Logix 5000 Controller Import/ Export Project Components

1756 ControlLogix, 1756 GuardLogix, 1769 CompactLogix, 1769 Compact GuardLogix, 1789 SoftLogix, 5069 CompactLogix

Rockwell Automation

Original Instructions

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.

These 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).

The following icon may appear in the text of this document.

Identifies information that is useful and can help to make a process easier to do or easier to understand.

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Summary of Changes

This manual includes new and updated information. Use these reference tables to locate changed information.

Grammatical and editorial style changes are not included in this summary.

Global changes

None in this release.

New or enhanced features

This table lists updates in this version of the manual related to new or enhanced features.

Change	Topic
Added an overview of the Data exchange process.	Data exchange on page 31
Added guidelines for importing and exporting tags with data exchange files.	Importing and exporting tags with data exchange files on page 35
Added steps for using the Synchronize dialog, a new feature in Logix Designer version 38.	Synchronize a project with a data exchange file on page 35
Added a list of property names used in EPLAN® Electric P8™.	Property names in EPLAN Electric P8 on page 36

Preface

This manual contains import and export specifications for the Logix Designer application components.

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Studio 5000 environment

The Studio 5000 Automation Engineering & Design Environment® combines engineering and design elements into a common environment. The first element is the Studio 5000 Logix Designer® application. The Logix Designer application is the rebranding of RSLogix 5000® software and will continue to be the product to program Logix 5000™ controllers for discrete, process, batch, motion, safety, and drive-based solutions.

The Studio 5000® environment is the foundation for the future of Rockwell Automation® engineering design tools and capabilities. The Studio 5000 environment is the one place for design engineers to develop all elements of their control system.

Additional resources

These documents contain additional information concerning importing and exporting projects and project components.

Table 1. Additional resources

Resource	Description
Logix 5000™ Controllers Import/Export Reference Manual, publication 1756-RM014	Provides detailed reference information and examples for importing and exporting projects and components.
Logix 5000 Controllers Security Programming Manual, publication 1756-PM016	Describes how to configure security for controller projects using the Logix Designer application.

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.

Software and Cloud Services Agreement

Review the Rockwell Automation Software and Cloud Services Agreement here.

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 at this URL:

Studio 5000 Logix Designer Open Source Attribution List

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.

Additional considerations for rungs

Introduction - Rungs

This chapter explains import and export of rungs.

Export considerations for rungs

One rung or a contiguous set of selected rungs may be exported to an L5X file.

The export file may also include any program-scoped tags, controller-scoped tags, Add-On Instructions, user-defined data types, and user-defined string types that are referenced by the rungs. If they exist in the project, the definitions for the referenced tags, Add-On Instructions, and user-defined data types are exported to the L5X file. In the case of rungs exported from an Add-On Instruction routine, if they exist at the time of export, any parameters and local tags referenced are exported.

Import considerations for rungs

When importing rungs, configure how the referenced components are imported during import configuration. By default, referenced components that collide with project components are not imported.

Considerations when importing rungs.

Topic	Consideration
Pending Edits exist	If rungs are imported into a program or equipment phase that contains pending edits, all pending edits in the program are accepted if **Accept Program Edits** is selected during import of the rungs. Similarly, all pending edits in the program are finalized if **Finalize All Edits In Program** is selected during import of the rungs.
Accepted Edits exist	Rungs cannot be imported into a program or equipment phase that contains routines with Accepted Edits or Test Edits. Existing edits must first either be assembled or canceled.
First scan	When importing rungs into an existing program, the S:FS bit is not set during the program's next scan. This applies when importing rungs into an existing equipment phase as well.
Collision handling	If selecting **Overwrite Selected Rungs** in the **Import Rungs** dialog box, the imported rungs overwrite the rungs selected in the project. If **Overwrite Selected Rungs** is cleared, the imported rungs are inserted before the selected rungs in the project if the ladder editor is in Insert mode, they are

Topic	Consideration
	inserted after the selected rungs if the ladder editor is in Append mode.
Tag scope	When exporting rungs from a program or equipment phase and import them into an Add-On Instruction routine, any referenced controller-scoped or program-scoped tags are converted on import. The tag is converted to a local tag unless local scoped is not allowed (for example, a Motion Group tag cannot be a local tag), in which case the tag is converted to an InOut parameter. When exporting rungs from an Add-On Instruction routine and import them into a routine in a program or equipment phase, the referenced parameters and local tags are converted on import. The parameter or local tag are converted to a program-scoped or phase-scoped tag unless it is not allowed (for example, a Motion Group tag cannot be a program-scoped tag), in which case it is converted to a controller-scoped tag.

For considerations for referenced user-defined types, Add-On Instructions, and tags that may be imported with the rungs, see import and export of user-defined types, Add-On Instructions and tags.

Additional considerations for routines

Introduction - Routines

This chapter explains the import and export of routines.

Export considerations for routines

A routine can be exported to an L5X file. Routines of all language types, Function Block Diagram, Sequential Function Chart, Ladder Diagram and Structured Text, may be exported. However, routines may not be exported from an Add-On Instruction container and SoftLogix external routines may not be exported.

The export file may also include any program-scoped tags, controller-scoped tags, Add-On Instructions, user-defined data types, and user-defined string types that are referenced by the routine. If they exist in the project, the definitions for the referenced tags, Add-On Instructions, and user-defined data types are exported to the L5X file.

Import considerations for routines

When importing a routine, configure how the referenced components are imported during import configuration. By default, referenced components that collide with project components are not imported.

Considerations when importing a routine.

Topic	Consideration
Pending Edits exist	If a routine is imported into a program or equipment phase that contains other routines with pending edits, all pending edits in the program are accepted if **Accept Program Edits** is selected during import of the routine. Similarly, all pending edits in the program are finalized if **Finalize All Edits In Program** is selected during import of the routine.
Accepted Edits exist	A routine may not be imported into a program or equipment phase that contains routines with accepted edits or test edits. Existing edits must first either be assembled or canceled.
Routine type	An existing routine may not be overwritten by a routine that is another routine type.
First scan	When importing a routine into an existing program, the S:FS bit is not set during the program's next scan. This applies when importing a routine into an existing equipment phase as well.
SFC routine execution configuration	SFC execution settings are configured on a controller project, not per SFC routine. If exporting an SFC routine and importing it into another project with different SFC

Topic	Consideration
	execution settings, the functionality of the routine could change. For example, a change in the **Last Scan of Active Steps** setting could leave physical outputs in an undesired state.
Equipment Phase state routines	When state routines are created while online with the controller and logic edits are accepted but not tested, the routine will behave as if it was not implemented.

For considerations for referenced user-defined types, Add-On Instructions, and tags that may be imported with the rungs see import and export of user-defined types, Add-On Instructions, and tags.

Additional considerations for programs and equipment phases

Introduction - Programs

This chapter explains import and export of programs. All topics apply to equipment phases as well; any exceptions are noted.

Export considerations for programs

A program can be exported to an L5X file. The exported program includes all of its program tags and routines, which are imported with the program automatically.

