Guideline for Library Handling in TIA Portal

TIA Portal

https://support.industry.siemens.com/cs/ww/en/view/109747503

Siemens Industry Online Support

Warranty and Liability

Note

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These Application Examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice.
If there are any deviations between the recommendations provided in these Application Examples and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We do not accept any liability for the information contained in this document.
Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act ("Produkthaftungsgesetz"), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract ("wesentliche Vertragspflichten"). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.
Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of the Siemens AG.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.
Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.
Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit http://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under http://www.siemens.com/industrialsecurity.

Table of Contents

1 Introduction

1.1 The type instance concept

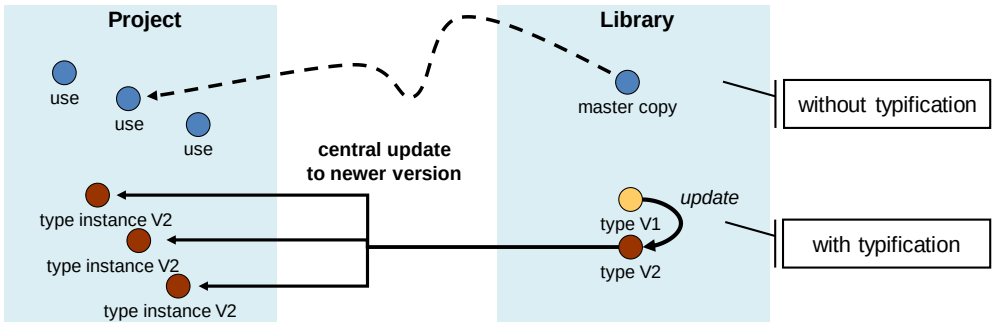
1.1.1 Objective of the guideline

This guideline describes the creation of a corporate library by using typified library elements in TIA Portal.

You have the option to create types or master copies of TIA Portal elements (e.g. function blocks) (see Figure 1-1: Typification with user libraries).
By using types, they are “instanced” in the project All type instances have a fixed connection to your type in the library. This is how you can control the version of the library elements and make changes centrally that are then updated in the complete project in a system-supported way.
Master copies are independent copies of TIA Portal elements. They have no connection to your places of use in the project and no option to control the version in a system-supported way or any central changeability.

Your decision should always be based on these differences, whether you want to create type or master copies in the libraries.

Figure 1-1: Typification with user libraries



1.1.2 Advantages of type instance concept

Based on the following advantages, this guideline recommends the use of types:

- Central update function of all instances in the project
- Central display of the places of use of the types
- System-supported version management of library elements
- System-supported change tracking of library elements
- Consistent library elements for several users
- Typified elements (type instances) are especially highlighted in the project

1.1.3 Using the type instance concept

In order to make optimum use of the advantages of the type instance concept, there needs to be a defined workflow. The optimum workflow is described in chapter 2 Creating a Corporate Library.

1.1.4 Topics not covered by this guideline

TIA Portal

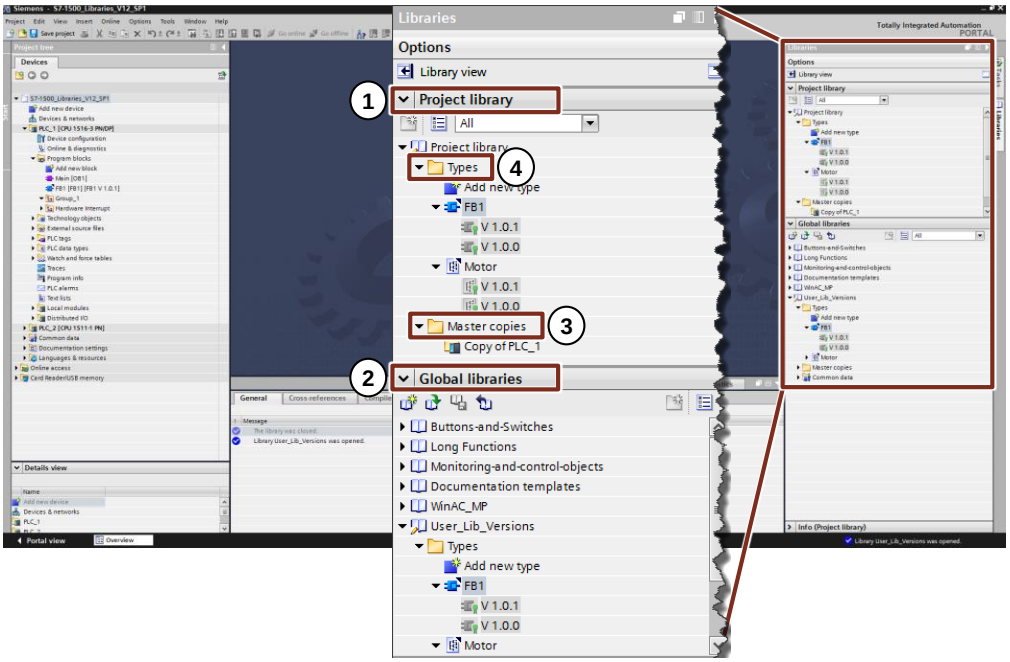
All descriptions and step-by-step instructions in this guideline as well as the system behavior relates to TIA Portal.

SIMATIC S7-1200 und S7-1500 PLCs

This guideline is related to SIMATIC PLCs of the product families S7-1200 and S7-1500.

1.2 Libraries and library elements in TIA Portal

Figure 1-2: Libraries in TIA Portal



1. Project library
   - Integrated in the project and managed using the project
   - Allows the reusability within the project
2. Global library
   - Independent library
   - Can be used in several projects
   - To distribute ready library versions to other users

A library includes two different types of storage of library elements:

3. Master copies
   - Copies of configuration elements in the library (e.g. blocks, hardware, PLC tag tables, etc.)
   - Copies are not connected with the elements in the project.
   - Master copies can also be made up of several configuration elements.
4. Types
   - Types are connected with your usage locations in the project. When types are changed, the usage locations in the project can be updated in a system-supported way.
   - Supported types are blocks (FCs, FBs), PLC data types, HMI screens, HMI faceplates, HMI UDTs, scripts, etc. see chapter 8.2 Overview of all library elements and chapter 8.3 Differences of the types for PLC and HMI.
   - Subordinate control elements are automatically typified.
   - Types are versioned: Changes can be made by creating a later version.

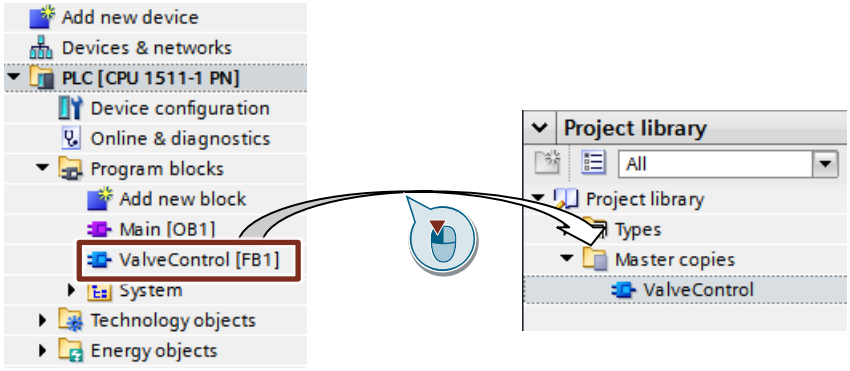
1.3 Handling of types and master copies

Based on PLC blocks, this chapter describes the basic handling of types and master copies in project and global libraries.

1.3.1 Storing blocks as master copies in the project library

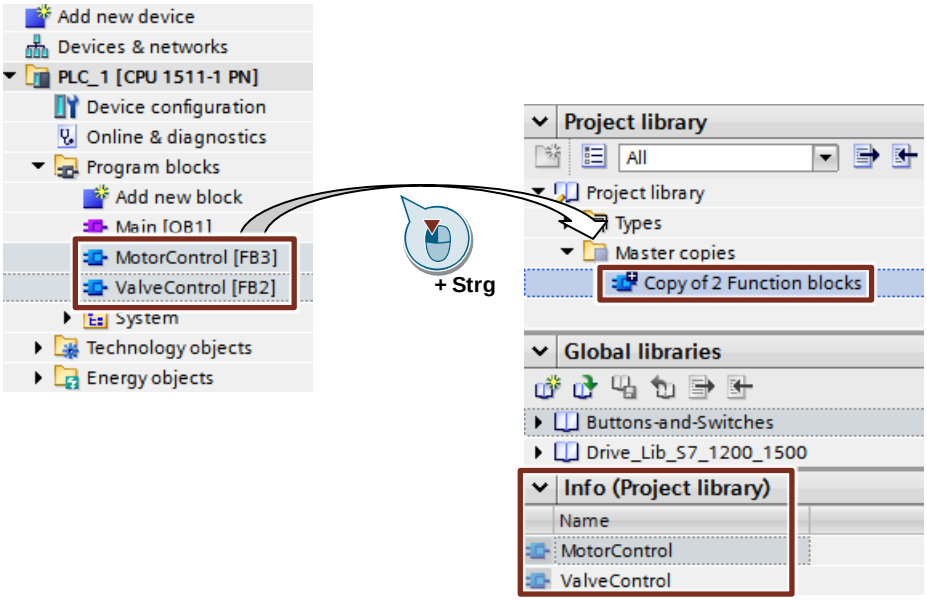
Blocks can be stored as loose copies in "Master copies" in the "Project library" using drag-and-drop.

Figure 1-3: Storing blocks as master copies in the project library



If you want to store several blocks as master copies in the project library, you can store the blocks individually or as a group. In order to store several blocks as a group, when storing using drag-and-drop, you have to press the key "Ctrl" or "Alt". You can give the group any name. The content of the group is displayed under "Info".

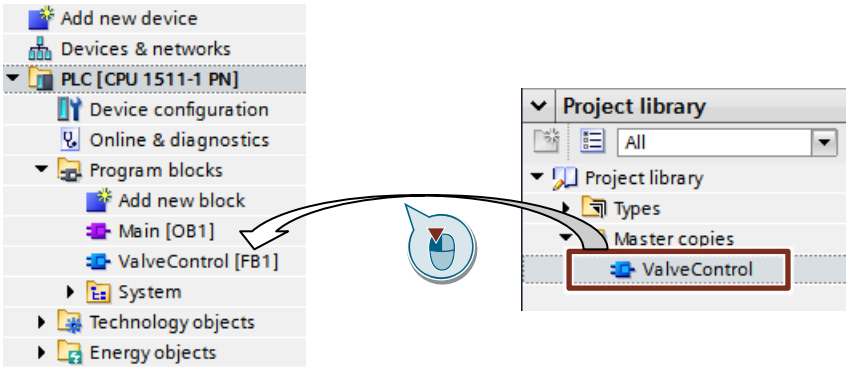
Figure 1-4: Storing several blocks as master copies in the project library



1.3.2 Adding blocks as master copy in the project

The "Project library" is used as storage for reusable elements (e.g., blocks) within a project. The blocks can be added in the device using drag-and-drop.

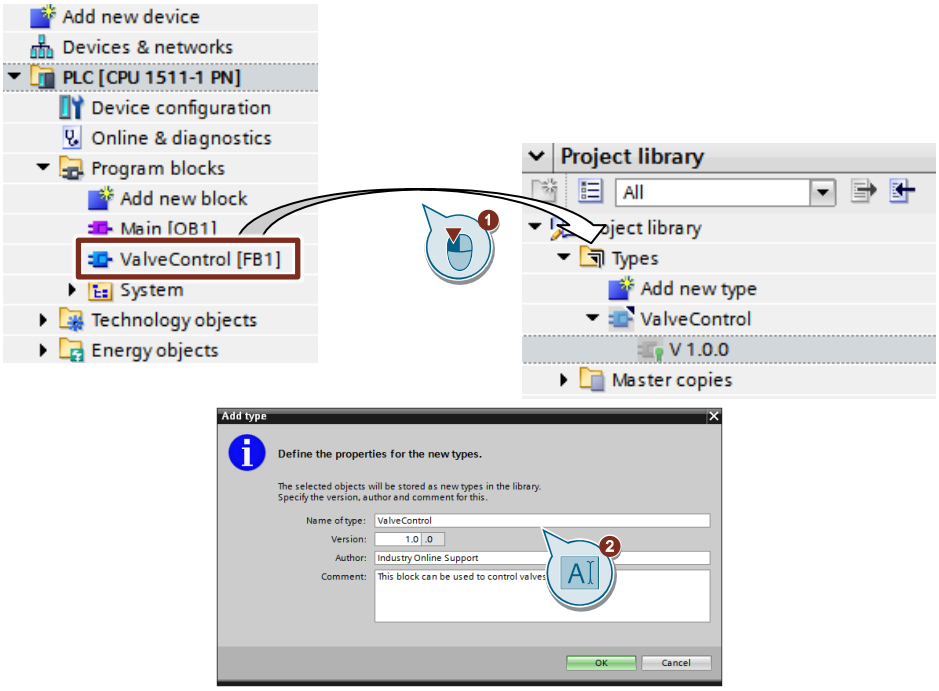
Figure 1-5: Adding blocks as master copy in the project



1.3.3 Storing block as type in the project library

Blocks can be stored as versionable items in "Types" in the "Project library" using drag-and-drop. A type has meta data (name, version, author, comment), which can be updated when creating it.

Figure 1-6: Storing block as type in the project library

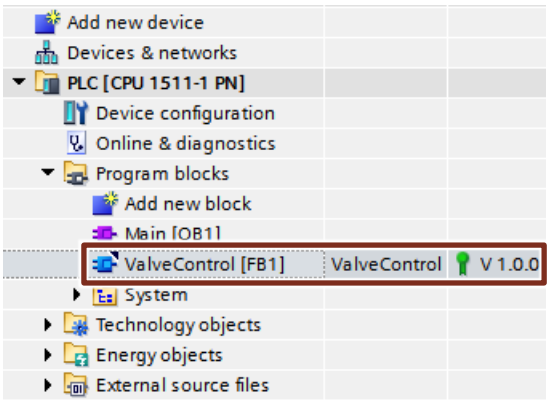


Note In order to typify safety blocks, they have to be accessed in the F sequence program. The F program has to be consistent.

Result

All typified blocks are marked with a triangle as type instance in the object icon, type name and version.

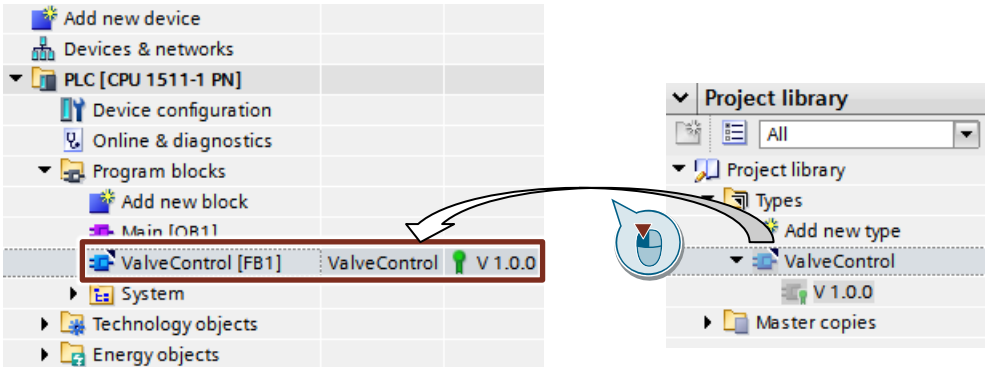
Figure 1-7: Type instances in program



1.3.4 Using typified blocks in project

The project library is used as storage for reusable elements (e.g. blocks) within a project. The typified blocks can be instanced in the program using drag-and-drop.

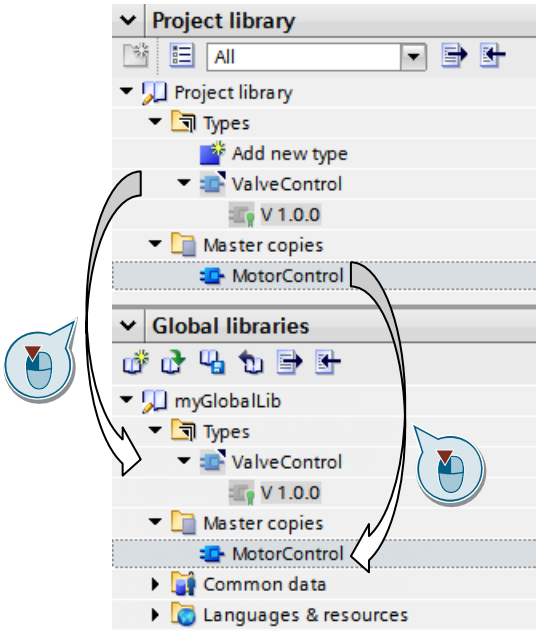
Figure 1-8: Instancing block as type in the project



1.3.5 Copying blocks as master copy/copying type in global library

- All elements of a project library can also be stored in a global library.
- Master copies can be stored directly in the global library as described above and stored using drag-and-drop.
- Types always have to be created in the project library before they can be stored in the global library.
- Mater copies and types can be copied from the project library to the global library using drag-and-drop.

Figure 1-9: Copying master copies/types from the project library to the global library



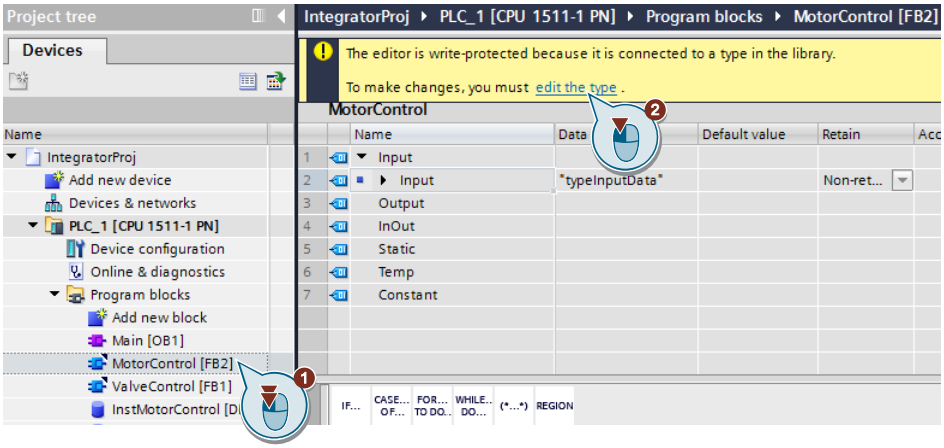
1.3.6 Editing type

Note Types that are edited, always require a test environment. This means the type has to be instanced in a PLC program

Editing and enabling individual types

When opening typified elements via the project navigation they are write protected in the editor. You can set the type to the status [in test], in order to edit it.

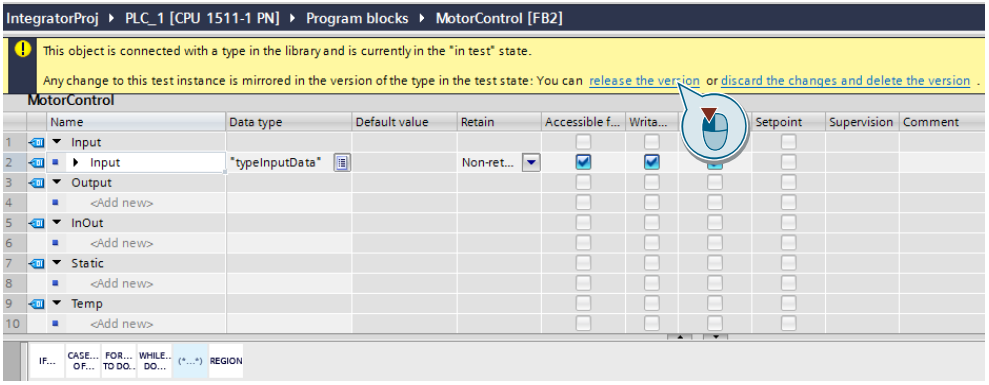
Figure 1-10: Editing type



Below the info area highlighted in yellow, there is the editor toolbar. The info area can be closed with a click on the exclamation mark and be opened again.

Once you have completed editing, you can either release the type or discard all changes.

Figure 1-11: Enabling type

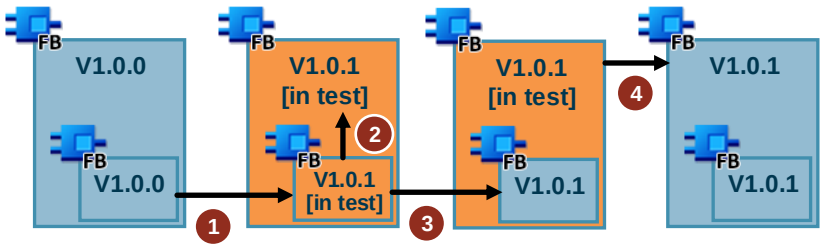


Note In order to typify safety blocks, they have to be accessed in the F sequence program. The F program has to be consistent.

Editing and releasing nested types

If you have several types that are nested in each other, these types have a direct effect on each other. In order that only the current versions of all types are used, edit a subordinate type and release it. Afterwards you have to update the higher-level one.

Figure 1-12: Editing and releasing nested types



1. Set the block that you want to edit [in test].
2. The TIA Portal will put the overlay type automatically in the status [in test], based on the direct dependence.
3. Release the subordinate block and assign a new version.
4. Release the overlay block and assign a new version.

Note You also have the option to release all types that are [in test] using the command “Release all".

1.3.7 Other basics

Recommendation

If it is the first time that you are dealing with libraries in TIA Portal, read the chapter 8 Valuable Information / Basics first.

1.4 Central storage for global libraries

Global libraries can be stored centrally as corporate libraries on a server. These global library can be opened automatically when starting the TIA Portal. As soon as newer versions of the global libraries are available, you can replace them centrally. A message informs the user of this global library about it the next time the library is opened.

Note Further information is available in the entry "When starting TIA Portal V13 onwards, how do you get a global library to open automatically and use it as corporate library, for example? in https://support.industry.siemens.com/cs/ww/en/view/100451450

and in the help of the TIA Portal in "Using libraries > Using global libraries > Using global corporate libraries"

2 Creating a Corporate Library

Corporate libraries are global libraries that are centrally provided to a user circle in a company for creating plant projects.

