Placeholder mechanism FAQ No.0007

Part	Version	Revision	Date	Status
en	6.2.3.0	001	2019-01-01	Released

Content

Introduction

Using Combivis studio 6 it is possible, to create and use library 'landscapes' widely cascaded into each other. Therefore it is absolutely essential, that different referrals always point to the same library all the time, especially when a library is using an interface defined in a different library, e.g. or different libraries use types defined in a third one.
If that pre-condition is not fulfilled, the references will not be unambiguous and this will lead to compiler errors, as library elements are only considered as equal if all three identifying elements (vendor, name and version) are as expected. Different versions, e.g. may hold elements with the same name but different in- and outputs or behaviour.

To get a grip on this this task, the so called placeholder concept was introduced into Combivis studio 6, that is to be described in detail within this document, mentioning also its background.
This document is addressed mainly to all library developers, bringing in a basic knowledge about possibilities and features of the IDE.

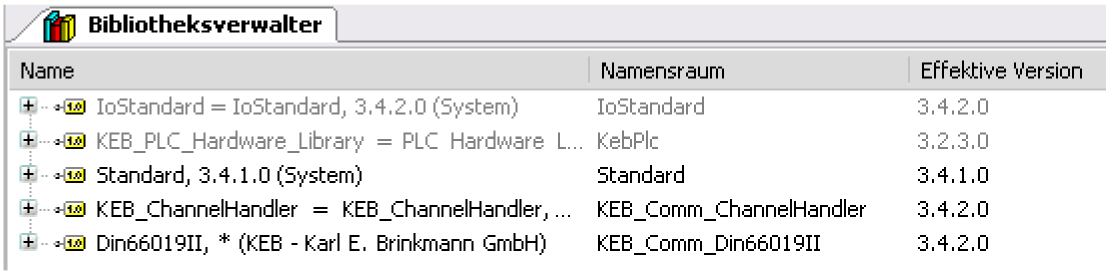
Prehistory

Different options for inserting a library into a IEC project exist in Combivis studio since all time – these methods are 'explicit library version', using always the newest library available in the IDE or using a so called placeholder. However, the different ways of inserting a library into a project cause different effects and behaviour, that might not be the intent of the developer or even not noticed at all.

Adding method 'explicit library version'

Using this method, the developer defines one single, exact version of a library that is to be used in the project.

A library inserted in a project in this way can easily be identified by the explicit version tag that follows the library's name in the library manager. The version displayed in the column 'Effective version' is equal to that or remains empty, if the former inserted version is not available in the current IDE. (This is additionally illustrated by the yellow explanation mark in front of the library name.)


Bibliotheksverwalter

Name	Namensraum	Effektive Version
IoStandard = IoStandard, 3.4.2.0 (System)	IoStandard	3.4.2.0
KEB_PLC_Hardware_Library = PLC Hardware L...	KebPlc	3.2.3.0
Standard, 3.4.1.0 (System)	Standard	3.4.1.0
KEB_ChannelHandler = KEB_ChannelHandler, ...	KEB_Comm_ChannelHandler	3.4.2.0
Din66019II, * (KEB - Karl E. Brinkmann GmbH)	KEB_Comm_Din66019II	3.4.2.0

In this case, the library 'Standard' is added explicitly in version '3.4.1.0' and this version is also available in the current IDE version.

A version of an explicitly added library will *never* change; this has the following consequences:

- Existing references to objects from this library will always remain available. (As long as the library's maintainer keeps to the strictly recommended convention, that all changes made to a released library result also in a change of the library's version.)

- The added version of that library has to be available in every IDE, in which the project is opened (more exact: compiled) or, in case of a library project, the library is used. (This criteria can be handled by using a project archive for transferring projects from one IDE (version) to another.
- In a more complex library 'landscape', in which multiple libraries depend on each other (like the KEB ChannelHandler – libraries), every library has to use exactly the same version in order to be usable. From a certain number of libraries this is no longer practically feasible.
- If there is an update to the referenced library, e.g. a bug fix, the fixed reference to the library's version has to be modified manually. (Referring to above paragraph three for all library's and projects that are using the library.)