The export file may also include any controller-scoped tags, Add-On Instructions, user-defined data types, and user-defined string types referenced by the program. The definitions for the referenced tags, Add-On Instructions, and user-defined data types are exported to the L5X file if they exist in the project.

As with other export types, I/O module data type definitions are not exported.
I/O module data types are created within a project when the associated I/O module is created in the project. On import, program-scoped tags may not be modified. As a result, if a program-scoped tag aliases an I/O module type tag, the I/O module must first exist in the project in order to import the program. To export the program so that it can be imported into a project with another module type, first alias the program-scoped tag to a controller-scoped tag of a non-I/O module type, and then alias the controller-scoped tag to the I/O module. These program-scoped tags can then be created during the import of the program if necessary.

Import considerations

When importing a program, the program-scoped tags and routines are imported as part of the program. The Operation, Final Name, Description, and any other settings of the program-scoped tags and routines cannot be modified; instead, the Operations are based on the Operation selected for the program.

Programs cannot overwrite equipment phases, and vice-versa. Programs and equipment phases must have unique names.

Configure how the referenced components are imported during import configuration. By default, referenced components that collide with project components are not imported.

Considerations when importing a program.

Topic	Consideration
Accepted or Test Edits exist	A program with Accepted Edits or Test Edits may not be overwritten.
Deletes of program-scoped tags and routines during program overwrite	When importing a program to replace an existing program, any tags or routines in the existing program that are not in the new program are deleted during import. However, if online and Import Logic Edits as Pending or Accept Program Edits is selected in the **Online Options** dialog box, then these tags and

Topic	Consideration
	routines cannot be deleted because they are referenced by existing logic until edits are finalized. In this situation, although they were identified during import configuration with an Operation of Delete, these tags and routines are not deleted as part of the import. Delete them in the Logix Designer editor after finalizing edits.
Safety program scheduled location	A Safety program cannot be scheduled in the **Controller Fault Handler** or **Power-Up Handler** folders.
Configuration of Equipment Phase state routines	In the configuration for an equipment phase state routine, when the **Complete State Immediately if not Implemented** option is selected in version 17.00.00 and later of the application, an implemented, but empty (no logic), phase state routine behaves the same as an unimplemented phase state routine. The state immediately completes and execution of the phase continues. The phase then enters the next state in the state machine. 00In version 16.00.00 or later of the application, if an equipment phase enters a state for which a state routine exists, but contains no logic, execution of the phase stops regardless of whether the **Complete State Immediately if not Implemented** option is selected. The routine does complete, but there is no logic to run. If importing a new state routine and, in the **Online Options** dialog box, select: • **Import Logic Edits as Pending**, an empty routine is created in the controller and the pending edits only exist in the offline project. • **Accepts Program Edits**, an empty routine is created in the controller and the logic is placed in a test edits container in the routine. If not actively testing edits, the routine is empty when running. • **Finalize All Edits in Program**, the routine is created with the new logic and is not empty. In the first two cases, if the **Complete State Immediately if not Implemented** option is selected, the empty routine completes immediately and allows phase execution to continue.
First scan	If a new program or equipment phase is created in a controller in Remote Run mode, logic in that program or equipment phase receives a value of 1 for the S:FS system flag (First Scan flag) until the main routine has run once. Any other logic imports (that is overwriting an existing program or equipment phase, or any routine or rung

Topic	Consideration
	imports) does not result in a value of 1 for S:FS system flag.
Pre-scan	Logic imported while online with the controller in Remote Run mode is not pre-scanned before it begins to run.
Program scheduled location while online	An imported program that is configured to overwrite an existing program cannot be scheduled into a location that differs from the existing program while online with the controller in Remote Run mode; the existing scheduled location is used.
Renamed tags	When overwriting an existing program and the imported program is modified such that a program-scoped tag has been renamed, during import, the existing tag is deleted and a tag with the new name is created. All logic references are updated to reference the new tag. Therefore, the online tag values are not preserved and the tag values from the imported tag are downloaded to the controller. To preserve the data values of the renamed tag and minimize the logic changes, rename the program tag in the online project to the new name before importing the modified program.
Importing multiple programs	Controller Fault Handler and Power-Up Handler Disabled: When importing multiple target programs, the Controller Fault Handler and Power-Up Handler are unavailable in the Schedule In list.

For considerations for referenced user-defined types, Add-On Instructions, program tags, and referenced tags that may be imported with a program see, Importing and exporting user-defined types, Add-On Instructions, and tags.

Additional considerations for user-defined types

Introduction - User Defined

This chapter explains import and export of user-defined data types and user-defined string types.

Export considerations for User-Defined Types

A user-defined type (either user-defined data type or user-defined string type) can be exported to an L5X file.

The export file may also include any Add-On Instructions, user-defined data types, and user-defined string types referenced by the exported user-defined type. The definitions for the referenced Add-On Instructions and data types are exported to the L5X file if they exist in the project and if **Include all referenced Add-On Instructions and User-Defined Types** is selected during export.

Also export user-defined data type references when a program, routine, set of rungs, or Add-On Instruction is exported.

Import considerations for user-defined types

When importing a user-defined type, configure how the referenced components are imported during import configuration. By default, referenced components that collide with project components are not imported.

User-defined types cannot overwrite Add-On Instructions. User-defined types and Add-On Instructions must have unique names.

Considerations when importing a user-defined type.

Topic	Consideration
Tag data	Imported tags that reference a user-defined data type in the import file may be affected if the user-defined data type is not imported as well. In this case, the imported tag's data may be converted if the existing data structure is different and tag data may be lost. If an existing user-defined data type is overwritten, project tag data may be converted if the data structure is different and tag data may be lost.
Data type modification while online	A user-defined data type that is referenced in the project may not be overwritten. If the existing user-defined data type is not referenced, it may be overwritten while online.
Final Name change	If the Final Name of a user-defined type reference is modified during import configuration, all logic, tags, Add-On Instructions, and other user-defined types in

Topic	Consideration
	the import that reference the user-defined type are updated to reference the new name. As a result, the edit date of any Add-On Instructions that references the user-defined type is updated.

See *Import and export Add-On Instructions* for considerations for referenced Add-On Instructions that may be imported with the user-defined type.

Import and export Add-On Instructions

Introduction

This chapter explains import and export of Add-On Instructions.

Export considerations for Add-On Instructions

A user-defined type (either user-defined data type or user-defined string type) can be exported to an L5X file.

The export file may also include any Add-On Instructions, user-defined data types, and user-defined string types referenced by the exported user-defined type. The definitions for the referenced Add-On Instructions and data types are exported to the L5X file if they exist in the project and if **Include all referenced Add-On Instructions and User-Defined Types** is selected during export.

Also export user-defined data type references when a program, routine, set of rungs, or Add-On Instruction is exported.

Create export files

When exporting an Add-On Instruction, the exported Add-On Instruction includes all of its parameters, local tags, and routines. These are imported with the Add-On Instruction automatically.

Optionally, include any nested Add-On Instructions or user-defined data types that are referenced by the exported Add-On Instruction. Referenced Add-On Instructions and data types are exported to the L5X file, if the **Include all referenced Add-On Instructions and User-Defined Types** check box is selected during the export.