Note For more information, please refer to the topic page "Libraries in the TIA Portal" https://support.industry.siemens.com/cs/ww/en/view/109738702

2.1 Programming guideline

In order to create a corporate library, the creators should observe a joint and agreed upon programming guideline. This offers the following advantages:

- Consistent continuous programming structure
- Easily readable and comprehensible
- Simple maintenance and reusability
- Easy and quick troubleshooting and error correction
- Efficient working on the same project/library with several creators

Note Recommendation: Programming Styleguide for S7-1200 and S7-1500 https://support.industry.siemens.com/cs/ww/en/view/81318674

2.2 Workflow

The workflow described shows you how you can efficiently create, distribute and further develop a corporate library. A corporate library is used as central storage for TIA Portal elements that is published for the users after completion.

This offers the following advantages:

- Securing an optimized workflow through
  - process-oriented working
  - clear assignment of roles integrator, developer, user
- consistent library throughout the complete workflow
- convenient editing and maintenance
- easy use in the plant projects

The workflow is split in the process areas:

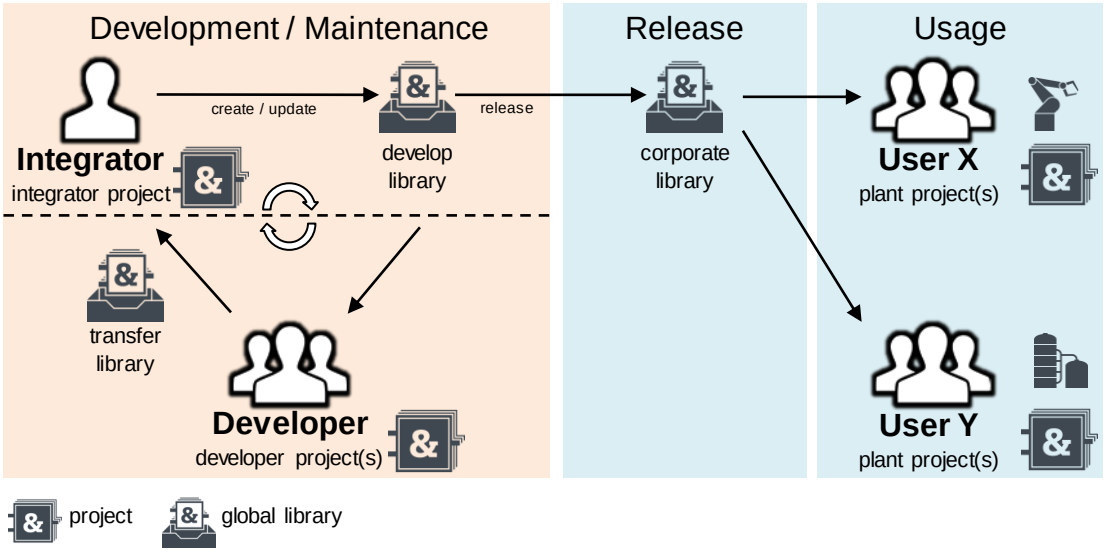
- "Creation/maintenance"
- "Publication"
- "Application"

This procedure is the basis for the creation of corporate libraries.

2.3 "Creation/maintenance" process

The "creation/maintenance" process describes how a "corporate library" (in a developer team) is created and/or updated. For this purpose, the definition of different roles, projects and libraries is necessary.

Figure 2-1: Workflow



2.3.1 "Integrator" role

The "integrator" is the head of the developer team. He/she is the central contact person for the "developer" and has the job to manage the "creation/maintenance" of the "development library". He/she brings together the delivered "transfer libraries" of the "developers" to an "integrator project". He/she creates the "developer library" from the "integrator project".

2.3.2 "Developer" role

The developer creates library elements or updates existing ones. Each developer uses its own "developer project" as development and test environment. Once the developer has completed the library element, he/she will transfer it to the "integrator" in a "transfer library". Only the library elements that are to be integrated in the "development library" are included in the "transfer project".

Note If there is only one "developer", he/she will also take on the role as "integrator".

2.3.3 "Integrator project" (area of responsibility of integrator)

The "integrator project" is used as integration environment for the "development library". It has the following features:

- Includes the entirety of all typified library elements of all developers
- Includes a simple call environment for the typified library elements
- May include older versions of library elements

2.3.4 "Development library" (area of responsibility of integrator)

The "integrator" manages the "development library". It is the basis for all “developers" and can run through an arbitrary number of development cycles that are controlled by the "integrator".

The "development library" has the following features:

- Global library
- It has a total version in which the meta data is entered
- It includes library elements with the following features
  - only the most current release version of all "developers"
  - Meta data (title, comment…) is maintained
  - Change history (new, changed, deleted library elements)
  - Handling instruction for "developer", if library elements have been deleted

2.3.5 "Developer project(s)" (area of responsibility of developer)

The "developer project" is the test environment of the "developers" in order to create or update library elements. It has the following features:

- Each "developer" uses its own "developer project".
- The "developer projects" include the test environment for the library elements to be created or to be updated.
- The "developer projects" are not exchanged among the "developers" and the "integrator".

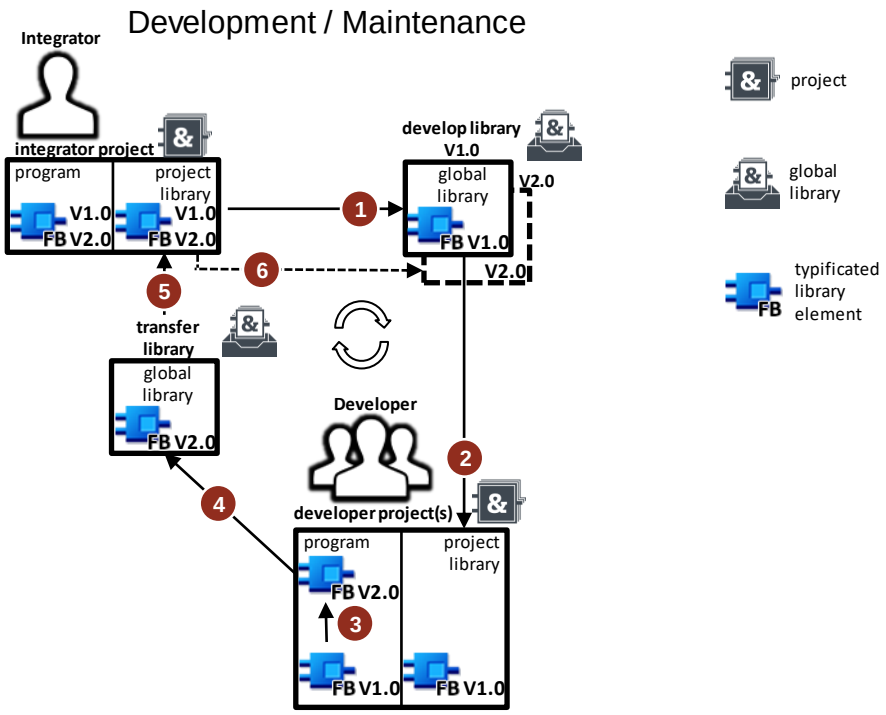
2.3.6 "Transfer library/ies" (area of responsibility of developer)

The "developer" transfers the created library elements to the "integrator" using a "transfer library". It has the following features:

- Global library
- It only contains the typified library element that the "developer" creates or updates.
- The library element is in the "released" status.

2.3.7 "Creation/maintenance" development cycle

Figure 2-2: Development cycle of a library element (here FB)



1. The "integrator" provides the "developers" with the current version of the "development library" as global library.

2. The "developer" updates their "test projects" with the current "development library".

3. The "developers" create or update library elements. The library elements receives a new version.

4. The "developers" supply created or updated library elements in "transfer libraries" to the "integrators".

5. The "integrator" integrates all delivered library element of the "transfer libraries" in the "integrator project".

6. The "integrator" creates the next version of the "development library" from the "integrator project".

The development cycle can be changed as often as desired before the "integrator" performs the "publication" process.

Versioning

When determining the versions of the types, the "integrators" and "developers" have to agree upon an approach. You have to determine who versions the types and how the version numbers are maintained. Further information can be found in chapter 8.6 Version numbers.

Rules for avoiding inconsistencies (area of responsibility of developer)

- Several developers must not edit the same library elements in parallel.
- The "developers" do not exchange "transfer libraries" among each other.
- The "developers" do not transfer "developer projects" to the "integrator". They are only edited by the respective "developer".

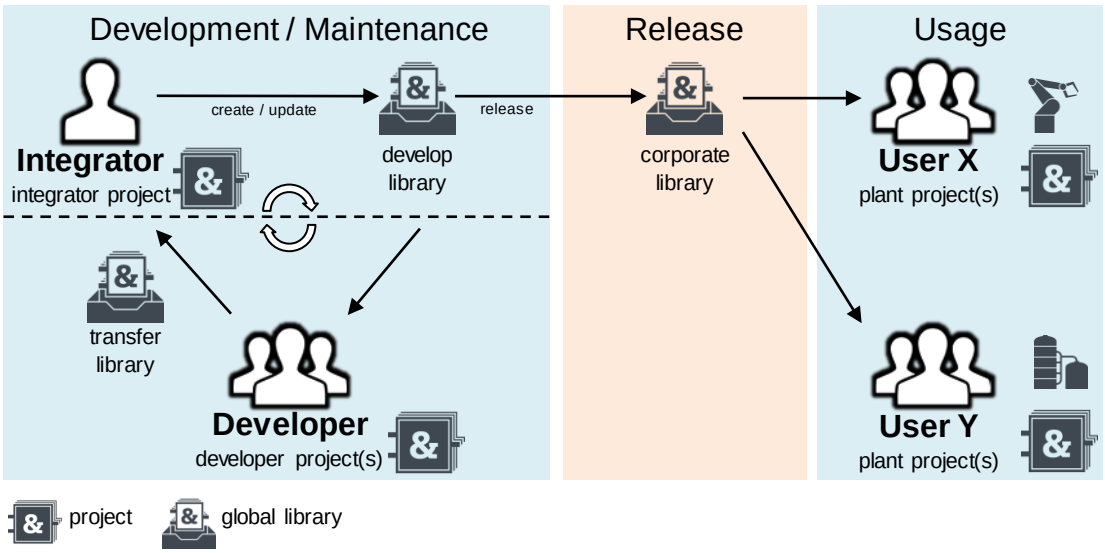
Rules for avoiding inconsistencies (area of responsibility of integrators)

- The "integrator" determines which "developer" edits which library element.

2.4 "Publication" process

The "publication" process describes how the "integrator" publishes a "corporate library".

Figure 2-3: Workflow



2.4.1 "Integrator" role

The "corporate library" corresponds to the "development library" with all its features. The "integrator" specifies a time when the "creation/maintenance" process is completed. The integrator then publishes the "development library" as "corporate library" for the "users".

2.4.2 Corporate library (area of responsibility of integrator)

The "corporate library" is released for the users. It has the following features:

- Copy of the "development library" that the "integrator" declares as "corporate library"
- The meta data (creator, date, version, comment…) of the "corporate library" is updated.
- Published as archived file (e.g. zal14, zip).
- The "integrator" archives all published "corporate libraries" for later traceability.
- Published together with a documentation with the following contents:
  - Block description (block function, interface description…)
  - Change history (new, changed, deleted library elements)
  - Handling instruction for "user", if library elements have been deleted
- The documentation is created by the "integrator" using an editable file format (e.g. docx file).
- The documentation is published by the "integrator" in a non-editable file format (e.g. pdf file).

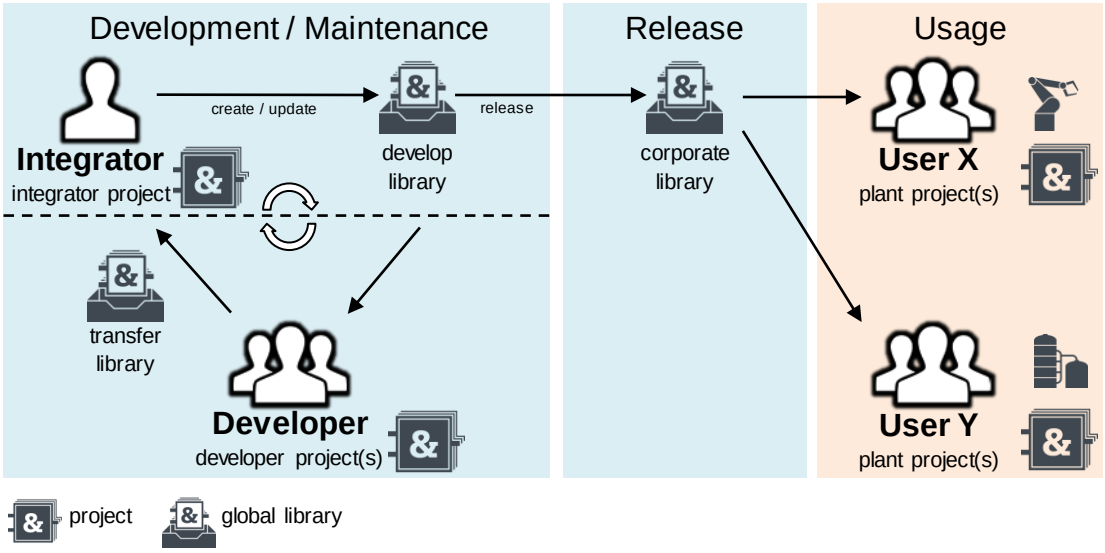
- In TIA Portal there is the option to access a user-defined documentation. You can create descriptions for your library elements that can be displayed by the TIA Portal.

  For more information start the help of the TIA Portal and open the chapter "Editing project data > Providing user-defined documentation".

2.5 "Application" process

The "application" process describes how to use a "corporate library".

Figure 2-4: Workflow



2.5.1 "User" role

"Users" are people that use a "corporate library" only in their "plant projects". They do not change library elements or the "corporate library".

2.5.2 Plant projects (area of responsibility of user)

The "plant project" is a specific project for a machine. It has the following features:

- Includes library elements of "corporate libraries"
- Includes plant-specific elements, which are not located in the corporate library
- Use only one version of a library element in order to keep a good overview and delete the version of the library elements that you do not use.

3 Application Cases in TIA Portal for the "Integrator"

3.1 Working with the "integrator project"

Please observe the following notes when working with the "integrator project":

Note If the library elements have to be available for S7-1200 and S7-1500 PLCs, always use a S7-1200 PLC for their creation.
Note the additional features of know-how protected blocks in chapter 8.7 Know-how protection.

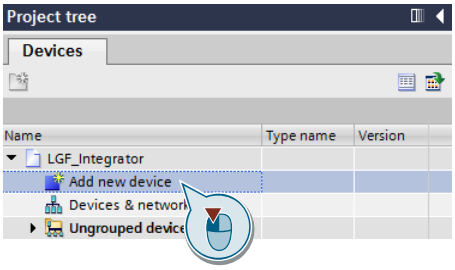
Note If you want to use the library element with PLCSIM Advanced, enable the “Support simulation during block compilation” option box for the project.

Detailed information can be found in the following user application in chapter 2.1 Configuration in the TIA Portal.
https://support.industry.siemens.com/cs/ww/en/view/109739660

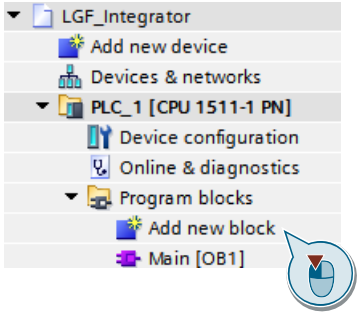
3.1.1 Adding new library elements in the "integration project" from "transfer libraries"

1. If you have no "integration project" yet, create a new TIA Portal project.

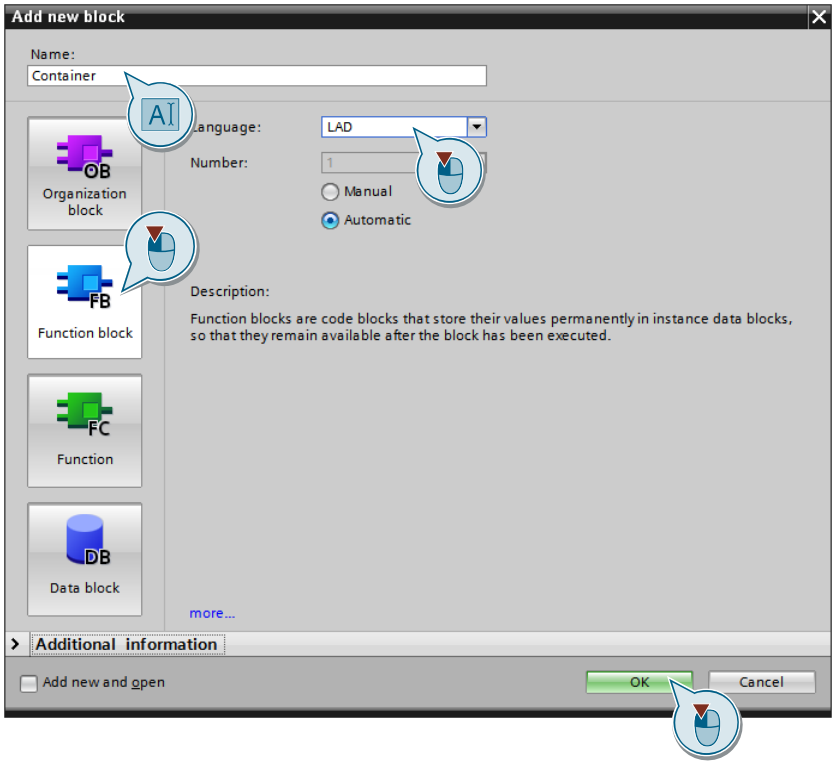
2. Add a device (S7-1200 or S7-1500 PLC) via "Add new device".



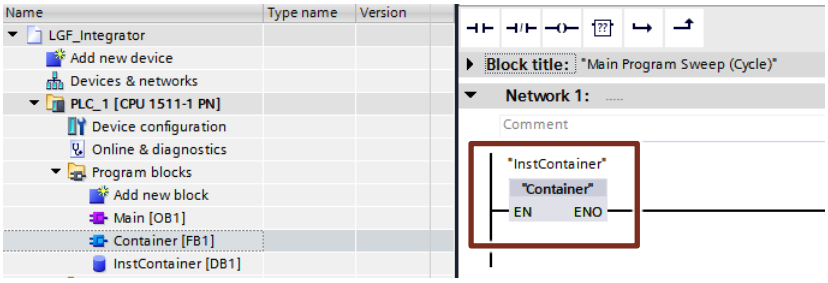
3. Create a new block using "Add new block".



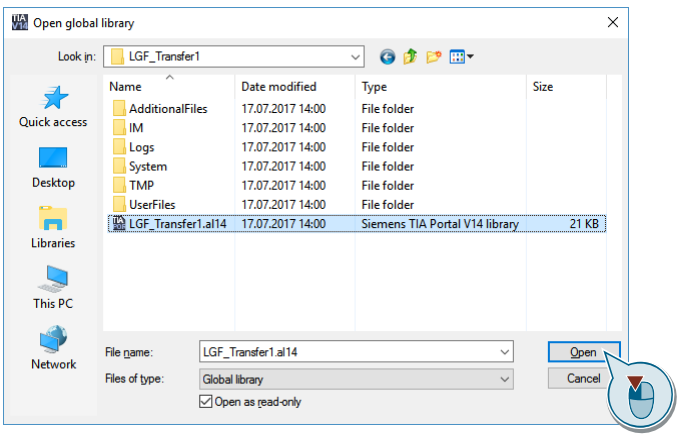
4. Give the FB the name "Container". Use the programming language LAD or FBD. Confirm the dialog using "OK".



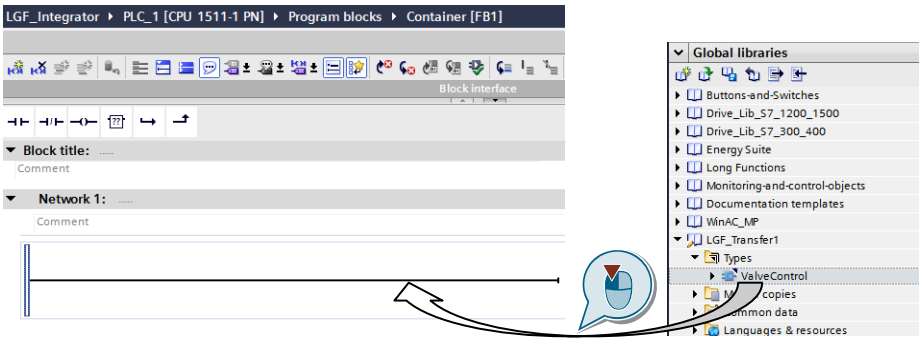
5. Access the "Container" FB in the "Main" OB and specify the name "InstContainer" for the appropriate instance data block.



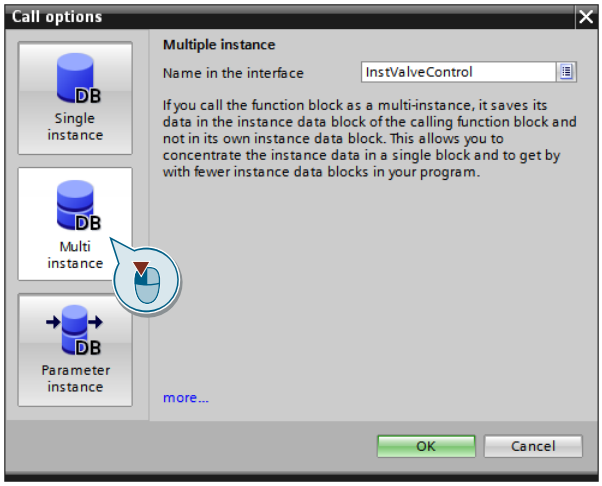
6. Open the "Container" FB by double-clicking on the "Main" OB.
7. Open the "transfer library" (here "LGF_Transfer1") that you received from the "developer".



8. Add the typified blocks from the "Types" folder to the "Container" FB. For a better overview, add an individual network to each block.



