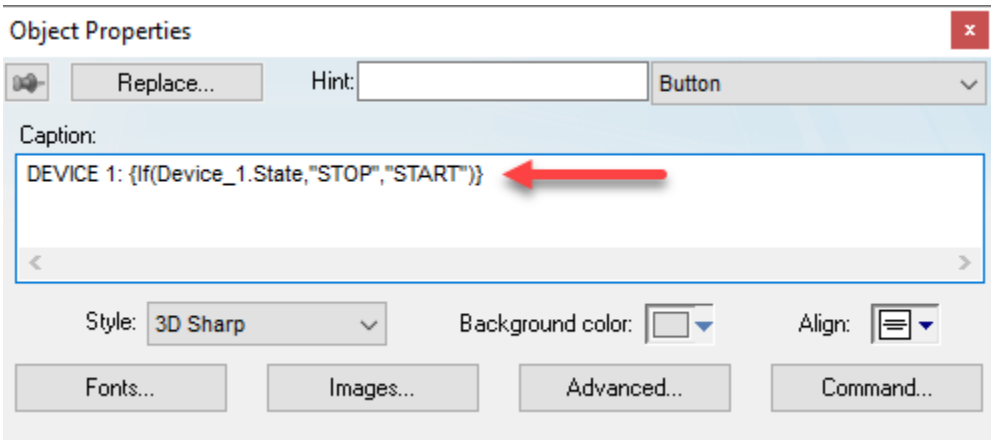
Please note that in this example, you are not using the `If` function to determine the current state of the device. You are using it to determine what the caption should be given the current state. That is why the strings seem to be reversed: **`"STOP"`** if the value is TRUE (i.e., the device is started), and **`"START"`** if the value is FALSE (i.e., the device is stopped).

Now, if you type this example without additional formatting in the **Caption** box, it will be displayed as a literal string because there is nothing to indicate it is an expression that should be evaluated before it is displayed. To indicate that, you must enclose the expression in braces (**`{}`**):



When the project screen is opened and the button is displayed, the string expression is evaluated and the caption is determined.

Taking this a step further, you can combine a string expression with plain text to form a longer string. Only the string expression — that is, only the contents of the braces — will be evaluated before it is displayed. The plain text will be displayed as is. For example:




Keep in mind that this method of enclosing a string expression in braces only works for settings and properties that normally require strings. There are other settings and properties (e.g., triggers) that explicitly require expressions, and those expressions should not be enclosed in braces. The description of each setting or property should make clear which type of input it accepts.

> **Note:** Braces are sometimes also known colloquially as "curly brackets", but that is not correct in the context of programming languages. Any occurences of the term "curly brackets" that appear elsewhere in this documentation are artifacts that will be corrected or removed as we continue to revise this documentation.

## How to read function descriptions

This is a key to reading the descriptions of the built-in functions.

Each function description is broken into several sections.

### Function attributes

Every function has certain attributes that are described in a single-row table:

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| *function name* | *group name* | *synchronous or asynchronous* | *yes or no* | *supported or not supported* | *supported or not supported* | *supported or not supported* |

First, obviously, the exact name of the function as it should be used in your project.

Next, the functions are organized into groups according to the type of calculcation they perform or the part of your project upon which they act. You can use the group names to find the functions you want in the Object Finder and in this documentation.

Next, the execution of the function is either synchronous or asynchronous:

**Synchronous**

When the function is executed on either the project server or the project client, that station requires some response or acknowledgement from the other. The project pauses, however briefly, while it waits for the response. In other words, the server and client must remain synchronized.

This is normally not an issue because most functions are executed almost instantly, but if a client makes unusually frequent function calls or your network is slow, your project may suffer decreased performance.

**Asynchronous**

The function can be executed on either the project server or the project client without waiting for the other. The project continues to run without interruption.

Next, the function either can (yes) or cannot (no) be used in string expressions.

Finally, the function is either supported or unsupported on each target platform:

**Windows**

Projects running in the full EcoStruxure Machine SCADA Expert 2020 software on a Windows or Windows Server computer, when the software is licensed and used as a project runtime server. For more information, see About the EcoStruxure Machine SCADA Expert 2020 software components on page 34.

**Thin Client**

Project screens viewed with the Thin Client software for Windows. Specifically, if the function is supported, it can be called in a project screen on the client station — for example, when a Command animation is triggered or the Screen Script is run. For more information, see Thin Clients on page 729.

**Mobile Access**

Project screens viewed in the browser via the Mobile Access web interface. For more information, see Mobile Access on page 756.

> **Note:** Some functions are described as "Executed on Server", which means that when the function is called in a project screen on a thin client, it is actually executed on the project runtime server using the server's local settings and resources. This most often applies to functions that perform date/time, file, and database operations.

### Syntax diagram and parameters

A basic syntax diagram shows how the function should be entered and what parameters it takes.

In most cases, a parameter can take either a literal value or the name of a project tag that contains the value. The data type of the parameter is indicated by its prefix:

**bool**

The parameter can take either a literal Boolean value *or* the name of a Boolean tag. For example, either `0` or `MyBoolTag`.

**num**

The parameter can take either a literal numeric value *or* the name of an Integer or Real tag. For example, either `45.6543` or `MyNumTag`.

**str**

The parameter can take either a text string enclosed in quotation marks *or* the name of a String tag. For example, either `"My string"` or `MyStrTag`.

The additional prefix opt indicates that a parameter is optional. If you do not specify a value for the parameter, the function will take the default value mentioned in the parameter description.

In the few cases where a parameter must take a project tag or some other special input, it will be fully explained in the parameter description.

### Return value

This section describes the value returned by the function, if any.

Some functions return a calculated value, depending on the nature of the function.

Other functions return an error code that indicates how well the function was executed. The possible codes and their meanings are provided in a table.

### Notes

This section describes any additional notes or cautions on the use of the function.

### Examples

This section shows how the function can be called in your project. Multiple examples are provided to show how the function can take both literal values and project tags, as well as how the function may be called if it has optional parameters.

## List of available functions

This is a complete list of the built-in functions that are available for use in scripts and expressions.

### ActiveX and .NET Control

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| XGet | Asynchronous | Yes | Supported | Supported | Not supported |
| XRun | Asynchronous | No | Supported | Supported | Not supported |
| XSet | Asynchronous | No | Supported | Supported | Not supported |

### Arithmetic

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| Abs | Synchronous | Yes | Supported | Supported | Supported |
| Div | Synchronous | Yes | Supported | Supported | Supported |
| Format | Synchronous | Yes | Supported | Supported | Supported |
| GetBit | Synchronous | Yes | Supported | Supported | Supported |
| Mod | Synchronous | Yes | Supported | Supported | Supported |
| ResetBit | Synchronous | No | Supported | Supported | Supported |
| SetBit | Synchronous | No | Supported | Supported | Supported |

### Database/ERP

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| DBCursorClose | Synchronous | No | Supported | Supported | Executed on Server |
| DBCursorColumnCount | Synchronous | Yes | Supported | Supported | Executed on Server |
| DBCursorColumnInfo | Synchronous | Yes | Supported | Supported | Executed on Server |
| DBCursorCurrentRow | Synchronous | Yes | Supported | Supported | Executed on Server |
| DBCursorGetValue | Synchronous | Yes | Supported | Supported | Executed on Server |
| DBCursorMoveTo | Synchronous | No | Supported | Supported | Executed on Server |
| DBCursorNext | Synchronous | No | Supported | Supported | Executed on Server |
| DBCursorOpen | Synchronous | No | Supported | Supported | Executed on Server |
| DBCursorOpen2 | Synchronous | No | Supported | Supported | Executed on Server |
| DBCursorPrevious | Synchronous | No | Supported | Supported | Executed on Server |
| DBCursorRowCount | Synchronous | Yes | Supported | Supported | Executed on Server |
| DBDelete | Synchronous | No | Supported | Supported | Executed on Server |

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| DBExecute | Synchronous | No | Supported | Supported | Executed on Server |
| DBInsert | Synchronous | No | Supported | Supported | Executed on Server |
| DBSelect | Synchronous | No | Supported | Supported | Not supported |
| DBUpdate | Synchronous | No | Supported | Supported | Executed on Server |

### Date & Time

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| ClockGetDate | Synchronous | Yes | Supported | Supported | Executed on Server |
| DateTime2UTC | Synchronous | Yes | Supported | Supported | Executed on Server |
| GetClock | Synchronous | Yes | Supported | Supported | Executed on Server |
| GetTimeZone | Synchronous | Yes | Supported | Supported | Executed on Server |
| GetTimeZoneExtra | Synchronous | Yes | Supported | Supported | Executed on Server |
| GetUTC | Synchronous | Yes | Supported | Supported | Executed on Server |
| SetTimeZone | Synchronous | No | Supported | Supported | Not supported |
| UTC2DateTime | Synchronous | Yes | Supported | Supported | Executed on Server |

### Email

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| CnfEmail | Synchronous | No | Supported | Supported | Executed on Server |
| GetStatusSendEmailExt | Synchronous | Yes | Supported | Supported | Not supported |
| SendEmail | Synchronous | No | Supported | Supported (see notes) | Executed on Server |
| SendEmailExt | Asynchronous | No | Supported | Supported | Not supported |

### Event Logger

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| SendEvent | Synchronous | No | Supported | Supported | Supported |

### File

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| DeleteOlderFiles | Synchronous | No | Supported | Executed on Server (see notes) | Not supported |

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| FileDelete | Synchronous | No | Supported | Executed on Server (see notes) | Not supported |
| FileReadFields | Synchronous | Yes | Supported | Not supported | Not supported |
| FileReadMessage | Synchronous | Yes | Supported | Not supported | Not supported |
| FileRename | Synchronous | No | Supported | Executed on Server (see notes) | Not supported |
| FileWrite | Synchronous | No | Supported | Executed on Server (see notes) | Supported |
| FileWriteFields | Synchronous | No | Supported | Not supported | Not supported |
| FileWriteMessage | Synchronous | No | Supported | Not supported | Not supported |
| FindFile | Synchronous | Yes | Supported | Executed on Server (see notes) | Not supported |
| FindPath | Synchronous | Yes | Supported | Executed on Server (see notes) | Not supported |
| GetFileTime | Synchronous | Yes | Supported | Executed on Server (see notes) | Not supported |
| GetLine | Synchronous | Yes | Supported | Supported | Supported |
| HST2TXT | Asynchronous | No | Supported | Executed on Server (see notes) | Not supported |
| ImportXML | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| LookupContains | Synchronous | Yes | Supported | Not supported | Not supported |
| LookupGet | Synchronous | Yes | Supported | Not supported | Not supported |
| LookupLoad | Synchronous | Yes | Supported | Not supported | Not supported |
| Print | Asynchronous | No | Supported | Supported | Not supported |
| WebGetFile | Synchronous | No | Supported | Supported | Not supported |

## FTP

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| CnfFTP | Synchronous | No | Supported | Supported | Not supported |
| FTPGet | Asynchronous | No | Supported | Supported (see notes) | Not supported |
| FTPPut | Asynchronous | No | Supported | Supported (see notes) | Not supported |
| FTPStatus | Synchronous | Yes | Supported | Supported (see notes) | Not supported |

## Graphic

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| SaveScreenShot | Synchronous | No | Supported | Supported (see notes) | Not supported |

### Log Message

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| Trace | Synchronous | No | Supported | Not supported | Supported (see notes) |

### Logical

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| False | Synchronous | Yes | Supported | Supported | Supported |
| If | Synchronous | Yes | Supported | Supported | Supported |
| True | Synchronous | Yes | Supported | Supported | Supported |

### Loop

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| For | N/A | No | Supported | Supported | Not supported |

### Module Activity

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| CleanReadQueue | Synchronous | No | Supported | Supported | Executed on Server |
| DisableMath | Asynchronous | No | Supported | Supported | Executed on Server |
| EnableMath | Asynchronous | No | Supported | Supported | Executed on Server |
| EndTask | Asynchronous | No | Supported | Executed on Server | Executed on Server |
| Exec | Synchronous or Asynchronous | No | Supported | Supported | Not supported |
| ExecIsRunning | Synchronous | Yes | Supported | Supported | Not supported |
| IsTaskRunning | Synchronous | Yes | Supported | Executed on Server | Executed on Server |
| KeyPad | Asynchronous | No | Supported | Supported | Supported |
| LogOn | Asynchronous | No | Supported | Supported | Supported |
| Math | Synchronous | No | Supported | Supported | Executed on Server |
| PostKey | Synchronous | No | Supported | Supported | Not supported |
| Recipe | Synchronous | No | Supported | Executed on Server (see notes) | Executed on Server (see notes) |
| Report | Synchronous | No | Supported | Supported | Executed on Server |
| RunGlobalProgramAsyncGetCurrent | Synchronous | No | Supported | Not supported (see notes) | Not supported |
| SecureViewerShow | Synchronous | No | Not supported | Secure Viewer only | Not supported |
| SetViewerInputs | Synchronous | No | Supported | Supported | Not supported |
| SetViewerPos | Synchronous | No | Supported | Supported | Not supported |

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| StartTask | Asynchronous | No | Supported | Executed on Server | Executed on Server |
| TaskUpdateConfig | Asynchronous | No | Supported | Not supported | Not supported |

## Screen

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| Close | Asynchronous | No | Supported | Supported | Supported |
| Open | Asynchronous | No | Supported | Supported | Supported (see notes) |
| OpenPrevious | Asynchronous | No | Supported | Supported | Supported |
| ShowInplaceApp | Asynchronous | No | Supported | Supported | Not supported |
| ShowMessageBox | Synchronous | No | Supported | Supported | Supported |

## Security

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| CheckESign | Synchronous | No | Supported | Supported | Supported |
| CheckSecurityControl | Synchronous | Yes | Supported | Supported | Supported |
| CreateUser | Synchronous | No | Supported | Supported | Not supported |
| GetLastESignUser | Synchronous | Yes | Supported | Supported | Supported |
| GetUserName | Synchronous | No | Supported | Executed on Server | Not supported |
| GetUserPwdAsync | Synchronous | Yes | Supported | Supported | Not supported |
| RemoveUser | Synchronous | No | Supported | Supported | Not supported |
| SetPassword | Synchronous | No | Supported | Supported | Not supported |
| SetUserGroup | Synchronous | No | Supported | Supported | Not supported |

## Statistical

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| Avg | Synchronous | Yes | Supported | Supported | Supported |
| Max | Synchronous | Yes | Supported | Supported | Supported |
| Min | Synchronous | Yes | Supported | Supported | Supported |

## String

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| CharToValue | Synchronous | No | Supported | Supported | Supported |
| CharToValueEx | Synchronous | No | Supported | Supported | Supported |
| DecryptData | Synchronous | Yes | Supported | Supported | Executed on Server |
| EncryptData | Synchronous | Yes | Supported | Supported | Executed on Server |
| StrCompare | Synchronous | Yes | Supported | Supported | Supported |
| StrCompareNoCase | Synchronous | Yes | Supported | Supported | Supported |

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| StrFromInt | Synchronous | Yes | Supported | Supported | Supported |
| StrFromReal | Synchronous | Yes | Supported | Supported | Supported |
| StrSetElement | Synchronous | No | Supported | Supported | Supported |
| StrUpper | Synchronous | Yes | Supported | Supported | Supported |
| ValueToChar | Synchronous | Yes | Supported | Supported | Supported |
| ValueWToChar | Synchronous | Yes | Supported | Supported | Supported |

## System Info

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| DBVersion | Synchronous | Yes | Supported | Supported | Executed on Server |
| GetAppHorizontalResolution | Synchronous | Yes | Supported | Supported | Executed on Server |
| GetAppVerticalResolution | Synchronous | Yes | Supported | Supported | Executed on Server |
| GetOS | Synchronous | Yes | Supported | Supported | Not supported |
| GetPerformanceMetric | Synchronous | Yes | Supported | Supported | Not supported |
| GetProductPath | Synchronous | Yes | Supported | Supported | Executed on Server |
| GetServerHostName | Synchronous | Yes | Not supported | Supported | Executed on Server |
| InfoAppAlrDsy | Synchronous | Yes | Supported | Supported | Executed on Server |
| InfoAppHstDsy | Synchronous | Yes | Supported | Supported | Executed on Server |
| NoInputTime | Synchronous | Yes | Supported | Keyboard input only | Not supported |
| ProductVersion | Synchronous | Yes | Supported | Supported | Executed on Server |
| SetDateFormat | Synchronous | No | Supported | Supported | Not supported |
| SetKeyboardLanguage | Synchronous | No | Supported | Supported | Not supported |
| WritePrivateProfileString | Synchronous | No | Supported | Supported | Executed on Server |

## Tags Database

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| ExecuteAlarmAck | Synchronous | No | Supported | Supported | Executed on Server |
| GetAlarmCount | Synchronous | Yes | Supported | Not supported | Not supported |
| GetAlarmInfo | Synchronous | Yes | Supported | Not supported (see notes) | Not supported (see notes) |
| GetTagValue | Synchronous | Yes | Supported | Supported | Supported |
| TagsDBAddClass | Synchronous | No | Supported (see notes) | Not supported | Not supported |

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| TagsDBAddClassMember | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBAddTag | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBBeginEdit | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBEndEdit | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBGetAllTags | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetClassMember | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetClassMemberCount | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetFirstClass | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetFirstClassMember | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetFirstTag | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetLoadedClass | Synchronous | Yes | Not supported | Supported | Not supported |
| TagsDBGetNextClass | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetNextClassMember | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetNextTag | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetPreloadCount | Synchronous | Yes | Not supported | Supported | Not supported |
| TagsDBGetTagCount | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetTagProperty | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBGetTree | Synchronous | Yes | Supported (see notes) | Not supported | Not supported |
| TagsDBPreload | Synchronous | No | Not supported | Supported | Not supported |
| TagsDBPreloadAdd | Synchronous | No | Not supported | Supported | Not supported |
| TagsDBRemoveClass | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBRemoveClass2 | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBRemoveClassMember | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBRemoveTag | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBRemoveTag2 | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBSetAllTags | Synchronous | No | Supported (see notes) | Not supported | Not supported |

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| TagsDBSetTagProperty | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBSetTrend | Synchronous | No | Supported (see notes) | Not supported | Not supported |
| TagsDBSync | Synchronous | No | Not supported | Supported | Not supported |

## Translation

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| Ext | Synchronous | Yes | Supported | Supported | Supported |
| SetLanguage | Synchronous | No | Supported | Supported | Supported |
| Translation | Synchronous | No | Supported | Not supported (see notes) | Executed on Server (see notes) |
| TranslationTmpClose | Synchronous | No | Supported | Not supported | Supported |
| TranslationTmpGet | Synchronous | Yes | Supported | Not supported | Supported |
| TranslationTmpLoad | Synchronous | No | Supported | Not supported (see notes) | Executed on Server (see notes) |

## Trigonometric

| Function | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|
| Cot | Synchronous | Yes | Supported | Supported | Supported |
| Tan | Synchronous | Yes | Supported | Supported | Supported |

## ActiveX and .NET Control functions

These functions are used to directly run ActiveX and .NET Control objects in the project, as well as to get and set property values on those objects.

### *XGet*

The function `XGet` gets the current value of a Property on an ActiveX Control or .NET Control object.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `XGet` | ActiveX and .NET Control | Asynchronous | Yes | Supported | Supported | Not supported |

#### Syntax

```
XGet(strName,strProperties)
```

`XGet(`*strName*`,`*strProperties*`)`

***strName***

The unique name of the ActiveX Control or .NET Control object, as specified in the **Name** box in the *Object Properties* dialog box.

***strProperties***

The Property for which you want to get the value. Available Properties are listed in the *Configuration* (for an ActiveX Control) or *Members* (for a .NET Control) dialog box.

#### Returned value

This function returns the value of the specified Property.

#### Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

Also, this function is not supported for .NET Control objects running on a Windows Embedded device.

#### Examples

Get the current value of the Color property on the ActiveX Control object named "ActXRec":

```
XGet("ActXRec","Color")
```

### *XRun*

The function `XRun` runs a Method on an ActiveX Control or .NET Control object.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `XRun` | ActiveX and .NET Control | Asynchronous | No | Supported | Supported | Not supported |

#### Syntax

```
XRun(strName,strMethod,Parameter1,…,ParameterN)
```

XRun(*strName*, *strMethod*, *Parameter1*, *ParameterN*)

***strName***

The unique name of the ActiveX Control or .NET Control object, as specified in the **Name** box in the *Object Properties* dialog box.

***strMethod***

The Method that you want to run. Available Methods are listed in the *Configuration* (for an ActiveX Control) or *Members* (for a .NET Control) dialog box.

***Parameter(1...N)***

Data of various types that are required by the Method to run. The number of parameters can range from 0 to 255 and depends on the specified Method. The data types (e.g., Boolean, Integer, Real or String) of referring tags must match the parameters on the Method.

### Returned value

This function returns the Method result as reported by the ActiveX Control or .NET Control object. Not all Methods return results.

### Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

Also, this function is not supported for .NET Control objects running on a Windows Embedded device.

### Examples

Run the XPos method on the ActiveX Control named "ActXCir," with four original values passed to the method:

```
XRun("ActXCir","XPos",FALSE,12,4.6,"This is my text.")
```

Run the XPos method on the ActiveX Control named "ActXCir," with four referring tags passed to the method:

```
XRun("ActXCir","XPos",TagA,TagB,TagC,TagD)
```

## *XSet*

The function XSet sets the value of a Property on an ActiveX Control or .NET Control object.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| XSet | ActiveX and .NET Control | Asynchronous | No | Supported | Supported | Not supported |

### Syntax

```
XSet(strName,strProperties,Value)
```

XSet(*strName*, *strProperties*, *Value*)

***strName***

The unique name of the ActiveX Control or .NET Control object, as specified in the **Name** box in the *Object Properties* dialog box.

***strProperties***

The Property that you want to set the value of. Available Properties are listed in the *Configuration* (for an ActiveX Control) or *Members* (for a .NET Control) dialog box.

***Value***

A tag, expression, or data value of any type; the value to which you want to set the Property.

### Return value

This function returns no value.

### Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

Also, this function is not supported for .NET Control objects running on a Windows Embedded device.

### Examples

Set the value of the Display property on the ActiveX Control named "ActXDisplay" to "Status Normal":

```
XSet("ActXDisplay","Display","Status Normal")
```

## Arithmetic functions

These functions are used to perform advanced arithmetic operations and bit manipulation on numeric values.

### *Abs*

`Abs` is a built-in function that gets the absolute value of a specified numeric value.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `Abs` | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

```
Abs(numValue)
```

`Abs(`*numValue*`)`

***numValue***

The numeric value of which the absolute value will be gotten.

#### Returned value

The absolute value of the specified numeric value.

#### Examples

Get the absolute value of -54.9788:

```
Abs(-54.9788)
```

Get the absolute value of the numeric value stored in a project tag:

```
Abs(MyReal)
```

### *Div*

This function returns the dividend of two whole numbers.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `Div` | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

```
Div(numNumerator,numDenominator)
```

`Div(`*numNumerator*`,` *numDenominator*`)`

***numNumerator***

The numerator of the division operation. Please note that if you specify a decimal value, then it will be truncated.

***numDenominator***

The denominator of the division operation. Please note that if you specify a decimal value, then it will be truncated.

### Return value

This function returns the dividend only as a whole number. The remainder is omitted.

> **Tip:** To get the remainder instead of the dividend, use the function Mod.

**Examples**

| Tag Name | Expression |
|---|---|
| **numValue** | **Div( 100, 8 )** // Returns the value 12 |
| **numValue** | **Div( 16, 4 )** // Returns the value 4 |
| **numValue** | **Div( 100, 2.5 )** // Returns the value 50 |

## *Format*

Format is a built-in scripting function that formats a numeric value and returns it as a string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Format | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

**Format(*strFlag*,*numValue*,*optStrDecimalMark*,*optStrThousandSep*)**

Format(*strFlag*, *numValue*{ | , *optStrDecimalMark*{ | , *optStrThousandSep* } })

***strFlag***

A description of how the given numeric value should be formatted, according to the syntax *%length.precisionFormat*, where:

- *length* is the minimum number of characters to be returned by the function — that is, the minimum length of the resulting string. If the value to be returned is shorter than this, it is padded with either blank spaces (**" "**) or zeroes (**"0"**); see "Examples" below. The value is not truncated even if the result is longer than the specified length. Applicable to formats d, x, X, o, b, f, e, E, g, G, s, c, and h.
- *.precision* is the number of decimal places for a floating-point number. Applicable for formats **f**, e, E, g, and G.
- *Format* is the specific format:

| Format | Description |
|---|---|
| d | Decimal |
| x | Hexadecimal (alphabetic characters in lowercase) |
| X | Hexadecimal (alphabetic characters in uppercase) |
| o | Octal |
| b | Binary |
| f | Floating-point |
| e | Scientific notation (e in lowercase) |
| E | Scientific notation (E in uppercase) |
| g | Rounded, in scientific notation when applicable (e in lowercase) |

| Format | Description |
|---|---|
| G | Rounded, in scientific notation when applicable (E in uppercase) |
| s | String (i.e., no change in number) |
| c | ASCII character (i.e., the numeric value is interpreted as an ASCII character code) |
| h | Hour (hh:mm:ss) |

Alternatively, the format can be set using the syntax ##.###, where the numeric value is rounded to the specified number of decimal places.

***numValue***

The numeric value to be formatted.

***optStrDecimalMark***

The character used as the decimal mark, which separates the integer and fractional parts of the numeric value.

This is an optional parameter; if no value is specified, then the default is a period (.). For example: **"123.45"**

***optStrThousandSep***

The character used as the thousands separator, which separates the *hundreds* and *thousands* digits of the numeric value.

This is an optional parameter; if no value is specified, then the default is a comma (,). For example: **"12,345"**

## Returned value

This function returns a string that contains the formatted numeric value. See "Examples" below.

## Notes

Format is similar to the printf function in other programming languages, and it allows most of the same formatting options. However, unlike printf, Format can be used to format only one numeric value at a time.

This function is particularly useful for formatting values to be printed in reports.

## Examples

| Expression | Returned value |
|---|---|
| Format("%d",12.34) | 12 |
| Format("%04d",12.34) | 0012 |
| Format("%4d",12.34) | 12 |

| Expression | Returned value |
|---|---|
| Format("%x",26) | 1a |
| Format("%04x",26) | 001a |
| Format("%4x",26) | 1a |

| Expression | Returned value |
|---|---|
| Format("%X",26) | 1A |
| Format("%04X",26) | 001A |
| Format("%4X",26) | 1A |

| Expression | Returned value |
|---|---|
| Format("%o",16) | 20 |
| Format("%04o",16) | 0020 |
| Format("%4o",16) | 20 |

| Expression | Returned value |
|---|---|
| Format("%b",2) | 10 |
| Format("%4b",2) | 0010 |
| Format("%04b",2) | 0010 |

| Expression | Returned value |
|---|---|
| Format("%0.1f",12.34) | 12.3 |
| Format("%06.1f",12.34) | 0012.3 |
| Format("%6.1f",12.34) | 12.3 |

| Expression | Returned value |
|---|---|
| Format("%e",12.34) | 1.234000e+001 |
| Format("%0.1e",12.34) | 1.2e+001 |
| Format("%09.1e",12.34) | 01.2e+001 |
| Format("%9.1e",12.34) | 1.2e+001 |

| Expression | Returned value |
|---|---|
| Format("%E",12.34) | 1.234000E+001 |
| Format("%0.1E",12.34) | 1.2E+001 |
| Format("%09.1E",12.34) | 01.2E+001 |
| Format("%9.1E",12.34) | 1.2E+001 |

| Expression | Returned value |
|---|---|
| Format("%0.1g",12.34) | 1e+001 |
| Format("%0.2g",12.34) | 12 |
| Format("%0.3g",12.34) | 12.3 |
| Format("%05.3g",12.34) | 012.3 |
| Format("%5.3g",12.34) | 12.3 |

| Expression | Returned value |
|---|---|
| Format("%0.1G",12.34) | 1E+001 |
| Format("%0.2G",12.34) | 12 |
| Format("%0.3G",12.34) | 12.3 |
| Format("%05.3G",12.34) | 012.3 |
| Format("%5.3G",12.34) | 12.3 |

| Expression | Returned value |
|---|---|
| Format("%s",12.34) | 12 |

| Expression | Returned value |
|---|---|
| Format("%04s",12.34) | 0012 |
| Format("%4s",12.34) | 12 |

| Expression | Returned value |
|---|---|
| Format("%c",97) | a |
| Format("%4c",97) | a |
| Format("%04c",97) | 000a |

| Expression | Returned value |
|---|---|
| Format("%h",30) | 00:00:30 |
| Format("%h",60) | 00:01:00 |
| Format("%h",90) | 00:01:30 |
| Format("%h",3600) | 01:00:00 |

| Expression | Returned value |
|---|---|
| Format("##.#",26.56789) | 26.6 |
| Format("#.##",26.56789) | 26.57 |
| Format("##.##",26.56789) | 26.57 |

## *GetBit*

GetBit is a built-in function that gets a specific bit in the value of a project tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetBit | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

**GetBit("*tagName*",*numBitNumber*)**

GetBit("*tagName*",*numBitNumber*)

***tagName***

The name of the project tag.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***numBitNumber***

The number or position (0...31) of the bit to get.

### Return value

If this function is executed successfully, it returns the value — either 0 or 1 — of the specified bit. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| 2 | Specified tag does not exist. |

### Notes

You can also reference the Bit property of a project tag in order to get/set a specific bit in the value of that tag. For more information, see Reference a tag property instead of a project tag on page 169.

### Examples

Get the bit in the fifth position (i.e., position 4) of **MyInteger**:

```
GetBit("MyInteger",4)
```

If the value of **MyInteger** is 14, the function returns 0. If the value of **MyInteger** is 19, the function returns 1.

## *Mod*

Mod is a built-in scripting function that gets the remainder from a division operation.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Mod | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Mod( numNumerator, numDenominator )
```

**numNumerator**

Integer or Real tag containing the Numerator of the function.

**numDenominator**

Integer or Real tag containing the Denominator of the function.

### Return value

Returns the remainder (as a real number) after dividing **numNumerator** by **numDenominator**.

> **Tip:** Use the Div function to get the whole number dividend of the operation.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **Mod( 50, 4 )** // Returned value = 2 |
| Tag | **Mod( 16, 4 )** // Returned value = 0 |
| Tag | **Mod( 100, 8.2 )** // Returned value = 1.600 |

## *Pow*

Pow is a built-in scripting function that gets the result of raising a numeric value to a specified exponent.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Pow | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Pow( numBase, numExponent )
```

**numBase**

Integer or Real tag containing the Base of the function.

**numExponent**

Integer or real tag containing the Exponent of the function.

### Returned value

Returns the result of raising the base to the exponent.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `Pow( 2, 3 )` // Returned value = 8 |
| Tag | `Pow( 10, 4 )` // Returned value = 10000 |

## *ResetBit*

Resets a single bit in an Integer tag to 0.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ResetBit | Arithmetic | Synchronous | No | Supported | Supported | Supported |

### Syntax

```
ResetBit(tagName,numBitNumber)
```

**tagName**

The name of an Integer tag where the bit value will be reset.

> **Note:** To directly specify the name of a tag, rather than take the value of the tag, you must enclose the tag name in double-quotes. For example, `ResetBit( "Second", 1 )`.

**numBitNumber**

A numeric tag or value specifying the position (0…31) of the bit to reset.

### Return value

| | |
|---|---|
| 0 | No error |
| 1 | Invalid parameter |
| 2 | Tag does not exist |

### Notes

You can use the Bit field to read/write values from specific bits in an integer tag. For example, enter `Second->b0` to access the LSB (Least Significant Bit of the Second tag), and `Second->b31` to access the MSB (Most Significant Bit of the Second tag).

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `ResetBit("numSource",4)` // If the tag `numSource` held a value of 16, then the function returns 0 and `numSource` holds a new value of 0. |

| Tag Name | Expression |
|---|---|
| Tag | **ResetBit("numSource",1)** // If the tag **numSource** held the value 19, then the function returns 0 and **numSource** holds a new value of 17. |

## *Round*

Rounds **numValue** to the nearest integer.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Round | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

**Round( *numValue* )**

**numValue**

A Real tag that holds the value to be rounded.

### Returned value

Returns the integer result of the round function.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **Round( "345.87" )** // Returned value = 346 |
| Tag | **Round( "65.323" )** // Returned value = 65 |

## *SetBit*

SetBit is a built-in function that sets (i.e., toggles) a specific bit in the value of a project tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetBit | Arithmetic | Synchronous | No | Supported | Supported | Supported |

### Syntax

**SetBit("*tagName*",*numBitNumber*)**

SetBit("*tagName*",*numBitNumber*)

***tagName***

The name of the project tag.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***numBitNumber***

The number or position (0…31) of the bit to set.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Success (i.e., no error). |
| 1 | Invalid parameter. |
| 2 | Specified tag does not exist. |

### Notes

You can also reference the Bit property of a project tag in order to get/set a specific bit in the value of that tag. For more information, see Reference a tag property instead of a project tag on page 169.

### Examples

Set the bit in the fifth position (i.e., position 4) of **MyInteger**:

```
SetBit("MyInteger",4)
```

If the old value of **MyInteger** was 14, the new value is 30. If the old value of **MyInteger** is 19, the new value is 3.

## *Sqrt*

Takes the square root of **numValue**.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Sqrt | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Sqrt( numValue )
```

**numValue**

Integer or Real tag to be square rooted.

### Returned value

Returns the square root of the value in the **numValue** tag.

> **Note:** If **numValue** has a negative value, then this function returns the value 0 and sets the quality of the returned tag to **BAD**.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **SQRT( 25 )** // Returns the value **5** |
| Tag | **SQRT( 67 )** // Returns the value **8.185353** |

## *Swap16*

Swaps the two lower bytes of a tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Swap16 | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Swap16( numValue )
```

**numValue**

Integer tag that holds the numeric value of the bytes to be swapped.

### Returned value

Returns the numeric value after swapping the bytes.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **Swap16( 16 )** // 16 = 0000000000010000 in binary. Returned value = 4096 = 0001000000000000 in binary. |
| Tag | **Swap16( 43760 )** // 43760 = 1010010111110000 in binary. Returned value = 61610 = 1111000010100101 in binary. |

## *Swap32*

Swaps two words in a tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Swap32 | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Swap32( numValue )
```

**numValue**

Integer tag that holds the numeric value of the words to be swapped.

### Returned value

Returns the numeric value after swapping the words.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **Swap32( 16 )** // 16 = 00000000000000000000000000010000 in binary. Returned value = 1048576 = 00000000000100000000000000000000 in binary. |
| Tag | **Swap32( 246333120 )** // 286333120 = 10101010101010101111111100000000 in binary. Returned value = -1094709586= 11111111000000001010101010101010 in binary. |

## *Trunc*

Truncates the value of ***numValue***.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Trunc | Arithmetic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Trunc( numValue )
```

**numValue**

Real tag to be truncated.

### Returned value

Returns the integer portion of the real number value of ***numValue***.

### Examples

| Tag Name | Expression |
|---|---|
| | **Trunc( 234.987 )** // Returned value = 234 |
| | **Trunc( -3465.9 )** // Returned value = -3465 |

## Database/ERP functions

These functions are used interact with external databases and ERP systems using SQL-like commands.

### *DBCursorClose*

`DBCursorClose` is a built-in function that closes an open database cursor and releases the SQL result set.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `DBCursorClose` | Database/ERP | Synchronous | No | Supported | Supported | Executed on Server |

#### Syntax

**`DBCursorClose(`*`numCur`*`,"`*`optStrErrorTag`*`")`**

`DBCursorClose(`*numCur*{ | `,"`*optStrErrorTag*`"` })

***numCur***

The cursor handle for the result set, which was returned by `DBCursorOpen` or `DBCursorOpenSQL`.

***optStrErrorTag***

The name of a project tag that will receive detailed error messages, if errors occur during project run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

#### Return value

In the case of success, this function returns 0.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |

| Value | Error Message |
|---|---|
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

Database/ERP functions emulate Structured Query Language (SQL) database operations. Before you use these functions, you should be familiar with how SQL statements are formed and executed.

When a cursor is closed, it is destroyed and cannot be used again. You must open a new cursor by calling either DBCursorOpen or DBCursorOpenSQL.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

```
DBCursorClose(nCursor)
```

```
DBCursorClose(nCursor,"TagError")
```

## *DBCursorColumnCount*

DBCursorColumnCount is a built-in function that gets the total number of columns in a SQL result set.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBCursorColumnCount | Database/ERP | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
DBCursorColumnCount(numCur,"optStrErrorTag")
```

DBCursorColumnCount(*numCur*{ | ,"*optStrErrorTag*" })

***numCur***

The cursor handle for the result set, which was returned by DBCursorOpen or DBCursorOpenSQL.

***optStrErrorTag***

The name of a project tag that will receive detailed error messages, if errors occur during project run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

In case of success, this function returns the number of columns in the SQL result set.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |

| Value | Error Message |
|---|---|
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

See also DBCursorRowCount.

Database/ERP functions emulate Structured Query Language (SQL) database operations. Before you use these functions, you should be familiar with how SQL statements are formed and executed.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

Get the number of columns in the SQL result set that is represented by the cursor handle stored in **nCursor**:

```
DBCursorColumnCount(nCursor)
```

## *DBCursorColumnInfo*

DBCursorColumnInfo is a built-in function that gets information about a column in a SQL result set. The column is specified by number rather than by name, so this function can be used to retrieve unknown column names.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBCursorColumnInfo | Database/ERP | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
DBCursorColumnInfo( numCur, numColumn, numTypeInfo, "optStrErrorTag" )
```

**numCur**

The cursor handle of the result set. The cursor handle is returned by DBCursorOpen or DBCursorOpenSQL.

**numColumn**

The number of the column about which you want to get information. Remember that a result set may include only some of the columns in the original database table.

**numTypeInfo**

The type of information you want to get about the column:

| Value | Description |
|---|---|
| 0 | Column name |
| 1 | Column data type |

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

In the case of success, this function returns 0.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

As used in a Math worksheet:

| Tag Name | Expression |
|---|---|
| **nErrorCode** | **DBCursorColumnInfo( nCursor, 2, 0 )** // Gets the column name of the second column in the result set. |

## *DBCursorCurrentRow*

DBCursorCurrentRow is a built-in function that gets the number of the current row (i.e., the cursor position) in a SQL result set.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBCursorCurrentRow | Database/ERP | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

**DBCursorCurrentRow( *numCur*, "*optStrErrorTag*" )**

**numCur**

The cursor handle of the result set. The cursor handle is returned by DBCursorOpen or DBCursorOpenSQL.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

In the case of success, this function returns the number of the current row.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |

| Value | Error Message |
|---|---|
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

As used in a Math worksheet:

| Tag Name | Expression |
|---|---|
| **nRow** | **DBCursorCurrentRow( nCursor )** |

## *DBCursorGetValue*

DBCursorGetValue is a built-in function that gets the value in the specified column of the current row (i.e., the cursor position) in a SQL result set.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBCursorGetValue | Database/ERP | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

**DBCursorGetValue(*numCur*,*strColumn*,"*optStrErrorTag*")**

DBCursorGetValue(*numCur*, *strColumn*{ | ,"*optStrErrorTag*" })

***numCur***

The cursor handle for the result set, which was returned by DBCursorOpen or DBCursorOpenSQL.

***strColumn***

The name of the column.

***optStrErrorTag***

The name of a project tag that will receive detailed error messages, if errors occur during project run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

This function returns the value in the specified column of the current row. If the value is NULL or the cursor is invalid, this function returns an empty string with BAD quality.

### Notes

Database/ERP functions emulate Structured Query Language (SQL) database operations. Before you use these functions, you should be familiar with how SQL statements are formed and executed.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

```
DBCursorGetValue(nCursor,"Column1")
```

```
DBCursorGetValue(nCursor,"Column1","TagError")
```

## *DBCursorMoveTo*

`DBCursorMoveTo` is a built-in function that moves the cursor to the specified row in a SQL result set and then copies that row's values to the mapped tags. If the specified row doesn't exist — that is, if it's outside the range of the result set — then the function returns an error code and doesn't change the mapped tags.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBCursorMoveTo | Database/ERP | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

```
DBCursorMoveTo( numCur, numRow, "optStrErrorTag" )
```

**numCur**

The cursor handle of the result set. The cursor handle is returned by `DBCursorOpen` or `DBCursorOpenSQL`.

**numRow**

The row of the result set to which the cursor will be moved.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

In the case of success, this function returns 0.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

As used in a Math worksheet:

| Tag Name | Expression |
|---|---|
| **nErrorCode** | **DBCursorMoveTo( nCursor, 4 )** // Moves the cursor to the fourth row of the result set and copies those values. |

## *DBCursorNext*

DBCursorNext is a built-in function that moves the cursor to the next row in a SQL result set and then copies that row's values to the mapped tags. If there is no next row — that is, if the current row is the last — the function returns an error code and doesn't change the mapped tags.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBCursorNext | Database/ERP | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

**DBCursorNext(*numCur*,"*optStrErrorTag*")**

DBCursorNext(*numCur*{ | ,"*optStrErrorTag*" })

***numCur***

The cursor handle for the result set, which was returned by DBCursorOpen or DBCursorOpenSQL.

***optStrErrorTag***

The name of a project tag that will receive detailed error messages, if errors occur during project run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

In the case of success, this function returns 0.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |

| Value | Error Message |
|---|---|
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

Database/ERP functions emulate Structured Query Language (SQL) database operations. Before you use these functions, you should be familiar with how SQL statements are formed and executed.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

```
DBCursorNext(nCursor)
```

```
DBCursorNext(nCursor,"TagError")
```

## *DBCursorOpen*

`DBCursorOpen` in a built-in function that selects a set of rows and columns in a database table, initializes the cursor at the first row of the result set, copies that row's values to mapped tags, and then returns a cursor handle that can be referenced by other Database/ERP functions.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBCursorOpen | Database/ERP | Synchronous | No | Supported | Supported | Executed on Server |

## Syntax

```
DBCursorOpen(strDBConn,strTable,optStrCondition,optStrColumns, optStrTags,optStrOrder,"optStrErr
```

DBCursorOpen(*strDBConn*, *strTable*{ | ,*optStrCondition*{ | ,*optStrColumns*{ | ,*optStrTags*{ | ,*optStrOrder*{ | ,"*optStrErrorTag*" } } } } } )

***strDBConn***

The name of the database connection. Connections are configured in the **Database/ERP** folder.

***strTable***

The name of the table in the database.

***optStrCondition***

A string specifying which rows of the table to select. This is equivalent to the SQL WHERE clause, and the string should follow the same syntax.

This parameter is optional; if no rows are specified, all rows of the table will be selected.

***optStrColumns***

A string specifying which columns of the table to select. This list of column names should be comma-delimited.

This parameter is optional; if no columns are specified, all columns of the table will be selected.

***optStrTags***

A string specifying the project tags to which the columns will be mapped. This list of tag names should be comma-delimited and in the same order as the columns specified by ***optStrColumns***. As the cursor is moved through the result set, the values in the current row are copied to these tags.

This parameter is optional; if no tags are specified, no values will be copied.

***optStrOrder***

The order in which the rows will be sorted. This is equivalent to the SQL ORDER BY clause, and the string should follow the same syntax.

This parameter is optional; if no order is specified, the rows will be left in the default order of the table.

***optStrErrorTag***

The name of a String tag that will receive detailed error messages, if errors occur during run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

## Return value

In the case of success, this function returns a numeric value that can be used as a cursor handle in other Database/ERP functions.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |

| Value | Error Message |
|---|---|
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

## Notes

This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

This function is equivalent to a SQL SELECT statement, except that it breaks the clauses of the statement into separate function parameters. If you know SQL and want to compose your own SELECT statement, you can use the function DBCursorOpenSQL instead.

By default, the database interface can have a maximum of 1000 database connections — including cursor handles — open at the same time. If this limit is reached, the database interface will automatically close the oldest connection before it opens a new one. As such, you should use the function DBCursorClose to close open cursor handles as soon as you have finished with them.

You can also increase the maximum number of database connections, if necessary. To do that, use a text editor to open your project file (typically located at EcoStruxure Machine SCADA Expert 2020 Projects \<*project name*>\<*project name*>.app) and edit the following property:

```
[StDB]
```

```
MaxConnections=<from 1 to 32767>
```

Please note that having a large number of database connections open at the same time can affect run-time performance.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

Open Table1 of DB1 and select all rows where Column1 has a value greater than 3; map Column1 to Tag1 and Column2 to Tag2; order the rows first by Column1, then by Column2, in descending order; and write error messages to TagError:

```
DBCursorOpen("DB1","Table1","Column1 > 3","Column1, Column2","Tag1, Tag2","Column1,
 Column2 DESC","TagError")
```

## *DBCursorOpenSQL*

DBCursorOpenSQL is a built-in function that selects a set of rows and columns in a database table, initializes the cursor at the first row of the result set, copies that row's values to mapped tags, and then returns a cursor handle that can be referenced by other Database/ERP functions. (This function is equivalent to a SQL SELECT statement.)

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBCursorOpenSQL | Database/ERP | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

```
DBCursorOpenSQL(strDBConn,strSQL,optStrTags,"optStrErrorTag")
```

DBCursorOpenSQL(*strDBConn*, *strSQL*{ | ,*optStrTags*{ | ,"*optStrErrorTag*" }})

***strDBConn***

The name of the database connection. Connections are configured in the **Database/ERP** folder, in the *Project Explorer*. For more information, see Database/ERP worksheet on page 508.

***strSQL***

A complete, syntactically correct SQL SELECT statement. Unicode characters, including symbols and accented letters, are not supported in this statement.

> **Note:**
>
> Braces ({}) can be used as escape characters in many programming languages, to enclose some part of a text string that should be handled differently during execution. In SQL, a statement might include literal text that should not be parsed or executed as part of the statement. The following example shows a valid SQL statement, with braces used to enclose the literal text:
>
> ```
> SELECT * INTO inmates FROM OPENROWSET
>  ('MSDASQL','Driver={Microsoft Text Driver (*.txt;
>  *.csv)};DEFAULTDIR=C:\;Extensions=CSV;','SELECT * FROM
>  flat.csv')
> ```
>
> In Studio, however, braces can be used to enclose project tags and expressions that should be evaluated in text strings that are not normally evaluated (e.g., in the caption of a Button object). As such, if you pass a SQL statement that includes braces to this function, the contents of the braces will be evaluated as a tag/expression rather than as part of the SQL statement, and the function will fail.

To pass the SQL statement so that it will be handled correctly by this function, create a new project tag that contains the literal text and then reference that tag in the SQL statement. For example:

```
$AuxTag = "{Microsoft Text Driver (*.txt; *.csv)}"

$DBCursorOpenSQL("inmates","SELECT * INTO inmates FROM
 OPENROWSET ('MSDASQL','Driver={AuxTag};DEFAULTDIR=C:
\;Extensions=CSV;','SELECT * FROM flat.csv')")
```

This note applies only to the `DBCursorOpenSQL` and `DBExecute` functions. Braces cannot be used like this in any other function calls in Studio.

***optStrTags***

A string that lists the project tags to which the columns will be mapped. This list of tag names should be comma-separated and in the same order as the columns specified by the WHERE clause of *strSQL*. As the cursor is moved through the result set, the values in the current row are copied to these tags.

This parameter is optional; if no tags are specified, no values will be copied.

***optStrErrorTag***

The name of a project tag that will receive detailed error messages, if errors occur during project run time.

**Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

In the case of success, this function returns a numeric value that can be used as a cursor handle in other Database/ERP functions.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |

| Value | Error Message |
|---|---|
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

## Notes

Database/ERP functions emulate Structured Query Language (SQL) database operations. Before you use these functions, you should be familiar with how SQL statements are formed and executed.

By default, the database interface can have a maximum of 1000 database connections — including cursor handles — open at the same time. If this limit is reached, the database interface will automatically close the oldest connection before it opens a new one. As such, you should use the DBCursorClose function to close open cursor handles as soon as you are done with them.

You can also increase the maximum number of database connections, if necessary. To do that, use a text editor to open your project file (typically located at EcoStruxure Machine SCADA Expert 2020 Projects \<*project name*>\<*project name*>.app) and edit the following property:

```
[StDB]
MaxConnections=<from 1 to 32767>
```

Please note that having a large number of database connections open at the same time can affect run-time performance.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

## Examples

Open Table1 of DB1 and select all rows where Column1 has a value greater than 3; map Column1 to Tag1 and Column2 to Tag2; order the rows first by Column1, then by Column2, in descending order; and write error messages to TagError:

```
DBCursorOpenSQL("DB1","SELECT Column1, Column2 FROM Table1 WHERE Column1 > 3 ORDER BY
 Column1, Column2 DESC","Tag1, Tag2","TagError")
```

## *DBCursorPrevious*

DBCursorPrevious is a built-in function that moves the cursor to the previous row of the result set and then copies that row's values to the mapped tags. If there is no previous row — that is, if the current row is the first — then the function returns an error code and doesn't change the mapped tags.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBCursorPrevious | Database/ERP | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

```
DBCursorPrevious( numCur, "optStrErrorTag" )
```

**numCur**

The cursor handle of the result set. The cursor handle is returned by DBCursorOpen or DBCursorOpenSQL.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

In the case of success, this function returns 0.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |

| Value | Error Message |
|---|---|
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

As used in a Math worksheet:

| Tag Name | Expression |
|---|---|
| **nErrorCode** | **DBCursorPrevious( nCursor )** |

## *DBCursorRowCount*

DBCursorRowCount is a built-in function that gets the total number of rows in a SQL result set.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBCursorRowCount | Database/ERP | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

**DBCursorRowCount(*numCur*,"*optStrErrorTag*")**

DBCursorRowCount(*numCur*{ | ,"*optStrErrorTag*" })

***numCur***

The cursor handle for the result set, which was returned by DBCursorOpen or DBCursorOpenSQL.

***optStrErrorTag***

The name of a project tag that will receive detailed error messages, if errors occur during project run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

In case of success, this function returns the number of rows in the SQL result set.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

Database/ERP functions emulate Structured Query Language (SQL) database operations. Before you use these functions, you should be familiar with how SQL statements are formed and executed.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

See also `DBCursorColumnCount`.

### Examples

```
DBCursorRowCount(nCursor)
```

```
DBCursorRowCount(nCursor,"TagError")
```

## *DBDelete*

`DBDelete` is a built-in function that deletes selected rows from a database table. (This function is equivalent to a SQL DELETE statement.)

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBDelete | Database/ERP | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

```
DBDelete( strDBConn, strTable, strCondition, "optStrErrorTag" )
```

**strDBConn**

The name of the database connection. Connections are configured in the **Database/ERP** folder.

**strTable**

The name of the table in the database.

**strCondition**

A string that specifies which rows of the table to select. This is equivalent to the SQL WHERE clause, and the string should follow the same syntax.

> **Tip:** To delete all rows in the table, make the condition statement a single space (" ").

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

In the case of success, this function returns the number of rows that were deleted from the database table.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

As used in a Math worksheet:

| Tag Name | Expression |
|---|---|
| **nRowsDeleted** | **DBDelete( "DB1", "Table1", "Column1 > 1000", "TagError" )** // Deletes all rows in Table1 where the value of Column1 is greater than 1000. The returned value (i.e., the number of rows deleted) is written to **TagError**. |
| Tag | **DBDelete( "DB1", "Table1", " " )** // Deletes all rows of Table1. |

## *DBExecute*

DBExecute is a built-in scripting function that executes a custom SQL statement on an external database. If the statement is a query (e.g., SELECT), the database values are copied to specified array tags.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBExecute | Database/ERP | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

**DBExecute(*strDBConn*,*strSQL*,*opStrTags*,*optNumMaxRows*,*optStrErrorTag*)**

DBExecute(*strDBConn*, *strSQL*{ | ,*optStrTags*, *optNumMaxRows*{ | ,*optStrErrorTag*}})

***strDBConn***

The name of the database connection. Connections are configured in the **Database/ERP** folder, in the *Project Explorer*. For more information, see Database/ERP worksheet on page 508.

***strSQL***

A complete, syntactically correct SQL statement. Unicode characters, including symbols and accented letters, are not supported in this statement.

**Note:** Curly brackets (**{}**) can be used as escape characters in many programming languages, to enclose some part of a text string that should be handled differently during execution. In SQL, a statement might include literal text that should not be parsed or executed as part of the statement. The following example shows a valid SQL statement, with curly brackets used to enclose the literal text:

```
SELECT * INTO inmates FROM OPENROWSET
 ('MSDASQL','Driver={Microsoft Text Driver (*.txt;
 *.csv)};DEFAULTDIR=C:\;Extensions=CSV;','SELECT * FROM
 flat.csv')
```

In Studio, however, curly brackets can be used to enclose project tags and expressions that should be evaluated in text strings that are not normally evaluated (e.g., in the caption of a Button object). As such, if you pass a SQL statement that includes curly brackets to this function, the contents of the curly brackets will be evaluated as a tag/expression rather than as part of the SQL statement, and the function will fail.

To pass the SQL statement so that it will be handled correctly by this function, create a new project tag that contains the literal text and then reference that tag in the SQL statement. For example:

```
$AuxTag = "{Microsoft Text Driver (*.txt; *.csv)}"

$DBExecute("inmates","SELECT * INTO inmates FROM
 OPENROWSET ('MSDASQL','Driver={AuxTag};DEFAULTDIR=C:
\;Extensions=CSV;','SELECT * FROM flat.csv')")
```

> This note applies only to the `DBCursorOpenSQL` and `DBExecute` functions. Curly brackets cannot be used like this in any other function calls in Studio.

***optStrTags***

A comma-separated list of the names of array tags in your project, to which the columns of a SQL SELECT result set will be mapped. The database values will be copied to these array tags, with the first row of the result set being copied to array index 0. Make sure the arrays are large enough to receive all of the rows in the result set.

This parameter is required only when *strSQL* contains a SQL SELECT statement. For all other types of statements, this parameter is ignored and can be omitted. However, if you need to maintain the syntax of the function in order to continue through to *optStrErrorTag*, give this parameter an empty string ("").

***optNumMaxRows***

The maximum number of rows to be copied from a SQL SELECT result set. In most cases, to copy all of the rows, specify a number greater than the expected number of rows in the result set.

This parameter is required only when *strSQL* contains a SQL SELECT statement. For all other types of statements, this parameter is ignored and can be omitted. However, if you need to maintain the syntax of the function in order to continue through to *optStrErrorTag*, give this parameter a value of 0.

***optStrErrorTag***

The name of a project tag that will receive detailed error messages, if errors occur during project run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

## Return value

In the case of success, this function returns the total number of rows that were affected by the SQL statement.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |

| Value | Error Message |
|---|---|
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

Please note this is the value returned by the function itself; in the case of a SQL SELECT statement, the database values are copied to the array tags specified by *optStrTags*.

### Notes

Database/ERP functions emulate Structured Query Language (SQL) database operations. Before you use these functions, you should be familiar with how SQL statements are formed and executed.

### Examples

```
DBExecute("DB1","INSERT INTO Table1(Column1,Column2) values(1,1)")
```

```
DBExecute("DB1","SELECT max(Column1),max(Column2) FROM
 Table1","MyArray1,MyArray2",1,"TagError")
```

## *DBInsert*

`DBInsert` is a built-in function that inserts one new row into a database table. (This function is equivalent to a SQL INSERT statement.)

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBInsert | Database/ERP | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

```
DBInsert( strDBConn, strTable, strValues, optStrColumns, "optStrErrorTag" )
```

**strDBConn**

The name of the database connection. Connections are configured in the **Database/ERP** folder in the *Project Explorer*.

**strTable**

The name of the table in the database.

**strValues**

A string that lists the values to be written in the new row. This list of values should be comma-delimited, and string values must be enclosed in single quotes.

**optStrColumns**

A string that lists the columns into which the values will be written. This list of column names should be comma-delimited and in the same order as the values specified by strValues.

This parameter is optional. If no columns are specified, the values will be written in the default column order of the database table.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

## Return value

In the case of success, this function returns 1.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |

| Value | Error Message |
|---|---|
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

You can use the Database Gateway (StADOSvr) to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 815.

### Examples

As used in a Math worksheet:

| Tag Name | Expression |
|---|---|
| **nErrorCode** | **DBInsert( "DB1", "Table1", "1, 'one'", "Column1, Column2" )** |

## *DBSelect*

DBSelect is a built-in function that selects a result set from an external database (equivalent to a SQL SELECT statement), maps the columns of the result set to array tags in your project, and then copies the values from the result set to the array tags.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DBSelect | Database/ERP | Synchronous | No | Supported | Supported | Not supported |

### Syntax

**DBSelect(*strDBConn*,*strTable*,*strTags*,*strColumns*,*strCondition*,*strOrder*,*optNumMaxRows*,*optStrErrorTa**

DBSelect(*strDBConn*, *strTable*, *strTags*, *strColumns*, *strCondition*, *strOrder*{ | , *optNumMaxRows*{ | , *optStrErrorTag* } })

***strDBConn***

The name of the database connection. Connections are configured in the **Database/ERP** folder in the *Project Explorer*.

***strTable***

The name of the database table or view from which you want to select.

***strTags***

A comma-separated list of the names of array tags in your project, to which the columns of the database table or view will be mapped. The database values will be copied to these array tags, with the first row of the result set being copied to array index 0. Make sure the arrays are large enough to receive all of the rows in the result set.

***strColumns***

A comma-separated list of which columns you want to select in the database table or view. The order of this list should correlate with the order of the list that you specified for *strTags*.

To select all of the columns in the table or view, in their original order, specify an empty string ("") for this parameter.

For more information about selecting columns in a view, see "Notes" below.

***strCondition***

A statement specifying which rows in the database table or view to select. This is equivalent to the SQL WHERE clause and must follow the same syntax.

To select all of the rows in the table or view, specify an empty string ("") for this parameter.

***strOrder***

A statement specifying the order in which the rows should be sorted. This is equivalent to the SQL ORDER BY clause and must follow the same syntax.

To leave the rows in their original order, specify an empty string ("") for this parameter.

***optNumMaxRows***

The maximum number of rows to be copied. In most cases, to copy all of the rows, specify a number greater than the expected number of rows in the result set.

This parameter is optional; if no value is specified, only the first row of the result set will be copied.

***optStrErrorTag***

The name of a String tag that will receive detailed error messages, if errors occur during run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

## Return value

In the case of success, this function returns the total number of rows in the SQL result set.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |

| Value | Error Message |
|---|---|
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

Please note this is the value returned by the function itself; the database values are copied to the array tags specified by strTags.

### Notes

This feature emulates Structured Query Language (SQL) database operations. You should be familiar with how SQL statements are formed and executed before you use this feature.

For this function, you can select from either an entire database table or a specific database view. A view is a virtual table that is created by a stored query; users can query the view just as they would query a table. If you select from a view, however, you must select all of the columns in the view. In other words, when you specify the list of columns for *strColumns*, you must include all of the columns in the view, albeit in the order you choose. It is assumed that the view was created to include only the columns you need for this function.

As an alternative to this function, you can use the DBCursorOpenSQL function which allows you to compose your own SQL SELECT statement without the limitations of this function.

### Examples

```
DBSelect("DB1","Table1","Array1,Array2","Column1,Column2","","")
```

```
DBSelect("DB1","Table1","Array1,Array2","Column1,Column2","Column2 <
 Column1","Column1",4,"TagError")
```

## *DBUpdate*

`DBUpdate` is a built-in function that selects a result set and then writes the same value to all rows of a specified column. (This function is equivalent to a SQL UPDATE statement.)

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `DBUpdate` | Database/ERP | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

```
DBUpdate( strDBConn, strTable, strValues, strColumns, optStrCondition,
 "optStrErrorTag" )
```

**strDBConn**

The name of the database connection. Connections are configured in the **Database/ERP** folder in the *Project Explorer*.

**strTable**

The name of the table in the database.

**strValues**

A string that lists the values to be written to the columns. This list of values should be comma-delimited, and string values must be enclosed in single quotes.

**strColumns**

A string that lists the columns into which the values will be written. This list of column names should be comma-delimited and in the same order as the values specified by ***strValues***.

**optStrCondition**

A string that specifies which rows of the table to select. This is equivalent to the SQL WHERE clause, and the string should follow the same syntax.

This parameter is optional. If no rows are specified, all rows of the table will be selected.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during run time.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Return value

In the case of success, this function returns the number of rows that were updated.

In the case of error, this function returns one of the following possible values:

| Value | Error Message |
|---|---|
| -1 | DBERROR_DATABASE_ERROR |
| -2 | DBERROR_CONNECTION_OPEN_ERROR |
| -3 | DBERROR_CURSOREMPTY_FAILURE |
| -4 | DBERROR_CURSORMOVE_FAILURE |
| -5 | DBERROR_CURSORFETCH_FAILURE |
| -6 | DBERROR_CURSORNOTOPEN_FAILURE |

| Value | Error Message |
|---|---|
| -7 | DBERROR_CURSOR_EOF |
| -8 | DBERROR_CURSOR_BOF |
| -9 | DBERROR_INVALID_COMMAND |
| -10 | DBERROR_INVALID_CURSOR |
| -100 | DBERROR_NOT_ENOUGH_PARAMETERS |
| -101 | DBERROR_INVALID_PARAMETER_DB |
| -102 | DBERROR_INVALID_PARAMETER_TABLE |
| -103 | DBERROR_INVALID_PARAMETER_COLLIST |
| -104 | DBERROR_INVALID_PARAMETER_CONDITION |
| -105 | DBERROR_INVALID_PARAMETER_ORDER |
| -106 | DBERROR_INVALID_PARAMETER_SQL |
| -107 | DBERROR_INVALID_PARAMETER_CURSOR |
| -108 | DBERROR_INVALID_PARAMETER_VALUELIST |
| -109 | DBERROR_INVALID_PARAMETER_TAGLIST |
| -110 | DBERROR_INVALID_PARAMETER_ERRORTAG |
| -111 | DBERROR_INVALID_PARAMETER_MAXROWS |
| -112 | DBERROR_INVALID_PARAMETER_SQL_QUERY |
| -113 | DBERROR_INVALID_PARAMETER_ROW |
| -114 | DBERROR_INVALID_PARAMETER_COLUMNNUMBER |
| -115 | DBERROR_INVALID_PARAMETER_COLUMNINFO |
| -199 | DBERROR_TOO_MANY_PARAMETERS |
| -200 | DBERROR_TCP_COMM_FAILURE |
| -201 | DBERROR_EXCEEDED_OPEN_BUFFER |
| -202 | DBERROR_EXECUTING_ONLY_CLEANUP_COMMANDS |

### Notes

This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

### Examples

As used in a Math worksheet:

| Tag Name | Expression |
|---|---|
| Tag | **DBUpdate( "DB1", "Table1", "'X'", "Column2", "Column1 = 1", "TagError" )** // In Table1 of DB1, for all rows where Column1 equals 1, writes "X" to Column2. |

## *SyncAlarm*

Synchronizes the alarm database.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SyncAlarm | Database/ERP | Asynchronous | No | Supported | Executed on Server | Executed on Server |

### Syntax

```
SyncAlarm( optStrStartDate, optStrEndDate )
```

***optStrStartDate***

The start date, formatted according to the current date format on the project runtime server. For more information, see About the date format and how to change it on page 694.

This parameter is optional; if no value is specified, the current date is used by default.

***optStrEndDate***

The end date, formatted according to the current date format on the project runtime server. For more information, see About the date format and how to change it on page 694.

This parameter is optional; if no value is specified, the start date is used by default.

### Returned value

| Value | Description |
|---|---|
| 1 | Fail to start synchronization; the database is probably being synchronized. |
| 0 | Success |
| -1 | Invalid group number |
| -2 | The format is not set to "Database". |
| -4 | Start date specified is invalid. |
| -5 | End date specified is invalid. |
| -6 | Start date is greater than the end date. |

### Notes

This function is executed asynchronously, so it does not return the result of the synchronization. To get that information, use the `SyncAlarmStatus` function.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`SyncAlarm()`** // Synchronizes the database using the current date |
| Tag | **`SyncAlarm("10/20/2004")`** // Synchronizes the database only for the day 10/20/2004 |
| Tag | **`SyncAlarm("10/20/2004", "10/28/2004")`** // Synchronizes the database from 10/20/2004 to 10/28/2004 |

## *SyncAlarmStatus*

Returns the status of a previously called `SyncAlarm` function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SyncAlarmStatus` | Database/ERP | Synchronous | Yes | Supported | Executed on Server | Executed on Server |

### Syntax

```
SyncAlarmStatus()
```

This function takes no parameters.

### Returned value

| Value | Description |
|---|---|
| 3 | Synchronization has finished. |
| 2 | Fail synchronizing |
| 1 | Still synchronizing |
| 0 | No synchronization is being executed. |
| -1 | The format is not set to "Database". |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `SyncAlarmStatus()` |

## *SyncEvent*

Synchronizes the event database.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SyncEvent` | Database/ERP | Asynchronous | No | Supported | Executed on Server | Executed on Server |

### Syntax

```
SyncEvent( optStrStartDate, optStrEndDate )
```

***optStrStartDate***

The start date, formatted according to the current date format on the project runtime server. For more information, see About the date format and how to change it on page 694.

This parameter is optional; if no value is specified, the current date is used by default.

***optStrEndDate***

The end date, formatted according to the current date format on the project runtime server. For more information, see About the date format and how to change it on page 694.

This parameter is optional; if no value is specified, the start date is used by default.

### Returned value

| Value | Description |
|---|---|
| 1 | Fail to start synchronization; the database is probably being synchronized. |
| 0 | Success |
| -1 | Invalid group number |
| -2 | The format is not set to "Database". |
| -4 | Start date specified is invalid. |
| -5 | End date specified is invalid. |
| -6 | Start date is greater than the end date. |

### Notes

This function is executed asynchronously, so it doesn't return the result of the synchronization. To get that information, use the `SyncEventStatus` function.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **SyncEvent()** // Synchronizes the database using the current date |
| Tag | **SyncEvent("10/20/2004")** // Synchronizes the database only for the day 10/20/2004 |
| Tag | **SyncEvent("10/20/2004", "10/28/2004")** // Synchronizes the database from 10/20/2004 to 10/28/2004 |

## *SyncEventStatus*

Returns the status of a previously called `SyncEvent` function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SyncEventStatus | Database/ERP | Synchronous | Yes | Supported | Executed on Server | Executed on Server |

### Syntax

```
SyncEventStatus()
```

This function takes no parameters.

### Returned value

| Value | Description |
|---|---|
| 3 | Synchronization has finished. |
| 2 | Fail synchronizing |
| 1 | Still synchronizing |
| 0 | No synchronization is being executed. |
| -1 | The format is not set to "Database". |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `SyncEventStatus()` |

## *SyncTrend*

Synchronizes the trend database.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SyncTrend | Database/ERP | Asynchronous | No | Supported | Executed on Server | Executed on Server |

### Syntax

**SyncTrend( *numGroup*, *optStrStartDate*, *optStrEndDate* )**

***numGroup***

Trend group/worksheet number.