Add-On Instruction definition references may also be exported when a program, routine, set of rungs, or user-defined data type is exported.

If an Add-On Instruction uses Message (MSG) instruction and InOut parameters of type MESSAGE, consider exporting a rung containing the Add-On Instruction to include the MESSAGE tags. This captures the message configuration data, such as type and path.

In deciding how to manage Add-On Instruction definitions in export files, consider the goals in storing the definitions.

If	Then
Want to store many Add-On Instructions that share a set of common Add-On Instructions or user-defined data types in a common location	See *Export to separate files*.
Want to distribute an Add-On Instruction as one file	See *Export to one file*.
Want to manage each Add-On Instruction as a standalone instruction	

If	Then
Want to preserve the instruction signature on Add-On Instruction	

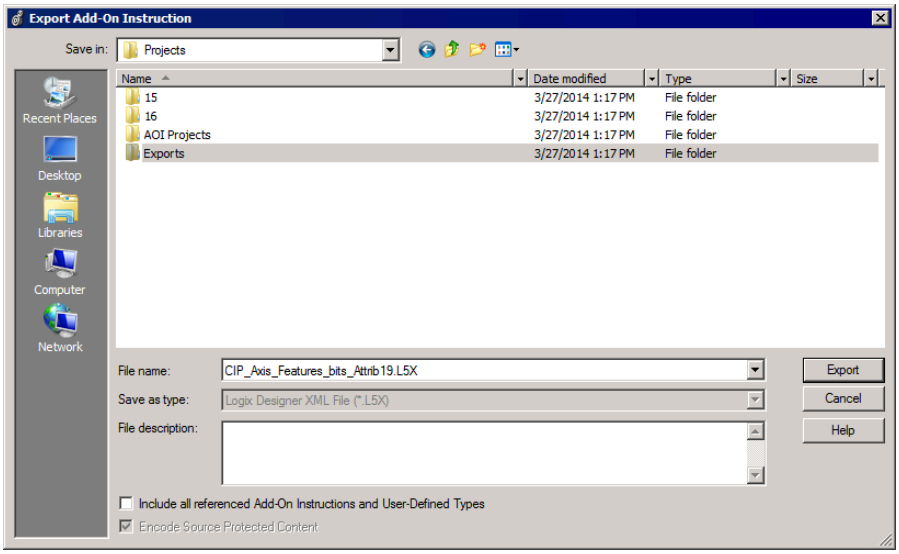
Add-On Instructions with instruction signatures are encrypted upon export to prevent modifications to the export file.

Export to separate files

To store many Add-On Instructions that share a set of common Add-On Instructions or user-defined data types in a common location, export each Add-On Instruction and user-defined data types to separate files without including references.

To export to separate files:

1. Select the Add-On Instruction in the **Controller Organizer**, and choose **Export Add-On Instruction**.
2. Select the common location to store the L5X file.



3. Type a name for the file.
4. Clear the **Include all referenced Add-On Instructions and User-Defined Types** check box.
5. Select **Export**.
6. Repeat the previous steps to individually export the other shared Add-On Instructions and user-defined data types.

By using export in this way, manage the shared Add-On Instruction and user-defined data types independently of the Add-On Instructions that reference them. Using this approach, can update the shared component without regenerating all the export files for the instructions that reference it. That is, it is stored in only one file instead of in every file whose instruction references it. This approach makes it easier to maintain the instructions to update only one export file.

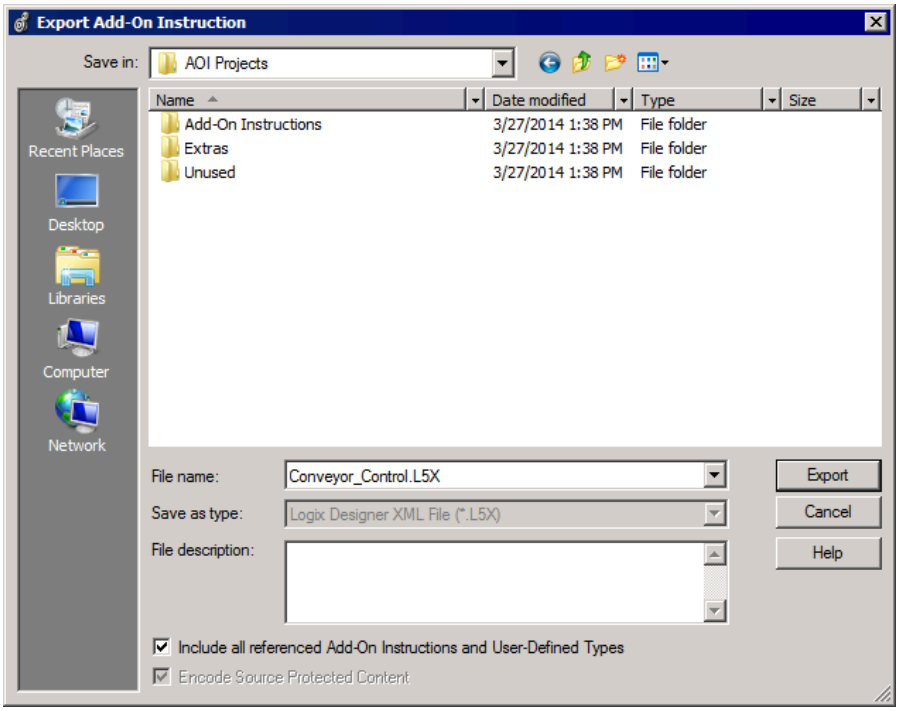
To use Add-On Instructions that were exported in a separate file without references, first import any user-defined data types of Add-On Instructions that the exported instruction references before the import of the referencing instruction can be successful. (This assumes that the referenced user-defined data types of Add-On Instructions do not exist in the project.) To do this, work from the bottom up. Import the lowest-level user-defined data types and any user-defined data types that reference them. Then, import the lowest-level Add-On Instructions, followed by any Add-On Instructions that reference those low-level Add-On Instructions. Once all items referenced by the Add-On Instruction are in place, the import of the Add-On Instruction succeeds.

Export to one file

To manage each Add-On Instruction as a standalone, export the instruction and any referenced Add-On Instructions or user-defined data types into one export file. By including any referenced Add-On Instructions or user-defined data types, also makes it easier to preserve the instruction signature of an Add-On Instruction.

To export to one file:

1. Select the Add-On Instruction in the **Controller Organizer** and choose **Export Add-On Instruction**.
2. Choose the location to store the L5X file.



3. Type a name for the file.
4. Select the **Include all referenced Add-On Instructions and User-Defined Types** check box.
5. Select **Export**.

This procedure exports the selected Add-On Instruction and all referenced instructions into the same export file. This file can be used to distribute an Add-On Instruction. When the exported Add-On Instruction is imported into the project, the referenced instructions are also imported in one step.

Import an Add-On Instruction

Import an Add-On Instruction that was exported from another Logix Designer project. When importing an Add-On Instruction, the parameters, local tags, and routines are imported as part of the Add-On Instruction. Once the project has the Add-On Instruction, use it in programs.

Import considerations for Add-On Instructions

This section covers import guidelines for Add-On Instructions or Add-On Instruction references.

