



# Application Code Manager User Manual



## Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.



**WARNING:** Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.

---



**ATTENTION:** Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

---

**IMPORTANT** Identifies information that is critical for successful application and understanding of the product.

---

Labels may also be on or inside the equipment to provide specific precautions.

---



**SHOCK HAZARD:** Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.

---



**BURN HAZARD:** Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

---



**ARC FLASH HAZARD:** Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

---

<b>Application Code Manager</b>	<b>Chapter 1</b>	
	Application Code Manager Overview.....	7
	Design process .....	8
	Library objects .....	11
	Templates.....	12
	Schedules .....	13
	Design Automation Concept.....	13
	Create a central ACM database .....	13
	Upgrading a central ACM database.....	15
	Initial configuration of Application Code Manager.....	15
	Upgrade the Application Code Manager application.....	16
	Navigate the Application Code Manager user interface .....	17
<b>Connect to an ACM database</b>	<b>Chapter 2</b>	
	Connect to an ACM database.....	21
	Connection Properties settings .....	22
	Advanced Properties settings.....	23
<b>Project Development</b>	<b>Chapter 3</b>	
	Project Development .....	27
	Project commands .....	27
	Create a new project .....	28
	Create a new project from an existing project.....	29
	Add a Historian object .....	29
	Add a Historian sub-object.....	30
	Historian Point Type parameters .....	31
	Generate a Historian object .....	32
	Add an HMI object .....	33
	Add a Display sub-object .....	34
	HMI Display parameters .....	34
	Generate an HMI object .....	35
	Add an Alarm Group sub-object.....	36
	FTA E Alarm Group parameters.....	37
	Find an object .....	37
	Update a project library .....	37
	Extract attached files .....	38
	Indications.....	39
<b>Registered Libraries</b>	<b>Chapter 4</b>	
	Registered Libraries.....	41
	Register an ACM library object .....	42

Reconstitute an ACD file .....	42
View registered library usage.....	43
Share libraries, templates, and schedules .....	44
Local library and template file location .....	45
Library Repositories .....	45

## Chapter 5

### Configure controllers

Configure controllers .....	47
Generate a controller file .....	48
Add a new controller .....	49
Move or copy a controller to another project .....	49
Select Destination Project dialog box.....	50
Add a controller from an ACD or L5X file.....	51
Update using an ACD or L5X file .....	51
Detach from ACD/L5X files .....	53
Contribute instances .....	53
Merge controllers .....	53
Add a hardware module.....	54
Delete a hardware module.....	54
Use Copy and Paste Special .....	55
Add a new instance .....	56
Add a new software object to a controller .....	56
Generate a partial program .....	57
Generate a partial routine .....	58
Object Configuration Wizard .....	58
Auto Create Linked Objects.....	59

## Chapter 6

### Use ACM Tools

Use ACM Tools .....	61
Import Export Manager .....	61
Import a schedule .....	63
Import Export Manager Import tab settings .....	64
Export a schedule.....	64
Import Export Manager Export tab settings .....	66
Compare a project to a saved schedule .....	66
Import Export Manager Compare Tab settings .....	67
Update controller parameters with the related tag values .....	67
Import Export Manager Tags Import tab settings.....	68
Generate a Tag Configurator workbook .....	68
Import Export Manager Tags Export tab settings .....	69
Create an ACM partial import file .....	69
Import Export Manager IAB/Architect tab settings .....	70

Create a new schedule template .....	70
Database Manager .....	73
Create an ACM database .....	73
Upgrade an ACM database.....	74
Backup an ACM database.....	75
Restore an ACM database.....	75
Delete an ACM database.....	77
Database Manager settings.....	77
Log Debug Information.....	78
Log File Viewer .....	79
ACM Default Settings .....	79
Document Template Editor.....	80
Components on the Document Template Editor .....	81
Menu bar.....	82
Tool bar .....	84
Output panel .....	86
Editor pane .....	86
Supported instructions .....	87
Types datasets .....	93
Technical view of looping statements .....	95
Getting to know the ACM database structure .....	97
Working with SQL Queries in Document Generation Scripts.....	103
Add ACM library content to an existing ACD project .....	107

## Chapter 7

<b>Reports</b>	Reports.....	109
	Report command reference.....	110
	Generate a report .....	111

## Chapter 8

<b>ACM Console</b>	ACM Console.....	113
	Open the ACM Console.....	114
	List all commands .....	115
	Generate a limited list of commands .....	115
	Generate detailed command information .....	115
	Console scripts .....	116
	Create an Application Code Manager Console script .....	116
	Run an Application Code Manager Console script .....	116
	Extended scripts .....	117
<b>Legal Notices</b>	Legal Notices .....	123



# Application Code Manager

## Application Code Manager Overview

Studio 5000® Application Code Manager is a tool that enables more efficient project development with libraries of reusable code. Application Code Manager creates modular objects with customizable configuration parameters using the reusable content. Application Code Manager can also create the associated visualization, historical and alarming elements for a project.

Use Application Code Manager to:

- Create databases
- Register libraries
- Create projects
- Manage objects

Application Code Manager includes an additional command-line tool, the ACM Console. Use the ACM Console to perform the following actions:

- Edit Parameters
- Export All Projects
- Export Libraries by Attribute
- Generate Controller (as an L5X or ACD file)
- Import Project
- Publish Library

Before starting a project, become familiar with the basic concepts used in Application Code Manager; the design process, the different library objects, the available templates, the use of schedules, and the design outputs from Application Code Manager.

## Activation

Activating Application Code Manager provides access to all Application Code Manager features.

When not activated Application Code Manager has the following limitations:

- Application Code Manager can only connect to a local database instance.
- Application Code Manager can only support a single controller per project. Attempting to import projects with multiple controllers will result in an error.

## See also

[Design process](#) on [page 8](#)

[Library objects](#) on [page 11](#)

[Templates](#) on [page 12](#)

[Schedules](#) on [page 13](#)

[Initial configuration of Application Code Manager](#) on [page 15](#)

## Design process

The Application Code Manager (ACM) design process introduces a modular, object-based approach to the creation of ACD controller code, FactoryTalk® View SE/ME display content, FactoryTalk Historian Tag and Alarms import configuration.

The Studio 5000 ACM design process separates function and configuration into two separate layers of data, and divides the design process into two distinct workflows, library management and project execution.

The design process involves a suite of applications:

- The Studio 5000 Logix Designer® application
- The Library Designer
- The Library Object Manager application
- The Application Code Manager application
- FactoryTalk View Studio

## Library management workflow: Studio 5000 Logix Designer

The library management workflow begins when a specific instance of ACD controller code is created in the Logix Designer application. The specific instance is a single project containing a single controller. The project includes a logical structure containing these Logix objects:

- Controller Tags
- Tasks
- Motion Groups
- Add-On Instructions
- Data Types
- Trends
- I/O Configurations

Each Logix object has an internal hierarchy of elements. Example: a task may contain one or more programs, each of which may contain one or more routines.

Every project has one controller. There may be one, many, or no instances of any type of Logix object in the project when the specific instance is created. This single instance of controller code is saved to an ACD file.



Traditionally, controller code was designed and configured for a specific project. In the library management workflow, content is not designed for a specific project, but to provide a widely applicable set of functions known as project components that are then used to create library objects. Each library object is an independent functional entity that can be easily configured to meet a wide range of applications and can be used in many projects.

## Library Designer

Use the Library Designer to assign the project, the controller, and any of the Logix objects to one or many library objects. Each library object defines a set of functions, capabilities, and connections. Example: those that support function of the valve, motor, and controller modules. Rather than being tied to one application, library objects can be configured to meet the needs of multiple applications. The Library Designer allows the publishing of a library directly into an ACM database. Options include the ability to specify the location where the library will be published in the ACM database, and the ability to specify the status of the library, either Published or Pending.

Custom properties called "Decorations" can be added to a library object using the Library Designer. Decorations include parameters, sub-objects, functions, substitutions, and external references. Decoration lets the library object be configured when it is implemented in a project in the ACM application.

Logix objects can be restricted to a single library object or assigned to multiple library objects, each with a different set of decorations. A library object can contain a single Logix object, or a Logix object can be added as an element of a more complex library object. Example: a P\_Alarm Add-On Instruction can be assigned to a valve library object and can also be an element of a Motor or Pump library object.

Each ACD file can support multiple projects, controller libraries, and library objects. The ACD is not required to contain a project or controller library. While decoration is stored as part of the ACD file, it is treated as a separate layer of information from the base controller code and does not affect code execution.

Decoration controls how the library object is instantiated, including configurations such as naming, tag values, conditional inclusion, and connections to other library objects. One or many distinct instances of a library object can be instantiated within an ACM project and each instance can be separately configured. Using Library Designer each Logix object can be published directly to the ACM database or to a file in HSL4 format.

## Library Object Manager

The library management workflow continues by opening the decorated ACD file in the Library Object Manager application. The Library Object Manager application can be used to publish each library object, either directly to the ACM database or to a file in HSL4 format. HSL4 files can be distributed individually or as part of a repository.

Use Library Object Manager to add HMI displays (FactoryTalk View SE/ME), FactoryTalk Alarms and Events configurations, FactoryTalk View ME Alarms and Historian (FactoryTalk Historian SE) components to the library object. This can only be done after the library object has been published from the ACD file to a folder or ACM database. The features added in the Library Object Manager application are saved to the individual HSL4 file or database entry for the library object and are not saved to the original ACD file.

Each library object file saved from the Library Object Manager application is classified within a four-level hierarchy:

*Solution -> Library Type -> Category -> Catalog Number*

Example: an analog input module might be classified as:

- **Solution: (RA-LIB) ACM**  
Solution will, in most cases, name the library object repository for the library object.
- **Library Type: Modules**  
Library Type is a general classification for the library object based on its function, such as module, control module or design pattern.
- **Category: Analog**  
Category is a more specific classification for the library object, based on its function.
- **Catalog Number: 1734-IE2C/C**  
The specific identifier for the library object.

Each library object file must have a distinct version number per solution. Just as the same Logix object can be used to create one or many library objects within the Library Designer, the same library object can be used to create one or many distinct library object files (versions) within the Library Object Manager application.

Library objects can be quickly distributed, then registered into and configured for multiple Projects in multiple locations. Library objects are available to any project that requires the functionality the library object provides.

Using Library Object Manager new library objects can be created and distributed rapidly to meet the needs of specific applications.

## FactoryTalk View Studio

Use FactoryTalk View Studio to create Site Edition (SE) and Machine Edition (ME) graphic displays. When the graphic displays are exported to XML they are called "Symbol objects." The XML files can be imported into the Library Object Manager application and the graphic displays added as non-Logix content to library objects.

## The Project Execution Workflow: Application Code Manager

In the Application Code Manager application, library objects become the building blocks used to rapidly create and deploy projects.

Execution is simply a matter of registering, adding, and configuring the library objects. Projects can be completed without requiring high-end programming support.

In the project execution workflow, library objects are selected in the ACM application and then the library object parameters are configured to meet the requirements of the current application. The workflow is complete when the Project to ACD controller code is created.

During the project, new library objects can be created using Library Designer, library objects from previous projects can be reused, or library objects can be shared from other databases.

After the project is complete, it can be used to create new library objects so that future projects can use the solutions developed.

### See also

[Application Code Manager Overview](#) on [page 7](#)

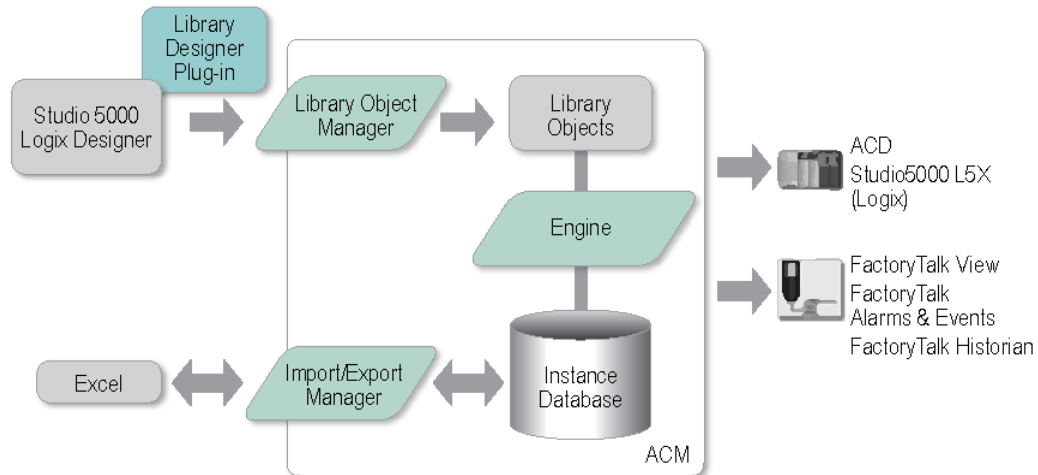
[Project Development](#) on [page 27](#)

## Library objects

A library object is the class definition of an object, it can contain links to other libraries. A library object is instantiated. When instantiating a library, all the linked libraries can be instantiated to new objects, or can continue to link to an existing instantiated object in the library.

Individual library object files (HSL4) are XML formatted and registered in the ACM database. A library object typically defines parameters, subclasses, user

interface contents, and portions of controller code (example: Logix) and HMI code (example: FactoryTalk View SE/ME).



Library objects contain controller code, as well as decoration. Decoration is a set of custom properties applied to a library object using the Library Designer. Decoration can be inherited from a library object that is higher in scope such as the Controller and Project object. Decoration that is applied to a library object is inherited by, or available to, all elements that are contained within the library object. Decoration can also be applied directly to an element, overriding inheritance from the library object and from library objects of higher scope.

ACM provides the option to include Project Data during the controller code generation. If this option is selected all instances with their parameter values, as well as all libraries (zipped) will be included as part of the Controller's Custom Properties.

Every element created and included by ACM will also have project data custom properties which includes information about the instance and the library that owns or created the element.

## See also

[Application Code Manager Overview](#) on [page 7](#)

[Registered Libraries](#) on [page 41](#)

## Templates

A template defines the static content and format of design output (example: a FactoryTalk View display). A template is not a class definition. A template is not instantiated. Templates have a variety of formats (example: xml, csv, docx, xlsx) and are stored in the ACM program folder or an individual user folder.

Before a template can be used the initial configuration of ACM must be completed, the local library must be registered, and the template installed.

## See also

[Application Code Manager Overview](#) on [page 7](#)

[Initial configuration of Application Code Manager](#) on [page 15](#)

[Local library and template file location](#) on [page 45](#)

[Register an ACM library object](#) on [page 42](#)

## Schedules

Use schedules to display or edit project data, typically parameter values.

Read-only schedules called "views" are temporarily generated for certain ACM reports (example: the **I/O Schedule** report).

The **Import Export Manager** tool can be used to export a schedule to a Microsoft® Excel® spreadsheet. Exporting a schedule to a spreadsheet can be useful for:

- Bulk additions, duplication, and changes
- Transferring project contents
- Snapshots
- Backups
- Version comparison

Import the schedule spreadsheet to ACM using the **Import Export Manager** tool.

## See also

[Application Code Manager Overview](#) on [page 7](#)

[Import Export Manager](#) on [page 61](#)

## Design Automation Concept

The project design outputs are generated automatically by ACM. The objects (instances) and parameter values, stored in the ACM database, are combined with various templates to create the following design outputs:

- Logix
- FactoryTalk View SE/ME graphics
- FactoryTalk Historian SE import file
- FactoryTalk Alarms and Events import file
- Excel (Schedules)

## See also

[Application Code Manager Overview](#) on [page 7](#)

## Create a central ACM database

When planning an ACM deployment if there are multiple people collaborating on projects make sure to select a computer to use as the ACM database server that can be accessed by all users in the project. This can be a standard computer that does not belong to a particular user or a project computer. The

computer must always be turned on and available. Microsoft SQL Server 2016 is the only software required on the shared computer.



Note: The ability to connect to a remote database is only available if the product has a Standard activation license and is not available in Lite mode.

Install SQL Server 2016 via the ACM installation media. Select only SQL Server 2016 when presented with the selection of install options.

When configuring the SQL Server note the following considerations when supporting multiple user connections to the database:

1. **Add Users:** Normally ACM uses Windows Authentication to connect to the ACM database. When the ACM database is located in a remote computer, local users must be created using the SQL Server Management Studio. Create one user account that is shared by all the project collaborators or add an individual user for each collaborator. Only assign users to the ACM database after the ACM database is created.

If the SQL user is not a sysadmin, creation of the database for the first time is a multi-step process:

- a. Create users in SQL.
  - Minimum **Server Roles** should be:
    - dbcreator
    - public
  - Permissions on **Securables** should be:
    - Connect SQL
    - Control Server
    - View any database (optional, but if not, the database needs to be mapped to this user in SQL)
- b. Launch the database manager in ACM and create the database using the new user.

Creation of this database must be done from ACM and cannot be done manually in Microsoft SQL Server.

- c. In order for the user to connect and update the database, add the newly created database to the User Mapping in SQL and set the memberships on the database to:
  - db\_owner
  - db\_datareader
  - db\_datawriter
  - public
- d. In order for the user to back-up the database, add the newly created database to the User Mapping in SQL and set the memberships on the database to:
  - db\_backupoperator

2. **Record Database Connection Information:** Record the SQL Server computer name and/or computer IP address, the SQL Server instance name, the SQL Server authentication (username and password), and the ACM database name. This information is required by ACM users attempting to connect to the central ACM database.

### See also

[Initial configuration of Application Code Manager](#) on [page 15](#)

[Connect to an ACM database](#) on [page 21](#)

## Upgrading a central ACM database

After installing a new version of ACM, the database may sometimes require upgrading due to the adding of new tables or fields. You will be notified to upgrade the database if applicable.

The failure of the upgrade process could be related to you having insufficient permissions of the central server to execute the upgrade commands. The upgrade process could also fail due to the requirement of additional prerequisite software needed on the SQL server host machine.

If upgrading ACM from v2.xx to v4.xx then this error is likely due to the x64 versions of the following components not being installed:

- Microsoft System CLR Types for SQL Server 2012 (x64)
- Microsoft SQL Server 2012 Management Objects (x64)

It is recommended that when installing a new version of ACM on a client machine on which the same version of ACM is also installed on the SQL server host machine to ensure all prerequisite software is also installed.

### See also

[Create a central ACM database](#) on [page 13](#)

[Connect to an ACM database](#) on [page 21](#)

## Initial configuration of Application Code Manager

Before using Application Code Manager, initial configuration must be completed. In this procedure, a database is created, the database connection method is specified, and the default ACM libraries are registered.



Tip: If the SQL Server installation option was selected, this procedure was completed automatically during installation.

### To complete the initial configuration of Application Code Manager

1. Open Application Code Manager. On the main menu, click **Tools > Database Manager**.

- In **Server Name**, select the computer name and SQL Server instance from the drop down list or type a computer name and SQL Server instance in the following format:

<Computer Name> \ <SQL Server Instance>

- In **Log on to the server**, type the **User name** and **Password** to use to authenticate the connection to the SQL Server, then click **Connect**.

**Status** updates to **Connected**.

- In **Specify the database**, type a unique name for the ACM database.
- In **Actions**, select **Create database** and then click **Execute Task**.

Once the database is created an **Action completed successfully** message displays. Click **OK**.

- Click **Close** to close **Database Manager**.



Tip: The Application Code Manager title bar displays the application icon and application name followed by the computer name, SQL Server instance, and database name formatted as:

<Computer Name> \ <SQL Server Instance>.<DataBase Name>

If Application Code Manager is not able to connect to a database the title bar displays (Not Connected).

- In Application Code Manager, right click **Registered Libraries** and then select **Register**.

The **Open** dialog box displays the default library file location.

C:\User\Public\Public Documents\Rockwell Automation\Studio 5000\Libraries\Application Code Manager

- Open the **(RA-LIB) ACM** folder. Select all the objects in the folder and then click **Open**.

- The **Libraries Registration** window opens and displays the status of the library registration process.

Review any errors or warnings that were encountered.



Tip: The results of the registration process are also saved to a log file. Click **Show Log File** to view the contents of the Log file.

- Click **Finish** to close the **Libraries Registration** window.

The registered libraries display Application Code Manager under **Registered Libraries**.

## See also

[Create a central ACM database](#) on [page 13](#)

[Database Manager settings](#) on [page 77](#)

## Upgrade the Application Code Manager application

When upgrading the Application Code Manager application, consider updating the libraries used in the ACM projects for a particular ACM database and obsoleting the previous version libraries.

---

**IMPORTANT** It is recommended to back up the current ACM database using the **Database Manager** before performing an upgrade. With the **Export Used Libraries** option enabled, export all projects using the **Import Export Manager** can also back up database.

---



## To upgrade the Application Code Manager application

1. Install a newer version of Application Code Manager.
2. Open Application Code Manager. A message appears with a prompt to upgrade the database. Click **OK**. The **Database Manager** displays.
3. Select **Upgrade database** in the **Actions - Tasks** area and then click **Execute Task**.

Click Close to close **Database Manager**.

4. Right click **Registered Libraries** and then select **Register** to ensure any new libraries are added to the database.

In the **Open** dialog box, browse the default library file location.

C:\User\Public\Documents\Studio 5000\Libraries\Application Code Manager

5. Open each folder. Select all the objects in the folder and then click **Open**.
6. The **Libraries Registration** window opens and displays the status of the library registration process.

Review any errors or warnings that were encountered.



Tip: Results of the registration process are also saved to a log file. Click **Show Log File** to view the contents of the Log file.

7. Click **Finish** to close the **Libraries Registration** window.

The registered libraries display in Application Code Manager under **Registered Libraries**.

### See also

[Import Export Manager](#) on [page 61](#)

## Navigate the Application Code Manager user interface

Application Code Manager is composed of a menu bar, a toolbar, and several different panes that navigate through different objects. The objects (instances) contained in a project are arranged in a hierarchy. There are three ways to view the hierarchy, which are accessed by clicking one of the three panes (System View, Controller Preview, and Class View). Hide or float the panes so that they can be arranged according to preference.

This table provides an overview of the user interface panes.

Pane	Description
System View	<p>A pane that contains a tree control that enables navigation through the different components of an ACM project.</p> <p>The System View pane displays branches for the objects contained in the project including FactoryTalk Historian Server objects, FactoryTalk View SE/ME Server objects, Libraries used in the project, and Controller objects.</p> <ul style="list-style-type: none"> <li>• <b>Historian branch</b> The Historian branch contains all Historian ScanClass objects in the Project. The Historian objects are organized in a 3-level hierarchy: <i>Historian</i> --&gt; <i>ScanClass</i> ----&gt; <i>Object</i> The Historian Scan Class definitions are Sub-objects in the FT_Historian object (instance).</li> <li>• <b>HMI branch</b> The HMI branch contains all HMI objects in the Project. HMI objects are organized in a 3-level hierarchy: <i>HMI</i> --&gt; <i>Displays</i> ----&gt; <i>Object</i> <i>HMI</i> --&gt; <i>Colors</i> ----&gt; <i>Object</i> and <i>HMI</i> --&gt; <i>Alarms</i> ----&gt; <i>Object</i></li> <li>• <b>Used Libraries branch</b> The Used Libraries branch displays the libraries that are used in the project. Libraries are organized in a 4-level hierarchy: <i>Solution</i> --&gt; <i>Library Type</i> ----&gt; <i>Library Category</i> -----&gt; <i>Library Catalog Number (Library Version)</i></li> </ul>
Controller Preview	<p>A pane that contains a tree control that enables navigation through the different controllers.</p> <p>The Controller Preview pane displays all the project data organized into the following folders:</p> <ul style="list-style-type: none"> <li>• Controllers</li> <li>• Controller-specific data</li> <li>• Logix-specific objects</li> <li>• Task/Programs</li> <li>• Control Modules</li> </ul> <p>Controller Preview displays Logix content similar to its final state after it is generated. Use this view to add objects, as well as generate code.</p>
Class View	<p>A pane that contains a tree control that enables navigation through objects grouped by controller.</p> <p>Objects are organized by the Library object catalog numbers and show the instances below them. Use this view to add, copy, or delete any object instances as well as generate code and reports or navigate the library in the <b>Registered Libraries</b> tree.</p>
Object Identifiers	<p>Displays the identification information for the currently selected object. Includes</p> <ul style="list-style-type: none"> <li>• <b>Name.</b> The name of the object in the database</li> <li>• <b>Description.</b> Object category or user specified label for the object.</li> <li>• <b>Catalog Number.</b> The specific identifier assigned to the object in the library.</li> <li>• <b>Solution.</b> The name of the library object repository for the library object</li> </ul>

Pane	Description
Object Parameters	<p>This pane displays the properties tab for the object currently selected in either the <b>System</b>, <b>Controller</b>, or <b>Class</b> view.</p> <p>When applicable, a toolbar is present in the parameter pane that provides the ability to:</p> <ul style="list-style-type: none"> <li>• Sort parameters alphabetically</li> <li>• Sort parameters into groups</li> <li>• Show only visible parameters</li> <li>• Show all (visible and hidden) parameters</li> <li>• Open additional properties</li> </ul> <p>The parameter name shows in the column on the left and the parameter value shows in the column on the right. Change parameter values by typing a new value in the right column. A description of the selected parameter displays at the bottom of the parameters tab.</p> <p>Some objects have additional tabs for sub-object parameters, (example: analog input of a 1756-IF16 or Attachments for libraries). Sub-object parameters show on an additional tab labeled with the sub-object display name.</p> <p>Each row in the sub-object parameters tab represents a sub-object. By default, sub-objects sort alphabetically by name. Sort sub-objects in groups by clicking on a column header.</p> <p>The sub-object name shows in the <b>Name</b> column. Additional columns display the sub-object parameters (example: Channel). Change the sub-object name and the sub-object parameter values by typing a new value below the column header.</p> <p>The sub-object name can be configured as read-only. When the sub-object name is read-only, the value is shown dimmed.</p> <p>If an object or sub-object parameter value is changed click <b>Apply changes</b> to save the changes to the ACM database.</p> <p>The version number of the Application Code Manager software displays in lower left corner of the object <b>Parameter</b> tab.</p>
Registered Libraries	<p>A pane that contains a tree view display of all libraries in the connected ACM database. These libraries can be added to an ACM project.</p> <p>The libraries are organized in a 4-level hierarchy with statistical information in parenthesis:</p> <p><i>Solution (Number of objects)</i>  <i>--&gt; Library Type (Number of objects)</i>  <i>----&gt; Library Category (Number of objects)</i>  <i>-----&gt; Library Catalog Number (Library Version)</i></p>
Library Repositories	<p>This pane displays the configured library repositories. A library repository refers to another ACM database that can monitored for library updates. Libraries from these source databases can be easily replicated over to the currently active database for use within a project. It is unavailable to create an object directly from a library in a library repository. The library must first be replicated to the current active ACM database before it can be used to create an object. Add or remove library repositories in this pane.</p>

## See also

[Create a new project](#) on [page 28](#)

[Add a Historian object](#) on [page 29](#)

[Add a new HMI object](#) on [page 33](#)

[Update a project library](#) on [page 37](#)



## Connect to an ACM database

### Connect to an ACM database

To use Application Code Manager it must be connected to an ACM database. During the initial configuration of Application Code Manager the default database was configured. Use this procedure to open the **Connection** properties sheet and update the database connection configuration if the original ACM database server becomes unavailable, if the authentication method needs to be modified, or if a different database is to be used.