***optStrStartDate***

The start date, formatted according to the current date format on the project runtime server. For more information, see About the date format and how to change it on page 694.

This parameter is optional; if no value is specified, the current date is used by default.

***optStrEndDate***

The end date, formatted according to the current date format on the project runtime server. For more information, see About the date format and how to change it on page 694.

This parameter is optional; if no value is specified, the start date is used by default.

### Returned value

| Value | Description |
|---|---|
| 1 | Fail to start synchronization; the database is probably being synchronized. |
| 0 | Success |
| -1 | Invalid group number |
| -2 | The format is not set to "Database". |
| -4 | Start date specified is invalid. |
| -5 | End date specified is invalid. |
| -6 | Start date is greater than the end date. |

### Notes

This function is executed asynchronously, so it doesn't return the result of the synchronization. To get that information, use the `SyncTrendStatus` function.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`SyncTrend(1)`** // Synchronizes the group 1 database using the current date |
| Tag | **`SyncTrend(1, "10/20/2004")`** // Synchronizes the group 1 database only for the day 10/20/2004 |
| Tag | **`SyncTrend(1, "10/20/2004", "10/28/2004")`** // Synchronizes the group 1 database from 10/20/2004 to 10/28/2004 |

## *SyncTrendStatus*

Returns the status of a previously called `SyncTrend` function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SyncTrendStatus` | Database/ERP | Synchronous | Yes | Supported | Executed on Server | Executed on Server |

### Syntax

`SyncTrendStatus(`*numGroup*`)`

**numGroup**

Trend group/worksheet number.

### Returned value

| Value | Description |
|---|---|
| 3 | Synchronization has finished. |
| 2 | Fail synchronizing |
| 1 | Still synchronizing |
| 0 | No synchronization is being executed. |
| -1 | The format is not set to "Database". |

### Examples

| Tag Name | Expression |
| --- | --- |
| Tag | `SyncTrendStatus( 1 )` |

## Date & Time functions

These functions are used to interact with the system clock or manipulate timestamps.

### *ClockGetDate*

ClockGetDate is a built-in function that calculates the date that corresponds to a specified number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ClockGetDate | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

#### Syntax

**ClockGetDate( *numSeconds*, *optTimeZone* )**

ClockGetDate(*numSeconds*{ | ,*optTimeZone*})

***numSeconds***

A number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970. This number is typically provided by the function GetClock, but it can be any number.

***optTimeZone***

A time zone name (string) or index (integer).

| Value | Description |
|---|---|
| -*i* | A time zone index (*i*), up to the maximum index returned by the function GetTimeZoneCount. |
| 0 | The current time zone in either the project runtime server or the project thin client, depending on where the function is called.<br>This might be different from the current time zone in the local computer's system settings, if the function SetTimeZone was previously executed. |
| 1 | Coordinated Universal Time (UTC). |
| 2 | The current time zone in the local computer's system settings. |
| "*timezone*" | The full name of a time zone (e.g., "Central Standard Time"). |

This paramater is optional; if no value is specified, the default value is 0.

#### Returned value

If this function is executed successfully, it returns a string that contains the date that corresponds to the specified number of seconds. If a time zone is specified, the date is adjusted to reflect the difference between the current time zone and the specified time zone.

The date string is formatted according to the current date format. For more information, see About the date format and how to change it on page 694.

If this function is not executed successfully — for example, if the specified time zone is invalid — it returns a value of 0.

#### Notes

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

### Examples

Get the date when the system clock started counting (i.e., 01/01/1970):

```
ClockGetDate(0,"Greenwich Mean Time")
```

Get a returned value of 04/15/2002:

```
ClockGetDate(1018886359)
```

Get the current number of seconds elapsed since 12:00 AM GMT on January 1, 1970, then format that as a date, then adjust it from the current time zone to Central Standard Time:

```
ClockGetDate(GetClock(),"Central Standard Time")
```

## *ClockGetDayOfWeek*

This function calculates the day of the week that corresponds to a specified number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ClockGetDayOfWeek | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
ClockGetDayOfWeek( numSeconds, optTimeZone )
```

ClockGetDayOfWeek(*numSeconds*{ | ,*optTimeZone*})

***numSeconds***

A number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970. This number is typically provided by the function GetClock, but it can be any number.

***optTimeZone***

A time zone name (string) or index (integer).

| Value | Description |
|---|---|
| -*i* | A time zone index (*i*), up to the maximum index returned by the function GetTimeZoneCount. |
| 0 | The current time zone in either the project runtime server or the project thin client, depending on where the function is called.<br>This might be different from the current time zone in the local computer's system settings, if the function SetTimeZone was previously executed. |
| 1 | Coordinated Universal Time (UTC). |
| 2 | The current time zone in the local computer's system settings. |
| "*timezone*" | The full name of a time zone (e.g., "Central Standard Time"). |

This paramater is optional; if no value is specified, the default value is 0.

### Returned value

If this function is executed successfully, it returns the day of the week as an integer:

| Value | Description |
|---|---|
| 0 | Sunday |
| 1 | Monday |
| 2 | Tuesday |
| 3 | Wednesday |
| 4 | Thursday |
| 5 | Friday |
| 6 | Saturday |

If a time zone is specified, the day is adjusted to reflect the change from the current time zone to the specified time zone.

### Notes

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

### Examples

Get the day of the week when the system clock started counting (i.e., Wednesday):

```
ClockGetTime(0)
```

Get a returned value of `1` (Monday):

```
ClockGetDayOfWeek(1018886359)
```

Get the current number of seconds elapsed since 12:00 AM GMT on January 1, 1970, and then calculate the day of the week:

```
ClockGetDayOfWeek(GetClock())
```

## *ClockGetTime*

This function calculates the time that corresponds to a specified number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ClockGetTime | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
ClockGetTime( numSeconds, optTimeZone )
```

ClockGetTime(*numSeconds*{ | ,*optTimeZone*})

***numSeconds***

A number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970. This number is typically provided by the function GetClock, but it can be any number.

***optTimeZone***

A time zone name (string) or index (integer).

| Value | Description |
|---|---|
| -*i* | A time zone index (*i*), up to the maximum index returned by the function GetTimeZoneCount. |
| 0 | The current time zone in either the project runtime server or the project thin client, depending on where the function is called.<br>This might be different from the current time zone in the local computer's system settings, if the function SetTimeZone was previously executed. |
| 1 | Coordinated Universal Time (UTC). |
| 2 | The current time zone in the local computer's system settings. |
| "*timezone*" | The full name of a time zone (e.g., "Central Standard Time"). |

This paramater is optional; if no value is specified, the default value is 0.

### Returned value

If this function is executed successfully, it returns a string that contains the time that corresponds to the specified number of seconds. If a time zone is specified, the time is adjusted to reflect the change from the current time zone to the specified time zone. The time is formatted as HH:MM:SS.

If this function is not executed successfully — for example, if the specified time zone is invalid — it returns a value of 0.

### Notes

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

### Examples

Get the time when the system clock started counting (i.e., 00:00:00):

```
ClockGetTime(0,"Greenwich Mean Time")
```

Get a returned value of 10:59:19:

```
ClockGetTime(1018886359)
```

Get the current number of seconds elapsed since 12:00 AM GMT on January 1, 1970, then format that as a time, then adjust it from the current time zone to Central Standard Time:

```
ClockGetTime(GetClock(),"Central Standard Time")
```

## *DateTime2Clock*

This function calculates how many seconds have elapsed since 12:00 AM GMT on January 1, 1970, given a specified date and time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DateTime2Clock | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
DateTime2Clock( strDate, strTime, optTimeZone )
```

`DateTime2Clock(`*strDate*`,` *strTime*`{ | ,` *optTimeZone* `})`

***strDate***

The date to be used in the calculation, formatted according to the current date format. For more information, see About the date format and how to change it on page 694.

***strTime***

The time to be used in the calculation, formatted as HH:MM:SS.

***optTimeZone***

A time zone name (string) or index (integer).

| Value | Description |
|---|---|
| `-`*i* | A time zone index (*i*), up to the maximum index returned by the function `GetTimeZoneCount`. |
| `0` | The current time zone in either the project runtime server or the project thin client, depending on where the function is called.<br>This might be different from the current time zone in the local computer's system settings, if the function `SetTimeZone` was previously executed. |
| `1` | Coordinated Universal Time (UTC). |
| `2` | The current time zone in the local computer's system settings. |
| `"`*timezone*`"` | The full name of a time zone (e.g., `"Central Standard Time"`). |

This paramater is optional; if no value is specified, the default value is `0`.

### Returned value

This function returns the number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970, given the specified date and time. If a time zone is specified, then the number of seconds is adjusted to reflect the change from the current time zone to the specified time zone.

### Notes

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

### Examples

Get the number of seconds elapsed when the system clock started counting:

```
DateTime2Clock("01/01/1970","00:00:00","Greenwich Mean Time")
```

Get the number of seconds elapsed at 10:59:19 AM on April 15, 2002, in the current time zone:

```
DateTime2Clock("04/15/2002","10:59:19")
```

## *DateTime2UTC*

This function converts a date and time from the specified time zone to Coordinated Universal Time (UTC).

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `DateTime2UTC` | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
DateTime2UTC( strDateTime, optTimeZone )
```

DateTime2UTC(*strDateTime*{ | ,*optTimeZone* })

***strDateTime***

The date and time to be converted. The date must be formatted according to the current date format; for more information, see About the date format and how to change it on page 694. The time must be formatted as HH:MM:SS.

***optTimeZone***

A time zone name (string) or index (integer).

| Value | Description |
|---|---|
| -*i* | A time zone index (*i*), up to the maximum index returned by the function GetTimeZoneCount. |
| 0 | The current time zone in either the project runtime server or the project thin client, depending on where the function is called.<br>This might be different from the current time zone in the local computer's system settings, if the function SetTimeZone was previously executed. |
| 1 | Coordinated Universal Time (UTC). |
| 2 | The current time zone in the local computer's system settings. |
| "*timezone*" | The full name of a time zone (e.g., "Central Standard Time"). |

This paramater is optional; if no value is specified, the default value is 0.

### Returned value

If this function is executed successfully, it returns a string that contains the specified date and time converted from the specified time zone to UTC. Otheriwse, it returns one of the following possible values:

| Value | Description |
|---|---|
| -5 | Invalid time zone name or index. |
| -4 | Invalid or improperly formatted date/time. |
| -3 | The specified value is not a string. |
| -2 | Invalid number of parameters. |

### Notes

The list of available time zones varies by operating system version and configuration. If necessary, you can use the functions GetTimeZoneCount and GetTimeZone to generate a list that is specific to your computer.

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

### Examples

Convert the specified date and time from the current time zone to UTC:

```
DateTime2UTC("07/15/2013 19:54:46")
```

Convert the specified date and time from Central Standard Time to UTC:

```
DateTime2UTC("07/15/2013 19:54:46","Central Standard Time")
```

## GetClock

`GetClock` is a built-in function that gets how many seconds have elapsed since 12:00 AM GMT on January 1, 1970, according to the system clock.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetClock` | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
GetClock(optTimeZone)
```

`GetClock({ | optTimeZone})`

***optTimeZone***

A time zone name (string) or index (integer).

| Value | Description |
|---|---|
| *-i* | A time zone index (*i*), up to the maximum index returned by the function `GetTimeZoneCount`. |
| `0` | The current time zone in either the project runtime server or the project thin client, depending on where the function is called.<br>This might be different from the current time zone in the local computer's system settings, if the function `SetTimeZone` was previously executed. |
| `1` | Coordinated Universal Time (UTC). |
| `2` | The current time zone in the local computer's system settings. |
| `"`*timezone*`"` | The full name of a time zone (e.g., `"Central Standard Time"`). |

This paramater is optional; if no value is specified, the default value is `0`.

### Return value

This function returns the number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970, according to the system clock. If a time zone is specified (i.e., if *optTimeZone* is any value other than `0`), the number of seconds is adjusted to reflect the difference between the current time zone and the specified time zone.

### Notes

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

## GetTimeZone

This function gets a specified time zone name or index.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetTimeZone` | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
GetTimeZone( optTimeZone )
```

GetTimeZone({ | *optTimeZone*})

***optTimeZone***

A time zone name (string) or index (integer).

| Value | Description |
|---|---|
| *-i* | A time zone index (*i*), up to the maximum index returned by the function GetTimeZoneCount. |
| 0 | The current time zone in either the project runtime or the Viewer module / thin client, depending on where the function is called from. |
| "*timezone*" | The full name of a time zone (e.g., "Central Standard Time"). |

This parameter is optional; if no value is specified, the default value is 0.

### Return value

If *optTimeZone* is a time zone name, this function returns the corresponding index. If *optTimeZone* is a time zone index, this function returns the corresponding name. Otherwise, this function returns one of the following possible values:

| Value | Description |
|---|---|
| 2 | Invalid number of parameters. |
| 3 | Invalid time zone name or index. |

### Notes

The list of available time zones varies by operating system version and configuration. If necessary, you can use this function in combination with the function GetTimeZoneCount to generate a list that is specific to your computer.

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

### Examples

Get the current time zone:

```
GetTimeZone()
```

Get the time zone name that corresponds to time zone index 24:

```
GetTimeZone(-24)
```

Get the time zone index that corresponds to Central Standard Time:

```
GetTimeZone("Central Standard Time")
```

### *GetTimeZoneCount*

This function gets the number of available time zones on the local computer.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetTimeZoneCount | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

#### Syntax

**GetTimeZoneCount()**

GetTimeZoneCount()

This function has no parameters.

#### Returned value

This function returns the number of available time zones on the local computer, depending on where the function is called from (i.e., the project runtime or the Viewer module / thin client).

#### Notes

The list of available time zones varies by operating system version and configuration. If necessary, you can use this function in combination with the function GetTimeZone to generate a list that is specific to your computer.

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

### *GetUTC*

This function gets the current Coordinated Universal Time (UTC) on the local computer.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetUTC | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

#### Syntax

**GetUTC()**

GetUTC()

This function has no parameters.

#### Returned value

This function returns a string that contains the current Coordinated Universal Time (UTC) on the local computer, depending on where the function is called from (i.e., the project runtime server or the project thin client). The date is formatted according to the current date format; for more information, see About the date format and how to change it on page 694. The time is formatted as HH:MM:SS.

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

### *Hour2Clock*

Converts time in the **HH:MM:SS** format into seconds.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Hour2Clock | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
Hour2Clock( strTime )
```

***strTime***

The number of hours, minutes, and seconds in **HH:MM:SS** format.

### Returned value

Returns the number of seconds equivalent to the total number of hours, minutes, and seconds specified.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `Hour2Clock( "01:00:00" )` // Returned value = 3600 |
| Tag | `Hour2Clock( "10:01:01" )` // Returned value = 36061 |

## *SetSystemDate*

Sets the date in the operating system's clock.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetSystemDate | Date & Time | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
SetSystemDate(strDate)
```

SetSystemDate(*strDate*)

***strDate***

The date to which the system clock should be set, formatted according to the current date format. For more information, see About the date format and how to change it on page 694.

### Returned value

This function does not return any value.

### Notes

For this function to be executed successfully and change the system settings, you must run EcoStruxure Machine SCADA Expert 2020 with Administrator privileges. To run as an administrator once, right-click the program icon and then click **Run as administrator** on the shortcut menu. To always run as an administrator, open the program's *Properties* dialog box, click the **Compatibility** tab, and then select the **Run this program as an administrator** check box.

### Examples

Set the system clock to April 15, 2002:

```
SetSystemDate("04/15/2002")
```

## *SetSystemTime*

Sets the time in the operating system's clock.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetSystemTime | Date & Time | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
SetSystemTime(strTime)
```

SetSystemTime(*strTime*)

***strTime***

The time (in HH:MM:SS format) in which to set the clock.

### Returned value

This function does not return any value.

### Notes

For this function to be executed successfully and change the system settings, you must run EcoStruxure Machine SCADA Expert 2020 with Administrator privileges. To run as an administrator once, right-click the program icon and then click **Run as administrator** on the shortcut menu. To always run as an administrator, open the program's *Properties* dialog box, click the **Compatibility** tab, and then select the **Run this program as an administrator** check box.

### Examples

Set the system clock to 3:45:18 PM:

```
SetSystemTime("15:45:18")
```

## *SetTimeZone*

This function sets the time zone for the Viewer module / thin client, separate from the local computer's system settings.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetTimeZone | Date & Time | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
SetTimeZone( optTimeZone )
```

SetTimeZone({ | *optTimeZone*})

***optTimeZone***

A time zone name (string) or index (integer).

| Value | Description |
|---|---|
| -*i* | A time zone index (*i*), up to the maximum index returned by the function GetTimeZoneCount. |
| 0 | The current time zone in the Viewer module / thin client. |
| 1 | Coordinated Universal Time (UTC). |
| 2 | The current time zone in the local computer's system settings. |

| Value | Description |
|---|---|
| "*timezone*" | The full name of a time zone. |

This parameter is optional; if no value is specified, the default value is 2.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -3 | Invalid time zone name or index. |
| -2 | Invalid number of parameters. |
| -1 | Invalid function call; this function can only be called from the Viewer module / thin client. The project runtime will always use the local computer's system settings. |
| 0 | Function executed successfully. |

### Notes

The list of available time zones varies by operating system version and configuration. If necessary, you can use the functions GetTimeZoneCount and GetTimeZone to generate a list that is specific to your computer.

### Examples

Reset the time zone to the local computer's system settings:

```
SetTimeZone()
```

Set the time zone to UTC:

```
SetTimeZone(1)
```

Set the time zone to index 24:

```
SetTimeZone(-24)
```

Set the time zone to Central Standard Time:

```
SetTimeZone("Central Standard Time")
```

## *UTC2DateTime*

This function converts a date and time from Coordinated Universal Time (UTC) to the specified time zone.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| UTC2DateTime | Date & Time | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
UTC2DateTime( strUTC, optTimeZone )
```

UTC2DateTime(*strUTC*{ | ,*optTimeZone*})

***strUTC***

The date and time to be converted. The date must be formatted according to the current date format; for more information, see About the date format and how to change it on page 694. The time must be formatted as HH:MM:SS.

***optTimeZone***

A time zone name (string) or index (integer).

| Value | Description |
|---|---|
| `-i` | A time zone index (*i*), up to the maximum index returned by the function `GetTimeZoneCount`. |
| `0` | The current time zone in either the project runtime server or the project thin client, depending on where the function is called.<br>This might be different from the current time zone in the local computer's system settings, if the function `SetTimeZone` was previously executed. |
| `1` | Coordinated Universal Time (UTC). |
| `2` | The current time zone in the local computer's system settings. |
| `"timezone"` | The full name of a time zone (e.g., `"Central Standard Time"`). |

This paramater is optional; if no value is specified, the default value is `0`.

## Returned value

If this function is executed successfully, it returns a string that contains the specified date and time converted from UTC to the specified time zone. Otheriwse, it returns one of the following possible values:

| Value | Description |
|---|---|
| `-5` | Invalid time zone name or index. |
| `-4` | Invalid or improperly formatted date/time. |
| `-3` | The specified value is not a string. |
| `-2` | Invalid number of parameters. |

## Notes

The list of available time zones varies by operating system version and configuration. If necessary, you can use the functions `GetTimeZoneCount` and `GetTimeZone` to generate a list that is specific to your computer.

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

## Examples

Convert the specified date/time from UTC to the current time zone:

```
UTC2DateTime("07/15/2013 19:54:46")
```

Convert the specified date and time from UTC to Central Standard Time:

```
UTC2DateTime("07/15/2013 19:54:46","Central Standard Time")
```

Convert the current date and time from UTC to Central Standard Time:

```
UTC2DateTime(GetUTC(),"Central Standard Time")
```

## Email functions

These functions are used to configure and send email from within a project.

### *CnfEmail*

`CnfEmail` is a built-in function that configures the email settings used by other features in the project that can send email, such as Alarm worksheets and the `SendEmail` and `SendEmailExt` functions.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `CnfEmail` | Email | Synchronous | No | Supported | Supported | Executed on Server |

#### Syntax

**`CnfEmail(strSMTP,strFrom,strPOP3,strUser,strPassword,optNumTimeout,optNumAuthType,optStrSMTPUser,op`**

`CnfEmail(`*strSMTP*, *strFrom*, *strPOP3*, *strUser*, *strPassword*{ | , *optNumTimeout*{ | ,{ *optNumAuthType* | 1 | 2 }{ | , *optStrSMTPUser*, *optStrSMTPPassword* } } })

***strSMTP***

The hostname or IP address of the outgoing email server, which is also known as the SMTP server. You can include a port number if the server does not use one of the standard SMTP ports.

> **Note:** For projects that will run on Windows Embedded devices, you must specify an IP address.

***strFrom***

The email address from which emails will be sent and at which emails may be received. This should be a valid address on the POP3 server (see *strPOP3* below).

***strPOP3***

The hostname or IP address of the incoming email server, which is also known as the POP3 server. You can include a port number if the server does not use one of the standard POP3 ports.

> **Note:** For projects that will run on Windows Embedded devices, you must specify an IP address.

***strUser***

The username to be used to log onto the POP3 server.

***strPassword***

The password to be used to log onto the POP3 server.

***optNumTimeout***

The timeout limit (in seconds) to be used when sending email. If no response is received from the SMTP server within this period of time, then the operation is aborted.

This is an optional parameter; if no timeout is specified, then the project will keep trying forever until it receives a response. You should specify some timeout, however, to make sure that your project won't freeze.

***optNumAuthType, optStrSMTPUser, optStrSMTPPassword***

By default, SMTP servers do not require authentication for outgoing email. If your server does require authentication, set *optNumAuthType* to `1` (unencrypted) or `2` (encrypted via TLS/SSL), and then specify the username and password. (If your SMTP username and password

are the same as your POP3 username and password, then you can skip *optStrSMTPUser* and *optStrSMTPPassword*. The project will automatically use the values from *strUser* and *strPassword*.)

> **Note:** Encryption via TLS/SSL is not supported in projects running on Windows Embedded target systems.

### Return value

| Value | Description |
|---|---|
| 0 | Success |
| 1 | Invalid format for *strSMTP* |
| 2 | Invalid format for *strFrom* |
| 3 | Invalid format for *strPOP3* |
| 4 | Invalid format for *strUser* |
| 5 | Invalid format for *strPassword* |
| 6 | Invalid format for *optNumTimeout* |
| 7 | Wrong number of parameters |
| 8 | Error getting host IP address (invalid POP3 server) |
| 9 | Error connecting to POP3 server |
| 10 | Error sending username |
| 11 | Error sending password |
| 12 | SMTP server does not support selected authentication mode |
| 13 | Invalid SMTP username |
| 14 | Authentication failed |

### Notes

The email configuration created by this function works only within the Windows process where the function was called.

For example, if you place a Button object in a screen and then set the object to call this function when it is pressed, the resulting email configuration will work only on the Client station where the screen is displayed and the button is pressed. It will not work on any other Client stations nor on the Server station, because the project viewer running on the Client station only exchanges data (i.e., changes in tag values) with the data server running on the Server station. One cannot directly call functions on the other; it can only use triggers to force the other to call functions. Please note that is true even when the Client station and the Server station are the same physical device, because the project viewer and the data server are two separate processes in Windows.

If you want an email configuration to apply to your project's background tasks — for example, to be able to send emails when alarms become active — then you must either use the *E-mail Settings* dialog to configure default settings for the entire project OR call this function in some place like the project's Startup Script, a Script Group, or a Math worksheet.

### Examples

```
CnfEmail("smtp.company.com","Robert@company.com","pop.company.com","RobertH","Shades556",100)

CnfEmail("smtp.company.com:4455","Robert@company.com","pop.company.com:9900","RobertH","Shades556",

CnfEmail("195.11.22.33:4455","Robert@company.com","195.66.77.88:9900","RobertH","Shades556",5,2,"Jo
```

## *GetStatusSendEmailExt*

Returns status of the last email sent using the `SendEmailExt` function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetStatusSendEmailExt` | Email | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

`GetStatusSendEmailExt({ | `*optTagName*`})`

**optTagName**

*Optional* tag that causes the function to update its return value. This parameter is optional but you must use it when configuring this function for an object animation (e.g., Text Data Link, Position).

### Returned value

| | |
|---|---|
| -2 | Incorrect version of the **`INDMail.DLL`** library. |
| -1 | The **`INDMail.DLL`** library is corrupted. |
| 0 | `SendEmailExt` function is not being executed. |
| 1 | Still sending last email. Cannot execute the `SendEmailExt` function. |
| 2 | Last email was sent successfully. You can execute the `SendEmailExt` function again. |
| 3 | There was an error sending the last email. Execute the `SendEmailExt` function again. |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`GetStatusSendEmailExt( Second )`** |
| Tag | **`GetStatusSendEmailExt()`** |

## *SendEmail*

`SendEmail` is a built-in function that sends an email message.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SendEmail` | Email | Synchronous | No | Supported | Supported (see "Notes" below) | Executed on Server |

### Syntax

**`SendEmail(`*`strSubject`*`,`*`strMessage`*`,`*`strTo`*`)`**

`SendEmail(`*strSubject*`, `*strMessage*`, `*strTo*`)`

***strSubject***

The subject of the email.

***strMessage***

The message body of the email, up to 255 characters long.

***strTo***

The email address of the intended recipient.

### Return value

| Value | Description |
|---|---|
| 0 | Success |
| 1 | Invalid format for strSubject |
| 2 | Invalid format for strMessage |
| 3 | Invalid format for strTo |
| 4 | Wrong number of parameters |
| 5 | Start socket error |
| 6 | Error getting host IP Address (i.e., invalid SMTP server) |
| 7 | Error connecting to SMTP server |
| 8 | Error sending HELO command (i.e., initialization) |
| 9 | Error sending MAIL command (i.e., the "From" address) |
| 10 | Error sending RCPT command (i.e., the "To" address) |
| 11 | Error sending DATA (i.e., the message body) |
| 12 | Error sending SMTP authentication command |
| 13 | Invalid username |
| 14 | Invalid password |
| 15 | Data Protection is enabled (see "Notes" below). |

### Notes

Before you can send any email, you must either call the `CnfEmail` function or use the *E-mail Settings* dialog box to configure your project's email settings. Incorrect settings can result in several different error codes (see "Returned value" above).

Also, `SendEmail` cannot be used when encryption via TLS/SSL is enabled, nor to send an email that contains Unicode characters. Use the `SendEmailExt` function instead.

This function cannot be executed on thin clients (i.e., in the Viewer module) when Data Protection is enabled. It must be executed on the project runtime server. For more information, see Enable Data Protection to encrypt sensitive information on page 113.

### Examples

```
SendEmail("Hi!","How are you?","rogers@pnd.net")
```

```
SendEmail(statusSummary,statusDetail,adminAddress)
```

## *SendEmailExt*

`SendEmailExt` is a built-in function that sends email messages with attachments.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SendEmailExt` | Email | Asynchronous | No | Supported | Supported | Not supported |

### Syntax

**`SendEmailExt(strSubject,strMessage,strTo,optStrCc,optStrBcc,optStrFile1,…,optStrFileN)`**

`SendEmailExt(`*strSubject*`,` *strMessage*`,` *strTo*`{ | ,`*optStrCc*`{ | ,`*optStrBcc*`{ | ,`*optStrFile1*`,…,`*optStrFileN*`}}})`

***strSubject***

The email subject (up to 1024 characters).

***strMessage***

The email message (up to 900 characters).

***strTo***

The recipient's address. You can specify more than one recipient, using a semicolon (`;`) to separate the addresses.

***optStrCc***

The recipients' addresses to be Cc'ed. You can specify more than one recipient, using a semicolon (`;`) to separate the addresses.

This is an optional parameter, but if you need to use subsequent parameters, then you can specify a null string ("") here.

***optStrBcc***

The recipients' addresses to be Bcc'ed. You can specify more than one recipient, using a semicolon (`;`) to separate the addresses.

This is an optional parameter, but if you need to use subsequent parameters, then you can specify a null string ("") here.

***optStrFile1...optStrFileN***

Complete file paths and names of file attachments.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | Some or all of the specified file attachments were not found. |
| -3 | Wrong number of parameters (at least three parameters are required). |
| -2 | The library `INDMail.DLL` is the wrong version. |
| -1 | The library `INDMail.DLL` is corrupted. |
| 0 | Function executed successfully. |
| 1 | Cannot send email because another email is still pending. |
| 2 | Cannot send email because a new thread cannot be created. |
| 15 | Data Protection is enabled (see "Notes" below). |

### Notes

Before you can send any email, you must either call the `CnfEmail` function or use the *E-mail Settings* dialog box to configure your project's email settings. Incorrect settings can result in several different error codes (see "Returned value" above).

This function cannot be executed on thin clients (i.e., in the Viewer module) when Data Protection is enabled. It must be executed on the project runtime server. For more information, see Enable Data Protection to encrypt sensitive information on page 113.

### Examples

```
SendEmailExt("Subject","Message","Sam@universe.com","","","C:\Projects eport.txt")
```

```
SendEmailExt("Subject","Message","David@Ohio.net","Ted@Austin.com","Bart@Springfield.gov","C:
\TechRef51.doc")
```

## Event Logger functions

These functions are used to send events and comments to the Event Logger.

### *SendEvent*

`SendEvent` is a built-in function that sends a specified event to the project's event history. The event can include a comment and up to ten custom fields for other data.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SendEvent` | Event Logger | Synchronous | No | Supported | Supported | Supported |

#### Syntax

**`SendEvent(strEvent,optNumFlag,optStrComment,optCustom1,…,optCustom10)`**

`SendEvent(`*strEvent*{ | ,*optNumFlag*{ | ,*optStrComment*{ | ,*optCustom1*,...,*optCustom10*}}})

***strEvent***

The event message or description.

***optNumFlag***

A numeric flag that indicates whether to associate a comment with the event. Any non-zero value is considered TRUE.

This parameter is optional; if no value is specified, the default value is 0 (i.e., FALSE).

***optStrComment***

The text of the comment to be associated with the event.

This parameter is optional. If no value is specified, the resulting behavior varies depending on where the function is called:

- If the function is called by a background task (e.g., Math, Scheduler) on the server station, a dialog box is displayed on that station in order to get the comment text. The dialog box is displayed by the project runtime software itself, regardless of whether the local Viewer module is also running on the station.

  The function is executed synchronously, however, which means the background task is suspended while it waits for the execution to be completed, and the execution is completed only after the event — including the comment, if any — is saved in the event log. If no one is watching the station, or if the station is running "headless" (i.e., without a display or with the display turned off), the background task might be suspended indefinitely and the project's overall run-time performance might be severely affected while the function waits for the comment text.

  Therefore, we recommend that you do not use this function in any background task unless you either specify a value for this parameter or omit the comment entirely (see *optNumFlag* above).

- If the function is called by a project screen on a Thin Client, a dialog box is displayed by that Thin Client in order to get the comment text. It is expected that the current user at the client station will promptly respond to the dialog box, but even if they do not, only the project screen that called the function will be suspended while the function waits for the comment text. The project's overall run-time performance should not be affected.
- If the function is called by a project screen on Mobile Access, the event is saved with an empty comment. A dialog box is not displayed at all.

***optCustom1 ... optCustom10***

Custom data that will be included with the event. You can specify any type of data (e.g., num, str) for each field. The number of custom fields is set in the **Options** tab of the project settings.

> **Note:** This feature is not supported in projects that are configured to run on the Embedded target platform. For more information, see About target platforms, product types, and target systems on page 100.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Function executed successfully. |
| 1 | The Event Logger is disabled. See "Notes" below. |
| 2 | The Event Logger is enabled, but the **Custom Messages** option is not selected. See "Notes" below. |

### Notes

In order to use this function in your project, you must enable the Event Logger and then select the **Custom Messages** option. For more information, see Events on page 394.

### Examples

Send the event message to the event history:

```
SendEvent("Valve Open")
```

Send the event message concatenated with a tag value:

```
SendEvent("Valve Open Tank No. " + SelectedTank)
```

Prompt the user for a comment to associate with the event:

```
SendEvent("Valve Open Tank No. " + SelectedTank,1)
```

Associate a comment with the event, but use a tag value instead of a user comment:

```
SendEvent("Valve Open Tank No. " + SelectedTank,1,ValveOpenComment)
```

Prompt for a user comment, and then also include custom data:

```
SendEvent("Valve Open Tank No. " +
 SelectedTank,1,,Tank[SelectedTank].Temperature,Tank[SelectedTank].Level,Tank[SelectedTank].Pres
```

> **Note:** The examples above show how the addition operator (**+**) can be used to concatenate strings and tag values, but it can be used this way only in a Built-in Language interface (e.g., a Math worksheet). In a VBScript interface (e.g. a Script worksheet), you must use the ampersand (**&**) instead.

## File functions

These functions are used to read from, write to, print, move, and delete external files.

### *DeleteOlderFiles*

`DeleteOlderFiles` is a built-in function that deletes files that are in a specified path, that match a specified mask, and that are older than a specified date and time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DeleteOlderFiles | Files | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Not supported |

#### Syntax

```
DeleteOlderFiles(strPath,strMask,strDate)
```

`DeleteOlderFiles(`*strPath*`,` *strMask*`,` *strDate*`)`

***strPath***

The file path or location of the files to be deleted.

***strMask***

A mask or filter, using wildcard characters, that specifies the files to be deleted. For example, `"*.hst"` means all files with the `.hst` extension.

***strDate***

The cut-off date and time. Any files that are older than the specified value will be deleted. By default, Studio uses the following date/time format: **`MM/DD/YYYY HH:MM:SS`**

The date format can be localized on each station, however, and the date that is specified for this function must use the current date format on the station where the function will be executed (which is typically the project runtime server; see "Notes" below), rather than the current date format on the station where the function is called. For more information, see About the date format and how to change it on page 694.

Specifying a time is optional; you can specify a date only.

#### Returned value

This function returns the number of files that were deleted.

#### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*`.app`) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

#### Examples

Delete all history files (`*.hst`) in the project's history folder that are older than 6:00 PM on February 25, 2015:

```
DeleteOlderFiles("C:\Studio\Project\Hst\","*.hst","02/25/2015 18:00:00")
```

## *DirCreate*

Creates the specified directory.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DirCreate | File | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

```
DirCreate( strDirectory, optBooFullPath )
```

**strDirectory**

The name and file path of the directory to create.

**optBooEmptyOnly**

Optional flag. If omitted or if this parameter has the value 0, the directory is created only if all previous directories exist. If this parameter has the value different from 0, the full path specified in the strDirectory parameter is created.

### Returned value

| | |
|---|---|
| -1 | Invalid parameters |
| 0 | Failed to create the directory (e.g., Drive does not exist.) |
| 1 | Directory created successfully. |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*.app) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **DirCreate("C:\Studio\Temp")** // The Temp folder is created in the C:\Studio path (only if the C:\Studio path already exists). |
| Tag | **DirCreate("C:\Studio\Temp",1)** // The C:\Studio\Temp full path is created. |

## *DirDelete*

Deletes the specified directory.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DirDelete | File | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

```
DirDelete( strDirectory, optBooEmptyOnly )
```

**strDirectory**

The name and file path of the directory to delete.

> **Tip:** This parameter supports wildcards ( * and ? ).

**optBooEmptyOnly**

Optional flag. If this parameter has a value of 1, then the directory is deleted only if it is empty. By default — that is, if the parameter is omitted or has a value of 0 — the directory is deleted whether it is empty or not.

### Returned value

| Value | Description |
|---|---|
| -2 | Attempted to delete a non-empty directory when this action is not allowed (i.e., **optBooEmptyOnly** does not equal 0). |
| -1 | Invalid parameters. |
| 0 | Failed to delete the directory (i.e., directory does not exist). |
| 1 | Directory deleted successfully. |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*.app) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `DirDelete( "C:\Studio\Temp" )` // The Temp folder from C:\Studio is deleted. |
| Tag | `DirDelete( "C:\Studio\Temp", 1 )` // The Temp folder from C:\Studio is deleted only if it is empty. |

## *DirLength*

Returns the size of a specific directory.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DirLength | File | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Not supported |

### Description

Returns the size of a specific directory.

### Syntax

`DirLength(`*strPath*`)`

**strPath**

The path of the directory that will be checked.

### Returned value

| | |
|---|---|
| -2 | Directory does not exist. |
| -1 | Invalid parameters |
| *n* | Size (in bytes) of the files and sub-folders in the directory |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (`<project name>.app`) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

Also, this function is executed synchronously, which means that the project pauses while it waits for the function to return. As such, if the specified directory is unusually large, then the project could be paused for several seconds while size of the directory is calculated.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`DirLength("C:\Studio")`** // Returns the size (in bytes) of all files and sub-folders from C:\Studio. |

## *DirRename*

Renames directories.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `DirRename` | File | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

`DirRename(`*strPath*`,` *strDirectoryFrom*`,` *strDirectoryTo*`)`

**strPath**

The path of the directory that will be renamed.

> **Tip:** This function supports wildcard ( **`*`** and **`?`** ).

**strDirectoryFrom**

The original name of the directory that will be renamed.

**strDirectoryTo**

The target name used to rename the original directory.

### Returned value

| | |
|---|---|
| -1 | Invalid parameters |
| 0 | Failed to rename the directory (e.g., strDirectoryFrom does not exist.) |
| 1 | Directory renamed successfully. |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*.app) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **DirRename("C:\Studio\","Temp", "New")** // C:\Studio\Temp is renamed to C:\Studio\New. |

## *FileCopy*

Copies the file(s) configured in the strSourceFile parameter to the path/file configured in the strTargetFile parameter.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| FileCopy | File | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

FileCopy(*strSourceFile*, *strTargetFile*{ | , *optNumTimeOut* })

**strSourceFile**

The file path and name the file(s) to be copied.

> **Tip:** This function supports wildcards (***** and **?**).

**strTargetFile**

The file path where the file(s) are to be copied.

**optNumTimeOut**

Numerical tag containing an integer to set the timeout time for the operation.

If you use the optNumTimeOut parameter, the function returns the value -1 after the specified timeout time and the scan continues. Though the function returns a -1, it does not cancel the copying procedure. Instead, it creates an internal process to finish the copying procedure.

### Returned value

| | |
|---|---|
| -1 | Timeout time expired. |
| 0 | Failed to copy file(s). |
| 1 | File(s) copied successfully. |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*.app) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

Also, this function is executed synchronously, which means that the project pauses while it waits for the function to return. As such, if the function is called to copy files from or to another volume across a slow network, then the project could be paused for long time.

### Notes

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `FileCopy( "C:\Studio\Project\HST\*.hst", "C:\Temp\Hst\", 1000 )` |
| Tag | `FileCopy( "C:\Studio\Project opert.txt", "C:\Temp\Tuesday_Report.txt", 500 )` |

## *FileDelete*

FileDelete is a built-in function that deletes a specified file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| FileDelete | File | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

```
FileDelete(strFile)
```

FileDelete(*strFile*)

***strFile***

The file path and name of the file to be deleted.

### Returned value

The function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Failure: file not deleted. |
| 1 | Success: file deleted. |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this

function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*.app) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

### Examples

Delete a file named ReadMe.txt:

```
FileDelete("C:\Users\Me\Documents\ReadMe.txt")
```

## *FileLength*

Gets the size of a file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| FileLength | File | Synchronous | Yes | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

FileLength(*strFile*)

**strFile**

The file path and name of the file.

### Returned value

Returns the size of the specified file in bytes.

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*.app) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `FileLength( "C:\Readme.txt" )` |

## *FileReadFields*

The function FileReadFields reads values contained in the fields of a CSV file, and then it writes those values to a series of project tags or array elements.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| FileReadFields | File | Synchronous | Yes | Supported | Not supported | Not supported |

## Syntax

**FileReadFields(*strFilename*,*numOffset*,*strStartTagName*,*numNumberOfTags*)**

FileReadFields(*strFilename*, *numOffset*, *strStartTagName*, *numNumberOfTags*)

***strFilename***

The file path and/or name of the CSV file. If the file is located inside your project folder, you can specify either just the file name or the file path relative to that folder. If the file is located outside your project folder, you must specify the absolute file path.

***numOffset***

The number of bytes to skip in the CSV file before reading values. To read from the start of the file, *numOffset* should be 0.

You can use this parameter to start reading from any position in the file, as long as you know how many bytes to skip. In most cases, you will simply take the value returned by the previous execution of this function (see "Returned value" below) and use it to resume reading where you previously stopped.

However, if you already know the structure of the file and where you want to start in it, you can do that. For example, if you know that each line of the file is exactly 100 bytes and you want to read from the start of the fifth line, *numOffset* should be 400.

> **Note:** The number of bytes per character in a file depends on the text encoding (i.e., ANSI, UTF-8, UTF-16, or other), the byte order, and the language or character set.

***strStartTagName***

The name of the first project tag or array element in the series that will receive the values read from the CSV file. Project tags should be sequentially numbered with a numerical suffix, and the series is determined by incrementing that suffix (e.g., **MyTag1**, **MyTag2**, **MyTag3**, and so on). Array elements are handled similarly: the series is determined by incrementing the array index (e.g., **MyArray[1]**, **MyArray[2]**, **MyArray[3]**, and so on). You do not need to begin the numbering with 1.

> **Note:** If the tag name is not enclosed in quotes, the function will try to use the value of specified tag.

***numNumberOfTags***

The number of project tags or array elements in the series that will receive values read from the CSV file. For example, if *strStartTagName* is MyTag4 and *numNumberOfTags* is 5, five values will be read from the file and then written to the tags **MyTag4**, **MyTag5**, **MyTag6**, **MyTag7**, and **MyTag8**.

## Returned value

If this function is successfully executed, it returns the position of the last byte read from the CSV file (including 0 if no bytes were read), which can be used in turn as the offset for the next batch of fields to be read.

If this function fails, it returns a negative value.

## Notes

"CSV" is an abbreviation of "comma-separated values", and in most cases, a CSV file is simply a plain text file that uses commas (,) to delimit its data fields. Only comma delimiters — as opposed to tabs, spaces, pipes, or other characters — are supported in CSV files. Each field in the file contains exactly one value, even if the value is empty, and each project tag or array element will receive exactly one value read from the file.

When this function is executed, it will read at most one line of data. If a line feed is found before reading the specified number of values, execution of the function will be aborted without reading any values.

### Examples

```
FileReadFields("C:\FieldFiles\FieldFile01.csv",400,"IntValueTag003",5)

FileReadFields("FieldFile02.csv",0,"IntValueTag[0]",10)

FileReadFields("FieldFile03.csv",0,"IntValueTag[IndexTag]",7)
```

## *FileReadMessage*

The function `FileReadMessage` reads a message (i.e., a string of characters) from a text file, and then it writes that message to a project tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `FileReadMessage` | File | Synchronous | Yes | Supported | Not supported | Not supported |

### Syntax

```
FileReadMessage(strFilename,numOffset,strMessageTag,numCharsToRead)
```

`FileReadMessage(`*strFilename*`,` *numOffset*`,` *strMessageTag*`,{` *numCharsToRead* `|` `0` `})`

***strFilename***

The file path and/or name of the text file. If the file is located inside your project folder, you can specify either just the file name or the file path relative to that folder. If the file is located outside your project folder, you must specify the absolute file path.

***numOffset***

The number of bytes to skip in the text file before reading characters. To read from the start of the file, *numOffset* should be `0`.

You can use this parameter to start reading from any position in the file, as long as you know how many bytes to skip. In most cases, you will simply take the value returned by the previous execution of this function (see "Returned value" below) and use it to resume reading where you previously stopped.

However, if you already know the structure of the file and where you want to start in it, you can do that. For example, if you know that each line of the file is exactly 100 bytes and you want to read from the start of the fifth line, *numOffset* should be `400`.

> **Note:** The number of bytes per character in a file depends on the text encoding (i.e., ANSI, UTF-8, UTF-16, or other), the byte order, and the language or character set.

***strMessageTag***

The name of the project tag (String type) that will receive the message read from the text file.

> **Note:** If the tag name is not enclosed in quotes, the function will try to use the value of specified tag.

***numCharsToRead***

The number of characters to read from the text file, starting from the position specified by *numOffset*. If this value is `0`, all characters up to the next line feed (LF) will be read.

### Returned value

If this function is successfully executed, it returns the position of the last byte read from the text file (including 0 if no bytes were read), which be used in turn as the offset for the next message to be read.

If this function fails, it returns a negative value.

### Notes

This function can be used to read from any plain text file, as long as the correct file extension is specified in *strFilename*.

### Examples

```
FileReadMessage("C:\Data\Messages01.txt",0,"MsgTag",0)
```

```
FileReadMessage("Messages02.txt",0,"MsgTag",140)
```

## *FileRename*

`FileRename` is a built-in function that renames a specified file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `FileRename` | File | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

```
FileRename(strOldName,strNewName)
```

`FileRename` (*strOldName*, *strNewName*)

***strOldName***

The old path and name of the file.

***strNewName***

The new path and name of the file.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Failure |
| 1 | Success |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*.app) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

### Examples

```
FileRename("C:\readme.txt","C:\readthis.txt")
```

## *FileWrite*

`FileWrite` is a built-in function that writes a string to a specified text file. If the file does not exist, it will be created when the function is executed.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `FileWrite` | File | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Supported |

### Syntax

```
FileWrite(strFileName,strWriteText,optNumAppend)
```

`FileWrite(`*strFileName*`,`*strWriteText*`{ | ,`*optNumAppend*`})`

***strFileName***

The name of the text file.

By default, the file must be located in your project folder (i.e., the folder that contains the file *<project name>*`.APP`), on the computer that host the project runtime server. If the file is or should be located in another folder, specify the complete file path.

***strWriteText***

The text to be written to the file.

***optNumAppend***

A numerical flag indicating how the text should be written to the file:

| Value | Description |
|---|---|
| `0` | Create a new ASCII file with the specified file name. If the file already exists, overwrite it. |
| `1` | Append to an existing ASCII file with the specified file name. If the file does not exist, create it. |
| `2` | Create a new Unicode file (UTF-16LE) with the specified file name. If the file already exists, overwrite it. |
| `3` | Append to an existing Unicode file (UTF-16LE) with the specified file name. If the file does not exist, create it. |

This parameter is optional; if no value is specified, or if the specified value is not one of the values listed in the table above, the default value is `0`.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| `0` | Function successfully executed. |
| `-1` | Invalid parameter(s). |
| `-2` | Failed to open file. Either the drive is write-protected or the file name is invalid. |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (`<project name>.app`) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

### Examples

Create a new ASCII file in the project folder:

```
FileWrite("est.txt","This is a test.")
```

Append to an existing ASCII file in the project folder:

```
FileWrite("est.txt","This is a test.",1)
```

Append to an existing Unicode file in the Documents folder:

```
FileWrite("C:\Users\MyUser\Documents\est.txt","This is a test.",3)
```

## *FileWriteFields*

The function `FileWriteFields` reads values contained in a series of project tags or array elements, and then it writes those values to the fields of a CSV file .

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| FileWriteFields | File | Synchronous | No | Supported | Not supported | Not supported |

### Syntax

```
FileWriteFields(strFilename,numOffset,strStartTagName,numNumberOfTags)
```

`FileWriteFields(`*strFilename*`,{` *numOffset* `| -1 },` *strStartTagName*`,` *numNumberOfTags*`)`

***strFilename***

The file path and/or name of the CSV file to which the values will be written. If the file is located inside your project folder, you can specify either just the file name or the file path relative to that folder. If the file is located outside your project folder, you must specify the absolute file path.

***numOffset***

The number of bytes to skip in the CSV file before writing values. To write to the start of the file, *numOffset* should be `0`.

You can use this parameter to start writing to any position in the file, as long as you know how many bytes to skip. In most cases, you will simply take the value returned by the previous execution of this function (see "Returned value" below) and use it to resume writing where you previously stopped.

However, if you already know the structure of the file and where you want to start in it, you can do that. For example, if you know that each line of the file is exactly 100 bytes and you want to write to the start of the fifth line, *numOffset* should be `400`.

You can also specify a value of `-1`, which will automatically append the values to the end of the file.

> **Note:** The number of bytes per character in a file depends on the text encoding (i.e., ANSI, UTF-8, UTF-16, or other), the byte order, and the language or character set.

***strStartTagName***

The name of the first project tag or array element in the series from which the function will read the values to be written to the CSV file. Project tags should be sequentially numbered with a numerical suffix, and the series is determined by incrementing that suffix (e.g., **`MyTag1`**, **`MyTag2`**, **`MyTag3`**, and so on). Array elements are handled similarly: the series is determined by incrementing the array index (e.g., **`MyArray[1]`**, **`MyArray[2]`**, **`MyArray[3]`**, and so on). You do not need to begin the numbering with 1.

> **Note:** If the tag name is not enclosed in quotes, the function will try to use the value of specified tag.

***numNumberOfTags***

The number of project tags or array elements in the series to read. For example, if *strStartTagName* is `MyTag4` and *numNumberOfTags* is `5`, values will be read from the tags **`MyTag4`**, **`MyTag5`**, **`MyTag6`**, **`MyTag7`**, and **`MyTag8`** and then written to the CSV file.

### Returned value

If this function is successfully executed, it returns the position of the last byte written to the CSV file (including 0 if no bytes were written), which can be used in turn as the offset for the next batch of values to be written.

If this function fails, it returns a negative value.

### Notes

"CSV" is an abbreviation of "comma-separated values", and in most cases, a CSV file is simply a plain text file that uses commas (`,`) to delimit its data fields. Only comma delimiters — as opposed to tabs, spaces, pipes, or other characters — are supported in CSV files. Each field in the file can contain exactly one value, even if the value is empty.

When this function is executed, it will write at most one line of data. If a line feed is found before writing the specified number of tags, execution of the function will be aborted without writing any values.

### Examples

```
FileWriteFields("C:\FieldFiles\FieldFile01.csv",400,"IntValueTag003",5)

FileWriteFields("FieldFile02.csv",-1,"IntValueTag[0]",10)

FileWriteFields("FieldFile03.csv",0,"IntValueTag[IndexTag]",7)
```

## *FileWriteMessage*

The function `FileWriteMessage` reads a message (i.e., a string) from a project tag, and then it writes that message to a text file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `FileWriteMessage` | File | Synchronous | No | Supported | Not supported | Not supported |

## Syntax

**FileWriteMessage(*strFilename*,*numOffset*,*strMessage*,*numAddLineFeed*)**

FileWriteMessage(*strFilename*,{ *numOffset* | -1 },*strMessage*,{ *numAddLineFeed* | 0 | 1 })

***strFilename***

The file path and/or name of the text file. If the file is located inside your project folder, you can specify either just the file name or the file path relative to that folder. If the file is located outside your project folder, you must specify the absolute file path.

***numOffset***

The number of bytes to skip in the text file before writing the message. To write to the start of the file, *numOffset* should be 0.

You can use this parameter to start reading from any position in the file, as long as you know how many bytes to skip. In most cases, you will simply take the value returned by the previous execution of this function (see "Returned value" below) and use it to resume reading where you previously stopped.

However, if you already know the structure of the file and where you want to start in it, you can do that. For example, if you know that each line of the file is exactly 100 bytes and you want to read from the start of the fifth line, *numOffset* should be 400.

You can also specify a value of -1, which will automatically append the message to the end of the file.

> **Note:** The number of bytes per character in a file depends on the text encoding (i.e., ANSI, UTF-8, UTF-16, or other), the byte order, and the language or character set.

***strMessage***

The message to be written to the text file.

***numAddLineFeed***

A boolean value specifying whether to add a line feed (LF) to the end of the message. If this value is 1, a line feed is added. If this value is 0, a line feed is not added.

## Returned value

If this function is successfully executed, it returns the position of the last byte written to the text file (including 0 if no bytes were written), which can be used in turn as the offset for the next message to be written.

If this function fails, it returns a negative value.

## Notes

This function can be used to write to any plain text file, as long as the correct file extension is specified in *strFilename*.

## Examples

```
FileWriteMessage("C:\Data\Messages01.txt",0,MsgTag,1)
```

```
FileWriteMessage("Messages02.txt",-1,"Append this text.",0)
```

## *FindFile*

`FindFile` is a built-in scripting function that searches for all files that match a given search string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `FindFile` | File | Synchronous | Yes | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

```
FindFile(strFile,optTagFilesFound,optNumTimeout)
```

`FindFile(`*strFile*{ | `,"`*optTagFilesFound*`"`{ | `,`*optNumTimeout*}})

***strFile***

The name of the file(s) to search for.

You can use wildcards (**`*`**) to find multiple files. For example, `*.gif` to find all GIF files or `log*.txt` to find all log files in a sequence (e.g., `log001.txt`, `log002.txt`, `log003.txt`).

By default, the function only searches the project folder, but you can specify a file path (either relative or absolute) to search elsewhere. For example, if *strFile* is defined as...

```
\\<host name or IP address>\Logs\log*.txt
```

...then the function will search the `Logs` directory on the specified server.

***optTagFilesFound***

An array (of String type) that will receive the names of the matching files. The array name must be enclosed in quotes; if it is not, the function will try to get the contents of the array.

This parameter is optional. If no value is specified, the file names will not be saved and the function will only return the number of files found. For more information, see "Returned value" below.

The specified array will receive only the file names and not their paths, even if you define *strFile* to search outside the project folder. Also, the file names will be saved starting at array position 1 (e.g., **`MyArray[1]`**).

Please keep in mind that this function is executed synchronously, so there might be some delay in updating the specified array. As such, you should not develop a project screen so that it tries to use the array immediately after it calls this function. For example, you should not write a VBScript procedure that calls this function on one line and then references the array on the next. You can change this behavior, if necessary, by forcing this function to be executed asynchronously on the client. For more information, see "Notes" below.

***optNumTimeout***

The timeout period (in milliseconds) for the function to be successfully executed.

This parameter is optional. If no value is specified, the project runtime will continue searching until it has completely searched the location defined by *strFile*.

Please keep in mind that this function is executed synchronously, so if *optNumTimeout* is not specified and *strFile* is poorly defined, the entire project — both the project runtime server and its clients — might pause while it searches for the files. You can change this behavior, if necessary, by forcing this function to be executed asynchronously on the client. For more information, see "Notes" below.

> **Note:** This parameter is not supported if the target platform is Windows Embedded.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -1 | Function timed out. |
| 0 | No matching files found. |
| *n* | Number of matching files found. |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*.app) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

If you change these functions to be executed on the client, also check the scope of the array that is specified for *optTagFilesFound*. If the scope is Local, the array will be updated only on the client where this function is executed. If the scope is Server, the array will be updated first on the client where this function is executed and then that update will be sent to the server.

Also, when this function is called on the client (regardless of where it is executed), the array that is specified for *optTagFilesFound* will be updated only after the entire script (e.g., Command animation) that contains this function is executed.

As an alternative to calling this function on the client — for example, if you need to have the array updated immediately rather than after the entire script is executed — consider creating a Global Procedure that contains this function and any other associated code, and then call the RunGlobalProcedureOnServer function on the client in order to run that procedure.

### Examples

Find all text files in the project folder:

```
FindFile("*.txt")
```

Find all Microsoft Word files in the project folder, and then send the names of the matching files to **StringArray**, within a timeout period of 1000 milliseconds:

```
FindFile("*.doc","StringArray",1000)
```

## *FindPath*

FindPath is a built-in function that finds a specified directory.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| FindPath | File | Synchronous | Yes | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

```
FindPath(strPathName)
```

FindPath(*strPathName*)

***strPathName***

The path of the directory that you want to find.

If you want to find the root directory of a drive other than drive C — in other words, if you want to determine whether that drive exists — you must include an asterisk (*****) at the end of the file path. For more information, see "Examples" below.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Path not found. |
| 1 | Path found. |

This function always returns `0` if the value for *strPathName* is the path and name of a file rather than the path of a directory. To find a file, use the `FindFile` function.

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*`.app`) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

### Examples

Find the Windows directory on drive C:

```
FindPath("C:\Windows")
```

Find the root directory of drive E:

```
FindPath("E:\*")
```

## *GetFileAttributes*

Reads the attributes of a specified file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetFileAttributes` | File | Synchronous | Yes | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

`GetFileAttributes(`*strFile*`)`

**strFile**

The file path and name of the file from which to read the attributes.

### Returned value

| | |
|---|---|
| −1 | Error |

| | |
|---|---|
| 1 | Read only |
| 2 | Hidden |
| 4 | System |
| 16 | Directory |
| 32 | Archive |
| 128 | Normal |
| 256 | Temporary |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*.app) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **GetFileAttributes( "C:\Readme.txt" )** |

## *GetFileTime*

GetFileTime is a built-in function that gets the date and time that a file was last modified.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetFileTime | File | Synchronous | Yes | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

```
GetFileTime(strFileName,optNumFormat)
```

GetFileTime(*strFileName*{ | ,{ *optNumFormat* | 0 | 1 | 2 }})

***strFileName***

The file path and name of the file to be read.

***optNumFormat***

A numeric flag specifiying the format of the returned data:

| Value | Description |
|---|---|
| 0 | Date and time (i.e., MM/DD/YYYY HH:MM:SS). |
| 1 | Date only (i.e., MM/DD/YYYY). |
| 2 | Time only (i.e., HH:MM:SS). |

This parameter is optional; if no value is specified, the default value is 0.

### Returned value

This function returns the date and/or time that the file was last modified. The date will be formatted according to the current date format; for more information, see About the date format and how to change it on page 694. The time will be formatted as HH:MM:SS.

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (`<project name>.app`) and then edit the following property:

```
[Options]
ExecuteFileFunctionsOnClient=1
```

### Examples

Get the date and time that the file `Readme.txt` was last modified:

```
GetFileTime("C:\Readme.txt")
```

Get the date only that the file `History.txt` was last modified:

```
GetFileTime("C:\History.txt",1)
```

## *GetHSTInfo*

Returns the Start Time, End Time, and Duration of the specified history (*.HST) file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetHSTInfo | File | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

`GetHSTInfo(`*strFileName*`,` *numInfoType*`)`

**strFileName**

The file path and name of the history file to be read.

**numFormat**

A numeric flag specifying the type of information to be returned:

- 0: Returns the Start Time of the file.
- 1: Returns the End Time of the file.
- 2: Returns the Duration (in hours) of the file.

### Returned value

If the file cannot be read or the specified information cannot be returned, then an error is generated:

| | |
|---|---|
| **-1** | Failed to retrieve the Start Time; verify the history file exists and is valid. |
| **-2** | Failed to retrieve the End Time; verify the history file exists and is valid. |
| **-3** | Internal program error; please contact Technical Support. |
| **-4** | The Studio TCP/IP server returned a Time that is incompatible with the format specified in the project screen or Web page. Please use the **Verify Project** tool to update the project and try again. |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `GetHSTInfo( "batch", 0 )` |
| Tag | `GetHSTInfo( "hst/02060801.hst", 1 )` |
| Tag | `GetHSTInfo( "C:\batch.bat", 2 )` |

## *GetLine*

`GetLine` is a built-in function that gets a line of text from an external file and then stores that line in a project tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetLine` | File | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

**`GetLine(strFileName,Search,"strTagStore",optNumCase,"optNumTagOveflow",optNumRunFromServer)`**

`GetLine(`*strFileName*`,` *Search*`,"`*strTagStore*`"{ | ,`*optNumCase*`{ | ,"`*optNumTagOverflow*`"{ | ,`*optNumRunFromServer*`}}})`

***strFileName***

The name of the external file. It can be either a simple file name (e.g., **`file.txt`**) or a fully qualified path name (e.g., `C:\path\to\file.txt`). We recommend that you do not specify a "relative" file path (e.g., `..\file.txt`), because this function may look for the file in different locations depending on where and how it is executed. For more information, see "Notes" below.

***Search***

There are two options for this parameter, depending on the data type of the value or tag that you specify:

- If it is a string value or tag, this function will search the external file for the first occurance of the string and then copy the entire line that contains the occurance to the tag specified for *tagStore*. Additional occurances will be counted (see "Returned value" below) but not copied.
- If it is a numeric value or tag, this function will go to that line number in the external file and then copy the entire line to the tag specified for *tagStore*. The first line of the external file is line 0.

***tagStore***

The name of the String tag to which the line will be copied.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***optNumCase***

A numeric flag that specifies whether the search is case-sensitive:

| Value | Description |
|---|---|
| 0 | Not case-sensitive |
| 1 | Case-sensitive |

This parameter is optional; if no value is specified, the default value is 0.

***optNumTagOverflow***

The name of an Integer tag that will receive a code that indicates whether the line overflows the String tag specified for *tagStore*.

This parameter is optional; if no value is specified, the overflow code will not be received.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

The overflow code is separate from the value returned by the function itself (see "Returned value" below). The possible values of the overflow code are:

| Value | Description |
|---|---|
| 0 | OK |
| 1 | Overflow |

The size limit for String tags in projects running on Windows is 1024 characters, which means that if the tag specified for *optNumTagOverflow* receives an overflow code of 1, the line is more than 1024 characters long and it overflowed the tag specified for *tagStore*.

***optNumRunFromServer***

A numeric flag that specifies whether this function should be run from the project thin client or the project runtime server:

| Value | Description |
|---|---|
| 0 | Run from the project thin client |
| 1 | Run from the project runtime server |

This parameter is optional; if no value is specified, the default value is 0.

For more information about how this parameter affects the execution of this function, see "Notes" below.

## Returned value

If this function is executed successfully, it returns the total number of lines in which the search string was found. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | String was not found in the target file |
| -1 | File not found |
| -2 | Invalid *strFileName* parameter |
| -3 | Invalid *strSeqChar* parameter (obsolete) |
| -4 | Invalid *tagStore* parameter |
| -5 | Invalid *optNumCase* parameter |
| -6 | Invalid *optNumTagOverflow* parameter |
| -7 | Invalid number of parameters |
| -8 | Invalid line number |

## Notes

This function supports only ASCII and UTF-16LE text encoding on Windows and Windows Embedded. (UTF-16LE is the Unicode implementation that is natively supported by Windows.) Therefore, if you use this function to get text from a UTF-8 or UTF-16BE encoded file, the function should be executed successfully but you might see some unknown or invalid characters.

Where this function will look for the file depends on where the function was called and on the values specified for *strFileName* and *optNumRunFromServer*:

| If function is called on... | ...and *strFileName* is... | ...and *optNumRunFromServer* is... | ...it looks for the file... |
|---|---|---|---|
| Project runtime server, in a background task | File name only | 0 | On the server, in the project folder and its Web sub-folder. (*optNumRunFromServer* is ignored.) |
| | | 1 | |
| | File path and name | 0 | On the server, in the specified file path. (*optNumRunFromServer* is ignored.) |
| | | 1 | |
| Secure Viewer (incl. local Viewer) | File name only | 0 | Not valid; file cannot be found. |
| | | 1 | On the server, in the project folder and its `\Web` sub-folder. |
| | File path and name | 0 | On the client, in the specified file path. |
| | | 1 | On the server, in the specified file path. |
| Mobile Access | File name only | 0 | On the server, in the location specified in the browser's address bar. Typically, that is the project's `\Web` sub-folder, which contains the project screens published as web pages, but it might be another location if the web pages were copied or moved after they were published. The latter is often true for Mobile Access. Check the web server to see what folder(s) it is configured to serve. |
| | | 1 | On the server, in the project folder and its `\Web` sub-folder. |
| | File path and name | 0 | On the client, in the specified file path. |
| | | 1 | On the server, in the specified file path. |

### Examples

Find the first occurrence of "EcoStruxure Machine SCADA Expert 2020" in a file located on the client, and then store the line that contains the occurrence in the tag **`ReturnedLine`**:

```
GetLine("C:\Readme.txt","EcoStruxure Machine SCADA Expert 2020","ReturnedLine")
```

Find the third line of a file located on the server, and then store that line in the tag **`ReturnedLine`**:

```
GetLine("C:\Readme.txt",2,"ReturnedLine",0,"Overflow",1)
```

## *HST2TXT*

`HST2TXT` is a built-in function that exports historical data from the trend history files (`*.hst`) to a plain text file (`*.txt`).

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `HST2TXT` | File | Asynchronous | No | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

```
HST2TXT(strStartDate,strStartTime,numDuration,numGroup,strTargetFile,optStrSeparator,optNumMilli
```

HST2TXT(*strStartDate*, *strStartTime*, *numDuration*, *numGroup*, *strTargetFile*{ | , *optStrSeparator*{ | , *optNumMilliseconds*{ | , *optStrFormat*{ | , *optNumInterval*}}}})

***strStartDate***

The start date (e.g., 04/14/2002) of the data to be exported. The date must be in the format used by the project runtime server. For more information, see "Notes" below.

***strStartTime***

The start time (e.g., 06:30:00) of the data to be exported.

***numDuration***

The duration (in hours) of the data to be exported, starting from the specified start date and time.

***numGroup***

The trend group or worksheet number. For more information, see Trend worksheet on page 413.

***strTargetFile***

The name of the text file to which the data will be exported.

By default, the file must be located in your project folder (i.e., the folder that contains the file *<project name>*.app), on the computer that hosts the project runtime server. If the file is or should be located in another folder, specify the complete file path.

***optStrSeparator***

The character that will be used to separate the values in each line of the file.

This parameter is optional; if no value is specified, the default value is a TAB character (**\t**).

***optNumMilliseconds***

A numeric flag that indicates whether to show millisecond-precision in the timestamp on each entry:

| Value | Description |
|---|---|
| 0 | Do not show milliseconds. |
| 1 | Show milliseconds. |

This parameter is optional; if no value is specified, the default value is 0.

***optStrFormat***

The date format that will be used in the timestamp on each entry:

| Value | Description |
|---|---|
| DMY | Day, Month, Year |
| MDY | Month, Day, Year |
| YMD | Year, Month, Day |

This parameter is optional; if no value is specified, the default value is DMY.

***optNumInterval***

The interval between entries to be exported. Only entries at this interval are exported to the specified file. For example, if *optNumInterval* is 10, only every tenth entry is exported.

This parameter is optional; if no value is specified, every entry is exported.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | Invalid date format (see "Notes" below). |
| -3 | Invalid number of parameters. |
| -2 | DLL functions not found. |
| -1 | InStudiot.dll not found in the program folder. |
| 0 | Function executed successfully. |
| 1 | Error. Previous execution of the function HST2TXT has not yet been completed. |

### Notes

If a Trend group/worksheet is configured to save historical data to proprietary history files instead of an external database, the files will be saved in your project folder at *<project name>*\Hst*.hst, on the computer that hosts your project runtime server. These history files are in a proprietary binary format, which is why it is necessary to export the historical data to a separate text file if you want to reuse your trend history in other applications.

Since the history files are located on the server, the export must be done on the server, which means this function is executed on the server even when it is called in a project screen on a thin client. You cannot export to a file on the thin client.

Furthermore, since this function is executed on the server, the date that you specify for *strStartDate* must be in the date format used by the server. If you specify a date in a different date format — even the format used by the client — the server will not recognize it and this function will return a value of -4 (i.e., invalid date format). For more information, see About the date format and how to change it on page 694.

Although this function can be called while the project is either running or stopped, it can be executed only after the project has ran at least once. If you try to call this function before the project has ran — for example, if you start the EcoStruxure Machine SCADA Expert 2020 application and then immediately enter this function in the Watch window on page 71 — it will fail without error.

If you want to export historical data without running the project, you can also use the command-line utility HST2TXT.exe. It is especially useful if you want to do the export as part of a longer, scripted procedure. For more information, see Converting Trend History Files from Binary to Text on page 421.

> **Tip:** To export to a comma-separated values (CSV) file that can be imported into Microsoft Excel and other spreadsheet applications, specify a comma separator and a file name with the CSV suffix (e.g., history.csv).

### Examples

Export 0.1 hour (i.e., six minutes) of historical data from Trend group/worksheet 3 to a slash-separated text file:

```
HST2TXT("04/14/2002","06:30:00",0.1,3,"data.txt","\")
```

Export 0.1 hour (i.e., six minutes) of historical data from Trend group/worksheet 3 to a CSV file using the MDY date format:

```
HST2TXT("04/14/2002","06:30:00",0.1,3,"data.csv",",",0,"MDY")
```

Export every tenth entry in 0.1 hour (i.e., six minutes) of historical data from Trend group/worksheet 3 to a CSV file using the MDY date format:

```
HST2TXT("04/14/2002","06:30:00",0.1,3,"data.csv",",",0,"MDY",10)
```

## *HST2TXTIsRunning*

Returns the status of the `HST2TXT` function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `HST2TXTIsRunning` | File | Synchronous | Yes | Supported | Executed on Server | Not supported |

### Syntax

**`HST2TXTIsRunning()`**

`HST2TXTIsRunning()`

This function takes no parameters.

### Returned value

| Value | Description |
|---|---|
| `0` | HST2TXT is still running. |
| `-1` | Last conversion process was executed properly. |
| `-2` | Reserved. |
| `-3` | File not found. There are no history files in the configured time interval for the group specified. |
| `-4` | Cannot open history file. |
| `-5` | Cannot create/open ASCII file. |
| `-6` | Cannot read file information from history file. |
| `-7` | Invalid file type. |
| `-8` | Cannot read header information from history file. |
| `-9` | Invalid number of tag in the header information (**`0 > nTags > 250`**). |
| `-10` | Cannot create Header file (**`.hdr`**). |
| `-20` | `InStudiot.dll` was not found. |
| `-30` | Cannot access DLL function. |

## *ImportXML*

The function `ImportXML` is used to import Studio XML files into your project during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `ImportXML` | File | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**`ImportXML(`*`strXMLFile`*`,`*`optStrDestFile`*`,`*`optNumFileType`*`,`*`optNumReplaceDuplicate`*`)`**

`ImportXML(`*strXMLFile*{ | ,*optStrDestFile*{ | ,*optNumFileType*{ | ,*optNumReplaceDuplicate*}}})

**strXMLFile**

The file path and name of the XML file to be imported. If you specify only a file name with no path, the runtime will automatically look in the Screen and Web sub-folders of your project folder.

**optStrDestFile**

The new name of the project file, if any.

This parameter is optional; if no value is specified, the default value is `"screen"`.

**optNumFileType**

A numeric flag indicating the type of Studio XML file to be imported. At this time, only Studio XML Screen files are supported.

| Value | Description |
|---|---|
| 0 | Studio XML Screen file |

This parameter is optional; if no value is specified, the default value is `0`.

**optNumReplaceDuplicate**

A numeric flag indicating whether the imported XML file should replace an existing project file of the same name. For example, `Objects.xml` and `Objects.scc` would be duplicates.

| Value | Description |
|---|---|
| 0 | Do not replace duplicate project file. |
| 1 | Replace duplicate project file. |

This parameter is optional; if no value is specified, the default value is `0`.

## Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -2 | Invalid license; this function is not available in Demo Mode. |
| -1 | Cannot execute this function on the project client (i.e., Viewer). Execute the function on the project server. |
| 0 | Function executed successfully. |
| 1 | Invalid number of parameters. |
| 2 | Invalid parameter data type. |
| 3 | Cannot replace duplicate file. |
| 4 | Failed to load import module. (ImportXML does not work when your project is running as a Windows service. For more information, see Run a project as a Windows service on page 138.) |
| 5 | File type (optNumFileType) not supported. |
| 6 | Failed to create destination file. |
| 7 | Internal error. |
| 8 | Failed to save imported screen. |
| 9 | Failed to retrieve screen file. |
| 10 | Internal XML file error. |

## Notes

This function can be executed only on the project server, where the EcoStruxure Machine SCADA Expert 2020 develoment environment must be installed and running with at least a Runtime license. You can use the functions `StartTask`, `EndTask`, and `IsTaskRunning` with the parameter `"Studio"` to programmatically control the development environment.

For more information about Studio XML Screen files and how they are created, see Import a Studio XML Screen on page 222.

### Examples

Look for `Screen1.xml` in the Screen and Web sub-folders, and then import it to create `Screen1.scc` if it does not already exist:

```
ImportXML("Screen1.xml")
```

Look for `Screen1.xml` at the specified file path, and then import it to create `Screen1.scc` if it does not already exist:

```
ImportXML("C:\Users\<user name>\Documents\EcoStruxure Machine SCADA Expert 2020
 Projects\SMA_Project\Screen\Screen1.xml")
```

Look for `Screen1.xml` at the specified file path, and then import it to create `Screen9.scc` if it does not already exist:

```
ImportXML("C:\Users\<user name>\Documents\EcoStruxure Machine SCADA Expert 2020
 Projects\SMA_Project\Screen\Screen1.xml","Screen9.scc")
```

Look for `Screen1.xml` at the specified file path, and then import it to replace `Screen9.scc`:

```
ImportXML("C:\Users\<user name>\Documents\EcoStruxure Machine SCADA Expert 2020
 Projects\SMA_Project\Screen\Screen1.xml","Screen9.scc",0,1)
```

## *LookupContains*

This function verifies that an external file contains the specified keyword in its key column.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| LookupContains | File | Synchronous | Yes | Supported | Not supported | Not supported |

### Syntax

`LookupContains(`*strKey*`)`

**strKey**

The keyword to look for in the file's keywords column.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Specified keyword not found. |
| 1 | Specified keyword found. |

### Notes

The external file must already be loaded by calling the function `LookupLoad`.

### Examples

```
LookupContains( "customer167" )
```

## *LookupGet*

This function gets a value from an external file by cross-referencing from a specified keyword.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| LookupGet | File | Synchronous | Yes | Supported | Not supported | Not supported |

### Syntax

LookupGet(*strKey*)

**strKey**

The keyword to look for in the file's keywords column.

### Returned value

This function returns (as a string) the cross-referenced value from the file's specified values column.

If no value is found, then this function returns strKey.

### Notes

The external file must already be loaded by calling the function LookupLoad.

### Examples

```
LookupGet( "customer167" )
```

## *LookupLoad*

This function loads an external file — typically, a delimited text file — that can be used to look up table values. One column of the file is designated as the keywords column, and another column is designated as the values column.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| LookupLoad | File | Synchronous | Yes | Supported | Not supported | Not supported |

### Syntax

LookupLoad(*strFileName*, *numColKey*, *numColValue*, *strDelimiters*)

**strFileName**

The file path and name of the external file.

**numColKey**

The number of the column/field that contains the keywords.

**numColValue**

The number of the column/field that contains the desired values.

**strDelimiters**

The delimiter that separates the columns/fields.

### Returned value

This function returns the number of rows/lines in the specified file.

If the specified file cannot be found, then this function returns a negative number as an error code.

### Notes

This function only loads the specified file; it doesn't do anything with the file. To use the file, call the LookupContains and LookupGet functions.

Also, to load another file, simply call this function again. Only one file can be loaded at a time, however; the new file replaces the old in the project's memory.

### Examples

```
LookupLoad( "C:\Temp\customerlist.csv", 1, 4, "," )
```

## *PDFCreate*

Creates a PDF file from the specified source file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| PDFCreate | File | Synchronous | No | Supported | Supported | Not supported |

### Syntax

PDFCreate(*strSourceFile*{ | ,*optStrPdfFile*})

**strSourceFile**

String specifying the file path and name of the desired source file ( ***.doc**, ***.txt**, or ***.rtf** ). If a file path is not specified, the function will look for the source file in the project folder.

**optStrPdfFile**

*Optional* string specifying the file path and name of the created PDF file. If a file path is not specified, then the PDF file will be saved in the same location as the source file. If this parameter is omitted — that is, if no file path or name is specified at all — then the PDF file will be saved in the same location and with the same name as the source file. Only a new extension is added. For example, C:*path**to*\MyDocument.rtf becomes C:*path**to*\MyDocument.pdf.

### Returned value

| Value | Description |
|---|---|
| 0 | Success |
| 1 | Error in PDF profile information |
| 3 | Error saving PDF file |
| 4 | Job canceled |
| 101 | Error initializing PDF resource |
| 102 | Specified source file not found |
| 103 | Error generating PDF file |
| 104 | Wrong number of parameters |
| 105 | Wrong parameter type |

**Note:** This function only supports the execution of one job at a time. If more than one user or command attempts to call the function at the same time, then the function will fail and return a value of **101**.

### Examples

| Tag Name | Expression |
|---|---|
| | **PDFCreate( "C:\Report1.rtf" )** |
| | **PDFCreate( "C:\Report2.doc", "C:\Converted1.pdf" )** |

| Tag Name | Expression |
|---|---|
| | `PDFCreate( "C:\Report3.txt", "C:\Data\Converted1.pdf" )` |

## *Print*

`Print` is a built-in function that prints the contents of a specified text file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `Print` | File | Asynchronous | No | Supported | Supported | Not supported |

### Syntax

```
Print(strFilePath,optNumOrientation)
```

`Print(`*strFilePath*{ | ,*optNumOrientation* })

***strFilePath***

The file path and name of the text file to be printed.

***optNumOrientation***

A numeric value that indicates the paper oriententation:

| Value | Description |
|---|---|
| 0 | Portrait |
| 1 | Landscape |

This parameter is optional; if no value is specified, the default value is 0.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Invalid parameter(s). |
| 1 | Valid parameters. |

This function only checks whether the specified parameters are valid, before it tries to use those parameters to print the file. In other words, the return value does not indicate whether the file is successfully printed. To determine that, view the print queue in Windows.

### Notes

This function can only be used to print the contents of text files. It cannot be used to print information in any other format (e.g., pictures, binary files, etc.).

This function is based on legacy code, which means it cannot use printer settings that were previously configured by the `PrintSetup` function. Instead, it always uses the default printer on the computer or device that hosts the project runtime. You can use VBScript in your project to change the default printer in Windows, however. For example:

```
Dim WSHNetwork
Set WSHNetwork = CreateObject("WScript.Network")
WSHNetwork.SetDefaultPrinter "<printer name>"
Set WSHNetwork = Nothing
```

### Examples

Print the contents of `ReadMe.txt` in portrait mode:

```
Print("C:\ReadMe.txt")
```

Print the contents of the file specified by the **MyTextFile** tag, in landscape mode:

```
Print(MyTextFile,1)
```

## *RDFileN*

Launches a *File Browser* window allowing you to select a file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| RDFileN | File | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Not supported |

### Syntax

```
RDFileN( "tagSelectedFile", strSearchPath, strMask, optNumChangeDir )
```

**tagSelectedFile**

Name of the string tag receiving the name and path of a selected file. The tag name **must** be enclosed in quotes, or the project will try to get the contents of the tag. Moreover, it must be a valid tag name — it cannot be a VBScript variable name, for example.

**strSearchPath**

The file path (directory) to search.

**strMask**

The mask used to filter the files.

**optNumChangeDir**

*Optional* numeric tag that indicates whether the operator will be able to change the browsing directory. If this parameter is omitted or set **TRUE** ( **1** ), then the window opened by this function will allow the operator to navigate to different directories. If it is set **FALSE** ( **0** ), then the window will be restricted to the directory specified by **strSearchPath**.

### Returned value

| | |
|---|---|
| 0 | Success |
| 1 | One of the parameters is not a string |
| 2 | Parameter 1 contains an invalid tag name |
| 3 | The user canceled the operation |

### Notes

By default, this function is executed on the project runtime server even when it is called in a screen on a project thin client, which means it only affects directories and files that are located on the server and all paths must be specified in that context. You can change the default behavior to force this function to be executed on the client where it is called, but if you do so, you must keep in mind that the change will affect not just this function but also several other File functions. To change the behavior of these functions, use a text editor to open your project file (*<project name>*.app) and then edit the following property:

```
[Options]
```

```
ExecuteFileFunctionsOnClient=1
```

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `RDFileN( "FileName", "C:\Studio\", "*.doc", 1 )` |

## *WebGetFile*

`WebGetFile` is a built-in function that gets a file from a specified URL and then saves it locally.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `WebGetFile` | File | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
WebGetFile(strURL,strLocalPath)
```

`WebGetFile(`*strURL*`,` *strLocalPath*`)`

***strURL***

The URL (i.e., the web address) of the file you want to get. You can specify a port number as part of the URL.

***strLocalPath***

The complete local file path where you want to save the downloaded file.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| xxx | HTTP status code; see below. |
| 0 | Success. |
| -1 | Invalid number of parameters. |
| -2 | Invalid URL. |
| -3 | Invalid port number. |
| -4 | Error while opening a connection to the specified server. Make sure the server name (i.e., the domain or host name) and port number are correct. |
| -5 | Error while saving the file. Make sure the local path is correct and you have the necessary privileges to save files there. |

In some cases, this function might successfully connect to the web server but still fail to get a file from the specified URL. When that happens, the web server should provide a three-digit HTTP status code which indicates the reason for the failure (e.g., "404 File Not Found"), and then this function will relay that code as its return value.

If this function continues to return errors, use a web browser on the same computer that hosts the project runtime to confirm it can access the network and go to the specified URL.

### Examples

Get the file named `myfile.txt`:

```
WebGetFile("http://www.the-internet.com/myfile.txt","C:\myfile.txt")
```

Get the file specified by the project tag **myURL**:

```
WebGetFile(myURL,myFilePath)
```

# FTP functions

Use the FTP functions to configure the FTP settings for your project, as well as to get files from and put files on a remote server.

## *CnfFTP*

`CnfFTP` is a built-in function that configures the FTP settings used by other features in the project that can transfer files, such the `FTPGet` and `FTPPut` functions.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `CnfFTP` | FTP | Synchronous | No | Supported | Supported | Not supported |

### Syntax

**`CnfFTP(strServer,optStrUser,optStrPassword,optNumPassiveMode,optNumPort)`**

`CnfFTP(`*strServer*`{ | ,`*optStrUser*`,`*optStrPassword*`{ | ,`*optNumPassiveMode*`{ | ,`*optNumPort*`}}})`

***strServer***

The host name or IP address of the FTP server.

***optStrUser***
***optStrPassword***

The username and password that will be used to log on to the FTP server.

These parameters are optional; if no values are specified, the project will log on anonymously by default (i.e., *optStrUser* is `"anonymous"` and *optStrPassword* is `""`).

***optNumPassiveMode***

A numeric flag that specifies whether passive mode is enabled. (Passive FTP can be used to bypass some firewall configurations.) This parameter can have the following possible values:

| Value | Description |
|---|---|
| `0` | Passive mode is disabled (default). |
| `1` | Passive mode is enabled. |

This parameter is optional; if no value is specified, passive mode is disabled by default.

***optNumPort***

The port number of the FTP server.

This parameter is optional; if no value is specified, port 21 is used by default.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| `-3` | Invalid user name. |
| `-2` | Invalid server name. |
| `-1` | Invalid number of parameters. |
| `0` | Success. |

> **Note:** This function does not actually connect to the specified server, so these error codes do not show the quality of the connection. They only show whether the FTP settings have been successfully configured.

### Notes

You must either call this function at least once or use the *FTP Settings* dialog box to configure your project's FTP settings before you can call the `FTPGet` and `FTPPut` functions to transfer files.

### Examples

Configure the FTP settings using passive mode and the default port 21:

```
CnfFTP("ftp.mycompany.com","admin","12345",1)
```

## *FTPGet*

`FTPGet` is a built-in function that gets a file from an FTP server and then saves it on the local computer.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `FTPGet` | FTP | Asynchronous | No | Supported | Supported (see "Notes" below) | Not supported |

### Syntax

```
FTPGet(strRemoteFile,strLocalFile,optNumTransferType,optNumOverwrite)
```

`FTPGet(`*strRemoteFile*`,` *strLocalFile*`{ | ,`*optNumTransferType*`{ | ,`*optNumOverwrite*`}})`

***strRemoteFile***

The full path and name of the file that you want to get from the FTP server, using the syntax "***/folder/filename.extension***". Some FTP servers are case sensitive, so you should always use correct capitalization.

***strLocalFile***

The full path and name where you want to save the file on the local machine, using the syntax "**C:*folder**filename.extension***".

***optNumTransferType***

A numerical flag that specifies the type of file transfer:

| Value | Description |
|---|---|
| 0 | Unknown (default). |
| 1 | ASCII. |
| 2 | Binary. |
| 10 | Unknown, without caching. |
| 11 | ASCII, without caching. |
| 12 | Binary, without caching. |

This parameter is optional; if no value is specified, the transfer type is unknown (0) by default.

***optNumOverwrite***

A numeric flag that specifies whether the local file (specified by ***strLocalFile***) may be overwritten if it already exists:

| Value | Description |
|---|---|
| 0 | Do not overwrite (default) — return an error if the file already exists. |
| 1 | Overwrite. |

This parameter is optional; if no value is specified, the default value is 0.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -5 | Invalid transfer type. |
| -4 | Invalid local file. |
| -3 | Invalid remote file. |
| -2 | Unknown system error. |
| -1 | Invalid number of parameters. |
| 0 | Success. |
| 1 | Failed to create FTP thread or open connection to the server. |
| 15 | Data Protection is enabled (see "Notes" below). |

### Notes

Before you can call this function, you must either call the `CnfFTP` function or use the *FTP Settings* dialog box to configure your project's FTP settings.

Also, this function is executed asynchronously, so you must call the `FTPStatus` function to confirm that the transfer is completed.

This function cannot be executed on thin clients (i.e., in the Viewer module) when Data Protection is enabled. It must be executed on the project runtime server. For more information, see Enable Data Protection to encrypt sensitive information on page 113.

### Examples

Get the file `040303.txt` from the previously specified FTP server, and then save it at `C:\Report.txt`:

```
FTPGet("/Reports/040303.txt","C:\Report.txt")
```

## *FTPPut*

`FTPPut` is a built-in function that puts a file from the local computer on an FTP server.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| FTPPut | FTP | Asynchronous | No | Supported | Supported (see "Notes" below) | Not supported |

### Syntax

```
FTPPut(strLocalFile,strRemoteFile,optNumTransferType )
```

`FTPPut(`*strLocalFile*`,` *strRemoteFile*`{ | ,`*optNumTransferType*`})`

***strLocalFile***

The full path and name of the file on the local machine that you want to put on the FTP server, using the syntax **"C:*folder**filename.extension*"**.

***strRemoteFile***

The full path and name where you want to put the file on the FTP server, using the syntax "***/folder/filename.extension***". Some FTP servers are case sensitive, so you should always use correct capitalization.

***optNumTransferType***

A numeric flag that specifies the type of file transfer:

| Value | Description |
|---|---|
| 0 | Unknown (default). |
| 1 | ASCII. |
| 2 | Binary. |

This parameter is optional; if no value is specified, the transfer type is unknown (0) by default.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -5 | Invalid transfer type. |
| -4 | Invalid local file. |
| -3 | Invalid remote file. |
| -2 | Unknown system error. |
| -1 | Invalid number of parameters. |
| 0 | Success. |
| 1 | Failed to create FTP thread or open connection to the server. |
| 15 | Data Protection is enabled (see "Notes" below). |

### Notes

Before you can call this function, you must either call the `CnfFTP` function or use the *FTP Settings* dialog box to configure your project's FTP settings.

Also, this function is executed asynchronously, so you must call the `FTPStatus` function to confirm that the transfer is completed.

This function cannot be executed on thin clients (i.e., in the Viewer module) when Data Protection is enabled. It must be executed on the project runtime server. For more information, see Enable Data Protection to encrypt sensitive information on page 113.

### Examples

Put the file `Report.txt` on the previously specified FTP server at `/Reports/040303.txt`:

```
FTPPut("C:\Report.txt","/Reports/040303.txt")
```

## *FTPStatus*

`FTPStatus` is a built-in function that gets the current status of any file transfers that were started by the functions `FTPGet` and `FTPPut`.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| FTPStatus | FTP | Synchronous | Yes | Supported | Supported (see "Notes" below) | Not supported |

## Syntax

```
FTPStatus("optStrTagStatus")
```

FTPStatus("*optStrTagStatus*")

***optStrTagStatus***

The name of the string tag that will receive a text description of the current status of the file transfer(s). The description corresponds to the actual status code returned by the function (see "Returned value" below). However, this parameter is optional; if no value is specified, the description will not be received.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

## Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -9 | Transfer pending |
| -8 | Error receiving the file (see status string for details) |
| -7 | Error establishing connection (see status string for details) |
| -6 | Error opening connection (see status string for details) |
| -2 | Invalid *strStatusTag* |
| 0 | No transaction is being executed |
| 1 | Transaction executed successfully |
| 2 | Resolving name |
| 3 | Name resolved |
| 4 | Connecting to server |
| 5 | Connected to server |
| 6 | Closing connection |
| 7 | Connection closed |
| 8 | Sending request |
| 9 | Request sent |
| 10 | Receiving response |
| 11 | Intermediate response received |
| 12 | Response received |
| 13 | Request completed |
| 15 | Data Protection is enabled (see "Notes" below). |

## Notes

Unlike the other FTP functions, you can call this function at any time, but it will not provide useful information unless you have previously called FTPGet or FTPPut.

This function cannot be executed on thin clients (i.e., in the Viewer module) when Data Protection is enabled. It must be executed on the project runtime server. For more information, see Enable Data Protection to encrypt sensitive information on page 113.

### Examples

Get the current status of a file transfer, and then store a text description of the status in the string tag named "StatusDescription":

```
FTPStatus("StatusDescription")
```

## Graphic functions

These functions are used to manipulate and print project screens.

### *AutoFormat*

Automatically formats a real number to a preset number of decimal places, according to the virtual table of settings created by the `SetDecimalPoints` function. (This is similar to the `Format` function, except that you do not need to specify the number of decimal places.)

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `AutoFormat` | Graphic | Synchronous | Yes | Supported | Supported | Not supported |

#### Syntax

`AutoFormat(`*numValue*`)`

**numValue**

The real number to be formatted.

#### Returned value

This function returns a formatted string.

#### Examples

In the following examples, the `SetDecimalPoints` function has already been used to set 3 decimal places for values greater than equal to 1.5 and 1 decimal place for values less than or equal to −3.

| Tag Name | Expression |
|---|---|
| Tag | **`AutoFormat( 1.543210 )`** // Returned value = "1.543" |
| Tag | **`AutoFormat( -3.123456 )`** // Returned value = "−3.1" |

### *GetScrInfo*

The function `GetScrInfo` gets information about an open project screen.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetScrInfo` | Graphic | Synchronous | Yes | Supported | Supported | Not supported |

#### Syntax

**`GetScrInfo(`*`strScreenName`*`,"`*`tagResult`*`",`*`optNumResultType`*`,`*`optNumID`*`)`**

`GetScrInfo(`*strScreenName*`,"`*tagResult*`"{ | ,{ `*optNumResultType* ` | 0 | 1 | 2 | 3 | 4 }{ | ,`*optNumID* ` } })`

***strScreenName***

The name of the screen about which you want to get information.

***tagResult***

The name of the project tag that will receive the information.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***optNumResultType***

A numeric flag specifying the type of information to be retrieved by the function:

| Value | Description |
|---|---|
| 0 | Default value. Writes the TOP, LEFT, BOTTOM and RIGHT screen coordinates to each consecutive position of the array tag specified for *tagResult*. |
| 1 | Writes the TOP screen coordinate to the tag specified for *tagResult*. |
| 2 | Writes the LEFT screen coordinate to the tag specified for *tagResult*. |
| 3 | Writes the BOTTOM screen coordinate to the tag specified for *tagResult*. |
| 4 | Writes the RIGHT screen coordinate to the tag specified for *tagResult*. |

This is an optional parameter; the default value is 0.

***optNumID***

The specific instance number of the screen. (The ID is assigned when the screen is opened with the function `Open`.) This is an optional parameter; the default ID is 0.

## Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | Invalid tag specified for *tagResult*. |
| -3 | *optNumResultType* is 0, but an array tag is not specified for *tagResult*. |
| -2 | Memory allocation error. |
| -1 | String not specified for *strScreenName* and/or *tagResult*. |
| 0 | Function executed successfully. |

## Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

## Examples

Retrieve the TOP, LEFT, BOTTOM and RIGHT coordinates of the "main" screen and then write them to the first four positions of the array tag TagXY:

```
GetScrInfo("main","TagXY[0]")
```

Retrieve the BOTTOM coordinate of the "main" screen and then write it to TagXY:

```
GetScrInfo("main","TagXY",3)
```

Retrieve the LEFT coordinate of the "main" screen with ID 10 and then write it to TagXY:

```
GetScrInfo("main","TagXY",2,10)
```

## *PrintSetup*

The function `PrintSetup` displays a standard print setup dialog box on the client, in which the user can select and configure a printer for printing project screens.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `PrintSetup` | Graphic | Asynchronous | No | Supported | Supported | Not supported |

### Syntax

**`PrintSetup()`**

`PrintSetup()`

This function takes no parameters.

### Return value

This function returns no value.

### Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

## *PrintWindow*

The function `PrintWindow` prints a screenshot of a project screen. The screen does not need to be open and active; the function can print a screen running in the background or even closed screen file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `PrintWindow` | Graphic | Asynchronous | No | Supported | Supported | Not supported |

### Syntax

**`PrintWindow(`*`strScreenName`*`,`*`optNumOrientation`*`,`*`optNumID`*`,`*`optStrMnemonicList`*`)`**

`PrintWindow(`*strScreenName*`{ | ,{ `*optNumOrientation*` | 0 | 1 }{ | ,`*optNumID*`{ | ,`*optStrMnemonicList*` } } })`

***strScreenName***

The name of the screen to be printed. If this parameter is omitted, then the currently active screen will be printed. (This parameter must be omitted when executing the function on a Windows Embedded target system.)

***optNumOrientation***

A numeric flag specifying the print orientation:

| Value | Description |
|---|---|
| 0 | Portrait |
| 1 | Landscape |

This parameter is optional; if no value is specified, the default value is 0.

> **Note:** This parameter is not supported in projects running on Windows Embedded target systems.

***optNumID***

The specific ID number of the screen. (This number is assigned when the screen is opened using the function `Open`.)

This parameter is optional; if no value is specified, the default value is 0.

***optStrMnemonicList***

A string that describes how the custom properties of any generic objects or linked symbols in the screen will be completed when the screen is printed. This string has the following syntax...

```
#Label:Value
```

...where ***Label*** is the name of the property and ***Value*** is the tag, expression or literal value that the property will receive. You can declare two or more mnemonics, as long as they are separated by spaces. See the Examples section below for an example.

> **Note:** This parameter does not work for a screen that is already open, because if the screen has been opened, the custom properties have received their values.

## Returned value

This function does not support printing to Studio PDF2. (Studio PDF2 is a PDF printer that is installed with EcoStruxure Machine SCADA Expert 2020 for use with other features such as the `PDFCreate` function.) If Studio PDF2 is selected as the printer on the computer or device where this function is executed, the specified screen is not printed and this function returns `-1`.

Otherwise, this function returns no value.

## Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

## Examples

Print the currently active screen in portrait orientation:

```
PrintWindow()
```

Print the screen named "Main" in landscape orientation:

```
PrintWindow("Main",1)
```

Print the screen specified by the tag **MyScreenName**:

```
PrintWindow(MyScreenName)
```

Print the screen named "Main" with ID 10:

```
PrintWindow("Main",1,10)
```

Print the screen named "Main", replacing the custom properties **Mne1** and **Mne2** with the values of **Tag1** and **Tag2**, respectively:

```
PrintWindow("Main",1,0,"#Mne1:Tag1 #Mne2:Tag2")
```

> **Tip:** You can use this function to print graphical reports that include Alarm/Event Control and Trend Control objects.

## *ResetDecimalPointsTable*

Resets the virtual table of settings created by the `SetDecimalPoints` function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `ResetDecimalPointsTable` | Graphic | Synchronous | No | Supported | Supported | Not supported |

```
ResetDecimalPointsTable()
```

### Syntax

`ResetDecimalPointsTable()`

This function takes no parameters.

### Return value

This function returns no value.

### Examples

| Tag Name | Expression |
|---|---|
| | **ResetDecimalPointsTable( )** // Resets the virtual table of settings. |

## *RGBColor*

Returns the number of the color defined by the RGB (Red, Green, Blue) codes.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `RGBColor` | Graphic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

`RGBColor(`*numRed*, *numGreen*, *numBlue*`)`

**numRed**

Red code from the RGB code.

**numGreen**

Green code from the RGB code.

**numBlue**

Blue code from the RGB code.

### Returned value

This function returns the number of the color defined by the RGB (Red, Green, Blue) codes.

### Examples

| Tag Name | Expression |
|---|---|
| TagColor | **RGBColor(51,153,102)** // This function returns the value 13434828, which is the color code for Sea Green. |
| TagColor | **RGBColor(TagRed,TagGreen,TagBlue)** // This function returns the color code of the RGB values set in the tags TagRed, TagGreen and TagBlue, respectively. |

> **Tip:** See the list of RGB Codes and Color values for the most used colors in the Color Interface section.

## *RGBComponent*

`RGBComponent` is a built-in scripting function that gets the level of a color component (red, green, or blue) in a specified color.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `RGBComponent` | Graphic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

`RGBComponent(`*numColor*`,{` *numComponent* `| 0 | 1 | 2})`

**numColor**

The decimal code for a 24-bit RGB color, which can be any integer value between 0 and 16777215. (This color model is also known as "Truecolor" or "millions of colors.")

**numComponent**

The color component for which you want to get the level: `0` is red, `1` is green, and `2` is blue.

### Returned value

This function returns an integer value between 0 or 255, which represents the level of the color component in the specified color.

### Notes

For a list of frequently used RGB color codes and their equivalent "plain English" names, see Color Interface.

### Examples

Get the level of red in color code 13434828 (i.e., sea green):

```
RGBComponent( 13434828, 0 )
```

Get the level of the component specified by **TagComponent** in the color specified by **TagCode**:

```
RGBComponent( TagCode, TagComponent )
```

## *SaveScreenShot*

`SaveScreenShot` is a built-in function that takes a screen shot of a project screen and then saves it as an image file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SaveScreenShot` | Graphic | Synchronous | No | Supported | Supported (see "Notes" below) | Not supported |

## Syntax

**SaveScreenShot(*optStrScreenName*,*optStrOutputFile*,*optNumFormat*)**

SaveScreenShot({ | *optStrScreenName*{ | ,*optStrOutputFile*{ | ,{ *optNumFormat* | 0 | 1 | 2 | 3 | 4 | 5 }}}})

***optStrScreenName***

The file path and name of a project screen file (*.scc or *.scr). If no file path is specified, the file must be located in the Screen sub-folder of the project folder. For example: EcoStruxure Machine SCADA Expert 2020 Projects\<*project name*>\Screen\<*screen name*>.scr

Whether the screen must be open depends on where the function is executed. See "Notes" below.

This parameter is optional; if no value (or "") is specified, the currently open and active screen is used.

***optStrOutputFile***

The file path and name of the output file.

This parameter may be either optional or required, depending on where the function is executed. See "Notes" below.

***optNumFormat***

The format of the image file:

| Value | Description |
|---|---|
| 0 | BMP |
| 1 | JPG |
| 2 | PNG |
| 3 | GIF |
| 4 | TIFF |
| 5 | Auto |

This parameter is optional; if no value is specified, the default is 1 (JPG).

## Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Success. |
| -1 | Wrong number of parameters. |
| -2 | Wrong parameter types. |
| -3 | Invalid file path for *optStrOutputFile*. |
| -4 | *optStrOutputFile* cannot be empty. |
| -5 | Wrong format / invalid option for *optNumFormat*. |
| -6 | Failed to save output file. |
| -7 | Failed to create compatible bitmap. |
| -10000 | Project is not running. This error typically occurs when you try to call the function in the *Watch* window. See "Notes" below. |

### Notes

This function behaves somewhat differently depending on where it is executed:

**On the server**

In order for the function to be executed on the project runtime server, the Viewer task (i.e., the server's local Viewer program) must be started. The function can be called either by background tasks (e.g., Script and Math worksheets) or in the *Watch* window, but it will fail if the Viewer task is not also started. This is because the Viewer task is used to render the screen. For more information about starting run-time tasks, see Runtime Tasks on page 136.

*optStrOutputFile* is optional; if no value (or `""`) is specified, the file is saved in the Web sub-folder of the project folder and the file name is either the value of *optStrScreenName* (if specified) or simply `ScreenShot.jpg`. For example: `<project name>\Web\ScreenShot.jpg`

**Secure Viewer**

If the function is executed by Secure Viewer running on a remote station, the screen may be either open or closed. Also, *optStrOutputFile* is required; you must specify a complete file path and name for the output file.

### Examples

Take a screen shot of the current screen, and then save it as `ScreenShot.jpg`:

```
SaveScreenShot()
```

Take a screen shot of `main.scc`, and then save it as `main.jpg`:

```
SaveScreenShot("main.scc")
```

Take a screen shot of the current screen, and then save it as a bitmap with the name of the currently logged user:

```
SaveScreenShot("",UserName,0)
```

## *SetDecimalPoints*

Sets the number of decimal places to be displayed, for a specified range of real numbers. This setting will be used by all screen objects and animations that have the **Auto Format** option enabled, as well as by the `AutoFormat` function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SetDecimalPoints` | Graphic | Synchronous | No | Supported | Supported | Not supported |

### Syntax

`SetDecimalPoints(`*numBaseValue*`,` *numDecimalPoints*`)`

**numBaseValue**

The base value of the range of real numbers. For negative values, the range includes all real numbers less than or equal to that value. For positive values, the range includes all real numbers greater than or equal to that number. (You can set the other limit of the range by calling the function again with a new set of parameters.)

**numDecimalPoints**

The number of decimal places to be displayed, for the range of real numbers specified by **`numBaseValue`**.

### Returned value

| | |
|---|---|
| 0 | Error |

| | |
|---|---|
| 1 | Success |

### Notes

If you call this function more than once with different parameters for each call, then you can build a virtual table of format settings. You can set a different number of decimal places for each range of real numbers, and all of the settings are saved for the duration of runtime or until you reset the table using the `ResetDecimalPointsTable` function.

> **Note:** This formatting does *not* change the actual value of any tag or expression. It only changes how the value is displayed by on-screen objects.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `SetDecimalPoints( 1.5, 3 )` // Displays 3 decimal places for all real numbers greater than or equal to 1.5. |
| Tag | `SetDecimalPoints( -3, 1 )` // Displays 1 decimal place for all real numbers less than or equal to −3. |

## *SetDisplayUnit*

Finds all tags and all Grid object and Trend Control object values that have a specific engineering unit (as stored in the **Unit** tag field), and then sets the **DisplayUnit**, **UnitDiv**, and **UnitAdd** fields on those tags.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SetDisplayUnit` | Graphic | Synchronous | No | Supported | Supported | Not supported |

### Syntax

`SetDisplayUnit(`*strUnitOrigin*, *strDisplayUnit*, *numDiv*, *numAdd*`)`

**strUnitOrigin**

The engineering unit to be matched.

**strDisplayUnit**

The new value for the **DisplayUnit** tag field.

**numDiv**

The new value for the **UnitDiv** tag field.

**numAdd**

The new value for the **UnitAdd** tag field.

### Returned value

| | |
|---|---|
| 0 | Success. |
| -1 | Wrong number of parameters. |
| -2 | strUnitOrigin parameter is empty. |
| -3 | numDiv parameter is invalid (equal to 0). |

### Notes

This function only affects how the tag values are displayed on screen; it does not change the actual tag values in any way.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`SetDisplayUnit( "C", "F", 0.555556, 32 )`** // For all tags and object values with a **Unit** of "C", the **DisplayUnit** tag field is set to "F", the **UnitDiv** tag field is set to 0.555556, and the **UnitAdd** tag field is set to 32. |

## *SetTagDisplayUnit*

Sets the **DisplayUnit**, **UnitDiv**, and **UnitAdd** properties on a specific tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SetTagDisplayUnit` | Graphic | Synchronous | No | Supported | Supported | Not supported |

### Syntax

`SetTagDisplayUnit(`*strTagName*`,` *strDisplayUnit*`,` *numDiv*`,` *numAdd*`)`

**strTagName**

The name of the specific tag on which the **DisplayUnit**, **UnitDiv** and **UnitAdd** tag fields will be set.

> **Note:** If this parameter is given a tag, then that tag should *contain* the name of the tag on which the tag fields will be set.

**strDisplayUnit**

The new value for the **DisplayUnit** tag field.

**numDiv**

The new value for the **UnitDiv** tag field.

**numAdd**

The new value for the **UnitAdd** tag field.

### Returned value

| | |
|---|---|
| 0 | Success. |
| -1 | Wrong number of parameters. |
| -2 | Specified tag doesn't exist. |
| -3 | numDiv parameter is invalid (equal to 0). |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`SetTagDisplayUnit( "TagTemp", "F", 0.555556, 32 )`** // For the tag "TagTemp", the **DisplayUnit** tag field is set to "F", the **UnitDiv** tag field is set to 0.555556, and the **UnitAdd** tag field is set to 32. |

## Log Message functions

These functions are used to display status and debug messages in the *Output* window (for local runtime) or *Remote LogWin* window (for remote runtime).

### *Trace*

`Trace` is a built-in function that displays a text message in the *Output* window. It is typically used to debug the project.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access | |
|---|---|---|---|---|---|---|---|
| `Trace` | Log Message | Synchronous | No | Supported | Supported | Not supported | Supported (see "Notes" below) |

#### Syntax

```
Trace(strOutputMessage)
```

`Trace` (*strOutputMessage*)

***strOutputMessage***

The text of the message to be displayed.

#### Return value

This function returns no value.

#### Notes

On Mobile Access, trace messages are displayed in the activity log in the browser console. For more information, see Use the activity log to troubleshoot the Mobile Access web interface on page 804.

#### Examples

Display static text that reports a specific event:

```
Trace("Beginning step 5.")
```

Display a date or time stamp by referencing the appropriate system tag:

```
Trace(Date)
```

Concatenate static text, tag references, and function calls to form a complex message:

```
Trace("The current second of the minute is " + Second + " and the system tick is " +
 GetTickCount() + " ms.")
```

## Logarithmic functions

These functions are used to perform logarithmic operations on numeric values.

### *Exp*

Calculates the value of **e** (2.718282) raised to the power of **numValue**.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Exp | Logarithmic | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

Exp (*numValue*)

**numValue**

Integer or Real tag containing the exponent of **e**.

#### Returned value

Returns the value of **e^( numValue )**.

#### Examples

| Tag Name | Expression |
|---|---|
| Tag | **Exp( 1 )** // Returned value = 2.718282 |
| Tag | **Exp( 5.25896 )** // Returned value = 192.281415 |

### *Log*

Calculates the natural log of **numValue**.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Log | Logarithmic | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

Log (*numValue*)

**numValue**

Integer or Real tag from which the natural log is taken.

#### Returned value

Returns the value of **ln( numValue )**.

> **Note:** If **numValue** has a negative value, then this function will return the value 0 and it will set the quality of the returned tag to **BAD**.

#### Examples

| Tag Name | Expression |
|---|---|
| Tag | **Log( 2.718282 )** // Returned value = 1 |
| Tag | **Log( 100 )** // Returned value = 4.605170 |

## *Log10*

Calculates the log base 10 of **numValue**.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Log10 | Logarithmic | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

Log10(*numValue*)

**numValue**

Integer or Real tag, from which the log base 10 is taken.

### Returned value

Returns the value of **log10( numValue )**.

> **Note:** If **numValue** has a negative value, then this function will return the value 0 and it will set the quality of the returned tag to **BAD**.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **Log10( 1000 )** // Returned value = 3 |
| Tag | **Log10( 43.05 )** // Returned value = 1.633973 |

## Logical functions

These functions are used to perform logical operations (e.g., if/then, true/false) on tags and expressions.

### *False*

Determines whether the specified tag or expression is logically false.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `False` | Logical | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

`False(`*TagOrExpression*`)`

**TagOrExpression**

Tag or expression to be used in the function.

#### Return value

| | |
|---|---|
| 0 | If the tag or expression is *not* logically false. |
| 1 | If the tag or expression is logically false. |

> **Tip:** You may find this function useful if you need to return an actual value of 0 when the expression returns some value other than 0.

#### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`False( 1 )`** // Returned value = 0 |
| Tag | **`False( 5 < 2 )`** // Returned value = 1 |

### *If*

`If` is a built-in function that evaluates a specified tag/expression to determine whether it is logically true or false, and then it returns a corresponding value.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `If` | Logical | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

**`If(`*`numExpression,numThen,optNumElse`*`)`**

`If(`*numExpression*`,`*numThen*`{ | ,`*optNumElse*`})`

***numExpression***

The tag or expression to be evaluated.

***numThen***

The tag, expression, or value that is returned if *numExpression* is logically true (i.e., not 0).

***optNumElse***

The tag, expression, or value that is returned if *numExpression* is logically false (i.e., 0).

This parameter is optional; see "Returned value" below.

### Returned value

This function returns either *numThen* or *optNumElse*, depending on how *numExpression* is evaluated.

If *numExpression* is logically false and *optNumElse* is not specified, this function returns no value. Furthermore, if a project tag is configured to receive the value returned by this function but the function returns no value, the project tag retains its existing value.

### Notes

*numExpression* can be a combination of logic statements (e.g., **AND**, **OR**, **NOT**). For example:

```
If(TagA>TagB AND TagA=10,1,0)
```

Both *numThen* and *optNumElse* can be another function, including the `If` function itself. Therefore, you can use the functions in cascade. For example:

```
If(TagA>TagB,If(TagA<TagC,1,2),3)
```

This is not recommended when using the `If` function in VBScript, however, because of how the function has been implemented. In VBScript, each parameter is fully evaluated when the function is executed, so if one of the parameters is another function, that function will also be executed regardless of whether it should be. Therefore, we recommend that you use VBScript's own **If**…**Then**…**Else** statement instead, wherever possible.

### Examples

Evaluate the expression "5 > 4", and then return the corresponding value (i.e., return 10, because the expression is logically true):

```
If(5>4,10,6)
```

Evaluate the expression "5 < 2", and then return the corresponding value (i.e., return 2, because the expression is logically false):

```
If(5<2,0,2)
```

Evaluate the expression "3 = 9", and then return the corresponding value (i.e., return no value, because the expression is logically false and *optNumElse* is not specified):

```
If(3=9,67)
```

## *Toggle*

Returns the toggled value from the contents of **numValue** tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `Toggle` | Logical | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Toggle( numValue )
```

**numValue**

Boolean tag containing the value to be toggled.

### Returned value

Numerical result (0 or 1) of the value to be toggled.

### Notes

This function does not actually change the value of the tag, but it can be used in a command or operation that does.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`Toggle( MyBoolTag )`** // Returned value = 1 if MyBoolTag value equals 0, or 0 if MyBoolTag value equals 1 |
| Tag | **`Toggle( numValue )`** // Returned value = toggled value of the number in the numValue tag |

## *True*

Determines whether the specified tag or expression is logically true.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `True` | Logical | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

`True(`*TagOrExpression*`)`

**TagOrExpression**

Tag or expression to be used in the function.

### Return value

| | |
|---|---|
| 0 | If the tag or expression is not logically true. |
| 1 | If the tag or expression is logically true. |

> **Tip:** You may find this function useful if you need to return an actual value of 1 when the expression returns some value other than 0.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`True( 1 )`** // Returned value = 1 |
| Tag | **`True( 5 < 2 )`** // Returned value = 0 |

## Loop functions

Loop functions are used to implement an incrementing loop within a script.

### *For...Next*

`For` and `Next` are built-in functions that implement an incrementing loop in a Math worksheet. The section of script included in the loop begins with the **`For()`** call and ends with the **`Next`** notation. The **`Next`** notation directs back to the beginning of the loop.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `For` | Loop | N/A | No | Supported | Supported | Not supported |

#### Syntax

**`For(`*`numInitialValue`*`,`*`numFinalValue`*`,`*`numStep`*`)`**

`For(`*numInitialValue*`,` *numFinalValue*`,` *numStep*`)`

***numInitialValue***

The initial or starting value of the loop.

***numFinalValue***

The final or ending value of the loop.

***numStep***

The step or increment value.

#### Returned value

This function returns the current value of the loop.

#### Notes

This type of **`For`**...**`Next`** loop can be used only in Math worksheets. It cannot be used in any Command animation or VBScript interface. When you are using VBScript, you should use that language's own tools for looping. For more information, see Looping Through Code on page 1243.

You must partner every **`For()`** call with a **`Next`** notation, although you may have any number of worksheet rows between them. And as shown in the example below, you must place the **`Next`** notation in the **Tag Name** column of the worksheet.

The loop ends when its current value equals the value of *numFinalValue*. Specifically, each time the worksheet is executed and the **`Next`** notation is encountered, the following happens:

1. The value of *numStep* is added to the current value of the loop;
2. The execution returns to the **`For()`** call at the beginning of the loop; and
3. The current value of the loop is compared to the value of *numFinalValue*. If the current value is less than *numFinalValue*, the loop is executed again. If the current value is greater than or equal to *numFinalValue*, the loop is skipped and execution resumes with the first row after the **`Next`** notation.

#### Examples

##### Example of **`For`**...**`Next`** loop in a Math worksheet

| Tag Name | Expression |
|---|---|
| `MyTag` | `For(1,5,1)` |
| *`tagname`* | *`expression`* |
| *`tagname`* | *`expression`* |

| Tag Name | Expression |
| --- | --- |
| *tagname* | *expression* |
| Next | |

## Module Activity functions

These functions are used to manage a project's various runtime modules — such as background tasks, the data server, and the project viewer — as well as those modules' interactions with the operating system.

### *AppActivate*

`AppActivate` is a built-in scripting function that activates (i.e., brings to the front) another application window that is already open.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `AppActivate` | Module Activity | Asynchronous | No | Supported | Supported | Not supported |

#### Syntax

`AppActivate(`*strAppTitle*`{ | ,{` *optNumActiv* `| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}{ | ,` *optNumTimeout* `}})`

**strAppTitle**

The full title (as shown in the title bar) of the application window.

**optNumActiv**

Controls how the specified window is to be activated:

| Value | Command | Description |
|---|---|---|
| 0 | **SW_HIDE** | Hides the currently active window and then activates the specified window. |
| 1 | **SW_SHOWNORMAL** | Activates and displays the specified window. If the window is minimized or maximized, then it is restored to its original size and position.<br>You should use this command when displaying a window for the first time. |
| 2 | **SW_SHOWMINIMIZED** | Activates the specified window and then minimizes it. |
| 3 | **SW_SHOWMAXIMIZED** | Activates the specified window and then maximizes it. |
| 4 | **SW_SHOWNOACTIVATE** | Displays the specified window, but does not activate it. If the window is minimized or maximized, then it is restored to its original size and position. |
| 5 | **SW_SHOW** | Activates and displays the specified window in its current size and position. This is similar to **SW_SHOWNORMAL** except that if the window is minimized or maximized, then it remains in that state. |
| 6 | **SW_MINIMIZE** | Minimizes the specified window and then activates the next open window. |
| 7 | **SW_SHOWMINNOACTIVATE** | Displays the specified window as a minimized window, but does not activate it. |
| 8 | **SW_SHOWNA** | Displays the specified window in its current size and position, but does not activate it. This is similar to **SW_SHOWNOACTIVATE** except that if the window is minimized or maximized, then it remains in that state. |

| Value | Command | Description |
|---|---|---|
| 9 | **SW_RESTORE** | Activates and displays the specified window. If the window is minimized or maximized, then it is restored to its original size and position.<br>You should use this command when restoring a minimized window. |

This is an optional parameter. If no value is specified, then the default command is **SW_RESTORE**.

**optNumTimeout**

The timeout period (in milliseconds) for the function to be successfully executed. If, for whatever reason, the function is not executed in this period, then it is aborted.

This is an optional parameter. If no value is specified, then the default timeout is five seconds (or 5000 milliseconds).

### Returned value

This function will return one of the following values:

| Value | Description |
|---|---|
| 0 | ERROR: The specified application window was not activated or otherwise did not respond within the timeout period. |
| 1 | SUCCESS: The specified application window was successfully activated. |

### Notes

AppActivate is similar to the function ShowWindow in the Microsoft Windows API, and it allows many of the same options. For more information, please refer to the Windows API documentation.

### Examples

Show the Microsoft Word document named test.doc:

```
AppActivate( "test.doc – Microsoft Word", 5 )
```

## *AppIsRunning*

AppIsRunning is a built-in scripting function that verifies another application window is open and running.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| AppIsRunning | Module Activity | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

AppIsRunning(*strAppTitle*{ | ,*optNumTimeout*})

**strAppTitle**

The full title (as shown in the title bar) of the application window.

**optNumTimeout**

The timeout period (in milliseconds) for the function to be successfully executed. If, for whatever reason, the function is not executed in this period, then it is aborted.

This is an optional parameter. If no value is specified, then the default timeout is five seconds (or 5000 milliseconds).

### Returned value

This function will return one of the following values:

| Value | Description |
|---|---|
| 0 | ERROR: The specified application window is not open or otherwise did not respond within the timeout period. |
| 1 | SUCCESS: The specified application window is open and running. |

### Notes

AppIsRunning is similar to the function IsWindow in the Microsoft Windows API. For more information, please refer to the Windows API documentation.

### Examples

Verify the Microsoft Word document named test.doc is open and running:

```
AppIsRunning( "test.doc – Microsoft Word" )
```

## *AppPostMessage*

AppPostMessage is a built-in scripting function that sends a Windows system message to another application window.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| AppPostMessage | Module Activity | Synchronous | No | Supported | Supported | Not supported |

### Syntax

AppPostMessage(*strAppTitle*, *strMessage*, *numWParam*, *numLParam*{ | , *optNumTimeout* })

**strAppTitle**

The full title (as shown in the title bar) of the application window.

**strMessage**

The name or code of the system message.

> **Note:** The **CLOSE**, **MINIMIZE**, **MAXIMIZE** and **RESTORE** messages can be given as string values enclosed in quotes. All other message codes must be given as numeric values.

**numWParam**

Additional message-specific information.

**numLParam**

Additional message-specific information.

**optNumTimeout**

The timeout period (in milliseconds) for the function to be successfully executed. If, for whatever reason, the function is not executed in this period, then it is aborted.

This is an optional parameter. If no value is specified, then the default timeout is five seconds (or 5000 milliseconds).

### Returned value

This function will return one of the following values:

| Value | Description |
|---|---|
| 0 | ERROR: The system message was not sent, or the specified application window did not respond, within the timeout period. |

| Value | Description |
|---|---|
| 1 | SUCCESS: The system message was successfully sent. |

### Notes

AppPostMessage is similar to the function PostMessage in the Microsoft Windows API, and it allows many of the same options. For more information, including a list of available system messages, please refer to the Windows API documentation.

### Examples

Close the Calculator application:

```
AppPostMessage( "Calculator", "CLOSE", 3, 1 )
```

## *AppSendKeys*

Sends keyboard commands to the active application.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| AppSendKeys | Module Activity | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
AppSendKeys( strKeys1, strKeys2, … , strKeysN )
```

**strKeys (1-N)**

String tags containing the keyboard commands to be used.

### Returned value

No returned values.

### Examples

| Tag Name | Expression |
|---|---|
| | `AppSendKeys( "S", "t", "u", "d", "i", "o", "<ENTER>" )` |
| | `AppSendKeys( "<Alt>F" )` |

**Note:** You can specify **<ALT>**, **<CTRL>**, or **<SHIFT>** in the text to send a code equal to the Alt, Ctrl, or Shift keyboard commands. To send the < character, specify **<<** in the text.

## *CleanReadQueue*

CleanReadQueue is a built-in function that removes all read messages from the communications module.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| CleanReadQueue | Module Activity | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

```
CleanReadQueue()
```

```
CleanReadQueue()
```

This function has no parameters.

### Return value

This function returns no value.

### Notes

This function has been deprecated. You should not use it in new projects, but it is still supported in projects that were created using previous versions of EcoStruxure Machine SCADA Expert 2020.

## *CloseSplashWindow*

Closes the EcoStruxure Machine SCADA Expert 2020 splash window.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| CloseSplashWindow | Module Activity | Synchronous | No | Supported | Executed on Server | Executed on Server |

### Syntax

```
CloseSplashWindow()
```

This function takes no parameters.

### Returned value

No returned values.

### Examples

| Tag Name | Expression |
|---|---|
| | **CloseSplashWindow()** |

## *DisableMath*

`DisableMath` is a built-in function that pauses the execution of all Math worksheets.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| DisableMath | Module Activity | Asynchronous | No | Supported | Supported | Executed on Server |

### Syntax

**DisableMath()**

```
DisableMath()
```

This function has no parameters.

### Return value

This function returns no value.

### Notes

To resume the execution of Math worksheets, call the function `EnableMath`.

## *EnableMath*

`EnableMath` is a built-in function that resumes the execution of all Math worksheets.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `EnableMath` | Module Activity | Asynchronous | No | Supported | Supported | Executed on Server |

### Syntax

**`EnableMath()`**

`EnableMath()`

This function has no parameters.

### Return value

This function returns no value.

### Notes

In most cases, execution was paused by calling the function `DisableMath`.

## *EndTask*

`EndTask` is a built-in function that stops a specified execution task or runtime module that is currently running.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `EndTask` | Module Activity | Asynchronous | No | Supported | Executed on Server | Executed on Server |

### Syntax

**`EndTask(`*`strTask`*`)`**

`EndTask({ `*strTask*` | "{ BGTask | Core | DBSpy | XDB | Driver`*drivername*` | LogWin | MobileAccess | OPCClient | OPCUAClient | OPCUAServer | OPCXMLClient | OPCServer | TCPClient | TCPServer | Viewer }" })`

***strTask***

The name of the task or module to stop (must be one of the following):

| Value | Description |
|---|---|
| `BGTask` | Background Task |
| `Core` | Core Runtime |
| `DBSpy` | Watch |
| `XDB` | Database/ERP Runtime |
| `Driver`*`drivername`* | Driver Runtime (for the specified driver) |
| `LogWin` | LogWin |
| `MobileAccess` | Mobile Access Runtime |
| `OPCClient` | OPC DA 2.05 Client Runtime |
| `OPCUAClient` | OPC UA Client Runtime |

| Value | Description |
|---|---|
| OPCUAServer | OPC UA Server Runtime |
| OPCXMLClient | OPC XML/DA Client Runtime |
| OPCServer | Studio SCADA OPC Server |
| TCPClient | TCP/IP Client Runtime |
| TCPServer | TCP/IP Server Runtime |
| Viewer | Viewer |

For more information, see Runtime Tasks on page 136.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Failure |
| 1 | Success |

### Examples

Stop the MOTCP driver:

```
EndTask("DriverMOTCP")
```

Stop the Viewer module, which is used to view the local project runtime:

```
EndTask("Viewer")
```

## *Exec*

Exec is a built-in function that executes a command as if it were entered using the computer's command-line interface (CLI).

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Exec | Module Activity | Synchronous or Asynchronous | No | Supported | Supported | Not supported |

### Syntax

```
Exec(strCommand,optNumState,optNumSync,"optTagReturnOrHandle")
```

Exec(*strCommand*{ | ,{ *optNumState* | 0 | 1 | 2 | 3 | 4 | 7 }{ | ,{ *optNumSync* | 0 | 1 },"*optTagReturnOrHandle*" } })

***strCommand***

The command to be executed.

***optNumState***

A numeric flag that specifies the initial state of the Windows program (if any) that is run by the command:

| Value | Description |
|---|---|
| 0 | Hide the program and give control to another one. |

| Value | Description |
|---|---|
| 1 | Make the program active and display it. |
| 2 | Make the program active and display it as an icon. |
| 3 | Make the program active and maximize it. |
| 4 | Display the program at its most recent size. The program is still active. |
| 7 | Display the program as an icon. The program is still active. |

This parameter is optional; if no value is specified, the default value is 1.

This parameter is not supported when the project's target platform is **Embedded**; regardless of which value is specified, the function is executed with the default value.

***optNumSync***

A numeric flag that specifies whether the command is executed synchronously or asynchronously:

| Value | Description |
|---|---|
| 0 | Execute asynchronously. This function will return immediately. |
| 1 | Execute synchronously. This function will return when the execution is completed. |

This parameter is optional; if no value is specified, the default value is 0.

***optTagReturnOrHandle***

The name of a project tag that will receive feedback about the execution of the command:

- If the command is executed asynchronously, the tag will receive a handle that can used with the `ExecIsRunning` function to determine whether the command is still being executed.
- If the command is executed synchronously, the tag will receive the command's exit code. That exit code is separate from this function's own return value.

This parameter is optional, but given its nature, there is no default value.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

## Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Command was not executed successfully. |
| 1 | Command was executed successfully. |

This return value indicates only whether the execution started successfully, particularly if the command was executed asynchronously. It does not indicate when or how the execution was completed. To determine that, use the `ExecIsRunning` function.

## Examples

Start Notepad, display the program, and then immediately continue to the next line of the script:

```
Exec("C:\Windows\System32\notepad.exe",4)
```

Start MS Paint, make the program active, and then immediately continue to the next line of the script:

```
Exec("C:\Windows\System32\mspaint.exe")
```

Call an external batch file, execute it synchronously, hide the program, wait until the execution is completed, and then store the exit code in the tag **execReturn**:

```
Exec("CMD /C call C:\Temp\MyBatch.bat",0,1,"execReturn")
```

Call an external VBScript file, execute it asynchronously, hide the program, store the handle in the tag **execHandle**, and then continue to the next line of the script:

```
Exec("CMD /C call C:\Temp\MyScript.vbs",0,0,"execHandle")
```

## *ExecIsRunning*

`ExecIsRunning` is a built-in function that determines whether a command that was executed asynchronously by the `Exec` function is still being executed.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `ExecIsRunning` | Module Activity | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
ExecIsRunning(numHandle,"optTagReturn")
```

`ExecIsRunning(`*numHandle*{ | ,"*optTagReturn*" })

***numHandle***

The handle that was previously stored in the project tag specified by the *optStrReturnOrHandle* parameter of the `Exec` function.

***optTagReturn***

The name of a project tag that will receive the exit code returned by the command or program when the execution is completed.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 1 | Program is still running. |
| 0 | Execution has completed successfully (i.e., the program has closed). |
| -1 | Invalid number of parameters. |
| -2 | Invalid handle number. Check the value for *numHandle*. |
| -3 | Execution has completed, but the return tag is invalid. Check the value for *optTagReturn*. |

### Examples

Get the handle stored in the tag **execHandle**, and then determine whether that command is still being executed:

```
ExecIsRunning(execHandle)
```

Get the handle stored in the tag **execHandle**, determine whether that command is still being executed, and then store the exit code in the tag **execReturn**:

```
ExecIsRunning(execHandle,"execReturn")
```

## *ExitWindows*

Exits the Windows operating system in a specified manner.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ExitWindows | Module Activity | Asynchronous | No | Supported | Supported | Not supported |

### Syntax

ExitWindows(*numExitCode*)

**numExitCode**

A numeric code specifying how Windows should be exited:

| Value | Description |
|---|---|
| 0 | Restart |
| 1 | Log off |
| 2 | Shut down |

### Returned value

No returned values.

### Examples

| Tag Name | Expression |
|---|---|
| | **ExitWindows( 1 )** |

## *IsScreenOpen*

The function IsScreenOpen that a project screen is open on a client.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| IsScreenOpen | Module Activity | Asynchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
IsScreenOpen(strScreen,optNumID)
```

IsScreenOpen(*strScreen*{ | ,*optNumID*})

***strScreen***

The name of the project screen to be verified.

***optNumID***

The specific ID number of the screen. (This number is assigned when the screen is opened using the function `Open`.)

This parameter is optional; if no value is specified, the default value is 0.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Screen is not open. |
| 1 | Screen is open. |

### Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

### Examples

Is the screen named "main" open?

```
IsScreenOpen("main")
```

Is the screen named "main" with ID 10 open?

```
IsScreenOpen("main",10)
```

## *IsTaskRunning*

`IsTaskRunning` is a built-in function that checks whether a specified execution task or runtime module is currently running.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| IsTaskRunning | Module Activity | Synchronous | Yes | Supported | Executed on Server | Executed on Server |

### Syntax

```
IsTaskRunning(strTask)
```

IsTaskRunning({ *strTask* | "{ BGTask | Core | DBSpy | XDB | Driver*drivername* | LogWin | MobileAccess | OPCClient | OPCUAClient | OPCUAServer | OPCXMLClient | OPCServer | TCPClient | TCPServer | Viewer }" })

***strTask***

The name of the task or module to check (must be one of the following):

| Value | Description |
|---|---|
| BGTask | Background Task |
| Core | Core Runtime |
| DBSpy | Watch |
| XDB | Database/ERP Runtime |
| Driver*drivername* | Driver Runtime (for the specified driver) |

| Value | Description |
|---|---|
| LogWin | LogWin |
| MobileAccess | Mobile Access Runtime |
| OPCClient | OPC DA 2.05 Client Runtime |
| OPCUAClient | OPC UA Client Runtime |
| OPCUAServer | OPC UA Server Runtime |
| OPCXMLClient | OPC XML/DA Client Runtime |
| OPCServer | Studio SCADA OPC Server |
| TCPClient | TCP/IP Client Runtime |
| TCPServer | TCP/IP Server Runtime |
| Viewer | Viewer |

For more information, see Runtime Tasks on page 136.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Specified task is not running. |
| 1 | Specified task is running. |

### Examples

Check the MOTCP driver:

```
IsTaskRunning("DriverMOTCP")
```

Check the Viewer module, which is used to view the local project runtime:

```
IsTaskRunning("Viewer")
```

## *IsViewerInFocus*

Verifies that the project viewer (`Viewer.exe`) is in focus on the display.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| IsViewerInFocus | Module Activity | Synchronous | Yes | Supported | Not supported | Not supported |

### Syntax

```
IsViewerInFocus()
```

This function takes no parameters.

### Returned value

| | |
|---|---|
| 0 | Viewer is not in focus. |
| 1 | Viewer is in focus. |

### Examples

```
IsViewerInFocus()
```

## *KeyPad*

`KeyPad` is a built-in function that displays a Virtual Keyboard (for Thin Clients) or Data Input (for Mobile Access) dialog box in order to prompt the user to enter a value.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `KeyPad` | Module Activity | Asynchronous | No | Supported | Supported | Supported |

### Syntax

```
KeyPad("strTagName",optStrKeyboardName,optNumIsPassword,optStrHint,optNumMin,optNumMax,optNumESi
```

`KeyPad("`*strTagName*`"{ | ,`*optStrKeyboardName*`{ | ,`*optNumIsPassword*`{ | ,`*optStrHint*`{ | ,`*optNumMin*`,`*optNumMax*`{ | ,`*optNumESign*`{ | ,`*optStrConfirmation*`}}}}}})`

***strTagName***

The name of the project tag that will receive the entered value.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***optStrKeyboardName***

The type of Virtual Keyboard that will be displayed (e.g., **`AlphaNumeric`**, **`EnhKeypad`**, or **`Keypad`** ).

This parameter is optional; if no value is specified, the default type (as selected in the project settings) will be displayed.

This parameter is ignored when the function is executed on Mobile Access.

***optNumIsPassword***

A numeric switch that will cause the characters typed by the user to be displayed as asterisks (*). This is useful when the user must enter a password or some other value that should be obfuscated. To set this switch TRUE, specify any value other than 0.

This parameter is optional; if no value is specified, the default value is 0 (FALSE).

***optStrHint***

A hint or message to the user that will be displayed at the top of the dialog box, if the **Show Hint** option is selected in the project settings.

This parameter is optional; if either an empty string ("") or no value is specified, no hint will be displayed.

***optNumMin,optNumMax***

The minimum and maximum values that will be accepted as input. These values will also be displayed as Min and Max hints in the dialog box, depending on how it is configured. For Thin Clients, the keyboard type must be Keypad and the **Show MIN/MAX fields** option must be selected in the project settings. (These parameters are ignored for all keyboard types other than Keypad.) For Mobile Access, you only need to specify these values.

These parameters are optional, but you must specify both values in order to have them implemented. If you specify only one value — for example, Min but not Max— then it will be ignored.

***optNumESign***

A numeric switch that will require the user to e-sign the tag value change. The user will be prompted for their username and password, and the event will be recorded in the project log. To set this switch TRUE, specify any value other than 0.

This parameter is optional; if no value is specified, the default value is 0 (FALSE).

***optStrConfirmation***

A numeric switch that will cause a confirmation message to be displayed after the user enters the value. The user must acknowledge the message in order to continue. To set this switch TRUE, specify any value other than 0.

> **Note:** Confirmation cannot be automated or bypassed; only an actual key press, mouse click, or on-screen tap (depending on the client station) will acknowledge the message.

This parameter is optional; if no value is specified, the default value is 0 (FALSE).

## Returned value

This function will return one of the following possible values:

| Value | Description |
|---|---|
| 0 | Success |
| 1 | Error |
| 2 | Tag does not exist |
| 3 | Reentrant error, function is already executing |
| 4 | Invalid number of parameters |
| 5 | Internal error, contact Customer Support for more information |

## Notes

The Virtual Keyboard / Data Input dialog box is a standard interface for getting data input (i.e., numeric values and text) from the user on a client station that is equipped with a touchscreen instead of a physical keyboard. It can be invoked by several different screen objects and program features, in addition to this function. For more information, including examples of the dialog box itself, see Data Input on page 345.

## Examples

Display the default keyboard for the user to enter a value, and then write the entered value to tagA:

```
KeyPad("tagA")
```

Display the Enhanced Keypad for the user to enter a value, and then write the entered value to tagA:

```
KeyPad("tagA","EnhKeypad")
```

Display the Enhanced Keypad for the user to enter a value, obfuscate the characters typed by the user, and then write the entered value to tagA:

```
KeyPad("tagA","EnhKeypad",1)
```

Display the Enhanced Keypad for the user to enter a value, obfuscate the characters typed by the user, display "My Input" as a hint, with a Min of 0 and a Max of 100, and then write the entered value to tagA:

```
KeyPad("tagA","EnhKeypad",1,"My Input",0,100)
```

## *LogOff*

This function logs off the current user and then logs on the default user (typically "guest").

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| LogOff | Module Activity | Asynchronous | No | Supported | Supported | Supported |

### Syntax

```
LogOff()
```

This function takes no parameters.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Error. |
| 1 | Success. |

### Examples

```
LogOff()
```

## *LogOn*

LogOn is a built-in function that either logs on a specified user or displays a *Log On* dialog box.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| LogOn | Module Activity | Asynchronous | No | Supported | Supported | Supported |

### Syntax

```
LogOn(optStrUsername,optStrPassword)
```

LogOn({ | *optStrUsername*, *optStrPassword* })

***optStrUsername***

The name of the user to log on.