9. Define the call of the FB as multi-instances.



10. Create the "Container" block for dummy tag FCs in the temporary or static area. Connect them to the inputs and outputs of the called FCs.

Result

A new type from the "transfer library" is integrated in the project library of the "integration project". All typified blocks are called in the "Container" FB.

Repeat the process until all types are instanced in the project. If new types are added at a later point in time, they have to be instanced the same way.

3.1.2 Updating existing library elements in the "integrator project" from the "transfer libraries"

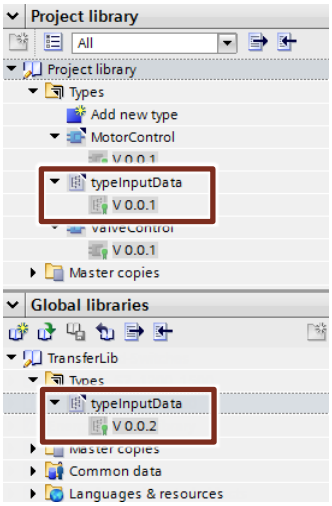
Preparation

Check the type versions in "TransferLib" and the project library in the "integrator project".

1. Open the "integrator project".

2. Open the "transfer library" you received from the "developer" with the updated library elements (here TransferLib).

3. Navigate to the project library of your "integrator project" and compare the versions of the types.



Note If you want to compare the differences of a project library and a global library, you can use the TIA Openness Library Compare tool:

https://support.industry.siemens.com/cs/ww/en/view/109749141

Result

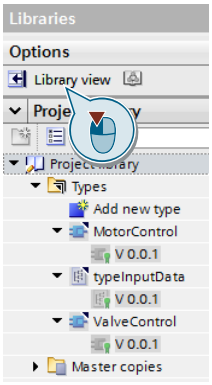
- If the versions of the types are **higher** in der "transfer library" integrate the library elements into your "integrator project".

- If the versions of the types are **lower** in der "transfer library" than the type versions in the "integrator project".
  Check with the "developer" to determine how this problem could occur. The "developer" obviously did not work with the current version of the "development library".

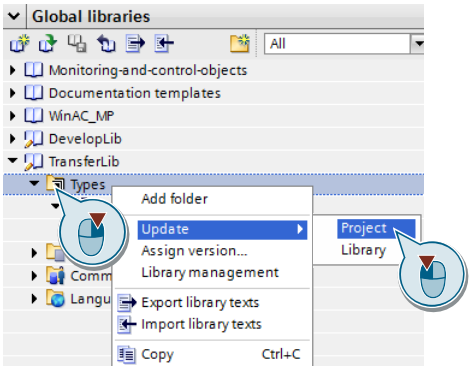
Updating existing library elements from the "transfer libraries" in the "integrator project"

The following step-by-step instruction shows you the path of integration of a subordinate type.

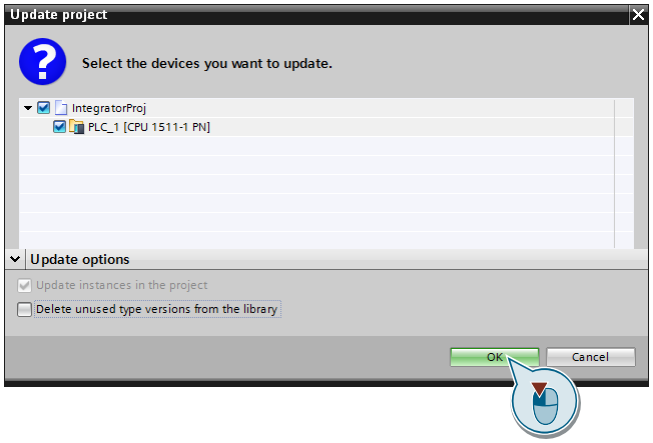
1. Open the "Library view".



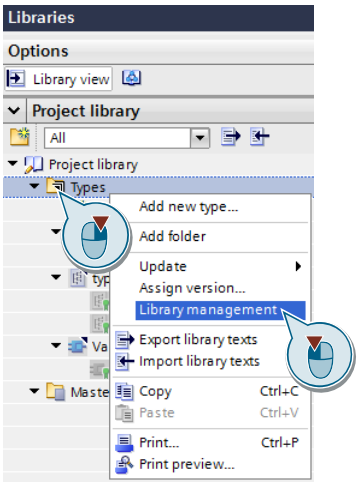
2. In order to integrate the updated types of the global "transfer library" into the "integrator project", right-click on the "Types" folder in the global "TransferLib" and select "Update > Project".



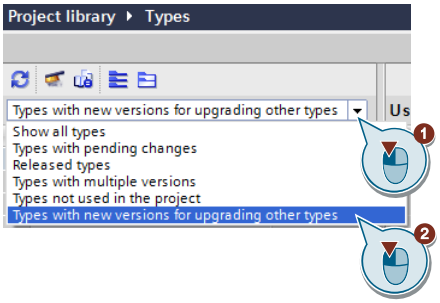
3. Confirm the dialog using "OK". The types that are not used can be deleted in the project library.



4. Right-click on "Types" in the project library and select "Library management".

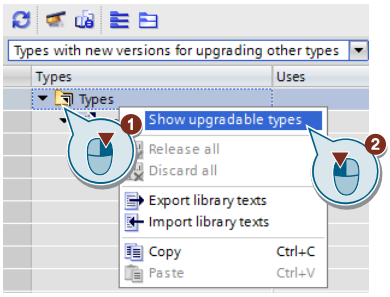


5. Check with the filter dialog whether the types to be updated have higher-level types. To do this, select "Types with new versions for upgrading other types".

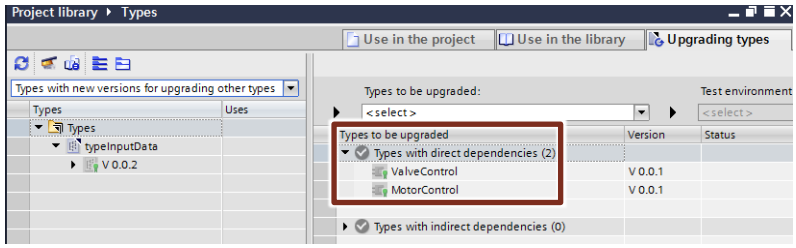


The complete filter dialog is described in chapter 8.5.2 Filter dialog in the library management.
The integration is completed if there are no types with dependencies shown here. This is where you can stop the step-by-step instruction.

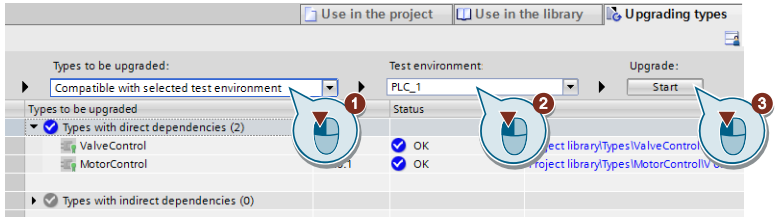
6. Right-click on the "Types" in the library management. And select "Show upgradable types".



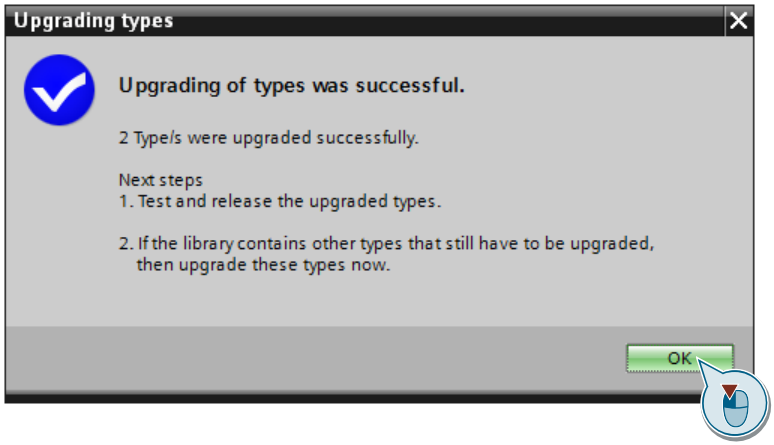
7. All types with the "typeInputDate" type are shown that are in direct or indirect dependency. Now you have the option to update all dependent types.



8. Select "Compatible with selected test environment" and "PLC_1" and click on "Start."

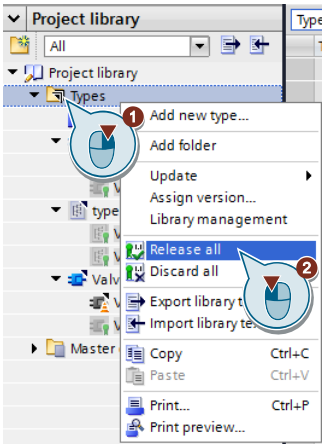


9. Confirm the dialog using "OK".

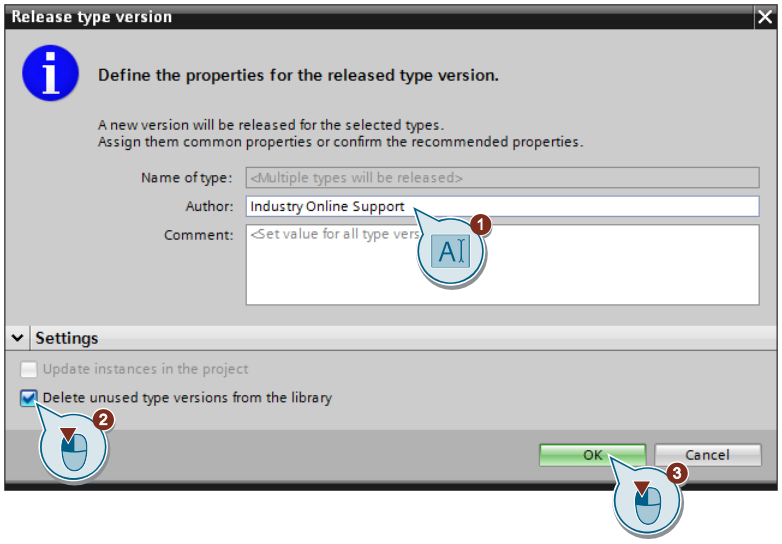


Now you have the option to update other types in the project or to release all types in the status [in test].

10. In order to release all types in status [in test] right-click on the "Types" folder in the project library and select "Release all".



11. Update the meta data, delete the types not used and confirm the dialog using "OK".



Result

The received type from the "transfer library" was integrated in the "integrator project". All overlay types have been updated. All types with older versions are deleted.

Repeat this process until all received types of the "transfer libraries" are integrated in the "integrator project" and all dependent types have been updated.

Recommendation

Use the filter dialogs in the library management check to check whether all types are updated and old types have been deleted.

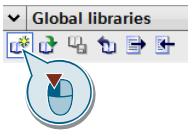
3.1.3 Tips on structuring the container FBs

- Structure the dummy tags for each block in a Struct tag for a better overview.
- Give each network a name of the respectively called block.
- Write meta data (developer, date…) in the network comment.

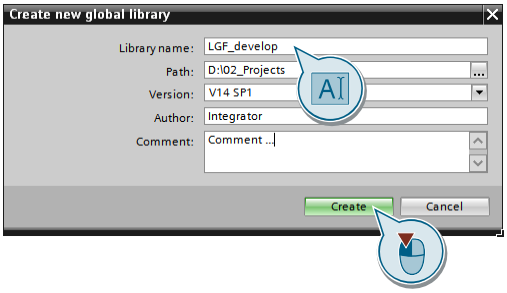
3.2 Working with the global "development library"

3.2.1 Creating the global "development library"

1. Create a global library using the "Create new globally library" button.



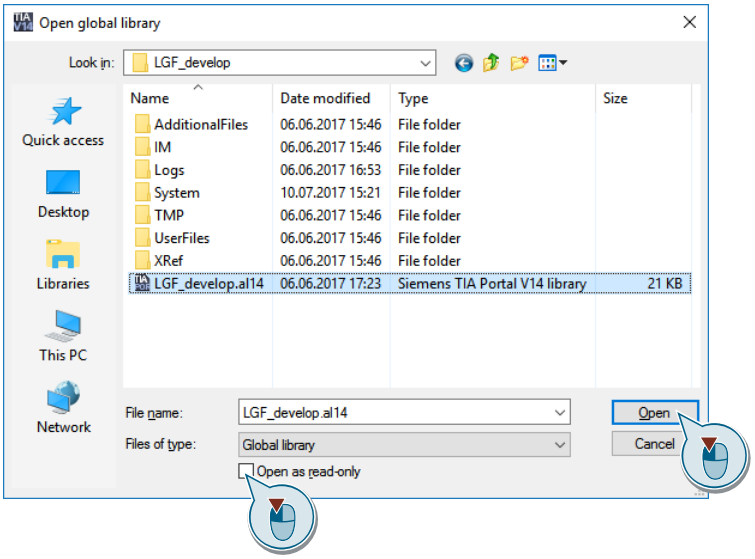
2. Maintain the meta data (creator…) in the dialog and click on "Create".



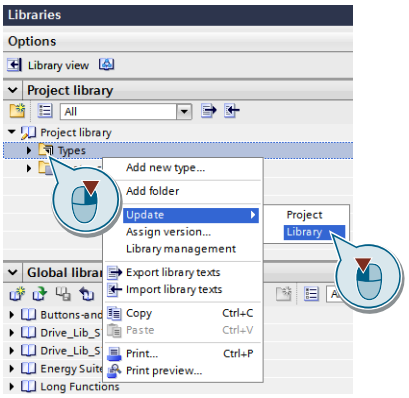
3.2.2 Updating blocks from the "integrator project" in the global "development library"

1. Open the "integrator project".

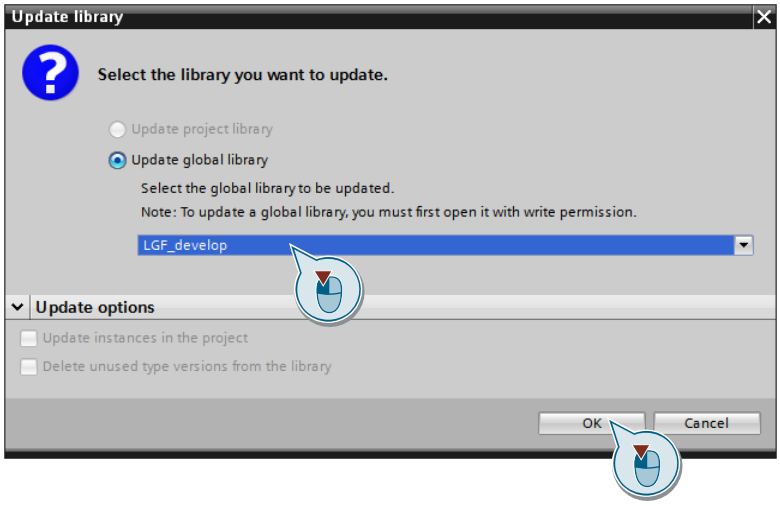
2. Open the global "development library" and untick "Open as read-only".



3. In order to add the types from the "integrator project" in the "development library", right-click on the "Types" folder and select "Update > Library".



4. Select "Update global library" in the dialog and your "development library". Confirm the dialog using "OK".



Result

The "development library" includes all types from the "integrator project". All updated types in the "development library" keep their older versions.

Recommendation

In order to avoid conflicts and to improve a better overview in the "development library", remove all older types, as described in the following chapter. Each library element should only have one version.

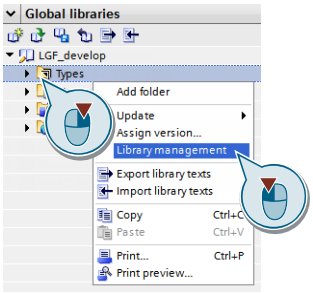
3.2.3 Deleting older type versions in the global "development library"

In order to clean up a global "development library", proceed as follows:

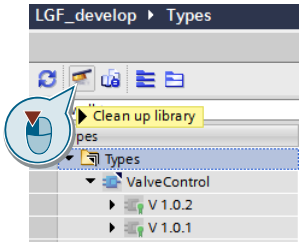
1. Open the "integrator project".

2. Open the global "development library" and untick "Open as read-only".

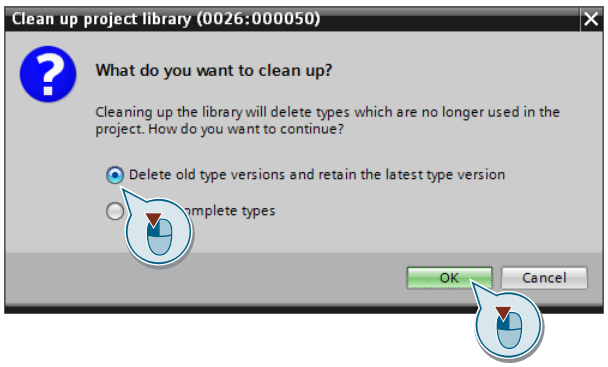
3. Right-click on the "Types" folder in the "development library" and select "Library management".



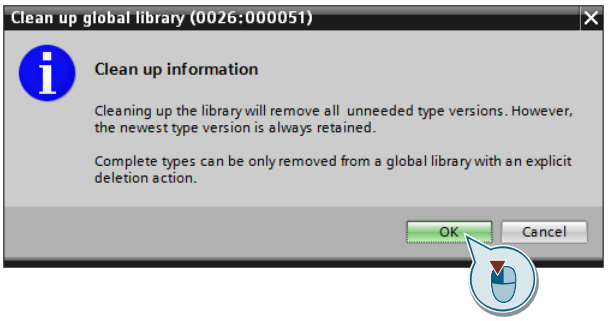
4. In order to only keep the latest versions, click the "Clean up library" button.



5. In the dialog, select "Delete old type versions and retain the latest type version" and confirm using "OK".



6. Confirm the dialog using "OK".



Result

The "development library" only includes the current version of the respective types.

Now you have a defined version of the "development library" again and you have two options.

1. You can distribute the current version of the "development library" to the "developers" for the further editing of library elements.
2. Release the "development library" as "corporate library" for the "users".

Recommendation

- Maintain the meta data (developer, version, comment…) of the "development library".

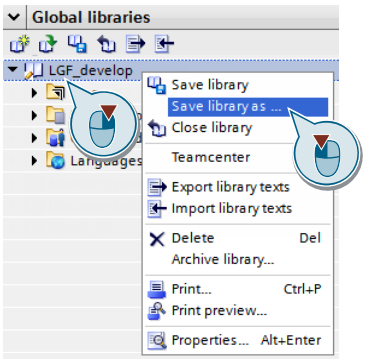
3.3 Working with the global "corporate library"

3.3.1 Creating and releasing the global "corporate library"

The "integrator" creates a copy of the global "development library" and declares it as global "corporate library".

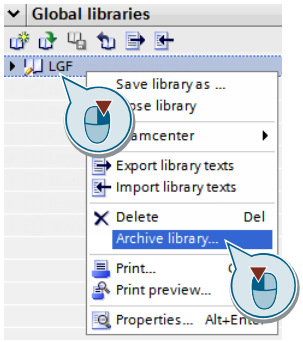
1. Open the global "development library" and untick "Open as read-only".

2. Right-click on your "development library" and select "Save library as …").

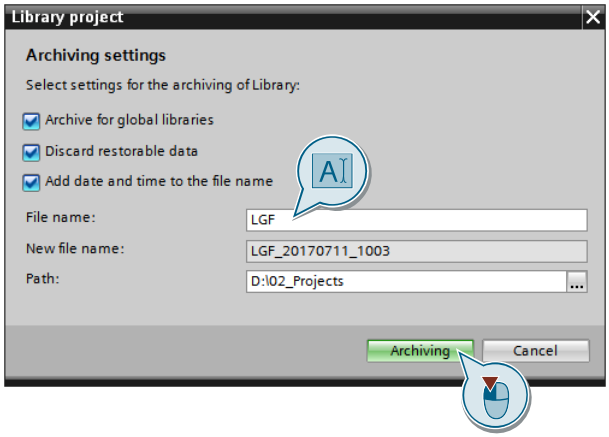


3. Specify a suitable name for the "corporate library".

4. Right-click on the global "corporate library" and select "Archive library…"



5. Set all options, specify the archive name and the storage path. Click on "Archiving".



Result

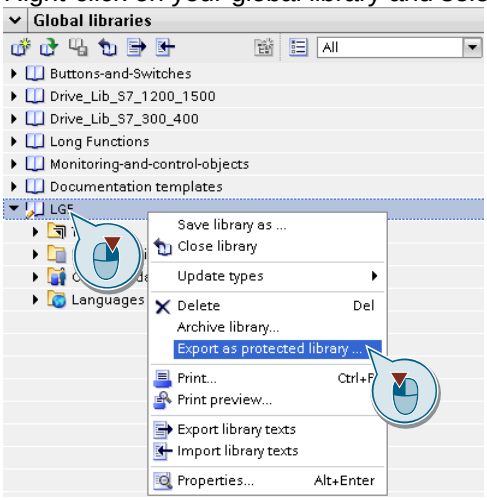
The global "corporate library" is a zipped file and can be published for the "users".

3.3.2 Creating write-protected libraries

You can protect global libraries by exporting them as protected libraries.

1. Open the global library.

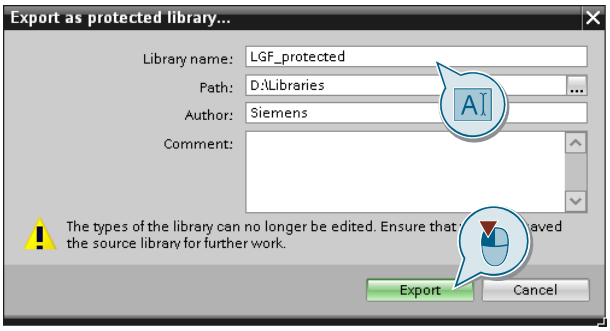
2. Right-click on your global library and select "Export as protected library...".



The dialog "Export as protected library" opens.

3. Enter a suitable library name and save directory for the protected library. Click "Export".

Figure 3-1



Result

The global library is saved in the folder you chose as a protected library and is opened in the "Global libraries" Taskcard.