### To connect to an ACM database

1. Open Application Code Manager. On the main menu, click **File > Connect**.
2. In **Data source** confirm that the database type is **Microsoft SQL Server (SqlClient)**.
3. In **Server Name**, select the computer name and SQL Server instance from the drop down list or type a computer name and SQL Server instance in the following format:

<Computer Name> \ <SQL Server Instance>



Tip: If the database is located on the same computer as Application Code Manager, type **localhost**.

4. In **Log on to the server**, either:
  - Choose **Use Windows Authentication** to allow SQL Server log on using the logged on user account credentials.
  - Choose **Use SQL Server Authentication** to allow SQL Server log on using SQL Server authentication.

If this method is selected then type the **User name** and **Password** to use to authenticate the connection to the SQL Server. If the SQL database is created by the ACM installer then the user name is "sa" and the default password is "ApplicationAdmin".



Tip: Select **Save my password** to allow Application Code Manager to retain the password and log on automatically.

5. In **Connect to a database**, determine whether to connect to a database on the server or to attach a database file.
  - Choose **Select or enter a database name** to select a database name from a pull-down list or enter a database name. The database must exist on the database server.
  - Choose **Attach a database file** and then type a database file name or click **Browse** to navigate to the file.

Then, in **Logical name** type the name of the database within the file that Application Code Manager should connect to.

6. (optional) Click **Advanced** to configure settings related to connection protocols, resiliency, content, initialization, pooling, replication, and additional security controls. Applying the recommended settings as defined on the help page for the Advanced Properties will improve performance, especially for network connections.
7. Click **Test Connection** to verify the connection settings specified. If the connection succeeds, click **OK** to close **Connection Properties**.

### See also

[Connection Properties settings](#) on page 22

[Advanced Properties settings](#) on page 23

[Initial configuration of Application Code Manager](#) on page 15

## Connection Properties settings

How do I open the Connection Properties dialog box?

- On the main menu, click **File >Connect**.

The **Connection Properties** dialog box defines the settings for how Application Code Manager connects with the SQL Server database. Use this dialog box to set the data source, the type of user authentication, to specify a different database file, or to modify advanced security settings.

This table describes the settings in the **Connection Properties** dialog box.

Setting	Description
Data source:	Database type. Always select Microsoft SQL Server (SqlClient)
Server name:	Selects a computer name and SQL Server instance from the list or type a computer name and SQL Server instance in the following format: <Computer Name> \ <SQL Server Instance>
Log on to the server	
Use Windows Authentication	Allows SQL Server log on using Windows authentication. When selected the logged on Windows user account credential will be sent to SQL Server to authenticate the session.
Use SQL Server Authentication	Allows SQL Server log on using SQL Server authentication. When selected the user name and password must be provided for authentication by the SQL Server. <ul style="list-style-type: none"> <li>• <b>User name.</b> The SQL Server user name, "sa" by default.</li> <li>• <b>Password.</b> The SQL Server password associated with the user name specified, "ApplicationAdm1n" by default.</li> <li>• <b>Save my password.</b> When selected, saves the SQL Server password specified so that it can be used in the next session.</li> </ul>
Connect to a database	
Select or enter a database name:	Select a database name from the list or enter a database name.
Attach a database file:	When selected specify the identifiers for the database file. SQL Server database files have two names, the operating system file name used to locate the database in the file system and the logical file name used to identify the database within SQL Server transactions. <ul style="list-style-type: none"> <li>• Type a database file name or use the <b>Browse</b> button to use the <b>Open</b> dialog to locate the database file by clicking through the file system.</li> <li>• In <b>Logical name</b>, type the logical name of the database.</li> </ul>
Advanced	Select to configure advanced database properties.

Setting	Description
Test Connection	<p>Tests the connection to the database.</p> <p>If a "Test connection succeeded." message is not returned, check that the following settings are correct:</p> <ul style="list-style-type: none"> <li>• Computer name</li> <li>• SQL Server authentication</li> <li>• Network access (remote SQL Server)</li> </ul>

## See also

[Initial configuration of Application Code Manager](#) on [page 15](#)

[Advanced Properties settings](#) on [page 23](#)

[Connect to an ACM database](#) on [page 21](#)

[Create an ACM database](#) on [page 73](#)

## Advanced Properties settings

How do I open the Advanced Properties dialog box?

1. On the main menu, click **File >Connect** to open **Connection Properties**.
2. Click **Advanced**.

The **Advanced Properties** dialog box provides a means of changing how the connection between ACM and the SQL Server passes information.

This table describes the settings in the **Advanced Properties** dialog box. The dialog box is divided into functional areas.



Note: Applying the recommended settings will improve ACM performance especially for network connections.

Area	Setting	Possible Values	Description
<b>Advanced</b>	MultipleActiveResultSets	True False (default)	When True, multiple result sets can be returned and read from one connection.
	Network Library	<b>blank (required if local)</b> Named Pipes (DBNMPNTW) Shared Memory (DBMSLPCN) <b>TCP/IP (DBMSSOEN)</b> <b>(recommended if networked)</b> VIA (DBMSGNET)	The network library used to establish a connection to an instance of SQL Server. <b>Do not use when the SQL Server is resident on the local host computer, value should be blank.</b>
	Packet Size	<b>8000 (recommended)</b>	Size in bytes of the network packets used to communicate with an instance of SQL Server. PacketSize may be a value in the range of 512 and 32767 bytes.
	Transaction Binding	Implicit Unbind (default) Explicit Unbind	Indicates the binding behavior of connection to the <b>System.Transactions</b> namespace. When set to <b>Implicit Unbind</b> , the connection detaches from the transaction when it ends, switching back to autocommit mode. When set to <b>Explicit Unbind</b> the connection remains attached to the transaction until the transaction is closed. The connection will fail if the associated transaction is not active or does not match the current transaction.

	Type System Version	Latest (default) SQL Server 2012 SQL Server 2008 SQL Server 2005	Indicates which server type system the provider will expose through the DataReader.
<b>Connection Resiliency</b>	ConnectRetryCount	<b>2 (recommended)</b>	Number of attempts to restore a connection. The number of reconnections attempted after identifying that there was a connection failure. This must be an integer between 0 and 255. Set to 0 to disable reconnecting on idle connection failures.
	ConnectRetryInterval	<b>5 (recommended)</b>	Delay between attempts to restore connection. The amount of time (in seconds) between each reconnection attempt after identifying that there was a connection failure. This must be an integer between 1 and 60.
<b>Context</b>	Application Name	.Net SqlClient Data Provider	The name of the application.
	Workstation ID		The name of the workstation connecting to SQL Server.
<b>Initialization</b>	ApplicationIntent	ReadWrite (default) ReadOnly	Declares the application workload type when connecting to a server.
	Asynchronous Processing	True False (default)	When true, enables usage of the Asynchronous functionality in the .Net Framework Data Provider.
	Connect Timeout	<b>30 (recommended)</b>	The length of time in seconds to wait for a connection to the server before terminating the attempt and generating an error. A value of 0 indicates no limit, and should be avoided in a ConnectionString because an attempt to connect waits indefinitely.
	Current Language		The SQL Server Language record name.
<b>Pooling</b>	Enlist	True (default) False	When <b>True</b> sessions in a Component Services environment should automatically be enlisted in a global transaction where required.
	Load Balance Timeout	30 (default)	The minimum amount of time (in seconds) for this connection to live in the pool before being destroyed.  When a connection is returned to the pool, its creation time is compared with the current time, and the connection is destroyed if that time span (in seconds) exceeds the value specified by <b>Load Balance Timeout</b> .  A value of zero (0) causes pooled connections to have the maximum connection timeout.
	Max Pool Size	<b>1000 (recommended)</b>	The maximum number of connections allowed in the pool. Valid values are greater than or equal to 1. Values that are less than <b>Min Pool Size</b> generate an error.
	Min Pool Size	1 (default)	The minimum number of connections allowed in the pool. Valid values are greater than or equal to 0. Zero (0) in this field means no minimum connections are initially opened. Values that are greater than <b>Max Pool Size</b> generate an error.
	PoolBlockingPeriod	Auto AlwaysBlock <b>NeverBlock (recommended)</b>	Defines the blocking period behavior for a connection pool. When connection pooling is enabled and a timeout error or other login error occurs, an exception will be thrown and subsequent connection attempts will fail for the next five seconds, the "blocking period". If the application attempts to connect within the blocking period, the first exception will be thrown again. Subsequent failures after a blocking period ends will result in a new blocking period that is twice as long as the previous blocking period, up to a maximum of one minute.
	Pooling	<b>True (recommended)</b> False	When <b>True</b> , the connection object is drawn from the appropriate pool, or if necessary, is created and added to the appropriate pool. Any newly created connection is added to the pool when closed by the application. In the next attempt to open the same connection, that connection will be drawn from the pool.  Connections are considered the same if they have the same connection string. Different connections have different connection strings.



<b>Replication</b>	Replication	False (default) True	Used by SQL Server in replication. Set to <b>True</b> if replication is supported using the connection.
<b>Security</b>	Authentication	NotSpecified (default) SqlPassword ActiveDirectoryPassword ActiveDirectoryIntegrated	Specifies the method of authenticating with SQL Server.
	Column Encryption Setting	Enabled Disabled (default)	Default column encryption setting for all the commands on the connection.
	Encrypt	True False (default)	When <b>True</b> , SQL Server uses SSL encryption for all data sent between the client and server if the server has a certificate installed.
	Integrated Security	True False (default)	Whether the connection is to be a secure connection or not. When <b>False</b> , User ID and Password are specified in the connection. When <b>True</b> , the current Windows account credentials are used for authentication.
	Password	*****	Indicates the password to be used when connecting to the data source.
	Persist Security Info	True False (default)	When <b>False</b> , security-sensitive information, such as the password, is not returned as part of the connection if the connection is open or has ever been in an open state.
	TrustServerCertificate	<b>True (recommended)</b> False	When <b>True</b> (and <b>Encrypt</b> is set to <b>True</b> ), SQL Server uses SSL encryption for all data sent between the client and server without validating the server certificate. If <b>TrustServerCertificate</b> is set to <b>True</b> and <b>Encrypt</b> is set to <b>False</b> , the channel is not encrypted.
	User ID	sa	Indicates the user ID to be used when connecting to the data source.
<b>Source</b>	AttachDbFilename		The name of the primary file, including the full path name, of an attachable database.
	Context Connection	True False (default)	When <b>True</b> , indicates the connection should be from the SQL Server context. Available only when running in the SQL Server process.
	Data Source	localhost\SQLACM (default)	Indicates the name of the data source to connect to.
	Failover Partner		The name or network address of the instance of SQL Server that acts as a failover partner.
	Initial Catalog	<i>Initial Database Name</i>	The name of the initial catalog or database in the data source.
	MultiSubnetFailover	True False (default)	If your application is connecting to a high-availability, disaster recovery (AlwaysOn) availability group (AG) on different subnets, setting this value to <b>True</b> configures SqlConnection to provide faster detection of and connection to the (currently) active server.
	TransparentNetworkIPResolution	True (default) False	If your application connects to different networks, setting this value to <b>True</b> configures SqlConnection to provide transparent connection resolution to the currently active server, independently of the network IP topology. When set to <b>True</b> , the application is required to retrieve all IP addresses for a particular DNS entry and attempt to connect with the first one in the list. If the connection is not established within 0.5 seconds, the application will try to connect to all others IP addresses in parallel. When the first IP address answers, the application will establish the connection with the respondent IP address. If <b>MultiSubnetFailover</b> is set to <b>True</b> , this setting is ignored. If <b>Failover Partner</b> is specified, this setting is ignored. The default setting is <b>False</b> if <b>Authentication</b> is set to either <b>Active Directory Password</b> or <b>Active Directory Integrated</b> , otherwise the default setting is <b>True</b> .
	User Instance	True False (default)	Indicates whether the connection will be re-directed to connect to an instance of SQL Server running under the user's account.

## See also

[Connection Properties settings](#) on [page 22](#)

[Import Export Manager](#) on [page 61](#)

## Project Development

### Project Development

Application Code Manager organizes components into projects. Use one of the following methods to create projects:

- Add a new project using the **Object Configuration Wizard**
- Create a new project from an existing project
- Import a project using ACM or ACM Console

Once a project is created, add objects for your application, such as:

- Historian objects
- HMI objects
- Alarm objects
- Controllers and controller objects

### See also

[Create a new project](#) on [page 28](#)

[Create a new project from an existing project](#) on [page 29](#)

### Project commands

In the **System View** pane, right-click the **Project** branch to view the project commands. Different branches have different commands available.

This table describes each project command.

Command	Branch	Description
View > Project History	Project (top)	Displays a report showing the Project History.
Export	Project (top), Used Libraries, Library Module	Starts the <b>Import Export Manager</b> .
Import	Project (top)	Starts the <b>Import Export Manager</b> .
Delete	Project (top), Historian object, Displays object, Alarms object	Deletes the selected object. Note: The Projects Delete dialog box supports multiple select.
Rename	Project (top), Historian object, Displays object, Alarms object	Renames the selected object.
Refresh	Project (top), Historian, ScanClass, Historian object, HMI, Displays, Displays object, Alarms, Alarms object	Refreshes the tree view.
Update All Child Objects	Historian, HMI	Opens the <b>Update Used Libraries</b> window.
Add	ScanClass, Colors, Displays, Alarms	Adds a new object (instance) to the selected Category.
Paste Special	ScanClass, Colors, Displays, Alarms	Displays PasteSpecial window where child objects and reference values can be included or excluded.
Paste	ScanClass, Colors, Displays, Alarms	Pastes a copied object in the selected location.
Copy	Historian object, Displays object	Copies the selected object.

Command	Branch	Description
Update	Used Libraries, Library Module	Displays the Update Used Library dialog.
Generate Historian	Historian object	Generates a copy of the selected Historian object from ACM. The Historian object is saved to an external folder.
Generate Displays	Displays Object	Generates a copy of the selected Displays object from ACM. The Display object is saved to an external folder.
Generate Alarms	Alarms Object	Generates a copy of the selected Alarms object from ACM. The Alarms object is saved to an external folder.
Generate Documentation	Project (top)	Generates the selected template file against the project object from ACM. As these templates are often custom built, generating at this level may not result in the desired outcome. The resulting file is saved to an external folder.
View > Project Library Usage Count	Used Libraries	Displays the <b>Project Library Usage</b> report for the selected object.
View > Project Library per Solution	Library Solution	Displays the <b>Project Library Usage per Solution</b> report for the selected object.
View > Project Library Usage per Library Type	Library Type	Displays the <b>Project Library Usage per Solution, Library Type</b> report for the selected object.
View > Project Library Usage per Category	Library Category	Displays the <b>Project Library Usage per Solution, Library Type, Category</b> report for the selected object.
View > Project Library Usage per Library	Library Object	Displays the <b>Project Library Usage per Library</b> report for the selected object.

## See also

[Create a new project on page 28](#)

[Add ACM library content to an ACD project on page 107](#)

[Add a Historian object on page 29](#)

[Add an HMI object on page 33](#)

## Create a new project

Create a new project when the new project is not similar to existing projects.

### To create a new project

1. Click **File** point to **New** and then click **Project**. The **Object Configuration Wizard** is displayed.
2. On the **Select a library** page click the + symbol to expand the library category and display the libraries registered in the connected ACM database.
3. Click a library to select it and then click **Next**.
4. If the library contains a library link, click the **Linked Libraries** tab.
5. Click the link and choose how to instantiate the linked library, select either:
  - a. **Create New Instance**. To create a new instance of the library the link is attached to.
  - b. **Link to Existing Instance**. To link to a library instance of the target library that has already been created.

6. Enter a unique name in the **Name** field. Enter a description in the **Description** field and click **Finish**.

The new project appears highlighted in the **System View** pane.

### See also

[Initial configuration of Application Code Manager](#) on [page 15](#)

[Create a new project from an existing project](#) on [page 29](#)

[Add a Historian object](#) on [page 29](#)

[Add an HMI object](#) on [page 33](#)

## Create a new project from an existing project

Use an existing project in the ACM database as the basis for a new project if the projects use the same components and libraries.

If the project is in a different ACM database, such as one from another installation, you must connect to that database, export the project, and then import it to the current database.

### To create a new project from an existing project

1. Click **File** point to **New** and then click **Project from Existing Project**. The **New Project Wizard** is displayed.
2. On the **Create a new Project from an existing Project** page in **From Project** type a Project name or select the project name from the list.
3. In **New Name**, type a name for the new project.
4. (optional) In **New Description**, type a description of the new project.
5. Click **Finish**.

The new project appears highlighted in the **System View** pane.

### See also

[Connect to an ACM database](#) on [page 21](#)

[Add a Historian object](#) on [page 29](#)

[Add an HMI object](#) on [page 33](#)

## Add a Historian object

Add a Historian object to the project to support integration of Historian data.

### To add a Historian object

1. In the **System View** pane, expand **Historian**, right click **ScanClass** and then click **Add**.

The **Object Configuration Wizard** appears.

2. Under **Select a library** in the **Solution** column, click the + symbol to expand a library category and display the Historian libraries registered in the connected ACM database.
3. Click a Historian library, the row highlights to indicate that it is selected. Click **Next**.
4. In **Name**, type a unique name for the Historian object and then click **Finish**.

The Historian object is added to the **System View** pane as a child object of **ScanClass**.

### See also

[Add a Historian sub-object](#) on [page 30](#)

[Historian Point Type parameters](#) on [page 31](#)

[Generate a Historian object](#) on [page 32](#)

[Update a project library](#) on [page 37](#)

## Add a Historian sub-object

Historian objects have sub-objects that represent the point types configured. A FactoryTalk Historian point is the basic building block for controlling data flow to and from the FactoryTalk Historian SE server. A single point is configured for each measurement value that needs to be archived.

### To add a Historian sub-object

1. In the **System View** pane, expand **Historian**, expand **ScanClass** and then click the Historian object to which the sub-object will be added.
2. In the object parameters pane, select the **Point Type** tab.
3. Right-click in the white space below the objects shown in the tab. Click **Add New**.

Repeat this step to add all of the points that require configuration.

4. Click the appropriate column and modify the parameters as needed.
5. Click **Apply Changes** to save the parameter configuration.

### Group sub-objects

Group sub-objects into categories based on the data columns.

- Double-clicking on a column heading updates the display in the parameters tab to group items with the same value together.
- Right-click anywhere in the parameters tab and select **Reset Grouping** to return to the default display.

## See also

[Add a Historian object](#) on [page 29](#)

[Historian Point Type parameters](#) on [page 31](#)

[Generate a Historian object](#) on [page 32](#)

## Historian Point Type parameters

The **Point Type** tab displays a columnar grid of the parameters defined for the points of the Historian object.

This table describes the parameters in the **Point Type** tab of the Historian object.

Parameter	Description
Name	The name of the point. Double-click to configure a unique name for the point.
ScanClassNo	The scan class determines the frequency at which input points are scanned for new values.
ExcDev	Exception Deviation. Specifies in engineering units how much a point's value must change before the interface considers it a significant value, and sends it to the server. As a general rule, you should set the exception slightly smaller than the precision of the instrument system.
ExcDevPercent	Specifies the exception deviation as a percentage of span instead of in engineering units. For digital, string and Blob tags, ExcDev and ExcDevPercent are ignored and display by applications as zero.
ExcMax	Exception Maximum. Specifies a limit on how long the interface can go without reporting a value to the Historian server. After the ExcMax time period elapses, the interface sends the next new value to the server, regardless of whether the new value is different from the last reported value.
ExcMin	Exception Minimum. Specifies a limit on how frequently the interface can report values to the server. Example: For the interface to wait a full ten minutes before reporting a new value to the server, set the ExcMin attribute to ten minutes. ExcMin is typically set to zero.
CompDev	Compression Deviation. Specifies in engineering units how much a value may differ from the previous value before it is considered to be a significant value. In most environments, set CompDev to the precision of the data source or hardware (instrument). Initially, set it to a lower value so that important data is not lost. After collecting data for a while, go back and check the data for your most important tags, and then adjust CompDev to a higher value, if necessary. Setting the CompDev attribute value too low causes too little data compression, and wastes space in the archive. Setting the value too high causes loss of useful data. For most flows, pressures, and levels, use a deviation specification of 1% or 2% of span. For temperatures, the deviation should usually be 1 or 2 degrees.
CompDevPercent	Specifies the compression deviation as a percentage of span instead of in engineering units. <b>Note:</b> For non-numeric tags, CompDev and CompDevPercent are ignored. They will be displayed by applications as zero.
CompMax	Compression Maximum. A point is archived if the elapsed time since the previous time the point was saved is greater than the maximum time. The recommended maximum time specification is one work shift (example: 8 hours). Duplicate values will be archived if the elapsed time exceeds CompMax. In most environments, set CompMax to the same value for all points in the system.
CompMin	Compression Minimum. A point is archived if the elapsed time since the previous time the point was saved is greater than or equal to the minimum time, and the value has changed by more than the deviation. For data points associated with interfaces that send exception reports, set CompMin to 0.
Compressing	Determines whether compression is turned on or off for the point. Compression should be turned on for all real-time points in the system. Set compression OFF for laboratory and manually entered tags so every value is recorded in the archive. To turn compression on, select the Compressing parameter for most points. With compression off, every value sent is saved in the archive. Compression affects digital points, since a new value is recorded only when the current value changes. Points of types Blob and string have a similar behavior; new events pass compression only when the value changes. String values are compared ignoring case. Example: "VaLuE" and "vaLue" are evaluated as equal. For Blob events, any change is significant.

Parameter	Description
Step	<p>The Step parameter affects only numeric points. It defines how numeric archived values are interpolated.</p> <ul style="list-style-type: none"> <li>When the checkbox is clear (OFF) archived values for the point are interpreted as a continuous signal and adjacent archived values are linearly interpolated. Example: at 12:00:00, the value 101.0 is archived and at 12:01:00, the value 102.0 is archived. A request for the archive value at 12:00:30 would return 101.5.</li> <li>When the checkbox is selected (ON) archived values for the point are interpreted discretely and adjacent archived values are not interpolated. An archived value is assumed constant until the next archived value is recorded. Example: at 12:00:00, the value 101.0 is archived, at 12:01:00, the value 102.0 is archived. A request for the value at 12:00:30 would return 101.0.</li> </ul> <p>Data coming from continuous signals (such as signals from thermocouples, flow meters, and other devices that provide continuous measurement) should be archived in points with the step flag OFF.</p> <p>Data coming from discrete measurements (such as sampled lab data, batch charge weight, and other user-defined inputs) should be archived in points with the step flag ON.</p> <p>In addition, the step flag affects the compression calculation. When it is ON a linear change of value greater than or equal to CompDev passes compression. When the step flag is OFF the complete compression algorithm is applied.</p>
DigitalSet	For digital points, the DigitalSet attribute specifies the name of the digital state set associated with the tag. The DigitalSet attribute has no meaning for non-digital tags.
Scan	<p>Toggling this checkbox, turns scanning ON or OFF for a point. By default scanning is turned ON (selected), which indicates that the program should be able to collect data for the point.</p> <p>Setting the Scan attribute to OFF (cleared) turns off data collection for that point.</p>
AdviseMode	Specifies that the point should run in the Advised data collection mode. In this mode, data is collected only when a value changes in the controller. It is not based on the scan rate. AdviseMode is selected by default as this mode is the most efficient because data is sent to the Historian server only when the value changes.
SubObject Description	Descriptive text that distinguishes this point from other points.

**See also**

[Add a Historian object](#) on [page 29](#)

[Add a Historian sub-object](#) on [page 30](#)

[Generate a Historian object](#) on [page 32](#)

**Generate a Historian object**

Once a Historian Object has been configured it can be used to generate a FactoryTalk Historian SE import file, this file is a comma-separated value (.csv) file.

**To generate a Historian object file**

1. In the **System View** pane, expand the **Historian** folder and the **ScanClass** folder so that the Historian objects are visible.
2. Right-click the Historian object to export and then choose the controller for which support is required.
  - Click **All Controllers** to create an export file that supports all controllers in the project.
  - Click a specific controller from the list if this object will only be used with that type of controller.

The **Save As** dialog box opens.



3. Use the tree control to browse to the location in which to save the Historian object file or use the default location:  
`C:\Users\\Documents\Studio 5000\Projects`  
 The variable <user name> is replaced by the logged in user account name.
4. In **File name** type a name for the file or use the default file name which is created according to the following pattern:  
`<Controller Name>_<Historian object Name>.csv`  
 The variable <Controller Name> is replaced by the name of the controller selected if a specific controller support was selected. If All Controllers are supported the controller portion of the default file name and the underscore are omitted.
5. Click **Save**. The Historian object generation is started.
6. Once the process is completed, the **Generation Complete** status dialog box appears, click **OK** to close the dialog box.

### See also

[Add a Historian object](#) on [page 29](#)

[Generate an HMI object](#) on [page 35](#)

## Add an HMI object

Add an HMI object to support FactoryTalk View SE, FactoryTalk View ME, or Alarms displays in the project.

### To add an HMI object

1. In the **System View** pane, expand **HMI** to see the HMI categories.
  - To add a FactoryTalk View SE or FactoryTalk View ME HMI object, right-click **Displays** and then click **Add**.
  - To add a FactoryTalk Alarms and Events object or FactoryTalk View ME alarm object, right-click **Alarms** and then click **Add**.