ATTENTION: Editing an L5K or L5X File The EditedDate attribute of an Add-On Instruction must be updated if the Add-On Instruction is modified by editing an L5K or L5X file. If the Logix Designer application detects edits to the Add-On Instruction, but the Edited Date attribute is the same, the Add-On Instruction is not imported.

When importing Add-On Instructions directly or as references, consider these guidelines.

Topic	Consideration
Tag Data	Imported tags that reference an Add-On Instruction in the import file may be affected if the Add-On Instruction is not imported as well. In this case, the imported tag's data may be converted if the existing Add-On Instruction's data structure is different. Tag data may be lost. If an existing Add-On Instruction is overwritten, project tag data may be converted if the Add-On Instruction's data structure is different. Tag data may be lost. See *Import Configuration* for more information.
Logic	Imported logic that references the Add-On Instruction in the import file may be affected if the Add-On Instruction is not imported. If an existing Add-On Instruction is used for the imported logic reference and the parameter list of the Add-On Instruction in the project is different, the project may not verify or it may verify but not work as expected. If an existing Add-On Instruction is overwritten, logic in the project that references the Add-On Instruction may be affected. The project may not verify or may verify but not work as expected. See *Import Configuration* for more information.
Add-On Instructions While Online	An Add-On Instruction cannot be overwritten during import while online with the controller, though a new Add-On Instruction may be created while online.
Final Name Change	If the Final Name of an Add-On Instruction is modified during import configuration, the edit date of the imported Add-On Instruction is updated. In addition, all logic, tags, user-defined data types, and other Add-On Instructions in the import file that reference the Add-On

Topic	Consideration
	Instruction are updated to reference the new name. As a result, the edit date of any Add-On Instruction that references the Add-On Instruction is updated. Add-On Instructions that are sealed with an instruction signature cannot be renamed during import.
User-Defined data types	Add-On Instructions cannot overwrite user-defined data types. Add-On Instructions and user-defined data types require unique names.
Instruction Signature	When importing an Add-On Instruction with an instruction signature into a project where referenced Add-On Instructions or user-defined data types are not available, consider removing the signature. Overwrite an Add-On Instruction that has an instruction signature by importing another Add-On Instruction with the same name into an existing routine. Add-On Instructions that are sealed with an instruction signature cannot be renamed during import.
Safety Add-On Instructions	Importing a safety Add-On Instruction into a standard project is not allowed. Importing a safety Add-On Instruction into a safety project that has been safety-locked or one that has a safety task signature is not allowed. Import a safety Add-On Instruction while online. Class, instruction signature, signature history, and safety instruction signature, if it exists, remain intact when an Add-On Instruction with an instruction signature is imported.

IMPORTANT: Importing an Add-On Instruction created in version 18.00.00 or later of the application, into an earlier project that does not support Add-On Instruction signatures causes the Add-On Instruction to lose attribute data and the instruction may no longer verify.

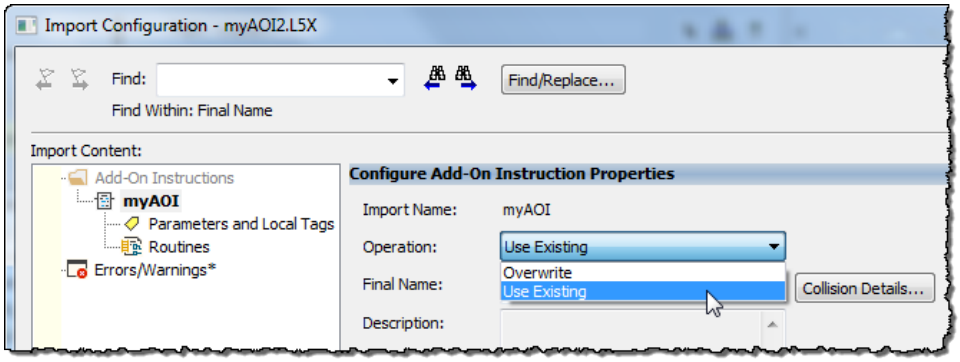
Import configuration

When selecting a file to import, use the **Import Configuration** dialog box to choose how the Add-On Instruction and referenced components are imported.

If no issues exist, select **OK** to complete the import.

If the Add-On Instruction collides with an Add-On Instruction in the project:

- Rename it by typing a new, unique name in the **Final Name** field.
- Choose **Overwrite** from the **Operation** menu.
- Choose **Use Existing** from the **Operation** menu.



Rename an Add-On Instruction only if it has not been sealed with an instruction signature.To rename an Add-On Instruction that has been source-protected, use the source key.

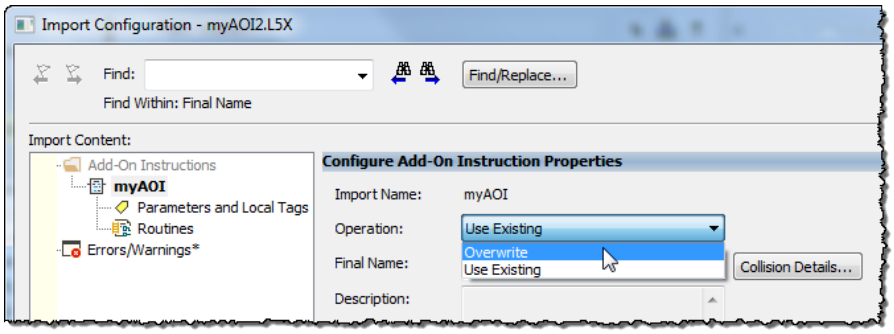
Use the **Collision Details** button to view the **Property Compare** tab, which shows the differences between the two instructions, and the **Project References** tab, which shows where the existing Add-On Instruction is used and how the arguments are updated to locations where the existing Add-On Instruction is called.

Update an Add-On Instruction to a later revision using import

To update an instruction to a later revision, import it from an .l5x file or copy it from an existing project. Update an Add-On Instruction only when offline.

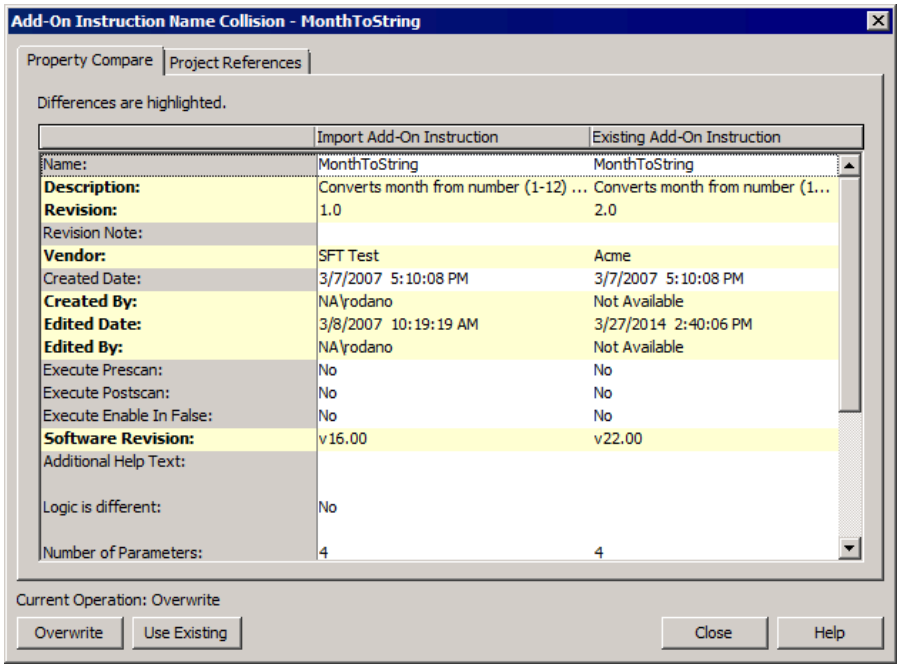
To update an Add-On Instruction to a later revision using import:

1. Select the **Add-On Instruction** folder and choose **Import Add-On Instruction**.
2. Select the file with the Add-On Instruction and select **Open**.
3. From the **Operation** list on the **Import Configuration** dialog box, choose **Overwrite**.