Properties of write-protected libraries

Write-protected libraries have the following properties:

- Once protected, the write protection on protected global libraries cannot be removed.
- No password is necessary for write protection.
- The global library as well as the types contained within it are protected against modifications.
- When a library is updated from a protected library, the types and type instances in the project are permanently write-protected.

The protected types and type instances of the protected library have the following properties:

- They can be opened in the editor but not modified.
- They cannot be edited.
- Assigning a new version is not possible.
  The update mechanism is retained.
- The connection of an instance to a type cannot be removed.

Note

If you want to deliver write-protected types to customers, try to create write-protected libraries with types in a new, higher version, and then deliver these.

If you create write-protected libraries, make sure not to delete your non-write-protected library.

3.3.3 Discontinuation of types

Library blocks, too, are subject to a lifecycle beginning with planning and development and ending with discontinuation. Blocks may cease to be needed, for instance because:

- The function of the block has been replaced by new system functions (newer firmware).
- The functions of a complex block have been divided among multiple other blocks.
- The block type has changed, e.g. FC → FB.

The behavior has been fundamentally changed/reworked, with the result that one knowingly limits the update potential.

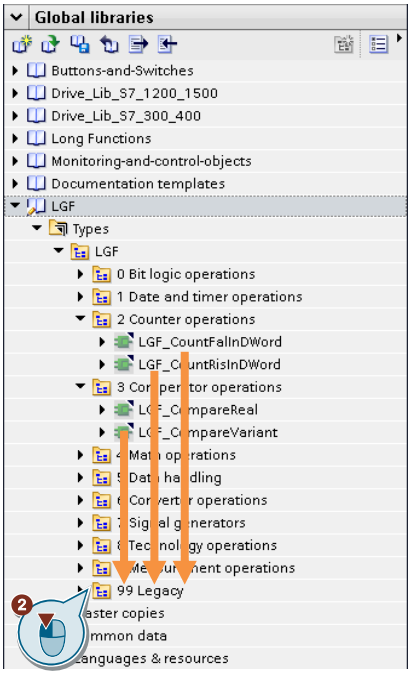
Note

Add the following notes to the blocks and in the documentation:

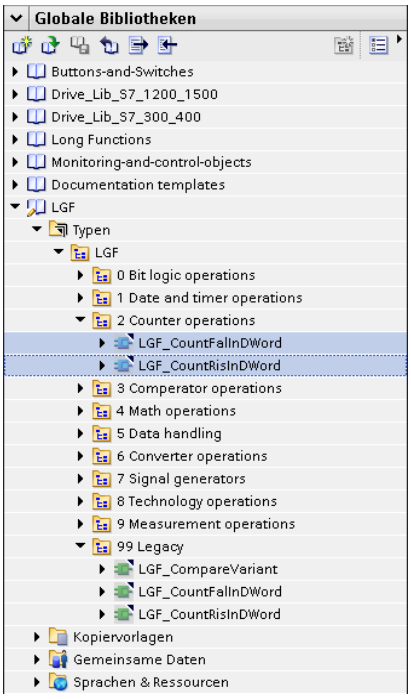
- The block "<block name>" has been moved to the "Legacy" folder.
- The block "<block name>" has been replaced by the "<Block name(s)/system function>" function(s)/system functions

Proceed as follows for blocks that are no longer needed:

1. Open the global library and uncheck the box "Open as read-only".
2. Create a folder in your library with the name "Legacy".
3. Highlight the block or blocks which are no longer needed and move it/them into the "Legacy" folder.



4. If necessary, add the new global library blocks. In the figure below, the FCs "LGF_CountFallnDWord" and "LGF_CountRisInDWord" are replaced with FBs of the same name.



Result

The outdated blocks are saved in the "Legacy" folder of the global library and thus remain in the library. In this way, the user can see that these blocks are no longer current, and can replace them.

Chapter 5.1.5 Replacing discontinued types describes how you can replace discontinued types.

Note So as not to be continually delivering obsolete types, the types in the "Legacy" folder can be deleted where necessary in subsequent releases of the global library.

4 Application Cases in TIA Portal for the "Developer"

4.1 Working with the "developer project"

Please observe the following notes when working with the "developer project":

Note If the library elements have to be available for S7-1200 and S7-1500 PLCs, always use a S7-1200 PLC for their creation.

Note If you want to use the library elements with PLCSIM Advanced, enable the "Support simulation during block compilation" parameter for the project.

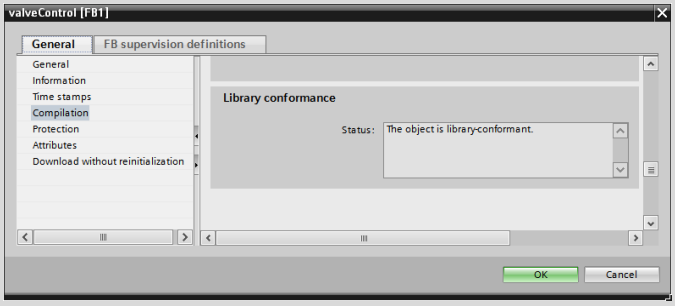
Detailed information can be found in the following application example in chapter 2.1 Configuration in the TIA Portal.

https://support.industry.siemens.com/cs/ww/en/view/109739660

Note The block that is to be created as type has to be compliant with the library.

The block includes…

- no global data access or individual instance call of an instance data block.
- no direct accesses to data blocks, PLC tags or PLC constants.



4.1.1 Test environment in the "developer project"

Recommendation

Store your test environment as master copy without types in your "developer project". This is how you can always bring your "developer project" with the types of the current "development library" up to date, using drag-and-drop.

4.1.2 Updating the "developer project" using the global "development library"

For you to always be able to develop the current block, update your "developer project" with the current "development library".

Case 1

You have not made any changes on your types since you have transferred the last "transfer library" to the "integrator".

Carry out the step-by-step instruction in chapter 3.1.2 Updating existing library elements in the "integrator project" from the "transfer libraries".

Use the following terms analogous:

"Transfer library" → "Development library"

"Integrator project" → "Developer project"

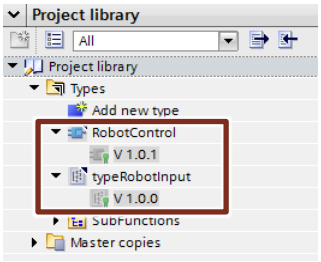
Case 2

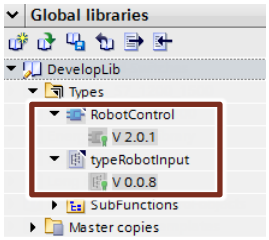
You have made changes on your types since you have transferred the last "transfer library" to the "integrator". Always avoid this case as far as this is possible.

Compare the versions of the types from your "developer project" that you have changed with the "development library". The changes you have made have to be checked using the types in the "development library".

1. Open your "developer project" and the global "development library".

2. Identify the types that you have changed in your "developer project" and in the "development library".

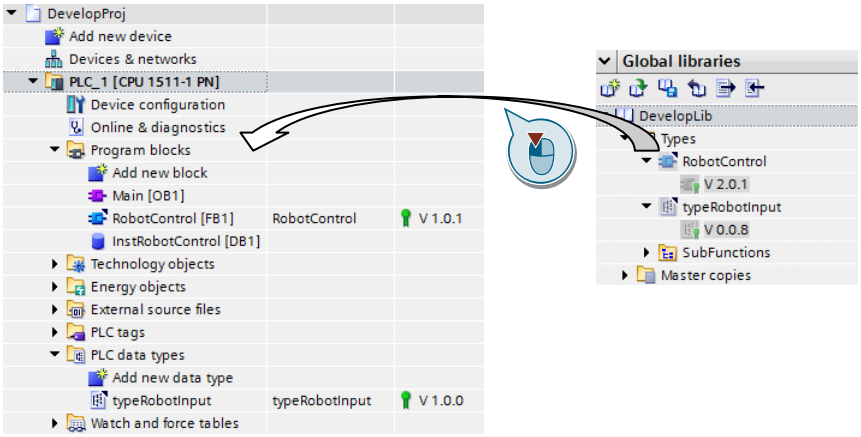




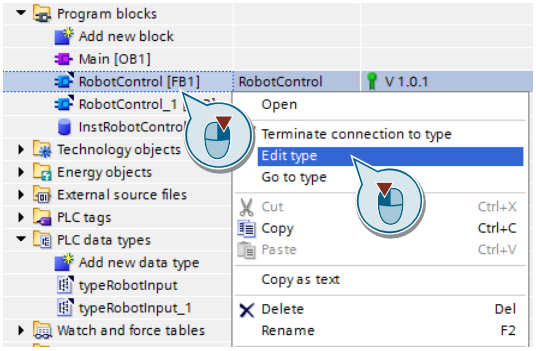
Note If you want to compare the differences of a project library and a global library, you can use the TIA Openness Library Compare tool:

https://support.industry.siemens.com/cs/ww/en/view/109749141

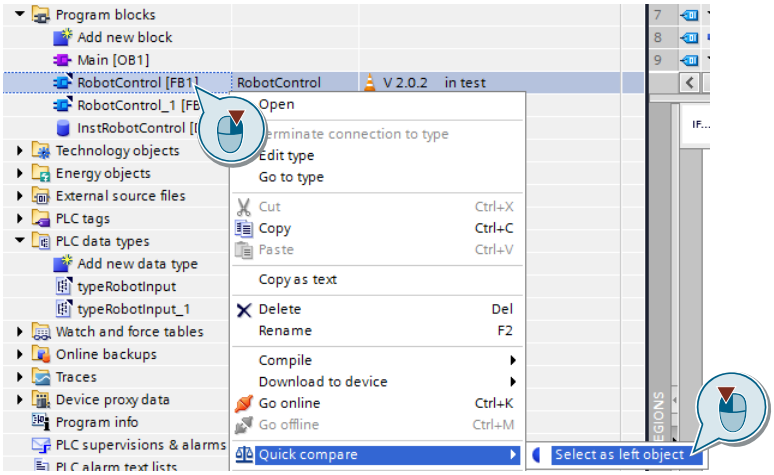
3. Add the newer types from the "development library" to your "developer project" using drag-and-drop.



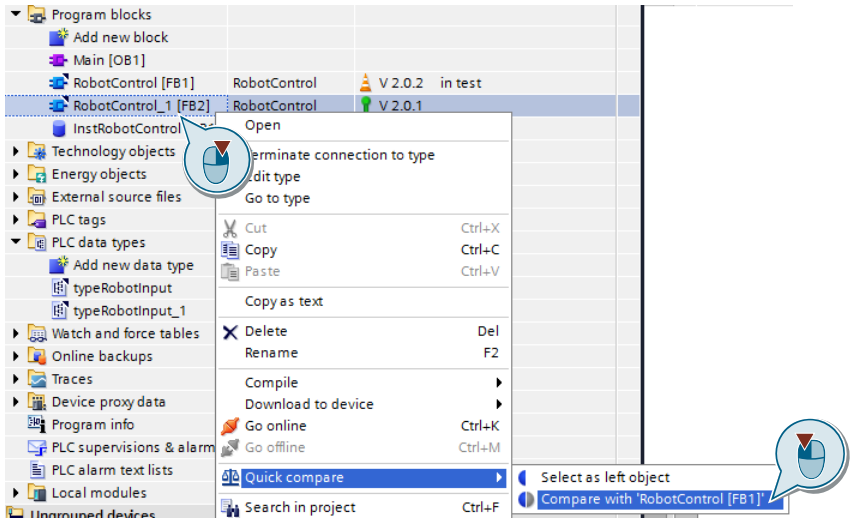
4. Right-click on your type and select "Edit type" in order to apply possible changes in the following steps.



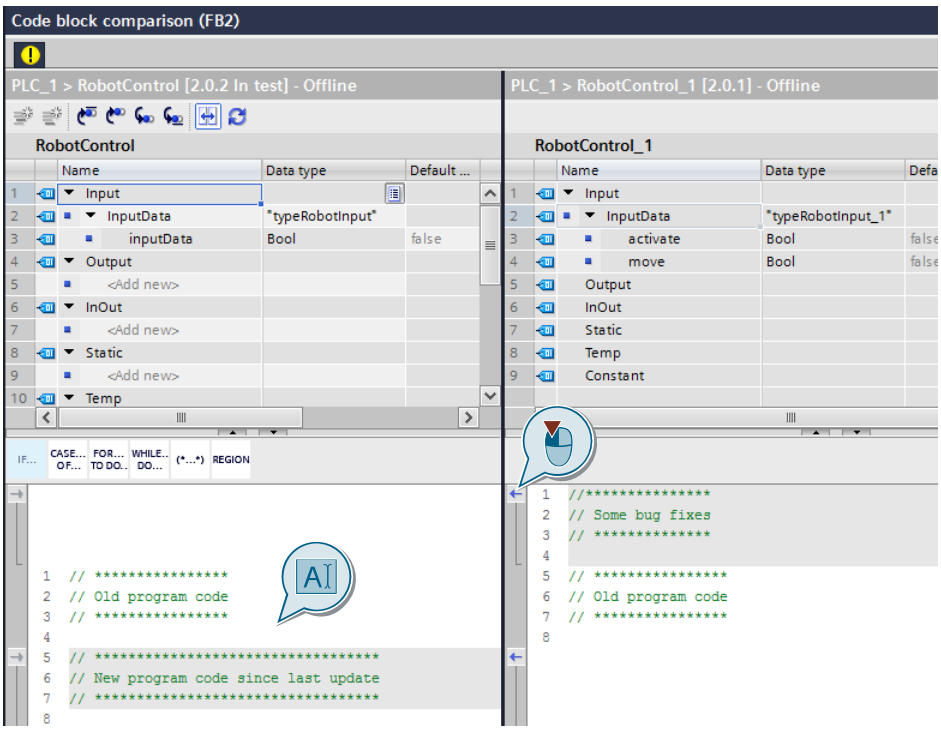
5. Right-click on your type and select "Quick compare > Select as left object".



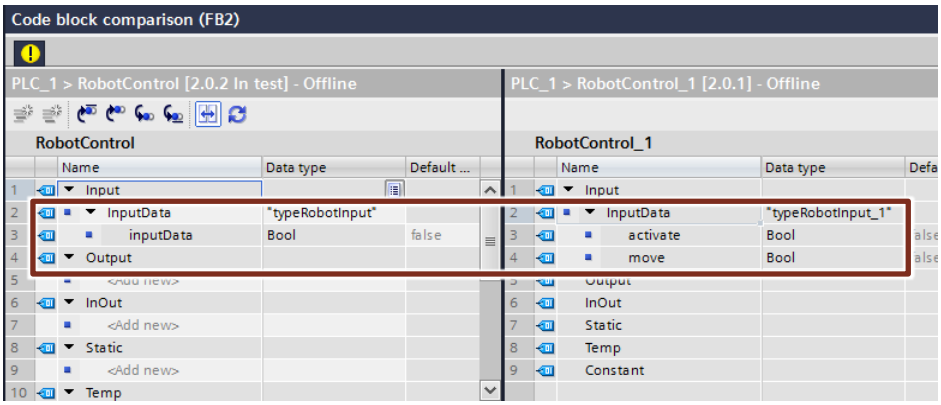
6. Right-click on the type you have added from the "development library" and select "Quick compare > Compare with…".



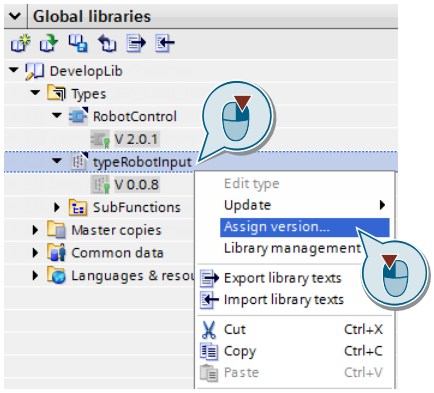
7. Apply the desired changes in the block code.



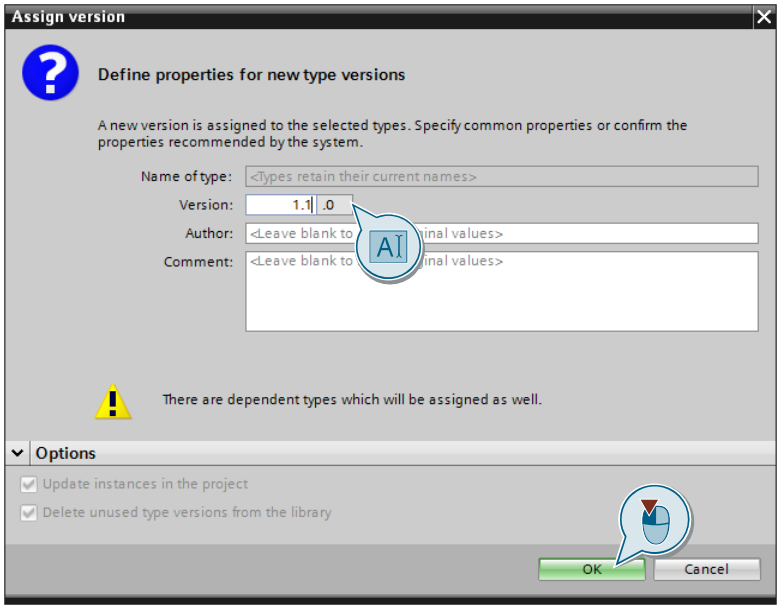
8. Check the changes in the block interface. In this case the overlay type (here "typeRobotInput") has changed.



9. If you want to apply the type from the "development library" and it has a lower version than your type, increase the version in the "development library". If the opposite is the case, proceed with step 11. Right-click on the type in the "development library" and select "Assign Version".

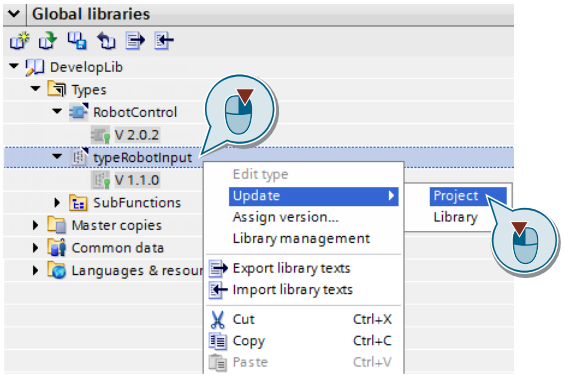


10. Assign a higher version than the type in your "developer project" and confirm the dialog using "OK".

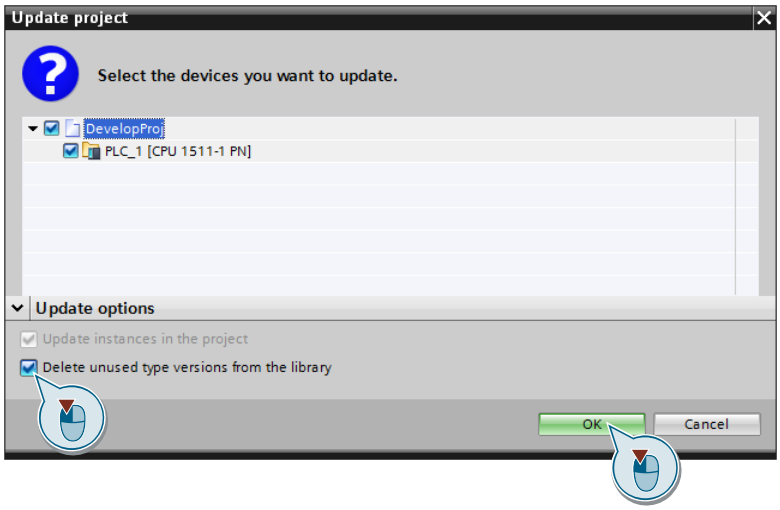


Note: The overlay block in the "development library" also gets a higher version. When all blocks have been compared, close the "development library" without saving it.

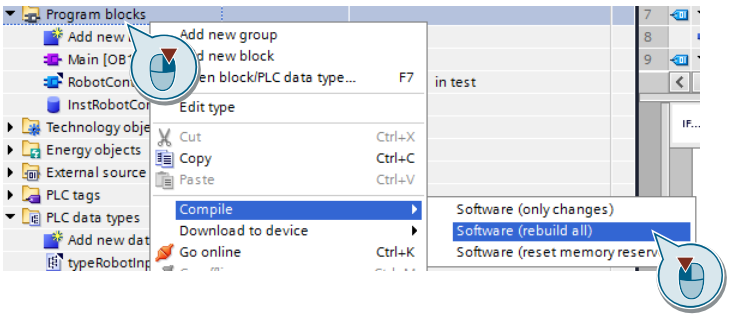
11. In order to integrate the changes in the block interface, right-click only the subordinate type and select "Update > Project" in the "development library".



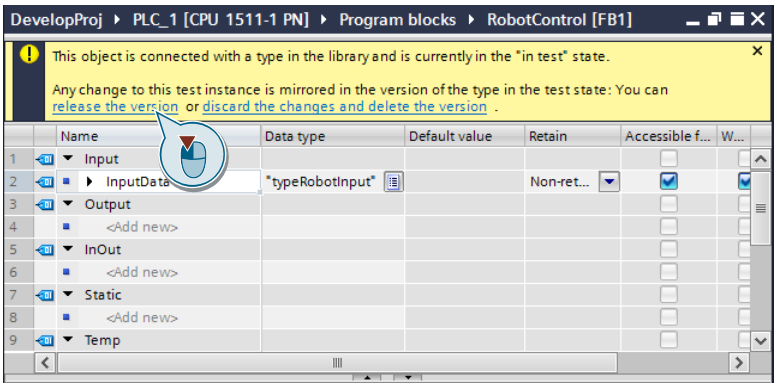
12. Delete all type versions that are not used and confirm the dialog using "OK".