The **Object Configuration Wizard** appears.

2. Under **Select a library** in the **Solution** column, click the + symbol to expand a library category and display the libraries registered in the connected ACM database.
3. Click a library, the row highlights to indicate that it is selected. Click **Next**.
4. In **Name**, type a unique name for the object and then click **Finish**.  
 The object is added to the **System View** pane as a child object of the HMI category.

## See also

[Add a Display sub-object](#) on [page 34](#)

[HMI Display parameters](#) on [page 34](#)

[Add an Alarm Group sub-object](#) on [page 36](#)

[FTAE Alarm Group parameters](#) on [page 37](#)

[Generate an HMI object](#) on [page 35](#)

[Update a project library](#) on [page 37](#)

## Add a Display sub-object

FactoryTalk View objects have sub-objects that represent the display types configured. A FactoryTalk View graphic display represents a run-time operator's view of plant activity. A graphic display can show system or process data, and provide an operator with ways to write values to external devices such as programmable controllers.

### To add a Display sub-object

1. In the **System View** pane, expand **HMI**, expand **Displays** and then click the FactoryTalk View object to which the sub-object will be added.
2. In the object parameters pane, select the **Displays** tab.
3. Right-click in the white space below the objects shown in the tab. Click **Add New**.

Repeat this step to add all of the Displays that require configuration.

4. Click the appropriate column and modify the parameters as needed.
5. Click **Apply Changes** to save the parameter configuration.

### Group sub-objects

Group sub-objects into categories based on the data columns.

- Double-clicking on a column heading updates the display in the parameters tab to group items with the same value together.
- Right-click anywhere in the parameters tab and select **Reset Grouping** to return to the default display.

## See also

[Add an HMI object](#) on [page 33](#)

[HMI Display parameters](#) on [page 34](#)

[Generate an HMI object](#) on [page 35](#)

## HMI Display parameters

The **Display** tab displays a columnar grid of the parameters defined for the displays of the FactoryTalk View object.

This table describes the parameters in the **Displays** tab.

Parameter	Description
Name	The name of the display. Double-click to configure a unique name for the display.
DisplayTitle	Specifies the content of the display title bar.
DisplayLeft	Sets the left edge of a display to the number of units specified from the left edge of the screen.
DisplayTop	Sets the top edge of a display to the number of units specified from the top edge of the screen.
DisplayWidth	Sets the width of a display.
DisplayHeight	Sets the height of a display.
DisplayBackColor	Sets the background color of the display.
DisplaySecurity	Identifies the security class of the screen.
UpdateRate	Defines the amount of time to update a display in response to an event. Default setting is 1.0 seconds. For best performance set to .1 seconds.
LeftIndexMax	Identifies the maximum number of entries in the left index.
VBAProjectObject	Adds a VBA item to the FactoryTalk View SE display. Click the ellipsis (...) button to select a VBA item.
SubObject Description	Provides a unique description of the display sub-object.

### See also

[Add a Display sub-object](#) on [page 34](#)

## Generate an HMI object

Once a HMI display object has been configured it can then be used to generate an import file for use with FactoryTalk View SE or FactoryTalk View ME. This file is an Extensible Markup Language (.xml) file.

### To generate an HMI object file

1. In the **System View** pane, expand the **HMI** folder and the display category folder so that the objects are visible.
2. Right-click the object to export.
  - For FactoryTalk View objects, select **Generate Displays** and then click **All Displays** to create an export file that supports all display types.
  - For Alarms objects, select **Generate Alarms** and then click a specific controller from the list if this object will only be used with that type of controller or click **All Controllers** if this object will be used with all project controllers.

The **Save As** dialog box opens.

3. Use the tree control to browse to the location in which to save the HMI object file or use the default location:

C:\Users\*<user name>*\Documents\Studio 5000\Projects  
The variable *<user name>* is replaced by the logged in user account name.

4. In **File name** type a name for the file or use the default file name which is created according to the following pattern:

*<Controller Name>\_<Object Name>.csv*

The controller portion of the default file name and the underscore are omitted when the **All Controllers** option is selected. For Alarms objects the variable <Controller Name> is replaced by the name of the controller selected if a specific controller was selected. FactoryTalk View objects are not controller specific.

5. Click **Save**. The HMI object generation is started.
6. Once the process is completed, the **Generation Complete** status dialog box appears, click **OK** to close the dialog box.

### See also

[Add a new HMI object](#) on [page 33](#)

## Add an Alarm Group sub-object

FactoryTalk Alarm and Event objects have sub-objects that represent the Alarm Groups configured in the FTAE system. In FactoryTalk Alarm and Events, alarms can be assigned to groups. A group represents a set of alarms with a common association. An alarm group can contain other groups organised in a hierarchy (FTAE only supports to 5 levels deep).

### To add an Alarm Group sub-object

1. In the **System View** pane, expand **HMI**, expand **Alarms** and then click the FactoryTalk FTAE object to which the sub-object will be added.
2. In the object parameters pane, select the **Alarm Group** tab.
3. Right-click in the white space below the objects shown in the tab. Click **Add New**.

Repeat this step to add all of the Alarm Group that require configuration.

4. An alarm group hierarchy is achieved by associating one alarm group to another using the ParentAlarmGroupID parameter.
5. Click the appropriate column and modify the parameters as needed.
6. Click **Apply Changes** to save the parameter configuration.

### Group sub-objects

Group sub-objects into categories based on the data columns.

- Double-clicking on a column heading updates the display in the parameters tab to group items with the same value together.
- Right-click anywhere in the parameters tab and select **Reset Grouping** to return to the default display.

## See also

[Add an HMI object](#) on [page 33](#)

[FTA E Alarm Group parameters](#) on [page 37](#)

[Generate an HMI object](#) on [page 35](#)

## FTA E Alarm Group parameters

The **Alarm Group** tab displays a columnar grid of the parameters defined for the alarm groups of the FactoryTalk Alarms and Events object.

This table describes the parameters in the **Alarm Group** tab.

Parameter	Description
Name	The name of the sub-object representing the Alarm Group (this is not the name that will be used in the FTA E system). Through name formatting a unique sub-object name is built up from the AlarmGroup and ParentAlarmGroupID parameters.
AlarmGroupID	Each AlarmGroupID must be unique (0 is reserved for the FTAETagServer ID).
ParentAlarmGroupID	The ParentAlarmGroupID determines the alarm group hierarchy (Default: 0 = FTAETagServer ID).
AlarmGroup	The alarm group name that will be created in the FTA E system and assigned to FTA E alarm tag objects that reference it.
SubObject Description	Provides a unique description of the alarm group sub-object.

## See also

[Add an Alarm Group sub-object](#) on [page 36](#)

## Find an object

Use the **Find** dialog box to search for an object or a library in the **Class View** pane, contents in the **Controller Preview** tree, or a library in the **Registered Libraries** tree.

### To find an object

1. On the toolbar, select **EDIT > Find**.
2. In **Find what**, enter the name of an object.
3. In **Look in**, select **Class View**, **Controller Preview**, or **Registered Libraries**.
4. (optional) In **Find Options**, select the check boxes of **Match case** and **Match whole word**.
5. Select **Find Next** or **Find Previous**.

## See also

[Project Development](#) on [page 27](#)

[Update a project library](#) on [page 37](#)

## Update a project library

Any of the libraries used in a project can be updated to incorporate additions or modifications to library components.

If **Newer Library Version Available** is enabled in Application Code Manager settings, indications appear next to objects that can be updated.

## To update a project library

1. In the **System View** pane, expand **Used Libraries** to see the library branches. Library versions are shown in parenthesis.
2. Right-click the library branch to be updated and then click **Update**. The **Update Used Libraries** dialog box is displayed.
3. A tab is shown for each library in the selected library branch. Libraries with available updates will have different revisions.  
If multiple libraries are included in the library branch, click the tab for the library to update. Each library tab has three sub-tabs.
4. In **Update Library Revision**, click the drop-down arrow to select the new revision number for the library.
5. Click the **Library Changes** sub-tab to review the parameter changes for the new version.
6. Click the **Objects to Update** sub-tab to define which objects in the library are to be updated.
  - Under **Update**, select the checkbox to update the project objects that reference this project library.
  - Clear the checkbox for each object that should not be updated.
7. Click the **Relink References to Removed Items** sub-tab and select the parameters and sub-parameters to relink to the objects that are affected by this update.
8. Click **Finish**.

### See also

[Upgrade the Application Code Manager application](#) on [page 16](#)

[Library objects](#) on [page 11](#)

[Indications](#) on [page 39](#)

## Extract attached files

Files attached to a project, Historian object, HMI display, FactoryTalk Alarms and Events server, controller, library, or library object can be extracted to a folder of your choice. When extracting the attachments from the various scopes in a project, all the objects within the scope have their attachment include conditions evaluated and only those attachments that evaluate to true will be extracted to their relative path. If the relative path is parameterized, the path will be resolved. If the resolved path is invalid, the attachment will be extracted to the root extraction folder. Attachments that are the same, but have a different relative path will be extracted into each of the relative paths. Attachments that already exist at the relative path will be overwritten.

## To extract attached files in System View

1. In **System View**, right-click on one of the following, then select **Extract Attached Files**:
  - The ACM project
  - A Historian object
  - An HMI display
  - A FactoryTalk Alarms and Events server
  - A controller or controllers
  - A library
  - A library object
2. In **Browse for Folder**, select a folder where the files should be extracted, then select **OK**.

## To extract attached files in Registered Libraries

1. In **Register Libraries**, right-click an object or its descendant objects with an attachment, and then select **Extract Attached Files**.
2. In **Browse for Folder**, select a folder where the file should be extracted or create a new folder, and then click **OK**.

## To extract attached files in Class View

1. In **Class View**, right-click an object or its descendant objects with an attachment, and then select **Extract Attached Files**.
2. In **Browse for Folder**, select a folder where the file should be extracted or create a new folder, and then click **OK**.



When you extract an attachment in **Class View**, if the extraction path of an attachment in the library is configured as `{ObjectName}\Folder1\Folder2` by an expression in Library Object Manager and the Object instance is "XV100", then upon extraction of the attachment the extracted relative path will be `XV100\Folder1\Folder2`.


## See also

[Project Development](#) on [page 27](#)

## Indications

If **Modified Objects** or **Newer Library Version Available** is enabled in Application Code Manager settings, indications appear next to objects that have been modified or libraries that can be updated.

Indication	Description
	This object has been modified since controller was last generated. The flag will be reset once the controller is generated.
	<b>Class View:</b> This library has a newer version available in <b>Registered Libraries</b> . <b>Registered Libraries:</b> This library has a newer version available in the configured <b>Library Repositories</b> .

Indication	Description
	This node or folder contains one or more libraries that has a newer version available in <b>Registered Libraries</b> .

### See also

[Update a project library](#) on [page 37](#)

[ACM Default Settings](#) on [page 79](#)



## Registered Libraries

### Registered Libraries

**Registered Libraries** displays all libraries (classes) in the connected ACM database in a tree view. Add a library to an ACM project to use the objects in it with your project.

A library object defines parameters, subclasses, user interface contents, and portions of controller code (example: Logix) and HMI code (example: FactoryTalk View SE/ME). It can also include custom properties that were applied using the Library Designer.

The library nodes on the **Registered Libraries** tree will show an indication if any of the selected library repositories contain a library with the same solution and catalog number but a higher revision number.

Right-click any branch in the **Registered Libraries** tree view to view the commands available.

This table describes each command. Commands appear at the applicable level of the tree.

Command	Description
Registered Library	
Register	Used when a library object is provided as a HSL4 file and needs to be incorporated into ACM. Registers one or more library files (HSL4).
Get updates	Updates a library object to a newer version. Notifications appear when <b>Newer Library Version Available</b> in the <b>ACM Default Settings</b> is selected.
View > Pending Libraries	Displays a report showing <b>Database Pending Libraries</b> report. Available only at the <b>Registered Libraries</b> branch.
View > Library Usage	Displays a report showing the <b>Registered Library</b> usage for the libraries in the connected ACM database.
Export	Exports the selected library to an HSL4 (xml) formatted file to a folder, the destination folder is automatically opened.
Extract Attached Files	Extracts the attached files in the selected library to a folder, the destination folder is automatically opened.
Reconstitute ACD	Opens the <b>ACD Re-constitution Wizard</b> .
Delete	Deletes the selected library in the connected ACM database. Libraries that are used in projects cannot be deleted. If the library selected for deletion is in use, a <b>Global Library</b> usage report is displayed instead.
Delete all unused	Deletes all unused libraries for the selected level in the connected ACM database.
Refresh	Refreshes the tree view.
ACM Library	
View > Pending Libraries per Solution	Displays a report showing the registered library usage for the pending libraries in the selected solution. Available only at a solution branch.
View > Library Usage per Solution	Displays a report showing the registered library usage for the libraries in the selected solution. Available only at a solution branch node.
View > Pending Libraries per Library Type	Displays a report showing the registered library usage for the selected pending library type. Available only at a library type branch node.
View > Project Library Usage per Library Type	Displays a report showing the registered library usage for the selected library type. Available only at a library type branch node.
View > Pending Libraries per Category	Displays a report showing the registered library usage for the selected pending library category. Available only at a library category branch node.

Command	Description
View > Library Usage per Category	Displays a report showing the registered library usage for the libraries in the selected library category. Available only at a library category branch node.
View > Library Usage per Library	Displays a report showing the registered library usage for the selected library. Available only at a library branch node.

**See also**

[Register an ACM Library](#) on [page 42](#)

[Reconstitute the ACD](#) on [page 42](#)

[View registered library usage](#) on [page 43](#)

[Share libraries, templates, and schedules](#) on [page 44](#)

**Register an ACM library object**

Register an ACM library object to use it in a project.

**Prerequisites**

- Verify that the ACM application is connected to the correct ACM database.



Tip: The connected database is shown in parenthesis in the **Application Code Manager** title bar.

**To register an ACM library object**

1. In the **Register Libraries** tree, right-click the library object to register.
2. Click **Register**.
3. The **Libraries Registration** window opens.
4. Review the information provided in the **Library Registration** window. Perform any indicated actions.
5. Click **Finish** to complete the registration process.

**See also**

[View registered library usage](#) on [page 43](#)

[Share libraries, templates, and schedules](#) on [page 44](#)

[Connect to an ACM database](#) on [page 21](#)


[Reconstitute the ACD](#) on [page 42](#)

**Reconstitute an ACD file**

Reconstitute an ACD file when the original Logix Designer project used to create the object is not available for editing. After reconstituting the ACD file, it can be edited using Logix Designer.

Hardware module libraries and ACM v1.x libraries do not support this feature.

## To reconstitute an ACD file

1. In the **Registered Libraries** tree view, right-click the library file(s).  
To select multiple files hold down the <Ctrl> key and select the library files.
2. Click **Reconstitute ACD**.  
The **ACD Reconstitution Wizard** opens.
3. The **Object Configuration** page displays the details of the objects to be included in the ACD file.  
Click **Next**.
4. The **Resolve Dependencies** page displays any dependencies between the library objects that must be resolved.
  - Click **Add Dependency Library**  and add the library that contains required add-on instructions (AOIs) or user-defined data types (UDTs).
  - Click **Next**.
5. The **Unresolved Names** page displays any objects that have unresolved name value. Click in the **Value** column and type valid values for any names highlighted on the page then click **Next**.
6. The **L5X Generation Successful** page displays a preview of the Logix content contained in the ACD file displays.
7. Click **Next** to display the **Save As** screen. In **File name** type a file name for the ACD file.
8. Click **Save** to save the file. The **ACD successfully saved** message displays with a link to open the folder where the file was saved.
9. Click **Finish** to close the wizard.

## See also

[Add a controller from an ACD or L5X file on page 51](#)

[Update using an ACD or L5X file on page 51](#)

[Detach from ACD/L5X files on page 53](#)

## View registered library usage

There are several different reports that provide information about registered library usage. Determine the scope of the report by navigating to the level of interest and then generate the report.

## To view registered library usage

1. Select the scope of the report.
  - To generate a report that includes all of the registered libraries in the connected ACM database:

Right-click **Registered Libraries**, then point to **View**, then click **Library Usage**.

- To generate a report that includes all of the registered libraries used in a solution:

In the **Registered Libraries** tree, right-click the solution folder, then point to **View**, then click **Library Usage per Solution**.

- To generate a report that includes all of the registered libraries used in a library type:

In the **Registered Libraries** tree, expand the solution folder, right-click the library type folder, then point to **View**, then click **Library Usage per Library Type**.

- To generate a report that includes all of the registered libraries used in a library category

In the **Registered Libraries** tree, expand the solution folder, expand the library type folder, right-click the library category folder, then point to **View**, then click **Library Usage per Library Category**.

- To generate a report that includes all of the registered libraries used for a library object

In the **Registered Libraries** tree, expand the solution folder, expand the library type folder, expand the library category folder, right click the library object then point to **View**, then click **Library Usage per Library**.

2. The selected report appears in a new window.

## See also

[Reports](#) on [page 109](#)

[Generate a report](#) on [page 111](#)

## Share libraries, templates, and schedules

All libraries in the **Registered Libraries** tree are available to all ACM users that are connected to the ACM database.

Schedule templates can be created that are limited to just a specific user account. If a template that is limited to a user account needs to be shared with other users, use Import/Export Manager to either copy or move the template from the users folder to the central ACM database.

ACM Program folder templates can be shared by placing the template files in a shared network folder.

Project data that can be reused in multiple ACM projects can be exported to a schedule and shared by placing the Schedule file in a shared network folder.

To avoid the possibility of ACM users overwriting each other's work in a central ACM database, ACM users should work in different projects or

branches of the same project tree view. Project work can be divided by function (example: Controller Hardware, Controller Software, HMI, Historian) or by area (example: Receiving, Mixer, Shipyard).

Use the partial export option to avoid exporting the same data to more than one schedule. If the same data is imported from more than one schedule, the last schedule imported will determine the data.

To help ensure that work is not lost, export a project schedule periodically for backup.

### See also

[Import Export Manager](#) on [page 61](#)

[Local library and template file location](#) on [page 45](#)

## Local library and template file location

By default library object files and template files are stored in this location:

**C:\Users\Public\Public Documents\Rockwell Automation\Studio 5000**

There are separate **Libraries** and **Templates** folders with an **Application Code Manager** subfolder in each. The library object repositories, **(RA-LIB) ACM** and **(RA-LIB) Process**, and the template repository, **(RA-TPL) ACM**, are located in their respective subfolders.

The **(RA-TPL) ACM** folder is set as the default documentation path for the Application Code Manager application. To change the documentation path open **ACM Default Settings** and specify the path to use.



Tip: If working in a collaborative design environment with a shared ACM database, template files can be copied to a shared folder so that all users can access them.

### See also

[Initial configuration of Application Code Manager](#) on [page 15](#)

[Registered Libraries](#) on [page 41](#)

[ACM Default Settings](#) on [page 79](#)

## Library Repositories

**Library Repositories** displays the configured library repositories. A library repository refers to another ACM database that can be monitored for library updates. Libraries from these source databases can be easily replicated over to the currently active database for use within a project. It is unavailable to create an object directly from a library in a library repository. The library must first be replicated to the current active ACM database before it can be used to create an object. Add or remove library repositories in this pane.

Click **View** and select **Library Repositories** to open this pane.

Right-click any branch in the **Library Repositories** tree view to view the commands available.

This table describes each command. Commands appear at the applicable level of the tree.

Command	Description
<b>Library Repositories</b>	
Replicate to Active Database	Loads the <b>Available Libraries</b> form to select libraries to copy or register to the current active ACM database.
Export	Exports the <b>Library Repositories</b> to an HSL4 (xml) formatted file.
Extract Attached Files	Extracts the attached files to a folder.
Disable All	Disables all update status checking of the databases.
Add	Adds a library repository.
Remove All	Removes all library repositories.
Refresh	Refreshes all library repositories.
<b>Library repository nodes</b>	
Replicate to Active Database	Loads the <b>Available Libraries</b> form to select libraries to copy or register to the current ACM database.
Export	Exports this library repository to an HSL4 (xml) formatted file.
Extract Attached Files	Extracts the attached files of this library repository to a folder.
Disable	Disables update status checking of this database.
Remove	Removes this library repository.
Connect	Sets this database as the current active database.
Refresh	Refreshes all contents in this library repository.
<b>Solution, library type, and category branch</b>	
Replicate to Active Database	Loads the <b>Available Libraries</b> form to select libraries to copy or register to the current active ACM database.
Export	Exports this branch to an HSL4 (xml) formatted file.
Extract Attached Files	Extracts the attached files of this branch to a folder.
Refresh	Refreshes all contents in this branch.
<b>Catalog number node</b>	
Replicate to Active Database	Loads the <b>Available Libraries</b> form to select libraries to copy or register to the current active ACM database.
Export	Exports this library to an HSL4 (xml) formatted file.
Extract Attached Files	Extracts the attached files in this library to a folder.
Refresh	Refreshes this node.

### See also

[Register an ACM library object](#) on [page 42](#)

[Reconstitute an ACD file](#) on [page 42](#)

## Configure controllers

### Configure controllers

Right-click any controller branch in the **Controller Preview** or **Class View** pane to view the controller configuration commands.

This table describes each **Controller** command. Commands appear at the applicable level of the tree.

Command	Description
Controller	
• Generate Controller	Generates a copy of the selected controller from ACM. Displays the <b>Logix Code Generation</b> dialog box. Generates code (example: L5X or ACD) for the selected controller.
• Add New	Adds a new controller object to the project using the <b>Object Configuration Wizard</b> .
• Add New from ACD/L5X	Adds a new controller object to the project from an ACD or L5X file. Additional objects may also be added depending on what is configured and available in the ACD file.
• Update from ACD/L5X	Updates the ACM project based on an existing ACD file.
• Detach from ACD/L5X	Removes the link between the ACM project and the attached ACD or L5X file.
• Merge Controller	Merges content between the original ACM project, updated ACM project, and existing ACD or L5X file. Utilizes the Logix Designer Compare Tool.
• Add New Program	Adds a new program to the current controller object using the <b>Object Configuration Wizard</b> .
• Add New Task	Adds a new task to the current controller object using the <b>Object Configuration Wizard</b> .
• Paste	Pastes the contents of the clipboard to the selected controller.
• Paste Special	Pastes the contents of the clipboard to the selected controller with options.
• Generate Partial Program	Displays the <b>Logix Code Generation</b> dialog box. Generates a partial program for the selected task.
• Generate Partial Routine	Displays the <b>Logix Code Generation</b> dialog box. Generates a partial routine for the selected task.
• Move Up	Moves the highlighted object up the tree.
• Move Down	Moves the highlighted object down the tree.
• Set Main Routine	This command is available when multiple main routines have been added from different library objects. There can only be one main routine. If a main routine is already selected this command is dimmed.
• Set Routine Instance Order	Displays the <b>Instance Execution Order</b> dialog box which specifies the order in which any contributing objects insert their code into the routines.
Object	
• Import	Starts Import Export Manager.
• Export	Starts Import Export Manager.
• Delete	Deletes the selected object.
• Copy	Copies the selected object to the clipboard.
• Rename	Renames the selected object.
• Update	Updates the selected libraries to the most recent version.
View	
• Project Library Usage per Library	Displays a report showing the registered library usage for the selected library. Available only at a library branch.
• Network Layout	Generates <b>Network Layout</b> report.
• Module I/O Schedule for Rack Module	Displays the <b>I/O Schedule</b> showing rack assignments.
• Chassis Layout	Displays or prints a report with the I/O Modules in the Controller (local) chassis. Pastes the contents of the clipboard to the selected Controller with options.
• Add New Instance	Adds a new task in the <b>Class View</b> pane.
• Navigate to Library	Navigates to the library in the <b>Registered Libraries</b> tree.
• Object References	Displays the Object References report.

Command	Description
<ul style="list-style-type: none"> <li>Exclude AOI (Add On Instructions)</li> </ul>	Excludes the AOI (included by default). Only available in the <b>Add-On Instructions</b> branch. When two AOIs with same major and minor revision, but different extended text are added in the <b>Controller Preview</b> tree, AOI conflict shows. You can use <b>Exclude AOI</b> to select which one to use.
<ul style="list-style-type: none"> <li>Include AOI (Add On Instructions)</li> </ul>	Includes the AOI. Only available in the <b>Add-On Instructions</b> branch.
<ul style="list-style-type: none"> <li>Refresh</li> </ul>	Refreshes the tree view.

### See also

[Generate a controller file](#) on [page 48](#)

[Merge controllers](#) on [page 53](#)

[Add a hardware module](#) on [page 54](#)

[Add a new software object to a controller](#) on [page 56](#)

## Generate a controller file

After configuring a controller in Application Code Manager, generate the Logix Designer files for the controller.

### To generate a controller file

1. In the **Controller Preview** or **Class View** pane, right-click the Controller folder or the controller object and then click **Generate Controllers**.
2. In the **Logix Code Generation** dialog box, select the **Generate** and **ACM Project Data** check boxes.
  - Select the **Overwrite Existing** check box to replace an earlier controller file with the same name.
  - Select **Create ACD** to create a Logix Designer Project file (.ACD) in addition to the Logix Designer XML file (.L5X).

---

**IMPORTANT** Generating controller files with **ACM Project Data** selected produces the most complete record of the controller data. Always select **ACM Project Data** when generating a controller file as part of a disaster recovery backup. If there is not a matching Studio 5000 Logix designer version installed and the **SoftwareRevision** is selected, the **Create ACD** checkbox is disabled.

---

3. In **Save Path**, type the path where the controller files should be saved or use the default path:  
 C:\Users\\AppData\Local\Rockwell Automation\Application Code Manager\Output  
 The variable <user name> is replaced by the logged in user account name.
4. Click **Generate**.  
 The controller files are generated. A status message displays when completed.
5. (optional) Click **Open Folder** to open the file explorer to the controller file location.



## See also

[Navigate the Application Code Manager user interface](#) on [page 17](#)

[Generate a partial program](#) on [page 57](#)

[Generate a partial routine](#) on [page 58](#)

## Add a new controller

Use the **Object Configuration Wizard** to add a controller from a predefined controller library that resides in the registered libraries, configure its parameters, and include it in the ACM design output.