***optStrPassword***

The specified user's password.

*optStrUsername* and *optStrPassword* are optional parameters. If they are not specified, the project will instead display a *Log On* dialog box, to prompt the station's current operator — whoever it is — to log on.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Error (e.g., username or password is invalid) or cancellation. |

| Value | Description |
|---|---|
| 1 | Success. |

### Examples

Display a *Log On* dialog box:

```
LogOn()
```

Log on username Albert with password EMC2:

```
LogOn("Albert","EMC2")
```

## *Math*

Math is a built-in function that executes a specified Math worksheet.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Math | Module Activity | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

```
Math(numWorksheet)
```

Math(*numWorksheet*)

***numWorksheet***

The number of the Math worksheet to be executed.

### Return value

This function returns no value.

### Notes

Executing a Math worksheet from inside another module will pause that module until the Math worksheet finishes. Consequently, use this function only when absolutely necessary to avoid decreasing the performance of the other modules.

### Examples

Run Math worksheet "6":

```
Math(6)
```

## *PostKey*

PostKey is a built-in function that posts a virtual-key code to the currently displayed project screen. In other words, it synthesizes a keystroke.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| PostKey | Module Activity | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
PostKey(numKeydownOrKeyup,numwParam,numlParam)
```

`PostKey({` *numKeydownOrKeyup* `| 0 | 1 },` *numwParam*`,` *numlParam*`)`

***numKeydownOrKeyup***

A numeric option with the following possible values:

| Value | Description |
|---|---|
| 0 | KeyDown event (i.e., the key specified by *numwParam* is pressed) |
| 1 | KeyUp event (i.e., the key specified by *numwParam* is released) |

***numwParam***

The virtual-key code to be posted, in either decimal or hexadecimal format.

For a list of available codes, go to: msdn.microsoft.com/library/dd375731

***numlParam***

Additional message information.

This parameter is not typically used, so in most cases, the value should be `0` to indicate that there is no additional information.

## Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

When this function is executed, it can only post either a KeyDown event or a KeyUp event, depending on the value that is specified for *numKeydownOrKeyup*. It cannot post both events at the same time. Therefore, in order to post a complete keystroke (i.e., press and release), you must call this function twice — once for each event — as shown in the following example:

```
PostKey(0,36,0)
PostKey(1,36,0)
```

You can call other functions or execute other code between the two `PostKey` function calls, and if you do, that code will be executed as if the specified key is being held. You can even nest `PostKey` function calls in order to post key combinations (e.g., **Alt+F4**).

Alternatively, if it is not feasible to call the `PostKey` function twice, you can use a Windows Shell object and the `SendKeys` method in VBScript in order to produce a similar effect. Unlike the `PostKey` function, however, the `SendKeys` method only needs to be called once in order to send a complete keystroke or even multiple keystrokes. For example:

```
Dim WshShell
Set WshShell = CreateObject("WScript.Shell")
WshShell.SendKeys "{HOME}"
Set WshShell = Nothing
```

For more information about using the `SendKeys` method like this, go to: social.technet.microsoft.com/wiki/contents/articles/5169.vbscript-sendkeys-method.aspx

## Return value

This function returns no value.

## Notes

VBScript uses special notation to indicate numbers that are of a different base. Prepending with **`&h`** indicates a hexadecimal number, while prepending with **`&o`** indicates an octal number. As such, if you use this function in VBScript and you want to specify a hexadecimal value for *numwParam*, you must use the special notation. For example, **`&h24`** instead of the more typical **`0x24`**.

### Examples

Post the virtual-key code for the **Home** key, in decimal:

```
PostKey(0,36,0)
```

Post the virtual-key code for the **Home** key, in hexadecimal:

```
PostKey(0,0x24,0)
```

Post the virtual-key code for the **Home** key, in hexadecimal, using VBScript's special notation:

```
$PostKey(0,&h24,0)
```

## *Recipe*

`Recipe` is a built-in function that executes a specified Recipe worksheet.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `Recipe` | Module Activity | Synchronous | No | Supported | Executed on Server (see "Notes" below) | Executed on Server (see "Notes" below) |

### Syntax

```
Recipe(strOperation&File)
```

`Recipe`(*strOperation&File*)

***strOperation&File***

An expression that specified the operation to be performed and the recipe to be used, in the following format:

```
<operation>:<file>
```

*operation* must be one of the following:

| Operation | Description |
|---|---|
| `Save` | Save tag values to the data file. |
| `Load` | Load tag values from the data file. |
| `Delete` | Delete the data file that is specified in the Recipe worksheet. |
| `Init` | Initialize the data file with value of 0 for all included tags. |

*file* must be the name of the Recipe worksheet itself (e.g., `Recipe1`), not the name of data file that is specified in the worksheet.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Success. |
| 1 | Expression is numeric, not string. |
| 2 | Expression does not contain ":". |

| Value | Description |
|---|---|
| 3 | Invalid operation. |
| 4 | Recipe task not found. |
| 5 | Disk error. |

### Notes

The Background Task must be running on the project runtime server in order to execute Recipe worksheets. For more information, see Runtime Tasks on page 136.

When this function is called on a project thin client, it is executed on the project runtime server. As such, if the recipe includes any project tags that are configured with scope of **Local** rather than **Server**, those tags are updated only on the server. For more information, see Choosing the Tag Scope on page 156.

### Examples

Execute Recipe1 and save the tag values to the data file:

```
Recipe("Save:Recipe1")
```

Execute Recipe5 and load the tag values from the data file:

```
Recipe("Load:Recipe5")
```

## *Report*

`Report` is a built-in function that executes the specified Report worksheet and sends the output to hard disk, printer, or PDF.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `Report` | Module Activity | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

```
Report("strFunction",optNumOrientation)
```

`Report("`*strFunction*`"{ | ,`*optNumOrientation*`})`

***strFunction***

A string specifying the operation to perform and the Report worksheet to output, using the syntax "*`Operation:worksheet`*".

*Operation* must be one of the following:

| Value | Description |
|---|---|
| `Disk` | Save file to hard disk. |
| `Prn` | Send report to default printer. |
| `Pdf` | Generate a PDF file of the report. |

*worksheet* is the name of the Report worksheet file ( **`*.rep`** ) to be executed. The file name cannot contain spaces. If it does, save the worksheet again with a new name.

***optNumOrientation***

The orientation of the output:

| Value | Description |
|---|---|
| **0** | Portrait |
| 1 | Landscape |

This parameter is optional; if no value is specified, the default value is 0 (portrait). Also, this parameter is ignored if *Operation* is other than `Prn`.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Success. |
| 1 | *strFunction* is configured with a numeric value (invalid). |
| 2 | *strFunction* does not contain "**:**" (invalid). |
| 3 | *strFunction* contains an invalid output type before the "**:**". |
| 4 | Background Task is not running (see tip below). |
| 5 | Disk error (e.g., disk full, read-only file cannot be overwritten, or invalid path). |
| 6 | Specifed Report worksheet file does not exist. |

> **Tip:** The Background Task must be running in order to execute this function. Otherwise, the operation will not be executed and the function will return the value `4` indicating error. For more information, see Runtime Tasks on page 136.

### Notes

This function is based on legacy code, which means it cannot use printer settings that were previously configured by the `PrintSetup` function. Instead, it always uses the default printer on the computer or device that hosts the project runtime. You can use VBScript in your project to change the default printer in Windows, however. For example:

### Examples

Execute Report1 and save it to hard disk:

```
Report("Disk:Report1.rep")
```

Execute Report2 and send it to the default printer, using portrait orientation:

```
Report("Prn:Report2.rep",0)
```

Execute Report3 and send it to the default printer, using landscape orientation:

```
Report("Prn:Report3.rep",1)
```

Execute Report1 and generate a PDF:

```
Report("Pdf:Report1.rep")
```

## *RunGlobalProcedureAsync*

This function executes a global procedure asynchronously, in its own thread, so that it does not slow down or interfere with other running scripts. The procedure is run on the project server, but it can be called by any local or remote client.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| RunGlobalProcedureAsync | Module Activity | Asynchronous | No | Supported | Not supported (see "Notes" below) | Not supported |

### Syntax

RunGlobalProcedureAsync(*strProcedureName*{ | ,*optStrArgument1*,…,*optStrArgumentN*})

```
RunGlobalProcedureAsync( strProcedureName )
RunGlobalProcedureAsync( strProcedureName, optStrArgument1, …, optStrArgumentN )
```

**strProcedureName**

The name of the procedure (i.e., a VBScript function or sub-routine defined in the Procedures folder) to run asynchronously.

**optStrArgument1, ..., optStrArgumentN**

Values that are passed to the procedure's parameters. Arguments must be passed as strings.

### Returned value

If the procedure is successfully executed, then this function will return a thread ID that can be used with the RunGlobalProcedureAsyncGetStatus function. Otherwise, this function will return an error code:

| Value | Description |
|---|---|
| -1 | Function is not supported by thin clients. |
| -2 | Invalid number of parameters. You must specify at least the procedure name. |
| -3 | Maximum number of threads exceeded. See note. |
| -4 | Failed to compile VBScript parameters for execution. |
| -5 | Failed to start the thread execution. |
| -100 | Internal error. Please contact technical support. |

### Notes

It is important to note this function can be called only in background tasks (e.g., Math, Scheduler, Script) by the project runtime server. It cannot be directly called by any project viewer or thin client, even if it is on the same computer or device as the server, because the project viewer runs in its own thread separate from the project runtime server. To indirectly call this function, configure a Math or Script worksheet to execute on a tag/expression trigger, and then configure a project screen to activate the trigger when needed. For example, configure the worksheet to execute when the value of **MyTag** is 1, and then configure a Button object in a project screen to toggle the value of **MyTag**.

Also, the maximum number of VBScript threads that can be executed asynchronously is configured by manually editing the project file (i.e., <*project name*>.APP) to change the following setting:

```
[Script]
MaxAsyncThreads=8
```

The default number of threads is 8, but the only real limit is determined by the available system resources. Increasing the number of threads may decrease runtime performance.

### Examples

Given the following procedure that is defined in the Procedures folder...

```
Function AddMe(intNumber)
  If intNumber >= 6 Then
    AddMe = 0
  Else
    AddMe = intNumber + 2
  End If
End Function
```

...the procedure is run by calling the RunGlobalProcedureAsync function...

```
RunGlobalProcedureAsync( "AddMe", "2" )
```

...and the function returns a thread ID that can be used with the RunGlobalProcedureAsyncGetStatus function.

## *RunGlobalProcedureAsyncGetCurrent*

RunGlobalProcedureAsyncGetCurrent is a built-in function that gets the thread ID of the procedure from which it is called, as long as the procedure was originally run by calling the RunGlobalProcedureAsync function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| RunGlobalProcedureAsyncGetCurrent | Module Activity | Synchronous | No | Supported | Not supported (see "Notes" below) | Not supported |

### Syntax

```
RunGlobalProcedureAsyncGetCurrent()
```

RunGlobalProcedureAsyncGetCurrent()

This function has no parameters.

### Return value

If this function succeeds, it returns the thread ID of the procedure from which it was called. It is the same ID that was returned by the RunGlobalProcedureAsync function when that function was called to run the procedure, and the ID can be subsequently passed as an argument to the RunGlobalProcedureAsyncGetStatus function.

If this function fails, it returns one of the following possible values:

| Value | Description |
|---|---|
| -1 | Function was not called from a procedure running asynchronously. |

### Notes

It is important to note this function can be called only in background tasks (e.g., Math, Scheduler, Script) by the project runtime server. It cannot be directly called by any project viewer or thin client, even if it is on the same computer or device as the server, because the project viewer runs in its own thread separate from the project runtime server. To indirectly call this function, configure a Math or Script worksheet to execute on a tag/expression trigger, and then configure a project screen to activate the trigger when needed. For example, configure the worksheet to execute when the value of **MyTag** is 1, and then configure a Button object in a project screen to toggle the value of **MyTag**.

This function can be used only to get the thread ID of the procedure from which it is called, because that procedure runs asynchronously in its own thread. This function cannot be used to get the thread IDs of any

other procedures, even if those procedures were also run by calling the RunGlobalProcedureAsync function. If you need to build a list of all of the procedures that are currently running, you can have each procedure call this function for itself and then store all of the returned values.

## *RunGlobalProcedureAsyncGetStatus*

This function gets the status of one or more global procedures that were run asynchronously by calling the RunGlobalProcedureAsync function. Each procedure is run in its own thread, so that it does not slow down or interfere with other threads.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| RunGlobalProcedureAsyncGetStatus | Module Activity | Synchronous | No | Supported | Not supported (see "Notes" below) | Not supported |

### Syntax

RunGlobalProcedureAsyncGetStatus({ | *optNumThreadID* | "*optTagThreadIDs*","*optTagStatus*","*optTagParameters*" })

```
RunGlobalProcedureAsyncGetStatus( )
RunGlobalProcedureAsyncGetStatus( optNumThreadID )
RunGlobalProcedureAsyncGetStatus( "optTagThreadIDs", "optTagStatus",
 "optTagParameters" )
```

**optNumThreadID**

The thread ID returned by the RunGlobalProcedureAsync function, if the procedure was successfully executed.

**optTagThreadIDs**

The name of an Array tag that will receive the thread IDs of all currently running and recently completed threads.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

**optTagStatus**

The name of an Array tag that will receive the statuses of all currently running and recently completed threads.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

**optTagParameters**

The name of an Array tag that will receive the parameters of all currently running and recently completed threads.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Returned value

If the function succeeds, then the possible returned values depend on how the function was called:

- If the function was called with no parameters...

```
RunGlobalProcedureAsyncGetStatus( )
```

…then the returned value is the total number of threads that are currently running.

- If the function was called with only the optNumThreadID parameter…

```
RunGlobalProcedureAsyncGetStatus( optNumThreadID )
```

…then the returned value is either `0`, indicating that the thread is still running, or the value that was returned by the procedure.

- If the function was called with the Array tags…

```
RunGlobalProcedureAsyncGetStatus( "optTagThreadIDs", "optTagStatus",
 "optTagParameters" )
```

…then the tags will receive the appropriate values for all currently running and recently completed threads.

If the function fails, then it returns one of the following errors:

| Value | Description |
|---|---|
| `-1` | Function is not supported by thin clients. |
| `-2` | Invalid thread ID. |
| `-3` | Invalid optTagThreadIDs. |
| `-4` | Invalid optTagStatus. |
| `-5` | Invalid optTagParameters. |
| `-100` | Internal error. Please contact technical support. |

### Notes

It is important to note this function can be called only in background tasks (e.g., Math, Scheduler, Script) by the project runtime server. It cannot be directly called by any project viewer or thin client, even if it is on the same computer or device as the server, because the project viewer runs in its own thread separate from the project runtime server. To indirectly call this function, configure a Math or Script worksheet to execute on a tag/expression trigger, and then configure a project screen to activate the trigger when needed. For example, configure the worksheet to execute when the value of **MyTag** is 1, and then configure a Button object in a project screen to toggle the value of **MyTag**.

Also, when the call to `RunGlobalProcedureAsync` succeeds, it returns an ID for the thread created and starts running the procedure in that thread. The status of the thread is stored in an internal buffer and can be retrieved using the `RunGlobalProcedureAsyncGetStatus` function. The buffer gets cleared when:

- The `RunGlobalProcedureAsyncGetStatus` function has been called and the thread status is different from `0` (thread is running); or
- The maximum buffer size has been exceeded, the thread is no longer running, and a call to start a new thread has been made.

## *RunGlobalProcedureOnFalse*

This function runs a global procedure when the value of a specified project tag or expression becomes FALSE.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `RunGlobalProcedureOnFalse` | Module Activity | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
RunGlobalProcedureOnFalse("strCondition",strProcedureOnFalse)
```

`RunGlobalProcedureOnFalse("strCondition",strProcedureOnFalse)`

**strCondition**

A project tag or expression that can be evaluated as either FALSE (zero) or TRUE (non-zero).

> **Note:** The condition should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the condition instead.

**strProcedureOnFalse**

The name of the procedure (i.e., a VBScript function or sub-routine defined in the Procedures folder of your project) to run when the value of the specified tag/expression becomes FALSE.

## Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Error |
| 1 | Success |

## Notes

Once this function is called, it is continuously executed by the project runtime client (i.e., the Viewer) until either it or the project runtime server is stopped. That means every time the value of the condition becomes FALSE, the procedure is run. However, the procedure is run only once *when* the value of the condition becomes FALSE; it is not repeatedly run *while* the value of the condition is FALSE. For the procedure to run again, the value of the condition must become TRUE and then FALSE again.

Also, the function can be called more than once, so that the same procedure can be run by different triggers. The project runtime client manages the execution of all instances of the function.

The procedure is run on the client where this function was called. To run a procedure on the project server, use the function `RunGlobalProcedureOnServer`.

The value of strCondition is passed to the procedure as an argument, so the procedure should be written to receive it. For example:

```
Sub UsingOnFalse(strCondition)
   .
   .
   .
End Sub
```

Please note that you do not actually have to use the argument in your procedure, only that you should write the procedure to receive it.

No other arguments can be passed to the procedure.

To ensure compatibility with previous versions of EcoStruxure Machine SCADA Expert 2020, passing the argument is disabled by default in existing projects and enabled by default in new projects. To change this for your project, open your project file (*<project name>*.APP) in a text editor and then edit the following property:

```
[Options]
EnableTagNameOnRunGlobalProcedureOnTag=<0 (disabled) / 1 (enabled)>
```

## Examples

When the value of **TagOnFalse** becomes FALSE, run the procedure **UsingOnFalse**:

```
RunGlobalProcedureOnFalse("TagOnFalse","UsingOnFalse")
```

## *RunGlobalProcedureOnServer*

The function `RunGlobalProcedureOnServer` runs a specified VBScript procedure, as defined in the **Procedures** folder in the Project Explorer. The procedure is run on the project runtime server, but it can be triggered by any client that calls this function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `RunGlobalProcedureOnServer` | Module Activity | Synchronous | No | Supported | Supported | Supported |

### Syntax

**`RunGlobalProcedureOnServer(strProcedureName,optStrArgument1,…,optStrArgumentN)`**

`RunGlobalProcedureOnServer(`*strProcedureName*{ | ,*optStrArgument1*,…,*optStrArgumentN*})

***strProcedureName***

The name of the procedure (i.e., a VBScript function or sub-routine defined in the **Procedures** folder) to run on the project runtime server.

***optStrArgument1,...,optStrArgumentN***

Values that are passed to the procedure's parameters. Arguments must be passed as strings, but the procedure will interpret them as the correct data types. For more information, see "Examples" below.

### Returned value

This function returns whatever value that is returned by the procedure.

### Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

Also, you cannot call this function in a procedure that was itself run by calling the function `RunGlobalProcedureOnServer`. If you attempt to do so, then the function will return an error. This is to prevent a possible memory leak caused by nested or recursive function calls.

You can still call other procedures directly, as you normally would in VBScript, or you can use the function `Eval` in VBScript to dynamically determine the procedure you are calling.

### Examples

Given the following procedure that is defined in the **Procedures** folder…

```
Function AddMe(intNumber)
  If intNumber >= 6 Then
    AddMe = 0
  Else
    AddMe = intNumber + 2
  End If
End Function
```

…the procedure is run by calling the function `RunGlobalProcedureOnServer`…

```
RunGlobalProcedureOnServer("AddMe","2")
```

…and it returns a value of 4.

## *RunGlobalProcedureOnTrigger*

This function runs a global procedure when the value or quality of a specified tag changes.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| RunGlobalProcedureOnTrigger | Module Activity | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
RunGlobalProcedureOnTrigger("strTagName",strProcedureOnTrigger)
```

RunGlobalProcedureOnTrigger("*strTagName*",*strProcedureOnTrigger*)

**strTagName**

The name of a project tag.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

**strProcedureOnTrigger**

The name of the procedure (i.e., a VBScript function or sub-routine defined in the Procedures folder of your project) to run when the value or quality of the specified tag changes.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Error |
| 1 | Success |

### Notes

Once this function is called, it is continuously executed by the project runtime client (i.e., the Viewer) until either it or the project runtime server is stopped. That means every time the value or quality of the specified tag changes, the procedure is run. Also, the function can be called more than once, so that the same procedure can be run by different triggers. The project runtime client manages the execution of all instances of the function.

> **Tip:** The procedure is run on the client where the function was called. To run a procedure on the server, use the function RunGlobalProcedureOnServer.

The value or quality of the specified tag is passed to the procedure as an argument, so the procedure should be written to receive it. For example:

```
Sub MyProcedure(strTrigger)
  .
  .
  .
End Sub
```

In practice, this means either...

```
strTrigger = $tagname->Value
```

…or…

```
strTrigger = $tagname->Quality
```

…depending on which one changed. You can then use the value or quality in your procedure.

Please note that you do not actually have to use the argument, only that you should write the procedure to receive it.

No other arguments can be passed to the procedure.

To ensure compatibility with previous versions of EcoStruxure Machine SCADA Expert 2020, passing the argument is disabled by default in existing projects and enabled by default in new projects. To change this for your project, open your project file (*<project name>*.APP) in a text editor and then edit the following property:

```
[Options]
EnableTagNameOnRunGlobalProcedureOnTag=<0 (disabled) / 1 (enabled)>
```

### Examples

When the value or quality of the tag **MyInteger** changes, run the procedure **MyProcedure**:

```
RunGlobalProcedureOnTrigger("MyInteger","MyProcedure")
```

The equivalent of **MyInteger->Value** or **MyInteger->Quality**, depending on which one changed, is passed to the procedure as an argument.

## *RunGlobalProcedureOnTrue*

This function runs a global procedure when the value of a specified project tag or expression becomes TRUE.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| RunGlobalProcedureOnTrue | Module Activity | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
RunGlobalProcedureOnTrue("strCondition",strProcedureOnTrue)
```

RunGlobalProcedureOnTrue("*strCondition*",*strProcedureOnTrue*)

**strCondition**

A project tag or expression that can be evaluated as either FALSE (zero) or TRUE (non-zero).

> **Note:** The condition should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the condition instead.

**strProcedureOnTrue**

The name of the procedure (i.e., a VBScript function or sub-routine defined in the Procedures folder of your project) to run when the value of the specified tag/expression becomes TRUE.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Error |
| 1 | Success |

### Notes

Once this function is called, it is continuously executed by the project runtime client (i.e., the Viewer) until either it or the project runtime server is stopped. That means every time the value of the condition becomes TRUE, the procedure is run. However, the procedure is run only once *when* the value of the condition becomes TRUE; it is not repeatedly run *while* the value of the condition is TRUE. For the procedure to run again, the value of the condition must become FALSE and then TRUE again.

Also, the function can be called more than once, so that the same procedure can be run by different triggers. The project runtime client manages the execution of all instances of the function.

The procedure is run on the client where this function was called. To run a procedure on the project server, use the function `RunGlobalProcedureOnServer`.

The value of strCondition is passed to the procedure as an argument, so the procedure should be written to receive it. For example:

```
Sub UsingOnTrue(strCondition)
   .
   .
   .
End Sub
```

Please note that you do not actually have to use the argument in your procedure, only that you should write the procedure to receive it.

No other arguments can be passed to the procedure.

To ensure compatibility with previous versions of EcoStruxure Machine SCADA Expert 2020, passing the argument is disabled by default in existing projects and enabled by default in new projects. To change this for your project, open your project file (*<project name>*.APP) in a text editor and then edit the following property:

```
[Options]
EnableTagNameOnRunGlobalProcedureOnTag=<0 (disabled) / 1 (enabled)>
```

### Examples

When the value of **TagOnTrue** becomes TRUE, run the procedure **UsingOnTrue**:

```
RunGlobalProcedureOnTrue("TagOnTrue","UsingOnTrue")
```

## *RunVBScript*

Executes a statement in VBScript language.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| RunVBScript | Module Activity | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
RunVBScript ( strScript, "optTagReturnError" )
```

**strScript**

Script statement that must be executed by the function.

**optTagReturnError**

Name of the tag that will receive the error (if any) generated by the statement (e.g., "Division by zero"). The tag name must be configured between double-quotes and it must be a String tag.

### Returned value

| | |
|---|---|
| 0 | Error |
| 1 | Success |

### Examples

| Tag Name | Expression |
|---|---|
| **TagResult** | **RunVBScript("MsgBox(Time)")** // Executes the MsgBox function from VBScript and displays the current time. |
| | **RunVBScript(TagStatement)** // Executes the statement configured in the value of the string tag **TagStatement**. |
| | **RunVBScript("$TagC=$TagA/$TagB", "TagError")** // Writes in **TagC** the result of **TagA** divided by **TagB**. The error generated by the operation (if any) is written to the string tag **TagError** (e.g., "Division by zero"). |

> **Tip:** This function is useful to execute VBScript statements from interfaces that support the built-in language only (e.g., Scheduler groups). You can also call VBSCript functions created in the Global Procedures.

> **Note:** The runtime station must support the VBScript statements configured in this function in order to execute them.

## *SecureViewerReload*

`SecureViewerReload` is a built-in function that closes the Secure Viewer program and then reloads it with a new configuration file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SecureViewerReload` | Module Activity | Synchronous | No | Not supported | Secure Viewer only | Not supported |

### Syntax

**SecureViewerReload(*strFileName*)**

`SecureViewerReload(`*strFileName*`)`

***strFileName***

The file path of an INI file (`*.ini`) that describes the new configuration. (If the file is located in the same folder as `Viewer.exe`, then only the file name is needed.) The file should be structured the same and contain all of the same settings as the default configuration file (`Viewer.ini`).

This parameter must specify either the name of a String tag or a text string enclosed in quotes.

### Return value

This function returns no value.

### Examples

Reload the Secure Viewer with the configuration file that is specified by the tag **configFile1**:

**SecureViewerReload(configFile1)**

Reload the Secure Viewer with the configuration file that is located at the specified file path:

```
SecureViewerReload("C:\Program Files\Secure Viewer\Bin\Config1.ini")
```

## SendKeyObject

The function `SendKeyObject` sends a key event code to objects in the currently displayed project screen. You can use this function to trigger Command animations on those objects.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SendKeyObject | Module Activity | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
SendKeyObject(numEvent,strMainKey,numShift,numCtrl,numAlt,strTargetScreen,optNumID)
```

SendKeyObject({ *numEvent* | 0 | 1 | 2 }, *strMainKey*{ | ,{ *numShift* | 0 | 1 },{ *numCtrl* | 0 | 1 },{ *numAlt* | 0 | 1 }, *strTargetScreen*{ | , *optNumID* } })

***numEvent***

A numeric value that indicates the type of key event to send to the screen. The following values are accepted:

| Value | Description |
|---|---|
| 0 | On Down |
| 1 | On While |
| 2 | On Up |

For more information about these key events, see Command animation on page 316.

> **Note:** If the "On While" event is specified, the "On While" script on the Command animation is executed just once for each time this function is executed. It is not continuously executed as if the key is pressed and held down, because this function does not have a parameter for specifying duration. If you want to cause that sort of behavior, you can include this function in an appropriately configured FOR loop.

***strMainKey***

The key to be sent to the screen. The following values are accepted:

| Value | Description |
|---|---|
| "A" ... "Z" | alphabetic characters A through Z |
| "+" | plus symbol |
| "-" | minus symbol |
| "*" | multiply symbol |
| "/" | divide symbol |
| "LEFT" | left arrow (←) |
| "UP" | up arrow (↑) |
| "RIGHT" | right arrow (→) |
| "DOWN" | down arrow (↓) |

| Value | Description |
|---|---|
| "HOME" | **Home** key |
| "END" | **End** key |
| "PAGEUP" | **Page Up** key |
| "PAGEDOWN" | **Page Down** key |
| "INSERT" | **Insert** key |
| "DELETE" | **Delete** key |
| "SPACE" | **Space** key |
| "RETURN" | **Return** key |
| "BACKSPACE" | **Backspace** key (if different from **Delete** key) |
| "ESCAPE" | **Escape** key |
| "F1" ... "F20" | function keys **F1** through **F20** |

The key must be enclosed in quotes, as shown.

***numShift***

A numeric value that indicates whether to include **Shift** with the specified key (e.g., **Shift+R**). The following values are accepted: `0` is no, `1` is yes.

This parameter is optional; if no value is specified, the default value is `0`.

***numCtrl***

A numeric value that indicates whether to include **Ctrl** with the specified key (e.g., **Ctrl+R**). The following values are accepted: `0` is no, `1` is yes.

This parameter is optional; if no value is specified, the default value is `0`.

***numAlt***

A numeric value that indicates whether to include **Alt** with the specified key (e.g., **Alt+R**). The following values are accepted: `0` is no, `1` is yes.

This parameter is optional; if no value is specified, the default value is `0`.

***strTargetScreen***

The name of the screen that will receive the key event code.

This parameter is optional; if no value is specified, the currently active screen is used.

***optNumID***

The specific ID number of the screen. (The ID number is assigned when the screen is opened with the function `Open`.)

This parameter is optional; if no value is specified, the default value is 0.

## Return value

This function returns no value.

## Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

Also, the parameters *numShift*, *numCtrl*, *numAlt* and *strTargetScreen* are all optional, but if you configure one of them, you must configure the others as well.

### Examples

Send **R** to the currently active screen:

```
SendKeyObject(0,"R")
```

Send **Ctrl+Shift+R** to the screen named "main" with ID 10:

```
SendKeyObject(0,"R",1,1,0,"main",10)
```

## *SetAppPath*

Sets the new file path for the project folder. After this function is executed, EcoStruxure Machine SCADA Expert will look for all of the project files (i.e.,screens, alarms, trends, database, events) in this folder.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetAppPath | Module Activity | Synchronous | No | Supported | Executed on Server | Not supported |

### Syntax

SetAppPath(*strPath*)

**strPath**

The file path.

### Returned value

| | |
|---|---|
| 0 | Failed to set path. |
| 1 | Succeeded in setting path |

### Examples

| Tag Name | Expression |
|---|---|
| | **SetAppPath( "C:\Studio\" )** |

> **Note:** If the computer is on a network, you can use the *//IP address or host name/Path* syntax to define a location on another node of the network.

## *SetViewerInFocus*

SetViewerInFocus is a built-in function that moves the Viewer program window in front of all other open windows.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetViewerInFocus | Module Activity | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
SetViewerInFocus()
```

SetViewerInFocus()

This function has no parameters.

### Return value

This function returns no value.

### Notes

For the purposes of this function, "Viewer program" is a generic term that includes:

- The local Viewer that is part of the project runtime for Windows
- The standalone Secure Viewer program for Windows

It does not include any version of Mobile Access, which uses a different technology to display project screens in web browsers.

Security features in the Windows operating system prevent program windows from moving themselves in front of other open windows without user input. As such, when this function is executed, the Viewer program will request the user's attention by blinking its icon in the Windows taskbar. (Some anti-virus software may also flag this as suspicious behavior.) It is only when the user selects the Viewer program that the program window will move to the front.

To work around this limitation, you must call this function at least once in your project's Startup Script. Allow twenty seconds more for your project to finish starting up, and then after that, any additional calls of this function should work as expected.

If you are not satisfied with the run-time behavior of this function, you can use the `AppActivate` function instead to achieve similar results. Also, you can use the `SetViewerPos` function to change the size and position of the Viewer program window, if necessary.

## *SetViewerPos*

`SetViewerPos` is a built-in function that sets the height, width, and position of the project viewer or thin client window.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SetViewerPos` | Module Activity | Synchronous | No | Supported | Supported | Not supported |

### Syntax

**`SetViewerPos(`*`numLeft`*`,`*`numTop`*`,`*`optNumWidth`*`,`*`optNumHeight`*`)`**

`SetViewerPos(`*numLeft*`,` *numTop*`{ | ,`*optNumWidth*`,` *optNumHeight*`})`

***numLeft***

The position (in pixels) of the left edge of the Viewer program window.

***numTop***

The position (in pixels) of the top edge of the Viewer program window.

***optNumWidth***

The width (in pixels) of the Viewer program window.

This parameter is optional; if no value is specified, the default value is the project's current display resolution.

***optNumHeight***

The height (in pixels) of the Viewer program window.

This parameter is optional; if no value is specified, the default value is the project's current display resolution.

### Return value

If the function fails, the return value is zero (0).

If the function succeeds, the return value is one (1).

### Notes

For the purposes of this function, "Viewer program" is a generic term that includes:

- The local Viewer that is part of the project runtime for Windows
- The standalone Secure Viewer program for Windows

It does not include any version of Mobile Access, which uses a different technology to display project screens in web browsers.

If you use this function in your project, make sure the **Start Maximized** option in your project settings is cleared. For more information, see Viewer tab on page 117.

### Examples

Set the Viewer so that its top-left corner is at 50x,50y and its size is 640x480:

```
SetViewerPos(50,50,640,480)
```

## *ShutDown*

This function stops all execution tasks and runtime modules, effectively shutting down the project.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ShutDown | Module Activity | Asynchronous | No | Supported | Supported (see "Notes" below) | Supported (see "Notes" below) |

### Syntax

```
ShutDown()
```

```
ShutDown()
```

This function takes no parameters.

### Returned value

This function does not return any value.

### Notes

This function only stops the project runtime server. It does not close the development environment on the server, if it happens to be open.

If this function is called from a project screen on a thin client, it only stops the Viewer module on that thin client. On Mobile Access, it returns the user to the logon screen.

Also, when this function is used in project screens on Mobile Access, it cannot be called from the **Screen_OnClose** sub-routine in the Screen Script.

## *StartTask*

StartTask is a built-in function that starts a specified execution task or runtime module.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StartTask | Module Activity | Asynchronous | No | Supported | Executed on Server | Executed on Server |

### Syntax

```
StartTask(strTask)
```

`StartTask({ strTask | "{ BGTask | Core | DBSpy | XDB | Driver | LogWin | MobileAccess | OPCClient | OPCUAClient | OPCUAServer | OPCXMLClient | OPCServer | TCPClient | TCPServer | Viewer }" })`

***strTask***

The name of the task or module to start (must be one of the following):

| Value | Description |
|---|---|
| `BGTask` | Background Task |
| `Core` | Core Runtime |
| `DBSpy` | Watch |
| `XDB` | Database/ERP Runtime |
| `Driver` | Driver Runtime (for all drivers; see "Notes" below) |
| `LogWin` | LogWin |
| `MobileAccess` | Mobile Access Runtime |
| `OPCClient` | OPC DA 2.05 Client Runtime |
| `OPCUAClient` | OPC UA Client Runtime |
| `OPCUAServer` | OPC UA Server Runtime |
| `OPCXMLClient` | OPC XML/DA Client Runtime |
| `OPCServer` | Studio SCADA OPC Server |
| `TCPClient` | TCP/IP Client Runtime |
| `TCPServer` | TCP/IP Server Runtime |
| `Viewer` | Viewer |

For more information, see Runtime Tasks on page 136.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| `0` | Failure |
| `1` | Success |

### Notes

If you use this function to start the Driver Runtime (e.g., `StartTask("Driver")`), it starts all of the drivers that are configured in your project. To start a specific driver, use the `Exec` function instead.

### Examples

Start the Viewer module, which is used to view the local project runtime:

```
StartTask("Viewer")
```

## *TaskUpdateConfig*

`TaskUpdateConfig` is a built-in function that updates a runtime task in order to reload a task worksheet during project run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TaskUpdateConfig | Module Activity | Asynchronous | No | Supported | Not supported | Not supported |

## Syntax

**TaskUpdateConfig(*strTask*,*numWorksheet*)**

TaskUpdateConfig ( { *strTask* | " { Math | Scheduler | Script | TCPClient | OPCUAClient | OPCXMLClient } " } , *numWorksheet* )

***strTask***

The name of the runtime task to be updated. It must be one of the following string values:

| Value | Description |
|---|---|
| Math | Math worksheets, which are executed as part of the Background Task |
| Scheduler | Scheduler worksheets, which are executed as part of the Background Task |
| Script | Script worksheets, which are executed as part of the Background Task |
| XDB | Database/ERP |
| OPCDAClient | OPC DA Client (Legacy) |
| OPCUAClient | OPC UA Client |
| OPCXMLClient | OPC XML/DA Client |
| TCPClient | TCP/IP Client |

For more information, see Runtime Tasks on page 136.

***numWorksheet***

The number of the worksheet to be reloaded.

## Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Failure (e.g., invalid task name, specified task not started, specified worksheet not found) |
| 1 | Success |

## Notes

If you use the project development enviroment to edit and then save a task worksheet in a project running locally, that worksheet is automatically and immediately reloaded by the project. You do not need to use this function to reload the worksheet.

This function is used to reload worksheet files that have been created or edited outside the project development environment and then manually added to the project folder. As such, it is not intended for typical users of this software. If you want more information about how to use this function, please contact your software distributor.

Worksheet files must be located in the project folder, and each worksheet file name must include both the type of worksheet and the worksheet number. For example, Math worksheet number 3 is saved at the following location:

**<*project name*>\Config\MATH003.MAT**

The specified task must be started before this function is called; calling this function does not start the specified task. If the task was not automatically started when the project was run, you can use either the *Runtime Tasks* command or the StartTask on page 1073 function to manually start the task.

### Examples

Update the Math task in order to reload worksheet number 3 (MATH003.MAT):

```
TaskUpdateConfig("Math",3)
```

## *ViewerPostMessage*

The function ViewerPostMessage posts a Windows System Message to the specified project screen.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ViewerPostMessage | Module Activity | Asynchronous | No | Supported | Supported | Not supported |

### Syntax

```
ViewerPostMessage(strScrTitle,numMessage,numwParam,numlParam,optNumID)
```

ViewerPostMessage(*strScrTitle*, *numMessage*, *numwParam*, *numlParam*{ | ,*optNumID*})

***strScrTitle***

The name of the screen to which the message will be posted.

***numMessage***

The number of the Windows System Message to be posted.

***numwParam***

Additional, message-specific information that is passed to **wParam** of the Windows System Message.

***numlParam***

Additional, message-specific information that is passed to **lParam** of the Windows System Message.

***optNumID***

The specific ID number of the screen. (This number is assigned when the screen is opened using the function Open.)

This parameter is optional; if no value is specified, the default value is 0.

### Return value

This function returns no value.

### Notes

This function emulates the PostMessage function in Microsoft Windows. For more information, including a complete list of available Windows System Messages, go to: msdn.microsoft.com/library/ms644944

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

### Examples

Send message 16 to the screen named "main" with ID 10:

```
ViewerPostMessage("main",16,3,1,10)
```

## Multimedia functions

These functions are used to play external audio and video files.

### *Play*

Plays a specified WAV audio file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `Play` | Multimedia | Asynchronous | No | Supported | Supported | Not supported |

> **Note:** For this function to work on a Thin Client, the target WAV file must be located in the same file path on the remote station.

#### Description

Plays a specified WAV audio file.

#### Syntax

`Play(`*strFileName*`{ | ,`*optNumSynchronous*`})`

**strFileName**

The file path and name of the WAV file to play.

**optNumSynchronous**

A numeric flag specifying whether the function executes synchronously or asynchronously:

| Value | Description |
|---|---|
| `0` | Asynchronous (i.e., the project continues without waiting for the function to return) |
| `1` | Synchronous (i.e., the project pauses while it waits for the function to return) |

This is an optional paramter; if no value is specified, then the default is `0`.

#### Return value

This function returns no value.

#### Examples

| Tag Name | Expression |
|---|---|
| | `Play( "C:\Sounds\Wav\alarm.wav" )` |

## Screen functions

These functions are used to open and close project screens.

### *Close*

`Close` is a built-in scripting function that closes an open project screen.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `Close` | Screen | Asynchronous | No | Supported | Supported | Supported |

#### Syntax

```
Close(optStrScreen,optNumID)
```

`Close ( optStrScreen { | , optNumID } )`

***optStrScreen***

The name of the screen to be closed. If this parameter is omitted, then the currently active screen will be printed.

***optNumID***

The specific ID or instance number of the screen to be closed, if there is more than one screen with the same name open. (The ID is assigned when the screen is opened with the `Open` function.)

This parameter is optional; if no value is specified, the default ID is 0.

#### Return value

This function returns no value.

#### Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

Also, in some cases, you do not need to call this function to close a screen because the screen will be closed automatically when another screen replaces it. For more information, see Screen Attributes on page 231.

#### Examples

Close the screen named "main":

```
Close("main")
```

Close the currently active screen:

```
Close()
```

Close the screen named "alarms":

```
Close("alarms")
```

Close the screen named "main" with ID 10:

```
Close("main",10)
```

## Open

`Open` is a built-in function that opens a project screen or screen group on the thin client with some specified settings.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `Open` | Screen | Asynchronous | No | Supported | Supported | Supported (see "Notes" below) |

### Syntax

```
Open(strScreenAndProperties,optNumX1,optNumY1,optNumX2,optNumY2,optNumResizeFlag,optNumID,optStr
```

`Open(`*strScreenAndProperties*{ `|` ,*optNumX1*{ `|` ,*optNumY1*,*optNumX2*,*optNumY2*,{ *optNumResizeFlag* `|` `0` `|` `1` }}}{ `|` ,*optNumID*}{ `|` ,*optStrMnemonicList*})

***strScreenAndProperties***

The name of the project screen or screen group to be opened. The screen file extension (either `.scc` or `.scr`) is assumed, so you do not need to include it if you are opening an individual project screen. If you are opening a screen group, however, you must include the screen group file extension (`.sg`).

> **Note:** If you have two screen files with the same name but different extensions in your project folder (e.g., `MyScreen.scc` and `MyScreen.scr`), the one with the preferred extension — as determined by whether the **Use .scr extension for screen files** option in the project settings is selected — will be opened. For more information, see Viewer tab on page 117.

If you specify only the screen name, it will be opened with its default properties (as specified in Screen Attributes). You can configure the properties as follows:

| Property | Syntax |
|---|---|
| Title Bar | `EnableTitleBar: Enable \| Disable; TitleBar:` *title* |
| System Menu | `SystemMenu: Enable \| Disable` |
| Minimize Box | `Minimize: Enable \| Disable` |
| Maximize Box | `Maximize: Enable \| Disable` |
| Style | `Style: Overlapped \| Popup \| Replace(Partial) \| Dialog \| Replace(Complete)` |
| Border | `Border: None \| Thin \| Resizing` |

Concatenate the screen name and the customized properties as a single string, using semicolons (`;`) to separate the properties in the string. For more information, see "Examples" below.

There is also a Clear Data property (`ClearData`) that applies only to screens opened in Mobile Access. When a screen is opened in Mobile Access, some of that screen's data is cached so that it can be quickly opened again. This is useful when certain screens are repeatedly closed and then reopened. However, if you see unexpected or unwanted behavior when you open a screen in this way, you may enable the Clear Data property (e.g., `ClearData: Enable`) to clear the cached data and open the screen as if for the first time.

***optNumX1*,*optNumY1*,*optNumX2*,*optNumY2***

The coordinates, in pixels, for the top-left (X1,Y1) and bottom-right (X2,Y2) corners of the screen.

These paramaters are optional; if no values are specified, the default screen size and position are used.

Please note the following special circumstances:

- You can open the screen at the mouse's current position by using **`Open("screen",1)`**, or **`Open("screen",1,-1,-1,-1,…)`** if the parameters at the end are needed.
- If *optNumX1* equals *optNumX2* and *optNumY1* equals *optNumY2*, the default screen size is used but the screen is centered at (X1,Y1).
- If *optNumX2* is less than *optNumX1* and/or *optNumY2* is less than *optNumY1*, or if all four parameters are set to **`-1`**, the parameters are ignored and the default screen size and position are used.

***optNumResizeFlag***

Specifies whether objects in the screen will be resized when the screen is opened:

| Value | Description |
|---|---|
| 0 | Screen objects will not be resized. |
| 1 | Screen objects will be automatically resized to fit the new dimensions of the screen, as specified by the coordinates described above. The resizing is done at the moment the screen is opened, so if the user changes the screen size after the screen is opened, the objects will not be resized again. |

This parameter is required if all four coordinates (*optNumX1*, *optNumY1*, *optNumX2*, *optNumY2*) are specified.

> **Note:**
>
> By default, text associated with a screen object (e.g., the caption on a Button object) is **not** resized along with the object. This might result in unexpected behavior during project run time, but it is done to prevent more serious issues that might result from automatically resizing fonts.
>
> If you want to automatically resize fonts, manually edit your project file (*<project name>*.APP) to add the following entry:
>
> ```
> [Desktop]
> ResizeFontWithScreen=1
> ```

***optNumID***

An ID or instance number to be assigned to the screen, because you can open multiple instances of the same screen file. (This ID is required when a screen is closed using the `Close` function.)

This parameter is optional; if no value is specified, the default value is 0.

***optStrMnemonicList***

A string that describes how the custom properties (also known as mnemonics) of screen objects and linked symbols will be completed when the screen is opened. This string must have the following syntax...

```
#Label:Value
```

...where *Label* is the name of the custom property and *Value* is the tag, expression, or literal value that the property will receive. If *Value* is a tag or expression, it will be evaluated when this function is executed.

You can declare more than one custom property as long as they are separated by spaces. You can also specify an external text file that contains the custom properties separated by either spaces or line returns. The file must have the .mne extension, and it must be located in the Web sub-folder of your project folder. If you want to save the file in a deeper sub-folder, you must specify a file path that is relative to the Web sub-folder. The file extension is assumed, so you do not need to include it in the file name.

For more information about how to use this parameter, see "Examples" below. For more information about custom properties in general, see Use custom properties to set property values when screens are opened on page 330.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Invalid parameter(s). |
| 1 | Valid parameters. |

The function only checks whether the parameters are valid, before it tries to use those parameters to open the screen. The function does not return any value to indicate whether the screen is successfully opened.

### Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

There are two known issues with using this function on Mobile Access. First, most of the additional properties for the *strScreenAndProperties* parameter are not supported. Only the Clear Data property (`ClearData`) is supported, and it applies only to screens opened in Mobile Access. Second, if you use the *optStrMnemonicList* parameter to complete custom properties in the specified project screen, but the screen also contains some VBScript that tries to set the same custom properties, the conflict will cause a VBScript compilation error and the screen will not open. For example, if you make the following function call to open a project screen…

```
Open("Screen",-1,-1,-1,-1,0,0,"#Mne3:1")
```

…but the screen contains a Button object that has the following VBScript attached to it…

```
$#Mne3:=1234
```

…the VBScript will not compile and the screen will not open.

### Examples

Open the screen using the default screen size, position, and ID:

```
Open("main")
```

Open the screen at the mouse's current position:

```
Open("main",1)
```

Open the screen at the mouse's current position and assign it an ID of 10:

```
Open("main",1,-1,-1,-1,0,10)
```

Open the screen using the default screen size but centered at the coordinates (500,250), do not resize the screen objects, and assign the screen an ID of 10:

```
Open("main",500,250,500,250,0,10)
```

Open the screen using the default screen size and position, use the default ID, and replace the custom properties **Mne1** and **Mne2** with **Tag1** and **Tag2**, respectively:

```
Open("main",-1,-1,-1,-1,0,0,"#Mne1:Tag1 #Mne2:Tag2")
```

Open the screen using the default screen size and position, use the default ID, and replace the custom properties with values defined in the mnemonics file located at: *<project name>*\Web\mnemonic.mne

```
Open("main",-1,-1,-1,-1,0,0,"mnemonic")
```

Open the screen using the default screen size but centered at the coordinates (500,250), do not resize the screen objects, assign the screen an ID of 10, and replace the custom properties with values defined in the mnemonics file located at: *<project name>*\Web\mnemonic.mne

```
Open("main",500,250,500,250,0,10,"mnemonic")
```

Open the screen using the default screen size and position, assign it an ID of 2, and replace the custom properties with values defined in the mnemonics file located at: *<project name>*\Web\Mnemonics\mnemonic.mne

```
Open("main",-1,-1,-1,-1,0,2,"Mnemonics\mnemonic")
```

Open the screen using the default screen size and position, do not resize the screen objects, use the default ID, and customize the screen properties:

```
Open("main; EnableTitleBar: Enable; TitleBar: Main Screen; SystemMenu: Enable;
 Maximize: Disable; Minimize: Disable; Style: Popup; Border: Thin",-1,-1,-1,-1,0,0,"")
```

Open the screen in Mobile Access and clear any cached data so the screen is opened as if for the first time:

```
Open("main; ClearData: Enable")
```

## *OpenPrevious*

OpenPrevious is a built-in scripting function that re-opens the last screen to be closed.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| OpenPrevious | Screen | Asynchronous | No | Supported | Supported | Supported |

### Syntax

```
OpenPrevious(optNumX1,optNumY1,optNumX2,optNumX2)
```

OpenPrevious({ | *optNumX1*,*optNumY1*,*optNumX2*,*optNumY2*})

***optNumX1***
***optNumY1***
***optNumX2***
***optNumY2***

The coordinates, in pixels, for the upper-left (X1,Y1) and lower-right (X2,Y2) corners of the screen.

These are optional parameters. If no values are specified, then the default screen size and location are used. For more information, see Screen Attributes on page 231.

### Returned value

This function returns one of the following values:

| Value | Description |
|---|---|
| 0 | Failure |
| 1 | Success |

### Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

### Examples

Open the previous screen using its default size and location:

```
OpenPrevious()
```

Open the previous screen in the top-left corner of the display and sized to 800x600:

```
OpenPrevious(0,0,800,600)
```

## *ShowInplaceInput*

`ShowInplaceInput` is a built-in function that shows a simple input box at a specified position on the project thin client.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ShowInplaceInput | Screen | Asynchronous | No | Supported | Supported | Not supported |

### Syntax

```
ShowInplaceInput("strOutputTag",numStartXPos,numStartYPos,optNumMin,optNumMax,optNumEnablePasswo
```

`ShowInplaceInput("`*strOutputTag*`",` *numStartXPos*`,` *numStartYPos*`{ | ,`*optNumMin*`,` *optNumMax*`{ | ,{` *optNumEnablePasswordMode* `| 0 | 1}{ | ,{` *optNumShowOSVK* `| 0 | 1}}}}})`

***strOutputTag***

The name of a project tag that will receive the input. The data type of the tag should be appropriate for the type of input that you want to get from the user. For example, an Integer tag cannot receive text input.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***numStartXPos***

The starting X position of the top-left corner of the input box — that is, the number of pixels between that corner and the left side of the thin client display.

This is the "starting" position because the user can move the input box after it is shown.

***numStartYPos***

The starting Y position of the top-left corner of the input box — that is, the number of pixels between that corner and the top of the thin client display.

This is the "starting" position because the user can move the input box after it is shown.

***optNumMin***

The minimum numeric value that will be accepted as input.

This parameter is optional; if no value is specified, any value will be accepted.

***optNumMax***

The maximum numeric value that will be accepted as input.

This parameter is optional; if no value is specified, any value will be accepted.

***optNumEnablePasswordMode***

An option to enable password mode, which obfuscates the operator's input as if it is a password:

| Value | Description |
|---|---|
| 0 | Show input as plain text. |
| 1 | Obfuscate input. |

This parameter is optional; if no value is specified, the default value is 0.

***optNumShowOSVK***

An option to show the operating system's virtual keyboard for user input:

| Value | Description |
|---|---|
| 0 | Do not show virtual keyboard. |
| 1 | Show virtual keyboard. |

This parameter is optional; if no value is specified, the default value is 0. Also, this parameter is relevant only when the thin client is running on an operating system that has a built-in virtual keyboard, such as Windows Embedded. It cannot be used to show EcoStruxure Machine SCADA Expert 2020's own Virtual Keyboard.

**Note:**

## Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Success. |
| -1 | Invalid tag specified for *tagOutput*. |
| -2 | Invalid number of parameters. |
| -3 | Thin client (a.k.a. Viewer) is not running. |

## Notes

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

### Examples

Show the input box at X 50, Y 50, and then save the input to the tag **UserInput**:

```
ShowInplaceInput("UserInput",50,50)
```

Show the input box at X 50, Y 50, and then save the input — which must be between 1 and 100 — to the tag **UserInput**:

```
ShowInplaceInput("UserInput",50,50,1,100)
```

Show the input box at X 50, Y 50 with the virtual keyboard, and then save the input — which must be between 1 and 100 — to the tag **UserInput**:

```
ShowInplaceInput("UserInput",50,50,1,100,0,1)
```

## *ShowMessageBox*

ShowMessageBox is a built-in function that shows a message box with one or more buttons to capture the user's response.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ShowMessageBox | Screen | Synchronous | No | Supported | Supported | Supported |

### Syntax

```
ShowMessageBox(strMessage,optNumButtons,optStrTitle)
```

ShowMessageBox(*strMessage*{ | ,{ *optNumButtons* | 0 | 1 | 2 | 3 | 4 | 5 }{ | ,*optStrTitle* } })

***strMessage***

The message body that will be displayed in the box.

***optNumButtons***

A numeric flag that specifies which set of buttons to display in the message box:

| Value | Description |
|---|---|
| 0 | **OK** button (default) |
| 1 | **OK** / **Cancel** buttons |
| 2 | **Abort** / **Retry** / **Ignore** buttons |
| 3 | **Yes** / **No** / **Cancel** buttons |
| 4 | **Yes** / **No** buttons |
| 5 | **Retry** / **Cancel** buttons |

> **Tip:** To add an exclamation icon to the box — that is, to make it an alert or warning rather than a plain message — add 48 (vbExclamation) to this parameter. For more information, see "Examples" below.

This parameter is optional; if no value is specified, the default value is 0.

***optStrTitle***

The text that is displayed in the title bar of the message box.

This parameter is optional; if no value is specified, no title is displayed.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -1 | Bad parameter. |
| 0 | Message box not displayed because the Viewer is not open; see "Notes" below. |
| 1 | User clicked **OK**. |
| 2 | User clicked **Cancel**. |
| 3 | User clicked **Abort**. |
| 4 | User clicked **Retry**. |
| 5 | User clicked **Ignore**. |
| 6 | User clicked **Yes**. |
| 7 | User clicked **No**. |

### Notes

Unlike other Screen functions, this function can be called from Global Procedures and Script worksheets, and when it is, the message box is displayed in the Viewer module on the project runtime server. If the Viewer module is not open — that is, if the Viewer task is not running — the message box cannot be displayed and the function returns `0`.

When this function is used in project screens on Mobile Access, it has been enhanced to duplicate the functionality of the VBScript function `MsgBox`.

> **Tip:** The message, button labels, and title (if any) can all be displayed in other languages during run time. For more information, see Project Localization on page 681.

### Examples

Display a plain message with an **OK** button:

```
ShowMessageBox("The action could not be completed.")
```

Display a question with **Yes** / **No** buttons:

```
ShowMessageBox("Continue with action?",4)
```

Display an alert with an **OK** button and a title:

```
ShowMessageBox("The action could not be completed.",0+48,"Alert")
```

## Security functions

These functions are used to manage users and groups in the project's security system.

### *BlockUser*

`BlockUser` is a built-in scripting function that blocks an existing user from logging onto a project. This allows you to disable a user account without deleting it.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `BlockUser` | Security | Synchronous | No | Supported | Supported | Not supported |

#### Syntax

`BlockUser(`*strUserName*`)`

**strUserName**

The name of the user to block.

#### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | User blocked successfully. |
| 1 | Invalid number of parameters. |
| 2 | Wrong parameter type. |
| 3 | Specified user does not exist. |
| 4 | User currently logged on does not have the rights to block (i.e., user does not have **Edit Security System**). |
| 5 | The operation on the distributed security system failed. |
| 6 | User cannot be blocked. |
| 7 | The current Security Mode does not allow user to be blocked/unblocked. |
| 8 | Internal error. |

#### Notes

If you use this function to block a user, then the option **User is blocked** is selected in the *User Account* dialog. For more information, see Creating and configuring users on page 671.

#### Examples

Block the user named Bob:

```
BlockUser( "Bob" )
```

Block the user named in position 3 of the array badUsers:

```
BlockUser( badUsers[3] )
```

Block the user that is currently logged on:

```
BlockUser( UserName )
```

## *CheckESign*

`CheckESign` is a built-in function that electronically signs a run-time event with a user name. You can call this function to secure scripts and expressions, just as you can select the **E-Sign** option to secure screen objects and animations.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `CheckESign` | Security | Synchronous | No | Supported | Supported | Supported |

### Syntax

**`CheckESign(`*`optNumLevel`*`,`*`optStrUserName`*`,`*`optStrPassword`*`)`**

`CheckESign({ | `*optNumLevel*`{ | ,`*optStrUserName*`{ | ,`*optStrPassword*`} } })`

***optNumLevel***

The security level (from 0 to 255, Runtime only) to which the user must have access in order to sign.

This parameter is optional; if no value is specified, the default value is 0.

***optStrUserName***

The name of a user configured in the project security system.

This parameter is optional; if no value is specified, a dialog box is displayed in the project viewer for the user to enter their user name and password.

***optStrPassword***

The password for the user specified by *optStrUserName*.

This parameter is optional; if no value is specified, a dialog box is displayed in the project viewer for the user to enter their password. (The user name is automatically filled.)

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -10000 | Function called outside the project viewer, without a project viewer running. |
| 0 | User name and/or password not accepted, or the user does not have access to the specified security level. |
| 1 | User name and password accepted. The event is saved in the event history with the user's signature. |

### Notes

Users, groups, and security levels are all managed in the project security system. For more information, see Project Security on page 639.

### Examples

Prompt the user for their user name and password — the user must have access to security level 20:

```
CheckESign(20)
```

Prompt the current user, as determined by the **`UserName`** system tag, for their password:

```
CheckESign(20,UserName)
```

This function is often called as the **If** condition in an **If**…**Then**…**Else** statement, so that the value returned by this function determines the result of the statement.

## *CheckSecurityLevel*

The function `CheckSecurityLevel` checks whether the current user has access to the specified security level.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `CheckSecurityLevel` | Security | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
CheckSecurityLevel(numLevel,optNumType)
```

`CheckSecurityLevel(`*numLevel*`{ | ,{ `*optNumType*` | 0 | 1 }})`

***numLevel***

The security level to be checked.

***optNumType***

The type of security level to check: `0` is Runtime, `1` is Development.

This is an optional parameter; if no value is specified, the default value is `0`.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| `0` | The current user does not have access to the specified security level. |
| `1` | The current user has access to the specified security level. |

### Notes

Each user can belong to multiple groups, and those groups typically have access to different security levels. Rather than check the groups to which a user belongs, and from that try to determine the security levels to which the user has access, you can use this function to check the security levels directly.

For more information about groups and security levels, see Group Account dialog on page 664.

### Examples

The user "Bob" belongs to the groups "GroupA" and "GroupB". GroupA has access to runtime levels 10 to 20, and GroupB has access to development levels 30 to 40. That means Bob has access to runtime security levels 10 to 20 and development security levels 30 to 40.

Given this, if Bob is the current user (i.e., the user who is currently logged onto the client where the function is executed) and this function is called to check for access to runtime security level 17…

```
CheckSecurityLevel(17)
```

…it returns a value of `1`, meaning that Bob has access.

Similary, if this function is called to check for access to development security level 33…

```
CheckSecurityLevel(33,1)
```

…it again returns a value of `1`.

However, if this function is called to check for access to runtime security level 25…

```
CheckSecurityLevel(25,0)
```

…it returns a value of `0`, meaning that Bob does not have access.

## *CreateUser*

The function `CreateUser` creates a new user in your project's security system.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `CreateUser` | Security | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
CreateUser(optStrUserName,optStrGroup,optStrPassw,optStrUserFullName)
```

`CreateUser({ | optStrUserName, optStrGroup{ | , optStrPassw{ | , optStrUserFullName } } })`

***optStrUserName***

***optStrGroup***

The name of the user to be created, and the group(s) to which the user will belong. You can specify multiple groups by separating them with a comma.

These are optional parameters; if no values are specified, a dialog box will be displayed on the client so that the current user can provide the information.

***optStrPassw***

The user's password.

This is an optional parameter; if no value is specified, the user will not have a password. (To specify one later, either call the function `SetPassword` or edit the user's settings in the project security system.)

***optStrUserFullName***

The full name of the user.

This is an optional parameter; if no value is specified, the user will not have a full name. (To specify one later, edit the user's settings in the project security system.)

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| `-1000` | Could not display dialog box. The function should executed on the client. |
| `-1` | Internal error. Please contact Support. |
| `0` | New user created successfully. |
| `1` | Invalid number of parameters. |
| `2` | Wrong parameter type. |
| `3` | User name already exists. |
| `4` | Group does not exist. |
| `5` | Failed to save to configuration file. |
| `6` | Invalid user. |
| `7` | User full name already exists. |

| Value | Description |
|---|---|
| 8 | Reentrant function call not allowed. |
| 9 | User clicked **Cancel** on the *Create User* dialog box. |
| 10 | Invalid password, check the minimum password size specified for the group. |
| 11 | Invalid group. (Group may not have **Runtime group** option selected.) |
| 12 | *Create User* dialog box is already displayed, cannot display another dialog box. (For example, if the user clicked **OK** without providing all of the required information.) |
| 13 | Current user does not have sufficient privileges to create users (i.e., the option **Edit Security System** is not selected in the Security System settings). |
| 14 | The current Security Mode does not allow a user to be created. |

### Notes

Users created with this function have the **Runtime user** option selected in the *User Account* dialog. For more information, see Creating and configuring users on page 671.

### Examples

Display the *Create User* dialog box on the client, to get the information from the current user:

```
CreateUser()
```

Create a user named "Bob" in the group "Admin", with the password "Chocolate":

```
CreateUser("Bob","Admin","Chocolate")
```

Create a user named "Albert" (full name "Albert Jones") in the group "Engineering", with the password "EMC2":

```
CreateUser("Albert","Engineering","EMC2","Albert Jones")
```

## *ExportSecuritySystem*

This function exports the security system configuration to an encrypted file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ExportSecuritySystem | Security | Synchronous | No | Supported | Not supported | Not supported |

### Syntax

`ExportSecuritySystem(`*strFileName*`,` *strPassword*`)`

```
ExportSecuritySystem( strFileName, strPassword )
```

**strFileName**

The complete file path and name where you want to save the configuration file.

**strPassword**

The main password for the security system. This same password will be used to protect the exported file.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -2 | Wrong parameter type. |
| -1 | Invalid number of parameters. |
| 0 | Couldn't write security data. |
| 1 | File exported successfully. |

### Notes

This function can be called only from scripts executed on the project server. None of the connected clients — not even the client running on the same computer as the project server, because it runs as a separate process on that computer — have the necessary access to the security system. Therefore, generally speaking:

- It can be called from the Startup Script (which is executed when the project itself is run), Script Groups (which are continuously executed by the Background Task), and any Global Procedures called by them; and
- It cannot be called from the Graphics Script (which is executed separately by each client), Screen Scripts (which are attached to individual screens), and Command animations.

For more information, see VBScript Interfaces in the Software on page 1205.

To work around this limitation, create a Script Group that actually calls the function, configure an appropriate tag/expression trigger in the **Execution** box of the Script worksheet, and then create a project screen that somehow changes the value of that tag/expression. Therefore, when a user on a connected client uses the screen to change the value, the Script Group will be executed on the project server.

### Examples

```
ExportSecuritySystem( "C:\security.txt" )
```

```
ExportSecuritySystem( "C:\security.txt", "mypa55w0rd" )
```

## *GetLastESignUser*

`GetLastESignUser` is a built-in function that gets the last user who electronically signed an event during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetLastESignUser | Security | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
GetLastESignUser()
```

`GetLastESignUser()`

This function has no parameters.

### Return value

This function returns (as a string) the name of the last user who electronically signed an event during run time. (Such events are generated when the user uses a screen object that has the **E-Sign** option selected or triggers a script that calls the function `CheckESign`. For more information, see Events on page 394.) If the user failed to provide a valid username and password, or if the user clicked **Cancel** to exit the *E-Sign* dialog box, this function returns an empty string ("").

### Notes

This function gets the last user on the client where the function is executed. If the function is executed on the project runtime server, it gets the last user of the server's local Viewer.

## *GetSecuritySystemStatus*

This function gets the status of the security system and its connection to the authentication server, when the security mode is either Distributed–Client or Domain (LDAP).

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetSecuritySystemStatus | Security | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
GetSecuritySystemStatus({ | { optNumType | 0 | 1 | 2 }})
```

```
GetSecuritySystemStatus()
GetSecuritySystemStatus( optNumType )
```

**optNumType**

The type of action to take to update the status.

| Value | Description |
|---|---|
| 0 | Perform a fast check using either Ping or Bind (depending on the server settings), but take no other actions. |
| 1 | Force reload of users and groups from the authentication server. |
| 2 | Clear cached users and groups. |

This is an optional parameter; if no value is specified, then the default is 0.

### Returned value

This function returns one of the following possible values:

| Value | Security Mode is Distributed–Client | Security Mode is Domain (LDAP) |
|---|---|---|
| 0 | No cache | Connection timeout |
| 1 | Updated cache | Bind timeout |
| 2 | Outdated local cache | Query timeout |
| 3 | Outdated server cache | Disconnected |
| 4 | Disconnected from server | Connected |
| 5 | N/A | No users or groups returned by query |
| 6 | N/A | Invalid user or group |

### Notes

This function returns the same value that is sent to the project tag configured in the **Status Tag** box, in the security system server settings. However, this function returns the value immediately, while the project tag configured in the **Status Tag** box is only updated periodically (according to **Synchonization Period** for Distributed–Client or **Retry Interval** for Domain (LDAP)). As such, there may be times when this function's returned value and the value of the project tag do not match.

Also, there are other actions besides calling this function that update the status:

- When a user logs on to the project. Specifically, if the user logs on via the built-in *LogOn* dialog (invoked by either calling the LogOn function or selecting the **LogOn** menu command in the Viewer), then the status is updated before the dialog is displayed.
- When the authentication server is offline and the retry interval (configured in the security system server settings) has elapsed.
- When the security system settings are opened in the development application.

Whenever the status is updated, the new value is immediately sent to the project tag configured in the **Status Tag** box.

### Examples

Get the status of the security system:

```
GetSecuritySystemStatus()
```

Force the security system to reload all users and groups from the authentication server:

```
GetSecuritySystemStatus( 1 )
```

## *GetUserFullName*

This function gets the full name (if any) of a specified user in the project security system.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetUserFullName | Security | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

GetUserFullName(*strUserName*,"*tagUserFullName*")

**strUserName**

The name of a user in the project security system.

**tagUserFullName**

The name of a tag (String type) that will receive the full name of the specified user. If the specified user does not have a full name defined, then the tag will receive an empty string ("").

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Specified user does not exist. |
| 1 | Success; specified user exists. |

### Examples

Get the full name of the currently logged user (via the system tag **UserName**):

```
GetUserFullName( UserName, "UserFullName" )
```

Get the full name of the user "engineer1":

```
GetUserFullName( "engineer1", "UserFullName" )
```

## GetUserNames

GetUserNames is a built-in function that gets a list of users that match a specified type and then stores the list in an array tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetUserNames | Security | Synchronous | No | Supported | Executed on Server | Not supported |

### Syntax

```
GetUserNames("tagUsers",optNumUserType,"opttagGroups")
```

GetUserNames("*tagUsers*"{ | ,*optNumUserType*{ | ,"*optTagGroups*" }})

***tagUsers***

The name of the array tag (String type) that will receive the list of users, starting at array position 0.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***optNumUserType***

A numeric flag that indicates the type of users to get:

| Value | Description |
|---|---|
| 0 | All users. |
| 1 | Only users that have the **Runtime user** option selected. |
| 2 | Only users that do not have the **Runtime user** option selected. |

The **Runtime user** option is a setting in the *User Account* dialog, in the project security system. It indicates that the user can be affected during run time by the Security functions, including this one. It is selected automatically if a user is created during run time — for example, by calling the CreateUser function — but it can also be selected or cleared manually. For more information, see Creating and configuring users on page 671.

This parameter is optional; if no value is specified, the default value is 0.

***optTagGroups***

The name of the array tag (String type) that will receive a list of the user groups to which the users belong, starting at array position 0. Each user's group(s) will be written to the corresponding array position. For example, for the user stored in **MyUsersArray[5]**, its group(s) are stored in **MyGroupsArray[5]**.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional; if no value is specified, the list of user groups will not be saved.

### Returned value

This function returns the number of users that match the specified type, or a negative number that can be one of the following:

| Value | Description |
|---|---|
| -1 | Invalid number of parameters. |
| -2 | *tagUsers* is invalid. |

| Value | Description |
|---|---|
| -3 | *optNumUserType* is invalid. |
| -4 | *opttagGroups* is invalid. |
| -5 | Error, function cannot be called in the Thin Client. |

### Examples

Get all users, and then store the user names in **MyUsersArray**:

```
GetUserNames("MyUsersArray")
```

Get all users that have the **Runtime user** option selected, and then store the user names in **MyUsersArray**:

```
GetUserNames("MyUsersArray",1)
```

Get all users that do not have the **Runtime user** option selected, store the user names in **MyUsersArray**, and store the corresponding group names in **MyGroupsArray**:

```
GetUserNames("MyUsersArray",2,"MyGroupsArray")
```

## *GetUserPwdAging*

The function GetUserPwdAging gets the age of the password for a specified user — that is, the time remaining until the password expires, or if it has expired, the time since it expired.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetUserPwdAging | Security | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
GetUserPwdAging(strUser)
```

GetUserPwdAging(*strUser*)

***strUser***

The name of the user.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| less than 0 | Number of hours since the password expired. |
| 0 | The specified user is not logged on. |
| greater than 0 | Number of hours until the password expires. |

> **Note:** If the function is not executed correctly (e.g., if the specified user name is invalid), or if the specified user is not logged on, then the function also returns BAD quality.