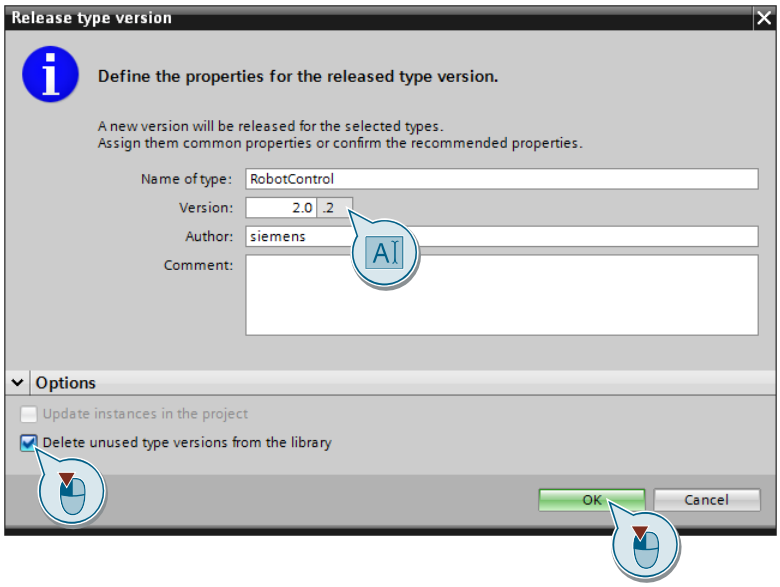
13. Right-click the "Program blocks" folder and select "Compile > Software (rebuild all)".



14. Release the block using "release the version".



15. Assign a version and confirm the dialog using "OK".



16. Perform the steps for all changed blocks.

17. Then delete the older type versions as described in chapter 3.2.3 Deleting older type versions in the global "development library".

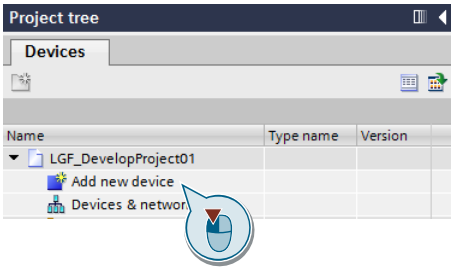
18. Close the "development library" without saving it.

4.1.3 Creating new typified elements

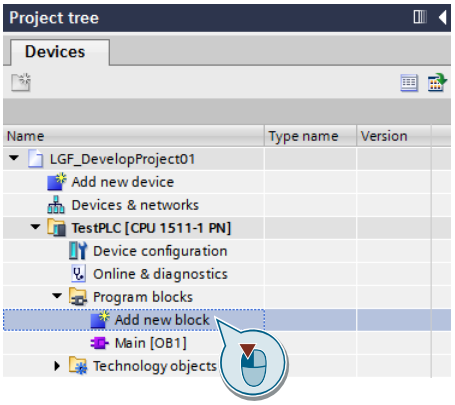
If you want to create new typified elements for the "development library", proceed as follows.

1. Open or create your "developer project".

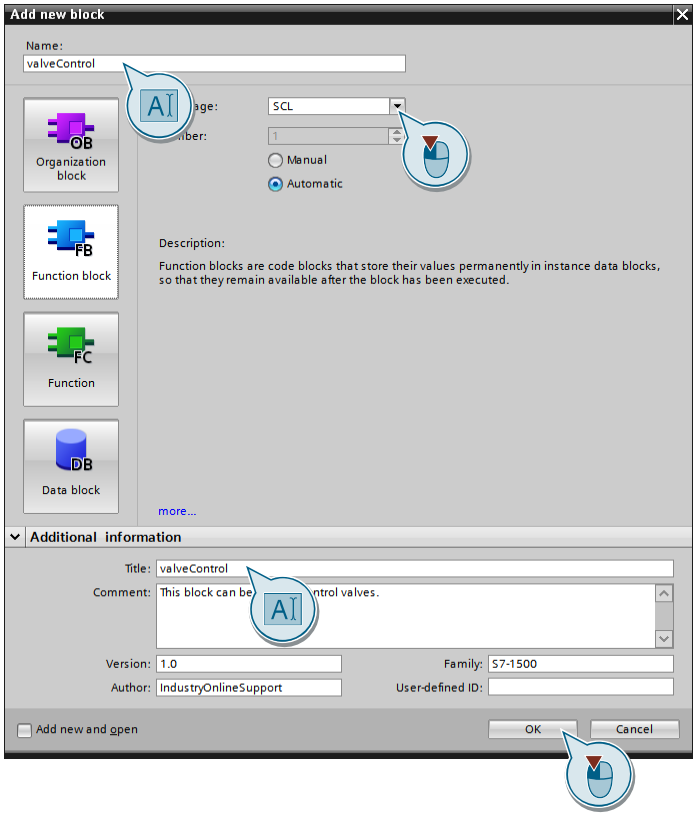
2. Create a test PLC using "Add new Device" with which you want to develop the new block.



3. Create an FC or FB using "Add new block".



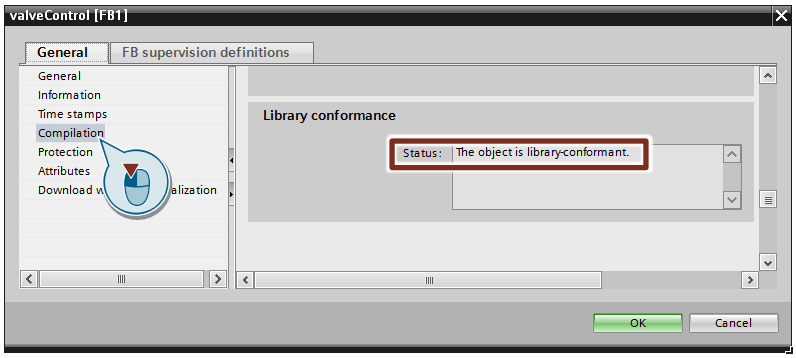
4. Assign a name, select the programming language and maintain the meta data in the dialog in "Additional information".



5. Program the block and test it in a suitable test environment.

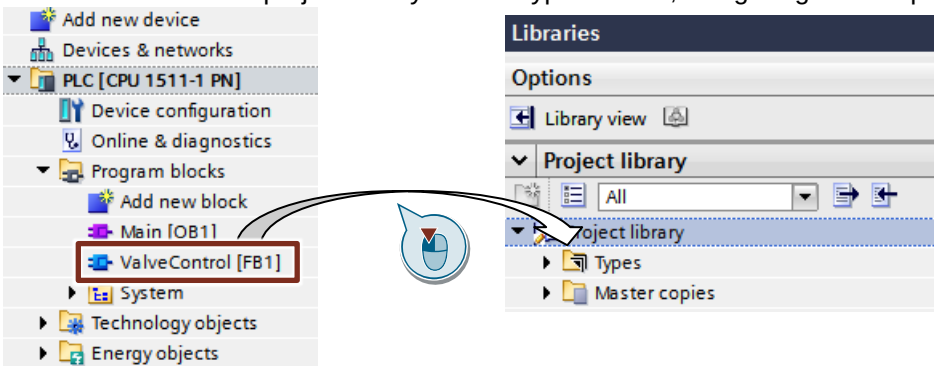
6. Compile the block.

7. Check whether you have programmed the block, compliant with the library. Right-click the block and navigate to "Compilation > Library conformance".

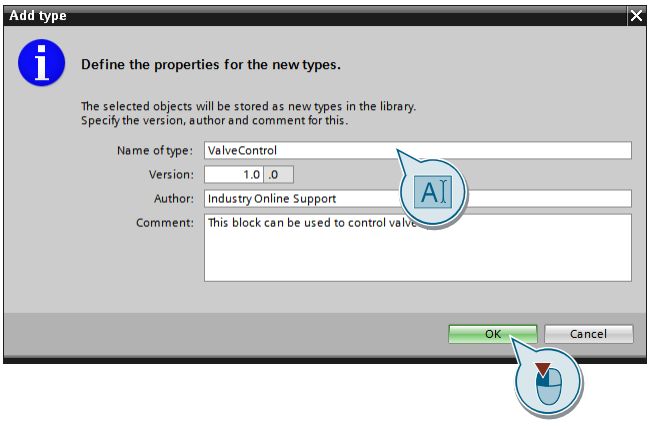


If the block is not compliant with the library, check whether the block is programmed in a modular fashion. It must not directly apply a global constant or a global tag. The data exchange may only take place via the block's interfaces.

8. Move the block in the project library to the "Types" folder, using drag-and-drop.



9. Specify the "Name of type", "Version", "Author" and "Comment" in the dialog. Click on "OK" to close the dialog.



Note Types that are edited, always require a test environment. This means the type has to be instanced in a PLC program

Result

The block is located as type with the "release" status in the project library of your "developer project". If the block has subordinate elements (block call, PLC data types), they are also stored as type with its own version in the project library of their "developer project".
The typified blocks and the subordinate elements are highlighted with a black triangle in the object icon as type.

4.1.4 Editing of typified elements

If you want to edit existing typified elements, proceed as follows.

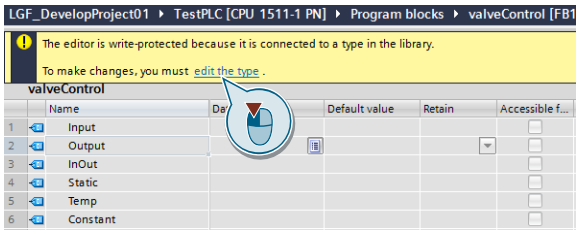
Prerequisite

There is a typified library element in your "developer project".

1. Open your "developer project".

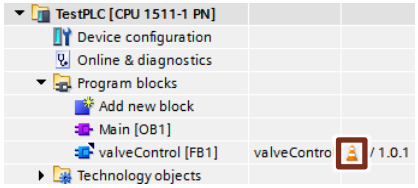
2. Open the typified block.

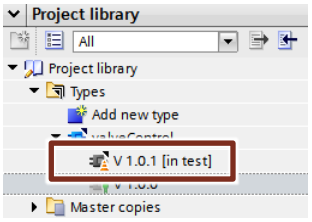
3. Click on "edit the type".



Alternatively, you can also right-click on the block in the project navigation and select "Edit type".

4. The block is in status [in test] (see icon on the block in the program and in the project library).

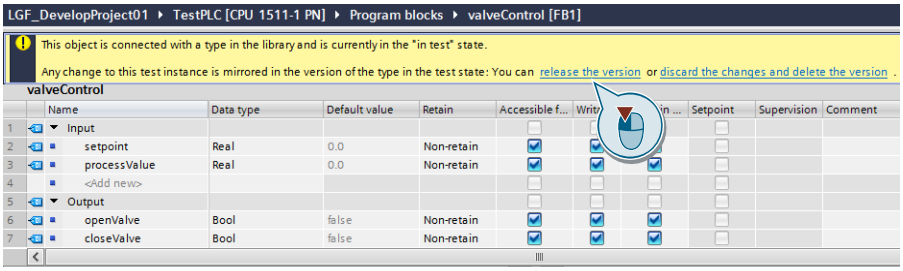




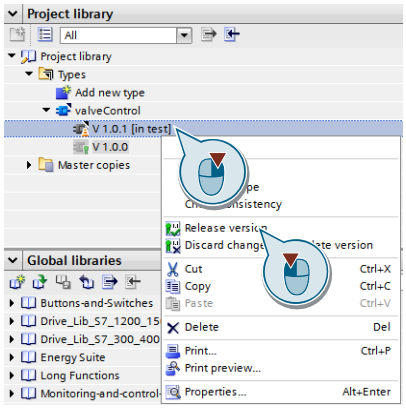
5. Edit the block and retain the status [in test] until it is finished.

Note In the status [in test] you can load and test blocks on SIMATIC S7-1200 and S7-1500 PLCs.

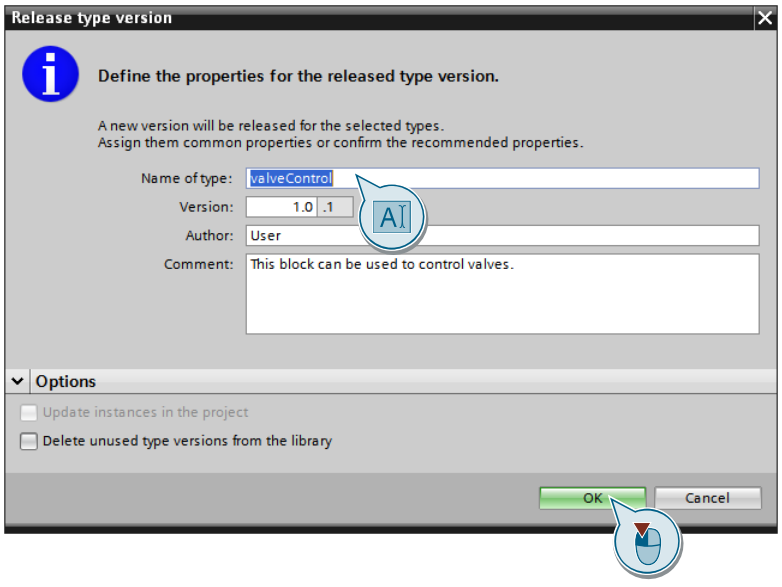
6. Once the block is ready click on "release the version".



Alternatively, you can right-click on the type in the project library and select "Release version".



7. Maintain the meta data in the dialog. Specify the version and optionally select, whether all older type versions are to be deleted in the project library and click on "OK."



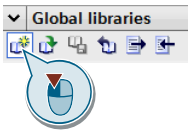
Result

The processed block is located as type with the "release" status in the project library of your "developer project". The version is higher than the original block.

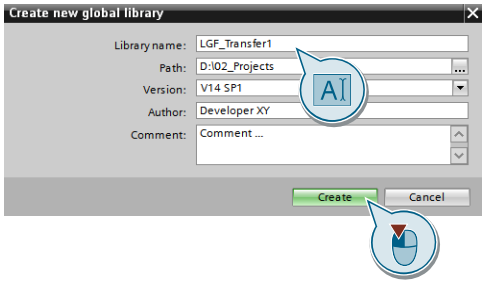
4.2 Working with the global "transfer library"

4.2.1 Creating a global "transfer library"

1. Click the "Create new global library" button in "Global libraries".



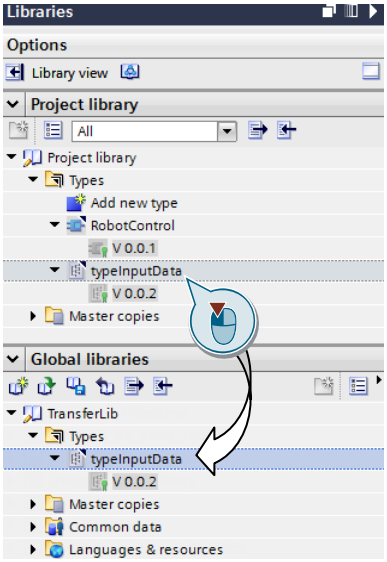
2. Specify the name, path, version, author and comment and click on "Create".



4.2.2 Copying typified elements in global "transfer library"

1. Open your "developer project".

2. Only copy the elements created or updated by you from the project library of your "developer project" into the "Types" folder of your "transfer library" using drag-and-drop.



Result

- The typified block (including subordinate elements) is located in the global "transfer library".
- Repeat the process until all created and changed elements from the project library of your "developer project" have been copied into the "transfer library".
- You can save the global "Transfer library" and supply it to the integrator for integration in the "corporate library".

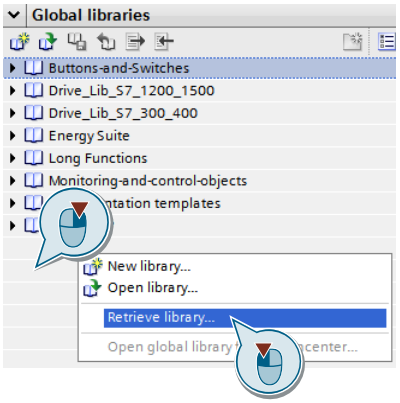
5 Application Cases in TIA Portal for the "User"

5.1 Working with the "plant project"

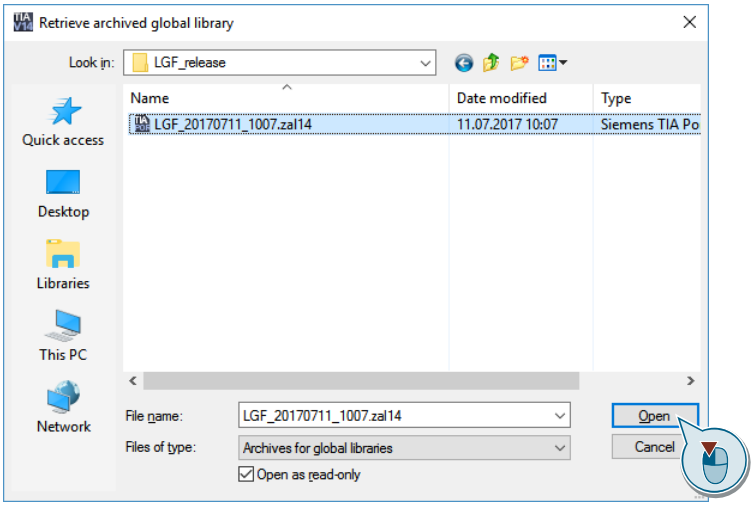
5.1.1 Opening archived global "corporate library"

1. Open your "plant project".

2. Right-click in the free space in "Global libraries" and select "Retrieve library…".



3. Navigate to the archived "corporate library" and click on "Open".

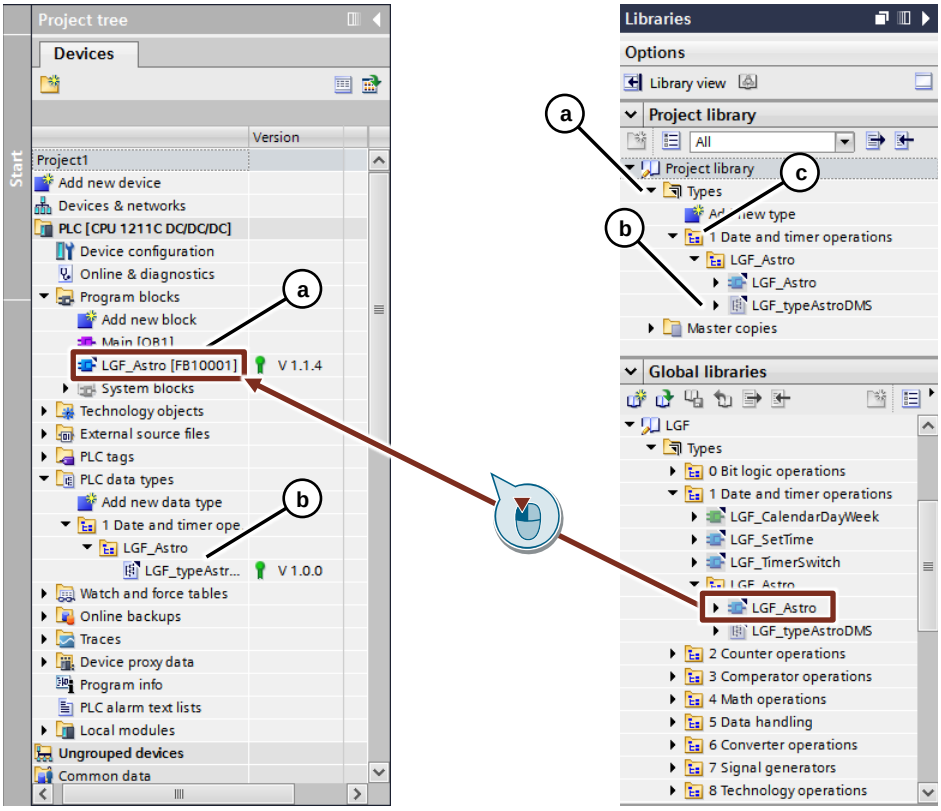


4. Select the storage path for the extracted "corporate library" and confirm the dialog using "OK". The extracted "corporate library" is opened automatically.

5.1.2 Adding typified library elements in "plant project"

1. Open your "plant project" and the global "corporate library".

2. Move the typified element (here FB LGF_Astro) to the "Program blocks" folder.



a. The added block is stored in the PLC program and in the "Project library" in "Types".

b. If a typified library element has a subordinate type, it is also instanced (here: "LGF_typeAstroDMS" is a subordinate type of "LGF_Astro").

c. If a typified library element is located in a folder structure in the global "corporate library", it is applied in the project library.

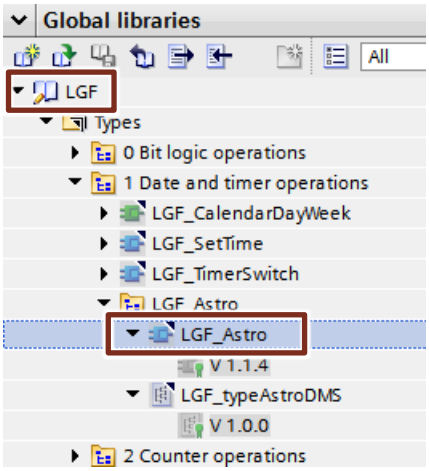
5.1.3 Updating selected types in the "plant project"

Prerequisite

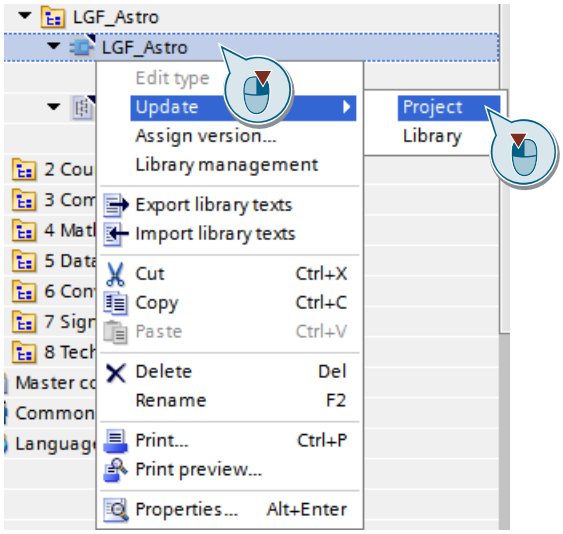
Your "plant project" has to contain typified blocks and the global "corporate library" has to contain a later version of the typified blocks.