Note: The ability to add multiple controllers to a project or open an existing project containing multiple controllers is only available if the product has a Standard activation license and is not available in Lite mode.

## To add a controller

1. In the **Controller Preview** or **Class View** pane, right-click the **Controllers** folder and then click **Add New**.

The **Object Configuration Wizard** displays. Controller objects display in category groups. A triangular marker next to the column name denotes the current category grouping displayed. Change category groups by clicking a column heading.

2. Click the + symbol to expand a category and display the controllers in that group.

Controllers registered in the connected ACM database are listed.

3. Click the desired controller and click **Next**.
4. In **Name**, enter a name for the controller.
5. (optional) In **Description**, modify the descriptive text as needed to help identify the controller.
6. In the **Parameters** tab, adjust **Controller**, **HMI**, **Historian**, and **Motion** parameters as needed.
7. Click **Finish**.

The controller adds to the **Controller Preview** and **Class View** trees.

## See also

[Connect to an ACM database](#) on [page 21](#)

[Add a controller from an ACD or L5X file](#) on [page 51](#)

[Update using an ACD or L5X file](#) on [page 51](#)

## Move or copy a controller to another project

For projects in the same database, **Copy To** and **Move To** allow you to copy or move a controller from one project to another project from the **Class View** pane.



Note: This feature is only available if the product has a Standard activation license and is not available in Lite mode.

## To move or copy a controller to another project

1. In the **Class View** pane, expand **Controllers**.
2. Right-click the controller that will be copied or moved to another project.
  - Select **Copy To**.  
When selecting **Copy To**, the controller will be copied and pasted to the destination project.
  - Select **Move To**.  
When selecting **Move To**, the controller will be cut and pasted to the destination project. When the controller has unresolvable references, they will be cleared.
3. In **Select Destination Project**, select an existing project or create a new project as the destination project.  
When there are name conflict between the moved controller and the controller in the existing project, rename the destination controller in **Rename Destination Controller**.
4. Click **OK**.
  - When prompted, select **OK**.
  - Once the **Log File Viewer** opens, track the errors, warning, informational, and debug events.

## See also

[Configure controllers](#) on [page 47](#)

[Select Destination Project dialog box](#) on [page 50](#)

[Add a new controller](#) on [page 49](#)

## Select Destination Project dialog box

The following table shows the options in the **Select Destination Project** dialog box.

Options	Descriptions
<b>Name</b>	Displays the name of the destination project.
<b>Description</b>	Displays the description of the destination project.
<b>New</b>	Opens the <b>Object Configuration Wizard</b> dialog box to create a new project.
<b>OK</b>	Starts the copying or moving process.
<b>Cancel</b>	Returns the home page.

**See also**

[Move or copy a controller to another project](#) on [page 49](#)

[Configure controllers](#) on [page 47](#)

**Add a controller from an ACD or L5X file**

Add controllers to the Application Code Manager project from an existing Logix Designer XML file (.L5X) or Logix Designer Project file (.ACD).

**To add a controller from an ACD or L5X file**

1. In the **Controller Preview** or **Class View** pane, right-click the **Controller** folder and then click **Add New from ACD/L5X**.
2. In the **Select an ACD/L5X File** dialog box, select the file to add then click **Open**.
3. In the **Import From ACD Wizard - Instance Import Actions** page review the items listed. Items with **Add** in the **Action** column are imported.
4. If any objects should not be imported, click **Add** and then select **Exclude**.
5. Select **Next**. The **Import** dialog displays. The information grid shows any errors, warning, info, or debug messages generated during the import. Review any warnings or errors listed.
  - Click **View Changes** to see a comparison of the original and updated information in an Excel spreadsheet.
  - Information buttons show how many of each type of message are present. Filter the displayed messages by clicking the button for the message type to toggle the display of those messages.
6. Select **Next**. The **Import From ACD Wizard - Content Imported** page displays.
7. Select **Finish**.

The new object will appear in the **Controller Preview** pane and **Class View** pane.

**See also**

[Update using an ACD or L5X file](#) on [page 51](#)

[Detach from ACD/L5X files](#) on [page 53](#)

[Merge controllers](#) on [page 53](#)

**Update using an ACD or L5X file**

Update controller configurations in an Application Code Manager project using information from a Logix Designer XML file (.L5X) or a Logix Designer Project file (.ACD).

## To update a controller using an ACD/L5X file

1. In the **Controller Preview** or **Class View** pane right-click the controller to update and select **Update from ACD/L5X**.
2. In **Select an ACD/L5X File** choose the ACD or L5X file to use to update the controller

The **Import from ACD Wizard - Instance Import Actions** page displays.

- Objects that are present in the file but not in the target controller will be selected to be added by default. If any of these objects should not be added select **Exclude** from the **Action** column.
- Objects that already exist in the target controller will default to being excluded from the update. If any of these objects should be updated using the information in the file select **Update** from the **Action** column.



Note: When excluding an object, information in the ACM project is preserved. When an update is performed, information from the ACD or L5X overwrites the ACM project information.

3. Select **Next**. The import operation proceeds. After the operation is finished the **Import Complete** page displays. The information grid shows any error, warning, info, or debug messages generated during the import. Review any **Warnings** or **Errors** listed.
  - Click **View Changes** to see a comparison of the original and updated information in an Excel spreadsheet.
  - Information buttons show how many of each type of message are present. Filter the displayed messages by clicking the button for the message type to toggle the display of those messages.
4. Select **Finish**. The **Import From ACD Wizard - Content Imported** dialog appears. If one or more objects are set to update, then **Update the Original Snapshot** will be checked by default. Clear the check box to prevent the snapshot information from updating the current configuration and then click **Finish** to close the wizard.



Note: Snapshot information is used for the original ACM project data when performing the merge.

If all items are excluded, by default **Update the Original Snapshot** is not checked.

5. If **Update the Original Snapshot** was selected click **Next**. The **Import From ACD Wizard - Original Snapshot Updated** confirmation page displays, click **Finish** to close the wizard.

## See also

[Detach from ACD/L5X Files](#) on [page 53](#)

[Reconstitute an ACD file](#) on [page 42](#)

[Generate a controller file](#) on [page 48](#)

## Detach from ACD/L5X files

If a controller is updated using an ACD or L5X file then the controller will maintain an attachment to that file. The next time ACM is started, the ACM application will attempt to establish a connection to the file. Depending on file systems, file size, and file location establishing a connection to an ACD or L5X file may take more time than expected. If needed, the ACM application can detach the file from the controller.

### To detach from an ACD/L5X file

- In the **Controller Preview** or **Class View** pane, right-click the controller to which the file is attached to and then select **Detach from ACD/L5X**.

### See also

[Merge controllers](#) on [page 53](#)

[Update using an ACD or L5X file](#) on [page 51](#)

[Add a controller from an ACD or L5X file](#) on [page 51](#)

## Contribute instances

Use **Contributing instances** to view and edit the properties of an object instance.

### To contribute an instance

1. In the **Controller Preview** pane, expand **Controllers**.
2. Right-click a node shared by multiple instances, and then select **Contributing instances**.

A list of contributing instances is shown.

3. Select an instance.

Its parameters property panel opens.

### See also

[Configure controllers](#) on [page 47](#)

[Add a new instance](#) on [page 56](#)

## Merge controllers

Use **Merge Controller** to add or update objects from an ACM project into the attached ACD file.

### To merge controllers

1. In either the **ClassView** or **Controller Preview** pane right-click the object to merge.

2. Select **Merge Controller**. The **Logix Designer Compare Tool** will be launched.

Refer to the Logix Designer Compare Tool help files for further information.



Note: After the merge file has been created, confirm all information is correct prior to deploying.

### See also

[Generate a controller file](#) on [page 48](#)

[Add a hardware module](#) on [page 54](#)

[Add a new software object to a controller](#) on [page 56](#)

## Add a hardware module

Add hardware modules to your controller configuration in the **I/O Configuration** branch in the **Controller Preview** pane.

### To add a hardware module

1. In the **Controller Preview** pane, expand the **Controllers** node, expand the controller object, expand the **I/O Configuration** folder and then right-click **Backplane** and then click **Add New**.

The **Object Configuration Wizard** displays.

2. Click the + symbol to expand the library category and display the hardware module libraries registered in the connected ACM database.
3. Select the library that will be copied to create the new hardware module and click **Next**.
4. (optional) In **Description**, type a unique description of the hardware module or accept the default description.
5. Click **Finish**.

The hardware module object appears in the **Controller Preview** pane under **Backplane**.

### See also

[Use Copy and Paste Special](#) on [page 55](#)

[Add a new instance](#) on [page 56](#)

[Add a new software object to a controller](#) on [page 56](#)

## Delete a hardware module

If a module is no longer relevant to the application it can be deleted from the project.

## To delete a module

1. In the **Class View** pane, right-click the module and then select **Delete**.
2. In the **Delete** dialog box, click **Yes** to confirm that the module should be deleted.
3. If the object is referenced by other objects the **Delete Action** dialog box displays.
  - a. Click **Clear** to remove all references to the module being deleted.
  - b. Click **OK** to delete the module.

The module is removed from the **Class View** and **Controller Preview** panes.

## See also

[Add a hardware module](#) on [page 54](#)

## Use Copy and Paste Special

**Copy and Paste Special** provides the ability to decide whether to include children, sub-objects, and reference values when pasting a copied object. If an object is simply pasted all of the associated items are copied and pasted along with the object. The **Copy and Paste Special** command is only available from the **Class View** pane.

## To use Copy and Paste Special

1. Select the object that will be copied to create the new object. Right-click and click **Copy**.
  - Right-click the destination of the copied object and click **Paste Special**.
  - Select the branch where the object will reside. Right-click and click **Paste Special**.
2. The **PasteSpecial** dialog box displays.
  - Select the **Include Children** check box to include Communication Module Children.
  - Select the **Include Sub-Objects** check box to include I/O Module Sub-Objects (example: Channels).
  - Select the **Include IO Reference Values** check box to maintain references to I/O objects.
  - Select the **Include Non-IO Reference Values** to maintain references to other types of objects
3. Click **Paste**.
4. The **Rename** dialog displays. The object names that must be changed to prevent a naming conflict are shown in red.
  - Enter a new name when a naming conflict exists. When there are no conflicts, the name is black in color.

- Click **OK** to paste the object.

### See also

[Delete a hardware module](#) on [page 54](#)

[Configure controllers](#) on [page 47](#)

## Add a new instance

Use **Add a new instance** to add a new library object instance to an existing library class in the **Class View**.

### To add a new instance

1. In the **Class View** pane, right-click the library class from which the new instance will be created and then click **Add New Instance**.

The **Object Configuration Wizard** is displayed.

2. (optional) In **Name**, type a new name for this instance.
3. (optional) In **Description**, type identifying information about this instance.
4. Configure the associated Task and Program.
5. Click **Finish** to add the new instance.

### See also

[Configure controllers](#) on [page 47](#)

[Add a new software object to a controller](#) on [page 56](#)

## Add a new software object to a controller

Add re-usable code for software objects (tasks, routines and programs) to the controller from the Application Code Manager library.

### To add a new software object to a controller

1. Use one of the following methods:
  - In **Controller Preview**, expand **Controllers**, expand the controller object, right-click a task or program object, then click **Add New**.
  - In **Class View**, expand **Controllers**, right-click the controller object, then click **Add New**.
2. In the **Object Configuration Wizard**, expand the library categories to display the software libraries registered in the ACM database.
3. Select the library object to add to the selected task or program, then select **Next**.
4. (optional) In **Description**, type a unique description of the software or accept the default description.



5. (optional) Edit software object parameters and fields in the tabs on the bottom half of the **Object Configuration Wizard**.
6. (optional) If the software object depends on other linked library objects, select the **Linked Libraries** tab, then select **Auto Create Linked Objects**. Select linked library creation options, then click **OK**.
7. Select **Finish**.

The software object appears in the **Controller Preview** and **Class View** panes.

### See also

[Generate a partial program](#) on [page 57](#)

[Generate a partial routine](#) on [page 58](#)

[Object Configuration Wizard](#) on [page 58](#)

[Auto Create Linked Objects](#) on [page 59](#)

## Generate a partial program

**Generate Partial Program** exports a Logix Designer XML file (L5X) that can be imported into the Logix Designer as a program.

### To generate a partial program

1. In the **Controller Preview** pane, expand **Controllers**, expand the controller object, right-click the program object to export and then click **Generate Partial Program**.

The **Logix Code Generation** dialog box displays.

2. (optional) To rename the file click in the **Save As** field. The text box becomes editable and the current file name is highlighted. Type the new file name.
3. Verify that the **Generate** and **ACM Project Data** check boxes are selected.

- Select the **Overwrite Existing** check box if you want to replace an earlier file with the same name.

The **Create ACD** option is disabled. Partial program and routine items can only be generated using .L5X files.

4. In **Save Path**, type the path where the exported files should be saved or use the default path:

```
C:\Users\\AppData\Local\Rockwell Automation\Application Code Manager\Output
```

The variable <user name> is replaced by the logged in user account name.

5. Click **Generate**.

The files are generated. A status message displays when completed.

- (optional) Click **Open Folder** to open the file explorer to the file location.

### See also

[Generate a partial routine](#) on [page 58](#)

## Generate a partial routine

**Generate Partial Routine** allows you to create an export file (L5X) that can be imported into the Logix Designer as a routine.

### To generate a partial routine

- In the **Controller Preview** pane, expand **Controllers**, expand the controller object, right-click the routine object to export and then click **Generate Partial Routine**.

The **Logix Code Generation** dialog box displays.

- (optional) To rename the file double-click in the **Save As** field. The text box becomes editable and the current file name is highlighted. Type the new file name.
- Verify that the **Generate** and **ACM Project Data** check boxes are selected.
  - Select the **Overwrite Existing** check box if you want to replace an earlier file with the same name.

The **Create ACD** option is disabled. Partial program and routine items can only be generated using .L5X files.

- In **Save Path**, type the path where the exported files should be saved or use the default path:

```
C:\Users\\AppData\Local\Rockwell
Automation\Application Code Manager\Output
```

The variable <user name> is replaced by the logged in user account name.

- Click **Generate**.

The files are generated. A status message displays when completed.

- (optional) Click **Open Folder** to open the file explorer to the file location.

### See also

[Add a new instance](#) on [page 56](#)

## Object Configuration Wizard

How do I open the Object Configuration Wizard?

Use one of these methods:

- In **Controller Preview**, right-click a folder, task, program, or I/O configuration node in the project, then select **Add New**, **Add New Task**, or **Add New Program**.
- In **Class View**, right-click the project or library, then select **Add New** or **Add New Instance**.

Use the **Object Configuration Wizard** to add a library object, program, or task to a project. The **Object Configuration Wizard** consists of two pages, the **Select a Library** page and the **Parameters** page.

- **Select a Library** displays the latest revisions of library objects, separated by category. Select **Show All Revisions** to display all revisions. To quickly search for a library, enter criteria in **Filter**.
- **Parameters** displays the name, description, and configuration parameters for a library object, program, or task. Parameters may be broken up into object-specific tabs, such as SubObjects, Linked Libraries, or Interface Links. Editable fields appear black, non-editable fields appear gray.



Note: If a library object contains links to other linked library objects, these objects can be created automatically. Select the **Linked Libraries** tab (if applicable), then select **Auto Create Linked Objects**.

## See also

[Add a new software object to a controller](#) on [page 56](#)

[Auto Create Linked Objects](#) on [page 59](#)

## Auto Create Linked Objects

How do I open Auto Create Linked Objects?

1. In **Controller Preview**, expand **Controllers**, expand the controller object, right-click a task or program object, then click **Add New**.
2. In the **Object Configuration Wizard**, expand the library categories to display the software libraries registered in the ACM database.
3. Select the library object to add to the selected task or program, then select **Next**.
4. If the software object depends on other linked library objects, select the **Linked Libraries** tab, then select **Auto Create Linked Objects**.

Use **Auto Create Linked Objects** to add objects linked by a library object to a project, rather than manually adding them to the project one at a time.

Hardware modules cannot be Auto Created but can still be linked to if they already exist in the project.

Column	Description
Link Name	Displays the name of the link in the object being added to the project. <b>Auto Create Linked Objects</b> uses the link name to find a matching library.
Library	Displays the name and version of a library that matches the link name. The latest version of the library is always used if the version was not specified. If an exact library match cannot be found, the <b>Library</b> column displays "No unique library matches link criteria". The <b>Action</b> field is locked to <b>Ignore</b> .

Column	Description
Action	<p>Sets the action that <b>Auto Create Linked Objects</b> should take for the linked library:</p> <ul style="list-style-type: none"> <li>• <b>Create New:</b> Creates a new library object with the parameters from the matched library object.</li> <li>• <b>Use Existing:</b> Uses the matched library object.</li> <li>• <b>Ignore:</b> Does not create or use the linked library object. Mandatory linked libraries cannot be set to <b>Ignore</b>.</li> </ul> <p>Select multiple linked objects and use the action list to sets the action to these objects.</p>
Object Name	<p>Browse for an existing object name or set the auto created library object name. Disabled if the library object is parameterized. When an exact match library is found, the match icon displays beside the object name.</p>
Task Name	<p>Assigns the auto created library object to a task. Create a new task by selecting <b>&lt;Add New...&gt;</b>. Disabled if <b>Action</b> is set to <b>Use Existing</b> or <b>Ignore</b>.</p> <p>Select multiple auto created library objects and use the task name list to assign these objects to one task.</p>
Program Name	<p>Assigns the auto created library object to a program. Create a new program by selecting <b>&lt;Add New...&gt;</b>. Disabled if <b>Action</b> is set to <b>Use Existing</b> or <b>Ignore</b>.</p> <p>Select multiple auto created library objects and use the program name list to assign these objects to one program. The program name list is disabled if these objects are unassigned.</p>

**See also**

[Add a new software object to a controller](#) on [page 56](#)

[Object Configuration Wizard](#) on [page 58](#)

## Use ACM Tools

### Use ACM Tools

Use Application Code Manager tools to assist in reusable code development and management.

This table describes each of the tools.

Tool	Description
Import Export Manager	Use to import or export ACM project content between Excel spreadsheets (Schedule). Use for bulk additions, duplications, changes; comparing versions; snapshots; backup customer parameter entry, and/or transferring project contents.
Database Manager	Use to create, delete, upgrade, back up, and restore ACM databases.
Plugins	Add capabilities to Application Code Manager by installing plugins through the <b>Tools &gt; Plugins</b> command.
Log Debug Information	Choose to include debug information in the ACM Log File to help with troubleshooting applications. Information is written to the ACM log file when design outputs are generated (example: <CLX>, FactoryTalk View, FactoryTalk Historian, Word) or when schedules are imported or exported.
Log Viewer	Use to view the contents of the most recent ACM log file. A new ACM log file is created for each ACM session.
Settings	Use to set the location of the system documentation, the language, and the default settings for Control Logix controllers.
Document Template Editor	Use to develop and test document generation templates.
Open Target ACD	Opens the directory and allows navigation to the ACD file location. Use to instantiate library content into an existing file, versus creating a new ACD file.

### See also

[Import Export Manager](#) on [page 61](#)

[Database Manager](#) on [page 73](#)

[ACM Default Settings](#) on [page 79](#)

[Add ACM library content to an existing ACD project](#) on [page 107](#)

### Import Export Manager

The **Import Export Manager** imports and exports ACM project content to and from schedules using the Excel file format (.xlsx and .xlsm). Schedules include the scope of the project (example: Project, Controller, Task, Program, Object, Tag) and content for the project (example: Device List, Device Interlocks).

The **Import Export Manager** has the following controls:

Control	Description
Menu Bar	Includes a <b>File</b> menu for identifying the current project and a <b>Tools</b> menu for setting up templates and configuring logging.
Project:	Displays the name of the current project selected. If creating a new project, this must be a unique name.

Control	Description
Import Tab	<p>Specifies what effect the import will have on the target project and the location of the schedule to import.</p> <ul style="list-style-type: none"> <li>• <b>New</b> - Creates a new project using the specified project name.</li> <li>• <b>Replace</b> - Overwrites the specified project with the project being imported.</li> <li>• <b>Update</b> - Merges the information being imported into the project specified.</li> <li>• <b>Auto Continue</b> - Continues the import process automatically.</li> <li>• <b>Excel File</b> - Enters the path to the schedule Excel file (.xlsx) to import or select the ellipsis to browse to the file location.</li> <li>• <b>Open Backup Folder</b> - Opens the ACM backup folder.</li> <li>• <b>Backup original</b> - Creates a backup of the original project.</li> </ul>
Export Tab	<p>Defines the scope and details of the export schedule.</p> <ul style="list-style-type: none"> <li>• <b>All Projects</b> - Exports all projects in the connected ACM database.</li> <li>• <b>Complete Project</b> - Exports the currently specified project.</li> <li>• <b>Partial</b> - Exports only specific objects and parameter values. To specify the objects to include in the export select the appropriate values from the <b>Type</b> and <b>Controller</b> drop-down menus.</li> <li>• <b>Use Template</b> - Limit the content of the schedule to either: <ul style="list-style-type: none"> <li><b>Local</b> - Locally available devices.</li> <li><b>Project</b> - Devices used in the project being exported.</li> <li><b>Global</b> - All devices in the library.</li> </ul> </li> <li>• <b>Show Modified</b> - Color codes "changed" (not equal to default) and "unchanged" (equal to default) parameter values in the export file.</li> <li>• <b>Export Used Libraries</b> - Includes the libraries used by the projects being exported in the export schedule.</li> <li>• <b>Export</b> - Clicking this button performs the export as specified.</li> <li>• <b>Export and Open</b> - Clicking this button performs the export as specified and opens the exported schedule file. Not available when exporting all projects.</li> </ul>
Compare Tab	<p>Compares the currently selected project to a previously saved schedule file and creates a report.</p> <ul style="list-style-type: none"> <li>• <b>Only Show Changes</b> - When this item is enabled the comparison report will only include items that are different between the current project and the specified schedule file.</li> <li>• <b>Excel File</b> - Provides the path and file name of the schedule file to compare with the current project. Use the ellipsis (...) button to navigate to a file to open.</li> <li>• <b>Compare</b> - Clicking this button runs the compare operation. The results of the comparison are displayed in an Excel spreadsheet.</li> </ul>
Tags Import	<p>Updates parameters in the ACM database with the related tag values in the Tag Configurator workbook. Typically, these values in the workbook are first uploaded from an online Logix Controller using Open Platform Communications (OPC). A Tag Configurator workbook can be used to change, download, and upload a controller's AOI or UDT instance tag values.</p> <ul style="list-style-type: none"> <li>• <b>Controller</b> - Select an ACM controller to update.</li> <li>• <b>Excel File</b> - Enter full path of the Tag Configurator workbook (.xlsm) to import or click the ellipsis (...) button to browse to the file location.</li> <li>• <b>Configuration File</b> - Enter the configuration file name (.xml) or click the ellipsis (...) button to browse to the file location.</li> <li>• <b>Import</b> - Import the AOI or UDT instance tag values from a Tag Configurator workbook to object parameter values as defined in the configuration file.</li> </ul>
Tag Export	<p>Generates a Tag Configurator workbook from ACM with the tag values of the configured ACM object. The Tag Configurator workbook is then used to write these tag values to an online Logix Controller via Open Platform Communications (OPC).</p> <ul style="list-style-type: none"> <li>• <b>Controller</b> - Select an ACM controller containing the tags to export to the Tag Configurator workbook (.xlsm).</li> <li>• <b>Template File</b> - Enter the template file name (.xls) or click the ellipsis (...) button to browse to the file location.</li> <li>• <b>Create New Configuration File</b> - Create and uses a new configuration file based on Tag members decorated with the OPC Tag Export option.</li> <li>• <b>Use Existing Configuration File</b> - Use the defined configuration file to generate the Tag Configurator workbook from the ACM project.</li> <li>• <b>Configuration File</b> - Enter the configuration file name (.xml) or click the ellipsis (...) button to browse to the file location.</li> <li>• <b>Export</b> - Export the AOI or UDT instance tag values to a Tag Configurator workbook.</li> </ul>
IAB/Architect	<p>Creates an ACM partial import file containing Logix hardware modules that are configured in Integrated Architecture Builder (IAB), RSLogix Architect, or Studio5000 Architect. The utility converts the IAB/Architect (.xml) file into an ACM partial import file (.xlsx). Import the partial import file when a corresponding ACM library exists in the database.</p> <ul style="list-style-type: none"> <li>• <b>IAB / RSLogix Architect XML</b> - Enter the configuration file name (.xml) or click the ellipsis (...) button to browse to the file location.</li> <li>• <b>Convert</b> - Create an ACM partial import file.</li> </ul>
Cancel	Closes the <b>Import Export Manager</b> .
<<Previous	Not available
Next>>	Not available

Control	Description
Finish	Closes the <b>Import Export Manager</b> . (Available after the import, export, or compare process is completed.)

## See also

[Import a schedule](#) on [page 63](#)

[Export a schedule](#) on [page 64](#)

[Compare a project to a saved schedule](#) on [page 66](#)

[Create a new schedule template](#) on [page 70](#)

## Import a schedule

Use the **Import Export Manager Import** tab to import objects (instances) and parameter values from a schedule (xlsx). Verify that the correct database name displays in the ACM Title Bar before starting the import process.



Tip: When importing objects (instances), a compatible library (class) must be registered in the ACM database.

The Catalog Number of the registered library must match the Catalog Number in the schedule and a Library with a revision greater than or equal to the revision in the schedule must be registered.

If a library with the same revision is registered, the library with the same revision is used.

If a library with the same revision is not registered and a library with a greater revision is registered, the newest library is used.

## To import a schedule

1. Click **Tools, Import Export Manager** to open the Import Export Manager. The most recently used project opens by default.
2. To choose a different project to import, click **File > Open** and then select the project from the list. If the ACM database does not contain any projects, the Project is blank.
3. Select the import method:
  - **New - Create a new project.**  
Use to bring projects into a new ACM database.
  - **Replace- Overwrite project.**  
Use to revert projects to a previously exported version.
  - **Update - Merge with existing project.**  
Use to add content from other projects, or earlier versions of the same project, to the current project
4. Type a schedule file name (xlsx) in the **Excel File** text box or click the ellipsis to browse to the file.
5. Click **Import**.
  - If you choose **Replace** as the import method, the **Import** dialog box opens.
  - If you choose **New** or **Update** as the import method, the **Import Complete** dialog box opens.