Adding method 'newest library version'

Using this method, the project or library developer just decides to use a certain type of library, not knowing which version will effectively be used at any point of compile. The version will always be the newest version available on the individual IDE the project (or, in case of a library, the project that uses the library) is opened. However, the effective version will be the same at all points this inserting method is used.

A library added this way is illustrated with a star ('*') behind the library name, the currently used effective version can be checked in the column 'effective version. This version may change, if a newer version of that library (same vendor, same library name but newer version) becomes available in any IDE this project is opened in.
In the previous screenshot the library DIN66019II was added using the method 'newest library version'.

The following effects will affect a project that contains libraries added this way:

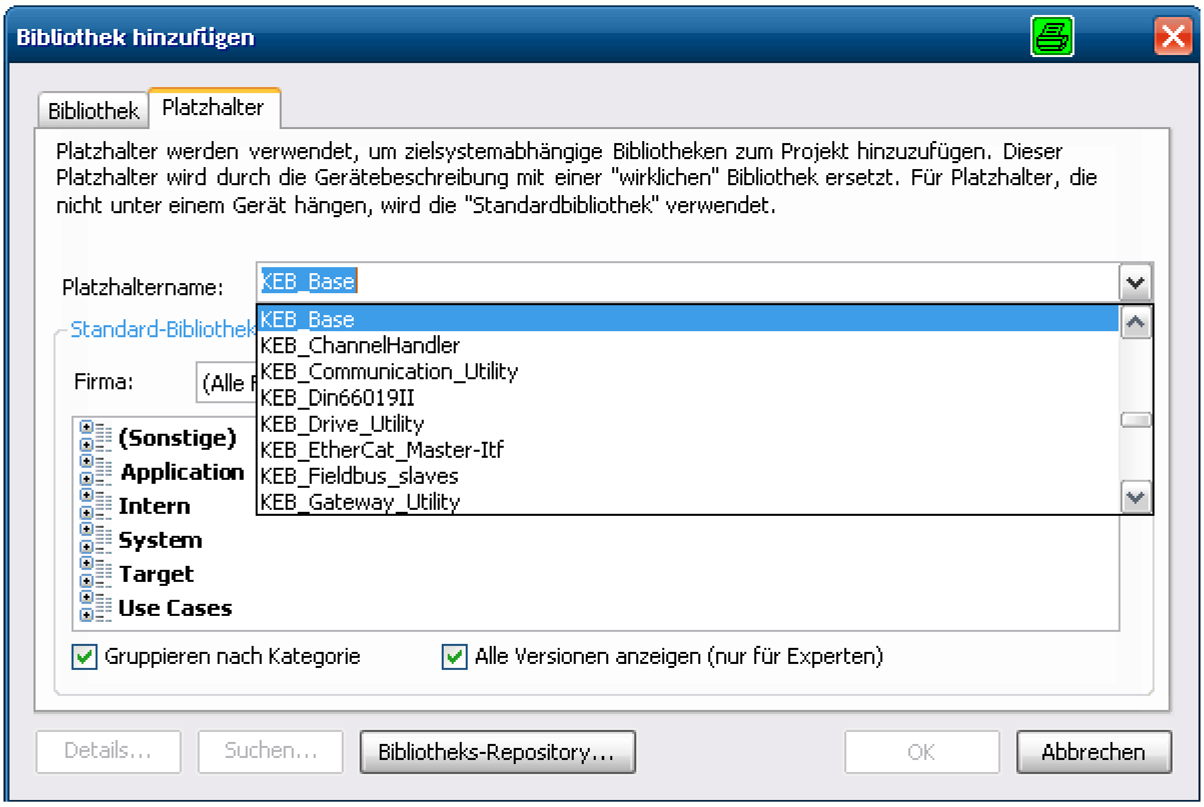
- Without any further action of the project maintainer any newer library versions will be used immediately on opening the project. No changes to the project needed.
- In a 'library landscape', it is at least secured, that at all points a library is inserted this way will have the same version and therefore all cross-references stay valid. (This is the reason, why this *was* the recommended way for all libs up to (not including) V3.4.x)
- If, at any later point, e.g. due to a newer Combivis studio 6 setup, a newer library is used in the project, different machine code will be generated and a online change or Login without download will not be possible anymore
- The previous paragraph is not even preventable using a project archive as it only contains the effective version of the currently used project – if there are newer versions existing in an IDE at the point an archive is extracted, those will be used instead!
- At the time the project is created, it is unclear, what exact machine code will result from the project. Also, it is not possible to tell, which library version was initially (effectively) used when the project was created.