### Examples

Get the age of the password for the user "John":

```
GetUserPwdAging("John")
```

Get the age of the password for the user who is currently logged on:

```
GetUserPwdAging(UserName)
```

## *GetUserState*

Use to see the current status of a selected user.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetUserState | Security | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

GetUserState(*strUserName*)

**strUserName**

The name of the user

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -3 | Specified user does not exist. |
| -2 | Wrong parameter type. |
| -1 | Invalid number of parameters. |
| 0 | Specified user is unblocked. |
| 1 | Specified user is blocked. |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **GetUserState( "Bob" )** |
| Tag | **GetUserState ( "Albert" )** |

## *ImportSecuritySystem*

This function imports a security system configuration from an external file.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ImportSecuritySystem | Security | Synchronous | No | Supported | Not supported | Not supported |

### Syntax

```
ImportSecuritySystem( strSecuritySystemPassword, strFileName, strFilePassword, optNumMode
 )
```

`ImportSecuritySystem(`*strSecuritySystemPassword*`,` *strFileName*`,` *strFilePassword*`{ | ,{` *optNumMode* `| 0 | 1 | 2}})`

**strSecuritySystemPassword**

The main password for the project's current security system configuration. (The security system must be enabled.)

**strFileName**

The complete file path and name of the configuration file that you want to import. (The file must have been previously exported from a EcoStruxure Machine SCADA Expert project using either the *Security System* configuration tool or the function `ExportSecuritySystem`.)

**strFilePassword**

The password for the specified configuration file.

**optNumMode**

A numeric flag indicating how the imported settings should be handled:

| Value | Description |
|---|---|
| `0` | Append the imported settings to the current settings. In the event of a conflict, replace with the imported settings. |
| `1` | Append the imported settings to the current settings. In the event of a conflict, keep the current settings. |
| `2` | Completely replace the current settings with the imported settings. |

This parameter is optional; if no value is specified, then the default value is `0`.

## Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| `-2` | Wrong parameter type. |
| `-1` | Invalid number of parameters. |
| `0` | Couldn't read security data. |
| `1` | File imported successfully. |

## Notes

This function can be called only from scripts executed on the project server. None of the connected clients — not even the client running on the same computer as the project server, because it runs as a separate process on that computer — have the necessary access to the security system. Therefore, generally speaking:

- It can be called from the Startup Script (which is executed when the project itself is run), Script Groups (which are continuously executed by the Background Task), and any Global Procedures called by them; and
- It cannot be called from the Graphics Script (which is executed separately by each client), Screen Scripts (which are attached to individual screens), and Command animations.

For more information, see VBScript Interfaces in the Software on page 1205.

To work around this limitation, create a Script Group that actually calls the function, configure an appropriate tag/expression trigger in the **Execution** box of the Script worksheet, and then create a project screen that somehow changes the value of that tag/expression. Therefore, when a user on a connected client uses the screen to change the value, the Script Group will be executed on the project server.

### Examples

```
ImportSecuritySystem( "curr3ntPa55w0rd", "C:\security.txt", "1mp0rtPa55w0rd" )

ImportSecuritySystem( "curr3ntPa55w0rd", "C:\security.txt", "1mp0rtPa55w0rd", 2 )
```

## *RemoveUser*

The function `RemoveUser` removes a specified user from your project's security system.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `RemoveUser` | Security | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
RemoveUser(strUserName)
```

`RemoveUser(`*strUserName*`)`

***strUserName***

The name of the user to be removed.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | The specified user was successfully removed. |
| 1 | Invalid number of parameters. |
| 2 | Wrong parameter type. |
| 3 | The current user does not have sufficient privileges to remove other users (i.e., the user does not have the option **Edit Security System** selected in the Security System settings). |
| 4 | The specified user cannot be removed. |
| 5 | The specified user does not exist. |
| 6 | Component-level failure (e.g., the LDAP server returned an error). |
| 7 | Failed to save changes to the configuration file. |
| 8 | The current security mode does not allow the specified user to be removed. |

### Examples

Remove the user named "Bob":

```
RemoveUser("Bob")
```

Remove the user specified by the tag **`InvalidUser`**:

```
RemoveUser(InvalidUser)
```

> **Note:** You can use this function to remove only those users who were created during project run time using the function `CreateUser`. You cannot use this function to remove users who were

created using the Security System configuration tools in the project development environment. For more information, see Project Security on page 639.

## SetPassword

`SetPassword` is a built-in function that sets a new password for a specified user in your project's security system.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SetPassword` | Security | Synchronous | No | Supported | Supported | Not supported |

### Syntax

**`SetPassword(optStrUserName,optStrNewPassword,optStrConfirmPassword,optStrCurrentPassword)`**

`SetPassword({ | optStrUserName{ | ,optStrNewPassword{ | ,optStrConfirmPassword{ | ,optStrCurrentPassword } } } })`

***optStrUserName***

The name of the user.

This parameter is optional; if no value is specified, a *Set Password* dialog box will be displayed on the client station so that the current user can select the name.

***optStrNewPassword***

The new password for the specified user.

This parameter is optional; if no value is specified, a *Set Password* dialog box will be displayed on the client station so that the current user can enter their new password.

***optStrConfirmPassword***

The new password again, to confirm that it has been entered correctly.

This parameter is optional; if no value is specified, a *Set Password* dialog box will be displayed on the client station so that the current user can enter their new password.

***optStrCurrentPassword***

The current (i.e., old) password for the specified user, to authorize the change.

This parameter is optional; if no value is specified and the function has been called by the current user in order to set their own password, a *Set Password* dialog box will be displayed on the client station so that they can enter their current password.

The current password is not required at all if this function has been called by an administrative user in order to set the password of another user.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -1000 | Could not display dialog box, because the function was called by the project runtime server. The function should be called on the client station. |
| -1 | Internal error. Please contact Customer Support. |
| 0 | Password set successfully. |
| 1 | Invalid number of parameters. |
| 2 | Wrong data type passed to parameter. |
| 3 | The specified user does not exist. |

| Value | Description |
|---|---|
| 4 | Reentrant call not allowed. |
| 5 | User clicked **Cancel** in the *Set Password* dialog box. |
| 6 | The specified group does not exist. |
| 7 | The specified password is too weak (i.e., it does not meet the requirements specified in the Group Account settings). |
| 8 | Invalid password. |
| 9 | Invalid user. |
| 10 | The current user does not have sufficient privileges to set user passwords (i.e., the **Edit Security System** option is not selected in the Group Account settings). |
| 11 | Server offline. |
| 12 | Communication error (e.g., the function was called on remote client that cannot communicate with the LDAP server). |
| 13 | Confirmation does not match new password. |
| 14 | *Set Password* dialog box is already displayed, cannot display another dialog box. (For example, if the user clicked **OK** without providing all of the required information.) |
| 15 | The current security mode does not allow user passwords to be changed. |

## Notes

If you do not want the *Set Password* dialog box to be displayed on the client station, you must provide valid arguments to all of the function paramaters. Otherwise, the dialog box will be displayed in order to get the remaining information from the current user.

In order to successfully execute this function, the current user must have sufficient privileges to set passwords, regardless of whether they are setting their own password or another user's password. If they do not have sufficient privileges, this function returns a value of 10 as described in "Return value" above. For more information, see Group Account dialog on page 664.

If your project's security mode is set to **Domain (LDAP)**, changing passwords is subject to LDAP server criteria and detailed error messages will be included in the project's run-time log (i.e., the *Output* window). For more information about security modes, see About security modes on page 640. For more information about LDAP server criteria and error messages, see the documentation for your specific LDAP server.

## Examples

Display the *Set Password* dialog box on the client, to get the information from the current user:

```
SetPassword()
```

Change the password for user "Admin", but display the dialog box in order to get the new password and authorization:

```
SetPassword("Admin")
```

Change the password for user "admin" to the value of the tag **newPassword**, but display the dialog box in order to get authorization:

```
SetPassword("Admin",newPassword,newPassword)
```

Change the password for user "admin" to the value of the tag **newPassword**, without displaying the dialog box:

```
SetPassword("Admin",newPassword,newPassword,"DLfVU89Y")
```

## *SetUserGroup*

`SetUserGroup` is a built-in function that sets the security group for a specified user. It can either overwrite or append to the user's current list of assigned groups.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SetUserGroup` | Security | Synchronous | No | Supported | Supported | Not supported |

### Syntax

**`SetUserGroup(`*`strUserName`*`,`*`strGroupName`*`,`*`optNumAppend`*`)`**

`SetUserGroup (` *strUserName* `,` *strGroupName* `{ | , {` *optNumAppend* `| 0 | 1 } } )`

***strUserName***

The name of the user that will have its group set.

***strGroupName***

The name of the group to be set for the specified user. To specify multiple groups, separate them with commas.

***optNumAppend***

A numerical flag that indicates how the group should be set:

| Value | Description |
|---|---|
| 0 | Overwrite the user's current list of assigned groups. |
| 1 | Append to the user's current list of assigned groups. |

This parameter is optional; if no value is specified, the default value is `0`.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -1 | Internal error. |
| 0 | Group set successfully. |
| 1 | Invalid number of parameters. |
| 2 | Wrong data type for parameter. |
| 3 | The specified user does not exist. |
| 4 | The specified group does not exist. |
| 5 | Failed to save changes to the Security System configuration file. |
| 6 | Invalid user (i.e., the **Runtime user** option is not selected for the specified user). |
| 7 | Invalid group (i.e., the **Runtime group** option is not selected for the specified group). |
| 8 | The current user does not have sufficient privileges to make changes (i.e., the current user does not belong to a group that has the **Edit Security System** option selected). |
| 9 | The current security mode (e.g., Domain) does not allow user groups to be changed. |

### Notes

This function does not send events to the Event Logger.

### Examples

Set the user "John" to the group "Admin":

```
SetUserGroup("John","Admin")
```

Set the user "Bob" to the groups "Admin" and "Operator":

```
SetUserGroup("Bob","Admin,Operator",0)
```

Add the current user to the group "Maintenance":

```
SetUserGroup(UserName,"Maintenance",1)
```

## *UnblockUser*

This function unblocks a blocked user in the security system.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| UnblockUser | Security | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
UnblockUser( strUserName )
```

**strUserName**

The name of the user to unblock.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | User unblocked successfully. |
| 1 | Invalid number of parameters. |
| 2 | Wrong parameter type. |
| 3 | Specified user does not exist. |
| 4 | User currently logged on does not have the rights to unblock (i.e., user does not have **Edit Security System**). |
| 5 | The operation on the distributed security system failed. |
| 6 | Specified user cannot be blocked. |
| 7 | The current Security Mode does not allow user to be blocked/unblocked. |
| 8 | Internal error. |

### Notes

If you use this function to unblock a user, then the option **User is blocked** is cleared in the *User Account* dialog. For more information, see Creating and configuring users on page 671.

### Examples

| Tag Name | Expression |
| --- | --- |
| Tag | `UnblockUser( "Bob" )` |
| Tag | `UnblockUser( "Albert" )` |

## Statistical functions

These functions are used to get certain statistics — such as average, maximum, and minimum — from two or more numeric values.

### *Avg*

Calculates the average value of a set of numbers.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Avg | Statistical | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

```
Avg( numValue1, numValue2, … , numValueN )
Avg( "tagArray", numSample, optNumIgnore )
```

> **Note:**
>
> This function has two formats:
>
> - If the first parameter is a numeric tag or value, you must use the **`Avg( numValue1, numValue2, … , numValueN )`** format.
> - If the first parameter is an array tag in double-quotes or a string tag, you must use the **`Avg( "tagArray", numSample, optNumIgnore )`** format.

**numValue (1...N)**

Integer or Real tags containing the numbers to be averaged together.

**tagArray**

Name of array tag (Real or Integer) containing the values to be averaged.

**numSample**

The number of array elements to be averaged.

**optNumIgnore**

Optional Integer or Real tag containing the value to be ignored in calculating the average.

#### Return value

Returns the average of the values.

#### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`Avg( 1, 2.34, 5, 7, 4, 8, 9.4 )`** // Returned value = 5.248571 |
| Tag | **`Avg( 1, 5, -9, 0, 5, 3 )`** // Returned value = 0.833333 |

| Tag Name | Expression |
|---|---|
| Tag | **`Avg( "tagArray[1]", 3 )`** // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 30 |
| Tag | **`Avg( "tagArray[1]", 3, 10 )`** // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 40 |

## *Max*

Returns the maximum value of a set of numbers.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Max | Statistical | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Max( numValue1, numValue2, … , numValueN )
Max( "tagArray", numSample, optNumIgnore )
```

> **Note:**
>
> This function has two formats:
>
> - If the first parameter is a numeric tag or value, you must use the **Max( numValue1, numValue2, … , numValueN )** format.
> - If the first parameter is an array tag in double-quotes or a string tag, you must use the **Max( "tagArray", numSample, optNumIgnore )** format.

**numValue (1...N)**

Integer or Real tags containing the numbers to be analyzed.

**tagArray**

Name of array tag (Real or Integer) containing the values to be analyzed.

**numSample**

The number of array elements to be analyzed.

**optNumIgnore**

Integer or Real tags containing the value to be ignored in the analysis.

### Return value

Returns the maximum value of the set.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **Max( 1, 2.34, 5, 7, 4, 8, 9.4 )** // Returned value = 9.4 |
| Tag | **Max( 1, 5, -9, 0, 5, 3 )** // Returned value = 5 |
| Tag | **Max( "tagArray[1]", 3 )** // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 60 |
| Tag | **Max( "tagArray[1]", 3, 10 )** // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 60 |

## *Min*

Returns the minimum value of a set of numbers.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Min | Statistical | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Min( numValue1, numValue2, … , numValueN )
```

```
Min( "tagArray", numSample, optNumIgnore )
```

> **Note:**
>
> This function has two formats:
>
> - If the first parameter is a numeric tag or value, you must use the **`Min( numValue1, numValue2, … , numValueN)`** format.
> - If the first parameter is an array tag in double-quotes or a string tag, you must use the **`Min( "tagArray", numSample, optNumIgnore )`** format.

**numValue (1...N)**

Integer or Real tags containing the numbers to be analyzed.

**tagArray**

Name of an array tag (Real or Integer) containing the values to be analyzed.

**numSample**

The number of array elements to be analyzed.

**optNumIgnore**

Integer or Real tags containing a value to be ignored in the analysis.

### Return value

Returns the minimum value of the set.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`Min( 1, 2.34, 5, 7, 4, 8, 9.4 )`** // Returned value = 1 |
| Tag | **`Min( 1, 5, -9, 0, 5, 3 )`** // Returned value = -9 |
| Tag | **`Min( "tagArray[1]", 3 )`** // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 10 |
| Tag | **`Min( "tagArray[1]", 3, 10 )`** // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 20 |

## *Rand*

Generates a random number between 0 and 1.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Rand | Statistical | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Rand()
```

This function has no parameters.

### Returned value

Returns a real number between 0 and 1.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`Rand()`** // Returned value = ?, Where: 0<?<1 |

## String functions

These functions are used to manipulate text strings or convert them into numeric values.

### *Asc2Str*

This function converts one or more Unicode character codes to a string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Asc2Str | String | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

**Asc2Str( *numChar1*, *numChar2*, … , *numCharN* )**

**numChar (1-N)**

A Unicode character code (in decimal).

#### Returned value

Returns a string comprising the converted codes.

#### Notes

Although the name of this function implies it only supports ASCII characters, it is in fact a legacy of previous versions of the software. The current version supports the full Unicode character set.

#### Examples

| Tag Name | Expression |
|---|---|
| Tag | **Asc2Str( 65 )** // Returned value = "A" |
| Tag | **Asc2Str( 65, 66, 67 )** // Returned value = "ABC" |
| Tag | **Asc2Str( Array[0], Array[1], Array[2] )** // Returned value = "ABC" |

### *CharToValue*

This function converts a string to Unicode character codes and then stores those values in an integer array.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| CharToValue | String | Synchronous | No | Supported | Supported | Supported |

#### Syntax

CharToValue("*tagString*","*tagArray*")

**tagString**

The name of the string tag, whose value will be converted.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

**tagArray**

The name of the integer array that will receive the converted values. If no array index is specfied, then the default is 0.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Return value

Returns the number of array elements used, which should be equal to the number of characters in the string.

### Examples

If StrTag = "ABC", then Array[0] = 65, Array[1] = 66, and Array[2] = 67:

```
CharToValue( "StrTag", "Array" )
```

If StrTag = "ABC", then Array[10] = 65, Array[11] = 66, and Array[12] = 67:

```
CharToValue( "StrTag", "Array[10]" )
```

## *CharToValueW*

This function converts a string to Unicode character codes, combines each two codes into a double-byte word, and then stores those values in an integer array.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| CharToValueW | String | Synchronous | No | Supported | Supported | Supported |

### Syntax

`CharToValueW("`*tagString*`","`*tagArray*`")`

**tagString**

The name of the string tag, whose value will be converted.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

**tagArray**

The name of the integer array that will receive the converted values. If no array index is specfied, the default is 0.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Return value

Returns the number of array elements used, which should be equal to *half* the number of characters in the string.

### Notes

Because of how each two character codes are combined into single value, this function only supports Unicode character codes 0 through 255. For character codes greater than 255, or when double-byte words are not needed, use the CharToValue function.

### Examples

If StrTag = "Studio", then Array[0] = 29779 ("St"), Array[1] = 25717 ("ud"), and Array[2] = 28521 ("io"):

```
CharToValue( "StrTag", "Array" )
```

If StrTag = "Studio", then Array[10] = 29779 ("St"), Array[11] = 25717 ("ud"), and Array[12] = 28521 ("io"):

```
CharToValue( "StrTag", "Array[10]" )
```

## *ClassMembersToStrVector*

Transfers values from a Class tag to an Array tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ClassMembersToStrVector | String | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
ClassMembersToStrVector( "strClassTag" , numStartPos, numNumPos,
 "strArrayTag", optBooStartPosTarget )
```

**strClassTag**

String value containing the Class tag name.

**numStartPos**

Start position (array index) of strClassTag.

**numNumPos**

Number of positions (array indexes) to be transferred from strClassTag.

**strArrayTag**

String value containing the array tag that will receive the values from strClassTag.

**optBooStartPosTarget**

Start position (array index) of strArrayTag. If omitted, the default value 1 is used.

### Returned value

| | |
|---|---|
| −6 | Array size of *strClassTag* is not big enough for *numStartPos* |
| −5 | *strClassTag* is not a Class tag |
| −4 | *strClassTag* is not found |
| −3 | *strArrayTag* is not found |
| −2 | Invalid data type of the parameters |
| −1 | Invalid number of parameters |
| 0 | Transferred successfully |

### Notes

If strClassTag has more than one member, the value of each member will be transferred to strArrayTag. Therefore, it is importatnt to make sure that the array size of strArrayTag is big enough to receive all values from strClassTag.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `ClassMembersToStrVector( "Classtag", 5, 3, "Arraytag" )` |
| Tag | `ClassMembersToStrVector( "Classtag", 5, 3, "Arraytag", 0 )` |
| Tag | `ClassMembersToStrVector ( TagName, 0, 1, ArrayName )` |

## *DecryptData*

`DecryptData` is a built-in function that decrypts a string of data using a specified key.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `DecryptData` | String | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
DecryptData(strEncryptedData,strKey)
```

`DecryptData ( strEncryptedData , strKey )`

***strEncryptedData***

The encrypted data to be decrypted, provided as a string value. It is limited to 1024 characters and must be properly encoded for the target platform, as shown in the following table:

| Target platform | Encoding for *strData* parameter |
|---|---|
| Windows | Base64 |

***strKey***

The key to be used to decrypt the encrypted data. It must be the same key that was originally used to encrypt the data.

### Return value

If this function succeeds, it returns the decrypted data as a plaintext string.

If this function fails, it returns an empty string with BAD quality.

### Notes

This function is typically used to decrypt data that was encrypted by the `EncryptData` function, but in theory it can be used to decrypt any data from any source as long as that data was encrypted using the same algorithm and then saved as an encoded string.

The `EncryptData` and `DecryptData` functions use 128-bit AES encryption. Ask a cybersecurity expert whether this type of encryption meets your needs. Never share your keys with unauthorized personnel, and be careful about where and how you store your keys.

### Examples

Decrypt the value of a tag using a specified key:

```
DecryptData(MyEncryptedData,"V4rM#ydar6T^rZn")
```

Decrypt the value of a tag using a key stored in another tag:

```
EncryptData(MyEncryptedData,MyKey)
```

## *EncryptData*

`EncryptData` is a built-in function that encrypts a string of data using a specified key.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `EncryptData` | String | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

**`EncryptData(`*`strData`*`,`*`strKey`*`)`**

`EncryptData ( strData , strKey )`

***strData***

The data to be encrypted, provided as a string value. The length of this string value is effectively limited to a number of characters, which varies by target platform. For more information, see "Notes" below.

***strKey***

The key to be used to encrypt the data. It can be either a string value up to 16 characters or a numeric value. There are no requirements on the composition of the key, but it should follow the same guidelines that you would use to create a strong password. For more information, see "Notes" below.

### Return value

If this function succeeds, it returns the encrypted data as an encoded string. For more information, see "Notes" below.

If this function fails, it returns an empty string with BAD quality.

### Notes

After you have used this function to encrypt the data, you can use the `DecryptData` function to decrypt it again.

The `EncryptData` and `DecryptData` functions use 128-bit AES encryption. Ask a cybersecurity expert whether this type of encryption meets your needs. Never share your keys with unauthorized personnel, and be careful about where and how you store your keys.

The actual limit on the length of the key is 128 bits, which is equal to 16 characters at 8 bits per character in the ANSI character set. Be careful about specifying a key that contains non-Latin characters or symbols (i.e., characters not included in the ANSI character set) because they can be up to 32 bits per character and therefore make it difficult to find the length of the key.

The size of the data to be encrypted (i.e., the length of the string value passed to the *strData* parameter) is effectively limited due to a combination of factors. First, the process of encrypting and then encoding data automatically increases the size of the data. Second, different platforms use different methods to encode data as plaintext. And third, this function returns a string value that is limited to 1024 characters, like all other string values in this software. Therefore, to get a return value that is not more than 1024 characters, the data to be encrypted must be much smaller.

With all of this in mind, we have tested this function and found the actual limits, as shown in the following table:

| Target platform | Limit on *strData* parameter | Encoding of return value |
|---|---|---|
| Windows | up to 381 characters | Base64 |

If you try to encrypt a longer string value, this function will fail.

### Examples

Encrypt a plaintext string using a specified key:

```
EncryptData("This is a secret.","V4rM#ydar6T^rZn")
```

Encrypt the value of a tag using a specified key:

```
EncryptData(MyData,"V4rM#ydar6T^rZn")
```

Encrypt the value of a tag using a key stored in another tag:

```
EncryptData(MyData,MyKey)
```

## *NCopy*

Copies a defined section of a larger string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| NCopy | String | Synchronous | No | Supported | Supported | Supported |

### Syntax

```
NCopy( strSource, numStartChar, numQtdChar )
```

**strSource**

The source string.

**numStartChar**

Integer tag containing a number corresponding to the first character being copied.

**numQtdChar**

The number of characters to be copied.

### Returned value

Returns a string that is part of the source string (as defined by the function).

### Notes

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `NCopy( "Studio version 7.0", 7, 7 )` // Returned value = "version" |
| Tag | `NCopy( "Technical Reference", 0, 9 )` // Returned value = "Technical" |

> **Note:** The first character in the string will be assigned the value 0.

## *Num*

Converts a string into a float.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Num | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Num( strValue )
```

**strValue**

The number of characters to be converted into float format.

### Returned value

Returns the number (formerly in a string format) in float format.

### Notes

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `Num( "321654.987" )` // Returned value = 321654.987 |
| Tag | `Num( "5.6589626246" )` // Returned value = 5.6589626246 |

> **Note:** The float string cannot use characters other than the numbers (0..9) and a decimal point (.), or the function returns the value 0.0.

## *Str*

Converts a number into a string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Str | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Str( numValue )
```

**numValue**

Integer or float tag containing a number to be converted to a string.

### Returned value

Returns the string, in a float format.

### Notes

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `Str( 321654.987 )` // Returned value = "321654.987" |
| Tag | `Str( 5.65896246 )` // Returned value = "5.658962" |

## *Str2Asc*

This function converts a character to its corresponding Unicode character code.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Str2Asc | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

Str2Asc(*strChar*)

**strChar**

The character to be converted.

### Returned value

Returns the Unicode character code (in decimal) for the specified character.

### Notes

Although the name of this function implies it only supports ASCII characters, it is in fact a legacy of previous versions of the software. The current version supports the full Unicode character set.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **Str2Asc( "C" )** // Returned value = 67 |
| Tag | **Str2Asc( "o" )** // Returned value = 111 |

## *StrCompare*

Compares two strings to see if they are identitical.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrCompare | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

**StrCompare(*strValue1*,*strValue2*)**

**strValue1**

A string, or a tag of String type. This is the first string in the comparison.

**strValue2**

A string, or a tag of String type. This is the second string in the comparison.

### Return value

| | |
|---|---|
| -1 | The value of ***strValue1*** is less than the value of ***strValue2***. |
| 0 | ***strValue1*** and ***strValue2*** are identical. |
| 1 | The value of ***strValue1*** is greater than the value of ***strValue2***. |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **StrCompare( "Text1", "Text2" )** // Returned value = **-1** |

| Tag Name | Expression |
|---|---|
| Tag<br>Tag1 = **"Text"** Tag2 = **"Text"** | **StrCompare( Tag1, Tag2 )** // Returned value = **0** |
| Tag Tag1 = **"Text1"** Tag2 = **"Text2"** | **StrCompare( Tag1, Tag2 )** // Returned value = **-1** |

## *StrCompareNoCase*

Compares two strings to see if they are identitical, ignoring the case of letters (i.e., the lower-case "a" is considered to have the same value as the upper-case "A").

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrCompareNoCase | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

**StrCompareNoCase(*strValue1*,*strValue2*)**

**strValue1**

A string, or a tag of String type. This is the first string in the comparison.

**strValue2**

A string, or a tag of String type. This is the second string in the comparison.

### Return value

| | |
|---|---|
| -1 | The value of ***strValue1*** is less than the value of ***strValue2***. |
| 0 | ***strValue1*** and ***strValue2*** are identical. |
| 1 | The value of ***strValue1*** is greater than the value of ***strValue2***. |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **StrCompareNoCase( "Text1", "TEXT1" )** // Returned value = **0** |
| Tag Tag1 = **"Text1"** Tag2 = **"TEXT1"** | **StrCompareNoCase( Tag1, Tag2 )** // Returned value = **0** |

## *StrFromInt*

Converts an integer into its string representation in another base number system, such as binary (base-2) or octal (base-8).

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrFromInt | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

**StrFromInt(*numValue*,*numBase*)**

**numValue**

The numeric value to be converted into a string.

**numBase**

The base number system to convert into.

### Return value

This function returns a string representation of the given integer, in the specified base number system. The returned value can be stored in any tag of String type.

### Notes

You can specify a real number instead of an integer, but only the whole part of the number will be converted. To convert the entire real number, use the `StrFromReal` function instead.

Also, if you do not need to change the base, then use the `Str` function instead.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`StrFromInt( 26, 2 )`** // Returned value = **`"11010"`** |
| Tag | **`StrFromInt( 26, 8 )`** // Returned value = **`"32"`** |

## *StrFromReal*

StrFromReal is a built-in scripting function that converts a real numerical value to a string value, in either floating-point or exponential notation.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `StrFromReal` | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

`StrFromReal(`*numValue*`,` *numPrecision*`,{` *strType* `| f | e | E})`

**numValue**

The numerical value to be converted.

**numPrecision**

The number of decimal places to be shown in the resulting string. Please note that the value will be rounded rather than truncated.

**strType**

A single-character code that specifies how the resulting string should be formatted, as described in the following table:

| Value of strType | Description |
|---|---|
| f | Formatted in floating-point notation. |
| e | Formatted in exponential notation with a lower-case "e". |
| E | Formatted in exponential notation with an upper-case "E". |

### Return value

This function returns a string representation of the given numerical value, with the specified precision and notation.

### Examples

```
StrFromReal( 263.355, 2, "f" )
```

...returns a string value of **"263.36"**.

```
StrFromReal( 263.355, 2, "e" )
```

...returns a string value of **"2.63e+002"**.

```
StrFromReal( 263.355, 2, "E" )
```

...returns a string value of **"2.63E+002"**.

## *StrFromTime*

Converts a timestamp from UTC standard notation into a formatted string, adjusted to reflect the Time Zone setting in the Control Panel of the local computer.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrFromTime | String | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
StrFromTime( numUTCTime, numType )
```

**numUTCTime**

An integer, or a tag of Integer type. A timestamp given in UTC standard notation.

**numType**

An integer, or a tag of Integer type. Specifies the format of the resulting string, as described in the following table:

| Value of numType | Description |
|---|---|
| 1 | Displays the *date* in the same format that is selected in the Control Panel on the local computer. |
| 2 | Displays the *time* in the same format that is selected in the Control Panel on the local computer. |
| 3 | Displays a standard 24-character string that shows both date and time. |
| 4 | Displays the abbreviated name of the day of the week. |
| 5 | Displays the full name of the day of the week. |

### Returned value

This function returns a string representation of the given timestamp, with the specified formatting. The returned value can be stored in any tag of String type.

If this function is called in a project screen on Mobile Access, it is executed using the system clock and date/time settings on the computer that hosts the project runtime server.

### Notes

The Coordinated Universal Time (UTC) standard counts the number of seconds elapsed since 12:00 AM GMT on January 1, 1970. Each day consists of 86,400 seconds.

### Examples

> **Note:** The examples below are for a computer set to Eastern Standard Time (or UTC −05:00).

| Tag Name | Expression |
|---|---|
| Tag | `StrFromTime( 86400, 1 )` // Returned value = `"1/1/70"` |
| Tag | `StrFromTime( 86400, 2 )` // Returned value = `"07:00:00 PM"` |
| Tag | `StrFromTime( 86400, 3 )` // Returned value = `"Thu Jan 01 19:00:00 1970"` |
| Tag | `StrFromTime( 86400, 4 )` // Returned value = `"Thu"` |
| Tag | `StrFromTime( 86400, 5 )` // Returned value = `"Thursday"` |

## *StrGetElement*

Gets a specific element from a string source.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrGetElement | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrGetElement( strSource, strDelimiter, numElementNumber )
```

**strSource**

The source string.

**strDelimiter**

Char used as delimiter between the elements.

**numElementNumber**

Number of the element which will be returned by the function. The first element has the number 1. The second element has the number 2 and so forth.

### Returned value

Returns the element (string value) retrieved from strSource.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `StrGetElement( "a\|b\|c", "\|", 2 )` // returned value = "b" |
| Tag | `StrGetElement( "a,b,c", ",", 3 )` // returned value = "c" |

## *StrLeft*

Copies the first characters of a larger string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrLeft | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrLeft( strSource, numQtdChar )
```

**strSource**

The source string.

**numQtdChar**

The number of characters to be copied.

### Returned value

Returns a string containing the left-most characters in the source string.

### Notes

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `StrLeft( "Studio version 7.0", 8 )` // Returned value = Studio v |
| Tag | `StrLeft ( "Technical Reference", 9 )` // Returned value = Technical |

## *StrLen*

Determines the length of a string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrLen | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrLen( strSource )
```

**strSource**

The string.

### Returned value

Returns an integer that is the number of characters in the string.

### Notes

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `StrLen( "Studio version 7.0" )` // Returned value = 18 |
| Tag | `StrLen( "Technical Reference" )` // Returned value = 19 |

## *StrLower*

Converts a string to all lower case characters.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrLower | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrLower( strSource )
```

**strSource**

The string to be converted.

### Returned value

Returns the string, where all the characters are in lowercase.

### Notes

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`StrLower( "Studio version 7.0" )`** // Returned value = "studio version 7.0" |
| Tag | **`StrLower( "Technical Reference" )`** // Returned value = "technical reference" |

## *StrRChr*

Isolates the final occurrence of a character sequence within a string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrRChr | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrRChr( strSource, strChrSequence )
```

**strSource**

The source string.

**strCharSequence**

The reference string.

### Returned value

Returns a string of characters following the last occurrence of a character within the source string.

### Notes

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`StrRChr( "Studio version 7.0", "i" )`** // Returned value = "ion 7.0" |
| Tag | **`StrRChr( "Technical Reference", "n" )`** // Returned value ="nce" |

## *StrRight*

Copies the last characters in a larger string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrRight | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrRight( strSource, numQtdChar )
```

**strSource**

The source string.

**numQtdChar**

The number of characters to be copied.

### Returned value

Returns a string containing the right-most characters in a source string.

### Notes

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`StrRight( "Studio version 7.0", 8 )`** // Returned value = "sion 7.0" |
| Tag | **`StrRight( "Technical Reference", 9 )`** // Returned value = "Reference" |

## *StrSetElement*

`StrSetElement` is a built-in function that sets the value of a specified element in a string of elements.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `StrSetElement` | String | Synchronous | No | Supported | Supported | Supported |

### Syntax

**`StrSetElement(`*`strSource`*`,`*`strDelimiter`*`,`*`numElementNumber`*`,`*`strValue`*`)`**

`StrSetElement (` *strSource* `,` *strDelimiter* `,` *numElementNumber* `,` *strValue* `)`

***strSource***

The source string, which is typically a series of elements or values that are separated by a character.

***strDelimiter***

The character used as separator between the elements of the source string.

***numElementNumber***

The number of the element to be set by this function. The first element is number 1, the second element is number 2, and so on.

***strValue***

The new value to be written at the specified element number. If there is an existing value, it will be overwritten by the new value.

### Return value

If this function is successfully executed, it returns the updated string.

### Examples

Start a new string of elements by concatenating the source string and the new value (returns `TEXT|abcd`):

```
StrSetElement("TEXT","|",2,"abcd")
```

Insert the new value into an empty element (returns `TEXT|abcd|efgh|`):

```
StrSetElement("TEXT|abcd||","|",3,"efgh")
```

Overwrite an existing value with the new value (returns `TEXT|abcd|5678|ijkl`):

```
StrSetElement("TEXT|abcd|efgh|ijkl","|",3,"5678")
```

## *StrStr*

Isolates the first occurrence of a character sequence within a string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrStr | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrStr( strSource, strCharSequence )
```

**strSource**

The source string.

**strCharSequence**

The reference string.

### Returned value

Returns the string of characters following the first occurrence of a character within the source string.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `StrStr( "Studio version 7.0", "i" )` // Returned value = "io version 7.0" |
| Tag | `StrStr( "Technical Reference", "n" )` // Returned value ="nical Reference" |

## *StrStrPos*

Finds the first occurrence of a character within a string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrStrPos | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrStrPos( strSource, strCharSequence )
```

**strSource**

The source string.

**strCharSequence**

The reference string.

### Returned value

Returns an integer corresponding to the first occurrence of a character within the source string.

### Notes

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `StrStrPos( "Studio version 7.0", "i" )` // Returned value = 4 |
| Tag | `StrStrPos( "Technical Reference", "a" )` // Returned value = 7 |

> **Note:** The first character in the string assigned the value 0.

## *StrTrim*

Removes unwanted spaces from a string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrTrim | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrTrim( strReference, optNumFlag )
```

**strReference**

A string, or a tag of String type that contains the source string.

**optNumFlag**

An *optional* integer or tag of Integer type:

| Value of optNumFlag | Description |
|---|---|
| 0 | Removes all spaces from both the beginning and the end of the string. |
| 1 | Removes all spaces only from the beginning of the string. |
| 2 | Removes all spaces only from the end of the string. |
| 3 | Removes all spaces except for single spaces between words. |

> **Note:** If no value is given for **optNumFlag**, then **0** is the default.

### Returned value

This function returns a string equal to **strReferance** minus the specified space characters. The returned value can be stored in any tag of String type.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `StrTrim( " Studio version 7.0 " )` // Returned value = `"Studio version 7.0"` |
| Tag | `StrTrim( " Studio version 7.0 ", 0 )` // Returned value = `"Studio version 7.0"` |
| Tag | `StrTrim( " Studio version 7.0 ", 1 )` // Returned value = `"Studio version 7.0 "` |
| Tag | `StrTrim( " Studio version 7.0 ", 2 )` // Returned value = `" Studio version 7.0"` |
| Tag | `StrTrim( " Studio  version  7.0 ", 3 )` // Returned value = `"Studio version 7.0"` |

## *StrTrimAll*

Eliminates a specific char from the whole string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrTrimAll | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrTrimAll( strReference, optStrTrimChar )
```

**strReference**

A The source string.

**optStrTrimChar**

Char that will be removed from the string. If this parameter is omitted, the space char will be removed from the string, by default.

### Returned value

Returns a string equal to strReference minus the characters removed by the function.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `StrTrimAll( "Studio version 7.0 ", " ")` // Returned value = "Studioversion7.0" |

## *StrUpper*

StrUpper is a built-in function that converts a string to all uppercase characters.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| StrUpper | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
StrUpper(strSource)
```

StrUpper(*strSource*)

***strSource***

The string to be converted.

### Return value

This function returns the specified string with all characters converted to uppercase.

### Examples

Convert the string "Technical Reference" (returns "TECHNICAL REFERENCE"):

```
StrUpper("Technical Reference")
```

Convert the string that is stored in the tag MyString:

```
StrUpper(MyString)
```

## *ValueToChar*

ValueToChar is a built-in function that converts an array of numeric values into Unicode characters and then returns those characters as a string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ValueToChar | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

**ValueToChar("*tagArray*",*numQtdChars*)**

ValueToChar("*tagArray*",*numQtdChars*)

***tagArray***

The name of the array that contains the values to be converted, as well as the index of the starting position in the array. If the starting position is not specified, the default is 0.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***numQtdChars***

The number of values to be converted (minimum of 1), starting from the specifed starting position.

### Return value

This function returns a string that comprises the converted values.

### Notes

Keep in mind that character codes in the ranges from 0 (U+0000) through 31 (U+001F) and from 128 (U +0080) through 159 (U+009F) are actually control characters rather than alphanumeric characters. In particular, 0 is a NUL character that will effectively end the string that is returned by this function.

### Examples

If MyArray[0] = 65, MyArray[1] = 66, and MyArray[2] = 67, then the returned value will be "ABC":

```
ValueToChar("MyArray",3)
```

If MyArray[10] = 65, MyArray[11] = 66, and MyArray[12] = 67, then the returned value will be "ABC":

```
ValueToChar("MyArray[10]",3)
```

## *ValueWToChar*

ValueWToChar is a built-in function that converts an array of 16-bit numeric values (a.k.a. double-byte words) into pairs of Unicode characters and then returns those characters as a string.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ValueWToChar | String | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

**ValueWToChar("*strTagArray*",*numQtdChars*,*optNumSwap*)**

`ValueWToChar("strTagArray",numQtdChars{ | ,optNumSwap})`

***strTagArray***

The name of the array that contains the values to be converted, as well as the index of the starting position in the array. If the starting position is not specified, the default is 0.

Each value in the array should be an unsigned integer in the range from 0 through 65535.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***numQtdChars***

The number of values to be converted (minimum of 1), starting from the specifed starting position.

***optNumSwap***

A numeric flag that causes the two bytes of each value to be swapped as they are converted:

| Value | Description |
|---|---|
| 0 | Do not swap |
| 1 | Swap the bytes |

This parameter is optional; if no value is specified, the default value is 0.

## Return value

This function returns a string that comprises the converted values.

## Notes

Each 16-bit numeric value is also known as a double-byte word. This function splits each word into two bytes and then converts each byte into a single Unicode character. As such, this function only supports 8-bit character codes — that is, character codes in the range from 0 (U+0000) through 255 (U+00FF). For character codes greater than 255, or when double-byte words are not needed, use the `ValueToChar` function instead.

Keep in mind that Unicode character codes in the ranges from 0 (U+0000) through 31 (U+001F) and from 128 (U+0080) through 159 (U+009F) are actually control characters rather than alphanumeric characters. In particular, 0 is a NUL character that will effectively end the string that is returned by this function.

## Examples

These examples use the following array:

| Position | Actual Value | Hexadecimal | Characters |
|---|---|---|---|
| 0 | 19833 | 4D 79 | M y |
| 1 | 21364 | 53 74 | S t |
| 2 | 29289 | 72 69 | r i |
| 3 | 28263 | 6E 67 | n g |
| 4 | 11776 | 2E 00 | . [NUL] |

Starting from position 0 and then coverting one value, this function returns "My":

```
ValueWToChar("MyArray",1)
```

Starting from position 0 and then coverting five values, this function returns "MyString.":

```
ValueWToChar("MyArray",5)
```

Starting from position 1 and then coverting three values, this function returns "String":

```
ValueWToChar("MyArray[1]",3)
```

Starting from position 1, coverting three values, and then swapping the bytes, this function returns "tSirgn":

```
ValueWToChar("MyArray[1]",3,1)
```

## System Info functions

These functions are used get information about the computer that is running the project (either server or client, depending on the function), as well as to change some project settings on that computer.

### *DBVersion*

`DBVersion` is a built-in scripting function that gets the version number of your project tags database.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `DBVersion` | System Info | Synchronous | Yes | Supported | Supported | Executed on Server |

#### Syntax

```
DBVersion()
```

`DBVersion()`

This function takes no paramters.

#### Returned value

This function returns a numerical value equal to the version number of the database.

#### Notes

This function only applies to the native database within your project. There currently is no function to get the version number of an external or historical database.

#### Examples

```
DBVersion()
```

### *GetAppHorizontalResolution*

`GetAppHorizontalResolution` is a built-in function that gets the default horizontal screen resolution (in pixels) of the project.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetAppHorizontalResolution` | System Info | Synchronous | Yes | Supported | Supported | Executed on Server |

#### Syntax

```
GetAppHorizontalResolution()
```

`GetAppHorizontalResolution()`

This function takes no parameters.

#### Returned value

This function returns the default value that is stored in the project file. It does not get the actual display size on the client.

### *GetAppPath*

Returns the file path of the project folder.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetAppPath | System Info | Synchronous | Yes | Supported | Executed on Server | Executed on Server |

#### Syntax

```
GetAppPath()
```

This function takes no paramters.

#### Returned value

Returns the file path as a string.

#### Examples

| Tag Name | Expression |
|---|---|
| Tag | **GetAppPath()** // Returned value = "C:\DemoProject\" |
| Tag | **GetAppPath()** // Returned value = "C:\Studio\Projects\<*project name*>\" |

> **Note:** This function must return the current path of the project, including the "\" at the end of the path.

### *GetAppVerticalResolution*

GetAppVerticalResolution is a built-in function that gets the default vertical screen resolution (in pixels) of the project.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetAppVerticalResolution | System Info | Synchronous | Yes | Supported | Supported | Executed on Server |

#### Syntax

**GetAppVerticalResolution()**

```
GetAppVerticalResolution()
```

This function takes no parameters.

#### Returned value

This function returns the default value that is stored in the project file. It does not get the actual display size on the client.

### *GetComputerIP*

Returns the first IP Address of the local computer.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetComputerIP | System Info | Synchronous | Yes | Supported | Supported | Not supported |

#### Syntax

```
GetComputerIP()
```

This function takes no parameters.

### Returned value

Returns the first IP Address of the local station as a string.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **GetComputerIP()** // Returned value = "192.168.0.1" |
| Tag | **GetComputerIP()** // Returned value = "248.12.2.78" |

## *GetComputerName*

Returns the local computer name.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetComputerName | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
GetComputerName()
```

This function takes no parameters.

### Returned value

Returns the local computer name as a string.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **GetComputerName()** // Returned value = "Terminal53" |
| Tag | **GetComputerName()** // Returned value = "BobsComputer" |

## *GetCursorX*

Gets the X-coordinate of the mouse cursor on the screen.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetCursorX | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
GetCursorX()
```

This function takes no parameters.

### Returned value

This function returns the X-coordinate of the cursor on the screen, or **-1** if an error occurs.

### Examples

| Tag Name | Expression |
|---|---|
| | **GetCursorX( )** // Returned value = 1024 |

## *GetCursorY*

Gets the Y-coordinate of the mouse cursor on the screen.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetCursorY | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
GetCursorY()
```

This function takes no parameters.

### Returned value

This function returns the Y-coordinate of the cursor on the screen, or **-1** if an error occurs.

### Examples

| Tag Name | Expression |
|---|---|
| | **GetCursorY( )** // Returned value = 768 |

## *GetDisplayHorizontalResolution*

Gets the horizontal resolution (in pixels) of the display connected to the local station.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetDisplayHorizontalResolution | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
GetDisplayHorizontalResolution()
```

This function takes no parameters.

### Returned value

This function returns the horizontal resolution of the display as an integer.

### Examples

| Tag Name | Expression |
|---|---|
| | **GetDisplayHorizontalResolution()** // Returned value = 1024 |

## *GetDisplayVerticalResolution*

Gets the vertical resolution (in pixels) of the display connected to the local station.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetDisplayVerticalResolution | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
GetDisplayVerticalResolution()
```

This function takes no parameters.

### Returned value

This function returns the vertical resolution of the display as an integer.

### Examples

| Tag Name | Expression |
|---|---|
| | **GetDisplayVerticalResolution( )** // Returned value = 768 |

## *GetHardKeyModel*

Returns the model name of your Hardkey.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetHardKeyModel | System Info | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
GetHardKeyModel()
```

This function takes no parameters.

### Returned value

If the Hardkey is installed, then the function returns a string with the Hardkey model name.

If the Hardkey is not installed, then the function returns `0`.

### Notes

You must attach the Hardkey before executing this function, or the function will not execute properly.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **GetHardKeyModel()** // Returned value = "Local Interface" |
| Tag | **GetHardKeyModel()** // Returned value = "Advanced Server" |

## *GetHardKeySN*

Returns the serial number of the Hardkey.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetHardKeySN | System Info | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
GetHardkeySN()
```

This function takes no parameters.

### Returned value

If the Hardkey is installed, then the function returns a string with the Hardkey serial number.

If the Hardkey is not installed, then the function returns `0`.

### Notes

You must attach the Hardkey before executing this function, or the function will not execute properly.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **GetHardkeySN()** // Returned value = 120.745 |

| Tag Name | Expression |
|---|---|
| Tag | **GetHardkeySN()** // Returned value = 224.941 |

## *GetIPAll*

Returns the number of IP Addresses assigned to the local station and stores the IP Addresses in a string array tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetIPAll | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

**GetIPAll( "*tagArrayIP*", *optRefresh* )**

**tagArrayIP**

Name of the string array tag receiving the IP addresses found. This name must be enclosed in quotes, or the project will try to get the contents of the array tag.

**optRefresh**

*Optional* tag that triggers a refresh of this function, if you use it in a Text Data Link animation. Every time this tag changes value, the project will refresh the function.

### Returned value

| | |
|---|---|
| *n* | Number of IP addresses found |
| −1 | Invalid number of parameters |
| −2 | Invalid parameter type |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **GetIPAll( "TagArrayIP" )** // Returned value = 1 |
| Tag | **GetIPAll( "TagArrayIP", Second )** // Returned value = 2 |

## *GetNetMACID*

Gets the MAC ID unique code from the currently installed network adapter(s).

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetNetMACID | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

**GetNetMACID( "*optTagMACID*", "*optTagAdapterName*" )**

**optStrMACID**

Name of a string tag, which receives the MAD ID of the network adapter. If there is more than one network adapter currently installed in the station, the user can configure a string array tag in this parameter, so each array position receives the MAC ID from one network adapter.

**optStrAdapterName**

Name of a string tag, which receives the name of the network adapter. If there is more than one network adapter currently installed in the station, the user can configure a string array tag in this parameter, so each array position receives the name from one network adapter. This parameter is optional.

### Returned value

| Value | Description |
|---|---|
| >0 | Number of network adapters found. |
| 0 | No network adapters found. |
| -1 | Invalid number of parameters. |
| -2 | One of the parameters is not string type. |
| -3 | Tag configured in optTagMACID does not exist. |
| -4 | Tag configured in optTagAdapterName does not exist. |

### Examples

| Tag Name | Expression |
|---|---|
| NumNIC | `GetNetMACID("MACIDTag")` |
| NumNIC | `GetNetMACID("MACIDTag", "AdapterName")` |
| NumNIC | `GetNetMACID("MACIDTag[1]", "AdapterName[1]")` |

## *GetOS*

`GetOS` is a built-in function that gets the operating system on the computer that hosts the project runtime.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetOS` | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

**`GetOS()`**

`GetOS()`

This function has no parameters.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 2 | Windows, Windows Server, or Windows Embedded Standard |

## *GetPerformanceMetric*

The `GetPerformanceMetric` function returns selected metrics for the graphics performance of your project runtime client.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetPerformanceMetric` | System Info | Synchronous | Yes | Supported | Supported | Not supported |

## Syntax

```
GetPerformanceMetric( numMetrictype, "optTagDescription" )
```

GetPerformanceMetric({ *numMetrictype* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 }{ | ,"*optTagDescription*" })

***numMetrictype***

The type of performance metric to get, as shown in the table below:

| Value of *numMetrictype* | Description, including the string that is written to *optTagDescription* |
|---|---|
| 0 | `Memory Allocation Enabled`<br>How the virtual memory is currently allocated to shared objects: 0 = Disabled; 1 = Always; 2 = Dynamic; 3 = Critical.<br>**Disabled**<br>The **Enable memory allocation** option (**Project Settings** > **Options** > **Performance Control**) is not selected. Objects are loaded into memory only when they are actually used, and then they are immediately removed afterwards.<br>**Always**<br>Objects are always kept in memory, in order to increase run-time performance.<br>**Dynamic**<br>Objects are kept in memory until the memory is needed for new objects. If enough memory is freed, then the memory allocation state may return from Dynamic to Always.<br>**Critical**<br>All unused objects are cleared from memory and memory allocation is disabled until the project is restarted. This is effectively the same as Disabled (see above), except that the **Enable memory allocation** option is still selected. |
| 1 | `Critical Free Memory Limit (KB)`<br>When the amount of free virtual memory (in KB) decreases to this limit, the memory allocation state is changed from Dynamic to Critical.<br>This limit is set in **Project Settings** > **Options** > **Performance Control**. |
| 2 | `Min Free Memory Limit (KB)`<br>When the amount of free virtual memory (in KB) decreases to this limit, the memory allocation state is changed from Always to Dynamic.<br>This limit is set in **Project Settings** > **Options** > **Performance Control**. |
| 3 | `Free Memory (KB)`<br>The amount of free virtual memory, in kilobytes. |
| 4 | `Free Memory (%)`<br>The amount of free virtual memory, as a percentage of total virtual memory. |
| 5 | `Total Allocation Allowed`<br>The maximum number of total shared objects that can be allocated. |
| 6 | `Total Allocation Count`<br>The number of total shared objects that are currently allocated. |
| 7 | `Total Allocation (%)`<br>The percentage ( **`count / allowed`** ) of total shared objects that are allocated. |

| Value of *numMetrictype* | Description, including the string that is written to *optTagDescription* |
|---|---|
| 8 | `Brush Allocation Allowed`<br>The maximum number of shared brush objects that can be allocated. |
| 9 | `Brush Allocation Count`<br>The number of shared brush objects that are currently allocated. |
| 10 | `Brush Allocation (%)`<br>The percentage ( **`count / allowed`** ) of shared brush objects that are allocated. |
| 11 | `Brush Allocation State`<br>How the memory for shared brush objects is currently allocated: `0` = Disabled; `1` = Always; `2` = Dynamic; `3` = Critical. |
| 12 | `Font Allocation Allowed`<br>The maximum number of shared font objects that can be allocated. |
| 13 | `Font Allocation Count`<br>The number of shared font objects that are currently allocated. |
| 14 | `Font Allocation (%)`<br>The percentage ( **`count / allowed`** ) of shared font objects that are allocated. |
| 15 | `Font Allocation State`<br>How the memory for shared font objects is currently allocated: `0` = Disabled; `1` = Always; `2` = Dynamic; `3` = Critical. |
| 16 | `Pen Allocation Allowed`<br>The maximum number of shared pen objects that can be allocated. |
| 17 | `Pen Allocation Count`<br>The number of shared pen objects that are currently allocated. |
| 18 | `Pen Allocation (%)`<br>The percentage ( **`count / allowed`** ) of shared pen objects that are allocated. |
| 19 | `Pen Allocation State`<br>How the memory for shared pen objects is currently allocated: `0` = Disabled; `1` = Always; `2` = Dynamic; `3` = Critical. |
| 20 | `Bitmap Allocation (%)`<br>The percentage ( **`count / allowed`** ) of shared bitmap objects that are allocated. |
| 21 | `Bitmap Allocation Count`<br>The number of shared bitmap objects that are currently allocated. |
| 22 | `Bitmap Allocation Allowed`<br>The maximum number of shared bitmap objects that can be allocated. |
| 23 | `Bitmap Allocation State`<br>How the memory for shared bitmap objects is currently allocated: `0` = Disabled; `1` = Always; `2` = Dynamic; `3` = Critical. |
| 24 | `Image Allocation Allowed`<br>The maximum number of shared image objects that can be allocated. |
| 25 | `Image Allocation Count`<br>The number of shared image objects that are currently allocated. |

| Value of *numMetrictype* | Description, including the string that is written to *optTagDescription* |
|---|---|
| 26 | `Image Allocation (%)`<br>The percentage ( **`count / allowed`** ) of shared image objects that are allocated. |
| 27 | `Image Allocation State`<br>How the memory for shared image objects is currently allocated: `0` = Disabled; `1` = Always; `2` = Dynamic; `3` = Critical. |
| 28 | `Buffer Allocation Allowed`<br>The maximum number of shared buffer objects that can be allocated. |
| 29 | `Buffer Allocation Count`<br>The number of shared buffer objects that are currently allocated. |
| 30 | `Buffer Allocation (%)`<br>The percentage ( **`count / allowed`** ) of shared buffer objects that are allocated. |
| 31 | `Buffer Allocation State`<br>How the memory for shared buffer objects is currently allocated: `0` = Disabled; `1` = Always; `2` = Dynamic; `3` = Critical. |

***optTagDescription***

The name of a String tag that will receive a description of the selected metric. The description is the same as shown in the table for *numMetrictype* above.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

This parameter is optional.

### Returned value

If this function is successfully executed, then it returns a positive value according to the type of metric that was selected (see *numMetrictype* above). Otherwise, if it was not successfully executed, then it returns a negative error value:

| Returned value | Description |
|---|---|
| `-1` | Invalid number of parameters. |
| `-2` | Wrong parameter type. |
| `-3` | Could not get the selected metric. |
| `-4` | Tag specified by *optTagDescription* does not exist. |
| `-5` | Could not write to tag specified by *optTagDescription*. |

### Notes

For more information about memory allocation, see Configure the performance control settings on page 112.

## *GetPrivateProfileString*

Reads a specified parameter from an `.ini` file using the standard `.ini` format.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetPrivateProfileString` | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
GetPrivateProfileString( strSection, strName, strDefault, strFileName )
```

**strSection**

The section name to be read.

**strName**

The parameter name to be read.

**strDefault**

The default setting for this parameter. If the parameter is not found in the **.ini** file, the function will return this default setting.

**strFileName**

The path and name of the **.ini** file to be read.

### Returned value

Returns the value of the specified parameter.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **GetPrivateProfileString( "boot loader", "timeout", "50", "C:\boot.ini" )** // Returned value = 30 |

## *GetProductPath*

GetProductPath is a built-in function that gets the path to the program directory.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetProductPath | System Info | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

```
GetProductPath()
```

GetProductPath()

This function takes no parameters.

### Returned value

Returns the path to the program directory as a string.

## *GetRegValue*

Gets a the value of a variable in the Windows registry.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetRegValue | System Info | Synchronous | Yes | Supported | Not supported | Not supported |

### Syntax

```
GetRegValue( numMainKey, strKey, strValueName )
```

**numMainKey**

Numeric tag with the following possible values:

| | |
|---|---|
| 0 | HKEY_LOCAL_MACHINE |
| 1 | HKEY_CLASSES_ROOT |
| 2 | HKEY_CURRENT_USER |
| 3 | HKEY_USERS |
| 4 | HKEY_CURRENT_CONFIG |
| 5 | HKEY_PERFORMANCE_DATA |

**strKey**

Path where the value is located in the Main Key.

**strVariableName**

Name of the variable to get. The maximum length is 255 characters.

### Returned value

If the function succeeds, then the function returns the variable value. Otherwise one of the following error codes will be returned:

| | |
|---|---|
| -1 | Invalid number of parameters or invalid Main Key. |
| -2 | Variable type is not supported. You can only read DWord or String values from the registry. |
| -3 | Failed to read the variable value; verify that you have the proper security rights. |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`GetRegValue( 0, "HARDWARE\DESCRIPTION\System", "SystemBiosDate" )`** // Returned value = "08/14/03" |
| Tag | **`GetRegValue( 2, "Control Panel\Current", "Color Schemes" )`** // Returned value = "Windows Standard " |

## *GetRegValueType*

Gets the data type of the value of a variable in the Windows registry.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetRegValueType | System Info | Synchronous | Yes | Supported | Not supported | Not supported |

### Syntax

**`GetRegValueType( `*`numMainKey`*`, `*`strKey`*`, `*`strValueName`*` )`**

**numMainKey**

Numeric tag with the following possible values:

| | |
|---|---|
| 0 | HKEY_LOCAL_MACHINE |
| 1 | HKEY_CLASSES_ROOT |
| 2 | HKEY_CURRENT_USER |
| 3 | HKEY_USERS |

| | |
|---|---|
| 4 | HKEY_CURRENT_CONFIG |
| 5 | HKEY_PERFORMANCE_DATA |

**strKey**

Path where the value is located in the Main Key.

**strVariableName**

Name of the variable to get. The maximum length is 255 characters.

### Returned value

| | |
|---|---|
| 1 | Variable type is String. |
| 0 | Variable type is DWord. |
| −1 | Invalid number of parameters or invalid Main Key. |
| −2 | Variable type is not supported. You can only read DWord or String values from the registry. |
| −3 | Failed to read the variable value; verify that you have the proper security rights. |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`GetRegValueType( 0, "HARDWARE\DESCRIPTION\System", "SystemBiosDate" )`** // Returned value = 1 |
| Tag | **`GetRegValueType( 2, "Control Panel\Desktop", "Smooth Scroll" )`** // Returned value = 0 |

## *GetServerHostName*

`GetServerHostName` is a built-in function that gets the host name of the project's Server station.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetServerHostName` | System Info | Synchronous | Yes | Not supported | Supported | Executed on Server |

### Syntax

```
GetServerHostName()
```

`GetServerHostName()`

This function takes no parameters.

### Returned value

Server host name for ISSymbol and 127.0.0.1 for others.

## *GetTickCount*

Gets the current value of the clock ticks counter.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetTickCount` | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

`GetTickCount()`

This function takes no parameters.

### Returned value

Returns an integer with the number of milliseconds counted by the clock for each initialization of the operational system.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **GetTickCount()** // Returned value = 9400907 |

## *InfoAppAlrDir*

`InfoAppAlrDir` is a built-in function that returns the file path of the project's Alarm sub-folder.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `InfoAppAlrDir` | System Info | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

**InfoAppAlrDir()**

`InfoAppAlrDir()`

This function takes no parameters.

### Returned value

Returns the Alarm directory of the current project as a string.

## *InfoAppHstDir*

`InfoAppHstDir` is a built-in function that returns the file path of the project's History sub-folder.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `InfoAppHstDir` | System Info | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

**InfoAppHstDir()**

`InfoAppHstDir()`

This function takes no parameters.

### Returned value

Returns the History directory for the current project as a string.

## *InfoDiskFree*

Returns free disk space on the local computer.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `InfoDiskFree` | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

`InfoDiskFree(`*strDisk*`)`

**strDisk**

The name of the disk volume to be checked.

### Returned value

Returns disposable free space in the disk in bytes.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`InfoDiskFree( "C" )`** // Returned value = 2803804605.000000 |

## *InfoResources*

Returns the local computer's disposable resources.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `InfoResources` | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

`InfoResources(`*numSelect*`)`

**numSelect**

A numeric flag that specifies which resource to examine:

| Value | Description |
|---|---|
| 0 | System functions (%) |
| 1 | GDI functions (%) |
| 2 | USER functions (%) |
| 3 | Memory (in bytes) |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`InfoResources( 0 )`** // Returned value = 76.000000 |
| Tag | **`InfoResources( 1 )`** // Returned value = 76.000000 |
| Tag | **`InfoResources( 2 )`** // Returned value = 80.000000 |
| Tag | **`InfoResources( 3 )`** // Returned value = 16150528.000000 |

> **Note:** The only valid selection on an Windows PC station is **`3`**. Selecting **`0`**, **`1`** or **`2`** returns **`0.000000`** only.

## *IsActiveXReg*

Determines whether an ActiveX control is registered with the operating system.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `IsActiveXReg` | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
IsActiveXReg( numType, strProgIDorFileName )
```

**numType**

A numeric flag that specifies a format for the strProgIDorFileName parameter:

| | |
|---|---|
| 0 | Verify by Program ID |
| 1 | Verify by File Name |

**strProgIDorFileName**

The program ID or file path of the ActiveX control.