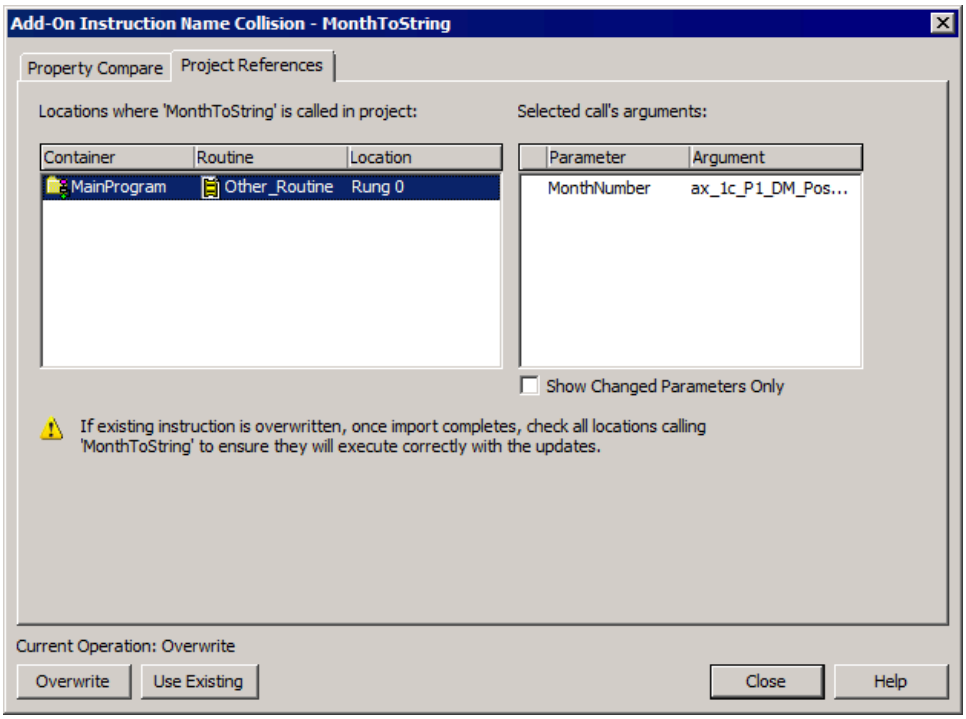
4. Select **Collision Details** to see any differences in the Add-On Instruction definition, and to see any changes that will be made to the logic where the Add-On Instruction is called.

Differences listed in the **Property Compare** tab are shown in bold in the **Name** column.



The **Compare** dialog box compares metadata for each instruction definition, such as description, revision, or edited date. For effective revision control, enter a detailed revision note in the **Add-On Instructions Definition** dialog box. To open the dialog box, right-click an Add-On Instruction and select **Open Definition**.

The **Project References** tab shows a list of locations where the Add-On Instruction is called, and for each location, shows how the arguments in the Add-On Instruction will be updated to adapt to the new parameters.



IMPORTANT: Beginning with version 18.00, when changing the parameters of an Add-On Instruction (add, delete, or move), each location where the Add-On Instruction is called is modified so that the existing arguments continue to match their previous parameters. Importing and Overwriting an existing Add-On Instruction may cause changes to the existing logic that uses the Add-On Instruction.

IMPORTANT: If the logic calling the Add-On Instruction is within a source-protected routine, and the key is not available, the arguments are not shown or updated. Instead, the Location will be identified as Source Not Editable.

For more information on updates to arguments, see *Logix 5000 Controllers Add-On Instructions*.

Import and export tag-based alarms and alarm definitions

Introduction

This chapter explains export and import of tag-based alarms and alarm definitions.

Create export files

Tag-based alarms and alarm definitions can be exported to an XML file for offline editing, and they are exported to the L5X file as part of a program, routine, user-defined data type, or Add-On Instruction export.

Keep these considerations in mind when editing alarms and alarm definitions in an XML file or in a spreadsheet:

- One message is exported for each alarm or alarm definition.
- Delete alarms and alarm definitions from the XML file, but those items are not removed from the Logix Designer project when importing the modified XML file.

To create export files (tag-based alarms):

1. On the main menu, select **Tools > Export > Alarms** or **Tools > Export > Alarm Definitions** to open the **Export Alarms and Alarm Definitions** dialog box.
2. Select a destination for the file and adjust the file name if necessary.
3. Select **Save**.

To open an XML file in an Excel spreadsheet, open it as an XML table. After editing, select **XML data** as the file type in the **Save as** dialog box.

Import alarms and alarm definitions

After exporting alarms or alarm definitions to an XML file, import the file into a project.

Keep these consideration in mind when importing alarms and alarm definitions from an XML file:

- If alarms and alarm definitions are deleted from the XML file, those items are not removed from the Logix Designer project when importing the modified XML file.

To import tag-based alarms and alarm definitions:

1. On the main menu, select **Tools > Import > Alarms** or **Tools > Import > Alarm Definitions** to open the **Import Alarms and Alarm Definitions** dialog box.
2. In the **Import** dialog box, select the file and select **Open** to open the **Collision Handling** dialog box.

3. On the **Collision Handling** dialog box, select the method the Logix Designer application should use in case of differences or duplication between the imported alarms and alarms that exist in the project.

   This table describes the handling options.

Option	Description
Create New Alarms & Overwrite Existing Alarms	New alarms in the XML file are created in the project, and existing alarms in the project are overwritten with the same alarms in the XML file. This is the default option.
Create New Alarms & Preserve Existing Alarms	New alarms in the XML file are created in the project, but existing alarms in the project are not overwritten with the same alarms in the XML file.
Skip New Alarms & Overwrite Existing Alarms	New alarms in the XML file are not created in the project, and existing alarms in the project are overwritten with the same alarms in the XML file.