Errors, warnings, info, and debug log entries display.



Tip: Filter the list of exceptions by clicking **Errors**, **Warnings**, **Info**, and/or **Debug**.

6. (optional) Click **Show Log File** on the **Import Complete** dialog box to display the entire contents of the most recent ACM log file in a text editor.
7. If you chose **Replace** as the import method, click **Next** to continue with the import and change the data in the ACM database.  
If you chose **New** or **Update** as the import method, click **Finish** to close the **Import Complete** dialog box.
8. Click **Finish** to close the **Import Export Manager**.

**See also**

[Import Export Manager Import Tab settings](#) on [page 64](#)

**Import Export Manager  
Import tab settings**

The **Import Export Manager Import** tab is used to import objects and parameter values from a schedule file.

This table describes the settings on the **Import Export Manager Import Tab**.

Setting	Description
New - Create new project	Select this option to create a new project based on the imported schedule. The project name in the schedule cannot exist in the connected ACM database.
Replace - Overwrite project	Select this option to replace the specified project with the project in the imported schedule. The project name in the schedule must be the same as the project name specified in the <b>Project</b> text box. When this import option is run the current version of the specified project is exported to the ACM backup folder and deleted from the ACM database before importing the schedule to the project.
Update - Merge with existing project (will not delete existing objects)	Select this option to update the specified project with the information contained in the imported schedule. When this import option is run the current version of the specified project is exported to the ACM backup folder and deleted from the ACM database before merging the imported schedule with the project. Schedule content that is new will be added to the specified project while schedule content that is different between the project in the database and the project in the schedule will be updated to match the content from the schedule. However, schedule content that was deleted or was not exported is not deleted from the selected Project.
Auto-Continue	Select this checkbox to automatically continue the import process using the specified schedule without requiring additional confirmations if no errors or warnings are present.
Excel File:	Type the path to the schedule Excel file (.xlsx) to import or click the ellipsis to browse to the file location.
Open Backup Folder	Opens the ACM backup folder.
Backup original	Creates a backup of the original project.
Import	Imports the specified schedule file to the specified project. The <b>Import</b> button is available when the import configuration is valid.

**See also**

[Import a schedule](#) on [page 63](#)

[Export a schedule](#) on [page 64](#)

**Export a schedule**


Use the **Import Export Manager Export** tab to export objects (instances) and parameter values to a schedule file. Schedules are primarily used for backup



and recovery. Verify that the correct database name displays in the ACM title bar before starting the export process.

## To export a schedule

1. Click **Tools, Import Export Manager** to open the **Import Export Manager**. The most recently used project opens by default.
2. To choose a different project to export, click **File > Open** and then select the project from the list.
3. Click the **Export** tab.
4. Configure the scope of the schedule:
  - **All Projects**  
Use to export all projects in the connected ACM database.
  - **Complete Project**  
Use to export the currently specified project.
  - **Partial**  
Use to export only specific objects and parameter values.
    - Under **Type**, choose either **Controller, Hardware, Object**, or **Library**, then specify the controller and instance to export.
5. (optional) Limit the content (example: Device List, Device Interlocks) of the schedule by selecting the **Use Template** check box. Under the **Type** category the following options are available:
  - Choose **Local** to limit the schedule to locally available devices.
  - Choose **Project** to limit the schedule to only those devices used in the project being exported.
  - Choose **Global** to include all devices in the library.
6. (optional) Select **Show Modified** to color code "changed" (not equal to default) and "unchanged" (equal to default) parameter values.
7. (optional) Select **Export Used Libraries** to include the libraries in the schedule.
8. Click **Export**.
9. Enter a path and file name in the **Save As** dialog.
10. Click **Save** to save the schedule as an Excel Workbook (\*.xlsx) file.
11. If there is an error or warning, the ACM Log File entries are displayed when the export is complete.
 



Tip: Errors, warnings, info, and debug information are displayed by default. Filter the list of exceptions by clicking **Errors, Warnings, Info**, and/or **Debug**.  
Clicking the **Show Log File** command will display the contents of the most recent ACM Log File.  
Click **Finish** to return to the **Import Export Manager**.
12. Click **Finish** to close the **Import Export Manager**.

## See also

[Import Export Manager Export tab settings](#) on [page 66](#)

## Import Export Manager Export tab settings

The **Import Export Manager Export** tab is used to export objects and parameter values to a schedule file.

This table describes the settings on the **Import Export Manager Export Tab**.

Setting	Description
All Projects	Select this option to export a schedule for each project in the connected ACM database. A template cannot be applied.
Complete Project	Select this option to export a schedule for the current project A template can be applied to the export.
Partial	Select this option to export a schedule that includes only selected information from the current project. With this option, these types of items can be specifically exported: <ul style="list-style-type: none"> <li>• Controller</li> <li>• Hardware</li> <li>• Object</li> <li>• Library</li> </ul>
Use Template	Select this check box to limit the items exported to the schedule using a schedule template. When selected, choose to use either a <b>Local</b> , <b>Project</b> , or <b>Global</b> schedule template.
Show Modified	Select this check box to apply color codes in the schedule to denote "changed" (not equal to default) and "unchanged" (equal to default) parameter values.
Export Used Libraries	Includes all the associated library files for the project/controller in the export.
Export	Exports the schedule. The <b>Export</b> button is available when a valid export configuration has been selected.
Export and Open	Exports the schedule and opens the schedule in Excel when the export is complete. The <b>Export and Open</b> button is available when a valid <b>Complete Project</b> or <b>Partial</b> export configuration has been selected.

## See also

[Import Export Template Manager](#) on [page 71](#)

[Export a schedule](#) on [page 64](#)

## Compare a project to a saved schedule

Use the **Import Export Manager Compare** tab to compare the current project to a previously saved schedule file and create a report that highlights the differences.

### To compare a project to a saved schedule

1. Click **Tools, Import Export Manager** to open the **Import Export Manager**. The most recently used project is opened by default.
2. To choose a different project to compare with, click **File > Open** and then select the project from the list.
3. Click the **Compare** tab.
4. In **Excel File** type the name of the schedule file to use for comparison or click the ellipsis to browse to the schedule file.

5. (optional) Select **Only Show Changes** to limit the content of the comparison report to items in the current project that differ from the schedule.
6. Click **Compare**.  
The comparison report opens in Excel.
7. Click **Finish** to close the **Import Export Manager**.

### See also

[Export a schedule](#) on [page 64](#)

[Import Export Manager Compare Tab settings](#) on [page 67](#)

## Import Export Manager Compare Tab settings

Use the **Import Export Manager Compare** tab to compare the current project to a previously saved schedule.

This table describes the settings on the **Import Export Manager Compare Tab**.

Control	Description
Only Show Changes	Select this check box to create a report that only lists the project content that has changed.
Excel File:	Click the ellipsis to identify the schedule file (*.xlsx) to compare with the current project.
Compare	Compares the current project to the selected schedule file. The <b>Compare</b> button is available when a valid schedule has been selected.
Cancel	Cancels the comparison. The <b>Cancel</b> button is available once the comparison is in progress.
Previous	Not used
Next	Not used
Finish	Closes the tab. The <b>Finish</b> button is available once the comparison file has been created.

### See also

[Compare a project to a saved schedule](#) on [page 66](#)

## Update controller parameters with the related tag values

Use the **Import Export Manager Tag Import** tab to update parameters in the ACM database with the related tag values in the Tag Configurator workbook. Typically, these values in the workbook are first uploaded from an online Logix Controller using Open Platform Communications (OPC). A Tag Configurator workbook can be used to change, download, and upload a controller's AOI or UDT instance tag values.

### To update controller parameters with the related tag values

1. Click **Tools > Import Export Manager > Tag Import**.
2. In the **Controller** box, select a controller.
3. In the **Excel File** box, do one of the following:
  - Enter the full path of the Tag Configurator workbook (.xslm).

- Click the ellipsis (...) button to browse to the file location.
4. In the **Configuration File** box, do one of the following:
    - Enter the full path of the configuration file name (.xml).
    - Click the ellipsis (...) button to browse to the file location.
  5. Click **Import**.

**See also**

[Import Export Manager Tags Import tab settings](#) on [page 68](#)

**Import Export Manager Tags Import tab settings**

The **Import Export Manager Tags Import** tab is used to update parameters in the ACM database with the related tag values in the Excel Tag Configurator workbook.

This table describes the settings on the **Import Export Manager Tags Import Tab**.

Setting	Description
Controller	Select an ACM controller to update.
Excel File	Enter the full path of the Tag Configurator workbook (.xism) to import or click the ellipsis (...) button to browse to the file location.
Configuration File	Enter the full path of the configuration file name (.xml) or click the ellipsis (...) button to browse to the file location.
Import	Import the AOI or UDT instance tag values from a Tag Configurator workbook to object parameter values as defined in the configuration file. The <b>Import</b> button is available when the import configuration is valid.

**See also**

[Update controller parameters with the related tag values](#) on [page 67](#)

**Generate a Tag Configurator workbook**

Use the **Import Export Manager Tag Export** tab to generate a Tag Configurator workbook from ACM with the tag values of the configured ACM object. The Tag Configurator workbook is then used to write these tag values to an online Logix Controller via Open Platform Communications (OPC).

**To generate a Tag Configurator workbook**

1. Click **Tools > Import Export Manager > Tag Export**.
2. In the **Controller** box, select a controller.
3. In the **Template File** box, do one of the following:
  - Enter the full path of the template file name(xls.).
  - Click the ellipsis (...) button to browse to the file location.
4. Select **Create New Configuration File** or **Use Existing Configuration File**.

5. In the **Configuration File** box, do one of the following:
  - Enter the full path of the configuration file name (.xml).
  - Click the ellipsis (...) button to browse to the file location.
6. Click **Export**.

### See also

[Import Export Manager Tags Export tab settings](#) on [page 69](#)

## Import Export Manager Tags Export tab settings

The **Import Export Manager Tags Export** tab is used to generate an Excel Tag Configurator workbook from ACM with the tag values of the configured ACM object.

This table describes the settings on the **Import Export Manager Tags Export Tab**.

Setting	Description
Controller	Select an ACM controller containing the tags to export to the Tag Configurator workbook (.xism).
Template File	Enter the full path of the template file name (.xls) or click the ellipsis (...) button to browse to the file location.
Create New Configuration File	Create and use a new configuration file based on Tag members decorated with the <b>OPC Tag Export</b> option.
Use Existing Configuration File	Use the defined configuration file to generate the Tag Configurator workbook from the ACM project.
Configuration File	Enter the full path of the configuration file name (.xml) or click the ellipsis (...) button to browse to the file location.
Export	Export the AOI or UDT instance tag values to a Tag Configurator workbook. The <b>Export</b> button is available when the export configuration is valid.

### See also

[Generate a Tag Configurator workbook](#) on [page 68](#)

## Create an ACM partial import file

Use the **Import Export Manager IAB/Architect** tab to create an ACM partial import file containing Logix hardware modules that are configured in the Integrated Architecture Builder (IAB) or Studio 5000 Architect. The utility converts the IAB/Architect (.xml) file to an ACM partial import file (.xlsx). Import the partial import file when a corresponding ACM library exists in the database.

### To create an ACM partial import file

1. Click **Tools > Import Export Manager > IAB/Architect**.
2. In the **IAB/Architect XML** box, do one of the following:
  - Enter the full path of the configuration file (.xml).
  - Click the ellipsis (...) button to browse to the file location.

3. Click **Convert**.
4. In **Object Configuration Wizard**, configure properties of the imported file.
5. Click **Finish**.

**See also**

[Import Export Manager IAB/Architect tab settings](#) on [page 70](#)

**Import Export Manager IAB/Architect tab settings**

The **Import Export Manager IAB/Architect** tab is used to create an ACM partial import file containing Logix hardware modules that are configured in the Integrated Architecture Builder (IAB) or Studio5000 Architect.

---

**IMPORTANT** Select the EPlan XML export option when exporting the configuration file from IAB.

---

This table describes the settings on the **Import Export Manager IAB/Architect Tab**.

Setting	Description
IAB/Architect XML	Enter the full path of the configuration file (.xml) or click the ellipsis (...) button to browse to the file location.
Convert	Create an ACM partial import file.

**See also**

[Import Export Manager](#) on [page 61](#)

[Create an ACM partial import file](#) on [page 69](#)

**Create a new schedule template**

Use the **Template Manager** tool of the **Import Export Manager** to create schedule templates. Schedule templates determine the scope of the schedule (local, project, or global), the library objects used, and the parameters that are visible.

**To create a new schedule template**

1. Click **Tools > Import Export Manager**.  
The **Import Export Manager** appears.
2. In the **Import Export Manager**, click **Tools > Import Export Template Editor**.  
The **Template Manager** appears with the **Template Editor** tab displayed.
3. In the **Template** area, make the following selections:
  - a. Select the scope of the template:
    - **Local**
    - **Project**

- **Global**
  - b. (if project scoped) Click the down arrow and select the project name.
  - c. Click the down arrow and select the schedule template.
- 4. Click **New**. The **Template Name** dialog box appears.
- 5. In **New Template Name**, enter a name for the new schedule template and click **OK**.  
The new schedule template name appears in the **Template** area.
- 6. In the **Library Section Selection** area, add the object and sub-object parameters to the new schedule template.
  - a. Select an object in the **Library Section Selection** area.
    - b. In the **Not Visible Parameters** area select the parameters to include.  
Click **Select All** to select all of the parameters for an object.
    - c. Click **Add >**.
    - d. Repeat as needed.
    - e. In the **Visible Parameters** area, order the parameters as they should appear in the template file.
- 7. Click **Save** to save the changes.
- 8. Click **Reload** to cancel all edits made since the last time the **Save** command was executed.
- 9. Click **Finish** to close the **Import Export Template Manager**.

### See also

[Import a schedule](#) on [page 63](#)

## Import Export Template Manager

How do I open the Template Manager?

1. Click **Tools, Import Export Manager** to open the **Import Export Manager**.
2. In the **Import Export Manager** window, click **Tools > Import Export Template Editor**.

The **Template Manager** can be used to create custom schedule templates, copy schedule templates, or move schedule templates from one location to another.

Three schedule template locations are available:

- **Local** – Located in the Windows User Folder. Available only to the ACM User.
- **Project** – Located in the ACM database. Available to all ACM users with this project currently open.
- **Global** – Located in the ACM database. Available to all ACM users connected to this ACM database.

**See also**

[Template Manager Template Editor tab settings](#) on [page 72](#)

[Template Manager Copy/Move Templates tab settings](#) on [page 72](#)

[Database Manager](#) on [page 73](#)

## Template Manager Copy/Move Templates tab settings

The **Copy/Move Templates** tab is used to copy or move a template.

This table describes the settings on the **Copy/Move Templates** tab.

Setting	Description
Source Template	Selects a schedule template. Select a schedule template location, a project name (if located in a project), and a schedule name.
<ul style="list-style-type: none"> <li>• Rename</li> <li>• Duplicate</li> <li>• Delete</li> </ul>	Rename the source template Duplicate the source template Delete the source template
Copy	Copies the source template to the location specified in <b>Destination Template</b> .
Move	Moves the source template to the location specified in <b>Destination Template</b> .
Destination Template	Selects a location to move or copy the selected source template to. Choose either <ul style="list-style-type: none"> <li>• <b>Global</b> - Places the template in the ACM database and makes it available to all users</li> <li>• <b>Project</b> - Places the template in the ACM database and makes it available when the specified project is opened.</li> <li>• <b>Local</b> - Places the template in the locate user template folder on the local computer. The template is only available to the local user.</li> </ul>
Open User Template Folder	Opens the file browser scoped to the local user template folder.
Finish	Close the <b>Template Manager</b> and returns to the <b>Import Export Manager</b> .

**See also**

[Import Export Template Manager](#) on [page 71](#)

## Template Manager Template Editor tab settings

Use the **Template Editor** tab of the **Template Manager** to create, delete, or edit schedule templates.

This table describes the settings on the **Template Editor** tab.

Setting	Description
Template	Selects a schedule template. Use the combo boxes to select a schedule template location, a project name (if located in a project), and a schedule name. Click <b>New</b> to create a new schedule template Click <b>Delete</b> to delete the selected schedule template. If the schedule template format is old, <b>Template Manager</b> asks for permission to upgrade the template format to the latest version when the schedule template is selected.
Library Section Selection	Selects a library object or sub-object definition from the registered libraries. The object or sub-object parameters display in the <b>Not Visible Parameters</b> or <b>Visible Parameters</b> list boxes.



Setting	Description
Not Visible Parameters	Displays the object or sub-object parameters that are not be included in schedules exported using this schedule template. To include a parameter in the exported schedule, select the item from the <b>Not Visible Parameters</b> list and then click <b>Add</b> to move it to the <b>Visible Parameters</b> list. Use the mouse key combinations Ctrl-click, Shift-Click, Click-and-Drag, and the <b>Select All</b> , and <b>Deselect All</b> buttons to select multiple parameters.
Visible Parameters	Displays the object or sub-object parameters included in schedules exported using this schedule template. To remove a parameter from the exported schedule, select the item from the <b>Visible Parameters</b> list and then click <b>Remove</b> to move it to the <b>Not Visible Parameters</b> list. Use the mouse key combinations Ctrl-click, Shift-Click, Click-and-Drag, and the <b>Select All</b> , and <b>Deselect All</b> buttons to select multiple parameters. Use the <b>Up</b> , <b>Down</b> , <b>Top</b> , and <b>Bottom</b> commands to control the order in which the parameters display in the exported schedule. Use the sort buttons to display the parameters in category groups or alphabetically.
Add >	Moves the parameter selected in the <b>Not Visible Parameter</b> list to the <b>Visible Parameter</b> list.
< Remove	Moves the parameter selected in the <b>Visible Parameter</b> list to the <b>Not Visible Parameter</b> list.
Save	Saves the schedule template to the schedule template settings file (C:\Documents and Settings\ <username>\Local Settings\Application Data\ Rockwell Automation\Application Code Manager \Templates\<template&gt;).< td=""> </template&gt;).<></username>
Reload	Cancel any template changes made since the last <b>Save</b> command was executed by reloading the schedule template from the schedule template settings file (C:\Documents and Settings\ <username>\Local Settings\Application Data\Rockwell Automation\Application Code Manager \ Templates\<template&gt;).< td=""> </template&gt;).<></username>
Finish	Closes <b>Template Manager</b> .

## See Also

[Template Manager Copy/Move Templates tab settings](#) on [page 72](#)

## Database Manager

The **Database Manager** controls how Application Code Manager interacts with the SQL Server database. It also provides the ability to create, backup, and recover databases.

## See also

[Create an ACM database](#) on [page 73](#)

[Upgrade an ACM database](#) on [page 74](#)

[Backup an ACM database](#) on [page 75](#)

[Restore an ACM database](#) on [page 75](#)

[Database Manager settings](#) on [page 77](#)

## Create an ACM database

Application Code Manager (ACM) uses a database to store re-usable code. At least one database is required, but additional databases can be created as needed.

### To create an ACM database

1. Open Application Code Manager, in the main menu, click **Tools > Database Manager**.

2. In **Server Name**, select the computer name and SQL Server instance from the drop down list or type a computer name and SQL Server instance in the following format:  
<Computer Name> \ <SQL Server Instance>
3. In **Log on to the server**, use Windows Authentication or type the **User name** and **Password** to use to authenticate the connection to the SQL Server, then click **Connect**.  
**Status** updates to **Connected**.
4. In **Specify the database**, type a unique name for the ACM database.
5. In **Actions**, select **Create database** and then click **Execute Task**.  
Once the database is created an **Action completed successfully** message displays. Click **OK**.
6. Click **Close** to close **Database Manager**.

### See also

[Connecting to an ACM database](#) on [page 22](#)

## Upgrade an ACM database

Application Code Manager (ACM) uses a database to store re-usable code. If ACM is upgraded to a newer version the existing ACM database can still be used, but the database schema must be upgraded.

### To upgrade an ACM database

1. Open Application Code Manager, in the main menu, click **Tools > Database Manager**
2. In **Server Name**, select the computer name and SQL Server instance from the drop down list or type a computer name and SQL Server instance in the following format:  
<Computer Name> \ <SQL Server Instance>
3. In **Log on to the server**, use Windows Authentication or type the **User name** and **Password** to use to authenticate the connection to the SQL Server, then click **Connect**.  
**Status** updates to **Connected**.
4. In **Specify the database**, select the name of the ACM database you want to upgrade.
5. In **Actions**, select **Upgrade database** and then click **Execute Task**.  
Once the database is created an **Action completed successfully** message displays. Click **OK**.
6. Click **Close** to close **Database Manager**.

## See also

[Connecting to an ACM database](#) on [page 22](#)

[Upgrade the Application Code Manager application](#) on [page 16](#)

[Backup an ACM database](#) on [page 75](#)

## Backup an ACM database

Application Code Manager (ACM) uses a database to store re-usable code. Making a backup of the ACM database on a regular schedule is recommended to ensure that your re-usable code library can be restored in case of hardware failure or unintended changes to the data.

---

**IMPORTANT** When backing up a remote database, the file location must be accessible with read and write access granted to the user account under which the remote database service runs.

---

## To backup an ACM database

1. Open Application Code Manager, in the main menu, click **Tools > Database Manager**
2. In **Server Name**, select the computer name and SQL Server instance from the drop down list or type a computer name and SQL Server instance in the following format:  
 <Computer Name> \ <SQL Server Instance>
3. In **Log on to the server**, use Windows Authentication or type the **User name** and **Password** to use to authenticate the connection to the SQL Server, then click **Connect**.  
**Status** updates to **Connected**.
4. In **Specify the database**, select the ACM database you want to backup.
5. In **Actions**, select **Backup database** and then click **Execute Recovery**. The **Select file to backup database** dialog box opens.
6. In **File name** type a name for the backup file being created and then click **Save**.  
 Once the backup file is created an **Action completed successfully** message displays. Click **OK**.
7. Click **Close** to close **Database Manager**.

## See also

[Connecting to an ACM database](#) on [page 22](#)

## Restore an ACM database

Application Code Manager (ACM) uses a database to store re-usable code. If ACM is being moved to new database server or if the existing database server has incorrect ACM data, the backup made of the database can be restored to the database server so that all of the previous data is available.

**IMPORTANT** When restoring a remote database, the file location must be accessible with read and write access granted to the user account under which the remote database service runs. Restoring an ACM database backup **onto a machine other than the one on which it was created** will require database ownership to be set. This is because the user associated with the backed-up database does not exist in the new environment. If ownership is not set, then ACM users will continuously receive the following message: “Timed out waiting for the heart beat notification from the SQL Broker from within the ACM application. Please check the log file for details”.

Database ownership can be set by the database administrator. The database administrator will need to run the script as below within either a SQL Management Studio query window or via a SQLCMD session:

```
ALTER AUTHORIZATION on DATABASE::[Database Name] TO [SQL User]
```

- The [Database Name] section must be replaced with the ACM database name.
- The [SQL User] section must be replaced with the name of the user who will be assigned ownership of the database.

Running the above script ensures that you will no longer receive the timeout messages.

## To restore an ACM database

1. Open Application Code Manager, in the main menu, click **Tools > Database Manager**
2. In **Server Name**, select the computer name and SQL Server instance from the drop down list or type a computer name and SQL Server instance in the following format:  
 <Computer Name> \ <SQL Server Instance>
3. In **Log on to the server**, use Windows Authentication or type the **User name** and **Password** to use to authenticate the connection to the SQL Server, then click **Connect**.  
**Status** updates to **Connected**.
4. In **Specify the database**, select the name of the ACM database on which you want to restore the data.
 
 Tip: If you are restoring databases to a new database server or if the database is not present on the database server, create the database before restoring the data to it.
5. In **Actions**, select **Restore database** and then click **Execute Recovery**. The **Selected database backup to restore** dialog box opens.
6. Select the database backup file (.bak) to restore and then click **Open**.
7. The **Confirm database restore** dialog box opens warning you that the current database will be overwritten and asking you if you are sure you want to restore the database.  
 Click **Yes** to confirm the restoration process.
8. Once the database is restored an **Action completed successfully** message displays. Click **OK**.
9. Click **Close** to close **Database Manager**.

## See also

[Connecting to an ACM database](#) on [page 22](#)

## Delete an ACM database

Application Code Manager (ACM) uses a database to store re-usable code. At least one database is required.

### To delete an ACM database

1. Open Application Code Manager, in the main menu, click **Tools > Database Manager**
2. In **Server Name**, select the computer name and SQL Server instance from the drop down list or type a computer name and SQL Server instance in the following format:  
<Computer Name> \ <SQL Server Instance>
3. In **Log on to the server**, use Windows Authentication or type the **User name** and **Password** to use to authenticate the connection to the SQL Server, then click **Connect**.  
**Status** updates to **Connected**.
4. In **Specify the database**, select the ACM database to delete.
5. In **Actions**, select **Delete database** and then click **Execute Task**. The **Confirm database delete** dialog box opens.
6. Click **Yes** to confirm deletion of the database  
Once the database is deleted an **Action completed successfully** message displays. Click **OK**.
7. Click **Close** to close **Database Manager**.

### See also

[Create an ACM database](#) on [page 73](#)

[Restore an ACM database](#) on [page 75](#)

## Database Manager settings

How do I open the Database Manager dialog box?

- On the main menu, click **Tools > Database Manager**.

The **Database Manager** dialog box contains the settings that control how Application Code Manager interacts with the SQL Server database. It also provides the ability to create, backup, and recover databases.

This table describes the settings in the **Database Manager** dialog box.