Adding method 'placeholder'

As a third and last method, the method using a placeholder exists. Like at using the method 'newest', the project / library developer does not know here, which version of the inserted library will be used effectively. In contrast to the method 'newest' the effective version depends here not just on the simple presence of a newer library in the IDE, but on the definition made for the placeholder. (Therefore, two options do exist, see '

Against the background of long-term maintenance of an IEC project, it is absolutely necessary that a saved project builds a container in which all versions are defined in a way that it is always possible to create the same machine code from it. Only in that way, a login without any code change (online change or complete download) is possible for a possibly long running project.
To archive this goal, it is on the one hand necessary to save the project as a project archive in order to have all used libraries and devices in a common file. On the other hand, the used libraries have to be added to the project in a way, that defines unchangeable versions for each library – following the above advises this means adding libraries using fixed versions or placeholders, but *never* the method 'newest.
Especially in library projects it is therefore essential, that all libraries are inserted using placeholders. For end-user projects the task may be solved using fixed versions as well.
Background: Placeholder definition'). If, during development of a library, none of the placeholder definitions are available the 'standard library selected on insertion is used. This is typically the newest version. ('*')
A library added by its placeholder is displayed as the placeholder's name followed by an equal sign and the effective library, including its version. Again, the version displayed in the column 'Effective version' is equal to that.

In the following screenshot, the library 'KEB_ChannelHandler' is inserted using its placeholder, using version 3.4.2.0 in this example.


Bibliothek hinzufügen

Bibliothek | Platzhalter

Platzhalter werden verwendet, um zielsystemabhängige Bibliotheken zum Projekt hinzuzufügen. Dieser Platzhalter wird durch die Gerätebeschreibung mit einer "wirklichen" Bibliothek ersetzt. Für Platzhalter, die nicht unter einem Gerät hängen, wird die "Standardbibliothek" verwendet.

Platzhaltername: KEB_Base

- KEB_Base
- KEB_ChannelHandler
- KEB_Communication_Utility
- KEB_Din66019II
- KEB_Drive_Utility
- KEB_EtherCat_Master-Itf
- KEB_Fieldbus_slaves
- KEB_Gateway_Utility

Standard-Bibliothek

Firma: (Alle F

- (Sonstige)
- Application
- Intern
- System
- Target
- Use Cases

✓ Gruppieren nach Kategorie ✓ Alle Versionen anzeigen (nur für Experten)

This method combines the positive aspects of the previous methods:

- Also in bigger and /or slightly confusing scenarios with various new libraries still one, strictly defined version will be used.
- *All* references to a library inserted via placeholder point to the same library version. Therefore no incompatible setups can come up anymore:
- Updates of a library version are released via the definition of their placeholder. No changes to existing projects or libraries necessary.

If there is a library to be added using a placeholder, in the dialog ‘Add library’ the tab ‘Placeholder’ has to be selected. In the upper input box, the placeholder’s name may either be entered or selected from a list of known placeholder definitions in this IDE. Afterwards, a so called standard library has to be selected, which will be used in case the placeholder is not supposed to be resolved, usually in a library project. (The replacement of a placeholder to a library in practice is only done in the final IEC *project* for the PLC.) Here, the reference ‘newest’ (‘*’) is commonly used.

Attention: It is possible to select a completely different library here; this is hardly ever the intent!