### Returned value

| | |
|---|---|
| 0 | ActiveX is not registered. |
| 1 | ActiveX is registered. |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `IsActiveXReg( 0, "ISSYMBOL.ISSymbolCtrl.1" )` // Returned value = 0 |
| Tag | `IsActiveXReg( 1, "C:\WinNT\system32\MediaPlayer.ocx" )` // Returned value = 1 |

## *IsAppChangedOnServer*

The function `IsAppChangedOnServer` checks whether the project files available on the server are newer than the files currently on the client.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `IsAppChangedOnServer` | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
IsAppChangedOnServer("optTagUpdateTrigger")
```

`IsAppChangedOnServer({ "`*optTagUpdateTrigger*`" })`

***optTagUpdateTrigger***

The name of a project tag that will server as a trigger. When the value of the specified tag changes, the function is executed. Unlike most other functions, once this function is called, it is kept in memory until the project is stopped.

To execute the function at regular intervals, use one of the system tags like **`Day`** or **`Month`**.

This parameter is optional; if no value is specified, the function is executed immediately and not kept in memory.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | FALSE (i.e., the project files on the server have not changed) |
| 1 | TRUE (i.e., the project files on the server have changed) |

If this function returns TRUE, you can use the function `ReloadAppFromServer` to update the client.

### Notes

For this function, the server is the computer or device that hosts the downloadable project files (e.g., screens) for your thin clients. It might be different from the project runtime server (a.k.a. the data server), depending on how you deploy your project. For more information, see Thin Clients and Mobile Access on page 727.

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

### Examples

Immediately check whether the project files have changed:

```
IsAppChangedOnServer()
```

When the value of the tag **CheckVersion** changes, check whether the project files have changed:

```
IsAppChangedOnServer("CheckVersion")
```

## *NoInputTime*

`NoInputTime` is a built-in function that returns the time elapsed since the last input from keyboard or mouse.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `NoInputTime` | System Info | Synchronous | Yes | Supported | Keyboard input only | Not supported |

### Syntax

```
NoInputTime("optTagUpdateTrigger")
```

`NoInputTime({ "`*optTagUpdateTrigger*`" })`

***optTagUpdateTrigger***

The name of a project tag. When the value of the tag changes, it triggers an update when this function is used in a Text Data Link animation.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Return value

This function returns the returns the time elapsed (in seconds) since the last input from keyboard or mouse.

### Notes

This function will not behave as expected if VBScript debugging is enabled, because the normal execution cycle is suspended during debugging and it is not possible to accurately measure the time elapsed without user input. For more information, see Debugging VBScript on page 1246.

This function cannot be implemented directly from a Text object.

## *ProductVersion*

`ProductVersion` is a built-in function that returns the version number of the EcoStruxure Machine SCADA Expert 2020 software.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `ProductVersion` | System Info | Synchronous | Yes | Supported | Supported | Executed on Server |

### Syntax

**`ProductVersion()`**

`ProductVersion()`

This function takes no paramters.

### Returned value

This function returns the program version number as a real number.

## *ReloadAppFromServer*

The function `ReloadAppFromServer` reloads the necessary project files from the server while maintaining the current state of the project on the client.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `ReloadAppFromServer` | System Info | Synchronous | No | Supported | Supported | Not supported |

### Syntax

**`ReloadAppFromServer()`**

`ReloadAppFromServer()`

This function takes no paramters.

### Returned value

This function always returns `0`.

### Notes

For this function, the server is the computer or device that hosts the downloadable project files (e.g., screens) for your thin clients. It might be different from the project runtime server (a.k.a. the data server), depending on how you deploy your project. For more information, see Thin Clients and Mobile Access on page 727.

If the **Enforce Web functionality equivalence in local project screens** option is selected in the project settings, this function cannot be called in Global Procedures, Script worksheets, or other background tasks. This is because the function behaves differently depending on whether you view project screens locally (i.e., on the same computer that hosts the project runtime) or remotely. For more information, see Preferences tab on page 129.

> **Tip:** Before you call this function, you can use the function `IsAppChangedOnServer` to check the project files that are already on the client. If they match the files on the server, you might choose not to call this function.

## *SaveAlarmFile*

Use this function to enable/disable the saving feature for alarm history and to set the path where the alarm history files must be handled.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SaveAlarmFile | System Info | Synchronous | No | Supported | Not supported | Not supported |

### Syntax

```
SaveAlarmFile( numType, optRemotePath )
```

**numType**

Tag containing the number and operation, as follows:

| | |
|---|---|
| 0 | Disable save the alarm file to the local disk |
| 1 | Enable save the alarm file to local disk |
| 2 | Enable save the alarm file to local disk and to the remote path specified in the **OptRemotePath** parameter |

**optRemotePath**

Tag containing the name of the remote computer where the alarm file will be saved simultaneously to the local computer and to the remote path when **numType** equals 2.

### Returned value

| | |
|---|---|
| 0 | Success |
| 1 | 2nd parameter is not a string |
| 2 | 2nd parameter is missing |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **SaveAlarmFile( 0 )** |
| Tag | **SaveAlarmFile( 1 )** |
| Rag | **SaveAlarmFile( 2, "Z:\Apps\AppDemo" )** |

## *SetAppAlarmPath*

Sets the *Alarm* path for the current project.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetAppAlarmPath | System Info | Synchronous | No | Supported | Executed on Server | Executed on Server |

### Syntax

SetAppAlarmPath(*strPath*)

**strPath**

The new *Alarm* path for the current project.

### Return value

This function returns no value.

### Examples

| Tag Name | Expression |
|---|---|
| | **SetAppAlarmPath( "C:\Studio\Alarm\" )** |

## *SetAppHstPath*

Sets the file path (directory) where Trend history files will be saved, in the proprietary format ( **.HST** ).

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetAppHstPath | System Info | Synchronous | No | Supported | Executed on Server | Executed on Server |

### Syntax

SetAppHstPath(*strPath*)

**strPath**

The file path (directory) where Trend history files will be saved.

### Return value

This function returns no value.

### Notes

This function is useful when you intend to change the file path during runtime. You can also set the file path to a network drive by mapping it on the local station, or by using the following syntax:

***Network Drive**File Path***

Please note that this function does **not** copy existing history files from the default directory to a new one; it only sets the file path for new history files saved after the function is called.

### Examples

| Tag Name | Expression |
|---|---|
| | **SetAppHstPath("C:\Studio\History\")** |

## *SetDateFormat*

SetDateFormat is a built-in function that sets the separator and format for displaying date strings.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetDateFormat | System Info | Synchronous | No | Supported | Supported | Not supported |

### Syntax

**SetDateFormat(*strSeparator*,*strFomat*)**

SetDateFormat(*strSeparator*, *strFormat*)

***strSeparator***

The separator character for the date string. You can specify any single character, but the most common characters are "/" and "-".

***strFormat***

A string that specifies the order of the Month (M), Day (D), and Year (Y) in the date string. You can specify any combination in any order, but the most common examples are shown in the following table:

| Value | Description |
|---|---|
| DMY | Day, Month, Year (e.g., DD/MM/YYYY) |
| MDY | Month, Day, Year (e.g., MM/DD/YYYY) |
| YMD | Year, Month, Day (e.g., YYYY/MM/DD) |

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Invalid parameter(s). |
| 1 | Valid parameters. |

The function only checks whether the parameters are valid, before it tries to use those parameters to set the date format. The function does not return any value to indicate whether the date format is successfully set.

### Notes

This function sets the date format only for the station (i.e., the project runtime server or project thin client) on which the function is executed. Each station can have its own date format. Also, this function sets the date format only for the EcoStruxure Machine SCADA Expert 2020 software itself; it does not change the date format used by the operating system. For more information, see About the date format and how to change it on page 694.

### Examples

Set the date format so that January 23, 2015, is displayed as "01/23/2015":

```
SetDateFormat("/","MDY")
```

Set the date format so that January 23, 2015, is displayed as "2015-01-23":

```
SetDateFormat("-","YMD")
```

## *SetKeyboardLanguage*

`SetKeyboardLanguage` is a built-in function that sets the language of any Virtual Keyboards displayed in the Viewer.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetKeyboardLanguage | System Info | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
SetKeyboardLanguage(strLanguage)
```

SetKeyboardLanguage(*strLanguage*)

***strLanguage***

The two-letter code for the language to be set. The currently available options include:

| Code | Language |
|---|---|
| EN | English (default) |
| GE | German |
| FR | French |
| CH | Chinese |
| JA | Japanese |

You can also specify other codes for additional languages. See "Notes" below.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Success |
| 1 | Error |

### Notes

This function is supported only on Windows-based client stations that use the Viewer program to display project screens. That includes:

- The local Viewer that is part of the project runtime for Windows
- The standalone Secure Viewer program for Windows

It does not include any version of Mobile Access, which uses a different technology to display project screens in web browsers.

When this function is executed, it causes the Viewer program to load an initialization file that is located in the program folder. Each language code that you can specify for the *strLanguage* parameter should have a corresponding VK initialization file (`VK<language code>.ini`), and that file contains the key definitions for the language. For example, `VKEN.ini` contains the key definitions for English, `VKFR.ini` contains the key definitions for French, and so on.

You can modify the existing VK initialization files in order to customize the key definitions, and you can also create new files for additional languages as long as they have their own unique codes (e.g., `VKES.ini` for Spanish). For more information about how to create or modify the files, contact your EcoStruxure Machine SCADA Expert 2020 software distributor.

Please note that if you create or modify any VK initialization files, you must distribute the new files to all client stations and make sure they are copied to the correct location in the program folder. (In other words, they must be in the same folder that contains `Viewer.exe`.) The changes cannot be automatically disseminated by the project runtime.

### Examples

Set the Virtual Keyboard language to French:

```
SetKeyboardLanguage("FR")
```

Set the Virtual Keyboard language according to the current value of **tagLanguage**:

```
SetKeyboardLanguage(tagLanguage)
```

## *SetRegValue*

Sets the value of a variable in the Windows registry.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetRegValue | System Info | Synchronous | No | Supported | Not supported | Not supported |

### Syntax

```
SetRegValue( numMainKey, strKey, strVariableName, numType, strOrNumValue )
```

**numMainKey**

Numeric tag with the following possible values:

| | |
|---|---|
| 0 | HKEY_LOCAL_MACHINE |
| 1 | HKEY_CLASSES_ROOT |
| 2 | HKEY_CURRENT_USER |
| 3 | HKEY_USERS |
| 4 | HKEY_CURRENT_CONFIG |
| 5 | HKEY_PERFORMANCE_DATA |

**strKey**

Path where the value is located in the Main Key.

**strVariableName**

Name of the variable to be set. The maximum length is 255 characters.

**numType**

Two types are currently supported:

| | |
|---|---|
| 0 | DWord |
| 1 | String |

**strOrNumValue**

Variable value.

### Returned value

| | |
|---|---|
| 0 | Success. |
| −1 | Invalid number of parameters or invalid Main Key. |
| −2 | Invalid type. |
| −3 | Failed to read the variable value; verify that you have the proper security rights. |

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `SetRegValue( 0, "HARDWARE\DEVICEMAP\SERIALCOMM", "\Device\Serial1", 1, "COM3" )` // Returned value = 0 if successful |
| Tag | `SetRegValue( 2, "Control Panel\Desktop", "Smooth Scroll", 0, 1 )` // Returned value = 0 if successful |

> **Note:** This register can affect the Windows system configuration. You should be extremely careful and edit the registry *only* when you are certain about the configuration.

## *SNMPGet*

Gets information from computers or network devices through the SNMP protocol.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SNMPGet | System Info | Synchronous | Yes | Supported | Supported | Not supported |

### Syntax

```
SNMPGet( strAddress, strCommunity, strOID, "strTagName" )
```

**strAddress**

The address of the machine/computer (e.g., **"127.0.0.1"** or **"localhost"** ).

**strCommunity**

SNMP community name when communicating with the computer (e.g., **"public"** ).

**strOID**

OID to be consulted (e.g., **".1.3.6.1.2.1.1.1.0"** ).

**strTagName**

Name of the tag that will receive the requested value.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Returned value

| Value | Description |
|---|---|
| 0 | No error |
| -1 | Invalid number of parameters |
| -2 | Invalid parameter |
| -5 | GET operation failed |
| -7 | Invalid tag name |
| -8 | Invalid tag type |
| -9 | This function is not supported on the current operating system |

If you receive any other values, please contact technical support.

### Examples

| Tag Name | Expression |
|---|---|
| ErrorTag | **SNMPGet( "127.0.0.1", "public", ".1.3.6.1.2.1.1.1.0", "SysDescrTag" )**<br>//ErrorTag will receive the error code. If the function succeeds, the value in the OID ".1.3.6.1.2.1.1.1.0" will be saved in the tag SysDescrTag. |

## *SNMPSet*

Uses the Simple Network Management Protocol (SNMP) to set a value on a target computer of network device.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SNMPSet | System Info | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
SNMPSet( strAddress, strCommunity, strOID, Value, optNumType )
```

**strAddress**

The address of the target computer or device (e.g., **"127.0.0.1"** or **"localhost"** ).

**strCommunity**

The SNMP community name (e.g., **"public"** ) when communicating with the target computer or device.

**strOID**

The Object ID (OID) to be set.

**Value**

The value to be set to the specified OID.

**optNumType**

A numeric value, or a tag of Integer type, specifying the data type of ***Value***. This is an optional parameter, but if it is included, then it must have one of following values:

| Value | Type | Description |
|---|---|---|
| 0 | OCTETSTRING | An ctet string variable |
| 1 | INTEGER32 | A 32-bit signed integer variable |
| 2 | TIMETICKS | A timeticks variable |
| 3 | GAUGE32 | A gauge variable |
| 4 | COUNTER32 | A counter variable |
| 5 | IPADDRESS | An IP address variable |
| 6 | OBJECTIDENTIFIER | An object identifier variable |
| 7 | SEQUENCE | An ASN sequence variable |
| 8 | OPAQUE | An opaque variable |

### Returned value

| Value | Description |
|---|---|
| 0 | No error |
| -1 | Invalid number of parameters |
| -2 | Invalid parameter |
| -5 | SET operation failed |
| -7 | Invalid tag name |
| -8 | Invalid tag type |
| -9 | This function is not supported on the current operating system |

If you receive any other values, please contact technical support.

### Examples

| Tag Name | Expression |
|---|---|
| | **SNMPSet( "127.0.0.1", "public", ".1.3.6.1.2.1.1.1.0", 123, 1 )** //Sets an integer value of 123 to the specified OID on the localhost (127.0.0.1). |

## *WritePrivateProfileString*

WritePrivateProfileString is a built-in function that writes a specified setting to the Viewer module's initialization file (`Viewer.ini`), using the standard properties format.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `WritePrivateProfileString` | System Info | Synchronous | No | Supported | Supported | Executed on Server |

### Syntax

**`WritePrivateProfileString(`*`strSection`*`,`*`strName`*`,`*`strValue`*`,`*`strFileName`*`)`**

`WritePrivateProfileString(`*strSection*, *strName*, *strValue*, *strFileName*`)`

***strSection***

The section name to be written.

***strName***

The parameter name to be written.

***strValue***

The value to be written.

***strFileName***

The path and name of the .ini file to be written.

### Returned value

The function returns 1 if the file was updated successfully.

### Notes

When running on Windows Embedded this function will rewrite the entire file, therefore its use is not recommended for lengthy files on Windows Embedded devices. The function will also add the following lines at the end of the file when on a Windows Embedded device:

```
[FileBackUpControl]
Valid=1
```

### Examples

```
WritePrivateProfileString(Section,Name,Value,FileName)
```

```
WritePrivateProfileString("Options","ds1","Value","C:\Viewer.ini")
```

## Tags Database functions

These functions are used to directly change the values of project tags.

### *ExecuteAlarmAck*

This function acknowledges an active alarm on the specified tag. The advantage of using this function is that if used from the Thin Client, the Alarm task will store the user name and station from which the alarm was acknowledged.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ExecuteAlarmAck | Tags Database | Synchronous | No | Supported | Supported | Executed on Server |

#### Syntax

**ExecuteAlarmAck("*strTagName*",*optStrComment*,*optStrAlarmType*)**

ExecuteAlarmAck("*strTagName*",{ | ,*optStrComment*{ | ,*optStrAlarmType*}})

***strTagName***

Name of the tag on which the alarm will be acknowledged.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***optStrComment***

An optional comment to send to the Alarm task, along with the user name and station.

***optStrAlarmType***

If more than one alarm is active on the specified tag, you can specify which alarm (e.g., Hi, Lo, HiHi, LoLo) to acknowledge. Otherwise, the function acknowledges the most recently activated alarm.

#### Returned value

| Value | Description |
|---|---|
| 0 | Successfully executed. |
| -1 | Invalid number of parameters. |
| -2 | Invalid tag name. |
| -3 | Executed, but did not wait for confirmation from Alarms task. See note. |

#### Notes

When this function is used to acknowledge an alarm, it typically waits for confirmation from the Alarms task before returning a value of 0 to indicate successful execution. In some cases, however, waiting for confirmation might cause the project runtime to hang. When that happens, if the function is properly formed with valid parameters, then it will execute as intended but it will not wait for confirmation.

#### Examples

Acknowledge the active Hi alarm on tag **A**, with the comment Hi alarm on tag A:

```
ExecuteAlarmAck ("A","Hi alarm on tag A","Hi")
```

## *ForceTagChange*

Forces the database to write a value to a tag and trigger events as if the tag changed, even if the new value is equal to the old value.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ForceTagChange | Tags Database | Synchronous | No | Supported | Supported | Not supported |

### Syntax

```
ForceTagChange( "strTagName", numValue )
```

**strTagName**

The name of the tag being forced to accept the new value.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

**numValue**

The new value to be written to the specified tag.

### Return value

This function returns no value.

### Examples

| Tag Name | Expression |
|---|---|
| n/a | `ForceTagChange( "TagA", 5 )` |

## *GetAlarmCount*

This function gets the number of active alarms.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetAlarmCount | Tags Database | Synchronous | Yes | Supported | Not supported | Not supported |

### Syntax

```
GetAlarmCount()
```

`GetAlarmCount()`

This function has no parameters.

### Returned value

This function returns the number of active alarms.

Please note that the function will return a valid value only when it is executed on the project server. If it is called by a Graphics Script, Screen Script, or Command animation on a project client, it will return `-1`.

## *GetAlarmInfo*

This function gets information about an active alarm.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| GetAlarmInfo | Tags Database | Synchronous | Yes | Supported | Not supported (see "Notes" below) | Not supported (see "Notes" below) |

### Syntax

**GetAlarmInfo(*numIndex*,*numInfo*)**

GetAlarmInfo(*numIndex*,{ *numInfo* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19})

**numIndex**

The index number of the alarm about which you want to get information.

The index number must be between 0 and the total number of alarms minus 1. To get the total number of alarms, use the function GetAlarmCount. For example, if GetAlarmCount returns 6, then numIndex must be between 0 and 5.

The list of alarms is sorted by the times when the alarms first became active, so that the oldest alarm is index 0. When an alarm is both normalized and acknowledged, it is removed from the list and the remaining alarms are shifted accordingly. For example, when the alarm at index 3 is removed, the alarm at index 4 is shifted to index 3, the alarm at index 5 is shifted to index 4, and so on.

> **Note:** It is possible for an alarm to be normalized (i.e., become inactive) and then become active again without being acknowledged. That alarm will remain on the list of alarms throughout, and its Activation Time — that is, time when the alarm first became active — will not be updated.

**numInfo**

The information or alarm property that you want to get:

| Value | Alarm Property |
|---|---|
| 0 | Alarm Group |
| 1 | Tag Name |
| 2 | Alarm Message |
| 3 | Alarm Type |
| 4 | Tag value when the alarm became active |
| 5 | Activation Time |
| 6 | Norm Time |
| 7 | Ack Time |
| 8 | Alarm Priority |
| 9 | Alarm Selection |
| 10 to 19 | Custom Field 1 to Custom Field 10<br>**Tip:** You can create custom fields in the Alarms worksheet. |

### Returned value

This function returns the current value of the specified property (numInfo) of the specified alarm (numIndex). For the property Alarm Type (i.e., numInfo is 3), this function returns one of the following possible values:

| Value | Alarm Type |
|---|---|
| 1 | HiHi |
| 2 | Hi |
| 4 | Lo |
| 8 | LoLo |
| 16 | Rate of change |
| 32 | Deviation+ |
| 64 | Deviation- |

For all other properties, this function returns the actual value of the property. As such, the project tag or VBScript variable that you configure to receive the returned value should be of the appropriate data type. For example, you may configure an Integer tag to receive the value of Alarm Group or Alarm Type, but you should configure a String tag to receive the value of Tag Name or Ack Time. It is safest to always configure a String tag to receive the value, because a String tag can hold any value as a string, but that might make it more difficult to process the value after it is received. Please keep this in mind as you design and develop your project.

If the specified alarm has not yet been normalized or acknowledged, this function returns no values at all for the properties Norm Time and Ack Time, respectively.

If the function is not executed successfully, it returns one of the following error codes:

| Value | Description |
|---|---|
| -1 | Function was called by a Graphics Script, Screen Script, or Command animation on a project client. (See "Notes" below.) |
| -2 | Invalid alarm index (numIndex). |
| -3 | Invalid alarm property (numInfo). |

### Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a thin client. For more information, see Using TagsDB functions to edit the tags database during run time on page 169.

Also, this function is similar to the function TagsDBGetAlarm, except that it gets information about a specified alarm in the list of active alarms rather than about an alarm on a specified tag.

### Examples

Get the type (e.g., 2, meaning Hi) of the first alarm:

```
GetAlarmInfo(0,3)
```

Get the name of the project tag (e.g., MyTag) that the second alarm is on:

```
GetAlarmInfo(1,1)
```

Get the date and time (e.g., 12/05/2013 14:11:29) when the newest alarm became active, and then use the function DateTime2UTC to convert that from the current time zone to Coordinated Universal Time (UTC):

```
DateTime2UTC(GetAlarmInfo(GetAlarmCount()-1,5))
```

## *GetTagValue*

`GetTagValue` is a built-in function that gets the current value of a project tag.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `GetTagValue` | Tags Database | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
GetTagValue("strTagName",optNumRefresh)
```

`GetTagValue("`*strTagName*`"{ | ,`*optNumRefresh*`})`

***strTagName***

The name of the project tag that you want to get the value of.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

***optNumRefresh***

The name of a project tag that will trigger refreshes after the initial execution of this function. In other words, whenever the value of the specified tag changes, this function will be executed again. (Normally, a function is executed only when it is called, such as when it is configured on a Button object and the object is clicked or tapped during run time.) To execute this function at regular intervals, specify a system tag such as **`Second`**, **`Minute`**, or **`Hour`**.

### Returned value

This function returns the current value of the project tag specified by *strTagName*. (The value of the project tag specified by *optNumRefresh* does not affect the returned value in any way.) If the specified tag does not exist, this function returns `-1` with BAD quality.

Please note that if this function does return `-1`, it might indicate either that the tag does not exist or that the current value of the specified tag actually is -1. You will need to take extra steps to determine which it is, and you can do so using one of the following methods.

First, you can specify another project tag to receive the value returned by this function, and then you can check if the quality of that tag is BAD (e.g., **`myReturned->Quality`**). For more information, see Reference a tag property instead of a project tag on page 169. However, when this function is called in VBScript, the returned value does not include quality, so this method will not work if you prefer to use VBScript in your project.

Second, you can call the function `TagsDBGetTagProperty` to check if the project tag specified by *strTagName* exists. It does not matter which property you try to get — if that function returns `-4`, it confirms the specified tag does not exist. However, TagsDB functions can be executed only in the full EcoStruxure Machine SCADA Expert 2020 software, so this method will not work if you try it on other platforms.

Third, as an alternative to the first and second methods above, you can create a VBScript procedure (in the Global Procedures interface, for example) that checks if a specified tag exists:

```
Function IsTagAvailable(tagName)
   $LocalPointer = tagName
   If $@LocalPointer->Name <> "" Then
      IsTagAvailable = 1
   Else
      IsTagAvailable = 0
   End If
End Function
```

The procedure above uses a tag pointer (@) to check the Name property of the specified tag. (Without the tag pointer, the procedure would check the Name property of **LocalPointer** itself.) If the property is not empty, the tag has a name and therefore exists.

### Examples

Get the current value of **myTag**:

```
GetTagValue("myTag")
```

Get the value of **myTag** and refresh it every second:

```
GetTagValue("myTag",Second)
```

Get the value of the tag specified by **myPointer** and refresh it whenever the value of **myTrigger** changes:

```
GetTagValue(myPointer,myTrigger)
```

## *SetTagValue*

Sets the value of the specified tag in the project tags database.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| SetTagValue | Tags Database | Synchronous | No | Supported | Supported | Supported |

### Syntax

```
SetTagValue( "strTagName", TagValue )
```

**strTagName**

The name of the tag that you want to set.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

**TagValue**

The new value to be set to the specified tag.

### Returned value

| Value | Description |
|---|---|
| -1 | Invalid tag name |
| 0 | No error |

### Examples

| Tag Name | Expression |
|---|---|
| TagA | **SetTagValue( "TagA", "Hello" )** // Return = Hello |
| TagA | **SetTagValue( "TagA", 123 )** // Return = 123 |
| TagA TagB = 15 | **SetTagValue( "TagA", TagB )** // Return = 15 |

## *TagsDBAddClass*

This function adds a new class to the tags database during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBAddClass | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

```
TagsDBAddClass( strClassName )
```

TagsDBAddClass(*strClassName*)

**strClassName**

The name of the class to be added.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -5 | A class with the specified name already exists in the tags database. |
| -4 | Maximum number of classes reached. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |
| -1 | The function TagsDBBeginEdit was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBAddClassMember*

This function adds a new class member to an existing class during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBAddClassMember | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

```
TagsDBAddClassMember( strClassName, strMemberName, strMemberType, strDescription )
```

TagsDBAddClassMember(*strClassName*, *strMemberName*,{ *strMemberType* | "Boolean" | "Integer" | "Real" | "String" }, *strDescription*)

**strClassName**

The name of the class to which the member will be added.

**strMemberName**

The name of the class member to be added.

**strMemberType**

The data type of the class member to be added. This parameter accepts only the following values: "Boolean", "Integer", "Real", or "String".

**strDescription**

A description of the class member to be added.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | The specified class does not exist in the tags database. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |
| -1 | The function TagsDBBeginEdit was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBAddTag*

This function adds a new project tag to the tags database during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBAddTag | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBAddTag( *strTagName*, *optstrTagType*, *optNumArraySize*, *optstrDescription*, *optNumScope* )**

TagsDBAddTag(*strTagName*{ | ,{ *optStrTagType* | "Boolean" | "Integer" | "Real" | "String" | "*classname*" }{ | ,*optNumArraySize*{ | ,*optStrDescription*{ | ,{ *optNumScope* | 0 | 1 }}}}})

**strTagName**

The name of the project tag to be added.

**optStrTagType**

The data type of the project tag to be added. This parameter accepts only the following values: "Boolean", "Integer", "Real", "String", or "*classname*".

This parameter is optional; if no value is specified, the default value is "Integer".

**optNumArraySize**

The array size of the project tag to be added.

This parameter is optional; if no value is specified, the default value is 0.

**optStrDescription**

A description of the project tag to be added.

This parameter is optional; if no value is specified, the default value is "".

**optNumScope**

The scope of the project tag to be added. This parameter accepts only the following values: 0 (Local) or 1 (Server).

This parameter is optional; if no value is specified, the default value is 1.

## Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -5 | The maximum number of tags (as determined by the project's target system) has been reached. |
| -4 | A project tag with the specified name already exists in the tags database. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |
| -1 | The function TagsDBBeginEdit was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

## Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBBeginEdit*

`TagsDBBeginEdit` is a built-in function that locks the tags database for editing during run time. You must call this function before you call any other Tags Database functions to add or remove project tags or set tag properties.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `TagsDBBeginEdit` | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**`TagsDBBeginEdit()`**

`TagsDBBeginEdit()`

This function has no parameters.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| `0` | Function executed successfully. |
| `-1` | Project runtime is busy; another function is editing the tags database. Try again later. |
| `-2` | Invalid number of parameters. |
| `-3` | Project runtime is busy; an Alarm or Trend task is using the tags database. Try again later. |

### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client; and
- The function `TagsDBEndEdit` must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

Please note this function has a persistent effect, which means that if you call the function to lock the tags database during run time and then stop the project, the database will remain locked and you will not be able to manually edit it.

Restarting the project may or may not unlock the database, depending on how you designed your project and which function call locked the database in the first place. As such, while the project is stopped, you should use the *Watch* window to manually call the function `TagsDBEndEdit`. When it is successfully executed, you can safely restart the project.

## *TagsDBEndEdit*

`TagsDBEndEdit` is a built-in function that finishes changes made by other Tags Database functions and releases the database so that normal run-time execution may resume.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `TagsDBEndEdit` | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBEndEdit()**

TagsDBEndEdit()

This function has no parameters.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Function successfully executed. |
| -1 | The function TagsDBBeginEdit was not successfully executed before this function was called. No changes were made. |
| -2 | Invalid number of parameters. |

### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client; and
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

Please note that when this function is executed, screens that were already open on connected clients will be reopened and their associated OnOpen screen scripts will be executed again.

## *TagsDBGetAlarm*

TagsDBGetAlarm is a built-in function that gets the current value of a property of an alarm condition during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetAlarm | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBGetAlarm(*strTagName*,*numAlarmType*,*numProperty*)**

TagsDBGetAlarm(*strTagName*,{ *numAlarmType* | -1 | 1 | 2 | 4 | 8 | 16 | 32 | 64 },{ *numProperty* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 })

***strTagName***

The name of the project tag on which the alarm property is set.

> **Tip:** To get other properties on project tags, use the function TagsDBGetTagProperty.

***numAlarmType***

The type of alarm on the specified project tag, identified by one of the following values:

| Value | Alarm Type |
|---|---|
| -1 | General properties that apply to all alarm types on the specified project tag |
| 1 | HiHi |
| 2 | Hi |
| 4 | Lo |
| 8 | LoLo |
| 16 | Rate |
| 32 | DeviationP |
| 64 | DeviationM |

***numProperty***

The alarm property that you want to get.

When getting a property of a specific alarm type (i.e., if *numAlarmType* is greater than 0), the property is identified by one of the following values:

| Value | Property | Data Type |
|---|---|---|
| 0 | Limit value | Real |
| 1 | Message | String |
| 2 | Alarm group (or worksheet) number | Integer |
| 3 | Priority | Integer |
| 4 | Selection | String |
| 5 | Custom field 1 | String |
| 6 | Custom field 2 | String |
| 7 | Custom field 3 | String |
| 8 | Custom field 4 | String |
| 9 | Custom field 5 | String |
| 10 | Custom field 6 | String |
| 11 | Custom field 7 | String |
| 12 | Custom field 8 | String |
| 13 | Custom field 9 | String |
| 14 | Custom field 10 | String |

When getting a property that applies to all alarm types on the specified project tag (i.e., if *numAlarmType* is -1), the property is identified by one of the following values:

| Value | Property | Data Type |
|---|---|---|
| 0 | Alarms Enabled | Boolean |
| 1 | Remote Ack tag | String (tag name) |
| 2 | Translation Enabled | Boolean |
| 3 | Dead Band Value | Real |
| 4 | Off | String |
| 5 | On | String |
| 6 | Ack | String |

| Value | Property | Data Type |
|---|---|---|
| 7 | Deviation Setpoint | String (tag name) |
| 8 | Deviation Dead Band | Real |

For more information about all of these alarm properties, see Tag Properties on page 164.

### Returned value

If this function is executed succesfully, it returns the value of the specified property. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | The specified project tag (*strTagName*) or alarm type (*numAlarmType*) does not exist. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |

### Notes

This function can only be executed on the project runtime server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBGetClassMember*

This function gets the data type or description of a specified class member during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetClassMember | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

```
TagsDBGetClassMember( strClassName, strMemberName, numPropertyID )
```

```
TagsDBGetClassMember(strClassName,strMemberName,{numPropertyID | 0 | 1})
```

**strClassName**

The name of the class that contains the member.

**strMemberName**

The name of the class member.

**numPropertyID**

The property to be gotten, identified by one of the following values:

| Value | Property | Data Type |
|---|---|---|
| 0 | Data type | String ("Boolean", "Integer", "Real", "String") |
| 1 | Description | String |

### Returned value

If this function is executed succesfully, then it returns the value of the specified property. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -5 | The specified member does not exist in the specified class. |
| -4 | The specified class does not exist in the tags database. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |

### Notes

This function can only be executed on the project runtime server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBGetClassMemberCount*

This function gets a count of the members in a specified class.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetClassMemberCount | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBGetClassMemberCount( *strClassName* )**

TagsDBGetClassMemberCount(*strClassName*)

**strClassName**

The name of the class in which to count members.

### Returned value

If this function is executed succesfully, then it returns the requested count of class members. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | The specified class does not exist. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |

### Notes

This function can only be executed on the project runtime server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBGetFirstClass*

This function returns the name of the first class in the tags database. It acts like an array pointer, with the array being the tags database, and it may be used in coordination with the function TagsDBGetNextClass

either to generate a list of classes in the tags database or to process the classes one at a time, depending on how you write your script.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetFirstClass | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBGetFirstClass()**

TagsDBGetFirstClass()

This function has no parameters.

### Returned value

If this function is executed succesfully, then it returns the name of the first class. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -3 | No classes found in the tags database. |
| -2 | Invalid number of parameters. |

### Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function may be called at any time; the function TagsDBBeginEdit does not need to have been executed previously. If that is the case, however, then TagsDBGetFirstClass and TagsDBGetNextClass can only generate a list of classes. They cannot be used, along with with the other Tags Database functions, to edit the classes.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBGetFirstClassMember*

This function returns the name of the first member in a specified class. It acts like an array pointer, with the array being the class members, and it may be used in coordination with the function TagsDBGetNextClassMember either to generate a list of members in the class or to process the class members one at a time, depending on how you write your script.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetFirstClassMember | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBGetFirstClassMember( *strClassName* )**

TagsDBGetFirstClassMember(*strClassName*)

**strClassName**

The name of the class in which to get the first member.

### Returned value

If this function is executed succesfully, then it returns the name of the first class member. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -5 | No members found in the specified class. |
| -4 | The specified class does not exist in the tags database. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |

### Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function may be called at any time; the function `TagsDBBeginEdit` does not need to have been executed previously. If that is the case, however, then `TagsDBGetFirstClassMember` and `TagsDBGetNextClassMember` can only generate a list of classes. They cannot be used, along with with the other Tags Database functions, to edit the classes.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBGetFirstTag*

`TagsDBGetFirstTag` is a built-in function that gets the name of the first tag in the tags database. It acts like an array pointer, with the array being the tags database, and it can be used in coordination with the `TagsDBGetNextTag` function either to generate a list of tags or to process the tags one at a time, depending on how you write your script.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetFirstTag | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**`TagsDBGetFirstTag(`*`optNumProjectOrSystem`*`)`**

`TagsDBGetFirstTag({ | {` *optNumProjectOrSystem* `| 0 | 1}})`

***optNumProjectOrSystem***

Numeric flag that determines whether to get from the user-defined Project Tags or from the pre-defined System Tags:

| Value | Description |
|---|---|
| 0 | Project Tags |
| 1 | System Tags |

This parameter is optional; if no value is specified, the default value is 1.

### Return value

If this function is executed succesfully, it returns the name of the first tag. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -2 | Invalid number of parameters. |

| Value | Description |
|---|---|
| -3 | No tags found in the tags database. |

### Notes

This function can be executed only on the project server. It cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function can be called at any time; the `TagsDBBeginEdit` function does not need to have been executed beforehand. If that is the case, however, then `TagsDBGetFirstTag` and `TagsDBGetNextTag` can be used only to generate a list of tags. They cannot be used — along with with the other Tags Database functions — to edit the tags unless `TagsDBBeginEdit` has been executed.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBGetLoadStatus*

`TagsDBGetLoadStatus` is a built-in function that checks whether a specified tag has been loaded into memory on a project thin client.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `TagsDBGetLoadStatus` | Tags Database | Synchronous | Yes | Not supported | Supported | Not supported |

### Syntax

**`TagsDBGetLoadStatus(`*`strFullTagName`*`)`**

`TagsDBGetLoadStatus(`*strFullTagName*`)`

***strFullTagName***

The name of the project tag to be checked. If it is an array and/or class, you must also specify the array position and/or class member, because only one position and/or member can be checked for each function call.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Returned value

This function will return one of the following possible values:

| Value | Description |
|---|---|
| 2 | Loaded. |
| 1 | Load pending. |
| 0 | Not loaded and no load pending. |
| -1 | This function can be called only on Thin Clients. |
| -2 | Invalid tag name; the specified project tag does not exist in the tags database. |
| -3 | Local tag specified. Only tags with Server scope can be specified for this function. |

### Notes

This function is typically called after the `TagsDBPreload` and `TagsDBPreloadWait` functions, to check whether a project tag selected for preloading has actually been loaded.

### Examples

The following example shows how the TagsDBPreload, TagsDBPreloadWait, TagsDBGetPreloadCount, and TagsDBGetLoadStatus functions can all be used in the **Screen_OnOpen** sub-routine of the Screen Script:

```
Sub Screen_OnOpen()

  Dim counter, preloadCount, preloadDone, loadStatus

  // Select tags for preloading.

  preloadCount = $TagsDBPreload("MyTagA")
  preloadCount = $TagsDBPreload("MyTagB")
  preloadCount = $TagsDBPreload("MyTagC")
  preloadCount = $TagsDBPreload("MyArray[1]")
  preloadCount = $TagsDBPreload("MyArray[2]")
  preloadCount = $TagsDBPreload("MyArray[3]")

  // Wait up to five seconds to load tags.

  counter = 0
  preloadDone = 0
  Do
    preloadDone = $TagsDBPreloadWait(1000)
    preloadCount = $TagsDBPreloadCount()
    counter = counter + 1
  Loop Until (preloadDone <> 0) Or (counter = 5)

  // Save preload count message to string tag, for later use.

  $PreloadCountMsg = preloadCount & " tag(s) waiting to be loaded."

  // Save load status message to string tag, for later use.

  loadStatus = $TagsDBGetLoadStatus("MyTagA")
  If (loadStatus = 2) Then
    $LoadStatusMsg = "MyTagA is loaded."
  ElseIf (loadStatus = 1) Then
    $LoadStatusMsg = "MyTagA is waiting to be loaded."
  Else
    $LoadStatusMsg = "Error while loading MyTagA."
  End If

End Sub
```

## *TagsDBGetNextClass*

This function returns the name of the next class in the tags database, after the function TagsDBGetFirstClass has been executed to get the first class. It acts like an array pointer, with the array being the tags database, and it may be used to generate a list of classes in the tags database or to process the classes one at a time, depending on how you write your script.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetNextClass | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

```
TagsDBGetNextClass()
```

TagsDBGetNextClass()

This function has no parameters.

### Returned value

If this function is executed succesfully, then it returns the name of the next class. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -3 | No class found; at end of database. |
| -2 | Invalid number of parameters. |

### Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function may be called at any time; the function TagsDBBeginEdit does not need to have been executed previously. If that is the case, however, then TagsDBGetFirstClass and TagsDBGetNextClass can only generate a list of classes. They cannot be used, along with with the other Tags Database functions, to edit the classes.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBGetNextClassMember*

TagsDBGetNextClassMember is a built-in function that returns the name of the next member in a specified class, after the TagsDBGetFirstClassMember function has been called to get the first member. This function acts like an array pointer, with the class being the array and the class members being the array elements, and it can be used to process the class members one at a time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetNextClassMember | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBGetNextClassMember()**

TagsDBGetNextClassMember()

This function has no parameters. You must include the empty parentheses, however, or else the function name might be interpreted as a tag name.

### Returned value

If this function is executed succesfully, it returns the name of the next class member. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -2 | Invalid number of parameters. |
| -3 | Wrong parameter type or inconsistent value. |
| -4 | The specified class does not exist in the tags database. |
| -5 | No members found; at end of class. |

### Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function may be called at any time; the function TagsDBBeginEdit does not need to have been executed previously. If that is the case, however, then TagsDBGetFirstClassMember and

TagsDBGetNextClassMember can only generate a list of classes. They cannot be used, along with with the other Tags Database functions, to edit the classes.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

### *TagsDBGetNextTag*

TagsDBGetNextTag is a built-in function that gets the name of the next tag in the tags database, after the TagsDBGetFirstTag function has been executed to get the first tag. It acts like an array pointer, with the array being the tags database, and it can be used either to generate a list of tags or to process the tags one at a time, depending on how you write your script.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetNextTag | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

#### Syntax

**TagsDBGetNextTag()**

TagsDBGetNextTag()

This function has no parameters.

#### Return value

If this function is executed succesfully, it returns the name of the next tag. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -2 | Invalid number of parameters. |
| -3 | No tag found; at end of database. |

#### Notes

This function can be executed only on the project server. It cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function can be called at any time; the TagsDBBeginEdit function does not need to have been executed beforehand. If that is the case, however, then TagsDBGetFirstTag and TagsDBGetNextTag can be used only to generate a list of tags. They cannot be used — along with with the other Tags Database functions — to edit the tags unless TagsDBBeginEdit has been executed.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

### *TagsDBGetPreloadCount*

TagsDBGetPreloadCount is a built-in function that gets the number of tags that are still waiting to be preloaded.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetPreloadCount | Tags Database | Synchronous | Yes | Not supported | Supported | Not supported |

#### Syntax

**TagsDBGetPreloadCount()**

TagsDBGetPreloadCount()

This function has no parameters.

### Returned value

If this function is executed successfully, it will return the number (greater than or equal to 0) of tags waiting to be preloaded. Otherwise, it will return one of the following possible values:

| Value | Description |
|---|---|
| -1 | This function can be called only on Thin Clients. |

### Notes

This function is typically called after the `TagsDBPreload` and `TagsDBPreloadWait` functions. In fact, when this function gets the number of tags that are still waiting to be preloaded, it only checks tags that were selected for preloading by calling the `TagsDBPreload` function.

If this function returns a value greater than or equal to 1, which indicates that one or more tags are still waiting to preloaded, you can use the `TagsDBGetLoadStatus` function to try to determine which tags those are.

### Examples

The following example shows how the `TagsDBPreload`, `TagsDBPreloadWait`, `TagsDBGetPreloadCount`, and `TagsDBGetLoadStatus` functions can all be used in the **`Screen_OnOpen`** sub-routine of the Screen Script:

```
Sub Screen_OnOpen()

  Dim counter, preloadCount, preloadDone, loadStatus

  // Select tags for preloading.

  preloadCount = $TagsDBPreload("MyTagA")
  preloadCount = $TagsDBPreload("MyTagB")
  preloadCount = $TagsDBPreload("MyTagC")
  preloadCount = $TagsDBPreload("MyArray[1]")
  preloadCount = $TagsDBPreload("MyArray[2]")
  preloadCount = $TagsDBPreload("MyArray[3]")

  // Wait up to five seconds to load tags.

  counter = 0
  preloadDone = 0
  Do
    preloadDone = $TagsDBPreloadWait(1000)
    preloadCount = $TagsDBPreloadCount()
    counter = counter + 1
  Loop Until (preloadDone <> 0) Or (counter = 5)

  // Save preload count message to string tag, for later use.

  $PreloadCountMsg = preloadCount & " tag(s) waiting to be loaded."

  // Save load status message to string tag, for later use.

  loadStatus = $TagsDBGetLoadStatus("MyTagA")
  If (loadStatus = 2) Then
    $LoadStatusMsg = "MyTagA is loaded."
  ElseIf (loadStatus = 1) Then
    $LoadStatusMsg = "MyTagA is waiting to be loaded."
  Else
    $LoadStatusMsg = "Error while loading MyTagA."
  End If

End Sub
```

### *TagsDBGetTagCount*

This function gets a count of the project tags in the tags database.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetTagCount | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

#### Syntax

**TagsDBGetTagCount( *numCountType* )**

TagsDBGetTagCount({ *numCountType* | 0 | 1 | 2 | 3 })

**numCountType**

The type of count to be performed, identified by one of the following values:

| Value | Description |
|---|---|
| 0 | The total number of project tags in the tags database. |
| 1 | The number of project tags supported by the target system. |
| 2 | The number of project tags that may still be created before exceeding the number supported by the target system. |
| 3 | The total number of pre-defined system tags in the tags database. |

#### Returned value

If this function is executed succesfully, then it returns the specified tag count. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |

#### Notes

This function can only be executed on the project runtime server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

### *TagsDBGetTagProperty*

TagsDBGetTagProperty is a built-in function that gets the value of a specified tag property during run time. Tag properties are the metadata on project tags, such as tag name, array size, data type, description, scope, and so on. In some cases, calling this function is the same as using tag fields (e.g., *tagname->property*).

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetTagProperty | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

#### Syntax

**TagsDBGetTagProperty(*strTagName*,*numPropertyID*)**

`TagsDBGetTagProperty(`*strTagName*`,{` *numPropertyID* `| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 })`

***strTagName***

The name of the project tag from which the tag property will be gotten.

***numPropertyID***

The tag property to be gotten, identified by one of the following values:

| Value | Property | Data Type | Same as... |
|---|---|---|---|
| 0 | Tag name | String | *tagname*->Name |
| 1 | Array size | Integer | *tagname*->Size |
| 2 | Data type | String ("Boolean", "Integer", "Real", "String", "*classname*") | - |
| 3 | Description | String | *tagname*->Description |
| 4 | Scope | Integer (0 for Local, 1 for Server) | - |
| 5 | Startup value | String | - |
| 6 | Minimum value | Real | *tagname*->Min |
| 7 | Maximum value | Real | *tagname*->Max |
| 8 | Engineering unit | String | *tagname*->Unit |
| 9 | Retentive value | Integer (0 for disabled, 1 for enabled) | - |
| 10 | Retentive properties | Integer (0 for disabled, 1 for enabled) | - |
| 11 | Dead band | Real | - |
| 12 | Smoothing | Integer (0 for disabled, 1 for enabled) | - |

**Tip:** To get alarm conditions on project tags, use the function TagsDBGetAlarm. Other tag properties not listed here cannot be gotten during run time.

## Returned value

If this function is executed succesfully, it returns the value of the specified tag property. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -2 | Invalid number of parameters. |
| -3 | Wrong parameter type or inconsistent value. |
| -4 | Project tag (as specified by *strTagName*) does not exist. |

## Notes

This function can only be executed on the project runtime server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBGetTrend*

This function determines the trend group to which a project tag is assigned.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBGetTrend | Tags Database | Synchronous | Yes | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBGetTrend( *strTagName*, *numProperty* )**

TagsDBGetTrend(*strTagName*,{ *numProperty* | 0 | 1 })

**strTagName**

The name of the project tag that is assigned to a trend group.

**numProperty**

The specific property to be gotten, identified by one of the following values:

| Value | Property | Data Type |
|---|---|---|
| 0 | Trend group (or worksheet) number | Integer |
| 1 | Log dead band | Real |

### Returned value

If this function is executed succesfully, then it returns the value of the specified property. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | Project tag (as specified by strTagName) does not exist. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |

### Notes

This function can only be executed on the project runtime server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBPreload*

TagsDBPreload is a built-in function that manually preloads a project tag into memory on a project thin client.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBPreload | Tags Database | Synchronous | No | Not supported | Supported | Not supported |

### Syntax

**TagsDBPreload(*strTagName*)**

TagsDBPreload(*strTagName*)

***strTagName***

The name of the project tag that you want to preload. If it is an array and/or class, and if you specify an array position and/or class member, only that position and/or member will be preloaded. Otherwise, all positions and/or members will be preloaded.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Returned value

If this function is executed successfully, it will return the number (greater than or equal to 1) of tags that are currently selected for preloading. Otherwise, it should return one of the following possible values:

| Value | Description |
|---|---|
| -1 | This function can be called only on Thin Clients. |
| -2 | Invalid tag name; the specified project tag does not exist in the tags database. |
| -3 | Local tag specified. Only tags with Server scope can be specified for this function. |

If this function returns a negative value that is not shown in the table above, please contact your software distributor for support.

### Notes

Each project thin client maintains a list of project tags to which it is subscribed. Being subscribed to a tag means the client keeps that tag loaded in memory, even when it is not being used, and the project runtime server sends updated tag values to the client only when those values change. This subscription model improves the run-time performance of the client and reduces unnecessary communication between the client and the server.

When the client opens a project screen for the first time, it automatically adds the project tags used in that screen to its subscription list. More specifically, the client does the following: it examines all of the objects and scripts included in the screen, it determines which tags are referenced by those objects and scripts, it adds those tags to its subscription list, and then it processes the list in order to get the latest values of the tags before it actually displays the screen. The client does all of this to make sure it can open the screen quickly and without errors, both the first time and every time thereafter.

In some cases, however, the client cannot determine for itself all of the tags it needs to add to its subscription list. Objects and scripts in the screen might not initialize correctly, and the screen might open slowly or with errors. When this happens, you can call the `TagsDBPreload` function to manually preload the missing tags.

For example, if a screen uses a large array tag — that is, an array tag with a large number of elements — it might take an unacceptably long time for the client to load the entire array and then open the screen. You can call this function to preload the specific elements of the array that might be used in the screen, or if you actually need to preload the entire array, you can make sure it is done before the screen is opened. Also, if a screen uses an indirect tag (e.g., **`@MyTag`**) that has no real value of its own until after the screen is opened and the tag value is resolved, you can can call this function to preload the base tag (e.g., **`MyTag`**).

You will typically call this function in the **`Screen_OnOpen`** sub-routine of the Screen Script for a given screen. Or, if you want to preload the same tag for several different screens, you can also call this function in the **`Graphics_OnStart`** sub-routine of the Graphics Script.

You can use this function in conjunction with the `TagsDBPreloadWait`, `TagsDBGetPreloadCount`, and `TagsDBGetLoadStatus` functions to manage the preloading of multiple tags. See "Examples" below.

Once this function is called to preload a tag for a screen, that tag is added to the client's subscription list and it does not need to be preloaded again for that screen or any other screen. If this function is called again for the same tag, it is ignored.

### Examples

The following example shows how the TagsDBPreload, TagsDBPreloadWait, TagsDBGetPreloadCount, and TagsDBGetLoadStatus functions can all be used in the **Screen_OnOpen** sub-routine of the Screen Script:

```
Sub Screen_OnOpen()

  Dim counter, preloadCount, preloadDone, loadStatus

  // Select tags for preloading.

  preloadCount = $TagsDBPreload("MyTagA")
  preloadCount = $TagsDBPreload("MyTagB")
  preloadCount = $TagsDBPreload("MyTagC")
  preloadCount = $TagsDBPreload("MyArray[1]")
  preloadCount = $TagsDBPreload("MyArray[2]")
  preloadCount = $TagsDBPreload("MyArray[3]")

  // Wait up to five seconds to load tags.

  counter = 0
  preloadDone = 0
  Do
    preloadDone = $TagsDBPreloadWait(1000)
    preloadCount = $TagsDBPreloadCount()
    counter = counter + 1
  Loop Until (preloadDone <> 0) Or (counter = 5)

  // Save preload count message to string tag, for later use.

  $PreloadCountMsg = preloadCount & " tag(s) waiting to be loaded."

  // Save load status message to string tag, for later use.

  loadStatus = $TagsDBGetLoadStatus("MyTagA")
  If (loadStatus = 2) Then
    $LoadStatusMsg = "MyTagA is loaded."
  ElseIf (loadStatus = 1) Then
    $LoadStatusMsg = "MyTagA is waiting to be loaded."
  Else
    $LoadStatusMsg = "Error while loading MyTagA."
  End If

End Sub
```

## *TagsDBPreloadWait*

TagsDBPreloadWait is a built-in function that pauses execution of the script in which it is called in order to preload project tags into memory on a project thin client.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBPreloadWait | Tags Database | Synchronous | No | Not supported | Supported | Not supported |

### Syntax

```
TagsDBPreloadWait(numTimeOut)
```

TagsDBPreloadWait(*numTimeOut*)

***numTimeOut***

The amount of time (in milliseconds) that the execution of the script will be paused. At the end of the specified timeout, this function checks whether any loads are still pending.

### Returned value

This function should return one of the following possible values:

| Value | Description |
|---|---|
| 1 | Success (i.e., all tags loaded). |
| 0 | Failure (i.e., function timed out while waiting for tags to be loaded). |
| -1 | This function can be called only on Thin Clients. |

If this function returns a negative value that is not shown in the table above, please contact your software distributor for support.

### Notes

This function must be called after the `TagsDBPreload` function. In fact, when this function checks whether any loads are still pending, it only checks tags that were selected for preloading by calling the `TagsDBPreload` function.

If this function returns a value of 0, which indicates that it timed out, you can use the `TagsDBGetPreloadCount` and `TagsDBGetLoadStatus` functions to try to determine why it timed out.

### Examples

The following example shows how the `TagsDBPreload`, `TagsDBPreloadWait`, `TagsDBGetPreloadCount`, and `TagsDBGetLoadStatus` functions can all be used in the **`Screen_OnOpen`** sub-routine of the Screen Script:

```
Sub Screen_OnOpen()

  Dim counter, preloadCount, preloadDone, loadStatus

  // Select tags for preloading.

  preloadCount = $TagsDBPreload("MyTagA")
  preloadCount = $TagsDBPreload("MyTagB")
  preloadCount = $TagsDBPreload("MyTagC")
  preloadCount = $TagsDBPreload("MyArray[1]")
  preloadCount = $TagsDBPreload("MyArray[2]")
  preloadCount = $TagsDBPreload("MyArray[3]")

  // Wait up to five seconds to load tags.

  counter = 0
  preloadDone = 0
  Do
    preloadDone = $TagsDBPreloadWait(1000)
    preloadCount = $TagsDBPreloadCount()
    counter = counter + 1
  Loop Until (preloadDone <> 0) Or (counter = 5)

  // Save preload count message to string tag, for later use.

  $PreloadCountMsg = preloadCount & " tag(s) waiting to be loaded."

  // Save load status message to string tag, for later use.

  loadStatus = $TagsDBGetLoadStatus("MyTagA")
  If (loadStatus = 2) Then
    $LoadStatusMsg = "MyTagA is loaded."
  ElseIf (loadStatus = 1) Then
    $LoadStatusMsg = "MyTagA is waiting to be loaded."
  Else
    $LoadStatusMsg = "Error while loading MyTagA."
  End If
```

```
End Sub
```

## TagsDBRemoveAlarm

`TagsDBRemoveAlarm` is a built-in function that removes an alarm condition from a project tag during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `TagsDBRemoveAlarm` | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

```
TagsDBRemoveAlarm(strTagName,numAlarmType)
```

`TagsDBRemoveAlarm(`*strTagName*`,{` *numAlarmType* `| 1 | 2 | 4 | 8 | 16 | 32 | 64 })`

***strTagName***

The name of the project tag from which the alarm condition will be removed.

***numAlarmType***

The type of alarm that you want to remove, identified by one of the following values:

| Value | Alarm Type |
|---|---|
| 1 | HiHi |
| 2 | Hi |
| 4 | Lo |
| 8 | LoLo |
| 16 | Rate |
| 32 | DeviationP |
| 64 | DeviationM |

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | The specified project tag (*strTagName*) does not exist. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |
| -1 | The function `TagsDBBeginEdit` was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function `TagsDBBeginEdit` must have been executed previously, in order to lock the tags database for editing; and

- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

### *TagsDBRemoveClass*

This function removes an existing class from the tags database during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBRemoveClass | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

#### Syntax

```
TagsDBRemoveClass( strClassName )
```

TagsDBRemoveClass(*strClassName*)

**strClassName**

The name of the class to be removed.

#### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -5 | The specified class does not exist in the tags database. |
| -4 | The specified class is being used (i.e., it has one or more tags associated with it). |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |
| -1 | The function TagsDBBeginEdit was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

#### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

### *TagsDBRemoveClassMember*

This function removes an existing class member from a specified class during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBRemoveClassMember | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

```
TagsDBRemoveClassMember( strClassName, strMemberName )
```

TagsDBRemoveClassMember(*strClassName*, *strMemberName*)

**strClassName**

The name of the class that contains the member to be removed.

**strMemberName**

The name of the class member to be removed.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -5 | The specified member does not exist in the specified class. |
| -4 | The specified class does not exist in the tags database. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |
| -1 | The function TagsDBBeginEdit was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBRemoveTag*

This function removes an existing project tag from the tags database during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBRemoveTag | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

```
TagsDBRemoveTag( strTagName )
```

TagsDBRemoveTag(*strTagName*)

**strTagName**

The name of the project tag to be removed.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | The specified tag does not exist in the tags database. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |
| -1 | The function TagsDBBeginEdit was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

Also, please note that executing this function does not completely remove the specified project tag from the tags database. That is not possible, due to how the database is maintained during run time. Instead, executing this function removes all uses of the specified tag in the project, so that they do not count against the tag limit on your runtime license, and then it reduces the tag to a single blank line in the database. Therefore, any time you execute this function to remove tags, you should stop the project as soon as it is practical to do so and then verify the project to remove the blank lines. For more information, see Verify the project on page 93.

## *TagsDBRemoveTrend*

This function removes a project tag from its trend group during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBRemoveTrend | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBRemoveTrend( *strTagName* )**

TagsDBRemoveTrend(*strTagName*)

**strTagName**

The name of the project tag to be removed.

### Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | Project tag (as specified by strTagName) does not exist. |
| -3 | Wrong parameter type or inconsistent value. |

| Value | Description |
|---|---|
| -2 | Invalid number of parameters. |
| -1 | The function TagsDBBeginEdit was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

## *TagsDBSetAlarm*

TagsDBSetAlarm is a built-in function that sets the value of a property of an alarm condition during run time.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBSetAlarm | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBSetAlarm(*strTagName*,*numAlarmType*,*numProperty*,*strOrNumVal*)**

TagsDBSetAlarm(*strTagName*,{ *numAlarmType* | -1 | 1 | 2 | 4 | 8 | 16 | 32 | 64 },{ *numProperty* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 }, *strOrNumVal*)

***strTagName***

The name of the project tag on which the alarm property will be set.

> **Tip:** To set other properties on project tags, use the function TagsDBSetTagProperty.

***numAlarmType***

The type of alarm on the specified project tag, identified by one of the following values:

| Value | Alarm Type |
|---|---|
| -1 | General properties that apply to all alarm types on the specified project tag |
| 1 | HiHi |
| 2 | Hi |
| 4 | Lo |
| 8 | LoLo |
| 16 | Rate |
| 32 | DeviationP |

| Value | Alarm Type |
|---|---|
| 64 | DeviationM |

***numProperty***

The alarm property that you want to set.

When setting a property of a specific alarm type (i.e., if *numAlarmType* is greater than 0), the property is identified by one of the following values:

| Value | Property | Data Type |
|---|---|---|
| 0 | Limit value | Real |
| 1 | Message | String |
| 2 | Alarm group (or worksheet) number | Integer |
| 3 | Priority | Integer |
| 4 | Selection | String |
| 5 | Custom field 1 | String |
| 6 | Custom field 2 | String |
| 7 | Custom field 3 | String |
| 8 | Custom field 4 | String |
| 9 | Custom field 5 | String |
| 10 | Custom field 6 | String |
| 11 | Custom field 7 | String |
| 12 | Custom field 8 | String |
| 13 | Custom field 9 | String |
| 14 | Custom field 10 | String |

When setting a property that applies to all alarm types on the specified project tag (i.e., if *numAlarmType* is -1), the property is identified by one of the following values:

| Value | Property | Data Type |
|---|---|---|
| 0 | Alarms Enabled | Boolean |
| 1 | Remote Ack tag | String (tag name) |
| 2 | Translation Enabled | Boolean |
| 3 | Dead Band Value | Real |
| 4 | Off | String |
| 5 | On | String |
| 6 | Ack | String |
| 7 | Deviation Setpoint | String (tag name) |
| 8 | Deviation Dead Band | Real |

For more information about all of these alarm properties, see Tag Properties on page 164.

***strOrNumVal***

The value to set to the specified property. The value must be of the appropriate data type.

## Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | The specified project tag (*strTagName*) does not exist. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |
| -1 | The function TagsDBBeginEdit was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

> **Note:** A specific alarm condition, as defined by the tag name and the alarm type (e.g., the HiHi alarm on **MyTag1**), can be in only one alarm group at a time. Therefore, please remember that when you assign an alarm condition to an alarm group, it will be removed automatically from its previous group, if any.
>
> Also, make sure that an alarm group with the correct group/worksheet number actually exists before you try to assign an alarm condition to it.

## *TagsDBSetTagProperty*

This function sets the value of a specified tag property during run time. Tag properties are the metadata on project tags, such as tag name, array size, data type, description, scope, and so on.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBSetTagProperty | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

### Syntax

**TagsDBSetTagProperty( *strTagName*, *numPropertyID*, *numOrStrPropertyValue* )**

TagsDBSetTagProperty(*strTagName*,{ *numPropertyID* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 }, *numOrStrPropertyValue*)

**strTagName**

The name of the project tag on which the tag property will be set.

**numPropertyID**

The tag property to be set, identified by one of the following values:

| Value | Property | Data Type |
|---|---|---|
| 0 | Tag name | String |
| 1 | Array size | Integer |

| Value | Property | Data Type |
|---|---|---|
| 2 | Data type | String ("Boolean", "Integer", "Real", "String", "*classname*") |
| 3 | Description | String |
| 4 | Scope | Integer (0 for Local, 1 for Server) |
| 5 | Startup value | String |
| 6 | Minimum value | Real |
| 7 | Maximum value | Real |
| 8 | Engineering unit | String |
| 9 | Retentive value | Integer (0 for disabled, 1 for enabled) |
| 10 | Retentive properties | Integer (0 for disabled, 1 for enabled) |
| 11 | Dead band | Real |
| 12 | Smoothing | Integer (0 for disabled, 1 for enabled) |



**Tip:** To set alarm conditions on project tags, use the function TagsDBSetAlarm. Other tag properties not listed here cannot be set during run time.

**numOrStrPropertyValue**

The value to set to the tag property. The value must be of the appropriate data type.

## Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | Project tag (as specified by strTagName) does not exist. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |
| -1 | The function TagsDBBeginEdit was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

## Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

### *TagsDBSetTrend*

This function assigns a project tag to a trend group.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| TagsDBSetTrend | Tags Database | Synchronous | No | Supported (see "Notes" below) | Not supported | Not supported |

## Syntax

```
TagsDBSetTrend( strTagName, numProperty, strOrNumVal )
```

TagsDBSetTrend(*strTagName*,{ *numProperty* | 0 | 1 },*strOrNumVal*)

**strTagName**

The name of the project tag to assign to the trend group.

**numProperty**

The specific property to be set, identified by one of the following values:

| Value | Property | Data Type |
|---|---|---|
| 0 | Trend group (or worksheet) number | Integer |
| 1 | Log dead band | Real |

**strOrNumVal**

The value to set to the specified property. The value must be of the appropriate data type.

## Returned value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | Project tag (as specified by strTagName) does not exist. |
| -3 | Wrong parameter type or inconsistent value. |
| -2 | Invalid number of parameters. |
| -1 | The function TagsDBBeginEdit was not executed successfully before this function was called. |
| 0 | Function executed successfully. |

## Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Using TagsDB functions to edit the tags database during run time on page 169.

> **Note:** A specific project tag can be in only one trend group at a time. Therefore, when you assign a project tag to a trend group, it will be removed automatically from its previous group, if any.

Also, make sure that a trend group with the correct group/worksheet number actually exists before you try to assign a project tag to it.

Finally, new trends will not start recording until either the system clock changes to a new day or you restart the project.

## *TagsDBSync*

`TagsDBSync` is a built-in function that forces the project thin client to get the latest value of a project tag from the project runtime server.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `TagsDBSync` | Tags Database | Synchronous | No | Not supported | Supported | Not supported |

### Syntax

```
TagsDBSync(strTagName)
```

`TagsDBSync(`*strTagName*`)`

***strTagName***

The name of the project tag to be synchronized. If it is an array and/or class, and if you specify an array position and/or class member, only that position and/or member will be synchronized. Otherwise, all positions and/or members will be synchronized.

> **Note:** If this parameter is enclosed in quotes, the string literal will be used. Otherwise, the value of the tag/expression will be used.

### Returned value

This function will return one of the following possible values:

| Value | Description |
|---|---|
| 0 | Success (i.e., tag value synchronized with project runtime server). |
| -1 | This function can be called only on Thin Clients. |
| -2 | Invalid tag name; the specified project tag does not exist in the tags database. |
| -3 | Local tag specified. Only tags with Server scope can be specified for this function. |

### Notes

If a project thin client is subscribed to a project tag in the tags database on the project runtime server, the server will send updates to the client whenever the tag value changes. In most cases these updates are effectively instantaneous, but sometimes there might be a delay due to poor run-time performance on the server, the client, or the network itself, and this delay can cause issues in the execution of scripts on the client.

This function forces the project thin client to synchronize a tag value — that is, get the latest value of the tag from the project runtime server — before it tries to use that tag value. This is especially useful when the client executes one script that sets the tag value and then another script that uses the tag value. For example, given the following scripts…

**`Graphics_OnStart`** sub-routine of the Graphics Script:

```
$MyTag = 90
$Open("Screen1")
```

**Screen_OnOpen** sub-routine of the Screen Script for Screen1:

```
myVar = $MyTag
```

…you would expect **myVar** to have a value of 90, after Screen1 is opened. However, setting the value of **MyTag** in the first script merely submits that change to the project runtime server. The server still needs to save the change in the tags database and then send the updated tag value back to the client. If there is a delay, as described above, then the second script might be executed using the old value of **MyTag**. This issue can be resolved by modifying the second script…

OnOpen script for Screen1:

```
$TagsDBSync("MyTag")
myVar = $MyTag
```

Again, in most cases, tag value updates are effectively instantaneous, so you should use this function only when it is necessary to resolve issues. Calling and executing unnecessary functions can affect run-time performance.

### Examples

Synchronize the project tag named **MyTag**:

```
TagsDBSync("MyTag")
```

Synchronize the entire array named **MyArray**:

```
TagsDBSync("MyArray")
```

Synchronize a single position of **MyArray**:

```
TagsDBSync("MyArray[123]")
```

## Translation functions

These functions are used to access the translation tool during runtime.

### *Ext*

`Ext` is a built-in function that uses the Translation Table feature to translate specified text.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `Ext` | Translation | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

```
Ext(strText)
```

`Ext` (*strText*)

***strText***

The text to be translated.

#### Return value

This function returns a translation of the specified text, according to the current target language.

#### Examples

Translate "Start":

```
Ext("Start")
```

Translate the text that is stored in the tag **`MyString`**:

```
Ext(MyString)
```

### *SetLanguage*

`SetLanguage` is a built-in function that sets the project translation to one of the languages configured in the Translation Table.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `SetLanguage` | Translation | Synchronous | No | Supported | Supported | Supported |

#### Syntax

```
SetLanguage(numLanguageID)
```

`SetLanguage` (*numLanguageID*)

***numLanguageID***

The ID number of the langauge that you want to set as the active translation. The language must be already configured in the Translation Table worksheet, and the ID number is shown in parentheses next to that language — for example, "English-United States (1033)". For more information about the ID number, see "Notes" below.

#### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Error |
| 1 | Success |

### Notes

The Translation Table uses Microsoft Locale ID (LCID) values to identify languages. LCID values were the proprietary language/region codes used in Microsoft Windows up until Windows 8. Microsoft changed how it handles language identifiers in Windows 10, which effectively deprecated LCID in the operating system, but we hardcoded the values into this software and continue to use them in order to maintain compatibility across all platforms and with existing projects.

### Examples

Set the language to "French-France (1036)":

```
SetLanguage(1036)
```

Set the language to "Portuguese-Brazil (1046)":

```
SetLanguage(1046)
```

## *TranslationLoad*

`TranslationLoad` is a built-in function that loads an external translation file and then either replaces or appends to the current translation.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `TranslationLoad` | Translation | Synchronous | No | Supported | Not supported (see "Notes" below) | Executed on Server (see "Notes" below) |

### Syntax

```
TranslationLoad(strFileName,numAppend)
```

`TranslationLoad(`*strFileName*`,` *numAppend*`)`

***strFileName***

The name of the translation file (.trn) to be loaded. The file must be located in the Web sub-folder of your project folder (e.g., `<project name>\Web\Translation.trn`).

Translation files are simply tab-separated text files, and you can use any spreadsheet or text editor to create and edit them, but in most cases you should be able to duplicate an existing file and then use the Translation Table worksheet in the project development environment to edit it. For more information, see Project Localization on page 681.

***numAppend***

A numeric flag that indicates whether the loaded translation file replaces or appends to the existing contents of the Translation Table. Specify one of the following values:

| Value | Description |
|---|---|
| 0 | Completely replace the contents of the Translation Table with the contents of the loaded translation file. |
| 1 | Keep the contents of the Translation Table, and then append the contents of the loaded translation file only if they are not already present. |

| Value | Description |
|---|---|
| 2 | Keep the contents of the Translation Table, and then append the contents of the loaded translation file, replacing any duplicates. |
| 3 | Reset the Translation Table to its initial contents when the project was run. If this value is specified, *strFileName* is ignored. |

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| -3 | Failed to load the translation file. The file format is invalid or you do not have the necessary privileges. |
| -2 | The specified file cannot be found. |
| -1 | Invalid parameter(s). |
| 0 | Function executed successfully. |

### Notes

The Translation Table is maintained by the project runtime server, and the contents of the table are available to all connected thin clients. Each client can call the `SetLanguage` function to translate the project screens that it displays, but changes to the Translation Table itself must be done on the server. As such, this function is not supported on Thin Clients; it must be called in a procedure, script, or worksheet that is executed on the server. If you want to trigger the execution of this function from a Thin Client, either configure the worksheet to have an appropriate execution control or use the `RunGlobalProcedureOnServer` function.

This function is supported on Mobile Access, because the Mobile Access runtime maintains its own Translation Table separate from the project runtime server and the contents of the table are available only to the Mobile Access web interface. In order to make changes to the Translation Table for Mobile Access, this function must be called in a project screen as it is being viewed in the Mobile Access web interface. Unfortunately, due to this fundamental difference between the project runtime server and the Mobile Access runtime, it is not possible to develop a project screen that behaves exactly the same for both.