Setting	Description
Server name:	Identifies the server that is hosting the SQL Server used by ACM. Use the drop-down list to select a computer name and SQL Server instance from, or enter a computer name and SQL Server instance in the following format: <b>&lt;Computer Name&gt; \ &lt;SQL Server Instance&gt;</b>
Refresh	Queries the network for a list of SQL Server computer names and instances and updates the selections available in the drop-down list with the results.
Log on to the server Using Windows Authentication or SQL Server Authentication <i>(Note: This account and password are configured during installation of SQL Server.)</i>	

Setting	Description
<ul style="list-style-type: none"> <li>User name:</li> </ul>	The SQL Server user name associated with the ACM database. The default user name is "sa".
<ul style="list-style-type: none"> <li>Password:</li> </ul>	The SQL Server password associated with the user name.
<ul style="list-style-type: none"> <li>ACM default sa password</li> </ul>	Use the default ACM password instead of typing a password. Check this box if the default SQL Server password was entered when SQL Server was installed.
<ul style="list-style-type: none"> <li>Connect</li> </ul>	When this button is clicked Application Code Manager attempts to connect to the database named in the <b>Select a database name</b> on the SQL Server entered in the <b>Server name</b> combo box.
<ul style="list-style-type: none"> <li>Status:</li> </ul>	Displays the result of the database connection attempt. <ul style="list-style-type: none"> <li>Connected</li> <li>Login failed for user</li> </ul>
Specify the database	
<ul style="list-style-type: none"> <li>Select a database name</li> </ul>	Identifies the database to manage. Enter a name or select an existing name from the list. If the name entered is not present on the SQL Server, the option is provided to create a new database.
Actions - Tasks	
<ul style="list-style-type: none"> <li>Create database</li> </ul>	Creates a database with the database name entered in the <b>Select a database name</b> on the SQL Server entered in the <b>Server name</b> when <b>Execute Task</b> is clicked.
<ul style="list-style-type: none"> <li>Upgrade database</li> </ul>	Upgrades the ACM database named in the <b>Select a database name</b> combo box with the current schema when <b>Execute Task</b> is clicked.
<ul style="list-style-type: none"> <li>Delete database</li> </ul>	Deletes the database named in <b>Select a database name</b> from the SQL Server identified in <b>Server name</b> when <b>Execute Task</b> is clicked.
<ul style="list-style-type: none"> <li>Execute Task</li> </ul>	Performs the selected task.
Actions - Disaster Recovery	
Backup database	Makes a backup copy of the database file named in <b>Select a database name</b> when <b>Execute Recovery</b> is clicked.
Restore database	Replaces the database file named in <b>Select a database name</b> combo box with the backup database file when <b>Execute Recovery</b> is clicked.
Execute Recovery	Performs the selected recovery.

### See also

[Create an ACM database](#) on [page 73](#)

[Upgrade an ACM database](#) on [page 74](#)

[Delete an ACM database](#) on [page 77](#)

[Backup an ACM database](#) on [page 75](#)

[Restore an ACM database](#) on [page 75](#)

## Log Debug Information

To assist in troubleshooting Application Code Manager enable logging of debug information to the ACM Log File.

Information is written to the ACM log file when design outputs are generated (example:<CLX>, FactoryTalk View, FactoryTalk Historian, Word) or when schedules are imported or exported.

Once the additional logging information is no longer needed, disable log debug information.

## To log debug information

- Click **Tools > Log Debug Information**.

A check mark appears next to the **Log Debug Information** entry in the **Tools** menu. Clicking the item again disables logging of debug information.

## See also

[Use ACM Tools](#) on [page 61](#)

## Log File Viewer

The **Log File Viewer** provides a quick way to view the most recent events sent to the ACM log file. The Log File Viewer can be opened from the Application Code Manager **Tools** menu and from the **Import Export Manager Tools** menu.

The **Log File Viewer** displays errors, warning, informational, and debug events in a a grid.

At the top of the **Log File Viewer** is a set of buttons listing of the type of log entries and the number of entries for each type:



The entries displayed in the event grid can be filtered by type of events by clicking the button corresponding to the item to remove from the list.

When an item is not included in the view the button is shaded gray:



By default, the **Log File Viewer** displays events in ascending date time order.

The order can be changed to be either ascending or descending and be based on any of the columns displayed in the events grid. Click on the column heading to sort the display by **Type**, **Date/Time** or **Description**. A small arrow appears next to the column heading being sorted and denotes whether the current sort order is by ascending (up arrow) or descending (down arrow) values.

## See also

[Use ACM Tools](#) on [page 61](#)

[Import Export Manager](#) on [page 61](#)

[Log Debug Information](#) on [page 78](#)

## ACM Default Settings

How do I open ACM Default Settings?

- From the **Tools** menu, select **Settings**.

Use **ACM Default Settings** to specify the location of the system file, the language shown in the display, how Control Logix content is generated, and whether or not to show indications.

Setting	Description
System Documentation Path	The location of the template files. By default these are located at <i>IUsers\Public\Documents\Studio 5000\Templates\Application Code Manager</i> .
Language	The language used by Application Code Manager. The default value uses the operating system language.
Auto-expand Controller Preview	Determines whether the tree in <b>Controller Preview</b> is automatically expanded. The default value is selected.
Control Logix Generation Defaults	Determines what is selected to include in Logix code generated by ACM and whether Application Code Manager should overwrite existing output with new code.
Overwrite Existing Output	The default setting for <b>Overwrite Existing</b> in the Logix Code Generation window.
ACM Project Data	The default setting for <b>ACM Project Data</b> in the Logix Code Generation window.
Create ACD	The default setting for <b>Create ACD</b> in the Logix Code Generation window.
Indications	Determines whether indications are shown.
Modified Objects	Sets whether an indication is shown when a library object has been modified. The default value is cleared.
Newer Library Version Available	Sets whether an indication is shown when a node or folder contains one or more libraries that has a newer version available in <b>Registered Libraries</b> . The default value is cleared.

### See also

[Use ACM Tools](#) on [page 61](#)

[Generate a controller file](#) on [page 48](#)

[Create a new schedule template](#) on [page 70](#)

[Indications](#) on [page 39](#)

## Document Template Editor

How do I open the Document Template Editor?

- From the **Tools** menu, select **Document Template Editor**.

The **Document Template Editor** provides an environment to develop and test document generation templates.



Note: The ability to use this is only available if the product has a Standard activation license and is not available in Lite mode.

The **Document Template Editor** includes the following functions:

- Highlights reserved words in the template language.
- Folds repeat blocks.
- Error checking and highlighting.
- Ability to save frequently used code fragments for later re-use.
- Generates an output template file.

### See also

[Components on the Document Template Editor](#) on [page 81](#)

[Menu bar](#) on [page 82](#)

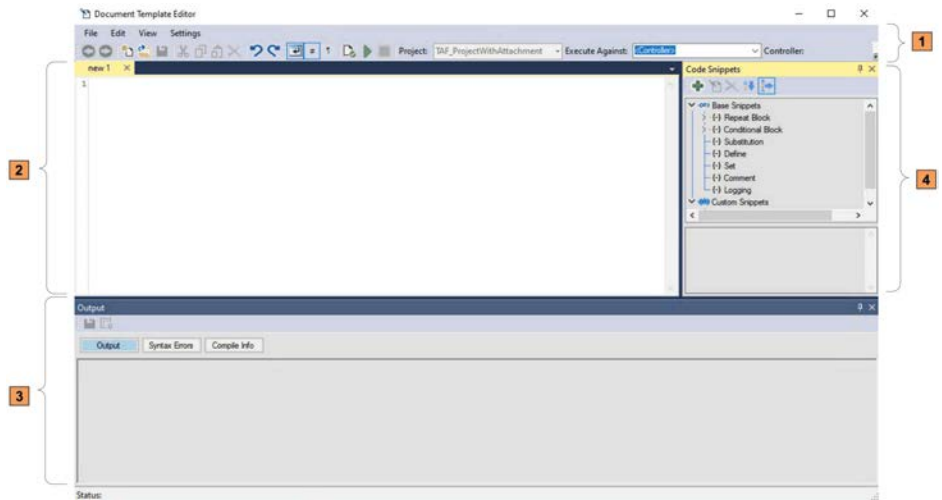


[Tool bar](#) on [page 84](#)

[Output panel](#) on [page 86](#)

[Editor pane](#) on [page 86](#)

## Components on the Document Template Editor



This table describes each section of the **Document Template Editor**.

Item	Name	Description
1	Menu and toolbar	Selects editing options and features or application configuration options in this section.
2	Editor	Develops and tests templates in this section.
3	Output	Displays the <b>Output</b> , <b>Syntax Error</b> , and <b>Compile Info</b> panes. <ul style="list-style-type: none"> <li>• <b>Output</b> Displays output results from the template after the generation process. You can save the output to a file.</li> <li>• <b>Syntax Error</b> Displays any current syntax error descriptions that appear in the <b>Editor</b> pane.</li> <li>• <b>Compile Info</b> Provides compilation information about the previously executed template generation process.</li> </ul>
4	Code Snippets	Displays usable code snippets, including <b>Base Snippets</b> and <b>Custom Snippets</b> . <ul style="list-style-type: none"> <li>• <b>Base Snippets</b> Includes snippets, which are the predefined template instructions that make up the template generation language.</li> <li>• <b>Custom Snippets</b> Includes snippets, which are reusable code fragments consist of pre-configured base snippets.</li> </ul> Drag snippets to the <b>Editor</b> pane for implementation in a template.

The **Editor**, **Output**, and **Code Snippets** panes can be undocked and moved around on the desktop to suit your requirements.

### See also

[Menu bar](#) on [page 82](#)

[Tool bar](#) on [page 84](#)

[Output panel](#) on [page 86](#)

[Editor pane](#) on [page 86](#)

## Menu bar

Selects editing options in the **Menu** bar. This table describes each option.

Item	Description
File	<ul style="list-style-type: none"> <li>• <b>New</b> Creates a new file.</li> <li>• <b>Open</b> Opens an existing file.</li> <li>• <b>Save</b> Saves the currently active template document if it changed.</li> <li>• <b>Save As</b> Saves the currently active template document with a different file name.</li> <li>• <b>Close All</b> Closes all active windows.</li> <li>• <b>Recent Files</b> Displays the last ten recently closed template document files.</li> <li>• <b>Exit</b> Closes the <b>Document Template Editor</b>.</li> </ul>
Edit	<ul style="list-style-type: none"> <li>• <b>Cut</b> Cuts the selected text in the <b>Editor</b> pane to the clipboard. This option is only active if more than one character is selected in the active <b>Editor</b> pane.</li> <li>• <b>Copy</b> Copies the selected text in the <b>Editor</b> pane to the clipboard. This option is only active if more than one character is selected in the active <b>Editor</b> pane.</li> <li>• <b>Paste</b> Pastes text from the clipboard into the current position selected in the <b>Editor</b> pane. This option is only active if the clipboard is not empty.</li> <li>• <b>Clear Clipboard</b> Clears all items copied to the clipboard.</li> <li>• <b>Delete</b> Deletes selected text in the <b>Editor</b> pane. This option is only active if more than one character is selected in the active <b>Editor</b> pane.</li> <li>• <b>Undo</b> Undoes the last operation executed in the active <b>Editor</b> pane.</li> <li>• <b>Redo</b> Redoes the last undo operation.</li> <li>• <b>Find / Replace</b> Starts the <b>Find / Replace</b> dialog box where text strings can be found and replaced.</li> </ul>

Find / Replace	<ul style="list-style-type: none"> <li>• <b>Match Case</b> Matches words that conform exactly to the sequence of upper and lower case text.</li> <li>• <b>Match whole word</b> Matches against entire word sequences. If a word contains the sequence but contains other characters, then the match will not be found.</li> <li>• <b>Regular Expression</b> Includes special non printable characters that may be included in a search word. As an example, a search word containing a carriage return character sequence would only be matched against words in the search document where the word includes the carriage return. These special characters will not be included in the search when this option is not selected i.e. a match would be found on a word that may include a special character like a carriage return character.</li> <li>• <b>Wildcards</b> Includes the "*" character in the search word. Characters appearing between the sequence will be matched i.e. in the search word "R*t", Repeat would be a match as "epea" are treated as wild card characters. If "?" is included in the search word then any single character will be matched in the placeholder where the wildcard is placed i.e. in the search word "Repe?at", "Repe1at" will be a match.</li> <li>• <b>Search up</b> Starts the search at the end of the document and works its way up to the start. Otherwise if not selected then the search would begin as normal from the beginning of the template document.</li> </ul>
View	<ul style="list-style-type: none"> <li>• <b>Restore Default Layout</b> Restores all undocked windows back to their original positions.</li> <li>• <b>Float All Windows</b> Undocks all docked windows.</li> <li>• <b>Snippets View</b> Closes or opens the <b>Code Snippets</b> pane.</li> <li>• <b>Output View</b> Closes or opens the <b>Output</b> pane.</li> </ul>
Settings > Preferences > General	<ul style="list-style-type: none"> <li>• <b>Show Toolbar</b> Adds or removes the tool bar.</li> <li>• <b>Show Menubar</b> Adds or removes the menu bar. Press F12 to show the menu bar if it is removed.</li> <li>• <b>Show Statusbar</b> Sets the status bar displayed at the bottom be visible or not.</li> </ul>

Settings > Preferences > Editing	<ul style="list-style-type: none"> <li>• <b>Display Line Numbers</b> Shows or hides the editor line numbers depending on this selection. This option is global for all open editor panes and the setting is saved and will be applied on application startup.</li> <li>• <b>Enable Line Wrap</b> Wraps the text in the editor to the space available in the Editor pane depending on this selection. This option is global for all open editor panes and the setting will be saved and applied on application startup.</li> <li>• <b>Display End Of Line Character</b> Shows the carriage return character when this option is selected. This option is global for all open editor panes and the setting will be applied on application startup.</li> </ul>
Settings > Preferences > Compile	<ul style="list-style-type: none"> <li>• <b>Compile Messages</b> Two selections are possible, either <b>Basic</b> or <b>Advanced</b>. When generating documentation compile information will be generated in the <b>Compile Info</b> pane. This information can be useful to debug a template or to determine how long the document generation process is spending within specific instruction loops. The <b>Advanced</b> option is verbose intensive and provides detailed information on the generation process.</li> <li>• <b>Send Messages to ACM log</b> Enables or disables the writing of document generation messages to the ACM log.</li> </ul>

**See also**

[Components on the Document Template Editor](#) on [page 81](#)

**Tool bar**

Selects application configuration options in the **Tool** bar. This table describes the options in the **Tool** bar.

Item	Description
New File	Creates a new template file. A new <b>Editor</b> pane will be created starting with the name new (x), (x) being the first new unique number used after the last new file was opened i.e. if a new unsaved file is already loaded with the name <b>new 1</b> , then this editor pane will get the name <b>new 2</b> . Once the file is saved to disk then this name will be replaced with the actual file name of the template document.
Open File	Opens the <b>File Browse dialog</b> to navigate and select a template file to open. Files of type (.txt), (.csv) and (.xml) can be selected and opened.

Save File	Saves the document template file. This button will be enabled when changes are made to the currently active <b>Editor</b> pane. If the file has not been previously saved but is a new file then the <b>Save As</b> dialog will be presented to the engineer allowing for a file name and location to be selected.
Cut Text	Cuts the selected text. This button will be enabled when at least one character in the active <b>Editor</b> pane is selected. When clicked, selected text will be cut from the page and stored on the clipboard.
Copy Text	Copies the selected text. This button will be enabled when at least one character in the active <b>Editor</b> pane is selected. When clicked, selected text will be copied and stored on the clipboard.
Paste Text	Pastes the copied text. This button will be enabled when there is information stored on the clipboard and if an active <b>Editor</b> pane is currently selected. When clicked, the contents of the clip board will be pasted into the active editor pane at the currently selected location.
Delete Text	Deletes the selected text. This button will be enabled when at least one character in the active editor pane is selected. When clicked then the selected text will be deleted from the <b>Editor</b> pane.
Undo	Undoes the previous edit operation in the active <b>Editor</b> pane.
Redo	Redoes the previous undo operation in the active <b>Editor</b> pane.
Word Wrap	Wraps the text in the editor to the space available in the <b>Editor</b> pane. If this function is switched off then text for a line will extend off the page and the horizontal draw bar will need to be used to view parts of the line that extend past the end of the editor panel. This option will only apply to the currently active editor window and will not be applied on application startup.
Line Numbers	Displays a vertical list of line numbers on the left side of the <b>Editor</b> pane when this option is enabled.
Show EOL Char	Displays end of line characters (CR / LF). This option will only apply to the currently active editor window and will not be applied on application startup.
Validate	Runs a validation check on the currently active editor pane. Any detected syntax errors will be displayed in the output <b>Syntax Error</b> pane. If no syntax errors are encountered then the following prompt will be displayed.
Generate	Runs a document generation process for the active <b>Editor</b> pane. Output, syntax errors and compile information will be displayed in the <b>Output</b> pane. Compile information will be displayed in the status bar on the left hand side and progress on the right side while the document generation process is running.
Stop Generation	Stops the generation process. This button will only be enabled when a document generation process is underway.
Selected Project	Displays the currently loaded ACM project. The <b>Document Generation Editor</b> will always use the loaded project that is active within ACM. This field is not configurable.
Selected ACM Object	Compiles document template against various available ACM objects. All available project objects will show in a grouped drop down list control. The categories of items available include Controller, Display, Alarms and Historian. The list will only contain items appearing in the project. All objects in the <b>Execute Against</b> list can be compiled against a specific controller or all controllers in the project by selecting a controller option from the <b>Controller</b> selection option (discussed in the next section), except for <b>Display</b> objects which can only be compiled against all controllers. When selecting a <b>Display</b> item, the <b>Controller</b> selection list will be disabled with the <b>Generate All Controllers</b> option will be pre-set.

Selected Controller	Displays all controllers configured in the ACM project. Once a project object has been selected as outlined in the previous section (excluding display objects), a single controller or all controllers can be selected to compile against. Output would then only be generated for the specific controller selection. This option will be disabled if a display object is selected in the <b>Execute Against</b> list discussed in the previous section.
---------------------	---

**See also**

[Components on the Document Template Editor](#) on [page 81](#)

**Output panel**

The **Output** panel contains three tabs for the following output information: Generated output, syntax error and compilation activity output.

Tab	Description
Output	The <b>Output</b> pane contains the result of a documentation generation process as processed through the loaded template. It is also possible to save the document to either (.txt), (.csv), (.xml) via the save button on the <b>Output</b> toolbar. The <b>Clear Display</b> button is also available for clearing all output from the pane.
Syntax Error	The <b>Syntax Error</b> panel contains a description of any syntax errors detected by the documentation engine. One line will be printed for every error detected. The line number of where the error occurs as well as the type of error will be included in the error message. The <b>Syntax Error</b> panel will only contain syntax errors for the currently active editor pane.
Compile Info	The <b>Compile Info</b> panel contains information pertaining to the document generation process. Each line will always print out a date and time for the event, followed by the actual message description. When the <b>Advanced</b> option is selected see section (Compile Options) then more detailed information would be printed out such as a list of project libraries in use, objects cached etc. Sometimes a compilation error may be encountered. An example of a compilation error would be where a defined object name could not be resolved. This error would be highlighted in blue as follows:

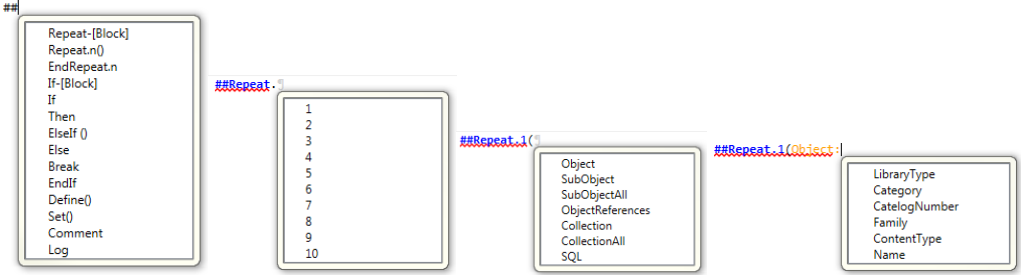
**See also**

[Components on the Document Template Editor](#) on [page 81](#)

**Editor pane**

This table describes the functions of the **Editor** pane.

Item	Description
Language keyword highlighting	The <b>Editor</b> pane supports language text highlighting for the documentation generation language. Document generation instructions are always highlighted in blue: <code>##Repeat.1</code> Library items in orange: <code>Object: LibraryType = "Module")</code> Comments will be printed in green: <code>##! Repeat loop 1 starting.</code>

Code completion	<p>The <b>Editor</b> pane provides code completion functionality where a list of keywords is supplied after a matching text sequence has been typed. This allows the engineer to add an instruction without having to type out the complete text sequence.</p> <p>Here is a graphic example of <b>Code Completion</b> options available through a sequence of typed characters:</p>  <p>With the completed line looking as follows: <code>##Repeat.1(Object: LibraryType = "Controller");</code></p>
Text folding	<p>Text folding consists of grouping a section of code so that it can be collapsed as a complete section to make other areas of the document easier to read. In the above example we are creating folding sections around each repeat block, identified by the repeat number. The Repeat.2 block is encased within the Repeat.1 block and is shown as collapsed within the Repeat.1 block. The Repeat.1 block can also be collapsed.</p>
Working with custom snippets	<p>Click <b>Add Code To Custom Snippet</b> will invoke the <b>Create New Snippet</b> editor where it will be possible to create a new custom snippet.</p> <p>Click <b>Insert Snippet</b> will display a list of base and custom snippets which can be selected for inclusion in the active template document.</p> <p><b>Tip:</b> Some sample snippets are provided. Right-click <b>Custom Snippets</b> and select <b>Import</b>.</p> <p>From the Open dialog box navigate to the default file location: C:\Users\Public\Documents\Studio 5000\Samples\Application Code Manager\Snippets Select the sample snippet to import, repeat if required.</p>
Executing a code fragment	<p>Right click in the <b>Editor</b> pane and select <b>Execute Selected Fragment</b> to execute a code fragment or highlight a section of code. The <b>Execute Selected Fragment</b> menu option is only be enabled if text has been selected in the editor:</p> <p><b>Important:</b> Select and execute a fragment could result in syntax errors being generated as it's possible to select any combination of lines for execution which could result in required block termination instructions been excluded when document generation processing takes place.</p> <p>In the above example the <code>##Repeat.1</code> block instruction has been selected and if executed will result in a syntax error as the fragment does not include an <code>##EndRepeat.1</code> instruction.</p>

## See also

[Components on the Document Template Editor](#) on [page 81](#)

## Supported instructions

Use instructions to develop a document template in the **Editor** pane. The **Document Template Editor** supports the following instructions:

- Repeat
- Break
- Conditional
- Define
- Set
- Substitution
- Comment
- Log

## See also

[Document Template Editor](#) on [page 80](#)

[Editor pane](#) on [page 86](#)

## Repeat instruction

Looping is the most important functionality in the template language and determines where the documentation engine attaches its data source. In

essence the looping statement opens a data set specified by a query string, and then iterates each record within the dataset. All the text contained within the loop will be repeated for every record returned by the specified query.

The basic layout of the looping statement is as follows:

```
##Repeat.n(RepeatType: Query)
... Text to be repeated ...
##EndRepeat.n
```

The looping statement consists of an opening and closing denominator, with the ##Repeat determining the beginning and the ##EndRepeat statement defining the ending of the block of text to be repeated. For every ##Repeat . n statement there *must* be a corresponding ##EndRepeat . n statement.

### Repeat Block Structure

Statement Breakdown	Description
##Repeat	This instruction informs the documentation that the current line denotes the start of the loop.
n	Specifies the nesting level of the repeat statement. This is used to attach substitution links.
Repeat Type	This is the type of loop being performed, the repeat type determines which data source is used to execute the repeat block.
Query	This is the query string used to filter the resulting dataset that is then iterated by the repeat statement.
##EndRepeat	This instruction informs the documentation that the current line denotes the end of the loop.
OrderBy	(Optional)This instruction allows the values returned in a ##Repeat statement based on the provided query to be sorted in ascending or descending order. See <b>Types datasets</b> for available fields. The sort order can be controlled by specifying <b>asc</b> for ascending order or <b>desc</b> for descending order.

### Repeat Types

Repeat Type	Data Source	Description
Object	Typed Dataset	All object instances matching the query
SubObject	Typed Dataset	All sub object instances matching the query, of the object instance currently iterating
SubObjectAll	Typed Dataset	All sub object instances matching the query
ObjectReferences	Typed Dataset	All objects referencing the instance specified in the query
Collection	Collection Dataset	All collection items of the currently iterating object or sub-object matching the query
CollectionAll	Collection Dataset	All collection items of all objects matching the query
SQL	Raw SQL data set	All items (rows) returned by the query.

**SubObject** and **Collection** repeat types are only valid when directly nested under other specific repeat types.

#### 1. SubObject

Must be directly nested under an Object repeat type



## 2. Collection

Must be directly nested under either an **Object**, **SubObject** or **SubObjectAll** repeat type.

When using the above two repeat types, care must be taken to *ensure* that the nesting levels follow numerically i.e. the statements are *directly nested*.

### Object, SubObject and SubObjectAll Queries

When querying a typed dataset, the following tables list the available operators that may be used to build the query statement.

#### Binary computational operators

Operator	Description
*	Multiply
/	Divide
% mod	Modulo
+	Add
-	Subtract
&	String Concatenate

#### Unary logical operators

Operator	Description
! - Not	Logical Not

#### Binary logical operators

Operator	Description
=	Equals
==	
!=	Not Equals
<>	Not Equals
>	Greater Than
>=	Greater Than And Equal To
<	Less Than
<=	Less Than And Equal To
&& and	Logical And
 or	Logical Or

#### Ternary logical operators

Operator	Description
?:	Logical If

#### Object Reference Queries

Object reference queries are designed to filter objects and subobjects referencing each other. This is achieved in ACM by declaring a parameter or sub-parameter (parameter on a subobject) as a Reference and then referencing the objects to each other.

The syntax for this query type is as follows:

```
##Repeat.n(ObjectReferences: ReferenceType = "id")
```

### Object References query reference types

Reference Type	ID
ObjectRef	Object ID. The query will return all objects referencing this ID.
SubObjectRef	SubObjectID. The query will return all parent objects of subobjects referencing this ID.

### Collection Dataset Queries

There is only one possible query on collection datasets:

```
Name = "CollectionName"
```

### Raw SQL Queries

There is no predefined syntax for raw SQL queries. When executing these repeat types the documentation engine passes the SQL query straight through the project database. Any valid SQL statement may be entered.

**Important:** SQL statements must be entered on a single line within the template i.e. no line-breaks may be present within the SQL statement.

### See also

[Supported instructions](#) on [page 87](#)

## Break instruction

The break statement terminates the closest enclosing loop in which it appears. Control is passed to the statement that follows the terminated statement, if any.