1. Open your "plant project" and the global "corporate library".

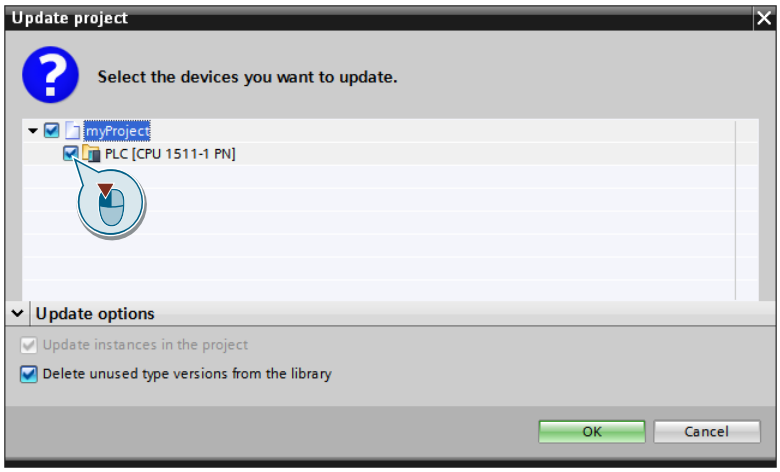
2. Navigate to the typified element in global "corporate library" (here "LGF") that is to be updated in your "plant project" (here "LGF_Astro").



3. Right-click on the typified library element and select "Update > Project".

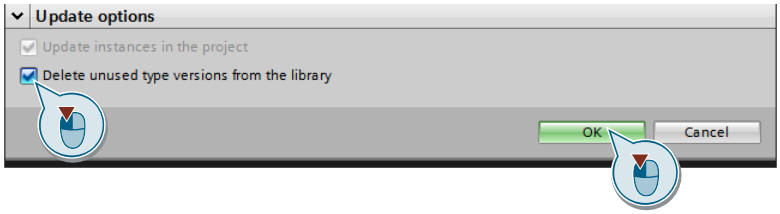


4. Select all PLCs in the dialog, in which the typified library elements are to be updated.



Note: For reasons of consistency, always update all PLCs in the project.

5. Select that (older) type versions from the project library of your "plant project" that are not used, are deleted and confirm the dialog using "OK".



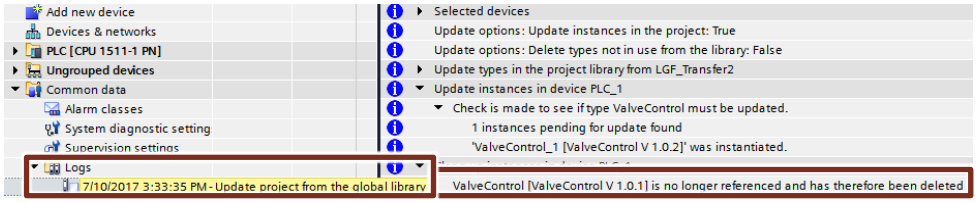
Result

All type instances of the selected library element have been updated in the selected PLCs and in the project library.

Note All typified library elements in the PLCs that are not called or instanced in the program are deleted from the program when the "plant project" is updated.

In "Common data > Logs" you have the option to trace all processes that are carried out during an update.

Figure 5-1: Project logs in TIA Portal



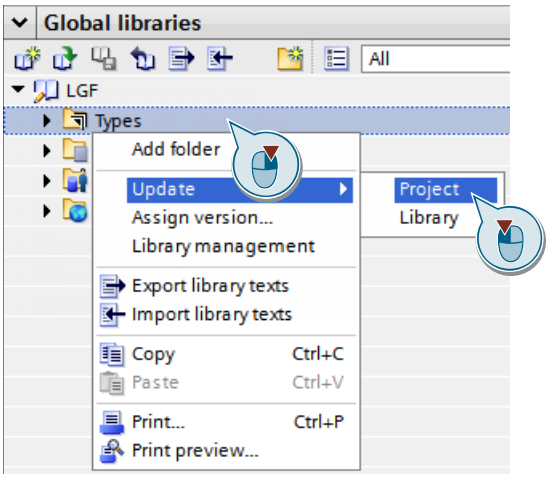
5.1.4 Updating all types in the "plant project"

Prerequisite

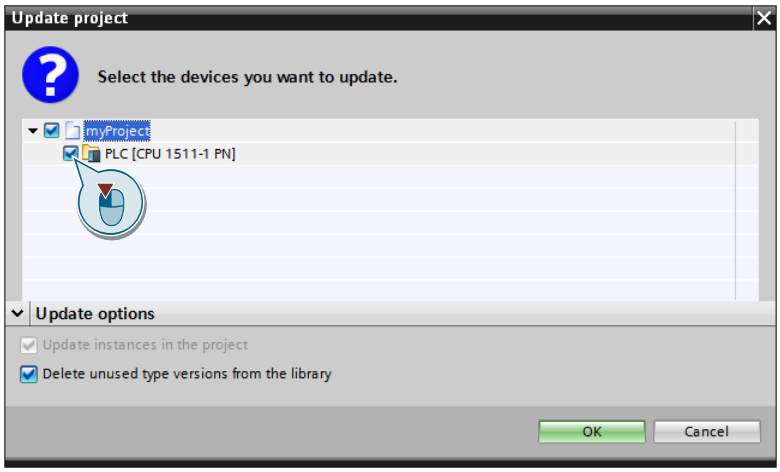
Your "plant project" has to contain typified blocks and the global "corporate library" has to contain a later version of the typified blocks.

1. Open your "plant project" and the global "corporate library".

2. Right-click on the "Types" folder and select "Update > Project".

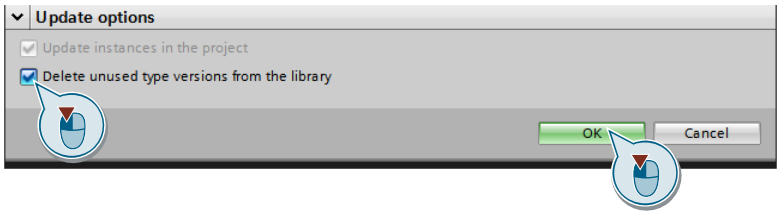


3. Select all PLCs in the dialog, in which the typified elements are to be updated.



Note: For reasons of consistency, always update all PLCs in the project.

4. Select that (older) type versions from the project library of your "plant project" that are not used, are deleted and confirm the dialog using "OK".



Result

All usage locations of all typified elements that are located in the global "corporate library" have been updated in the selected PLCs and in the project library of the "plant project".

Note All typified library elements in the PLCs that are not called or instanced in the program are deleted from the program when the "plant project" is updated.

Note When updating, all types are copied from the global "corporate library" into the project library.

Note If the interface of library blocks is changed, in certain cases (marked blue in Table 5-1) its calls in the program cannot be updated by compiling or "Update inconsistent block calls". Instead, you have to manually update each call via the context menu.

Table 5-1:

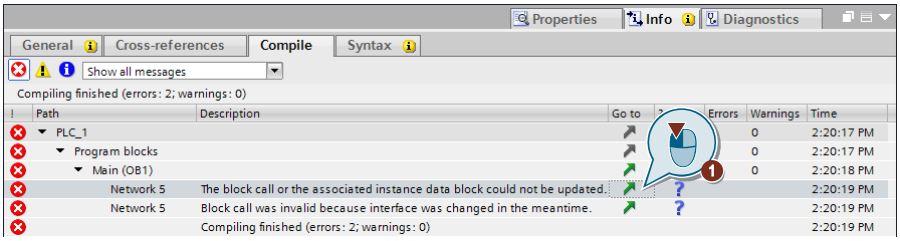
	Update with context menu	Update with button "Update inconsis-tent block calls"	Implicit updating during compilation
Parameters added	X	X	X
Parameters removed Actual parameter not interconnected	X	X	X
Parameters removed Actual parameter interconnected	X	-	-
Parameters renamed Actual parameter not interconnected	X	X	X
Parameters renamed Actual parameter interconnected	X	-	-

Procedure for changes at the interface of library blocks

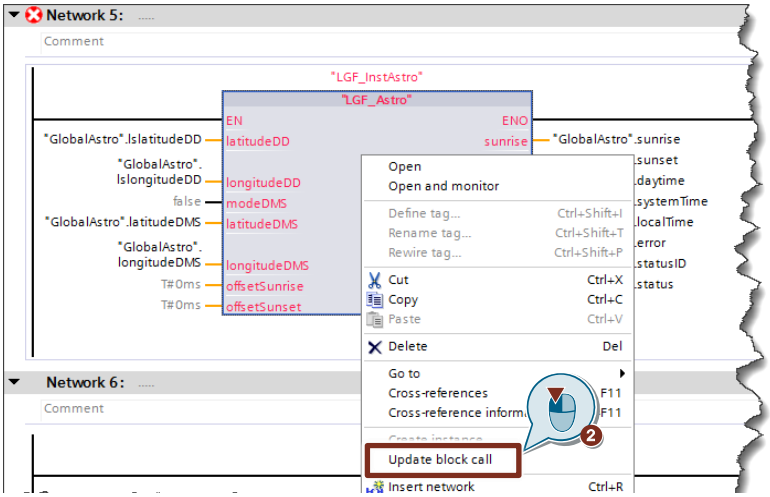
If the interface of library blocks has been changed, in certain cases the plant project cannot be compiled without errors after the update. Each call must be updated manually via the context menu.

To do so proceed as follows:

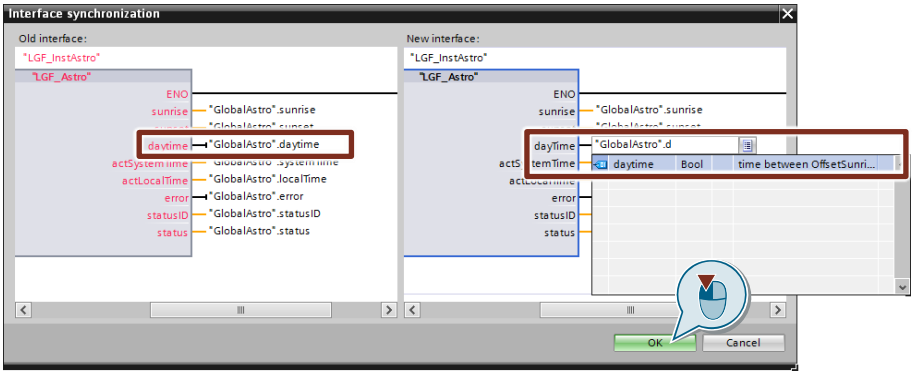
1. Compile your project. To open the call location of the library block, click on the "Go to" button next to the error message.



2. Right-click on the block call and select "Update block call".



3. The following dialog shows the interface synchronization between obsolete and new block. Check the interconnected formal parameters and correct them directly if necessary, e.g. by intellisense (see figure). The variable can also be moved from the left to the right-side using copy & paste or drag & drop. Confirm the dialog with "OK".



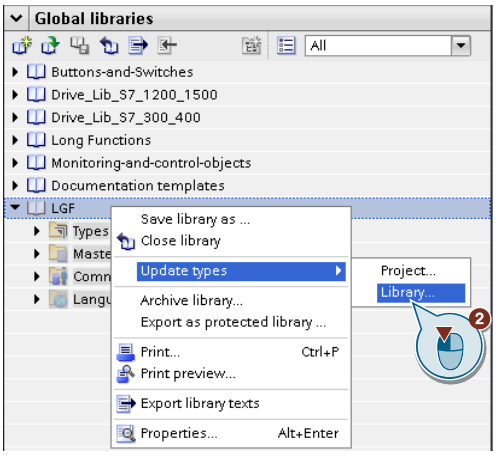
The block call is updated.

5.1.5 Replacing discontinued types

Proceed as follows to replace discontinued typed blocks:

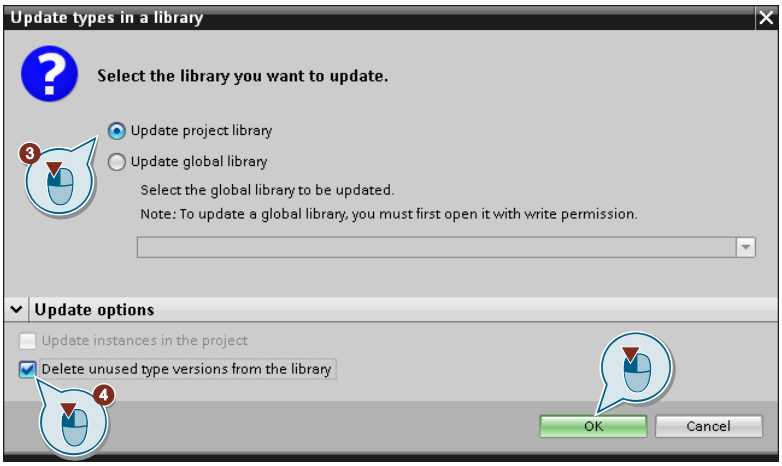
1. Open your "plant project" and the global "corporate library".

2. Right-click on the "Types" folder in the global library and select "Update types > Library…".

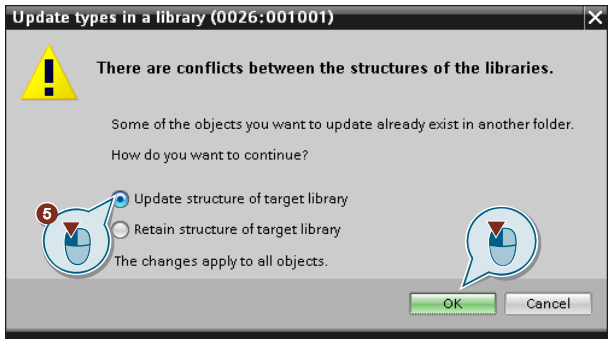


3. Choose from the "Update project library" dialog.

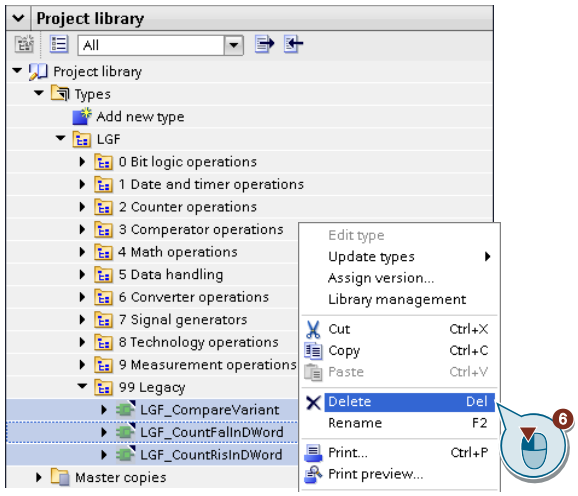
4. Select "Delete unused type versions from the library" to delete unused type versions from the project library of your "plant project". Confirm the dialog with "OK".



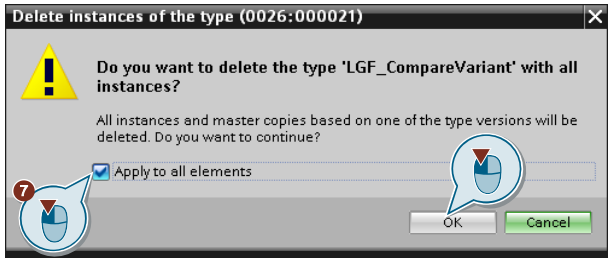
5. Select "Update structure of target library" to conform the structure of the project library to the structure of the global “corporate library". Confirm the dialog with "OK".



6. Delete the obsolete blocks in the "Legacy" folder. Confirm deletion in the next dialog.



7. In the next dialog, select "Apply to all elements". Confirm the dialog with "OK".



Result

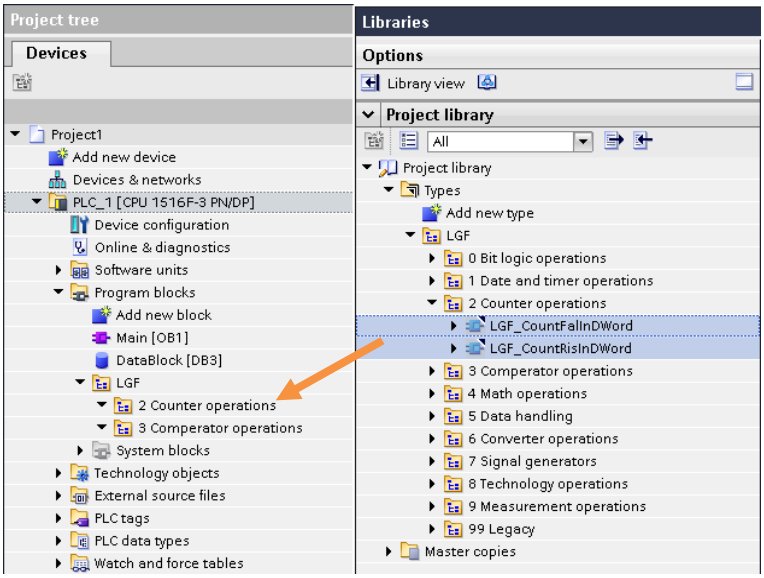
The new blocks are transferred from the global “corporate library" to the project library.

The obsolete typed blocks and all instances are deleted from the project library and the "plant project".

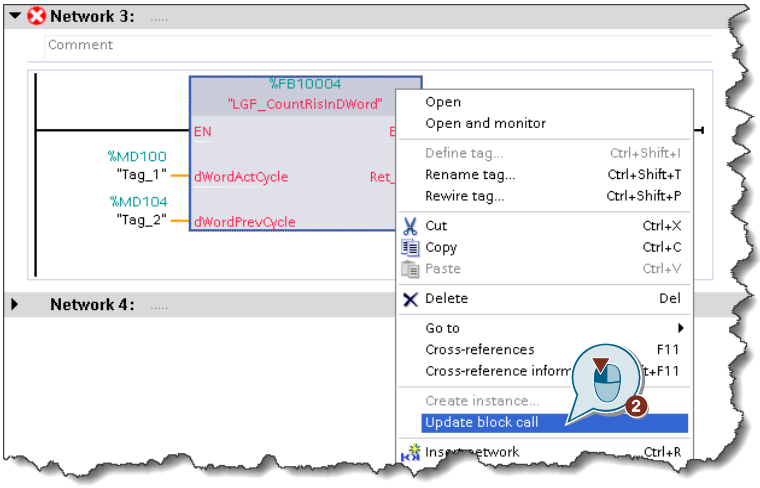
Example 1: Replacing obsolete blocks in a project with a new type

In the example below, the outdated FC "LGF_CountRisInDWord" is replaced by an FB of the same name:

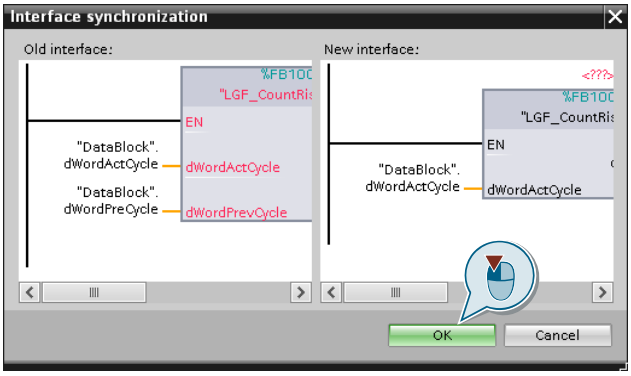
1. Drag and drop the new types from the project library into the "plant project".



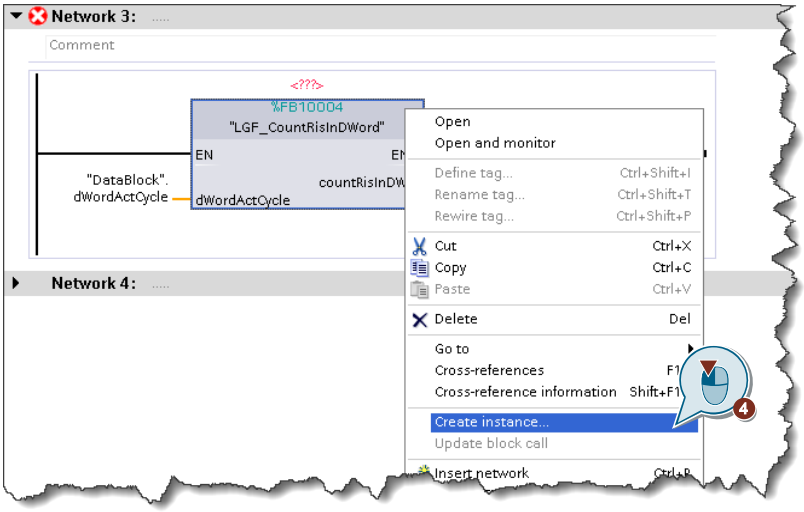
2. Open the point of use in the program, right-click on the block call and select "Update block call".



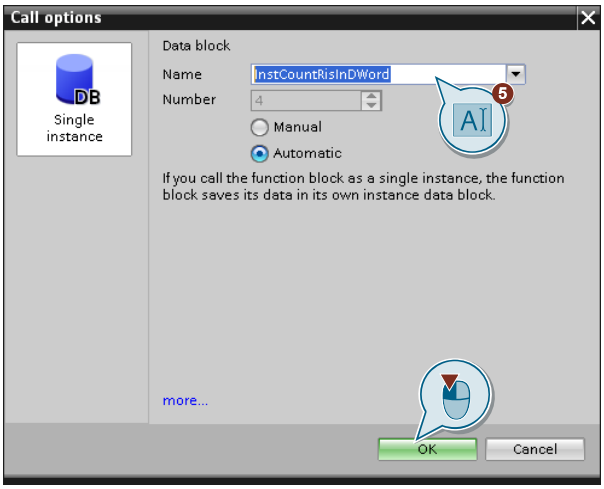
3. The following dialog shows the interface comparison between the old and the new block. Check the wired formal parameters and correct them directly where necessary, e.g. by drag and drop.
Click "OK" to close the dialog.



4. Right-click on the new block call and select "Create instance".



5. Give a new instance name and confirm by clicking "OK".



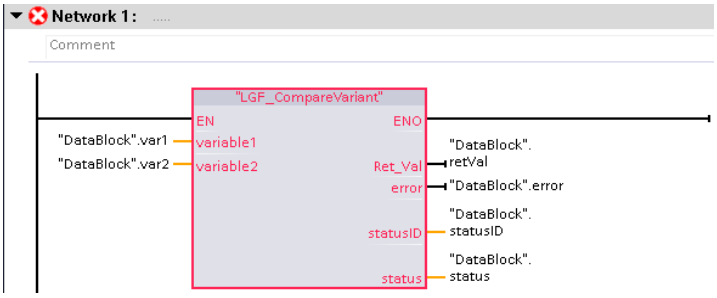
Result

The block call of the old block is replaced by the block call of the new block.