Import considerations

- Keep these considerations in mind when importing an XML file containing alarms or alarm definitions:
  - The import file must be an XML file. If the import file is not an XML file, try opening the file in Excel and, in the **Open XML** dialog box, select **As an XML table**. Then save the file as an XML file by selecting **XML data** as the file type in the **Save as** dialog box.
  - The XML file must be in the same alarm format as the exported file. If it was corrupted somehow or incorrectly edited after it was exported, try exporting the alarm list again and import the resulting XML file.
  - The XML file must contain only the attributes that are recognized as attributes of an alarm or an alarm definition, such as AckRequired, Latched, and Severity. If an XML file is edited incorrectly after export, an attribute might be altered to make it unrecognizable, or an unknown attribute might be added. Remove or correct any unrecognized attributes in the XML file.
  - The XML file must contain only the elements that are recognized as elements of an alarm or an alarm definition. If an XML file is edited incorrectly after export, an attribute might be altered to make it unrecognizable, or an unknown element might be added. In the alarm import XML file, elements appear in angle brackets, such as <Message>, <AlarmClass>, and <HMIGroup>. Remove or correct any unrecognized elements in the XML file.

Keep this consideration in mind when copying and pasting an Add-On Instruction in a project:

- When an Add-On Instruction (AOI) tag is copied and pasted in a project, alarm definitions associated with the tag are not included.

  After you copy and paste an AOI tag, open the **Alarm Definition list** and copy and paste the alarm definition for the tag. Use these steps:

  1. In the **Controller Organizer**, right-click **Alarms** and select **Edit Alarm Definitions**.
  2. Right-click the alarm definition for the AOI tag and select **Copy**.
  3. Right-click again and select **Paste**. The alarm definition is pasted into the list with **_000** added to the alarm name.
  4. Double-click the copy of the alarm definition to open the **Alarm Definition Properties** dialog box.
  5. In the **Input** box, change the input tag to the AOI tag that you copied and pasted.

Additional considerations for tags

Introduction - Tags

This chapter explains import and export of referenced tags.

Starting with version 24.00.00 of Logix Designer application, use program parameters to share data between programs in much the same way controller-scoped tags are used. Program parameters are imported and exported in the same way as tags in most instances. For more information on program parameters, refer to 1756-PM021, *Logix 5000 Controllers Program Parameters Programming Manual*.

Export considerations for tags

Tags are not exported to an .l5x file themselves, but they are exported to the L5X file as part of a program, equipment phase, or Add-On Instruction export (program-scoped tags with a program export, equipment phase-scoped tags with an equipment phase export, and parameters and local tags with an Add-On Instruction export).

Tags may also be exported to an L5X file as references from another component being exported (controller-scoped tag references with a program or equipment phase export or program-scoped and controller-scoped tag references with a rung or routine export). The definitions for the referenced tags are exported to the L5X file by default if they exist in the project.

When exporting logic, especially if it is intended for general use, be aware that logic that references a bit member of a tag or member of a user-defined data type tag cannot be replaced during import to reference another bit or user-defined data type member. To connect the logic reference to another bit or member of a tag, consider editing the logic before export so the reference is to a full tag name (and alias if need be) so that the reference can be connected to the desired tag on import.

Import considerations for tags

When importing a program, the program-scoped tags are imported with the program automatically (the same as they are for equipment phase). When importing an Add-On Instruction, the parameters and local tags are imported with the Add-On Instruction automatically.

During program or equipment phase imports, referenced controller-scoped tags may also be imported. For routine and rung imports, referenced tags may be imported as well. Configure how referenced tags are imported during import configuration. By default, referenced tags that collide with project components are not imported.

Keep these considerations in mind when importing tags.

Topic	Consideration
Tag data	Imported tags that reference an Add-On Instruction or user-defined data type in the import file may be affected if the Add-On Instruction or user-defined data type is not imported as well. In this case, the imported tag's data may be converted if the existing data structure is different and tag data may be lost.

Topic	Consideration
	If an existing Add-On Instruction or user-defined data type is overwritten, project tag data may be converted if the data structure is different and tag data may be lost. If the data is not convertible, it is overwritten with the default values for the type. When array dimensions are changed, existing array members retain their values and descriptions and new members have the default values and description (usually 0 and no description).
Consumed tags	Consumed tags cannot be imported from an L5X file. They are converted to base tags and a warning appears in the **Errors** or **Warning** pane during initial parsing of the L5X file.
Tag values while online	When importing into a controller while online, if existing tags are being overwritten by imported tags, the tag values are not written to the controller. Tag values are written only to the offline project. The tag values in the controller maintain their current values but other tag attributes are written to the controller. Values for tags that are created during import are written to the controller. However, existing tag values are never overwritten in the online controller. Prevent tag values from being overwritten in the offline project by selecting **Preserve existing tag values in offline project** on the **Import Configuration** dialog box.
Tag values while offline	Prevent tag values from being overwritten in the offline project by selecting **Preserve existing tag values in offline project** on the **Import Configuration** dialog box.
Tag attributes while online	Tag attributes (for example, External Access, Constant, and Style) are written to the online project and the offline project. If existing tags are to be overwritten with new attributes that are incompatible with existing user logic, the import is not allowed.
Tag scope	An import tag maintains the scope of the tag as it was when exported if the tag initially collides with another scoped tag in the project. In that case, an attention (red) flag appears on the tag indicating the scope collision. However, If changing the **Final Name** of a tag so that it subsequently collides with a tag of another scope in the project, the imported tag is changed to the scope of the existing tag. Resolve the attention flag that appears on initial collision due to a scope issue by changing the Final Name to avoid the collision with that tag or, if the import component is routine or rungs, change the scope of the

Topic	Consideration
	import tag by selecting the tag row and selecting **Toggle Tag Scope**.

Additional considerations for data exchange

Data exchange

Exchange data files represent Rockwell Automation devices in the automation system produced in a tool such as EPLAN© Electric P8 or AutoCAD© Electrical. Create a Logix Designer project file that represents a controller-centric subset of the system without having to manually add, name, or recreate high-level configuration of the associated devices.

- Import or export hardware diagrams to or from a standard file type to share data between different software applications and the Logix Designer Professional Edition using these formats:
  - AutomationML (AML)
  - Resource Description Framework (RDF)
  - Web Ontology Language (OWL)

The .rdf and .aml file types are used only for PLC data exchange as described below. When saving the Logix Designer project file to the data exchange file types, none of the tasks, programs, routines, Add-On Instructions, user-defined type data from the project is stored. Only export the project file types when exchanging data with eCAD tools.

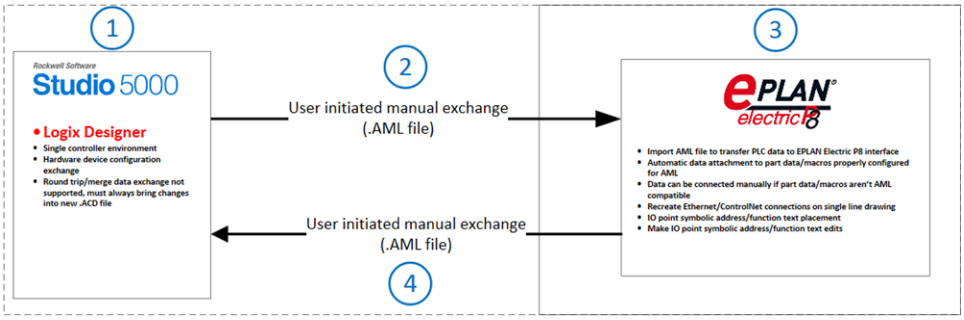
Import of eCAD data via data exchange file types is only possible when creating a new Logix Designer project. This data cannot be merged into an existing project.