The main goal for an IEC project: One project always leads to the same machine code

Against the background of long-term maintenance of an IEC project, it is absolutely necessary that a saved project builds a container in which all versions are defined in a way that it is always possible to create the same machine code from it. Only in that way, a login without any code change (online change or complete download) is possible for a possibly long running project.
To archive this goal, it is on the one hand necessary to save the project as a project archive in order to have all used libraries and devices in a common file. On the other hand, the used libraries have to be added to the project in a way, that defines unchangeable versions for each library – following the above advises this means adding libraries using fixed versions or placeholders, but *never* the method 'newest.
Especially in library projects it is therefore essential, that all libraries are inserted using placeholders.
For end-user projects the task may be solved using fixed versions as well.

Background: Placeholder definition

There are two options existing, where a placeholder may be resolved to a library version in practical; the first possibility is a definition inside a PLC's description, which has priority against the second one, the definition in the so called 'library profile'

Inside a PLC's device description

The first point the IDE will try to resolve a placeholder will always be the PLC's description. Here, typically all versions of those libraries are defined, that are close connected to the device's firm- or even hardware like all the system libraries. The background is, these libraries are often so called 'IEC frontends' to functions that are implemented directly in the PLC's firmware. Only the firmware's developer can be aware of the functions and features supported by the respective release and list the suitable versions in the device description.