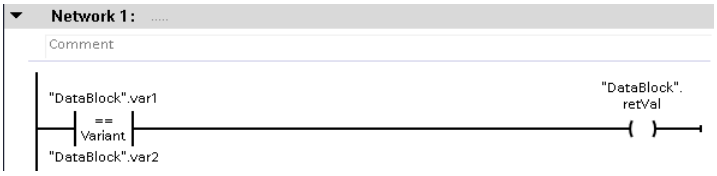
Example 2: Replacing obsolete blocks in a project with new system functions

In the following example, the FC "LGF_CompareVariant" is replaced by a comparator and an assignment.

1. Open the points of use in the program.



2. Replace the block with a comparator and an assignment.



In the SCL programming language you will use the IF instruction for this.

```
IF ("DataBlock".var1 = "DataBlock".var2)
THEN
    "DataBlock".retVal := TRUE;
ELSE
    "DataBlock".retVal := FALSE;
END_IF;
```

6 Creating Individual Libraries

If you want to create an individual global library (company and department-specific) from several global basic libraries, you have to maintain the following procedure in order to avoid inconsistencies and to get a clear structure. The workflow follows the procedure for creating the "corporate libraries".

Example for a basic library:

Library with general functions for (LGF) for STEP 7 (TIA Portal) and S7-1200 / S7-1500
https://support.industry.siemens.com/cs/ww/en/view/109479728

The following terms are adapted for the workflow.

- Basic library
  → corresponds to a global "corporate library"

The following terms get additions:

- "corporate-specific development library"

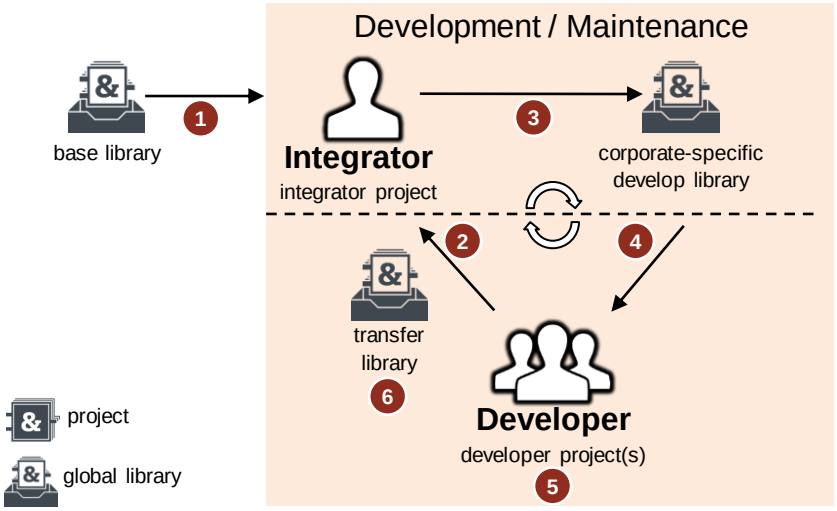
All other original terms (e.g. roles) are used unchanged.

Recommendation

- Use the folder structures for a better overview of your corporate-specific development library
- Create a folder for each basic library in your "corporate-specific development library".

Workflow for the integration of a basic library in a corporate-specific library

Figure 6-1: Workflow



1. The "integrator" integrates all or selected library elements of the "basic library" in the "integrator project". He/she structures all elements in a suitable folder structure.
   → Chapter 3.1 Working with the "integrator project"

2. The "integrator" integrates all delivered library element of the "transfer libraries" in the "integrator project".
   → just as step 1

3. The "integrator" creates the next version of the "corporate-specific development library" from the "integrator project".
   → Chapter 3.2 Working with the global "development library"

4. The "integrator" provides the "developers" with the current version of the "corporate-specific development library" as global library.

5. The "developers" update their "test projects" with the current "corporate-specific development library". They create or update library elements. The library elements receives a new version.
   → Chapter 4.1 Working with the "developer project"

6. The "developers" supply created or updated library elements in "transfer libraries" to the "integrators".
   → Chapter 4.2 Working with the global "transfer library"

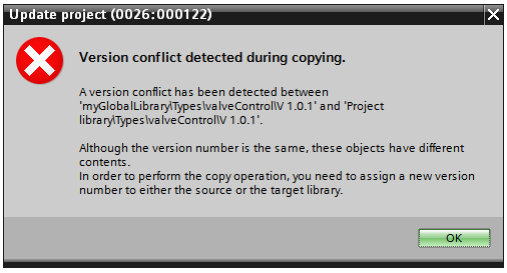
The development cycle can be changed as often as desired.

7 Troubleshooting

7.1 Finding solutions for conflicts with same versions

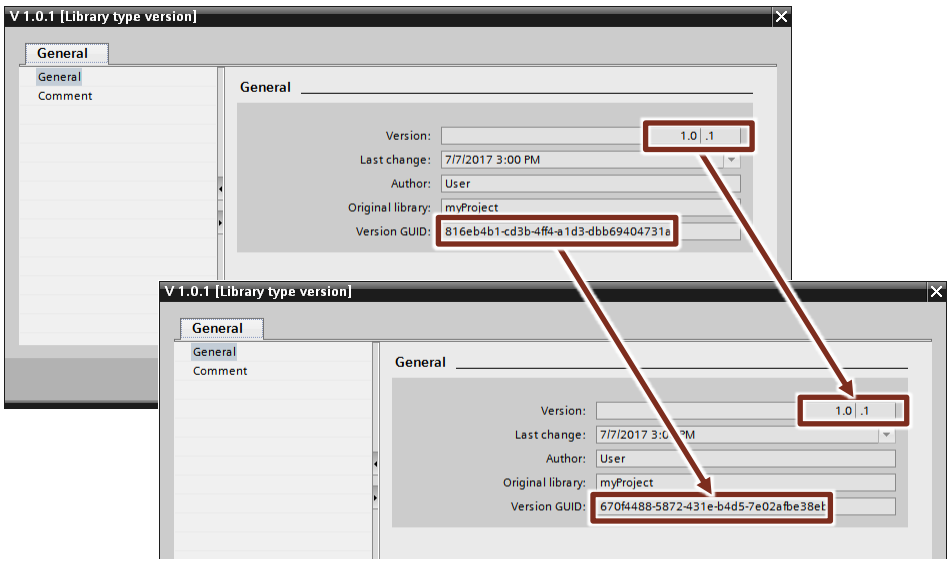
In the following situation there may be version conflicts between typified elements:

- Open a global library und copy a type (V1.0.0) into your project.
- Edit the type and create a new version (V1.0.1).
- Open another global library in which the type has the same version (V1.0.1).
- If you try to integrate the type in your project, the TIA Portal will report the following error.



If you compare the features of the library element in the project with the one in the global library, you will notice that they have the same "Version" but not the same "Version GUID". This means the two library elements are different.

Figure 7-1: Version conflict of two typified library elements



Solution (integrator)

- The "integrator" increases the version of the library element in the project library of its "integrator project" and then in the global "development library" for the "developers".

Solution (developer)

- The "developer" updates the library element in his/her "developer project" with the global "development library" that has been updated by the "integrator" (chapter 4.1.2 Updating the "developer project" using the global "development library").

Solution (user)

- The "user" must not carry out any changes on the types.
- The "user" compares the different types in the project library of the "plant project" in order to determine functional changes. If there are any differences it is recommended to contact the publisher ("integrator") of the "corporate library".
- The "user" deletes the types in the project library of the "plant project" with its instances. The deleted types are replaced with the types of the current "corporate library".

7.2 Closing several versions

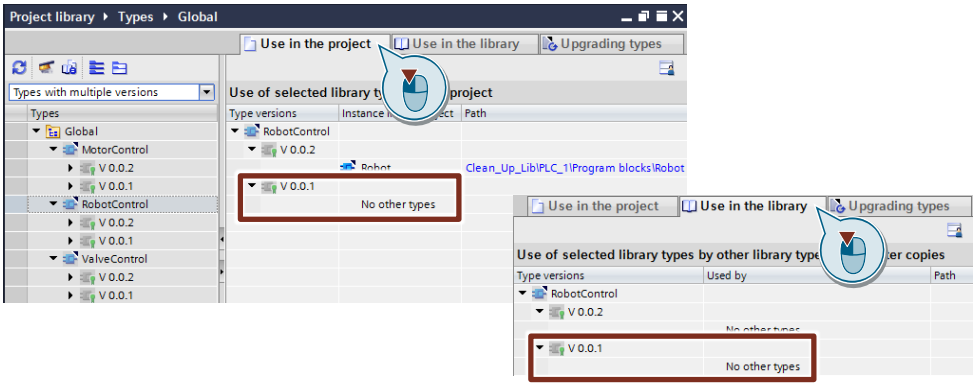
If there are several versions of the same type in project or global libraries, reduce the types to the current version. Use the library management in the library view in order to clean up the library. Before cleaning up, check whether the older versions are used in the project, in the project library or in the global library.

For more information, please refer to chapter 8.5 Library view and library management.

Preparation

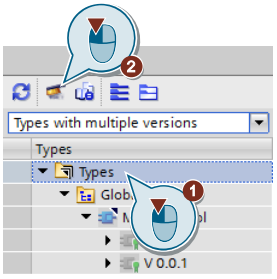
1. Open the "Library view".
2. Enable the library management with the focus on the project library.
3. Enable the filter "Types with multiple versions".

Case 1: Older type is not used in project nor in the project library

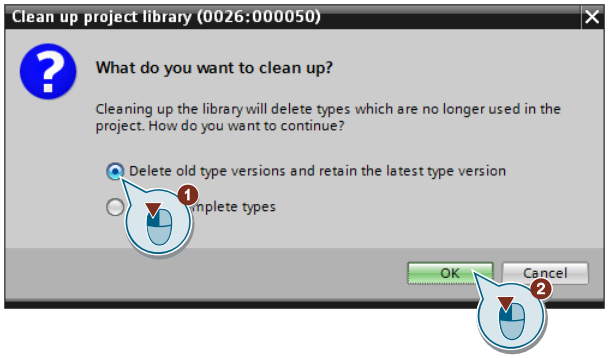


Solution (integrator, developer, user)

1. Select the "Types" file in the project library and click the "Clean up library" button.

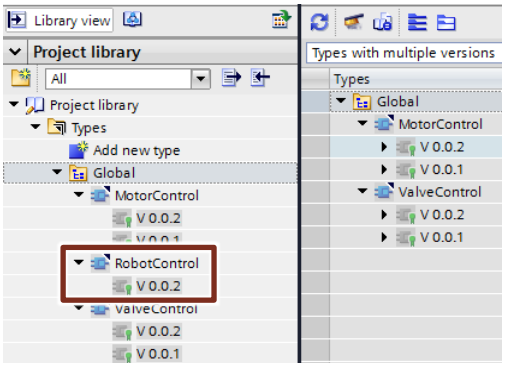


2. In the dialog, select "Delete old type versions and retain the latest type version".



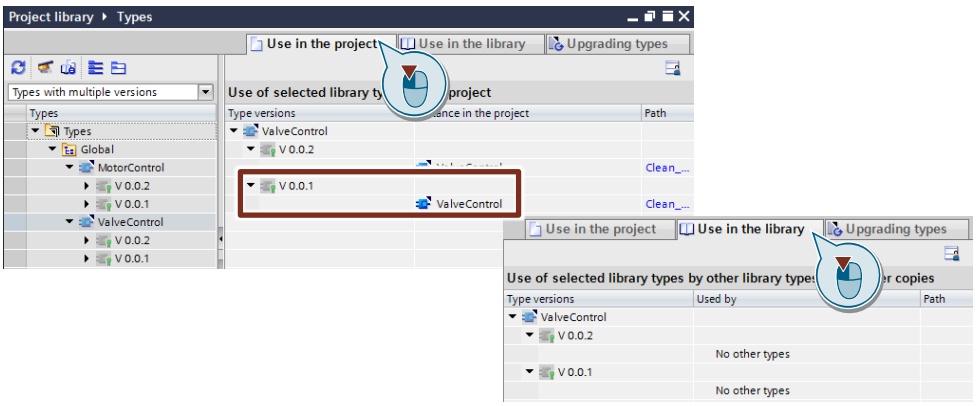
Result

The "RobotControl" type has one only version in the project library and is no longer displayed in the library management as type with several versions.



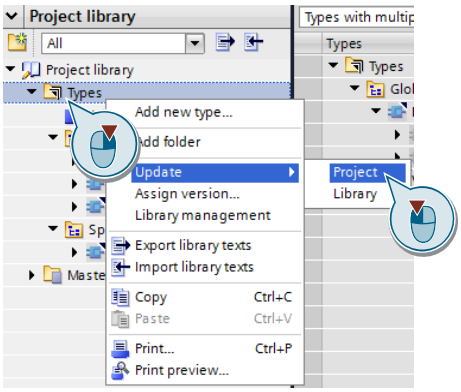
The two types "MotorControl" and "ValveControl" are still available in several versions. The reason for this is down to the following cases.

Case 2: Older type is still used in the project

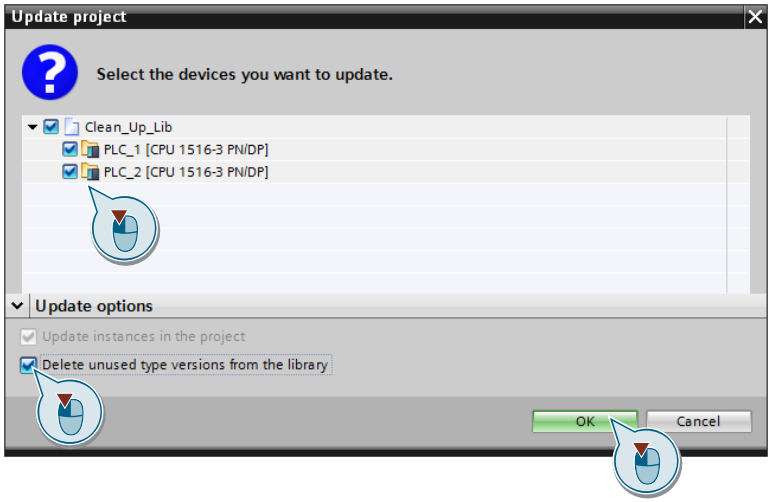


Solution (integrator, developer, user)

1. Right-click in the project library on the "Types" folder and select "Update > Project".

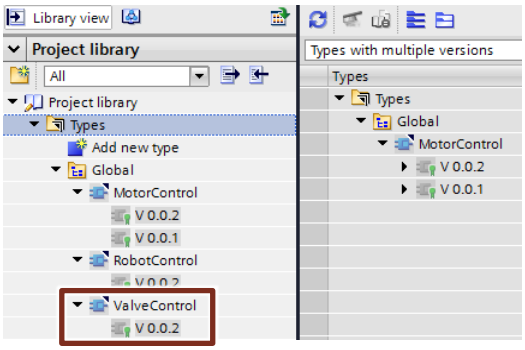


2. Select all PLCs that are to be updated, delete all types that are not used in the library and confirm the dialog using "OK".



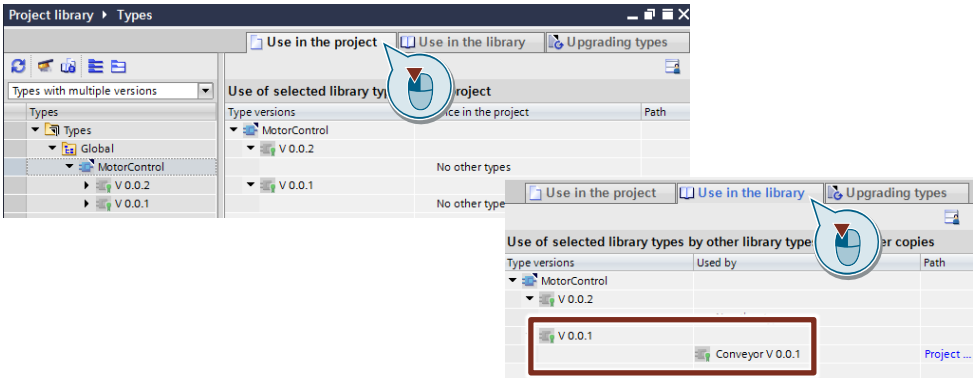
Result

The "ValveControl" type only has one version in the project library and is no longer displayed in library management as type with several versions.



The "MotorControl" type is still available with several versions. The reason can be found in the following case.

Case 3: Old version is used in overlay library type



Solution (integrator)

Note In this case the overlay type has to be updated so that it uses the later version. Arrange with the "developer" who edits the type how this conflict is to be solved.

1. Instance the overlay block (here "conveyor") in your project.
2. Carry out the step-by-step instruction in chapter 3.1.2 Updating existing library elements in the "integrator project" from the "transfer libraries".
3. Carry out the step-by-step instruction in chapter 3.2.3 Deleting older type versions in the global "development library"

7.3 Accidental assignment of a later version

If you assigned a later version to a type in your project unwittingly, you can bring your project back to the original state as follows.

Case 1: The type has no overlay types

Solution (integrator, developer)

1. Delete the type in the project library.
2. Confirm the dialog and select the option that all instances in the project are also deleted.
3. Navigate to the original types in global library and instance it in your project.

Result

The original type is automatically entered in the project library.

Case 2: The type has overlay type(s)

If you accidentally assign a new version to a type that has overlay types, they will also be reversioned. Therefore, you have to delete the overlay types first.

Solution (integrator, developer)

1. Identify the overlay types in the library view and delete them from the project library.
2. Delete the unwittingly reversioned types.
3. Navigate to the original types in global library and instance them in your project.

Result

The original type and the overlay types are back in their original state.

7.4 Accidental separation of an instance from type

If you have separated a type instance from associated types, this cannot be reversed, since a retypification would lead to a completely new type.

Solution (integrator, developer, user)

1. Delete the element in the project, which was originally a type instance.
2. Navigate to the original type in the project library and instance it in your project.

Result

The element is again instanced as type in your project.

7.5 Accidental deleting of a type instance

You have deleted a type instance accidentally.

Solution (integrator, developer, user)

1. Navigate to the unwittingly deleted type instance in your project library or the global library.
2. Instance the type in your project.

Result

The accidentally deleted type instance is back in your project.

7.6 Deleted types in global libraries are not deleted from the project

You get a new version of a global library. Various blocks are no longer available in the latest version and you also want to delete these blocks in your project.

Solution (integrator, developer, user)

1. Read the change history in order to identify the deleted types.
2. Manually delete the types in the project library. If the types are instanced in the project, also delete them manually.
3. Replace the deleted types, as described in the change history.

Result

You only use the types in your project that are located in the current "corporate library".

8 Valuable Information / Basics

8.1 Terms

Project library

Each TIA Portal project has its own library, the project library. Store the objects in the project library, which you would like to reuse within the project. The project library is opened, stored and closed together with the current project.

Global libraries

Global libraries are independent from a certain project and can be passed on to other users.

A joint access to global libraries, e.g., to a network drive is also possible when all users open global library write protected. The TIA Portal manages the global library automatically. As soon as a later version of an already existing global library is available, you will get the prompt to update the respective global library to the latest version.

Created global libraries from earlier versions of the TIA Portal can still be used. To do this, you have to upgrade them.

Note

The handling of global libraries are described in the following entries:

How do you open/upgrade global libraries in the TIA Portal?
https://support.industry.siemens.com/cs/ww/en/view/37364723

How can you open a global library with write access rights in STEP 7 (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/37364723

When starting TIA Portal V13 onwards, how do you get a global library to open automatically and use it as corporate library, for example?
https://support.industry.siemens.com/cs/ww/en/view/100451450

How do you update corporate libraries in STEP 7 (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/109739199

Master copies

Master copies are elements in the library that are not typified. They have no connection to the usage locations in the project.

Types

Types are versionable library elements and support a professional further development. Projects, in which the types are used can be updated as soon as later versions of the types are available.

Type instances

A type instance is the respective usage location of typified library elements in a project.

8.2 Overview of all library elements

Not every library element can be specified as type or master copy.

Table 8-1: Library elements

Library element	Type	Master copies
SIMATIC PLC		
OB	-	X
FB	X	X
FC	X	X
DB (global)	-	X
Technology objects	-	X
PLC tags	-	X
PLC data types	X	X
Watch and force tables	-	X
Traces	-	X
Text lists	-	X
SIMATIC HMI		
Screens	X	X
Faceplates	X	-
Templates (of screens)	-	X
Pop-up screens	-	X
Slide-in screens	-	-
HMI tags	-	X
Scripts	X	X
Protocols	-	X
HMI UDT	X	-
HMI style	X	-
HMI style sheet	X	-
User administration: User	-	X
User administration: Groups	-	X

8.3 Differences of the types for PLC and HMI

There are system-related differences between the typifiable objects for PLC and HMI:

Table 8-2: Differences of types for controller and HMI

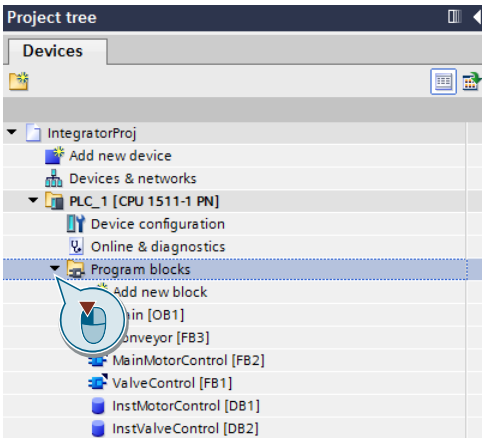
PLC	HMI
Subordinate control elements are typified.	Subordinate HMI elements are **not** typified.
Subordinate control elements are instanced.	Subordinate HMI elements are **not** instanced.
Control elements are edited in the program of a PLC. A device is always needed.	HMI screens and HMI scripts are edited in an HMI. Faceplates and HMI - UDTs are directly edited in the library.