Data exchange features differ between Studio 5000 Logix Designer and Studio 5000 Architect. Use the data exchange interface in Studio 5000 Architect to exchange I/O point tag/alias data and to conduct merge workflows into Studio 5000.

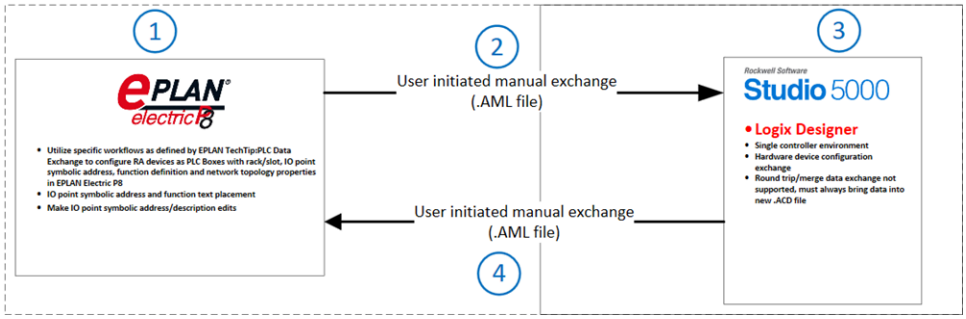
EPLAN Electric P8

For EPLAN Electric P8, use AML data files (.aml) and associated PLC data import/export workflows to exchange data between the Logix Designer application, versions 32.00 and later, and EPLAN Electric P8, versions 2.9 and later.

This workflow shows the process for data exchange from supported Studio 5000® applications to the EPLAN Electric P8 application when starting the data exchange within Studio 5000.



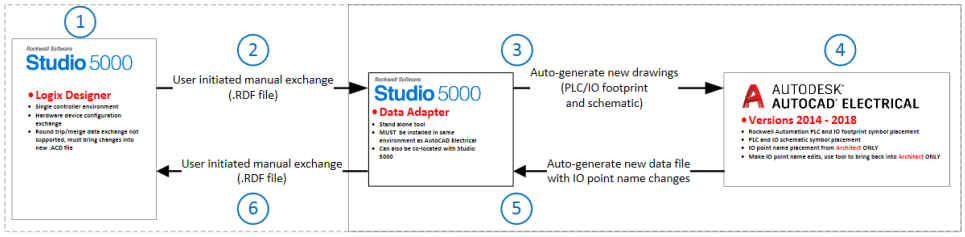
This workflow shows the process for data exchange from the EPLAN Electric P8 application to supported Studio 5000 applications when starting the data exchange within EPLAN.



AutoCAD Electrical

For AutoCAD Electrical, use the Studio 5000 Data Adapter tool to exchange data between the Logix Designer application version 32.00 and later and AutoCAD Electrical versions 2014 through 2018. The Data Adapter tool translates Rockwell Automation hardware data in OWL (.rdf) format for auto generation of two-dimensional footprint and PLC I/O schematic drawings in AutoCAD Electrical. Download the Studio 5000 Data Adapter tool from Rockwell Automation's Product Download and Compatibility Center (PCDC).

This workflow shows the supported process for bidirectional data exchange to AutoCAD Electrical applications.



Data exchange with other software applications

Import or export hardware diagrams to or from a standard file type to share data between different software applications.

Import and export in the Logix Designer application support these file formats:

- AutomationML (AML) uses an .aml file to exchange data.

  AML is an open industry standard format dedicated to automation usage. The AML specification includes a set of data structures implemented in the application. This implementation enables interoperability with other software applications that use the AutomationML Automation Project Configuration format, such as EPLAN Electric.

  Some Rockwell Automation devices are only partially supported by the AML format because they include capabilities that are not within the scope of the current AML standard. When a product is partially supported, it can be imported or exported but may require some additional configuration to reconfigure the unsupported features. Check the Errors window in the Logix Designer application for details.

- Web Ontology Language (OWL) uses an .rdf file to exchange data.

  Recommended for the most rich data exchange with all Rockwell Automation specific features identified for data exchange. The RDF (Resource Description Framework) serves

for data representation, OWL (Web Ontology Language) builds on RDF and defines the dictionary. Both are open standards for data exchange. They are used as a basis for a Rockwell Automation specific data format.

When exchanging data between Rockwell Automation software applications, always use the RDF file type.

Data exchange file type capabilities and limitations

This topic lists devices and components supported in data exchange files.

AutomationML (AML) is an XML-based data format for storing automation-related data. Import and export AML and RDF within the Logix Designer application.

The Logix Designer application supports these devices in exported and imported data exchange files. Rockwell Automation plans to support additional devices in future releases of the Logix Designer application.

- Chassis

  AML files store chassis size for the 1756 chassis only. If you synchronize a project with an AML file and then update the size for a non-1756 chassis, the size change does not appear as a difference in the Synchronize dialog.

- Controllers
  - Controllers with Ethernet

  NOTE: Controllers with configurable safety, such as ControlLogix 5590 controllers, are imported into the default controller configuration for a new project. For ControlLogix 5590 controllers, the deault configuration is Safety SIL2.

  NOTE: When you add a new device connected to a dual-Ethernet controller during an AML synchronization, that device is always imported into the Logix Designer project on the A1 Ethernet port.

- I/O modules
- ControlNet communication modules
- Ethernet communication module
- Motion modules
- Redundancy modules
- Specialty modules
- Ethernet switches
- Servo drives
- Ethernet network
- Ethernet connection
- PowerFlex devices

Some devices are not supported by AML or RDF data exchange with the Logix Designer application and, if they are present, are omitted from the export file. These devices are not supported:

- CompactLogix L2x, L3x, L4x, -L37ERM0, and -L37ERM0S controllers
- HMI devices
- Classic profiles (profiles that do not have a **Module Definition** dialog box) (excluding controllers)
- Devices not included:
  - 1734 Address Reserve module (1734-ARM)
  - 1794 Extended-Local I/O adapter (1794-FLA)
  - Unmanaged switches (1783-US)
  - Generic switch
  - Generic computer
  - Flex Terminal Base Unit device (1794-TB, 1797-TB)
  - Power Supply devices
  - End Cap and cable devices
- Stratix expansion module switches (1783-MX, 1783-MS, 1783-RMS)
- PowerFlex Peripheral modules
- DeviceNet
- ControlNet
- HART
- Other trunks

Some devices cannot be exported to AML. These devices are:

- CompactExtensionConnection (indicating a multi-part chassis)
- FlexLogixTrunk

IMPORTANT: Attempting to export a hardware diagram that includes either of these devices results in an error and the export does not succeed.