The syntax for the break statement is as follows:

```
##Break
```

### Example:

The break statement is useful when filtering loops with a conditional statement. The example below breaks the inner most loop once the filter-expression is matched.

```
##Repeat.1(Objects: query-expression)      <==
##If (filter-expression) ##Then           |
text here...                               |
##Break                                   ===
##EndIf
##EndRepeat.1
```

### See also

[Supported instructions](#) on [page 87](#)

## Conditional instruction

Conditional statements in the template language allow blocks of text to be included based on a logical statement. This conditional inclusion of text is

implemented using the if-else-end pattern found in most programming languages.

The conditional statement is defined as follows:

```

##If statement ##Then
... Text to be included if statement is true ...
##ElseIf statement ##Then
... Text to be included if statement is true ...
##Else
... Text to be included if statement is false ...
##EndIf

```

The `##ElseIf` and `##Else` portions of the conditional statement is optional and may be omitted.

## See also

[Supported instructions](#) on [page 87](#)

## Substitution instruction

The substitution functionality in the template language is used to replace text in the template with values from the attached data source. This allows the template to be replaced with actual application data for each specific record.

The substitution statement has the following format:

```
[n. {FieldName} ]
```

### Breakdown of [n.{Substitution}] syntax

Statement Breakdown	Description
n	The data source number.
FieldName	Any valid field name for the associated data source.

### Important:

#### Template language substitutions

In addition to being replaced within the body of a template, substitutions are also valid in the template language itself:

- Looping queries
- Conditional statements

#### Project parameters

With ACM the project parameters are global to all objects in a specific project. In order to simplify the substitution syntax, project parameters may be accessed directly within the template language.

The project parameter syntax is defined as follows:

```
[p. {AnyValidProjectParameterName} ]
```

## See also

[Supported instructions](#) on [page 87](#)

## Define instruction

The define instruction is used to define a new variable to be used throughout a document template. The variable is not bound by any Repeat loops.

The define instruction is typically used in conjunction with the ##Set instruction and is useful when a value from an outer Repeat loop needs to be passed to another outer Repeat loop.

The define statement has the following format:

```
##Define(variable, datatype, value)
```

The syntax for using the value of the variable in a template has the following format:

```
[d.{variable}]
```

Example:

```
##Define(myObjName, string, "")
##Define(myObjNameLength, int, 0)
```

## See also

[Supported instructions](#) on [page 87](#)

## Set instruction

The set instruction allows for a variable that is declared in the ##Define instruction to be updated with a new value. The set instruction accepts two parameters: 'variable' which is the name of the variable to be updated and 'value' which refers to the new variable value or reference.

The set statement has the following format:

```
##Set(variable, value)
```

The syntax for using the value of the variable in a template has the following format:

```
[d.{variable}]
```

Example:

```
##Set(myObjName, "[1.{ObjectName}]")
##Set(myObjNameLength, [1.{ObjectName.Length}])
```

A number of standard Microsoft functions can be used with the instruction when assigning or manipulating the value and are listed in the following table.

Function	Example
Round	##Set(myRound, Round(87.6))
Format	##Set(myFormat, Format(87.6, "###0.00"))
Modulus	##Set(myMod, Modulus(5,4))
ABS	##Set(myABS, ABS(-87.6))
INT	##Set(myINT, INT(87.6))

Function	Example
Length	<code>##Set(myLength, Length("[1.{ObjectDescription}"]))</code>
Ucase	<code>##Set(myUpperCase, Ucase("[1.{ObjectDescription}"]))</code>
Lcase	<code>##Set(myLowerCase, Lcase("[1.{ObjectDescription}"]))</code>
Truncate	<code>##Set(myTruncate, Truncate([1.{MaxValue}]))</code>
Sin	<code>##Set(mySin, Sin(87.6))</code>
Cos	<code>##Set(myCos, Cos(87.6))</code>
Tan	<code>##Set(myTan, Tan(87.6))</code>
now()	<code>##Set(myDateTime, now())</code> <code>##Set(myDateTimeFormatted, Format(now(), "dd MMM yyyy"))</code>
Today()	<code>##Set(myDate, Today())</code>
Rnd()	<code>##Set(myRandomize, Rnd())</code>
Trim	<code>##Set(myTrim, Trim("[1.{ObjectDescription}"]))</code>
IndexOf	<code>##Set(myIndexOf, IndexOf("[1.{ObjectDescription}]", "word"))</code>
Mid	<code>##Set(myMid1, Mid("[1.{ObjectDescription}]", 12, 22))</code>
Right	<code>##Set(myRight, Right("[1.{ObjectDescription}]", 12))</code>
Left	<code>##Set(myLeft, Left("[1.{ObjectDescription}]", 22))</code>
Substring	<code>##Set(mySubString1, substring("[1.{ObjectDescription}]", 12, 22))</code>
Pow	<code>##Set(myPower, Pow(2, 0) + Pow(2, 1) + Pow(2, 2) + Pow(2, 3) + Pow(2, 4))</code>

## See also

[Supported instructions on page 87](#)

## Comment instruction

Comments may be added to templates with the following syntax:

```
##! I am a comment
```

It is important to note that all comments are removed from the template during generation.

## See also

[Supported instructions on page 87](#)

## Log instruction

Logging may be added to templates with the following syntax:

```
##Log I am log message
```

It is important to note that all logging is removed from the template during generation. All log messages are written to the ACM log with level 'Debug'.

## See also

[Supported instructions on page 87](#)

## Types datasets

There are two typed datasets available to the documentation engine, one for Object repeat types and one for SubObject repeat types shows the available fields in the Object typed dataset. SubObject typed datasets contain all the fields shown in the SubObject fields table, with additional SubObject specific fields.

### Object fields

Field	Data Type	Original Database Table
ObjectID	Int	Objects
ProjectID	Int	Objects
ControllerID	Int	Objects
ParentObjectID	Int	Objects
ObjectName	String	Objects
ObjectDescription	String	Objects
LibraryID	String	RegisteredLibraries
CatalogNumber	String	RegisteredLibraries
Family	String	RegisteredLibraries
LibraryType	String	RegisteredLibraries
CoreType	String	RegisteredLibraries
Category	String	RegisteredLibraries
LibraryDescription	String	RegisteredLibraries
MajorRev	Int	RegisteredLibraries
MinorRev	Int	RegisteredLibraries
Owner	String	RegisteredLibraries

### SubObject fields

Field	Data Type	Original Database Table
SubObjectID	Int	SubObjects
SubObjectName	String	SubObjects
SubObjectDescription	String	SubObjects
SubObjectType	String	SubObjects

### See also

[Raw SQL data](#) on [page 94](#)

## Raw SQL data

Raw SQL datasets are dynamic and the fields available are dependent on the columns returned by the specified SQL query. As such this section will briefly define the important ACM database tables and outline how the data source filtering, mentioned previously, can be obtained using raw SQL statements.

The core ACM database layout, consists of four main tables: Objects, Parameters, SubObjects and SubParameters. Although not holding core project information, the RegisteredLibraries table provides a way to fine tune the object filtering based on CoreType (Solution), Family, LibraryType, etc.

### Reference Parameters

Objects within an ACM project can reference each other, normally either by an Object or SubObject reference. This relationship is denoted in the ACM database by the "RefType" column in both the Parameters and SubParameters table. This column contains an enumeration denoting what the data in the Value column of these tables represents. A full list of this enumeration is given in the table below.

### ACM reference types enumeration

Reference Type	Name
1	Immediate
2	ObjectRef
3	SubObjectRef

4	ParentObjectRef (Not Used)
5	ObjectNIBRef (Object External Reference)
6	Calculated
7	SubObjectNIBRef (SubObject External Reference)
8	InterfaceRef
9	InterfaceMemberRef
10	ParameterRef
11	SubParameterRef

### Dataset filtering/scoping

In order to provide dataset scoping on raw SQL queries, the documentation engine will create four temporary tables at the beginning of each generation session.

The four temporary tables are listed as follows:

1. #Objects
2. #Parameters
3. #SubObjects
4. #SubParameters

The following example shows the use of these temporary tables within an ACM documentation template. Consider the scenario to list the names of all the objects in system.

The first example will return all the Objects in the **database** regardless of the level from which the generation was executed within the ACM.

```
##Repeat.1(SQL: SELECT * FROM Objects)
[1.{Name}]
##EndRepeat.1
```

The second example which uses the temporary table #Objects, will only return the objects from the level which the document was generated (i.e. Project objects or Controller Objects).

```
##Repeat.1(SQL: SELECT * FROM #Objects)
[1.{Name}]
##EndRepeat.1
```

### See also

[Types datasets](#) on [page 93](#)

## Technical view of looping statements

Perform a technical review on the following methods after code editing.

### Object

```
##Repeat.1(Object: Query)
... Text to be repeated ...
##EndRepeat.1
```

```

for each object where Query
{
    ... Text to be repeated ...
}

```

### SubObject

```

##Repeat.1(Object: ...)
##Repeat.2(SubObject: Query)
... Text to be repeated ...
##EndRepeat.2
##EndRepeat.1

```

```

for each object where ...
{
    for each SubObject where Query And ObjectId =
    [1.{ObjectId}]
    {
        ... Text to be repeated ...
    }
}

```

### SubObjectAll

```

##Repeat.1(SubObjectAll: Query)
    ... Text to be repeated ...
##EndRepeat.1

```

```

for each SubObject where Query
{
    ... Text to be repeated ...
}

```

### Collection

```

##Repeat.1(Object/SubObject/SubObjectAll: ...)
##Repeat.2(Collection: Query)
... Text to be repeated ...
##EndRepeat.2
##EndRepeat.1

```



```

for each object/sub-object where ...
{
    for each item in collection And ObjectId =
[1.{ObjectId}]/SubObjectId =
                                [1.{SubObjectId}]
    {
        ... Text to be repeated ...
    }
}

```

### CollectionAll

```

for each Object where "has collection"
{
    for each item in collection
    {
        ... Text to be repeated ...
    }
}

```

### See also

[Repeat language](#) on [page 87](#)

## Getting to know the ACM database structure

A working knowledge of the structure of the ACM database will assist when writing document generation scripts for data extraction. There are several tables and views which will be useful for generating documentation.

In addition to the SQL tables there are also several views available. A view is a virtual table which can be queried in the same way as a table. Views can provide advantages over tables in the following ways:

- Views can represent a subset of data contained within a table.
- Views can join and simplify multiple tables into a single virtual table.

Important tables in use in the ACM database are as follows:

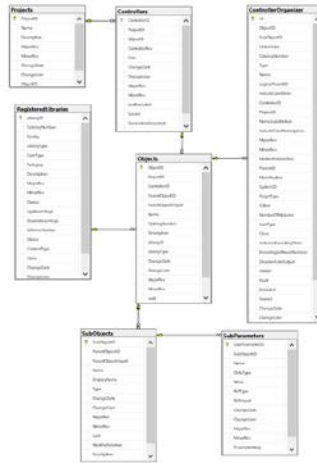


Table descriptions are as follows:

- **Projects:** Contains ACM project information. Columns available in this table are displayed below:

```

    dbo.Projects
    Columns
    ProjectID (PK, int, not null)
    Name (nvarchar(250), not null)
    Description (nvarchar(max), null)
    MajorRev (int, null)
    MinorRev (int, null)
    ChangeDate (datetime, null)
    ChangeUser (nvarchar(100), null)
    ObjectID (int, not null)
    
```

- **Controllers:** All controllers in use in projects in the ACM database will be recorded in this table. Columns available are displayed below:

```

    dbo.Controllers
    Columns
    ControllerID (PK, int, not null)
    ProjectID (FK, int, not null)
    ObjectID (int, not null)
    ControllerRev (int, null)
    Size (int, not null)
    ChangeDate (datetime, null)
    ChangeUser (nvarchar(100), null)
    MajorRev (int, null)
    MinorRev (int, null)
    LastExecuted (datetime, null)
    Source (nvarchar(max), null)
    GenerationSnapshot (varbinary(max), null)
    LastGeneratedTimestamp (datetime, null)
    
```

- **Objects:** All objects used in ACM projects will be recorded in this table. Columns available are displayed below:

dbo.Objects	
Columns	
PK	ObjectID (PK, int, not null)
	ProjectID (int, not null)
FK	ControllerID (FK, int, null)
	ParentObjectID (int, not null)
	ParentObjectImport (nvarchar(250), null)
	Name (nvarchar(250), not null)
	CatalogNumber (nvarchar(250), null)
	Description (nvarchar(max), null)
FK	LibraryID (FK, nvarchar(50), null)
	LibraryType (nvarchar(50), null)
	ChangeDate (datetime, null)
	ChangeUser (nvarchar(100), null)
	MajorRev (int, null)
	MinorRev (int, null)
	Lock (nvarchar(50), null)
	MarkForDeletion (bit, null)

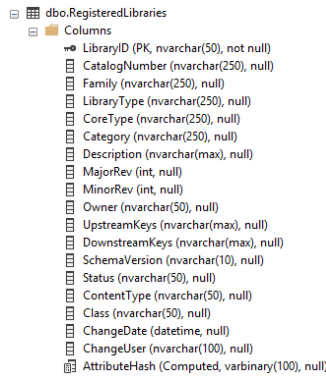
- **SubObjects:** All SubObjects linked to parent objects in use in ACM projects in the ACM database will be stored in this table. Columns available are displayed below:

dbo.SubObjects	
Columns	
PK	SubObjectID (PK, int, not null)
FK	ParentObjectID (FK, int, not null)
	ParentObjectImport (nvarchar(250), null)
	Name (nvarchar(250), not null)
	DisplayName (nvarchar(250), null)
	Type (nvarchar(100), null)
	ChangeDate (datetime, null)
	ChangeUser (nvarchar(100), null)
	MajorRev (int, null)
	MinorRev (int, null)
	Lock (nvarchar(50), null)
	MarkForDeletion (bit, null)
	Description (nvarchar(max), null)
	OverrideDesc (bit, null)

- **SubParameters:** All parameters created for SubObjects used in ACM projects will be recorded in this table. Columns available are displayed below:

dbo.SubParameters	
Columns	
PK	SubParameterID (PK, int, not null)
FK	SubObjectID (FK, int, not null)
	Name (nvarchar(250), not null)
	DataType (nvarchar(50), null)
	Value (nvarchar(max), not null)
FK	RefType (FK, int, null)
	RefImport (nvarchar(50), null)
	ChangeDate (datetime, null)
	ChangeUser (nvarchar(100), null)
	MajorRev (int, null)
	MinorRev (int, null)
	ParameterHelp (nvarchar(max), null)
	RefValue (nvarchar(max), null)

- **RegisterdLibraries:** All libraries registered in the ACM database will be recorded in this table. Columns available are displayed below:

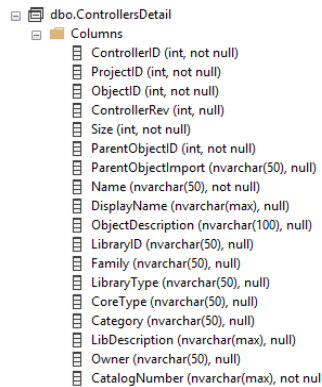


**View Descriptions are as follows:**

As mentioned earlier, several views are available which contain consolidated information that may simplify SQL query construction in that they make information from different tables available from a single source. Search parameters will need to be provided to ensure that only data for the loaded ACM project is retrieved.

Views available are as follows:

- **ControllerDetail:** Provides information about each controller including the controller name, library catalog number, description and other library information. Columns available in this view are displayed below:



- **ObjectsDetail:** Provides information about each object in use in projects within the ACM database. Useful information provided includes the object name, object description and library information including the current revision in use. Columns available in the view are displayed below:

```

dbo.ObjectsDetail
├── Columns
│   ├── ObjectID (int, not null)
│   ├── ProjectID (int, not null)
│   ├── ControllerID (int, null)
│   ├── ParentObjectID (int, not null)
│   ├── ObjectName (nvarchar(250), not null)
│   ├── ObjectDescription (nvarchar(max), null)
│   ├── LibraryID (nvarchar(50), null)
│   ├── Family (nvarchar(250), null)
│   ├── LibraryType (nvarchar(250), null)
│   ├── CoreType (nvarchar(250), null)
│   ├── Category (nvarchar(250), null)
│   ├── LibraryDescription (nvarchar(max), null)
│   ├── MajorRev (int, null)
│   ├── MinorRev (int, null)
│   ├── Owner (nvarchar(50), null)
│   └── CatalogNumber (nvarchar(250), null)

```

- **ProjectsDetail:** Provides information about projects available within the ACM database. Useful information includes the project name and library information for each project type library. Columns available in the view are displayed below:

```

dbo.ProjectsDetail
├── Columns
│   ├── ProjectID (int, not null)
│   ├── ObjectID (int, not null)
│   ├── ControllerID (int, null)
│   ├── ParentObjectID (int, not null)
│   ├── ObjectName (nvarchar(50), not null)
│   ├── ObjectDescription (nvarchar(100), null)
│   ├── Family (nvarchar(50), null)
│   ├── LibraryType (nvarchar(50), null)
│   ├── CoreType (nvarchar(50), null)
│   ├── Category (nvarchar(50), null)
│   ├── LibraryDescription (nvarchar(max), null)
│   ├── MajorRev (int, null)
│   ├── MinorRev (int, null)
│   ├── Owner (nvarchar(50), null)
│   └── CatalogNumber (nvarchar(max), not null)

```

- **SubObjectDetail:** Provides information about sub objects linked to objects within the ACM database. Useful information includes the name for the object to which the sub object belongs, the descriptor for the object to which the sub object belongs, sub object name, sub object description and library information for the object to which the sub object belongs. The columns available in the view are displayed below:

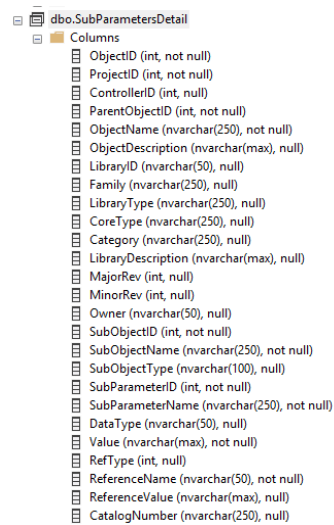
```

dbo.SubObjectsDetail
├── Columns
│   ├── ObjectID (int, not null)
│   ├── ProjectID (int, not null)
│   ├── ControllerID (int, null)
│   ├── ParentObjectID (int, not null)
│   ├── ObjectName (nvarchar(250), not null)
│   ├── ObjectDescription (nvarchar(max), null)
│   ├── LibraryID (nvarchar(50), null)
│   ├── Family (nvarchar(250), null)
│   ├── LibraryType (nvarchar(250), null)
│   ├── CoreType (nvarchar(250), null)
│   ├── Category (nvarchar(250), null)
│   ├── LibraryDescription (nvarchar(max), null)
│   ├── MajorRev (int, null)
│   ├── MinorRev (int, null)
│   ├── Owner (nvarchar(50), null)
│   ├── SubObjectID (int, not null)
│   ├── SubObjectName (nvarchar(250), not null)
│   ├── SubObjectDescription (nvarchar(max), null)
│   ├── SubObjectType (nvarchar(100), null)
│   └── CatalogNumber (nvarchar(250), null)

```

- **SubParameterDetail:** Provides information about sub parameters belonging to sub objects. These sub objects in turn would belong to a specific object. Useful information provided includes the sub parameter name, datatype, reference type, name of the sub object and

library information for the object to which the sub parameters parent (sub object) belongs. The columns available in the view are displayed below:



### Practical Scenario Overview:

The number of instances created in an ACM project for the library with CatalogNumber MsDinSiS are needed for a project management meeting. This information can be easily obtained from the ACM database via the use of a number of SQL queries. The first step is to get the Project Id of the relevant project in ACM. We know the name of the project and can get the Id from the Projects table via a SQL statement. The first step is to open either a query window within Microsoft SQL Server Management Studio, a new command line session for Microsoft sqlcmd or via the use of any other tool that provides access to the ACM database for querying purposes.

The following SQL query will retrieve the ProjectID for the known project name "RAMSProjectSmall":

```
SELECT ProjectID, Name
FROM [ACM_V400].[dbo].[Projects] WHERE Name = 'RAMSProjectSmall'
```

In the above query we are returning only the "ProjectID" and "Name" columns for records that match on the name "RAMSProjectSmall". The following result is returned for the given example:

ProjectID	Name
6	RAMSProjectSmall

From the above screenshot we can see that the ProjectID we are looking for has a value of 6. The next step is to write a query for the Objects table using the ProjectID with a filter value of 6 which will return only records for this project. In addition to the ProjectID we need to return records for only instances of the library with a catalog number of "MsDinSiS". We are also only interested

in the total count of returned records and not the records themselves. In order to accomplish this we will use the SQL COUNT instruction.

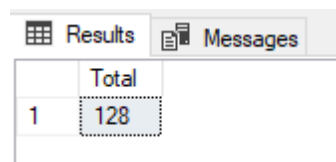
The query to return the required information will look as follows:

```
SELECT COUNT(*) AS Total FROM [ACM].[dbo].[Objects] WHERE ProjectID = 6 AND CatalogNumber = 'MsDinSiS'
```

Breakdown and Explanation:

- COUNT(\*) AS Total: Ensures that the total number of matched records rather than the records themselves is returned. The returned count column is given the name of "Total".
- FROM [ACM].[dbo].[Objects]: References the Objects table within the ACM database.
- WHERE ProjectID = 6 AND CatalogNumber = 'MsDinSiS': This section of the SQL statement provides the filtering required in order to ensure that only relevant records are returned. Firstly, only records relating to the project with an ID of 6 are returned (ProjectID = 6) and in addition to this, only objects that match on the name 'MsDinSiS' in the CatalogNumber column will be returned.

A total count for the number of matches will be returned in the result. An example is provided for a specific project where 128 matches are found:



	Total
1	128

## See also

[Types datasets](#) on [page 93](#)

## Working with SQL Queries in Document Generation Scripts

A valid SQL query will need to be constructed and verified before it can be incorporated into a document generation script via the use of a SQL keyword. A basic working knowledge of SQL would be required in order to accomplish this task. Queries would need to be written and tested via a SQL tool in order to determine that no syntax errors are present and that the correct information is returned. Examples of available tools include Microsoft SQL Server Management Studio and the Microsoft sqlcmd utility. More information about both utilities can be obtained via the Microsoft website.

A dataset returned via a SQL statement would need to be processed within a Document Generation Template via a looping statement in the form of a `##Repeat` statement. This statement allows for the Document Generation Engine to process each record within a dataset for output.

### Example Query 1

In this example we are going to return a list of all objects within the ACM database. The SQL statement that we will be using is as follows:

```
SELECT * FROM Objects.
```

Running this query when connected to a SQL database via SQL Management Studio or `sqlcmd` will return a list of all data contained within the `Objects` table within the ACM database.

---

#### IMPORTANT

In order to provide dataset scoping on raw SQL queries, the documentation engine will create four temporary tables at the beginning of each generation session. The four temporary tables are listed as follows:

1. `#Objects`
2. `#Parameters`
3. `#SubObjects`
4. `#SubParameters`

Each table will contain information pertaining to the currently open ACM project. It is possible to retrieve any information contained in the ACM database though by specifying the specific table or view. As an example, we could retrieve all data from the `Objects` table which does not have the preceding `#` in the name as with the temporary table name (`#Objects`).

---

We will be using a document generation `##Repeat` instruction to print each record contained within the SQL result set which is returned from the database. The statement we will be inserting into the Document Template Editor is as follows:

```
##Repeat.1(SQL: SELECT * FROM Objects)
[1.{Name}]
##EndRepeat.1
```

Once the SQL query completes execution, we will have a dataset containing matched data from the database. This matched data object can be thought of as a temporary table of information existing in memory. The `##Repeat` instruction will be used to iterate through this temporary table and in the above example we have elected to return data from the column named "Name". This column must exist within the returned dataset. Any column can be selected for returned data and the original ACM `Objects` table can be reviewed to check on what columns may contain information that may be useful for the task at hand. As an example: information on each object may be required and the required columns that are need are known ("Name" and "Description" columns in the `Objects` table). All columns are specified to be returned via the SQL statement by using the `"*"` wildcard character: (`SELECT * FROM Objects`). The following document generation script will enable us to return the information as required:



```

##Repeat.1(SQL: SELECT * FROM Objects)
[1.{Name}], [1.{Description}]
##EndRepeat.1

```

The highlighted section above shows the columns that will be returned ("Name" and "Description" fields). Running this script within the Document Template Editor would return a list of all information contained within the Objects table of the ACM database. Below is an example of output from the above script (returned information will vary depending on how the objects are created within the ACM database):

MyProject, Project Information

My\_Controller, ControlLogix Controller

Controller\_Fault\_Handler, Task Description

PowerUp\_Handler, Task Description

Unscheduled, Task Description

FTAlarmEvent\_Server, FactoryTalk Alarms and Events

FTViewSE\_Server, FactoryTalk View SE Display

FTHistorianSE\_Server, FactoryTalk Historian SE Scan Classes

Looking at the information returned above, we may decide that we only want to return objects related to the currently opened ACM project. Earlier in this section we spoke of the four temporary tables created behind the scenes when running a document generation script. One of these tables (#Objects) contains information pertaining to objects for the current ACM project. Switching the SQL script to use the temporary objects table can be achieved by changing the table name to the temporary table name:

```
SQL: SELECT * FROM #Objects
```

The complete script would now look as follows:

```

##Repeat.1(SQL: SELECT * FROM #Objects)
[1.{Name}], [1.{Description}]
##EndRepeat.1

```

#### Example Query 2

In this example we are going to be querying information from the temporary objects table which will provide information relating to the current ACM project only. We are going to retrieve information on objects relating to a specific library. Once again, we are going to be selecting the "Name" and "Description" fields but this time the query will be optimized and will only return these fields to the dataset. We will replace the "\*" wildcard used in the previous query with the actual fields we want to return.

Search parameters will be to a specific library so that we return objects for this library only. The WHERE part of the SQL statement will accomplish this and provide the filter. The catalog number field within the #Objects temporary table will be used where we will be returning records that match the value as specified. In the test example only record matches will be returned where the CatalogNumber column equals the value 'MsVlv2sS'.