8.4 Displaying type information in the project navigation

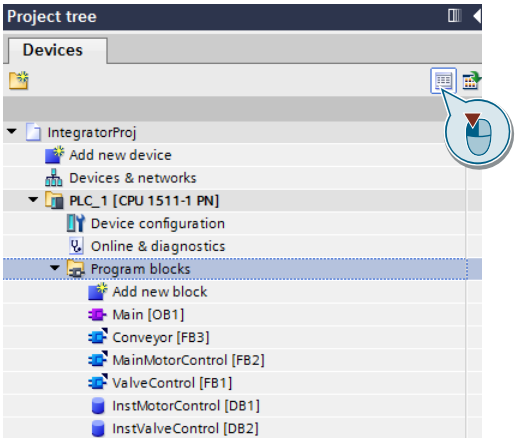
In order to display the type information (name, version) in the project navigation, proceed as follows.

1. Open the TIA Portal and your project.

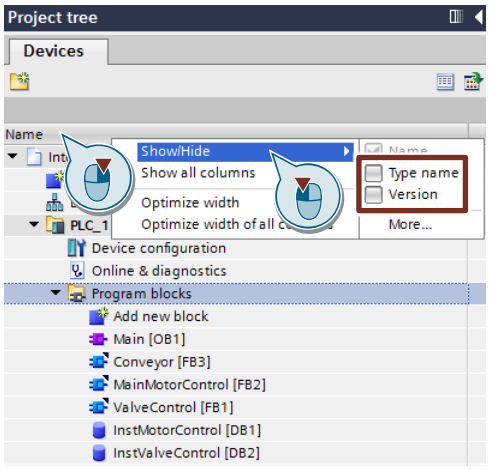
2. Open the "Program blocks" folder.



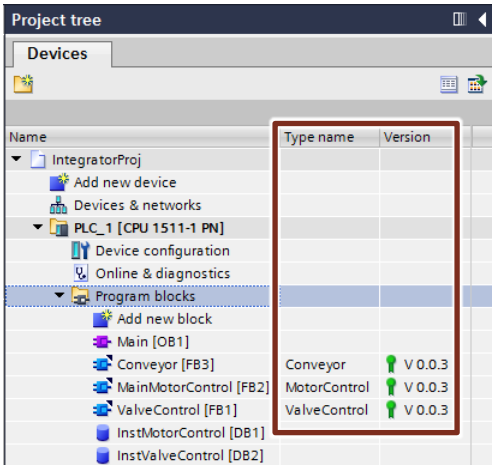
3. Click on the "Display/hide column header" button.



4. Right-click on the column header, select "Show/Hide" and enable “Type name" and "Version".



5. The type information is displayed in the project navigation for all type instances.

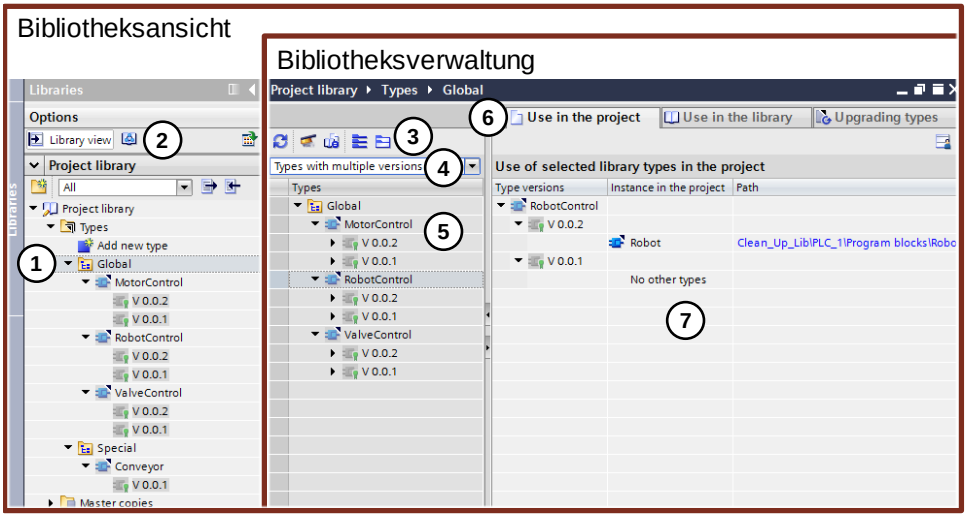


8.5 Library view and library management

8.5.1 Overview

The library view shows you the elements of a library in different views and, for example, in the detail view you will see further features of the individual elements.

Figure 8-1: Library view and library management



1. Library navigation
2. Buttons
   - Open/close library view
   - "Library management" button
     The library management always opens the respectively set focus in the library navigation.
3. Buttons
   - Update uses
   - Clean up library
   - Harmonize project
   - Expand all entries
   - Collapse all entries
4. Filter dialog (see chapter 8.5.2 Filter dialog in the library management)
5. Types that correspond to the filter
6. Tab
   - Use in the project
   - Use in the library
   - Upgrading types
7. Usage location of types that correspond to the filter

8.5.2 Filter dialog in the library management

The filter dialog consists of several filters, in order to show the types with different status and mutual dependencies.

- Show all types
  This filter shows all types.

- Types with pending changes
  This filter shows all types that are in the status [in test].
  If you want to finalize a project or a global library, check with this filter, whether all types are released.

- Released Types
  This filter shows all published types.

- Types with multiple versions
  This filter shows all types that have several versions.
  Recommendation: Delete old versions of the types in order to improve the clarity in the library and to avoid inconsistencies.

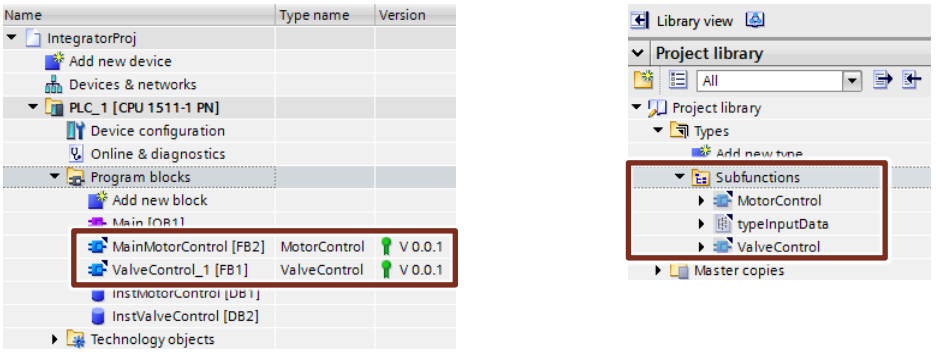
- Types not used in project
  This filter shows all types that have no instance in the project.
  Recommendation: Check whether all types are instanced in the project.

- Types with new versions for upgrading other types
  This filter shows all overlay types that can be updated based on later versions of subordinate types.
  Recommendation: Check whether overlay types can be updated and make sure the library is consistent.
  See chapter: 3.1.2 Updating existing library elements in the "integrator project" from the "transfer libraries".

8.5.3 Harmonizing project and libraries

If you renamed the types in the program or if they were renamed automatically by TIA Portal due to update mechanisms, there is an option to compare the names and paths of the project library with the programs in the project.

Figure 8-2: Type instance names do not correspond to the type names or folder structure

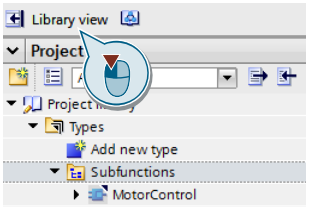


Solution

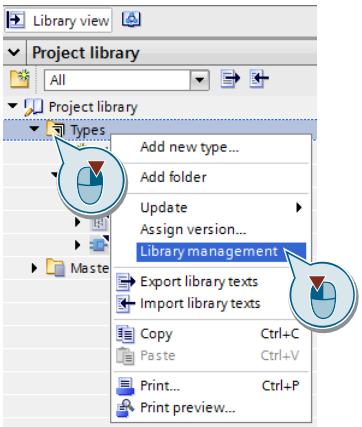
You can harmonize all type instances in all devices as follows, using the central harmonizing function in the library management.

- Type names are compared with the type instance names.
- Paths of the project library are compared with the program.

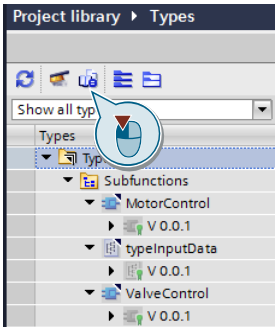
1. Open the "Library view".



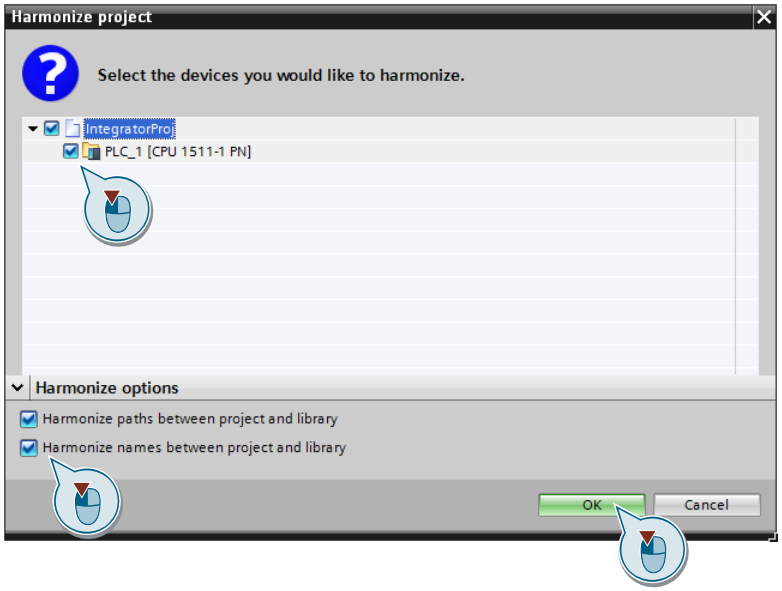
2. Right-click on the "Types" folder in the project library and select "Library Management".



3. Click on the "Harmonize project" button.

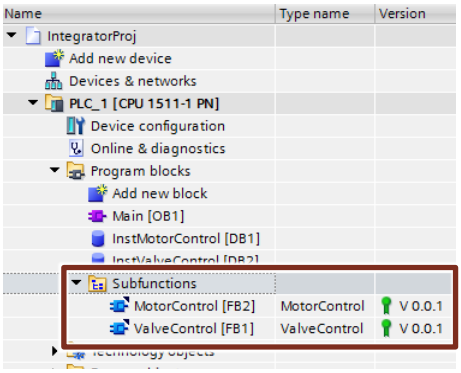


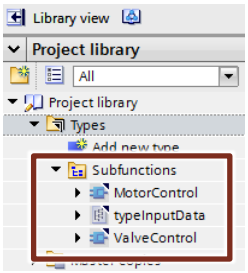
4. Select the PLC in which the type instances are to be harmonized. Select individually whether the paths and/or the names are to be adapted. Confirm the dialog using "OK".



Result

- Type instance names correspond to the type names.
- Type instances are in the same folder structure as the types in the project library.





8.6 Version numbers

The version of a library element always consists of three numbers. The following table suggests how you can assign the version numbers.

Table 8-3: Definition of version

Vx.y.z	Main version number	Auxiliary revision number	Revision number
	Non-compatible changes	Compatible changes	Error correction
	• Reduction of interfaces • Change of existing interfaces • Incompatible extension of functionality	• Extension of interfaces • Compatible extension of functionality	• Bugfix

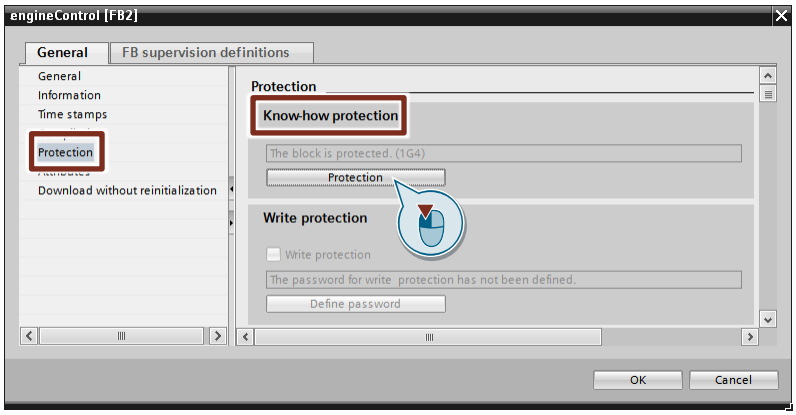
8.7 Know-how protection

Using the know-how protection, you can protect blocks from unauthorized access with the help of a password.

The blocks in global libraries have the following features:

- Know-how protected blocks can be typified.
- Know-how protected blocks only be recompiled with a password.

Figure 8-3: Know-how protection of blocks



Note For more information, please refer to the online help des TIA Portals at "Programming a PLC > Protecting blocks > Protecting blocks".

Note Know-how protected blocks using S7-1200

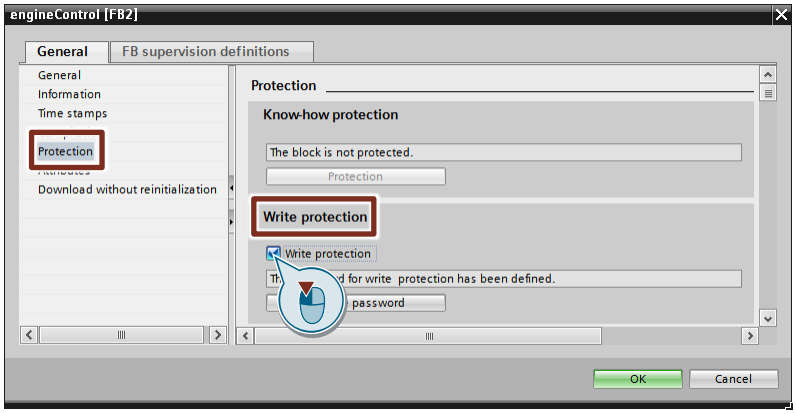
Know-how protected blocks can only be loaded in the PLC within the main version of the firmware without repeated compilation (requires password). This only applies to S7-1200 PLCs.

8.8 Write protection

You can set up a password-based write protection for blocks in order to avoid accidental changes. Blocks with a write protection have read-only access, the block properties can still be edited and compiled.

Please note that the write protection is not a know-how protection, since the write protection can be removed again at any time. If a block is write protected, you cannot set up a know-how protection additionally.

Figure 8-4: Write protection of blocks



8.9 User-defined documentation (User help)

You can create your own user-defined documentation to explain to others how your project functions or how to use individual library types.

8.9.1 Creating user-defined documentation

User-defined documentation supports multiple file types.

Recommendation: Create the user-defined documentation as a PDF file so that the user help can be opened independent of platform on many different computers.

The PDF files for user-defined documentation of individual blocks in the library will be created outside of the TIA Portal. You can create the user help in all available interface languages.

In order to retrieve user help for the right block, the file name must match the name of the block in the TIA Portal exactly, e.g. "LGF_Astro.pdf" for the block "LGF_Astro". In addition, the PDF files must be saved in the correct directory.

So that the items of user-defined documentation for library blocks can be delivered along with the library, save the PDF files in the following global library directory.

<Folder of the global library>\UserFiles\UserDocumentation\<Folder for the corresponding language>\<Object category>

The following table shows the folders for each language for the default interface languages:

Table 8-4

Language	Sub-folder for the respective language
German	\de-DE
English	\en-US
Spanish	\es-ES
French	\fr-FR
Italian	\it-IT
Chinese	\zh-CN

The following table shows **only** the English names of the most important object categories for libraries in the TIA Portal:

Table 8-5

Object category	English name of the sub-folder
Types in the library	Library types
Master copies in the library	Master Copies
Libraries in the "Libraries" task card or in the library view	Libraries

Example: Directory for a type in a library

The German and English PDF files for the types in the "LGF" library are saved in the following directories of the global library.

- German: "LGF\UserFiles\UserDocumentation\de-DE\Library Types"
- English: "LGF\UserFiles\UserDocumentation\en-US\Library Types"

Example: Directory for master copies in a library

The German and English PDF files for the master copies in the "LGF" library are saved in the following directories of the global library.

- German: "LGF\UserFiles\UserDocumentation\de-DE\Master Copies"
- English: "LGF\UserFiles\UserDocumentation\en-US\ Master Copies "

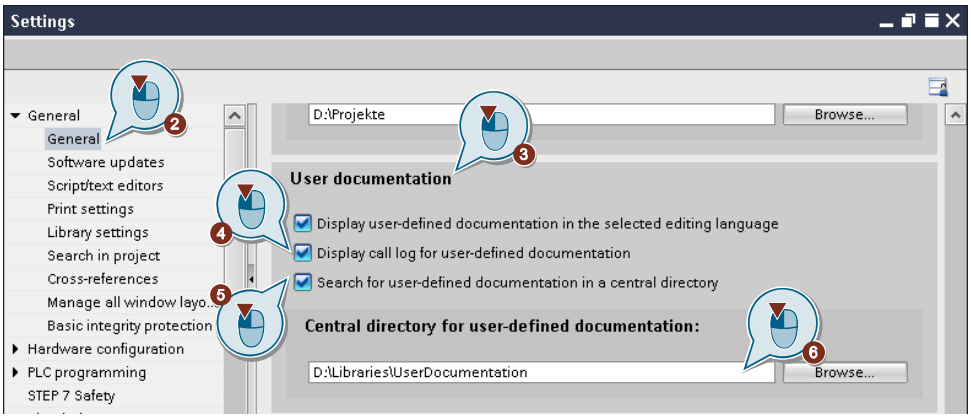
Alternative: Central directory for user-defined documentation

Alternatively, you can store the user-defined documentation in a central directory for all projects. The same rules that apply to the sub-folders of the global library also apply to the structure of sub-folders in the central directory.

To define a central storage location for user help, proceed as follows:

1. In the "Options" menu, select the "Settings" command.
2. Open the area "General > General".
3. Navigate to the "User documentation" section.
4. Activate the checkbox "Display call log for user-defined documentation" to display a log of the call-up of the user-defined documentation in the Inspector window.

5. Activate the "Search for user-defined documentation in a central directory" checkbox to store user-defined documentation in a central directory for projects.
6. In the "Central directory for user-defined documentation" field, specify the path where you want to store cross-project documentation.



Note To make it easier to connect to user-defined documentation, an access log (see Point 4) can be shown for the user-defined documentation. The messages in the log show which directories are being searched for documentation and whether the user-defined documentation has been successfully accessed. The expected file name is also shown. Here you can see how you must name your user-defined documentation and in which directories you have to save it. The access log matches the same order in which the user-defined documentation is searched.

8.9.2 Accessing user-defined documentation

You can access the user-defined documentation for a selected block in the "Library" task card or in the library view with the key combination <Shift+F1>. The respective PDF file is always opened with the default program in Microsoft Windows.

So that the user-defined documentation for the blocks can also be accessed in the project navigation, you must copy the directories with PDF files into the project directory "UserFiles" or into the central directory for user-defined documentation (see Alternative: Central directory for user-defined documentation).

9 Annex

9.1 Service and support

Industry Online Support

Do you have any questions or need support?

Siemens Industry Online Support offers access to our entire service and support know-how as well as to our services.

Siemens Industry Online Support is the central address for information on our products, solutions and services.

Product information, manuals, downloads, FAQs and application examples – all information is accessible with just a few mouse clicks at
https://support.industry.siemens.com

SITRAIN - Training for Industry

Highly qualified employees are a decisive success factor for any company. Ongoing skills development and expert knowledge strengthen a company's competitive position and innovative power. Our training courses for industry at locations around the globe will help you achieve this goal efficiently because they deliver directly applicable knowledge through innovative teaching methods and custom-tailored training concepts.

www.siemens.en/sitrain

Technical Support

Siemens Industry's Technical Support offers quick and competent support regarding all technical queries with numerous tailor-made offers
– from basic support right up to individual support contracts.

Please address your requests to the Technical Support via the web form:
www.siemens.en/industry/supportrequest

Service offer

Our service offer comprises, among other things, the following services:

- Product Training
- Plant Data Services
- Spare Parts Services
- Repair Services
- On Site and Maintenance Services
- Retrofit and Modernization Services
- Service Programs and Agreements

Detailed information on our service offer is available in the Service Catalog:
https://support.industry.siemens.com/cs/sc

Industry Online Support app

Thanks to the "Siemens Industry Online Support" app, you will get optimum support even when you are on the move. The app is available for Apple iOS, Android and Windows Phone:
https://support.industry.siemens.com/cs/ww/en/sc/2067

9.2 Links and literature

Table 9-1

No.	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the entry page of the application example https://support.industry.siemens.com/cs/ww/en/view/109747503
\3\	FAQ: When starting TIA Portal V13 onwards, how do you get a global library to open automatically and use it as corporate library, for example? https://support.industry.siemens.com/cs/ww/en/view/100451450
\4\	Topic page: Libraries in the TIA Portal https://support.industry.siemens.com/cs/ww/en/view/109738702
\5\	Programming Styleguide for S7-1200 and S7-1500 https://support.industry.siemens.com/cs/ww/en/view/81318674
\6\	SIMATIC S7-PLCSIM Advanced: Co-Simulation via API https://support.industry.siemens.com/cs/ww/en/view/109739660
\7\	Library of general functions (LGF) for STEP 7 (TIA Portal) and S7-1200 / S7-1500 https://support.industry.siemens.com/cs/ww/en/view/109479728
\8\	FAQ: How do you open/upgrade global libraries in the TIA Portal? https://support.industry.siemens.com/cs/ww/en/view/37364723
\9\	FAQ: How can you open a global library with write access rights in STEP 7 (TIA Portal)? https://support.industry.siemens.com/cs/ww/en/view/37364723
\10\	FAQ: How do you update corporate libraries in STEP 7 (TIA Portal)? https://support.industry.siemens.com/cs/ww/en/view/109739199

9.3 Change documentation

Table 9-2

Version	Date	Modification
V1.0	11/2017	First version
V1.1	10/2019	New chapters 3.3.2 Creating write-protected libraries 3.3.3 Discontinuation of types 5.1.5 Replacing discontinued types 8.9 User-defined documentation (User help)
V1.2	12/2019	Chapter expanded 5.1.4 Updating all types in the "plant project"