### Examples

Load `alternate.trn` and replace the current contents of the Translation Table:

```
TranslationLoad("alternate.trn",0)
```

Load `pt-BR.trn` and append it to the current contents of the Translation Table:

```
TranslationLoad("pt-BR.trn",1)
```

Reset the Translation Table to its initial contents when the project was run, or in other words, reload the default translation file (`Translation.trn`):

```
TranslationLoad("",3)
```

## *TranslationLookupClose*

`TranslationLookupClose` is a built-in function that closes a lookup map previously loaded into memory by the `TranslationLookupLoad` function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `TranslationLookupClose` | Translation | Synchronous | No | Supported | Not supported | Supported |

### Syntax

```
TranslationLookupClose(numTranslationLookupID)
```

`TranslationLookupClose(`*numTranslationLookupID*`)`

***numTranslationLookupID***

The ID number of the lookup map that you want to close, as it was returned by the `TranslationLookupLoad` function. Valid values range from 0 to 255.

### Return value

This function returns one of the following possible values:

| Value | Description |
|---|---|
| 0 | Lookup map successfully closed. |
| -1 | Function executed from the Viewer module, or other internal error. |
| -2 | Invalid parameter(s). |
| -3 | Other error. |

### Notes

Closing a lookup map frees the system resources used by that map. You should always close a lookup map when you are done with it.

This function is supported on Mobile Access, but it must be called in a project screen as it is being viewed in the Mobile Access web interface. For more information, see TranslationLookupLoad.

### Examples

Close the lookup map specified by the ID number stored in the tag **`MyLookupID`**:

```
TranslationLookupClose(MyLookupID)
```

## *TranslationLookupGet*

`TranslationLookupGet` is a built-in function that gets the translation of a specified source text, using a lookup map that was previously loaded into memory by the `TranslationLookupLoad` function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `TranslationLookupGet` | Translation | Synchronous | Yes | Supported | Not supported | Supported |

### Syntax

```
TranslationLookupGet(numTranslationLookupID,strSource)
```

`TranslationLookupGet(`*numTranslationLookupID*`,` *strSource*`)`

***numTranslationLookupID***

The ID number of the lookup map that you want to use, as it was returned by the `TranslationLookupLoad` function. Valid values range from `0` to `255`.

***strSource***

The source text for which you want to get the translation.

### Return value

If this function is successfully executed, it returns the translation of the specified source text. Otherwise, it returns one of the following possible values:

| Value | Description |
|---|---|
| *strSource* | Source text returned untranslated, because it was not found in the lookup map. |
| -1 | Function executed from the Viewer module, or other internal error. |
| -2 | Invalid parameter(s). |
| -3 | Other error. |

### Notes

This function is supported on Mobile Access, but it must be called in a project screen as it is being viewed in the Mobile Access web interface. For more information, see TranslationLookupLoad.

### Examples

Get the translation of "Tank #1", using the lookup map specified by the ID number stored in the tag **MyLookupID**:

```
TranslationLookupGet(MyLookupID,"Tank #1")
```

Load a lookup map for translating from "English-United States (1033)" to "Portuguese-Brazil (1046)", and then use it to get the translation of "Tank #1":

```
TranslationLookupGet(TranslationLookupLoad(1033,1046),"Tank #1")
```

## *TranslationLookupLoad*

`TranslationLookupLoad` is a built-in function that creates a lookup map from the Translation Table for the specified source and target languages, and then it loads the map into memory so that it can be used by the `TranslationLookupGet` function.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| `TranslationLookupLoad` | Translation | Synchronous | No | Supported | Not supported (see "Notes" below) | Executed on Server (see "Notes" below) |

### Syntax

```
TranslationLookupLoad(numSourceLanguageID,numTargetLanguageID)
```

`TranslationLookupLoad(`*numSourceLanguageID*`,` *numTargetLanguageID*`)`

***numSourceLanguageID,numTargetLanguageID***

The ID numbers of the source and target langauges, respectively. Both languages must be already configured in the Translation Table, and the ID numbers are shown in parentheses next to those languages — for example, "English-United States (1033)". For more information about the ID numbers, see "Notes" below.

### Return value

If this function is successfully executed, it returns the ID number (from 0 to 255) of the lookup map that was created and loaded. The ID number of the lookup map is separate and different from the ID numbers of the source and target languages. You can subsequently use the ID number of the lookup map as an argument when you call the `TranslationLookupGet` and `TranslationLookupClose` functions.

Otherwise, this function returns one of the following possible values:

| Value | Description |
|---|---|
| -4 | The source language is the same as the target language, or the specified languages have not been configured in the Translation Table. |
| -3 | Cannot have more than 256 lookup maps loaded in memory at the same time. |
| -2 | Invalid parameter(s). |
| -1 | Function executed from the Viewer module, or other internal error. |

## Notes

The Translation Table uses Microsoft Locale ID (LCID) values to identify languages. LCID values were the proprietary language/region codes used in Microsoft Windows up until Windows 8. Microsoft changed how it handles language identifiers in Windows 10, which effectively deprecated LCID in the operating system, but we hardcoded the values into this software and continue to use them in order to maintain compatibility across all platforms and with existing projects.

When you call this function, you can specify any two languages that have been configured in the Translation Table, even if they were both originally configured as target languages. When the lookup map is created, it automatically cross-references the specified languages. For example, if you configured the Translation Table for your project to include both an English-to-Spanish translation and an English-to-Portuguese translation, you can then specify Spanish and Portuguese when you call this function.

You can have up to 256 lookup maps loaded in memory at the same time, but each map uses system resources and that can affect run-time performance. You should use the `TranslationLookupClose` function to close a map when you are done with it.

The Translation Table and all lookup maps created from it are maintained by the project runtime server. As such, this function is not supported on Thin Clients; it must be called in a procedure, script, or worksheet that is executed on the server. If you want to trigger the execution of this function from a Thin Client, either configure the worksheet to have an appropriate execution control or use the `RunGlobalProcedureOnServer` function.

This function is supported on Mobile Access, because the Mobile Access runtime maintains its own Translation Table separate from the project runtime server and the contents of the table are available only to the Mobile Access web interface. In order to create a lookup map from the Translation Table for Mobile Access, this function must be called in a project screen as it is being viewed in the Mobile Access web interface. Unfortunately, due to this fundamental difference between the project runtime server and the Mobile Access runtime, it is not possible to develop a project screen that behaves exactly the same for both.

## Examples

Load a lookup map for translating from "English-United States (1033)" to "Portuguese-Brazil (1046)":

```
TranslationLookupLoad(1033,1046)
```

## Trigonometric functions

These functions are used to perform trigonometric operations (e.g., sine, cosine, tangent) on numeric values.

### *ACos*

Calculates the Arc Cosine of a value.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ACos | Trigonometric | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

```
ACos( numValue )
```

**numValue**

Numerical tag from which the Arc Cosine will be taken.

#### Returned value

Returns the Arc Cosine of numValue in radians.

#### Examples

| Tag Name | Expression |
|---|---|
| Tag | `ACos( 1 )` // Returned value = 0.000000 |
| Tag | `ACos( 0 )` // Returned value = 1.570796 |

### *ASin*

Calculates the Arc Sine of a value.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ASin | Trigonometric | Synchronous | Yes | Supported | Supported | Supported |

#### Syntax

```
ASin( numValue )
```

**numValue**

Numerical tag from which the Arc Sine will be taken.

#### Returned value

Returns the Arc Sine of numValue in radians.

#### Examples

| Tag Name | Expression |
|---|---|
| Tag | `ASin( 1 )` // Returned value = 1.570796 |
| Tag | `ASin( 0 )` // Returned value = 0.000000 |

## *ATan*

Calculates the Arc Tangent of a value.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| ATan | Trigonometric | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
ATan( numValue )
```

**numValue**

Numerical tag from which the Arc Tangent will be taken.

### Returned value

Returns the Arc Tangent of numValue in radians.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `ATan( 1 )` // Returned value = 0.785398 |
| Tag | `ATan( 0 )` // Returned value = 1.570796 |

## *Cos*

Calculates the Cosine of a value.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Cos | Trigonometric | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Cos( numAngle )
```

**numAngle**

The Angle (in radians) from which to calculate the Cosine.

### Returned value

Returns the Cosine of numAngle.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `Cos( 1.570796 )` // Returned value = 0.000000 |
| Tag | `Cos( 0 )` // Returned value = 1.000000 |

## *Cot*

Cot is a built-in function that calculates the cotangent of an angle.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Cot | Trigonometric | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Cot(numAngle)
```

Cot(*numAngle*)

***numAngle***

The angle in radians.

> **Tip:** 2π radians is the same as 360 degrees.

### Returned value

This function returns the cotangent of the specified angle.

The function cannot return the actual cotangent of π (i.e., **`Cot(Pi())`** or equivalent), because that is infinite. Instead, the function returns the largest number possible, given the limited precision of the value returned by the function `Pi`.

### Examples

Calculate the cotangent of 1 radian:

```
Cot(1)
```

Calculate the cotangent of π/2 radians:

```
Cot(Pi()/2)
```

Calculate the cotangent of the value of the project tag **`MyAngle`**:

```
Cot(MyAngle)
```

## *Pi*

Calculates the value of *pi*.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Pi | Trigonometric | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Pi()
```

This function takes no parameters. You must still include the parentheses, however, or it will be evaluated as a tag rather than a function.

### Returned value

Returns the value of *pi*.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | **`Pi()`** // Returned value = 3.141593 |

## *Sin*

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Sin | Trigonometric | Synchronous | Yes | Supported | Supported | Supported |

### Description

Calculates the Sine of a value.

### Syntax

```
Sin( numAngle )
```

**numAngle**

The Angle (in radians) from which to calculate the Sine.

### Returned value

Returns the Sine of numAngle.

### Examples

| Tag Name | Expression |
|---|---|
| Tag | `Sin( 0 )` // Returned value = 0.000000 |
| Tag | `Sin( 1.570796 )` // Returned value = 1.000000 |

## *Tan*

The function Tan calculates the tangent of an angle.

| Function | Group | Execution | String Exp. | Windows | Thin Clients | Mobile Access |
|---|---|---|---|---|---|---|
| Tan | Trigonometric | Synchronous | Yes | Supported | Supported | Supported |

### Syntax

```
Tan(numAngle)
```

Tan(*numAngle*)

***numAngle***

The angle in radians.

> **Tip:** 2π radians is the same as 360 degrees.

### Returned value

This function returns the tangent of the specified angle.

The function cannot return the actual tangent of π/2 (i.e., `Tan(Pi()/2)` or equivalent), because that is infinite. Instead, the function returns the largest number possible, given the limited precision of the value returned by the function Pi.

### Examples

Calculate the tangent of 1 radian:

```
Tan(1)
```

Calculate the tangent of π radians:

```
Tan(Pi())
```

Calculate the tangent of the value of the project tag **MyAngle**:

```
Tan(MyAngle)
```

# Overview of VBScript

VBScript is a simple, standard and flexible scripting language that allows you to implement logics and algorithms within your project.

EcoStruxure Machine SCADA Expert implements Visual Basic Scripting Edition 5.5 or higher. Because EcoStruxure Machine SCADA Expert hosts VBScript, you can take advantage of every feature provided by this language, such as:

- Syntax, operators and functions.
- The ability to create new variables and procedures (functions and/or sub-routines).
- Access to properties, methods and/or events from COM objects, including ActiveX controls.
- The ability to execute the logics in any platform that supports VBScript, which typically includes Windows computers and Windows Embedded devices.

> **Note:** If you are not sure if the image loaded on your device supports VBScript, please consult the hardware manufacturer. The hardware manufacturer must enable the support for VBScript on the Windows Embedded device, so project runtime server will be able to execute the scripts configured in the VBScript language on the device.

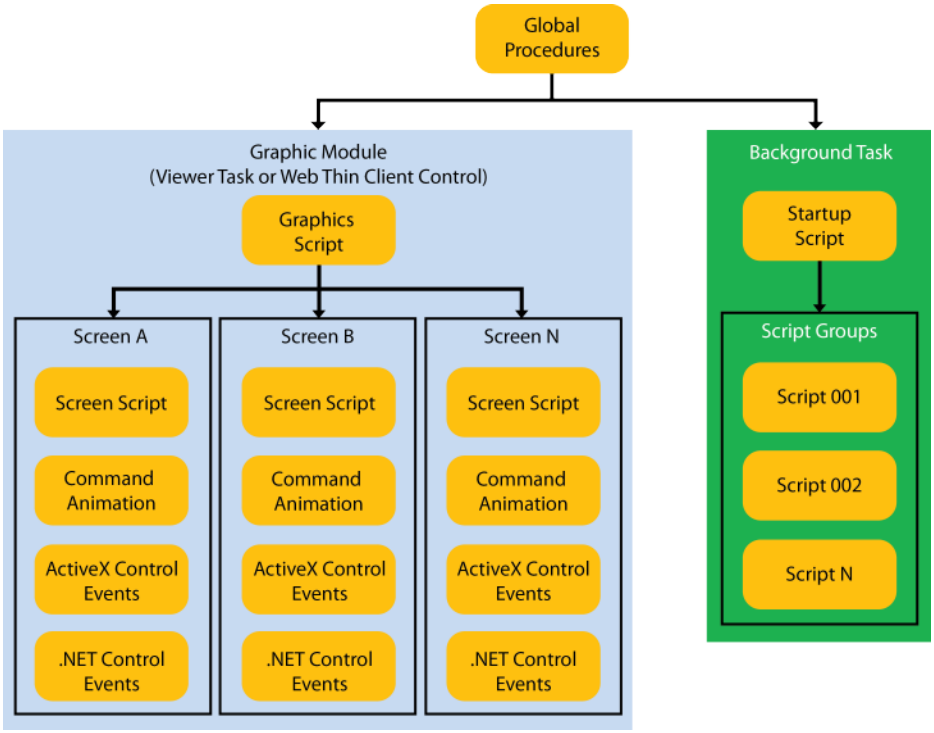
The aim of this documentation is to provide an overview about the integration of VBScript with EcoStruxure Machine SCADA Expert 2020. Furthermore, it can be used as a quick reference for the most used features of the language. For a full description of the language as well as its interfaces and functions, please consult Microsoft. (At the time of this writing, the VBScript documentation could be accessed directly at the Microsoft Developer Network. This link, however, is beyond our control and may change without notice.)

## VBScript Interfaces in the Software

The following table provides a summary of the VBScript interfaces supported by EcoStruxure Machine SCADA Expert 2020:

| Interface | Scope for Procedures and Variables | Execution | Functionality |
|---|---|---|---|
| Global Procedures | Graphics and Tasks | - | Declaration of Procedures |
| Graphics Script | Graphics Script interface only | Server (Viewer) + Thin Clients | Declaration of Variables<br>Declaration of Procedures<br>Execution |
| Screen Script | Screen where the script is configured | Server (Viewer) + Thin Clients | Declaration of Variables<br>Declaration of Procedures<br>Execution |
| Command animation | Object where the script is configured | Server (Viewer) + Thin Clients | Declaration of Variables<br>Execution |
| ActiveX Events | Object where the script is configured | Server (Viewer) + Thin Clients | Declaration of Variables<br>Execution |
| Startup Script | All Script Sheets from Tasks | Server (BGTask) | Declaration of Variables<br>Declaration of Procedures<br>Execution |
| Script Groups | Script Group only | Server (BGTask) | Declaration of Variables<br>Execution |

The following illustration shows the scope of each VBScript interface and the order in which they are scanned by EcoStruxure Machine SCADA Expert:




The illustration shows that the Global Procedures are shared by the Graphic Module and the Background Task. However, the other VBScript interfaces are either from the Graphic Module or from the Background Task, and they do not share variables or procedures between them. They are independent of each other.

> **Note:** Although the Graphics Script is scanned by EcoStruxure Machine SCADA Expert before the Screen Scripts, the procedures and variables declared in the Graphics Script interface are NOT available for any script interface configured on the screens. You must use the Global Procedures interface to implement procedures that must be available for all screens.

When writing your code in a VBScript interface, you can access any tag from the EcoStruxure Machine SCADA Expert tags database or any function from the Built-in Scripting Language by applying the "$" prefix to the tag/function name, as in the examples below:

```
$Time 'Returns the value of the tag Time from the tags database
$MyTag 'Returns the value of the tag MyTag from the tags database
$Open("main") 'Executes the Open() built-in function to open the "main" screen
```

Therefore, you can create scripts using built-in functions from EcoStruxure Machine SCADA Expert, tags from the EcoStruxure Machine SCADA Expert tags database, VBScript functions, VBScript variables, ActiveX properties, methods or events, and any other interface available. The EcoStruxure Machine SCADA Expert tags are shared by all modules from EcoStruxure Machine SCADA Expert, including the Graphic Module and the Background Task.

### *Global Procedures*

This Procedures interface is used create a library of VBScript functions and sub-routines that can be called by any other scripting interface in EcoStruxure Machine SCADA Expert 2020. The procedures declared here are never directly executed during runtime; they must be explicitly called by another script.

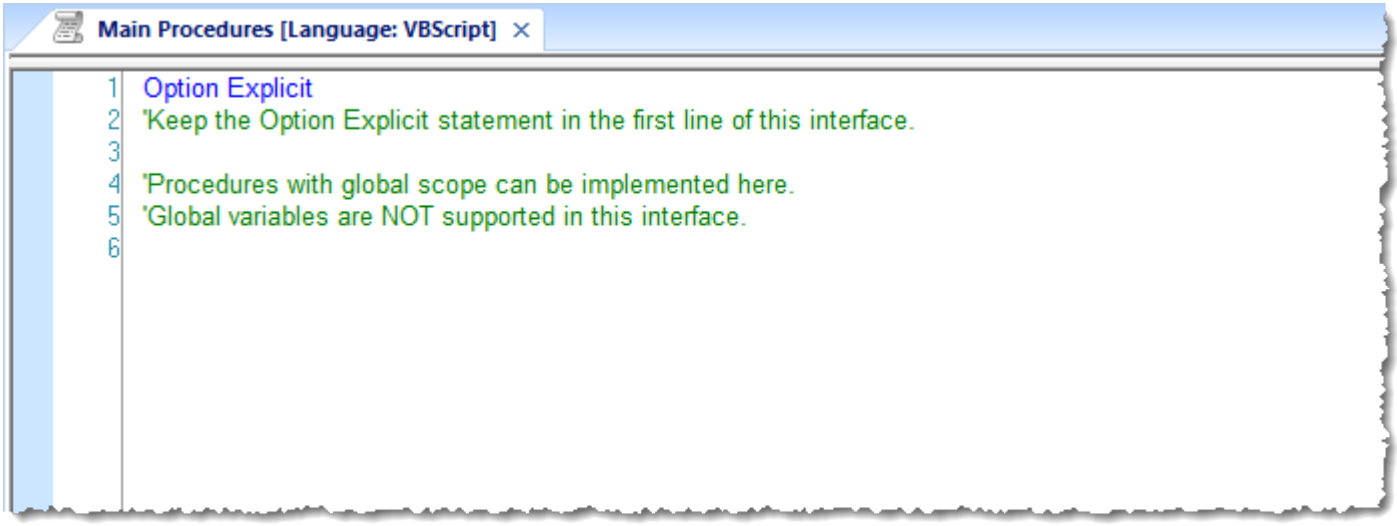
> **Note:** EcoStruxure Machine SCADA Expert 2020 will not prevent you from declaring two or more functions with the same name. (This includes functions imported from external files; see "Importing Functions from an External File" below.) However, if you do, then your project may behave unexpectedly during runtime. Make sure that all of your functions are named correctly.

You can create as many Procedures worksheets as you want, and each worksheet can contain as many functions and sub-routines as you want.

To use the Procedures interface:

1. In the **Global** tab of the Project Explorer, do one of the following:
   - To edit the default Procedures worksheet, open the **Procedures** folder and then double-click **Main Procedures**; or
   - To create a new Procedures worksheet, right-click the **Procedures** folder and then click **Insert** on the shortcut menu.

   Either way, the worksheet is opened for editing.



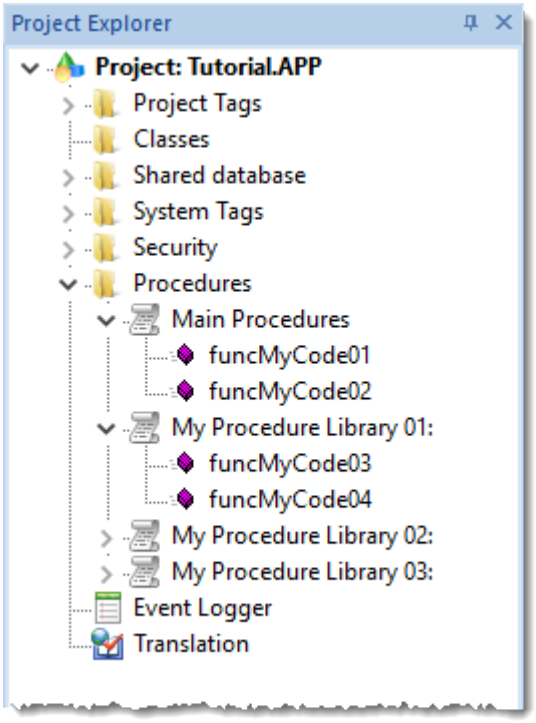

***Main Procedures worksheet***

2. Declare your functions and sub-routines in the interface. For example:

```
Option Explicit
'Keep the Option Explicit statement in the first line of this interface.

'Procedures with global scope can be implemented here
'Global variables are NOT supported in this interface

Sub MyMessage(message)
  MsgBox message, 0
End Sub

Function MyAdd(number1, number2)
  MyAdd = number1 + number2
  Call MyMessage("The sum is" & MyAdd & ".")
End Function
```

> **Note:** You can declare local variables within each procedure, but you cannot declare global variables in this interface. In most cases, you should use tags instead.

3. Save your changes. The worksheet is added to the **Procedures** folder in the Project Explorer.




***Procedure worksheets with declared functions***

### Organizing procedures into sub-folders

If you have many procedures in a single Procedures worksheet, then you may choose to organize them into sub-folders. To organize procedures:

1. In the **Procedures** folder, open the worksheet that you want to organize.
2. In the worksheet, insert the following line before the procedures that you want to group together:

```
'$region:foldername
```

...where *foldername* is the name of the sub-folder. For example:

```
'$region:My Subroutines
Sub MyMessage(message)
  MsgBox message, 0
End Sub

'$region:My Functions
Function MyAdd(number1, number2)
  MyAdd = number1 + number2
  Call MyMessage("The sum is" & MyAdd & ".")
End Function
```

3. Save your changes. The procedures are organized into sub-folders under the Procedures worksheet in the Project Explorer.

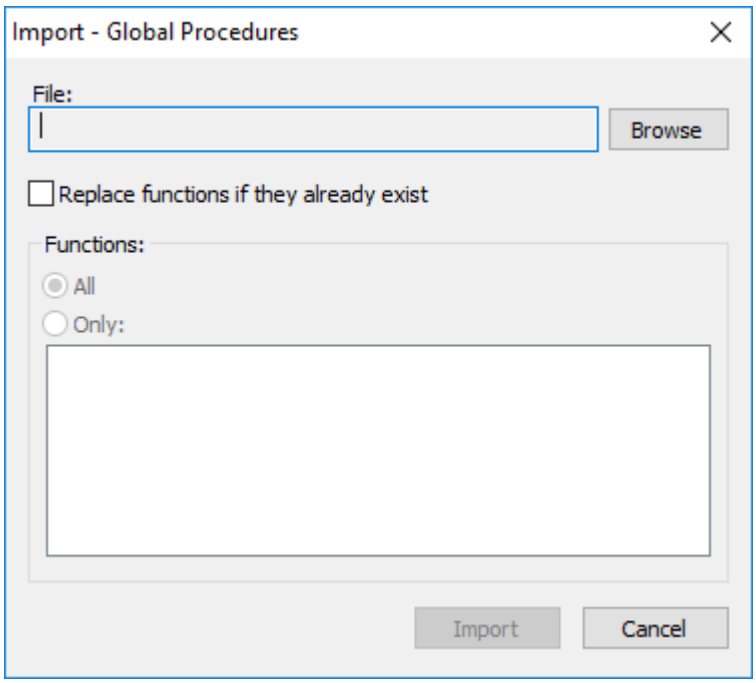
### Import functions from an external file

You can also import functions from an external file and add them to a Procedures worksheet. This is useful if you have a library of existing functions that you want to reuse.

To import functions into a worksheet:

1. Save and close all open screens and worksheets.
2. In the **Procedures** folder, right-click the Procedures worksheet into which you want to import functions, and then click **Import** on the shortcut menu.

   The *Import - Global Procedures* dialog box is displayed.




***Import – Global Procedures dialog box***

3. To the right of the **File** box, click **Browse**. A standard Windows file browser is displayed. Use it to locate and select a Procedures worksheet file. (This is a plain text file that has been saved with the **`.gis`** file extension.)
4. Select **Replace functions if they already exist** to overwrite functions in the worksheet with functions imported from the file, if the functions have the same names.
5. In the **Functions** area, do one of the following:
   - Click **All** to import all functions from the file; or
   - Click **Only** to import only selected functions from the files, and then select those functions in the list.
6. Click **Import**.

After the functions are imported, they should be displayed in the worksheet.

### Password protect a worksheet

You can put a password on any of your Procedures worksheets to prevent them from being edited or analyzed by other users. To protect a Procedures worksheet:

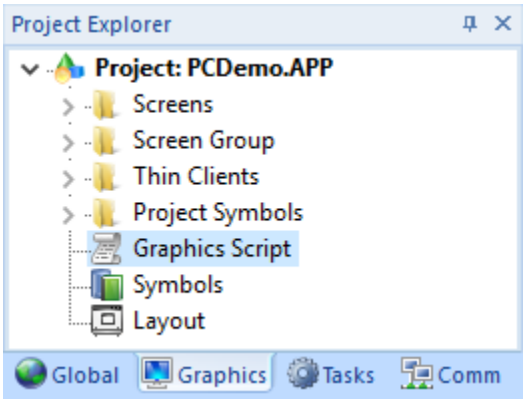
1. In the **Procedures** folder, right-click the worksheet and then click **Password Protection** on the shortcut menu. A *Password Protection* dialog is displayed.
2. In the **New Password** box, type your password.
3. In the **Confirm Password** box, type your password again.
4. Click **OK**.

Once this is done, you will be prompted for the password whenever you attempt to open the worksheet.

## *Graphic Module*

### GRAPHICS SCRIPT

The Graphics Script interface can be edited by its icon from the Graphics tab of the Project Explorer:




This interface can be used to execute logics on the following events, based on pre-defined sub-routines:

- **`Graphics_OnStart()`** : The code configured within this sub-routine is automatically executed just once when the graphic module is started. This interface is useful for initializing variables or executing logics that must be implemented when running the project.
- **`Graphics_WhileRunning()`** : The code configured within this sub-routine is automatically executed continuously while the graphic module is running. The rate in which this sub-routine is called depends on the performance of the platform where the project is running.
- **`Graphics_OnEnd()`** : The code configured within this sub-routine is automatically executed just once when the graphic module is closed.
- **`Graphics_OnScreenResize( width, height )`** : The code configured within this sub-routine is automatically executed just once when the screen resolution of the runtime station changes. The new width and height of the screen (in pixels) are passed to the sub-routine as parameters.

> **Note:** The **`Graphics_OnScreenResize()`** sub-routine is intended for projects running Windows Embedded devices that can switch between Portrait and Landscape display modes. It is *not* supported for projects running on Windows PC or in the Thin Client.

> **Important:** Do not change the name of the predefined sub-routines. If you do, then the system will not be able to automatically execute them.

Example:

```
'Variables with local scope can be declared and initialized here
Dim MyDate
MyDate = Date()
Dim MyValue
MyValue = 100

'Procedures with local scope can be implemented here
Function MyNewProcedure(nCount)
 MyNewProcedure = nCount + 1
End Function

Function AreaRec(side1, side2)
  AreaRec = side1 * side2
End Function

Sub CheckHiLimit(myValue, myHiLimit)
```

```
  If myValue > myHiLimit Then
    MsgBox("Value out of range")
  End If
End Sub

'This procedure is executed just once when the graphic module is started
Sub Graphics_OnStart()
  MsgBox("Welcome to the system!")
End Sub

'This procedure is executed continuously while the graphic module is running
Sub Graphics_WhileRunning()
  If $UserName = "Guest" Then
    $MyFlag = 0
  End If
End Sub

'This procedure is executed just once when the graphic module is closed
Sub Graphics_OnEnd()
  $LogOff()
End Sub
```

**When the Sub-routines Are Executed**

On the Server (i.e., where EcoStruxure Machine SCADA Expert 2020 is installed and running):

- The graphic module is the Viewer task.
- The **Graphics_OnStart()** sub-routine is executed once on the Server when the Viewer task is launched.
- The **Graphics_WhileRunning()** sub-routine keeps being executed on the Server while the Viewer task is running. The **Graphics_OnEnd()** sub-routine is executed once on the Server when the Viewer task is shut down.

On the Thin Client or Secure Viewer:

- The graphic module is the ISSymbol control.
- The **Graphics_OnStart()** sub-routine is executed once on the Thin Client station after logging in successfully.
- The **Graphics_WhileRunning()** sub-routine keeps being executed on the Thin Client station while the ISSymbol control is hosted by the Web Browser.
- The **Graphics_OnEnd()** sub-routine is executed once on the Thin Client station when the Web Browser is shut down (or when the ISSymbol control is no longer hosted by the Web Browser).

The execution of the Graphic Script sub-routines on the Server is completely independent of the execution on the Thin Client and Secure Viewer stations.

**Calling Graphics Script Procedures in Other VBScript Interfaces**

The three predefined sub-routines are strictly local to the Graphics Script interface and are executed only on the events described above. Other variables and procedures declared in the Graphics Script, however — under the headings **'Variables with local scope** and **'Procedures with local scope** — can be called in any other Screen Script or Command animation using the syntax **Graphics.*variable_name*** or **Graphics.*procedure_name***, respectively.

> **Note:** This feature is not supported on Mobile Access.

Taking the function **MyNewProcedure** that was declared in the example above, you could place a Button object on your project screen and then apply a Command animation to it with the following line:

```
$NewTag = Graphics.MyNewProcedure($OldTag)
```

### SCREEN SCRIPT

Each project screen has an associated screen script. To edit the screen script for a given project screen, open the screen worksheet for editing and then do one of the following:

- On the **Draw** tab of the ribbon, in the **Screen** group, click **Script**; or
- Right-click in the screen worksheet, and then click **Screen Script** on the shortcut menu.

This interface can be used to execute logics on the following events, based on preconfigured sub-routines:

- **`Screen_IsClosedByReplace()`**: This procedure determines whether the screen is automatically closed when another screen is opened to replace it. If the procedure is given a value of **`0`** or **`FALSE`**, then automatic closing is disabled. When the function is given a positive value (e.g., **`1`**) or **`TRUE`**, or if the procedure is not declared at all, then automatic closing is enabled.
- **`Screen_OnOpen()`**: The code configured within this sub-routine is automatically executed just once when the screen is opened.
- **`Screen_WhileOpen()`**: The code configured within this sub-routine is automatically executed continuously while its screen is open. The rate in which this sub-routine is called depends on the performance of the platform where the project is running.
- **`Screen_OnClose()`**:The code configured within this sub-routine is automatically executed just once when the screen is closed.

The variables and procedures declared in this interface are available for the VBScript interfaces of the screen where the Screen Script is configured.

> **Note:** Do not change the names of the preconfigured sub-routines described above. If you do, then the system will not be able to call them.

> **Note:**
>
> - The execution of the Screen Script sub-routines on the server is totally independent of the execution on the Thin Client stations. In other words, these sub-routines are executed asynchronously.
> - The procedures and/or variables declared in the Screen Script interface have local scope. They can be called only from the specific screen on which they are declared.

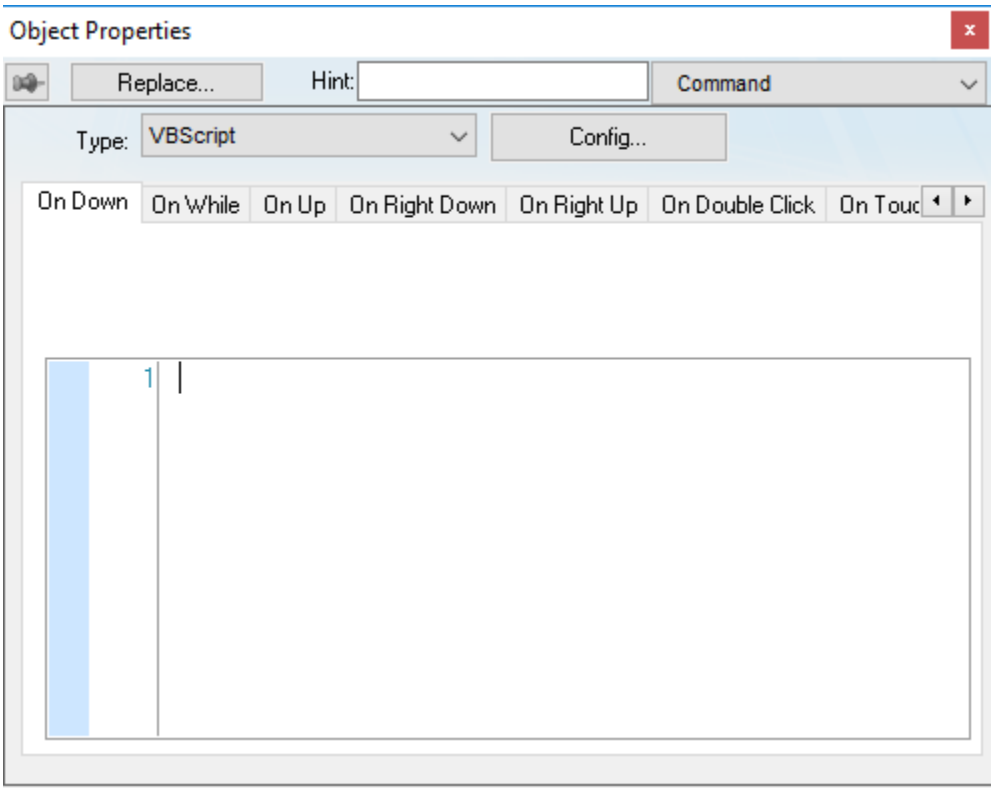
Example:

```
'Variables available on this screen can be declared and initialized here
Dim Counter

'Procedures available on this screen can be implemented here
Function AreaCircle(radius)
  AreaCircle = Sqr(radius) * $Pi()
End Function

Sub CheckLoLimit (myValue, myLoLimit)
  If myValue < myLoLimit Then
    MsgBox("Value out of range")
  End If
End Sub

'This procedure determines whether the screen is automatically closed
Function Screen_IsClosedByReplace()
  Screen_IsClosedByReplace = $ReplaceModeTag
End Function

'This procedure is executed just once when this screen is open
Sub Screen_OnOpen()
  MsgBox("The screen was open!")
End Sub

'This procedure is executed continuously while this screen is open
```

```
Sub Screen_WhileOpen()
  If Counter < 100 Then
    Counter = Counter + 1
  Else
    Counter = 0
  End If
  $SimulationTag = Counter
End Sub

'This procedure is executed just once when this screen is closed
Sub Screen_OnClose()
  MsgBox("The screen will be closed!")
End Sub
```

## COMMAND ANIMATION

On the **Graphics** tab, in the **Animations** group, click **Command** to add the animation to a selected object or group of objects. The animation enables you to click on the object or press a pre-defined key to execute the command at runtime. Double-click on the object to view its object properties.




***Object Properties: Command***

The Command animation provides one tag for each one of the events supported by it. Notice that more than one event can be configured simultaneously for the same Command animation:

- **On Down**: Executes the command/script once when the user clicks on the object with the left mouse button.
- **On While**: Keeps executing the command/script continuously while the mouse pointer is pressed on the object. The period (in milliseconds) of execution for the command/script is set in the Rate field from the Configuration dialog screen, except for the VBScript option, which is executed as fast as possible.
- **On Up**: Executes the command/script once when the user releases the left mouse button on the object.
- **On Right Down**: Executes the command/script once when the user clicks on the object with the right mouse button.

- **On Right Up**: Executes the command/script once when the user releases the right mouse button on the object.
- **On Double Click**: Executes the command/script once when the user double-clicks on the object with the left mouse button.
- **On Touch**, **On Touch Start**, **On Touch Delta**, **On Touch Complete**: These events are used for multi-touch gestures. For more information, see About Touch Events on page 364.

> **Note:** The events **On Right Down**, **On Right Up** and **On Double Click** are not supported in projects running on Windows Embedded target systems.

> **Tip:** An asterisk (*) on an event tab indicates that something is configured for that event. This makes it easier to see at a glance which events are configured.

**Type** menu: This setting defines the type of action that must be executed by the event of the Command animation. Notice that each event has its own type. Therefore, the same Command animation can be configured with different types of action for different events. The following types are supported:

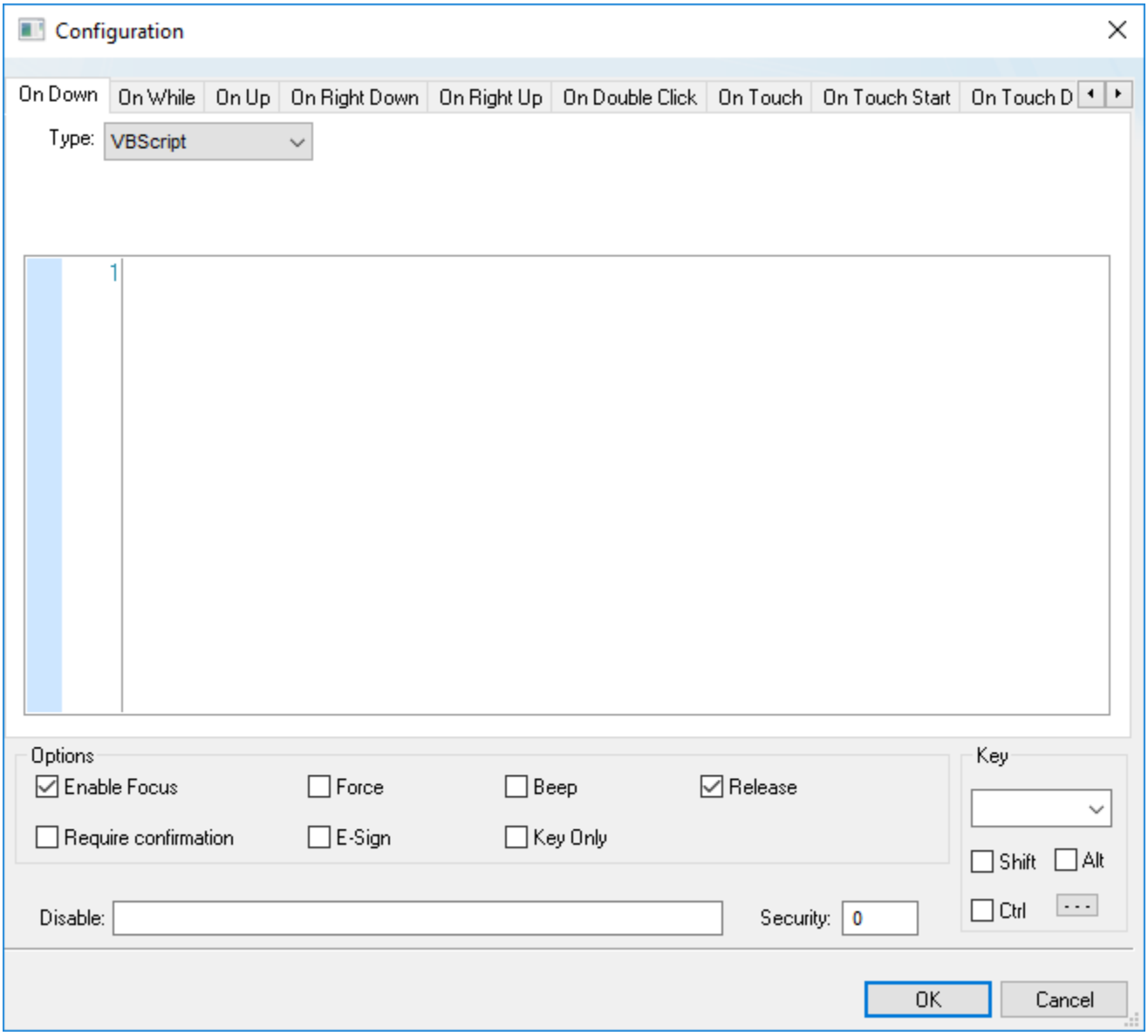
| Type | Description |
|---|---|
| **Built#in Language** | Allows you to configure a script using the EcoStruxure Machine SCADA Expert built-in language. When this type is selected, the user can configure up to 12 expressions for each event in the **Expression** column. The expressions are executed sequentially from the first row until the last one when the event is triggered. The result of each expression is written to the tag configured in the **Tag** column (if any). Consult the **Built-in Scripting Language** chapter for more information. |
| **VBScript** | Allows you to configure a script using the standard VBScript language. When this type is selected, the user can configure a script in the VBScript editor for the **Command** animation. Consult the **VBScript** chapter for further information about the VBScript language. |
| **Open Screen** | Allows you to configure the **Command** animation to open a specific screen when the event is triggered during runtime. This type is equivalent to the `Open` function. You can either type the screen name in the **Open Screen** field or browse it. Furthermore, you can type a string tag between curly brackets {TagName} in this field. When the event is executed, the project will attempt to open the named screen.<br>**Note:** The screen file extension (either `*.scc` or `*.scr`) is assumed, so you do not need to include it. However, if you have two screen files with the same name but different extensions in your project folder (e.g., `MyScreen.scc` and `MyScreen.scr`), the one with the preferred extension — as determined by whether the **Use .scr extension for screen files** option in the project settings is selected — will be opened. For more information, see Viewer tab on page 117. |
| **Close Screen** | Allows you to configure the **Command** animation to close a specific screen when the event is triggered during runtime. This type is equivalent to the `Close` function. You can either type the screen name in the **Close Screen** field or browse it. You can also type a string tag between curly brackets {TagName} in this field. When the event is executed, the project will attempt to close the named screen. |
| **Set Tag** | Allows you to configure the **Command** animation to set a tag when the event is triggered during runtime. You can either type the tag name in the **Set Tag** field or browse it. When the event is executed, the project will write the value 1 to the tag configured in this field. |
| **Reset Tag** | Allows you to configure the **Command** animation to reset a tag when the event is triggered during runtime. You can either type the tag name in the **Reset Tag** field or browse it. When the event is executed, the project will write the value 0 to the tag configured in this field. |
| **Toggle Tag** | Allows you to configure the **Command** animation to toggle a tag when the event is triggered during runtime. You can either type the tag name in the **Toggle Tag** field or browse it. When the event is executed, the project will toggle the value of the tag configured in this field. |

**Config** button: Launches the *Configuration* dialog, where the Command animation can be fully configured.

**Back to button**: Click to go back to the object properties of the underlying Button object.

### Configuration dialog

This dialog allows you to fully configure the Command animation…




***Configuration dialog***

The event tabs (e.g., On Down, On While, etc.) and the **Type** menu are the same as in the *Object Properties* dialog described above. The remaining settings are shared for all events:

- *Options* pane:
  - **Enable Focus** checkbox: When this option is checked, the object that the Command animation was applied to can receive the focus during runtime by the navigation keys.
  - **Force** checkbox: When this option is selected, any project tag that receives a value will trigger events as if the tag changed, even if the new value is equal to the old value. For example, if a tag has a value of 0 and the Command animation runs a procedure that writes 0 to that tag, all other tasks in the project runtime will recognize that the tag changed, even though it did not. This option is useful for making sure that events triggered by tag changes (e.g., **Write on Tag Change** on a communication driver) are always triggered when the Command animation is used.

Please keep in mind that if the tag's value does not actually change, the tag's timestamp (***tagname*->Timestamp**) is not updated either.

**Force** applies to both the procedure run by the Command animation itself and any global procedures called in that procedure, as long as they are run on the project runtime client where the Command animation is used (i.e., on the device where the button is pushed).

**Force** does not apply to global procedures that are run on the project runtime server using the function `RunGlobalProcedureOnServer`, even if the function is called in the procedure run by the Command animation. If you want to force tag changes in global procedures run on the server, use the function `ForceTagChange`.

- **Beep** checkbox: When this option is checked, a short beep is played when the Command is executed. This option is useful to provide an audio feed-back to the user, indicating that the Command was executed. It does not indicate, however, if the action triggered by the Command animation was successful or not.
- **Release** checkbox: When this option is checked, the On Up event is executed when you drag the cursor (or your finger) out of the object area (whether the button was released or not). This option is useful to make sure that the On Up event will always be executed after an On Down event, even if the user releases the mouse cursor out of the object area before releasing it.
- **Confirm** checkbox: When this option is checked, user will have to answer a confirmation question before executing the command. This option is useful for decreasing the accidental triggering of critical events during runtime.
- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the command.
- **Key Only** checkbox: When this option is checked, the user can *only* use the keyboard shortcut (configured in the *Key* pane described below) to execute commands.

- **Disable**: Disables action by the user when the result of the expression configured in this field is TRUE (value different from 0).
- **Security**: Security access level required to use the Command animation.
- *Key* group: Shortcut used to trigger the events On Down, While Down and On Up using a keyboard. (In other words, pressing this keyboard shortcut is the same as clicking the left mouse button.) This option is especially useful when creating projects for runtime devices that do not provide a mouse or touch-screen interface — the keyboard is the only physical interface available to interact with your project during runtime.
  - **Shift**, **Ctrl**, or **Alt** boxes: Click to create a key combination key, meaning the Shift, Ctrl and/or Alt key must be pressed with the key specified in the drop-down list.
  - Click the browse button (...) to open the *Key Modifer* dialog, which enables you to modify your combination keys. You can choose **Left**, **Right** or **Left or Right** to specify the position on the keyboard of the Shift, Ctrl or Alt key in the key combination. If you choose **Left or Right**, the command will be executed any time either of these keys is pressed in combination with the key specified in the drop-down list.

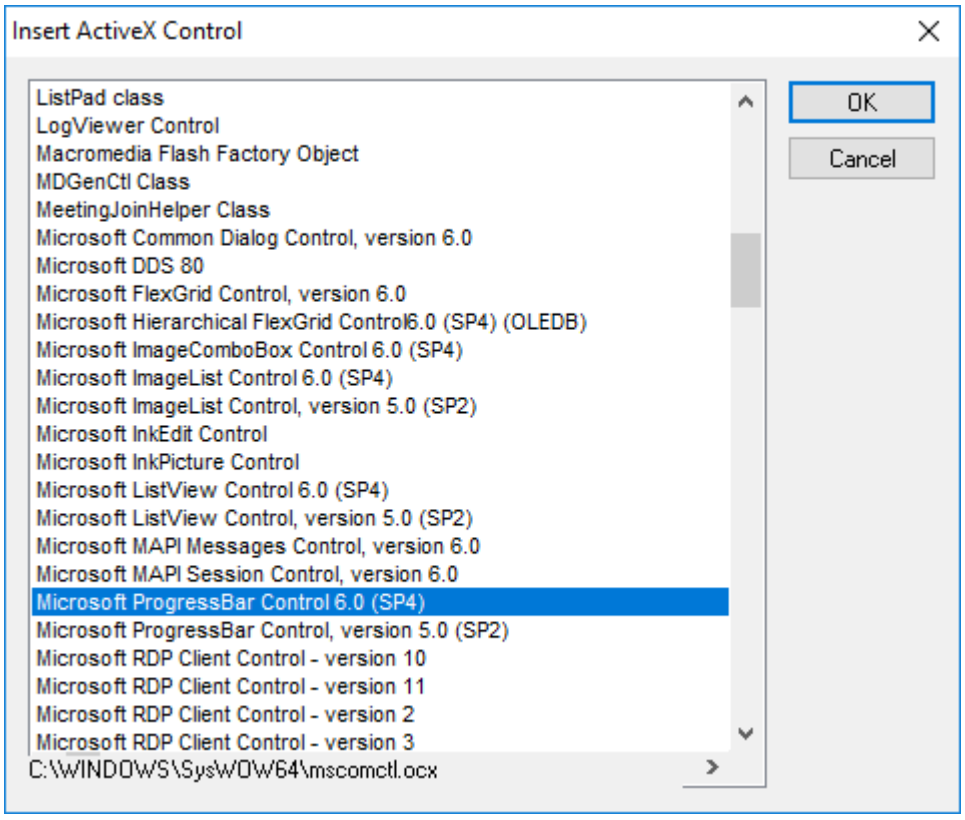
> **Tip:** If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects.

## ACTIVEX EVENTS

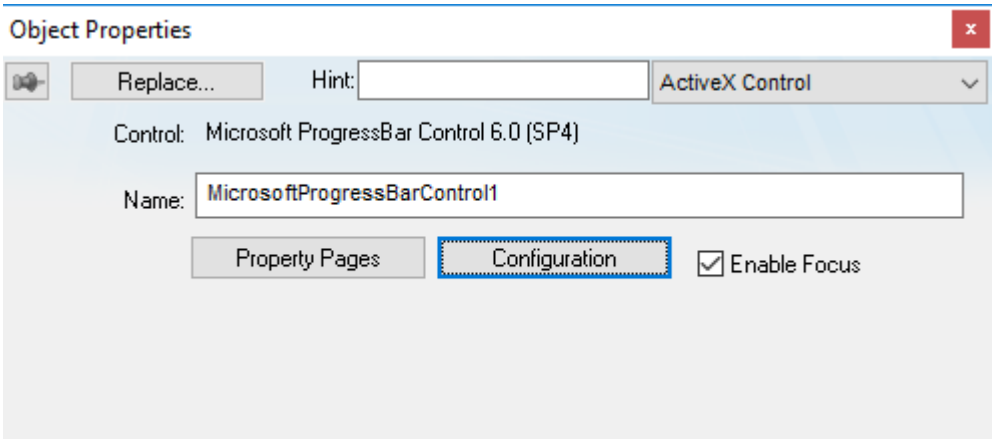
To edit the ActiveX Events interface, select the Script option from the Events tab of the ActiveX object inserted on the screen.

1. Click the ActiveX Control icon in the Active Objects toolbar.

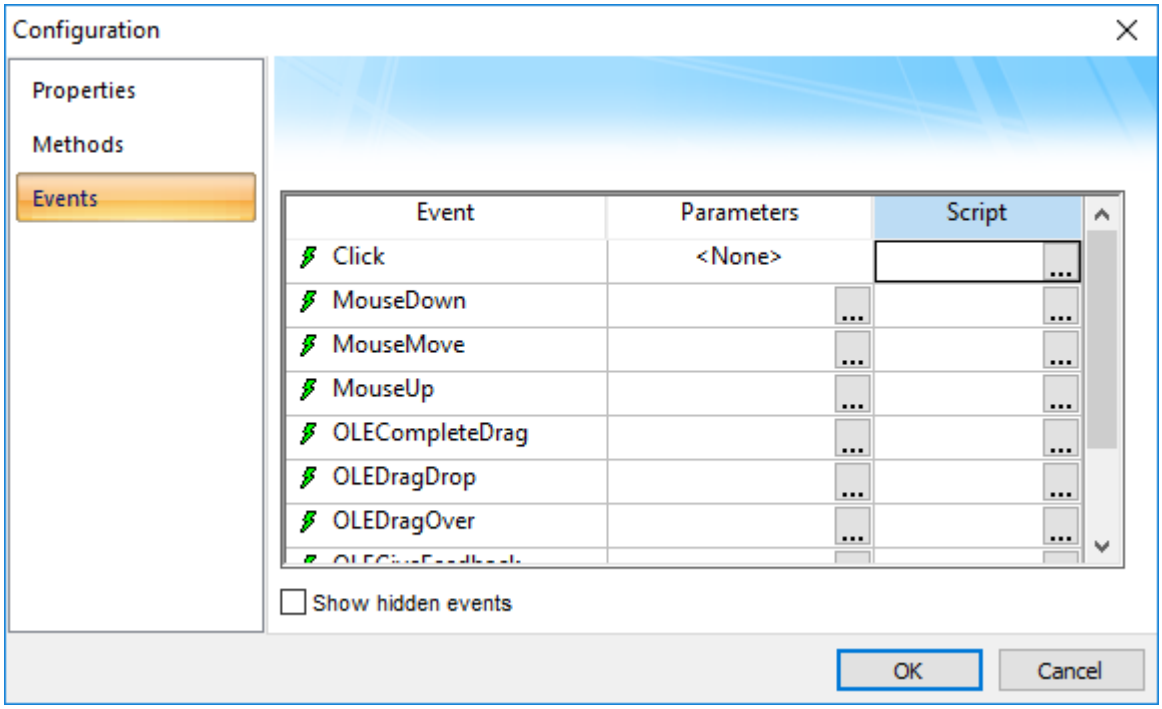
The *Insert ActiveX Control* dialog opens.



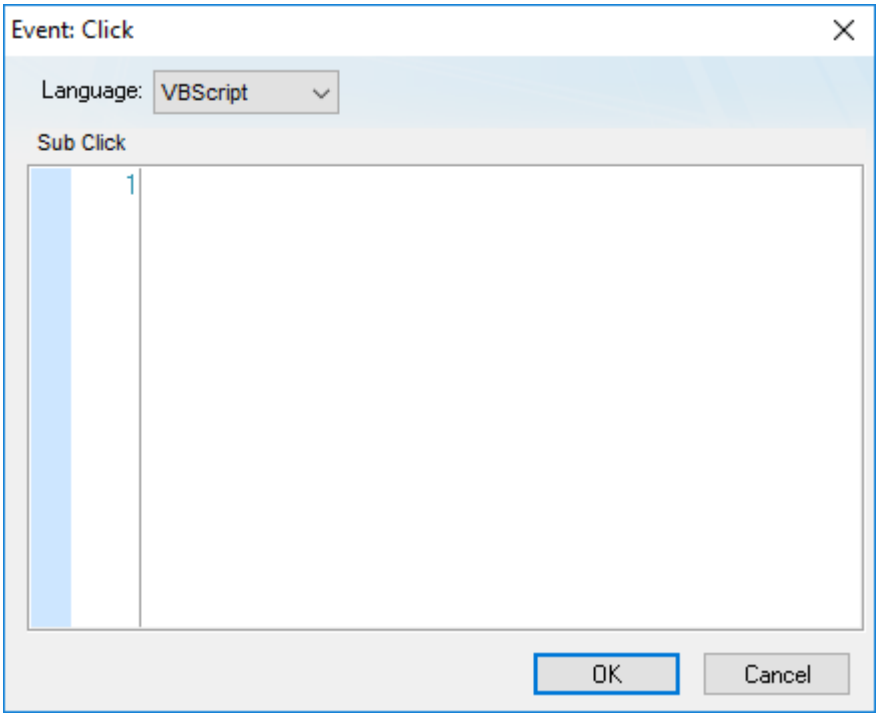

2. Select the ActiveX Control that you wish to use and then click **OK**.
3. The object that symbolizes the selected ActiveX Control will display. Right-click on this object to open the *Object Properties* dialog.



4. Click the **Configuration** button. The *Configuration* dialog will open. Click the Events tab.




5. Click the ... button in the Script column.




Use this interface to execute logics when an ActiveX object triggers an event.

Variables declared in this interface are available for this interface only (local scope). In other words, they are not available for any other object in the project.

You cannot implement procedures in this interface. You can, however, call procedures implemented in the Global Procedures or in the Screen Script interface for the same screen where the ActiveX object is configured.

> **Note:** For more information, see ActiveX Control object.

Example:

```
'The script below will be executed when the Calendar Control ActiveX
'triggers its "AfterUpdate" event
$MyYear = CalendarControl1.Year
$MyMonth = CalendarControl1.Month
$MyDay = CalendarControl1.Day
```

## *Background Task*

### SCRIPT WORKSHEET

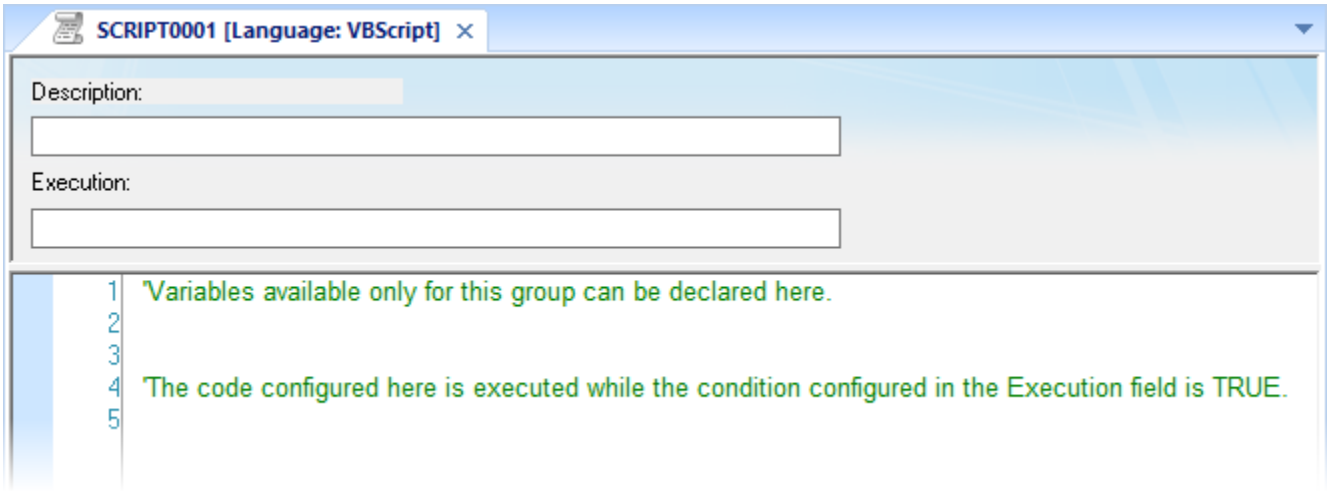
A Script worksheet is used to implement program logic (using VBScript) that should be continuously executed during runtime, rather than on specific actions like the user pressing a button on a screen.

> **Note:** The Script worksheet is functionally similar to the Math worksheet, except that it uses VBScript instead of the Built-in Scripting Language.

To create a new Script worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Script**;
- Right-click the **Script** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the File menu, click the **File** tab, and then select **Script Worksheet**.

To edit an existing Script worksheet, double-click it in the Project Explorer.




***Script worksheet***

The code configured in each Script worksheet is executed by the Background Task. The project scans the worksheets sequentially (based on the worksheet number) and executes only the groups in which the condition configured in the **Execution** field of the worksheet is TRUE (i.e., non-zero).

> **Note:** You must use the syntax supported by the Built-in Scripting Language in the **Execution** field. Only the body of the worksheet supports VBScript.

Variables declared in the worksheet have local scope for that specific group only. They are not available for any other VBScript interface.

You cannot define procedures (i.e., functions and subs) in the Script worksheet. However, you can call procedures defined in the Global Procedures or in the Startup Script.

Example:

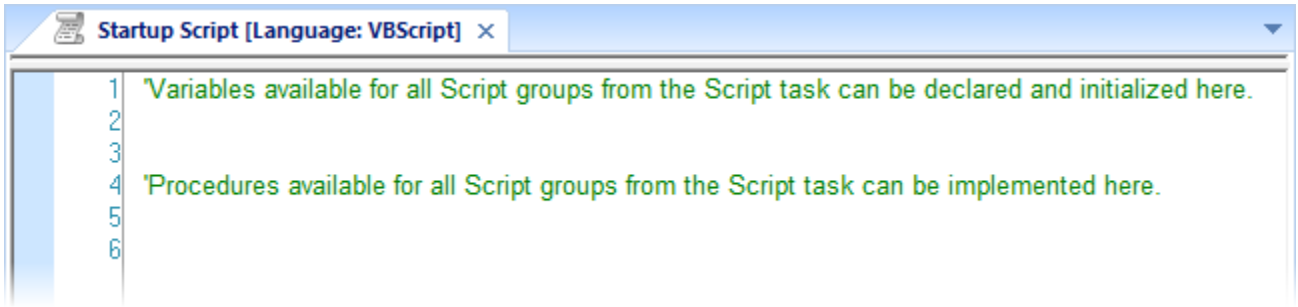
```
'Variables available only for this group can be declared here
Dim myVar, myTest
myTest = 1

'The code configured here is executed while the condition configured in the Execution
 field is TRUE
myVar = $FindFile("c:\*.txt")
If MyVar > 0 Then
  $TagNumOfFiles = myVar
End If
```

> **Note:** When any Script worksheet is saved during runtime (on-line configuration), the Startup Script will be executed again and the current value of the local variables of any Script worksheet will be reset.

## STARTUP SCRIPT WORKSHEET

The Startup Script worksheet is a VBScript interface that is automatically executed when the project is run.

To edit the Startup Script worksheet, double-click it in the Project Explorer. (It is located on the Tasks tab, in the Script folder.) The worksheet is displayed:




***Startup Script worksheet***

The code configured in this worksheet is executed just once when the Background Task module (BGTask) is started. This interface is useful for initializing variables or executing logics that must be implemented when the project is run.

You can declare and initialize variables and define procedures. However, variables or procedures declared in this interface will be available ONLY to the Script worksheets executed by the Background Task module — they are not available to any VBScript interface from the Graphic Module.

Example:

```
'Variables available for all Script groups from the Script task can be declared and
 initialized here
Dim MyVar, Counter
MyVar = 100

'Procedures available for all Script groups from the Script task can be implemented
 here

Function AreaEquTriangle(base, high)
  AreaEquTriangle = (base * high) / 2
```

```
End Function

Sub CheckLimits(myValue, myHiLimit, myLoLimit)
  If (myValue > myHiLimit Or myValue < myLoLimit) Then
    MsgBox("Value out of range")
  End If
End Sub

'The code configured here is executed just once when the Background task is started
If $GetOS() = 3 Then
  MsgBox ("Welcome! This project is running under Microsoft Windows Embedded operating
 system.")
Else
  MsgBox("Welcome! This project Is running under Microsoft Windows desktop operating
 system.")
End If
```

# Language Reference

## *Operators*

### Arithmetic Operators

| Symbol | Name | Description |
|---|---|---|
| `^` | Exponentiation | Raises a number to the power of an exponent. |
| `-` | Unary negation | Finds the difference between two numbers or indicates the negative value of a numeric expression. |
| `*` | Multiplication | Multiplies two numbers. |
| `/` | Division | Divides two numbers and returns a floating-point result. |
| `\` | Integer division | Divides two numbers and returns an integer result. |
| `Mod` | Modulus arithmetic | Divides two numbers and returns only the remainder. |
| `+` | Addition | Finds the sum of two numbers. |
| `-` | Subtraction | Finds the difference between two numbers or indicates the negative value of a numeric expression. |
| `&` | String concatenation | Forces string concatenation of two expressions. |

### Comparison Operators

| Symbol | Name | Description |
|---|---|---|
| `=` | Equality | Comparison is True if the first expression is equal to the second expression. |
| `<>` | Inequality | Comparison is True if the first expression is different from the second expression. |
| `<` | Less than | Comparison is True if the first expression is less than the second expression. |
| `>` | Greater than | Comparison is True if the first expression is greater than the second expression. |
| `<=` | Less than or equal to | Comparison is True if the first expression is less than or equal to the second expression. |
| `>=` | Greater than or equal to | Comparison is True if the first expression is greater than or equal to the second expression. |
| `Is` | Object equivalence | Compares two object reference variables. Comparison is True if both object names refer to the same object. |

### Logical Operators

| Symbol | Name | Description |
|---|---|---|
| `Not` | Logical negation | Performs logical negation on an expression. |
| `And` | Logical conjunction | Performs a logical conjunction on two expressions. |
| `Or` | Logical disjunction | Performs a logical disjunction on two expressions. |
| `Xor` | Logical exclusion | Performs a logical exclusion on two expressions. |
| `Eqv` | Logical equivalence | Performs a logical equivalence on two expressions. |
| `Imp` | Logical implication | Performs a logical implication on two expressions. |

### Assignment Operators

| Symbol | Name | Description |
|---|---|---|
| `=` | Assignment | Assigns a value to a variable or property. |

## *Constants*

### Color Constants

| Constant | Value | Description |
|---|---|---|
| **vbBlack** | &h00 | Black |
| **vbRed** | &hFF | Red |
| **vbGreen** | &hFF00 | Green |
| **vbYellow** | &hFFFF | Yellow |
| **vbBlue** | &hFF0000 | Blue |
| **vbMagenta** | &hFF00FF | Magenta |
| **vbCyan** | &hFFFF00 | Cyan |
| **vbWhite** | &hFFFFFF | White |

### Comparison Constants

| Constant | Value | Description |
|---|---|---|
| **vbBinaryCompare** | 0 | Perform a binary comparison |
| **vbTextCompare** | 1 | Perform a textual comparison |

### Date & Time Constants

| Constant | Value | Description |
|---|---|---|
| **vbSunday** | 1 | Sunday |
| **vbMonday** | 2 | Monday |
| **vbTuesday** | 3 | Tuesday |
| **vbWednesday** | 4 | Wednesday |
| **vbThursday** | 5 | Thursday |
| **vbFriday** | 6 | Friday |
| **vbSaturday** | 7 | Saturday |
| **vbUseSystemDayOfWeek** | 0 | Use the day of the week specified in your system settings for the first day of the week. |
| **vbFirstJan1** | 1 | Use the week in which January 1 occurs (default). |
| **vbFirstFourDays** | 2 | Use the first week that has at least four days in the new year. |
| **vbFirstFullWeek** | 3 | Use the first full week of the year. |

### Date Format Constants

| Constant | Value | Description |
|---|---|---|
| **vbGeneralDate** | 0 | Display a date and/or time. For real numbers, display a date and time. If there is no fractional part, display only a date. If there is no integer part, display time only. Date and time display is determined by your system settings. |
| **vbLongDate** | 1 | Display a date using the long date format specified in your computer's regional settings. |
| **vbShortDate** | 2 | Display a date using the short date format specified in your computer's regional settings. |
| **vbLongTime** | 3 | Display a time using the long time format specified in your computer's regional settings. |
| **vbShortTime** | 4 | Display a time using the short time format specified in your computer's regional settings. |

### Miscellaneous Constants

| Constant | Value | Description |
|---|---|---|
| vbObjectError | -2147221504 | User-defined error numbers should be greater than this value. |

### Box Constants – Buttons & Icons

| Constant | Value | Description |
|---|---|---|
| vbOKOnly | 0 | Display OK button only. |
| vbOKCancel | 1 | Display OK and Cancel buttons. |
| vbAbortRetryIgnore | 2 | Display Abort, Retry, and Ignore buttons. |
| vbYesNoCancel | 3 | Display Yes, No, and Cancel buttons. |
| vbYesNo | 4 | Display Yes and No buttons. |
| vbRetryCancel | 5 | Display Retry and Cancel buttons. |
| vbCritical | 16 | Display Critical Message icon. |
| vbQuestion | 32 | Display Warning Query icon. |
| vbExclamation | 48 | Display Warning Message icon. |
| vbInformation | 64 | Display Information Message icon. |
| vbDefaultButton1 | 0 | First button is the default. |
| vbDefaultButton2 | 256 | Second button is the default. |
| vbDefaultButton3 | 512 | Third button is the default. |
| vbDefaultButton4 | 768 | Fourth button is the default. |
| vbApplicationModal | 0 | Application modal. The user must respond to the message box before continuing work in the current application. |
| vbSystemModal | 4096 | System modal. On Win16 systems, all programs are suspended until the user responds to the message box. On Win32 systems, this constant provides a program modal message box that always remains on top of any other programs you may have running. |

### Box Constants – Selected Button

| Constant | Value | Description |
|---|---|---|
| vbOK | 1 | OK button was clicked. |
| vbCancel | 2 | Cancel button was clicked. |
| vbAbort | 3 | Abort button was clicked. |
| vbRetry | 4 | Retry button was clicked. |
| vbIgnore | 5 | Ignore button was clicked. |
| vbYes | 6 | Yes button was clicked. |
| vbNo | 7 | No button was clicked. |

### String Constants

| Constant | Value | Description |
|---|---|---|
| vbCr | Chr(13) | Carriage return |
| VbCrLf | Chr(13) & Chr(10) | Carriage return…linefeed combination |
| vbFormFeed | Chr(12) | Form feed; not useful in Microsoft Windows |

| Constant | Value | Description |
|---|---|---|
| **vbLf** | Chr(10) | Line feed |
| **vbNewLine** | Chr(13) & Chr(10) or Chr(10) | Platform-specific newline character; whatever is appropriate for the platform |
| **vbNullChar** | Chr(0) | Character having the value 0 |
| **vbNullString** | String having value 0 | Not the same as a zero-length string (""); used for calling external procedures |
| **vbTab** | Chr(9) | Horizontal tab |
| **vbVerticalTab** | Chr(11) | Vertical tab; not useful in Microsoft Windows |

### Tristate Constants

| Constant | Value | Description |
|---|---|---|
| **vbUseDefault** | −2 | Use default from computer's regional settings. |
| **vbTrue** | −1 | TRUE |
| **vbFalse** | 0 | FALSE |

### VarType Constants

| Constant | Value | Description |
|---|---|---|
| **vbEmpty** | 0 | Uninitialized (default) |
| **vbNull** | 1 | Contains no valid data |
| **vbInteger** | 2 | Integer subtype |
| **vbLong** | 3 | Long subtype |
| **vbSingle** | 4 | Single subtype |
| **vbDouble** | 5 | Double subtype |
| **vbCurrency** | 6 | Currency subtype |
| **vbDate** | 7 | Date subtype |
| **vbString** | 8 | String subtype |
| **vbObject** | 9 | Object |
| **vbError** | 10 | Error subtype |
| **vbBoolean** | 11 | Boolean subtype |
| **vbVariant** | 12 | Variant (used only for arrays of variants) |
| **vbDataObject** | 13 | Data access object |
| **vbDecimal** | 14 | Decimal subtype |
| **vbByte** | 17 | Byte subtype |
| **vbArray** | 8192 | Array |

## *Objects and Collections*

**Class Object**

The object created using the Class statement. Provides access to the events of the class.