The statement we will be inserting into the Document Template Editor is as follows:

```
##Repeat.1(SQL: SELECT Name, Description FROM #Objects WHERE
CatalogNumber='MsVlv2sS')
[1.{Name}], [1.{Description}]
##EndRepeat.1
```

As can be seen via the highlighted section, we have replaced the "\*" wildcard used in example 1 with the actual field names that are required from the dataset. This will improve efficiency and will be seen in the speed of execution of the query with only the required columns returned.

Once again, each record that is returned in the result-set will be processed by the ##Repeat instruction and we will be returning both available columns, namely "Name" and "Description".

Below is an example of some returned information (returned information will vary depending on how the objects are created within the ACM database):

```
XV_0001, Valve Two State 1
XV_0002, Valve Two State 2
XV_0003, Valve Two State 3
XV_0004, Valve Two State 4
XV_0005, Valve Two State 5
```

### Advanced Query

The following sample will return all libraries in use for a specific library type.

The SQL statement that we will be using is as follows:

```
SELECT COUNT(libs.CatalogNumber) AS Qty, libs.CatalogNumber,
libs.Description
FROM #Objects AS objs INNER JOIN RegisteredLibraries AS libs
ON objs.LibraryID = libs.LibraryID
WHERE libs.LibraryType = 'ControlModule'
GROUP BY libs.CatalogNumber, libs.Description
ORDER BY libs.CatalogNumber
```

The above query selects information across two tables (#Objects and RegisteredLibraries) and returns all libraries of the type 'ControlModule' in use within the currently loaded ACM project as we are querying information from the temporary #Objects table as discussed earlier.

Note that the query above is making use of the temporary #Objects table. The "#" will need to be removed when testing this query with a SQL tool like SQL Management Studio or sqlcmd as the temporary Objects table is only created by the Documentation Generation Engine when a documentation generation script is run within ACM. An error will be generated if the "#" is not removed as the table does not exist within the ACM database.

The complete document generation script would look as follows:

```
##Repeat.1(SQL: SELECT COUNT(libs.CatalogNumber) AS Qty,
libs.CatalogNumber, libs.Description FROM #Objects AS objs INNER JOIN
RegisteredLibraries AS libs ON objs.LibraryID = libs.LibraryID WHERE
libs.LibraryType = 'ControlModule' GROUP BY libs.CatalogNumber,
libs.Description ORDER BY libs.CatalogNumber)
[1.{Qty}] [1.{CatalogNumber}] [1.{Description}]
##EndRepeat.1
```

Once the data has been retrieved from the database, the result-set is then processed and iterated through via the ##Repeat instruction. Three columns would be created in the output. The first column would display the number of instances in which the selected library is used, followed by the catalog number for the library and finally the libraries description information in the last column.

Below is an example of some returned information (returned information will vary depending on how the objects are created within the ACM database):

112	MsAinSiS	Analog Input
128	MsDinSiS	Digital Input
128	MsVlv2sS	Valve Two State

## See also

[Raw SQL data](#) on [page 94](#)

## Add ACM library content to an existing ACD project

Add content from the ACM library to an existing ACD project by opening the ACD in Application Code Manager.

### To add ACM library content to an ACD project

1. Click **Tools** and click **Open Target ACD**. The **Open** dialog will display.
2. Select the desired ACD file and click **Open**.
3. The **Target ACD File** tab will replace the **Controller Preview** pane.

4. Drag and drop content from the registered libraries or an existing ACM project to the **Target ACD File** tab. The **Target ACD Generation Wizard - Object Configuration** page appears.



Tip: Click the **Override Calculated Parameters** button to unlock the calculated parameter of this object. The unlocked calculated parameter can be edited. After editing, click the **Enforce Calculated Parameters** button to lock this parameter.

5. (optional) Click **Options > Include Project Data** to include all instances with their parameter values, as well as all libraries (zipped) will be included as part of the Controller's Custom Properties.
6. Resolve any values in the **Unresolved Names** page, then click **Next**. The **Target ACD Generation Wizard - Merge Actions** page appears.
7. Review the list of merge actions.
  - If any merge actions are incorrect, click **Back** to update the parameters as needed.
  - If the merge actions are correct, click **Next**.

The **Target ACD Generation Wizard - L5X Generation Successful** page appears.

8. Click **Finish**. The **Save As** dialog box opens. In **File name** either:
  - Type a new name for the ACD file that contains the merged content.
  - Leave the original name to overwrite the existing ACD project (default).
9. Click **Save** to save the target ACD file.

## See also

[Update a project library](#) on [page 37](#)

## Reports

### Reports

Application Code Manager includes a variety of reports to help you track the usage of code objects in your projects.

This table lists the reports available:

Report	Description
Project History	For the current project, shows the major revision, minor revision, the user account that made the revision, the date and time that the revision occurred, the comment associated with the revision, and the status of the project at the revision point.
All Library Usage (Current Project)	For the current project, shows the name of the solution, the type of library, the category of the library object, the catalog number of the library object, the major revision of the library, the minor revision of the library, the number of usages of the library object in the project, and the status of the library.
Library Usage per Solution (Current Project)	For the selected library in the current project, shows the type of library, the category of the library object, the catalog number of the library object, the major revision of the library, the minor revision of the library, the number of usages of the library object in the solution, and the status of the library.
Library Usage per Library Type (Current Project)	For the selected library type in the current library of the current project, shows the category of the library object, the catalog number of the library object, the major revision of the library, the minor revision of the library, the number of usages of the library object in the solution, and the status of the library.
Library Usage per Category (Current Project)	For the selected category in the selected library type of the current library associated with the current project, shows the catalog number of the library object, the major revision of the library, the minor revision of the library, the number of usages of the library object in the solution, and the status of the library.
Library Usage for Selected Library (Current Project)	For the selected library object within a selected category in the selected library type of the current library associated with the current project, shows the total number of times the library object is referenced and the number of times the object is used by each controller in the current project.
Chassis Layout	For the chassis of the selected controller object, shows the slot assignments, module names, module type, and module information.
I/O Schedule	For the selected I/O module of the current controller object, shows the I/O points, type, module information, and description.
Network Layout	For the selected communication module of the current controller object, shows the IP Address assigned to the module the module name, the module type, and connections information.
Library Usage per Project	For the connected ACM database, shows the solution, the library type, the library category, the catalog number, the major revision, the minor revision and the number of times used in projects.
Pending Libraries in database	For the connected ACM database, shows the libraries that have not yet been published. Identifying them by solution, library type, category, catalog number, major revision and minor revision.
Library Usage per Solution in database	For the selected solution in the connected ACM database, shows the library type, the library category, the catalog number, the major revision, the minor revision and the number of times used in projects.
Pending Libraries per Solution in database	For the selected solution in the connected ACM database, shows the libraries that have not yet been published. Identifying them by library type, category, catalog number, major revision and minor revision.
Library Usage per Library Type in database	For the selected library type of a solution in the connected ACM database, shows the library category, the catalog number, the major revision, the minor revision and the number of times used in projects.

Report	Description
Pending Libraries per Library Type in database	For the selected library type of a solution in the connected ACM database, shows the libraries that have not yet been published. Identifying them by category, catalog number, major revision and minor revision.
Library Usage for Selected Library in database	For the selected library object within a library type, shows the total number of times the object is used and the usages per project.

**See also**

[Report command reference](#) on [page 110](#)

[Generate a report](#) on [page 111](#)

[View registered library usage](#) on [page 43](#)

**Report command reference**

**Reports** are generated by selecting the report command from the **View** context menu. Different reports are available depending on the object selected.

Use this table to locate each report. Items enclosed in brackets [] are replaced by the name of the item in your ACM database.

Report	Command
Project History	System View > [Project] > View > Project History
Project Library Usage	System View > [Project] > Used Libraries > View > Project Library Usage Count
Project Library Usage per Solution	System View > [Project] > Used Libraries > [Library folder] > View > Project Library Usage per Solution
Project Library Usage per Solution, Library Type	System View > [Project] > Used Libraries > [Library folder] > [Library type folder] > View > Project Library Usage per Library Type
Project Library Usage per Solution, Library Type, Category	System View > [Project] > Used Libraries > [Library folder] > [Library type folder] > [Library category folder] > View > Project Library Usage per Category
Project Library Usage per Library	System View > [Project] > Used Libraries > [Library folder] > [Library type folder] > [Library category folder] > [Library object] > View > Project Library Usage per Library
Chassis Layout	Controller Preview > [Controller object] > I/O Configuration > View > Chassis Layout
I/O Schedule	Controller Preview > [Controller object] > I/O Configuration > Backplane > [I/O Module] > View > Module I/O Schedule for Rack Module
Network Layout	Controller Preview > [Controller object] > I/O Configuration > Backplane > [Communication Module] > View > Network Layout
Library Usage	Registered Libraries > View > Library Usage
Pending Libraries	Registered Libraries > View > Pending Libraries
Database Library Usage per Solution	Registered Libraries > [Solution] > View > Library Usage per Solution

Report	Command
Database Pending Libraries per Solution	Registered Libraries > [Solution] > View > Pending Libraries per Solution
Database Library Usage per Library Type	Registered Libraries > [Solution] > View > Library Usage per Library Type
Database Pending Libraries per Library Type	Registered Libraries > [Solution] > [Library Type] > View > Pending Libraries per Library Type
Database Library Usage per Library	Registered Libraries > [Solution] > [Library Type] > [Library Object] > Library Usage per Library Type

## See also

[Reports](#) on [page 109](#)

[Generate a report](#) on [page 111](#)

## Generate a report

**Reports** are generated from the **View** context menu. Different reports are available depending on the object selected.

### To generate a report

1. Select the object in the tree that you want to report on. Reports are available from objects in the System View, Controller Preview, Class View, and Registered Libraries panes.
2. Right-click the object, select **View** and then choose the report to run.
3. The report appears in a new window.
  - If there are multiple pages in the report, use the navigation controls to move forwards and backwards through the report information.
  - To rerun the report to incorporate changed information, click **Refresh**.
  - To send the report to a printer, click **Print**.

The **Print** dialog box opens. Confirm the printer in **Select Printer** is correct. Optionally, specify the **Page Range** and **Number of copies** to print. By default one copy of all pages in the report are printed. Click **Print** to print the report.

- To save the report to a file, click **Export** and then choose **Excel**, **PDF**, or **Word**.

The **Save As** dialog box opens. In **File name**, type a name for the report being exported. **Save as type** is already selected for the appropriate file format (.pdf, .xlsx, or .docx).

Click **Save** to save the report

## See also

[Reports](#) on [page 109](#)

[View registered library usage](#) on [page 43](#)

[Report command reference](#) on [page 110](#)



## ACM Console

### ACM Console

The ACM Console is a command-line interface for Application Code Manager that supports scripting and is used to quickly perform operations in the ACM database.

The ACM Console commands can be executed directly from the command line which has a few additional arguments:

- Usage: ACMConsole.exe -p "C:\Script Files\Script.txt"
- Usage: ACMConsole.exe -s to run in silent mode
- Usage: ACMConsole.exe -c to continue on errors
- Usage: ACMConsole.exe -l <FILENAME> to specify an alternative log file

Execute ACMConsole.exe without any arguments to launch interactive console:

- -p Path of the script file
- -s Silent mode
- -c Continue on errors
- -l Alternative log filename
- help Display this help screen

This table lists the commands available in the ACM Console. For more information, such as parameters, data types, and usage information for a command, type `help <nameofcommand>`.

Command	Description
begincreate	Required before using any CREATE command(s).
clearlog	Clears the log memory.
createcontroller	Creates a controller from script file. After calling create command(s) the endcreate command must be called to initiate object creation.
createobjects	Creates objects from script file. After calling create command(s) the endcreate command must be called to initiate object creation.
createproject	Creates a project from script file. After calling create command(s) endcreate must be called to initiate object creation.
createdatabase	Creates an ACM database.
deletedatabase	Deletes the specified ACM database.
deletecontroller	Deletes a specified controller.
deleteproject	Deletes a specified project.
editparameters	Edits the values of an object instance's parameters.
endcreate	Required after using any create command(s), to initiate object creation.
exportallprojects	Exports all ACM projects to Excel files.
exportlibrariesbyattribute	Exports libraries filtered by attribute to HSL4 files.
exportlibrariesbyproject	Exports libraries used in a project to HSL4 files.
exportlibrariesbyquery	Exports libraries filtered by query to HSL4 files.

Command	Description
exportpartial	Exports part of an ACM project to an Excel file.
exportproject	Exports an ACM project to an Excel file.
extractattachmentsbycontroller	Extracts the attachments of a specified controller and its children objects to the output path.
extractattachmentsbyobject	Extracts the attachments of a specified object to the output path.
extractattachmentsbyproject	Extracts the attachments of a specified project and its children objects (including controllers and their children objects) to the output path.
generatealarms	Generates alarm import file (supports FactoryTalk Alarms and Events and FactoryTalk View ME Alarms).
generatecontroller	Generates the specified controller as an L5X or ACD file.
generatedisplays	Generates FactoryTalk View SE or FactoryTalk View ME display import files.
generatehistorian	Generates historian tag import file.
generatepartial	Generates the specified program or routine as an L5X file.
generateprojectcontrollers	Generates all the controllers of a specified project as L5X or ACD files.
help	Displays command help.
importproject	Imports an ACM project from an Excel file.
monitorlog	Begins a monitored log section.
publishbyscript	Extracts and publishes a library from an ACD using a script file.
publishlibrary	Extracts and publishes a library from an ACD.
registerlibrary	Registers the library into the ACM database.
run	Runs a script.
showlog	Shows the log memory content in the log viewer.
switchdatabase	Switches to a different ACM database at any point within a script.
writelogmessage	Writes a message to the log.

### See also

[Open the ACM Console](#) on [page 114](#)

[List all commands](#) on [page 115](#)

[Generate a limited list of commands](#) on [page 115](#)

[Generate detailed command information](#) on [page 115](#)

[Console Scripts](#) on [page 116](#)

## Open the ACM Console

The ACM Console is a separate application from Application Code Manager.

### To open the ACM Console

1. Minimize or close the Application Code Manager if it is open.
2. On the desktop, double-click the ACM Console icon or click **Start > Rockwell Software > ACM Console**.

The ACM Console appears.

### See also

[ACM Console](#) on [page 113](#)

[List all commands](#) on [page 115](#)

## List all commands

The help function in the ACM console can provide a list of all the ACM commands.

### To list all commands

1. Open the ACM Console.
2. At the **\$** prompt, type **help** and then press Enter.

A list of all commands is displayed.

### See also

[Generate a limited list of commands](#) on [page 115](#)

[Generate detailed command information](#) on [page 115](#)

## Generate a limited list of commands

Use the help function in the ACM Console to provide a limited list of commands.

### To generate a limited list of commands

1. Open the ACM Console.
2. At the **\$** prompt, type **help** and the first letters of the command that to limit the list to and then press the **Tab** key.

Example: typing "help g" and then pressing the **Tab** key results in the prompt automatically completing the "g" to generate, pressing tab again returns a comma delimited list of **generatecontroller** and **generatepartial**.

### See also

[Generate detailed command information](#) on [page 115](#)

[ACM Console](#) on [page 113](#)

[Open the ACM Console](#) on [page 114](#)

## Generate detailed command information

Use the help function in the ACM Console to provide detailed information about a specific command.

### To generate detailed command information

1. Open the ACM Console.
2. Type **help** then the full command and then press Enter.

Example: type **help exportproject**, then press **Enter**.

Detailed information on the **exportproject** command displays.

## See also

[ACM Console](#) on [page 113](#)

[Open the ACM Console](#) on [page 114](#)

## Console scripts

A console script is a text file (.txt) containing a set of valid Application Code Manager Console commands. The commands are run in order, top to bottom.

If using the CREATEPROJECT, CREATECONTROLLER, or CREATEOBJECTS commands, the console script requires an Extended script (.xml) file. The Extended script is called as part of the command to provide data to Application Code Manager about the project, controller, or object that is created.

## See also

[Create an Application Code Manager Console script](#) on [page 116](#)

[Run an Application Code Manager Console script](#) on [page 116](#)

[Extended scripts](#) on [page 117](#)

## Create an Application Code Manager Console script

Create an Application Code Manager Console script to store a set of commands for later or repeated use.

### To create an Application Code Manager Console script

1. Open a text editor, such as **Notepad**.
2. Enter one or more valid Application Code Manager Console commands into the script.

---

**IMPORTANT** Only enter one Application Code Manager Console command per line of the script.

---

3. Save the script as a plain text (.txt) file.

## See also

[Run an Application Code Manager Console script](#) on [page 116](#)

[ACM Console](#) on [page 113](#)

## Run an Application Code Manager Console script

Run an Application Code Manager Console script to execute a set of predefined commands on the Application Code Manager database.

### To run an Application Code Manager Console script

1. Double-click the **ACM Console** shortcut on the desktop.
2. Select **File > Run Script**.
3. In **Select an ACM Script File** locate the file, then click **Open**.

The script executes.

Alternatively:

- At the Application Code Manager Console prompt, type `run filelocationpath\scriptfilename.txt`, then press **Enter**. Replace `filelocationpath` with the file path, such as `C:\Users\<username>\Desktop\` and `scriptfilename.txt` with the script filename.

## See also

[ACM Console](#) on [page 113](#)

## Extended scripts

An Extended script is an XML (.xml) file that contains information about a project, controller, or library object. The Extended script is called from a CREATE command in an Application Code Manager Console script, which passes the object data contained in the Extended script to the Application Code Manager database.

Projects and controllers may not be defined in the same XML file. For best results, create three Extended scripts: one for the project, one for the controller, and one for all the associated objects.



Tip: When creating a Project or Controller XML file, set up a reference (@Project or @Controller) to use the project name passed in from the Application Code Manager Console script. This allows a single Project or Controller XML file to be used for multiple projects.

An example of an Application Code Manager Console script that calls an Extended script is:

```
BEGINCREATE
CREATEPROJECT "myProject" "C:\Script
Files\Project.xml"
CREATECONTROLLER "myProject" "myController"
"C:\Script Files\Controller.xml"
CREATEOBJECTS "myProject" "myController" "C:\Script
Files\Objects.xml"
ENDCREATE
```

## See also

[Generate an example Extended script](#) on [page 117](#)

## Generate an example Extended script

Generate an example Extended script to copy and paste the XML structures and parameters to a new Extended script file for editing.

### To generate an example Extended script

1. Create a Project, Controller, and Object in Application Code Manager.

2. Generate an L5X controller file with the **ACM Project Data** option enabled.
3. Open the .L5X controller file in a text editor.
4. Copy the <IObjs> and <ICOObjs> nodes to a new Extended script file.

## See also

[Generate a controller file](#) on [page 48](#)

## Library <IObj>

The Library <IObj> node represents a library instance configuration.

### Parent node:

- <IObjs>

Attribute	Description
Name	The name of the library object instance.
Task	The task name in which the instantiated library content will reside if the library is a task type library. If applicable, Project and Controller objects may be left empty.
Program	The program name in which the instantiated library content will reside if the library is a program type library. If applicable, Project and Controller objects may be left empty.
Description	The description message of the library instance.
Guid	The unique ID of the library in the ACM database. During instantiation, if a Guid is supplied, the database is queried to find a match. <ul style="list-style-type: none"> <li>• If the <b>Guid</b> is supplied then the <b>Sol</b>, <b>Catalog Number</b>, <b>Maj</b> and <b>Min</b> attributes are ignored.</li> <li>• If the <b>Guid</b> is not supplied then <b>Sol</b> and <b>Catalog Number</b> must be supplied.</li> </ul>
Sol	The solution category defined within the library. This is a mandatory field if <b>Guid</b> is not supplied.
Catalog Number	The catalog number as defined within the library. This is a mandatory field if <b>Guid</b> is not supplied.
Maj	The major revision number of the library. This is an optional field. If omitted and <b>Guid</b> is not supplied then the highest major library revision number will be selected.
Min	The minor revision number of the library. This is an optional field. It can only be included if the <b>Maj</b> attribute is present. If it is excluded then the highest minimum library revision number will be selected.

## Parameter <IPar>

The Parameter <IPar> node represents a library parameter.

### Parent node:

- <IPars>

Attribute	Description
Name	The name of the parameter. This is a mandatory field and must be supplied.
Value	The value assigned to the parameter. This is a mandatory field and must be supplied.

Attribute	Description
RefVal	This attribute only applies to reference type parameters. If supplied, this value adds the reference to the parameter value, such as <i>#refvalue</i> . If not supplied, then a reference value is not added to the parameters value.

## SubObject <ISObj>

The SubObject <ISObj> node represents a Sub Object. SubObject parameters must be supplied as <IPar> nodes within the child <IPars> node.

### Parent node:

- <ISObjs>

### Child node(s):

- <IPars>

Attribute	Description
Name	The name of the SubObject, such as <i>Inp_Intlk01</i> .
Type	The type of SubObject, such as <i>Interlock</i> .
Description	The description of the SubObject.
OverrideDesc	If formatting is applied to the SubObject description field in the library and this field is set to true, then the description can be overridden.

## Linked Library <ILLib>

The Linked Library <ILLib> node represents a linked library reference.

### Parent node:

- <ILLibs>

Attribute	Description
Name	The name displayed for the Linked Library.
Value	The instance name of the target library.

## Interface Links <IILink>

The Interface Links <IILink> node represents an interface link for a library instance. An Interface Link requires one <IMems> child nodes and one or more <IMem> grandchild nodes.

### Parent node:

- <IObjs>

### Child node(s):

- <IMems>

Attribute	Description
Name	The name of the interface link.

Attribute	Description
Keying	Can either be specified as <b>ExactMatch</b> or <b>Disabled</b> . <b>ExactMatch:</b> ACM matches input members against their respective <b>Key Id</b> and <b>Revision</b> field values. <b>Disabled:</b> ACM does not match input members against their respective <b>Key Id</b> and <b>Revision</b> field values.
RefInt	A reference to the library instance to which the Interface Link points.

### See also

[Interface Members <IMem>](#) on [page 120](#)

## Interface Members <IMem>

The Interface Member <IMem> node represents the configuration of an Interface Link member.

### Parent node:

- <IMems>

Attribute	Description
Name	The name of the Interface Member.
Value	The library instance name and output interface member to which to connect.
RefVal	The name of the output interface member in the destination library.

### See also

[Interface Links <IILink>](#) on [page 119](#)

## Controller Object <ICOObj>

The Controller Object <ICOObj> node represents an object that appears in the Application Code Manager Controller Preview pane.

Controller Objects are only required when additional object configuration is required for a controller object, such as Instance Execution Ordering.

### Parent node:

- <ICOObjs>

Attribute	Description
Name	The name to be assigned to the controller object.
Obj	The instantiated name of the library object.
COParent	The name of the parent within which the object resides.
Type	The type of object, such as <i>PROGRAM</i> , <i>ROUTINE</i> , or <i>AOI</i> .
NameSub	The substitution name, if applied.



Attribute	Description
Order	Specifies where a routine would be displayed under its parent program node within the ACM Controller Preview panel. For example, a routine with an <b>Order</b> value of 0 is displayed at the top of the tree. A routine with an <b>Order</b> value of 1 is displayed next in the list.
MarkDel	Excludes an AIO for a specific controller.
MainRtn	Specifies the level of the routine. <b>2:</b> Sets the routine as the active routine. <b>1:</b> Sets the routine as an inactive routine. <b>0:</b> Sets the routine as a normal routine.
InstExecOrder	Specified for ROUTINE types, otherwise the value is blank. The value specifies the routine execution order number. This value will determine in what sequence routines are processed when generating a controller from within ACM.



### Legal Notices

Rockwell Automation publishes legal notices, such as privacy policies, license agreements, trademark disclosures, and other terms and conditions on the [Legal Notices](#) page of the Rockwell Automation website.

#### End User License Agreement (EULA)

You can view the Rockwell Automation End User License Agreement (EULA) by opening the license.rtf file located in your product's install folder on your hard drive.

The default location of this file is:

C:\Program Files (x86)\Common Files\Rockwell\license.rtf.

#### Open Source Software Licenses

The software included in this product contains copyrighted software that is licensed under one or more open source licenses.

You can view a full list of all open source software used in this product and their corresponding licenses by opening the index.html file located your product's OPENSOURCE folder on your hard drive.

The default location of this file is:

C:\Program Files\Rockwell Automation\Application Code Manager\Release Notes\OPENSOURCE\index.htm

You may obtain Corresponding Source code for open source packages included in this product from their respective project web site(s). Alternatively, you may obtain complete Corresponding Source code by contacting Rockwell Automation via the **Contact** form on the Rockwell Automation website: <http://www.rockwellautomation.com/global/about-us/contact/contact.page>. Please include "Open Source" as part of the request text.

# Rockwell Automation support

Use these resources to access support information.

<b>Technical Support Center</b>	Find help with how-to videos, FAQs, chat, user forums, and product notification updates.	<a href="http://rok.auto/support">rok.auto/support</a>
<b>Knowledgebase</b>	Access Knowledgebase articles.	<a href="http://rok.auto/knowledgebase">rok.auto/knowledgebase</a>
<b>Local Technical Support Phone Numbers</b>	Locate the telephone number for your country.	<a href="http://rok.auto/phonesupport">rok.auto/phonesupport</a>
<b>Literature Library</b>	Find installation instructions, manuals, brochures, and technical data publications.	<a href="http://rok.auto/literature">rok.auto/literature</a>
<b>Product Compatibility and Download Center (PCDC)</b>	Get help determining how products interact, check features and capabilities, and find associated firmware.	<a href="http://rok.auto/pcdc">rok.auto/pcdc</a>

## Documentation feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at [rok.auto/docfeedback](http://rok.auto/docfeedback).

## Waste Electrical and Electronic Equipment (WEEE)



At the end of life, this equipment should be collected separately from any unsorted municipal waste.





Rockwell Automation maintains current product environmental information on its website at [rok.auto/pec](http://rok.auto/pec).

Allen-Bradley, expanding human possibility, Logix, Rockwell Automation, and Rockwell Software are trademarks of Rockwell Automation, Inc.

EtherNet/IP is a trademark of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomasyon Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenköy, İstanbul, Tel: +90 (216) 5698400 EEE Yönetmeliğine Uygundur

Connect with us.    

**rockwellautomation.com** ————— expanding **human possibility**<sup>™</sup>

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640

ASIA PACIFIC: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846