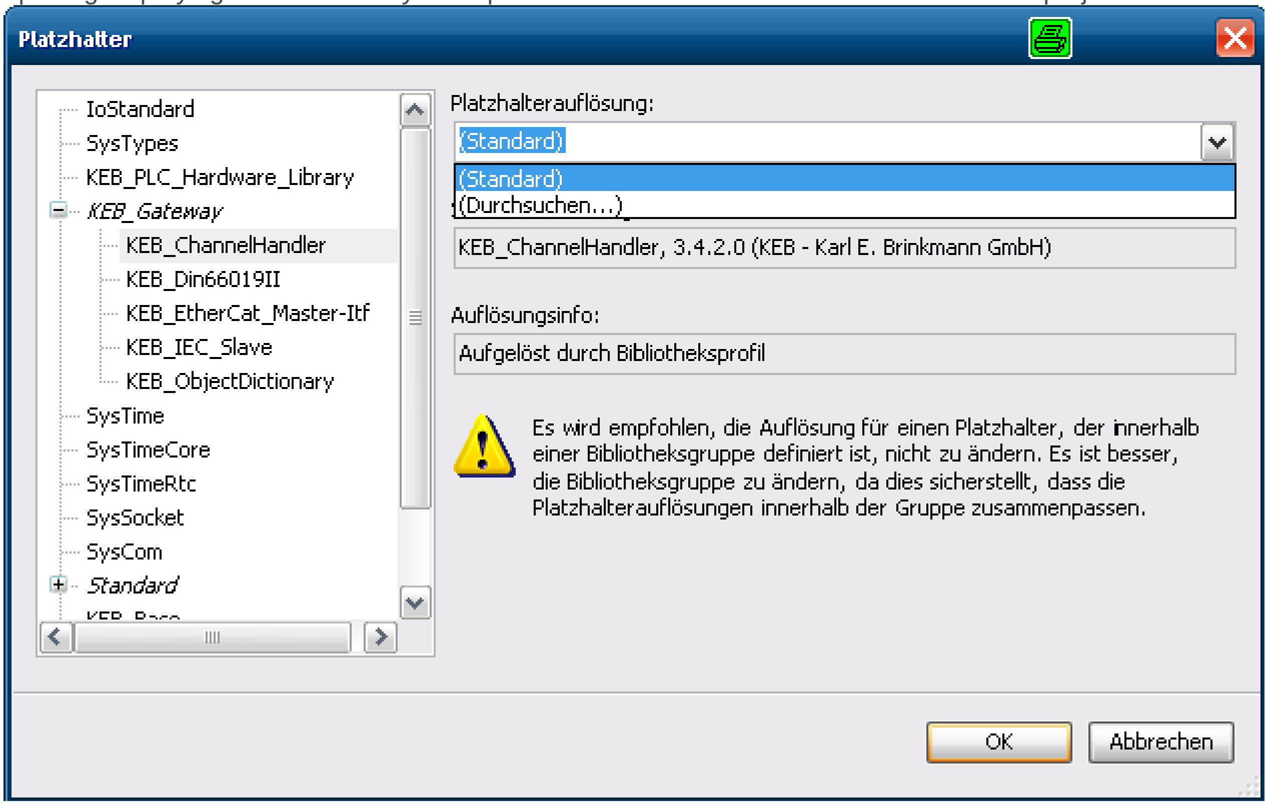
Inside the library profile

In addition to the device description there does the 'library profile' exist: Here, the IDE will try to resolve the placeholders that were not available in the device description via the used compiler version of the project. (See Project -> Project settings -> Compiler) This method is usually chosen for libraries that do not have a direct connection to the PLC's hard- or firmware, but instead already use a certain level of abstraction like the KEB Utilities or the ChannelHandler libraries. The core criterion is the compiler version, so if a set of newer libraries shall be used in the project, also the compiler version has to be increased. (The common restrictions remain unaltered – e.g. device's major version 'Y' in 3.Y.x.x has to compatible to the compiler's major version, etc.)

Overriding a placeholder definition in a project

Overriding a placeholder may lead to unexpected / faulty behaviour if airy used and therefore should only be used at exceptional cases or by skilled users!

Although placeholders should always point to the effective version desired by the respective developer, it might become necessary to override the released definitions for testing purpose or other reasons. This is possible via the push-button ‘Placeholders…’ inside a library manager. After a click on it, a Dialog is opening displaying a list of actually used placeholders and their resolution in the current project:



Platzhalterauflösung: (Standard)

(Standard)
(Durchsuchen…)

KEB_ChannelHandler, 3.4.2.0 (KEB - Karl E. Brinkmann GmbH)

Auflösungsinfo: Aufgelöst durch Bibliotheksprofil

Es wird empfohlen, die Auflösung für einen Platzhalter, der innerhalb einer Bibliotheksgruppe definiert ist, nicht zu ändern. Es ist besser, die Bibliotheksgruppe zu ändern, da dies sicherstellt, dass die Platzhalterauflösungen innerhalb der Gruppe zusammenpassen.

The IDE is differentiating between ‘resolved by device’ and ‘resolved by library profile’ and some placeholders are summarized inside a group, like the KEB Gateway libraries. Groups may be identified from their italic font and the ‘+’ in front of the group’s name, indicating the element might be extended. A overwritten placeholder will result in its name (and its possibly existing group name as well) printed in bold letters in the list.
The manual selection of a different library version is done just as one is familiar with adding a normal library: On selecting ‘Browse…’ inside a placeholder’s detailed view, the library repository is opened an a specific version may be selected.

Naming convention for KEB libraries and placeholders

KEB will use the namespace ‘KEB’ for all their placeholders an libraries and therefore all placeholders are kept in the format ‘KEB_[lib. name]. Also, if new libs are created, this convention is to be followed, as it simplifies the overview significantly.

Summary and code of practice

The described concept offers a solution for the library usage self-consistent and easy to use for Combivis studio 6 end-users. All KEB application libraries use placeholders by now and the following list shall help in deciding which method to use.

In a normal IEC end-user project…

- …libraries should always be added in a way, in which their versions are strictly defined (One project – one machine code)
    - Either fixed versions are used (for non-system libraries)
    - Or/and, especially for all system- and other hard- or firmware related libraries, placeholders are used. With this, there will be no unsupported functions offered (due to firmware incompatible system libraries), which may lead to unresolved references during download and also after any update of the IDE or the device the suitable, released set of libraries is used.
- …should only in very rare cases references via the option ‘newest’ (‘*’) be created.

In a library project…

- …further requirements do exist regarding the internal used libraries. The version of underlying base libraries has to be
    - …explicit
    - …possibly also be available in the same version than in other libraries inside an end-user project.
- To fulfil these requirements, libraries used in library projects should always be added using their placeholder. As so called ‘standard library’ for resolution during the library development, the expected placeholder resolution should be chosen, usually the option ‘newest’ (‘*’) is selected.

KEB Automation KG
Südstraße 38 • D-32683 Barntrup
fon: +49 5263 401-0 • fax: +49 5263 401-116
net: www.keb.de • mail: info@keb.de