Device properties

AML and RDF support storing a subset of device and tag properties. This table identifies the properties that are stored for data exchange with the Logix Designer application:

Device	Property
All	GUID
	Name
Chassis, Modules in Chassis, Independent physical devices	Catalog number

Device	Property
	Position
	Firmware version
	IP address
	Description

Unsupported properties are omitted in an AML export. Some properties can be calculated and restored upon import and others must be explicitly configured after import. Unsupported properties are:

- LeftRackSize and RightRackSize properties of CompactLogix chassis. If possible this data is calculated during post-processing and restored during import.
- Redundancy
- Safety Level
- Servo drive grouping
- Stratix expansion module owner assignment. If possible this data is calculated during post-processing and restored during import.

Importing and exporting tags with data exchange files

You can import and export tags using data exchange files. This allows you to give descriptive names to individual channels of I/O points. In Logix Designer, these tags are defined as alias tags to module-defined I/O tags. To export an alias tag to a data exchange file, it must meet these conditions:

- It must be a verified alias tag.
- It must be an alias to a module-defined Input or Output tag.
- The target module must support alias tags.
- The target module must be exported or, for synchronization, must already exist in the target AML file.
- The parent or target module must have a name. Unnamed modules are exported with auto-generated names, but alias tags that target those modules are not exported.
- The **Alias For** setting must point to the appropriate level of member for that specific module.

The appropriate member is typically one that contains CHANNEL in its data type; for example, CHANNEL_DI:I:0 (where PtXX is the appropriate member to target). For modules that do not contain a data type with the CHANNEL identifier, you can determine the appropriate member by performing an AML import and observing the level at which the tag is created.

Synchronize a project with a data exchange file

Use the Synchronize dialog to compare and synchronize a Logix Designer project file with a data exchange file. If a data exchange file contains updates made in a tool such as EPLAN Electric P8 or AutoCAD Electrical, you can import added devices and tags into your Logix Designer project.

Likewise, if your Logix Designer project has been updated, you can export devices and tags to an existing data exchange file.

You can add devices to a project or to a data exchange file in the Synchronize dialog, but you cannot use the Synchronize dialog to remove devices from an existing project or file.

Before you begin

- To synchronize a Logix Designer project with a data exchange file, the file must contain a controller of the same type and unique ID (GUID) as the project root controller.
- If the data exchange file contains a controller of the same type but with a different GUID, a dialog prompts you to select one of the project controllers to sync with.
- If the data exchange file contains no controllers of the same type as the Logix Designer project root controller, take one of these steps:
  - Morph the Logix Designer project to a controller that is present in the data exchange file, or
  - Edit the data exchange file using an external application, such as EPLAN, to add a controller of the same type as the Logix Designer project root controller.

To synchronize a project with a data exchange file

1. On the Logix Designer main menu, select **Tools > Synchronize AML/RDF**.
2. In the **Select File** dialog, select the data exchange file to compare and synchronize.
   In the **Synchronize AML/RDF** dialog, the open Logix Designer project appears in the left pane and the data exchange file appears in the right pane.
3. In either pane, select an item to import to the Logix Designer project or export to the data exchange file. For example, if the Logix Designer project contains a module that does not exist in the data exchange file, the module appears in the Current Logix Project pane under **Items unique to this source**.
4. Select **Export** or **Import** to include the device or tag in the export or import:
   - To include the item for export to the data exchange file, select **Export**. To include all items, select **Export all**.
   - To include the item for import to the Logix Designer project, select **Import**. To include all items, select **Import all**.

   The device or tag appears under **Items to export** or **Items to import**.

   To remove a device or tag from Items to export or Items to import, select the item and select **Undo**. Select **Undo all** to remove all components and start over.
5. Select **Finish**. The component is added to the Logix Designer project or to the data exchange file.

Property names in EPLAN Electric P8

These tables list the names of Logix Designer properties that have different names in EPLAN® Electric P8™.

Device properties in Logix Designer and EPLAN Electric P8

Property	Logix Designer name	EPLAN Electric P8 name
Device name	Controller name or Device name	PLC card name
Description	Controller description or Module description **Tip:** In an AML file, Description appears as Comment.	Function text
Catalog number	Device type	PLC type designation
Position	Slot **Tip:** In an AML file, Position appears as PositionNumber.	Position (slot/module)
Firmware version	Revision	Version (only the major revision is included)
IP address	IP address Physical network: Name (only visible in an AML file)	Physical network: Bus ID/item number Physical network: Name
GUID	ID (only visible in an AML file)	AutomationML GUID

Tag properties in Logix Designer and EPLAN Electric P8

Property	Logix Designer name	EPLAN Electric P8 name
Name	Name	Symbolic address
Description	Description	Function text
I/O Point Direction or Point Type	Expressed in a tag in this format: Local:2:**I**:Data.0 (**I** is for digital **input** direction) Local:8:**O**.Data.4 (**O** is for digital **output** direction)	PLC Connection Point, DO (Digital Output) PLC Connection Point, DI (Digital Input) PLC Connection Point, AO (Analog Output) PLC Connection Point, AI (Analog Input)
Automation ML Guid or Data Exchange Id	These properties appear only in L5X files and AML files. • In an L5X file, the value appears in this format: <Tag Name="TestChangedControllerTag" TagType="Alias" Radix="Decimal" AliasFor="Local:1:I.Data.5" ExternalAccess="Read/Write" OpcUaAccess="None"	Automation ML GUID In EPLAN Electric P8, the GUID appears only for these items: • The controller TagTable • Alias tag • Channel

Property	Logix Designer name	EPLAN Electric P8 name
	DataExchangeId="{857CA7B5-9111-4FA8-B8C7-027947E8F237}"> • In an AML file, the GUID appears at the end of a tag reference or at the end of a channel reference as ID="<guid>"	
Channel Number	Expressed in a tag in this format: Local:2:I:Data.**0** (**0 at the end** denotes channel 0) Local:1:I.**Ch4Data** (**Ch4Data at the end** denotes channel 4) **Note:** Logix Designer module profiles sometimes use compound types as opposed to atomic types. The atomic types from the AML file are converted to a compound type on import and are converted back to an atomic type on export to keep the integrity of the AML file working with EPLAN. Local:3:I.**Pt7** (**Pt7 at the end** denotes channel 7)	Channel Designation **Note:** This value is required for importing the AML file into Logix Designer. If it is not set, you can edit it, using a sequential value from 0 to N.
Data Type	Data Type **Note:** The Logix Designer application sets Data Type automatically to match the module profile.	Data Type **Note:** This value is required for importing the AML file into Logix Designer. If it is not set, you can edit it.

Rockwell Automation Support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, and product notification updates.	rok.auto/support
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Technical Documentation Center	Quickly access and download technical specifications, installation instructions, and user manuals.	rok.auto/techdocs
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	rok.auto/pcdc

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.

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.

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**®

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000
EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2663 0600
ASIA PACIFIC: Rockwell Automation SEA Pte Ltd, 2 Corporation Road, #04-05, Main Lobby, Corporation Place, Singapore 618494, Tel: (65) 6510 6608
UNITED KINGDOM: Rockwell Automation Ltd., Pitfield, Kiln Farm, Milton Keynes, MK11 3DR, United Kingdom, Tel: (44)(1908) 838-800

Publication 1756-PM019L-EN-P - September 2025
Supersedes Publication 1756-PM019K-EN-P - March 2022