**Debug Object**

An intrinsic global object that can send output to a script debugger, such as the Microsoft Script Debugger.

**Err Object**

Contains information about runtime errors. Accepts the Raise and Clear methods for generating and clearing runtime errors.

**Match Object**

Provides access to the read-only properties of a regular expression match.

**Matches Collection**

Collection of regular expression Match objects.

**Regular Expression (RegExp) Object**

Provides simple regular expression support.

**SubMatches Collection**

Collection of regular expression submatch strings.

### *Properties*

**Description**

Returns or sets a descriptive string associated with an error.

**FirstIndex**

Returns the position in a search string where a match occurs.

**Global**

Sets or returns a Boolean value that indicates if a pattern should match all occurrences in an entire search string or just the first one.

**HelpContext**

Sets or returns a context ID for a topic in a Help File.

**HelpFile**

Sets or returns a fully qualified path to a Help File.

**IgnoreCase**

Sets or returns a Boolean value that indicates if a pattern search is case-sensitive or not.

**Length**

Sets or returns a Boolean value that indicates if a pattern search is case-sensitive or not.

**Number**

Returns or sets a numeric value specifying an error. Number is the Err object's default property.

**Pattern**

Sets or returns the regular expression pattern being searched for.

**Source**

Returns or sets the name of the object or application that originally generated the error.

**Value**

Returns the value or text of a match found in a search string.

### *Statements*

**Call**

Transfers control to a Sub or Function procedure.

**Class**

Declares the name of a class, as well as a definition of the variables, properties, and methods that comprise the class.

**Const**

Declares constants for use in place of literal values.

**Dim**

Declares variables and allocates storage space.

**Do...Loop**

Repeats a block of statements while a condition is True or until a condition becomes True.

**Erase**

Reinitializes the elements of fixed-size arrays and deallocates dynamic-array storage space.

**Execute**

Executes one or more specified statements.

**ExecuteGlobal**

Executes one or more specified statements in the global namespace of a script.

**Exit**

Exits a block of Do…Loop, For…Next, Function, or Sub code.

**For Each...Next**

Repeats a group of statements for each element in an array or collection.

**For...Next**

Repeats a group of statements a specified number of times.

**Function**

Declares the name, arguments, and code that form the body of a Function procedure.

**If...Then...Else**

Conditionally executes a group of statements, depending on the value of an expression.

**Option Explicit**

Forces explicit declaration of all variables in a script.

**Private**

Declares private variables and allocates storage space. Declares, in a Class block, a private variable.

**Property Get**

Declares, in a Class block, the name, arguments, and code that form the body of a Property procedure that gets (returns) the value of a property.

**Property Let**

Declares, in a Class block, the name, arguments, and code that form the body of a Property procedure that assigns (sets) the value of a property.

**Property Set**

Declares, in a Class block, the name, arguments, and code that form the body of a Property procedure that sets a reference to an object.

**Public**

Declares public variables and allocates storage space. Declares, in a Class block, a private variable.

**Ramdomize**

Initializes the random-number generator.

**ReDim**

Declares dynamic-array variables, and allocates or reallocates storage space at procedure level.

**Rem**

Includes explanatory remarks in a program.

**Select**

Executes one of several groups of statements, depending on the value of an expression.

**Set**

Assigns an object reference to a variable or property, or associates a procedure reference with an event.

**Stop**

Suspends execution.

**Sub**

Declares the name, arguments, and code that form the body of a Sub procedure.

**While**

Executes a series of statements as long as a given condition is True.

**With**

Executes a series of statements on a single object.

## *Methods*

**Clear**

Clears all property settings of the Err object.

**Execute**

Executes a regular expression search against a specified string.

**Raise**

Generates a runtime error.

**Replace**

Replaces text found in a regular expression search.

**Test**

Executes a regular expression search against a specified string and returns a Boolean value that indicates if a pattern match was found.

**Write**

Sends strings to the script debugger.

**WriteLine**

Sends strings to the script debugger, followed by a newline character.

## *Functions*

| Function Names | | | |
|---|---|---|---|
| Abs | Array | Asc | Atn |
| CBool | CByte | CCur | CDate |
| CDbl | Chr | CInt | CLng |
| Conversions | Cos | CreateObject | CSng |
| CStr | Date | DateAdd | DateDiff |
| DatePart | DateSerial | DateValue | Day |
| Derived Math | Escape | Eval | Exp |
| Filter | FormatCurrency | FormatDateTime | FormatNumber |
| FormatPercent | GetLocale | GetObject | GetRef |
| Hex | Hour | InputBox | InStr |
| InStrRev | Int, Fix | IsArray | IsDate |
| IsEmpty | IsNull | IsNumeric | IsObject |
| Join | LBound | LCase | Left |
| Len | LoadPicture | Log | LTrim; RTrim; and Trim |
| Maths | Mid | Minute | Month |

| Function Names | | | |
|---|---|---|---|
| MonthName | MsgBox | Now | Oct |
| Replace | RGB | Right | Rnd |
| Round | ScriptEngine | ScriptEngineBuildVersion | ScriptEngineMajorVersion |
| ScriptEngineMinorVersion | Second | SetLocale | Sgn |
| Sin | Space | Split | Sqr |
| StrComp | String | StrReverse | Tan |
| Time | Timer | TimeSerial | TimeValue |
| TypeName | UBound | UCase | Unescape |
| VarType | Weekday | WeekdayName | Year |

## *Keywords*

**Empty**

The Empty keyword is used to indicate an uninitialized variable value. This is not the same thing as Null.

**False**

The False keyword has a value equal to 0.

**Nothing**

The Nothing keyword in VBScript is used to disassociate an object variable from any actual object.

**Null**

The Null keyword is used to indicate that a variable contains no valid data. This is not the same thing as Empty.

**True**

The True keyword has a value equal to -1.

## *Errors*

### VBScript Runtime Errors

| Error Number | Description |
|---|---|
| 5 | Invalid procedure call or argument |
| 6 | Overflow |
| 7 | Out of memory |
| 9 | Subscript out of range |
| 10 | This array is fixed or temporarily locked |
| 11 | Division by zero |
| 13 | Type mismatch |
| 14 | Out of string space |
| 17 | Can't perform requested operation |
| 28 | Out of stack space |
| 35 | Sub or function not defined |
| 48 | Error in loading DLL |
| 51 | Internal error |

| Error Number | Description |
|---|---|
| 91 | Object variable not set |
| 92 | For loop not initialized |
| 94 | Invalid use of Null |
| 424 | Object required |
| 429 | ActiveX component can't create object |
| 430 | Class doesn't support Automation |
| 432 | File name or class name not found during Automation operation |
| 438 | Object doesn't support this property or method |
| 445 | Object doesn't support this action |
| 447 | Object doesn't support current locale setting |
| 448 | Named argument not found |
| 449 | Argument not optional |
| 450 | Wrong number of arguments or invalid property assignment |
| 451 | Object not a collection |
| 458 | Variable uses an Automation type not supported in VBScript |
| 462 | The remote server machine does not exist or is unavailable |
| 481 | Invalid picture |
| 500 | Variable is undefined |
| 502 | Object not safe for scripting |
| 503 | Object not safe for initializing |
| 504 | Object not safe for creating |
| 505 | Invalid or unqualified reference |
| 506 | Class not defined |
| 507 | An exception occurred |
| 5008 | Illegal assignment |
| 5017 | Syntax error in regular expression |
| 5018 | Unexpected quantifier |
| 5019 | Expected ']' in regular expression |
| 5020 | Expected ')' in regular expression |
| 5021 | Invalid range in character set |

### VBScript Syntax Errors

| Error Number | Description |
|---|---|
| 1001 | Out of memory |
| 1002 | Syntax error |
| 1005 | Expected '(' |
| 1006 | Expected ')' |
| 1010 | Expected identifier |
| 1011 | Expected '=' |

| Error Number | Description |
|---|---|
| 1012 | Expected 'If' |
| 1013 | Expected 'To' |
| 1014 | Expected 'End' |
| 1015 | Expected 'Function' |
| 1016 | Expected 'Sub' |
| 1017 | Expected 'Then' |
| 1018 | Expected 'Wend' |
| 1019 | Expected 'Loop' |
| 1020 | Expected 'Next' |
| 1021 | Expected 'Case' |
| 1022 | Expected 'Select' |
| 1023 | Expected expression |
| 1024 | Expected statement |
| 1025 | Expected end of statement |
| 1026 | Expected integer constant |
| 1027 | Expected 'While' or 'Until' |
| 1028 | Expected 'While,' 'Until,' or end of statement |
| 1029 | Expected 'With' |
| 1030 | Identifier too long |
| 1037 | Invalid use of 'Me' keyword |
| 1038 | 'loop' without 'do' |
| 1039 | Invalid 'exit' statement |
| 1040 | Invalid 'for' loop control variable |
| 1041 | Name redefined |
| 1042 | Must be first statement on the line |
| 1044 | Cannot use parentheses when calling a Sub |
| 1045 | Expected literal constant |
| 1046 | Expected 'In' |
| 1047 | Expected 'Class' |
| 1048 | Must be defined inside a Class |
| 1049 | Expected Let or Set or Get in property declaration |
| 1050 | Expected 'Property' |
| 1051 | Number of arguments must be consistent across properties specification |
| 1052 | Cannot have multiple default property/method in a Class |
| 1053 | Class initialize or terminate do not have arguments |
| 1054 | Property Set or Let must have at least one argument |
| 1055 | Unexpected 'Next' |
| 1057 | 'Default' specification must also specify 'Public' |
| 1058 | 'Default' specification can only be on Property Get |

## Tips & Tricks

### *VBScript Editor IntelliSense*

IntelliSense provides an array of options that make language references easily accessible. When coding, you do not need to leave the Code Editor or the Immediate Mode command window to perform searches on language elements. You can keep your context, find the information you need, insert language elements directly into your code, and even have IntelliSense complete your typing for you.

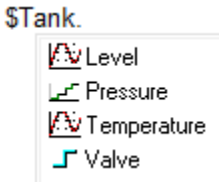
IntelliSense comprises the following options…

#### List Members

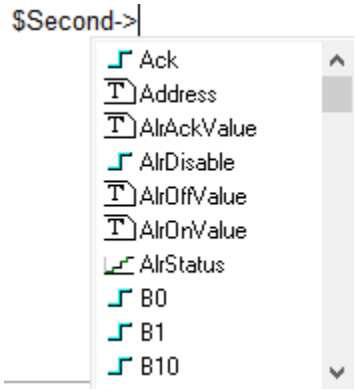
You can display a list of valid members from class tags, fields from any tag, properties/methods from an ActiveX object, or functions from the Built-in Scripting Language. Selecting from the list inserts the member into your code.

When you type the $ character on any VBScript interface, a list box will automatically open with the list of all tags available for the current project as well as all functions from the Built-in Scripting Language.

When you type the name of a class tag followed by the dot character ( . ) on any VBScript interface, a list box will automatically open with the list of all members from the class tag:



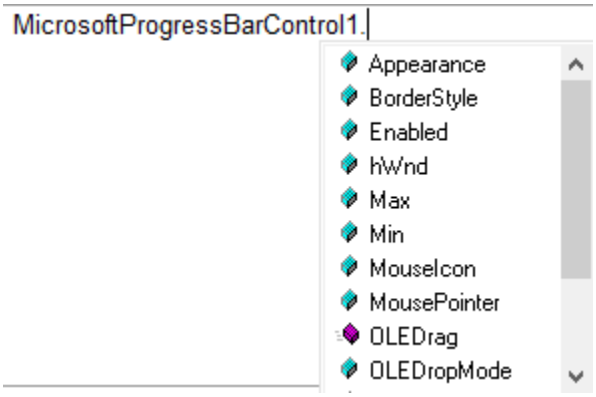
When you type the name of a tag followed by the hyphen and greater than characters ( -> ) on any VBScript interface, a list box will automatically open with the list of all fields available for this tag:




The items are displayed in alphabetic order, and each item has an icon to identify its main type, as follows:

| Icon | Type |
|---|---|
| | Boolean Tag |
| | Integer Tag |
| | Real Tag |
| | String Tag |
| | Class Tag |
| | Function from the Built-in Scripting Language |

When you type the name of an ActiveX control that is inserted on the screen followed by the dot character ( . ) on any VBScript interface from the screen where the ActiveX object is inserted, a list box will automatically open with the list of all properties and methods from the object:

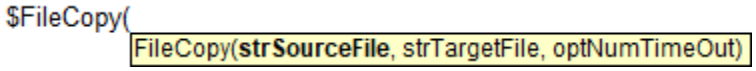



The items are displayed in alphabetic order, and each item has an icon to identify its main type, as follows:

| Icon | Type |
|---|---|
| | Property from the ActiveX object |
| | Method from the ActiveX object |

### Parameter Quick Info

The Quick Info option displays pop-up boxes with the information about the functions from the Built-in Scripting Language. The information includes all the parameters supported by this function, with the currently configured one in bold text.




### Complete Word

Complete word finishes a tag, member, field, function, or ActiveX property/method name once you have entered enough characters to disambiguate the term. After you type the first few letters of the name, you can press Ctrl+Space to complete the name automatically.

## *VBScript Compared to VBA*

While VBScript and Visual Basic for Applications (VBA) are similar and are both based on the Visual Basic standard language, there are advantages to using VBScript for EcoStruxure Machine SCADA Expert users:

- VBScript is supported for the Microsoft Windows Embedded operating system, and VBA is not.
- VBScript brings active scripting to a wide variety of environments, including Web client scripting in Microsoft Internet Explorer. This prevents operations that may present risks for the Thin Client user, such as direct access to local files.
- VBScript was designed to be simple and easy to learn, with some standards from VBA modified in VBScript to make it more straightforward. For example, in VBScript the user does not have to worry about the type of each variable when declaring them because VBScript assumes the proper type for each variable automatically.

The following table lists VBScript features that VBA does not have.

| Category | Feature/Keyword |
|---|---|
| Declarations | Class |
| Miscellaneous | Eval |
| | Execute |
| Objects | RegExp |
| Script Engine Identification | ScriptEngine |
| | ScriptEngineBuildVersion |
| | ScriptEngineMajorVersion |

The following table lists VBA features that VBScript does not have.

| Category | Omitted Feature/Keyword |
|---|---|
| Array Handling | Option Base |
| | Declaring arrays with lower bound <> 0 |
| Collection | Add, Count, Item, Remove |
| | Access to collections using ! character |
| Conditional Compilation | #Const |
| | #If...Then...#Else |
| Control Flow | DoEvents |
| | GoSub...Return, GoTo |
| | On Error GoTo |

| Category | Omitted Feature/Keyword |
|---|---|
| | On...GoSub, On...GoTo |
| | Line numbers, Line labels |
| Conversion | CVar, CVDate |
| | Str, Val |
| Data Types | All intrinsic data types except Variant |
| | Type...End Type |
| Date/Time | Date statement, Time statement |
| Debugging | Debug.Print |
| | End, Stop |
| Declaration | Declare (for declaring DLLs) |
| | Optional |
| | ParamArray |
| | Static |
| Error Handling | Erl |
| | Error |
| | Resume, Resume Next |
| File Input/Output | All traditional Basic file I/O |
| Financial | All financial functions |
| Object Manipulation | TypeOf |
| Objects | Clipboard |
| | Collection |
| Operators | Like |
| Options | Deftype |
| | Option Base |
| | Option Compare |
| | Option Private Module |
| Select Case | Expressions containing the **Is** keyword or any comparison operators |
| | Expressions containing a range of values using the **To** keyword |
| Strings | Fixed-length strings |
| | LSet, RSet |
| | Mid Statement |
| | StrConv |
| Using Objects | Collection access using ! |

## *Screen Events*

In addition to the Screen Script, you can configure logics using the Built-in Scripting Language for the On Open, While Open and On Close events for the screen (see the Screen Logic interface from the Screen Attributes dialog). If you configure the Screen Script (VBScript language) and the Screen Logic (Built-in Scripting Language), EcoStruxure Machine SCADA Expert will respect the following execution order:

| Event | Order of execution |
|---|---|
| When opening the screen | • Screen_OnOpen() sub-routine from the Screen Script interface (VBScript language)<br>• On Open from the Screen Logic interface (Built-in Scripting Language) |
| When closing the screen | • On Close from the Screen Logic interface (Built-in Scripting Language)<br>• Screen_OnClose() sub-routine from the Screen Script interface (VBScript language) |

## *MsgBox and InputBox Functions*

The **MsgBox()** and **InputBox()** functions from the VBScript language allow you to display pop-up messages during runtime. These functions are synchronous. When either one is executed, the remaining instructions from the code will not be executed before the pop-up messages launched by the functions are closed.

> **Note:** The text displayed in these pop-up messages are not affected by the Translation Tool of EcoStruxure Machine SCADA Expert, unless you configure the text explicitly using the $Ext() function from the Built-in Scripting Language.

## *VBScript Procedures*

In VBScript, there are two kinds of procedures; the Sub procedure and the Function procedure.

### Sub Procedures

A Sub procedure is a series of VBScript statements (enclosed by **Sub** and **End Sub** statements) that perform actions but don't return a value. A Sub procedure can take arguments (constants, variables, or expressions that are passed by a calling procedure). If a Sub procedure has no arguments, its **Sub** statement must include an empty set of parentheses ().

The following Sub procedure uses two intrinsic (built-in) VBScript functions, **MsgBox** and **InputBox**, to prompt a user for information. It then displays the results of a calculation based on that information. The calculation is performed in a Function procedure created with VBScript. The Function procedure is shown after the following discussion.

```
Sub ConvertTemp()
  temp = InputBox("Please enter the temperature in degrees F.", 1)
  MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub
```

### Function Procedures

A Function procedure is a series of VBScript statements enclosed by the **Function** and **End Function** statements. A Function procedure is similar to a Sub procedure, but can also return a value. A Function procedure can take arguments (constants, variables or expressions that are passed to it by a calling procedure). If a Function procedure has no arguments, its **Function** statement must include an empty set of parentheses. A Function returns a value by assigning a value to its name in one or more statements of the procedure. The return type of a Function is always a Variant.

In the following example, the **Celsius** function calculates degrees Celsius from degrees Fahrenheit. When the function is called from the **ConvertTemp** Sub procedure, a variable containing the argument value is passed to the function. The result of the calculation is returned to the calling procedure and displayed in a message box.

```
Sub ConvertTemp()
  temp = InputBox("Please enter the temperature in degrees F.", 1)
  MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub

Function Celsius(fDegrees)
  Celsius = (fDegrees - 32) * 5 / 9
End Function
```

### Getting Data Into and Out of Procedures

Each piece of data is passed into your procedures using an argument . Arguments serve as placeholders for the data you want to pass into your procedure. You can name your arguments any valid variable name. When you create a procedure using either the **Sub** statement or the **Function** statement, parentheses must be included after the name of the procedure. Any arguments are placed inside these parentheses, separated by commas. For example, in the following example, **fDegrees** is a placeholder for the value being passed into the **Celsius** function for conversion.

```
Function Celsius(fDegrees)
  Celsius = (fDegrees - 32) * 5 / 9
End Function
```

To get data out of a procedure, you must use a Function. Remember, a Function procedure can return a value; a Sub procedure cannot.

### Using Sub and Function Procedures in Code

A Function in your code must always be used on the right side of a variable assignment or in an expression. For example:

```
Temp = Celsius(fDegrees)
```

or

```
MsgBox "The Celsius temperature is " & Celsius(fDegrees) & " degrees."
```

To call a Sub procedure from another procedure, type the name of the procedure along with values for any required arguments, each separated by a comma. The **Call** statement is not required, but if you do use it, you must enclose any arguments in parentheses.

The following example shows two calls to the **MyProc** procedure. One uses the **Call** statement in the code; the other doesn't. Both do exactly the same thing.

```
Call MyProc(firstarg, secondarg)

MyProc firstarg, secondarg
```

Notice that the parentheses are omitted in the call when the **Call** statement isn't used.

## *Creating Constants*

A constant is a meaningful name that takes the place of a number or string and never changes. VBScript defines a number of intrinsic constants.

You create user-defined constants in VBScript using the Const statement. Using the Const statement, you can create string or numeric constants with meaningful names and assign them literal values. For example:

```
Const MyString = "This is my string."
Const MyAge = 49
```

Note that the string literal is enclosed in quotation marks ( " " ). Quotation marks are the most obvious way to differentiate string values from numeric values. You represent Date literals and time literals by enclosing them in number signs ( # ). For example:

```
Const CutoffDate = #6-1-97#
```

You may want to adopt a naming scheme to differentiate constants from variables. This will prevent you from trying to reassign constant values while your script is running. For example, you might want to use a "vb" or

"con" prefix on your constant names, or you might name your constants in all capital letters. Differentiating constants from variables eliminates confusion as you develop more complex scripts.

### Declaring Variables

A variable is a convenient placeholder that refers to a computer memory location where you can store program information that may change during the time your script is running. In VBScript, variables are always of one fundamental data type, Variant.

You declare variables explicitly in your script using the Dim statement, the Public statement, and the Private statement. For example:

```
Dim DegreesFahrenheit
```

You declare multiple variables by separating each variable name with a comma. For example:

```
Dim Top, Bottom, Left, Right
```

You can also declare a variable implicitly by simply using its name in your script. That is not generally a good practice because you could misspell the variable name in one or more places, causing unexpected results when your script is run. For that reason, the Option Explicit statement is configured by default in the Global Procedures interface to require explicit declaration of all variables. Unless you delete this statement, you need to declare all variables explicitly; otherwise, VBScript will generate errors during runtime indicating that the variable does not exist.

An expression should have the variable on the left side and the value you want to assign to the variable on the right. For example:

```
MyVar = 100
```

### Scope and Lifetime of Variables

A variable's scope is determined by where you declare it. When you declare a variable within a procedure, only code within that procedure can access or change the value of that variable. It has local scope and is a procedure-level variable. If you declare a variable outside a procedure, you make it recognizable to all the procedures in your script. This is a script-level variable, and it has script-level scope.

The lifetime of a variable depends on how long it exists. The lifetime of a script-level variable extends from the time it is declared until the time the script is finished running. At procedure level, a variable exists only as the procedure runs. When the procedure exits, the variable is destroyed. Local variables are ideal as temporary storage space when a procedure is executing. You can have local variables of the same name in several different procedures because each is recognized only by the procedure in which it is declared.

### How Boolean tags are handled in VBScript

By default, the numeric value of TRUE is different for Boolean tags in EcoStruxure Machine SCADA Expert 2020 than it is for Boolean variables in VBScript. This could cause problems when Boolean tags are used in VBScript, so EcoStruxure Machine SCADA Expert 2020 has been modified to change those tags are handled.

> **Note:** This topic applies only to project tags of Boolean type. Tags of other data types (e.g., Integer, Real) are handled normally at all times. For more information, see Choosing the Tag Data Type on page 156.

It is important to remember that while the Boolean states of FALSE and TRUE have the same meanings in all programming languages, and in fact they are reserved as keywords in VBScript, the numeric values of these states are different for Boolean tags versus Boolean variables, as shown in the table below:

| Boolean state | Numeric value for a... | |
|---|---|---|
| | ...Boolean tag in EcoStruxure Machine SCADA Expert 2020 | ...Boolean variable in VBScript |
| FALSE | 0 | 0 |
| TRUE | 1 | -1 |

These are the values that are actually stored in the project database. Any interpretation of these values as "false" or "true" is done by the software during run time. As such, if you tried to use Boolean tags with certain VBScript statements and operators — especially the logical NOT operator — you might get unexpected results.

To prevent any problems and make sure that Boolean tags have the correct values at all times, the VBScript interface in EcoStruxure Machine SCADA Expert 2020 has been modified to preprocess Boolean tags and handle them like Boolean variables. In other words, when a Boolean tag has an actual value of 1, the VBScript interface handles it as if it has a value of -1.

This modification was introduced in an earlier version of this software. If you used the latest version of this software to create your project, it includes the modification by default, so there is nothing you need to do.

In order to maintain backward compatibility, however, a project that was created in an earlier version of EcoStruxure Machine SCADA Expert 2020 and then upgraded to the latest version does not include the modification by default. When you open an upgraded but unmodified project, the following message is displayed:
Warning: For compatibility reasons your project is not using the VB Boolean mode. Please refer to the VBScript section of your technical reference manual for more information.

When this happens, you have two options for how to proceed. First, you can ignore the message and continue running your project as it was originally developed. Either your project does not include any VBScript code, or the code that it does include already works around this issue. As long as your project behaves as expected and you do not add new code, there is nothing you need to do.

Second, you can manually edit your project file in order to apply the modificiation — use a text editor to open your project file (*<project name>*.APP), and then add the following setting:

```
[Script]
VBBoolean=1
```

After you do this, however, you must thoroughly test your project in order to make sure that your VBScript code still behaves as expected.

### *Writing Real Values to Integer Tags*

By default, a Real (i.e., floating point) value is truncated at the decimal point when it is written to an Integer tag. This behavior is the same in both the Built-in Scripting Language and in VBScript.

You can change this behavior in VBScript, however, by manually editing your project file (*<project name>*.app) to change the following setting:

```
[Script]
TruncRealToInt=0 or 1
```

If **TruncRealToInt** is set to **1**, the project will behave as described above: Real values will be truncated at the decimal point without rounding. (For example, a value of **5.56** will be written as **5**.) This is the default setting for projects that were created with some earlier versions of this software and then upgraded to the latest version, in order to maintain backward compatibility.

If **TruncRealToInt** is set to **0**, VBScript functions and operations will round Real values to the nearest whole numbers. (For example, a value of **5.56** will be written as **6**.) This is the default setting for projects that are created with the latest version of this software.

> **Note:** This setting only affects the behavior of VBScript in EcoStruxure Machine SCADA Expert 2020. It does not affect the behavior of the Built-in Scripting Language.

## *Precedence of VBScript Operators*

VBScript has a full range of operators, including arithmetic operators, comparison operators, concatenation operators, and logical operators.

When several operations occur in an expression, each part is evaluated and resolved in a predetermined order called "operator precedence." You can use parentheses to override the order of precedence and force some parts of an expression to be evaluated before others. Operations within parentheses are always performed before those outside. Within parentheses, however, standard operator precedence is maintained.

When expressions contain operators from more than one category, arithmetic operators are evaluated first, comparison operators are evaluated next, and logical operators are evaluated last. Comparison operators all have equal precedence; that is, they are evaluated in the left-to-right order in which they appear. Arithmetic and logical operators are evaluated in the following order of precedence.

| Arithmetic | Comparison | Logical |
|---|---|---|
| Negation (-) | Equality (=) | Not |
| Exponentiation (^) | Inequality (<>) | And |
| Multiplication and division (*, /) | Less than (<) | Or |
| Integer division (\) | Greater than (>) | Xor |
| Modulus arithmetic (Mod) | Less than or equal to (<=) | Eqv |
| Addition and subtraction (+, -) | Greater than or equal to (>=) | Imp |
| String concatenation (&) | Is | & |

When multiplication and division occur together in an expression, each operation is evaluated as it occurs from left to right. Likewise, when addition and subtraction occur together in an expression, each operation is evaluated in order of appearance from left to right.

The string concatenation (&) operator is not an arithmetic operator, but in precedence it falls after all arithmetic operators and before all comparison operators. The Is operator is an object reference comparison operator. It does not compare objects or their values; it checks only to determine if two object references refer to the same object.

## *Logical Operator NOT*

The logical operator NOT behaves differently in VBScript than it does in the built-in scripting language.

### NOT Operator in VBScript

In VBScript, the NOT operator inverts the bits of a given numeric value, producing its complement number according to the "two's complement" system of signed numbers that is used by computers. The table below illustrates the behavior of the NOT operator in VBScript for the syntax...

```
result = NOT expression
```

| If *expression* is... | Then *result* is... |
|---|---|
| −3 | 2 |
| −2 | 1 |
| −1 | 0 |
| 0 | −1 |
| 1 | −2 |
| 2 | −3 |
| 3 | −4 |

> **Note:** By default, when you attempt to write any numeric value other than 0 to a Boolean tag, the tag automatically assumes a value of 1. Therefore, if VBScript's NOT operator is applied to a Boolean tag with a value of 1, then the value of the tag does not change; the operator returns a value of −2, but the tag cannot accept this value so it again assumes a value of 1.
>
> You can configure EcoStruxure Machine SCADA Expert to handle Boolean tags like Boolean variables in VBScript, so that the NOT operator in VBScript will work as expected. For more information, see How Boolean tags are handled in VBScript on page 1238.

### NOT Operator in Built-in Language

In contrast, the NOT operator in the Built-in Scripting Language toggles the given numeric value as if it is a natural boolean. The table below illustrates the behavior of the NOT operator in the Built-in Scripting Language for the syntax...

```
result = NOT expression
```

| If *expression* is... | Then *result* is... |
|---|---|
| 0 | 1 |
| ≠0 | 0 |

## *Using Conditional Statements*

You can control the flow of your script with conditional statements and looping statements. Using conditional statements, you can write VBScript code that makes decisions and repeats actions. The following conditional statements are available in VBScript:

- **If**...**Then**...**Else** statement
- **Select Case** statement

### Making Decisions Using If...Then...Else

The **If**...**Then**...**Else** statement is used to evaluate whether a condition is **True** or **False** and, depending on the result, to specify one or more statements to run. Usually the condition is an expression that uses a comparison operator to compare one value or variable with another. For information about comparison operators, see Comparison Operators.

**If**...**Then**...**Else** statements can be nested to as many levels as you need.

#### Running Statements if a Condition is True

To run only one statement when a condition is True, use the single-line syntax for the **If**...**Then**...**Else** statement. The following example shows the single-line syntax. Notice that this example omits the **Else** keyword:

```
Sub FixDate()
  Dim myDate
  myDate = #2/13/95#
  If myDate < Now Then myDate = Now
End Sub
```

To run more than one line of code, you must use the multiple-line (or block) syntax. This syntax includes the **End If** statement, as shown in the following example:

```
Sub AlertUser(value)
  If value = 0 Then
    AlertLabel.ForeColor = vbRed
    AlertLabel.Font.Bold = True
    AlertLabel.Font.Italic = True
  End If
```

```
End Sub
```

#### Running Certain Statements if a Condition is True and Running Others if a Condition is False

You can use an **If**…**Then**…**Else** statement to define two blocks of executable statements: one block to run if the condition is **True**, and the other block to run if the condition is **False**:

```
Sub AlertUser(value)
  If value = 0 Then
    AlertLabel.ForeColor = vbRed
    AlertLabel.Font.Bold = True
    AlertLabel.Font.Italic = True
  Else
    AlertLabel.Forecolor = vbBlack
    AlertLabel.Font.Bold = False
    AlertLabel.Font.Italic = False
  End If
End Sub
```

#### Deciding Between Several Alternatives

A variation on the **If**…**Then**…**Else** statement allows you to choose from several alternatives. Adding **ElseIf** clauses expands the functionality of the **If**…**Then**…**Else** statement, so you can control program flow based on different possibilities. For example:

```
Sub ReportValue(value)
  If value = 0 Then
    MsgBox value
  ElseIf value = 1 Then
    MsgBox value
  ElseIf value = 2 then
    Msgbox value
  Else
    Msgbox "Value out of range!"
  End If
End Sub
```

You can add as many **ElseIf** clauses as you need to provide alternative choices, but extensive use of the **ElseIf** clauses often becomes cumbersome. A better way to choose between several alternatives is the **Select Case** statement.

### Making Decisions with Select Case

The **Select Case** structure provides an alternative to **If**…**Then**…**ElseIf** for selectively executing one block of statements from among multiple blocks of statements. A **Select Case** statement provides capability similar to the **If**…**Then**…**Else** statement, but it makes code more efficient and readable.

A **Select Case** structure works with a single test expression that is evaluated once, at the top of the structure. The result of the expression is then compared to the values for each **Case** in the structure. If there is a match, the block of statements associated with that **Case** is executed, as in the following example:

```
Select Case Document.Form1.CardType.Options(SelectedIndex).Text
  Case "MasterCard"
    DisplayMCLogo
    ValidateMCAccount
  Case "Visa"
    DisplayVisaLogo
    ValidateVisaAccount
  Case "American Express"
    DisplayAMEXCOLogo
    ValidateAMEXCOAccount
  Case Else
    DisplayUnknownImage
    PromptAgain
```

```
End Select
```

Notice that the **Select Case** structure evaluates an expression once at the top of the structure. In contrast, the **If**…**Then**…**ElseIf** structure can evaluate a different expression for each **ElseIf** statement. You can replace an **If**…**Then**…**ElseIf** structure with a **Select Case** structure only if each **ElseIf** statement evaluates the same expression.

## *Looping Through Code*

Looping allows you to run a group of statements repeatedly. Some loops repeat statements until a condition is False; others repeat statements until a condition is True. There are also loops that repeat statements a specific number of times.

The following looping statements are available in VBScript:

- **Do**…**Loop**: Loops while or until a condition is True
- **While**…**Wend**: Loops while a condition is True
- **For**…**Next**: Uses a counter to run statements a specified number of times

### Using Do Loops

You can use **Do**…**Loop** statements to run a block of statements an indefinite number of times. The statements are repeated either while a condition is True or until a condition becomes True.

**Repeating Statements While a Condition is True**

Use the While keyword to check a condition in a **Do**…**Loop** statement. You can check the condition before you enter the loop (as shown in the following **ChkFirstWhile** example), or you can check it after the loop has run at least once (as shown in the **ChkLastWhile** example). In the **ChkFirstWhile** procedure, if **myNum** is set to 9 instead of 20, the statements inside the loop will never run. In the **ChkLastWhile** procedure, the statements inside the loop run only once because the condition is already False.

```
Sub ChkFirstWhile()
  Dim counter, myNum
  counter = 0
  myNum = 20
  Do While myNum > 10
    myNum = myNum - 1
    counter = counter + 1
  Loop
  MsgBox "The loop made " & counter & " repetitions."
End Sub

Sub ChkLastWhile()
  Dim counter, myNum
  counter = 0
  myNum = 9
  Do
    myNum = myNum - 1
    counter = counter + 1
  Loop While myNum > 10
  MsgBox "The loop made " & counter & " repetitions."
End Sub
```

**Repeating a Statement Until a Condition Becomes True**

There are two ways to use the **Until** keyword to check a condition in a **Do**…**Loop** statement. You can check the condition before you enter the loop (as shown in the following **ChkFirstUntil** example), or you can check it after the loop has run at least once (as shown in the **ChkLastUntil** example). As long as the condition is **False**, the looping occurs.

```
Sub ChkFirstUntil()
  Dim counter, myNum
  counter = 0
  myNum = 20
```

```
  Do Until myNum = 10
    myNum = myNum - 1
    counter = counter + 1
  Loop
  MsgBox "The loop made " & counter & " repetitions."
End Sub

Sub ChkLastUntil()
  Dim counter, myNum
  counter = 0
  myNum = 1
  Do
    myNum = myNum + 1
    counter = counter + 1
  Loop Until myNum = 10
  MsgBox "The loop made " & counter & " repetitions."
End Sub
```

#### Exiting a Do...Loop Statement from Inside the Loop

You can exit a **Do**…**Loop** by using the **Exit Do** statement. Because you usually want to exit only in certain situations, such as to avoid an endless loop, you should use the **Exit Do** statement in the **True** statement block of an **If**…**Then**…**Else** statement. If the condition is **False**, the loop runs as usual.

In the following example, **myNum** is assigned a value that creates an endless loop. The **If**…**Then**…**Else** statement checks for this condition, preventing the endless repetition.

```
Sub ExitExample()
  Dim counter, myNum
  counter = 0
  myNum = 9
  Do Until myNum = 10
    myNum = myNum - 1
    counter = counter + 1
    If myNum < 10 Then Exit Do
  Loop
  MsgBox "The loop made " & counter & " repetitions."
End Sub
```

### Using While…Wend

The **While**…**Wend** statement is provided in VBScript for those who are familiar with its usage. However, because of the lack of flexibility in **While**…**Wend**, it is recommended that you use **Do**…**Loop** instead.

### Using For…Next

You can use **For**…**Next** statements to run a block of statements a specific number of times. For loops, use a counter variable whose value increases or decreases with each repetition of the loop.

The following example causes a procedure called **MyProc** to execute 50 times. The For statement specifies the counter variable **x** and its start and end values. The **Next** statement increments the counter variable by 1.

```
Sub DoMyProc50Times()
  Dim x
  For x = 1 To 50
    MyProc
  Next
End Sub
```

Using the **Step** keyword, you can increase or decrease the counter variable by the value you specify. In the following example, the counter variable **j** is incremented by 2 each time the loop repeats. When the loop is finished, the total is the sum of 2, 4, 6, 8, and 10.

```
Sub TwosTotal()
  Dim j, total
```

```
  For j = 2 To 10 Step 2
    total = total + j
  Next
  MsgBox "The total is " & total
End Sub
```

To decrease the counter variable, use a negative **Step** value. You must specify an end value that is less than the start value. In the following example, the counter variable **myNum** is decreased by 2 each time the loop repeats. When the loop is finished, the total is the sum of 16, 14, 12, 10, 8, 6, 4, and 2.

```
Sub NewTotal()
  Dim myNum, total
  For myNum = 16 To 2 Step -2
    total = total + myNum
  Next
  MsgBox "The total is " & total
End Sub
```

You can exit any **For**…**Next** statement before the counter reaches its end value by using the **Exit For** statement. Because you usually want to exit only in certain situations, such as when an error occurs, you should use the **Exit For** statement in the **True** statement block of an **If**…**Then**…**Else** statement. If the condition is **False**, the loop runs as usual.

### *Support for ActiveX Controls*

Using the VBScript interfaces for the Graphic module (Graphics Script, Screen Script, Command animation, and ActiveX Events), you can use this syntax to access properties and methods directly from any ActiveX Control object inserted in the screen where the object is configured.

EcoStruxure Machine SCADA Expert will assign a unique name to the object on the screen. You can use the Name property (in the *Object Properties* dialog) to modify this name.

After inserting an ActiveX Control object on the screen, you can access properties and methods from this object from any VBScript interface associated with this screen. Use the syntax ***Object_Name.Properties_or_Method_Name***. Examples:

```
//Access the value of the property Day from the CalendarControl1 ActiveX object
CalendarControl1.Day

//Triggers the method AboutBox from the CalendarControl1 ActiveX object
CalendarControl1.AboutBox
```

## Debugging VBScript

The development application provides additional tools for debugging your VBScipt code.

> **Tip:** These tools are based on the debugging tools in Microsoft Visual Studio, so if you are experienced with Visual Studio, then these tools should be familiar to you as well.

The tools are available in the following VBScript interfaces in the development application:

- Procedures, in the **Global** tab of the Project Explorer;
- The Graphics Script, in the **Graphics** tab of the Project Explorer;
- Script groups, including the Startup Script, in the **Tasks** tab of the Project Explorer; and
- The Screen Script that is attached to each Screen worksheet.

At this time, debugging is not supported in the following VBScript interfaces:

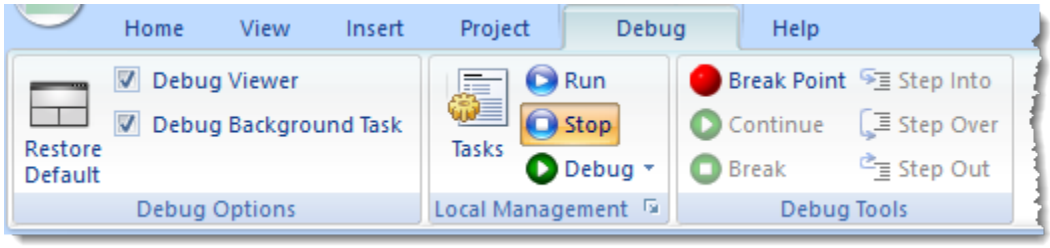
- The Command animation that is attached to each Button object; and
- ActiveX and .NET Control objects.

Generally speaking, you can debug your code by running your project in **Debug** mode (as opposed to the usual **Run** mode) and then observing how the values of project tags, locally-declared variables, and entire functions change as you step through the code. You can control the stepping by using the **Debug** tab of the ribbon, and you can observe the changing values in the *Watch* window.

### *About the Debug tab*

The **Debug** tab of the ribbon is used to debug the VBScript code in your project.

> **Note:** This tab of the ribbon is contextual: it appears only when you use a VBScript interface to view or edit your code.




***Debug tab of the ribbon***

The tools are organized into the following groups:

**Debug Options**

These are general options for the debugging tools.

First, the **Restore Default** tool restores the development environment to its default layout. It is the same as the **Restore Default** tool on the **View** tab of the ribbon, and it is useful when you have extensively resized and rearranged the windows to facilitate debugging.

Second, the **Debug**... options determine exactly which parts of the project runtime will be debugged. Normally, both parts are debugged at the same time, but if you want one to run without interruption while you focus on the other, then you can control that here.

**Local Management**

These tools are the same as in the **Local Management** group on the **Home** tab of the ribbon. You can use them to run and stop your project, as well as to check the states of the many tasks and modules that make up the project runtime.

**Debug Tools**

These tools control the actual code stepping when you run your project in Debug mode. For more information about these tools, see the rest of the "Debugging VBScript" section.

### *Set break points in your VBScript code*

Set break points in your VBScript code to indicate where run-time execution should be suspended.

As you develop your code, you will identify important sections where you want to focus your attention while debugging. For example, if your code includes a critical function that updates the values of several project tags, then you may want to suspend execution at the beginning of that function and observe the value changes as they happen. That is where you should set a break point.

A break point is a signal that tells the debugger to temporarily suspend execution of your project. When execution is suspended at a break point, your project is said to be in break mode. Entering break mode does not stop or end the execution of your project; execution can be resumed at any time.

You can think of break mode as being like a time-out. All the elements remain (functions, variables, and objects remain in memory, for example), but their movements and activities are suspended. During break mode, you can examine the elements' positions and states to look for violations or bugs. You can also make adjustments to the project while in break mode; for example, you can change the value of a variable.

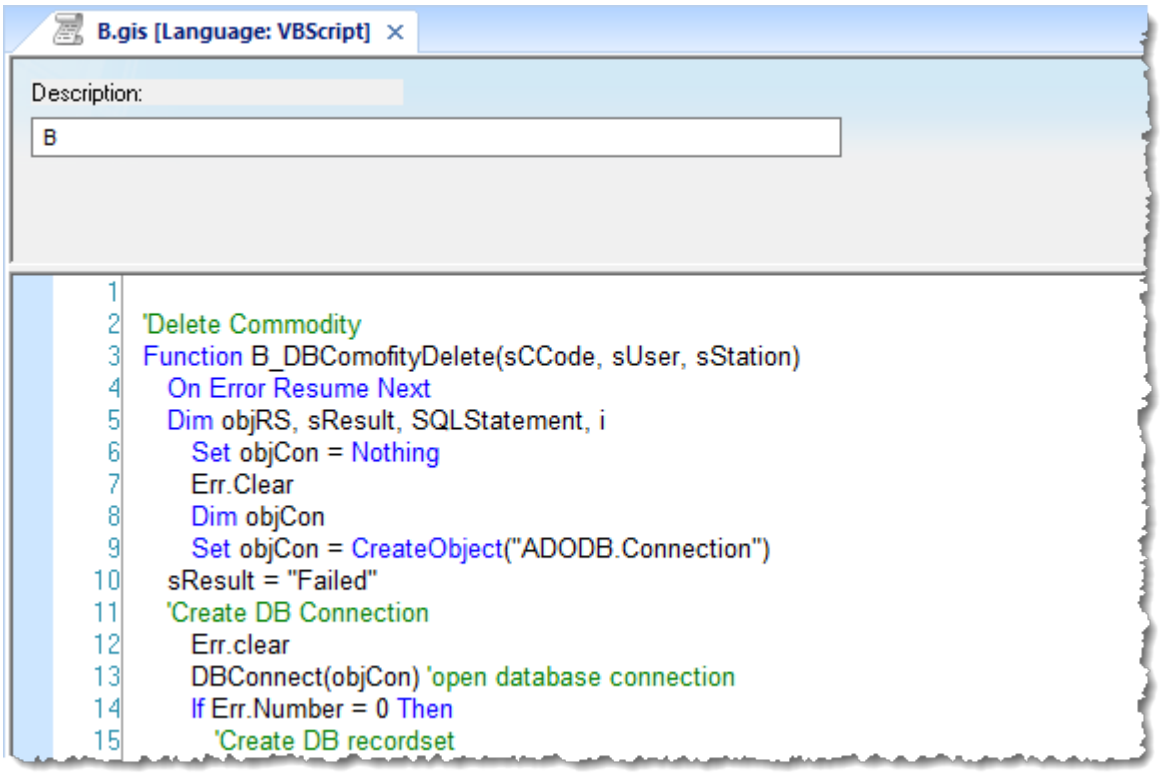
Break points provide a powerful tool that enables you to suspend execution where and when you need to. Rather than stepping through your code line by line, you can allow your project to run until it hits a break point, and then start to debug. This speeds up the debugging process. Without this ability, it would be almost impossible to debug a large project.

To set one or more break points in your VBScript code:

1. Open the VBScript worksheet that you want to debug.

   The worksheet must be in one of the VBScript interfaces that supports debugging. For more information, see Debugging VBScript on page 1246.

   The worksheet is opened for editing.



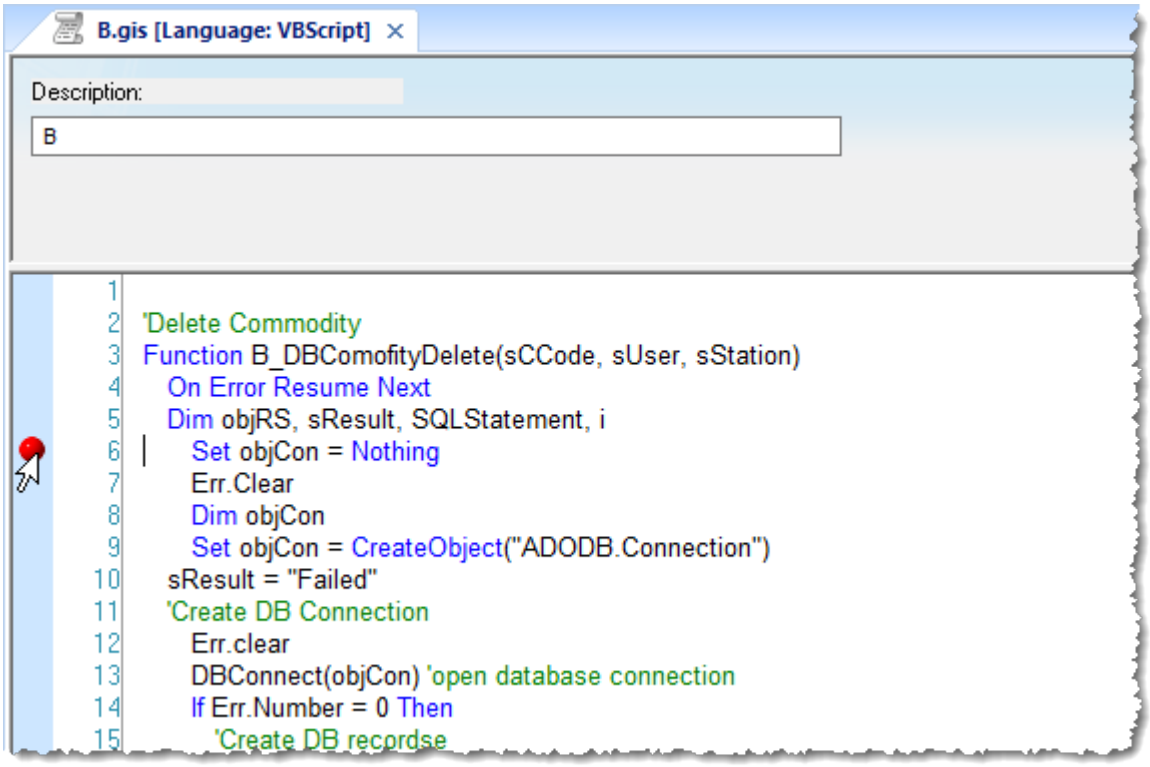

***Example of a VBScript interface***

2. In the worksheet, find the line of code where you want to set a break point.

> **Tip:** To show/hide line numbers in the VBScript interface, select **Line Number** on the **View** tab of the ribbon. The option is set independently for each worksheet; there is no global setting.

3. Do one of the following:
   - Click in the light blue gutter to the left of that line; or
   - Click in the line to place your cursor, and then click **Break Point** on the ribbon.

   A red break point symbol is inserted. If a break point cannot be inserted exactly where you clicked — for example, if it is a `Dim` statement that does not actually change any values — then it will be automatically inserted at the next possible line.




***Setting a break point***

4. Repeat these steps for each break point that you want to set.
5. Save and close all open VBScript worksheets.

Any break points that you set are saved with the worksheet(s).

## *Run your project in Debug mode*

Run your project in Debug mode in order to use the other debugging tools.

A project can be run either in normal Run mode or in Debug mode. You must run your project in Debug mode in order to use the other debugging tools such as break points, stepping, and the VBScript-specific tabs in the *Watch* window.

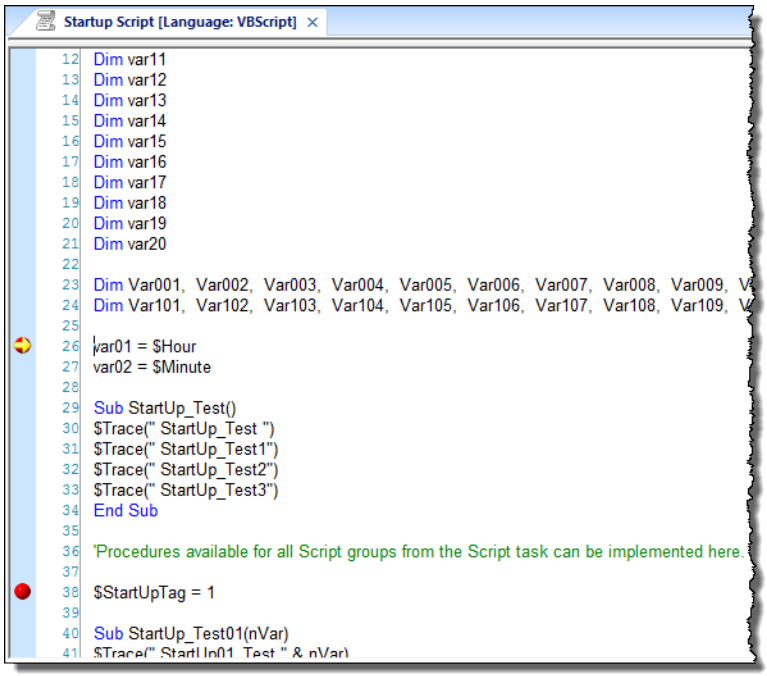
To run your project in Debug mode:

1. On **Debug** tab of the ribbon, in the **Local Management** group, click **Debug**.

   The project runtime is started in Debug mode. If the **Debug Viewer** option (on the **Debug** tab , in the **Debug Options** group) is selected and the Viewer task is set to start up, then the Secure Viewer is also started.

> **Note:** When a project is run locally, the standard Viewer is normally used. In contrast, when a project is run in Debug mode, the Secure Viewer is used for better thread and process management.

The VBScript code is executed up to the first break point that you set, assuming you set break points. (The "first" break point is the first one that comes in the logical, sequential execution of the code, not literally the first one that you set.) At that point, the project automatically enters break mode: execution is suspended and a yellow arrow is displayed in the VBScript interface to show exactly where in the code that execution was suspended.



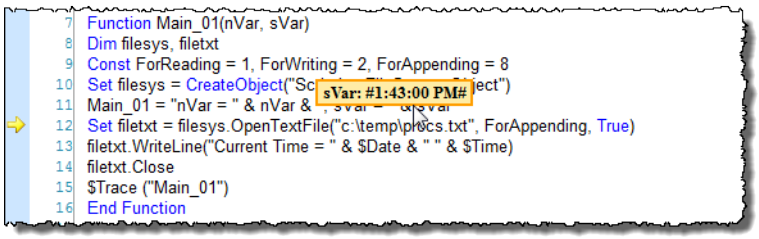

***The yellow arrow shows where execution was suspended***

If you did not set break points, then the project will continue to run until you click either **Stop** (to stop the project runtime) or **Break** (to manually enter break mode) on the **Debug** tab.

2. Once the project is in break mode, do one of the following:
   - Click **Continue** on the **Debug** tab to resume execution and continue to the next break point.
   - Check the *Watch* window to see the current state of the project. For more information, see Observe the current state in the Watch window on page 1250.
   - Click **Step Into**, **Step Over**, or **Step Out** on the **Debug** tab to step through the code one line at a time. For more information, see Step through your VBScript code on page 1252.

> **Note:**

While your project is in break mode, you can hover the mouse pointer over any VBScript variable in the code to get the current value of that variable. The value is shown in a pop-up box.




*Current value of VBScript variable in a pop-up box*

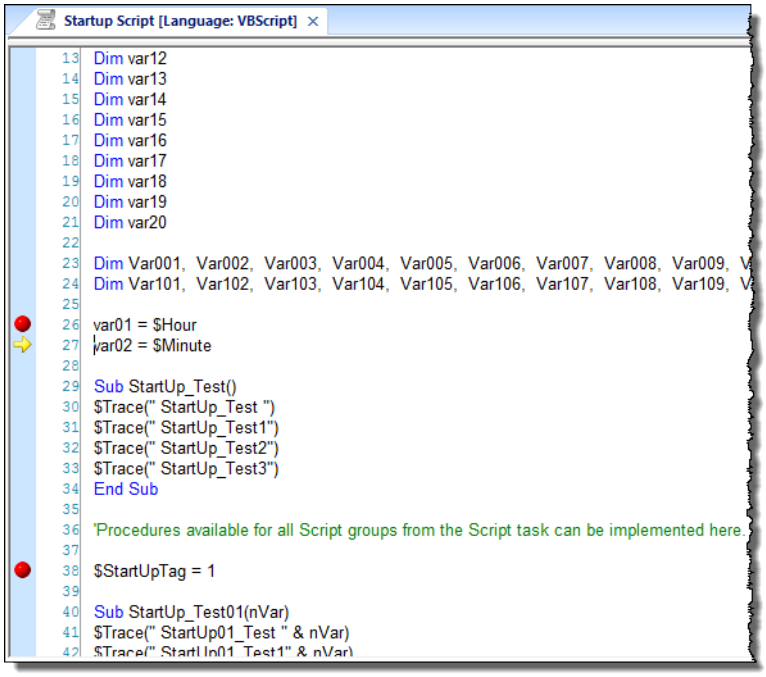
3. Repeat the previous step as desired.
4. When you have finished debugging your code, click **Stop** on the **Debug** tab.

**Note:** When you run a project in Debug mode, you will not be able to stop it until the project's Startup Script has been completely executed.

## Observe the current state in the Watch window

Use the *Watch* window to see the current state of the project while it is in break mode.

Your project must already be in break mode, either by reaching a break point that you set earlier or by using the **Break** tool to manually enter break mode, before you can use these tools.




*Project in break mode, with yellow arrow showing where execution is suspended*

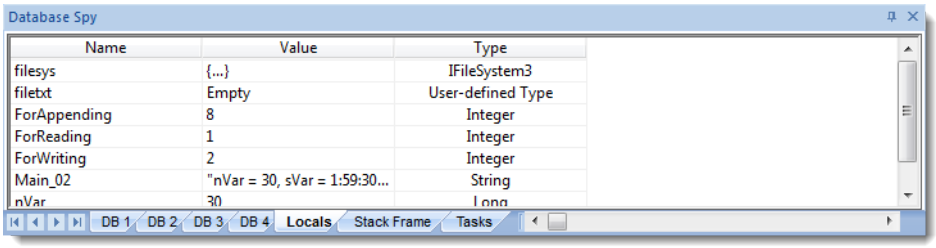
The *Watch* window has four basic tabs (**DB 1** through **DB 4**) that can be used at any time to view and adjust project tags and to excute in-line scripts. All four of those tabs work in the same way; four of them are provided simply to give you space to organize your work. For more information about using those tabs, see Watch window on page 71.

> **Tip:** You may enter the names of VBScript functions (but not sub-routines) in the *Watch* window to get the returned values of those functions.

In addition to those four tabs, however, are three new tabs that only work with VBScript debugging:

**Locals**

This tab shows all of the locally-declared VBScript variables and the current values of those variables.

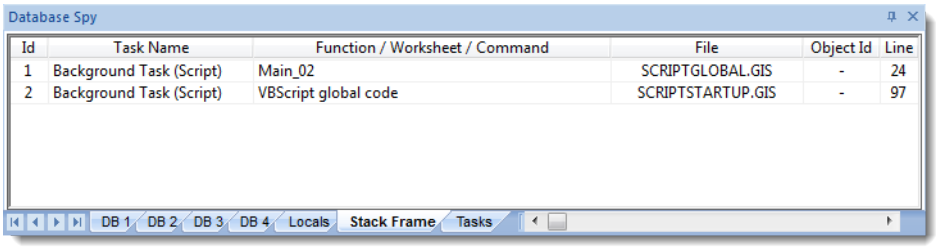



***Locals tab of the Watch window***

**Stack Frame**

This tab shows additional information about the VBScript interfaces that are currently open for debugging. The interfaces are listed in order of mostly recently executed, so that you can see how one script calls functions in another, and the **Line** column shows the most recently executed line in each interface.

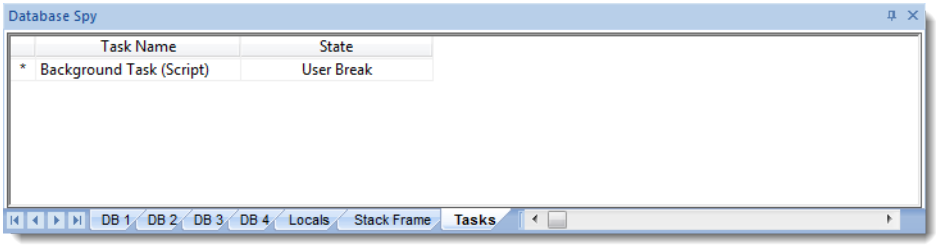
Therefore, in the screenshot below, you can see that execution was suspended at line 24 in the function **`Main_02`** in the Global Procedures, and that the function **`Main_02`** was previously called at line 97 of the Startup Script.




***Stack Frame tab of the Watch window***

**Tasks**

This tab shows all of the run-time tasks that being debugged and the current state of each task.




***Tasks tab of the Watch window***

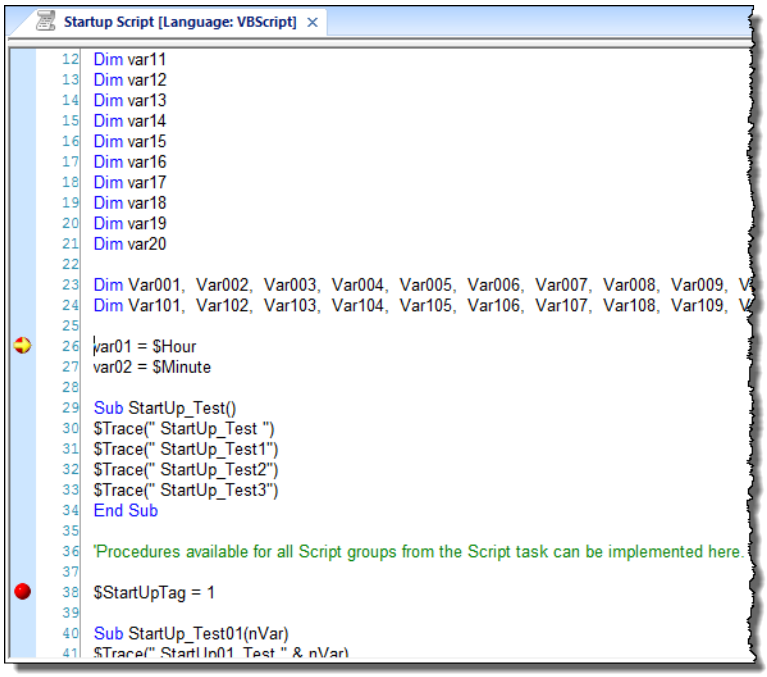
> **Note:** At this time, only Background Task (Script) can be debugged, so it is the only task that will ever be shown in this tab. Other tasks may be shown in the future, as the debugging feature is enabled for more VBScript interfaces.

Keep in mind that these tabs show information only when a project is running in Debug mode and the project is in break.

### *Step through your VBScript code*

Use the **Step Into**, **Step Over**, and **Step Out** tools to step through your VBScript code one line at a time.

Your project must already be in break mode, either by reaching a break point that you set earlier or by using the **Break** tool to manually enter break mode, before you can use these tools.

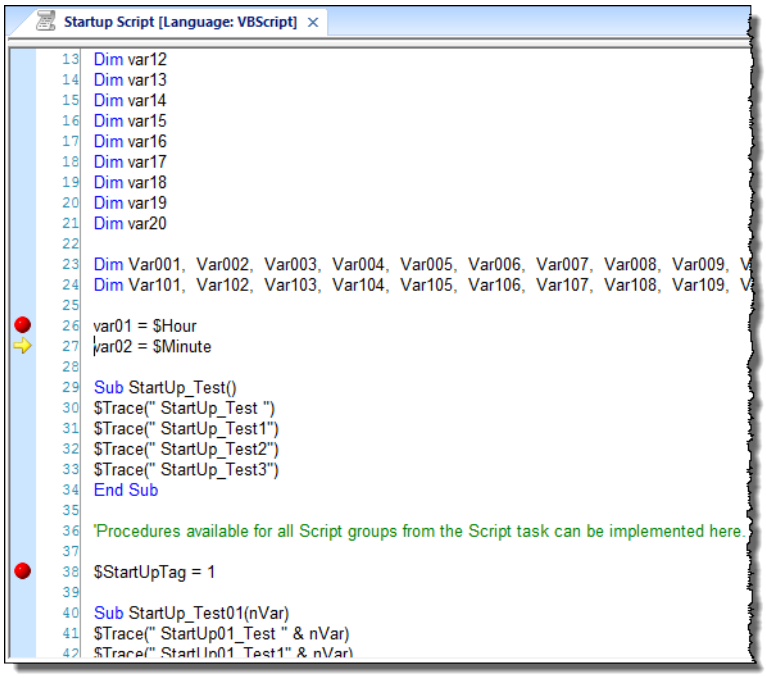



***Project in break mode, with yellow arrow showing where execution is suspended***

1. To advance one step in the code, no matter what the step may be, click **Step Into** on the **Debug** tab.

   This always moves the debugger a single step forward. If the next step is a function call, then the function is called and execution is suspended again at the first step of that function. That is what "step into" means: the debugger steps into the called function.

The yellow arrow is moved to show that the step has been taken.




***Yellow arrow showing one step forward***

2. To advance one step in the main script only, click **Step Over** on the **Debug** tab.

   If the next step is a function call, then the entire function is executed and the debugger continues with the main script. That is what "step over" means: the debugger steps over the called function in its entirety. The function is not skipped — it is still executed as written — but no additional time is spent stepping through it.

   > **Note:** You should use this tool only when you have already debugged the function and you trust it to execute correctly.

   The yellow arrow is moved to show that the step has been taken.

3. To finish executing a function and then continue with the main script, click **Step Out** on the **Debug** tab.

   That is what "step out" means: the debugger steps back out of a function that it has already stepped into. The function is not aborted — it is still executed as written — but no additional time is spent stepping through it.

   > **Note:** You should use this tool only when you have already debugged the function and you trust it to execute correctly.

   The yellow arrow is moved to show that the step has been taken.

After every step, the *Watch* window is updated to show the current state of the project. For more information, see Observe the current state in the Watch window on page 1250.