CODESYS Compatibility Information

CODESYS – OEM Documentation

Version 4.0

Content

1 Overview

CODESYS, the CODESYS Runtime System and the IEC-Project are used in different versions and so the compatibility between each of these components is an important aspect.

CODESYS can be updated very easy via a new available setup. With a newer CODESYS installation you get typically a new Compiler Version, a set of libraries and new Features (PlugIns).

The CODESYS Runtime System is not so easy to update, because an OEM customer has to integrate the Runtime System into their Controller and to retest the complete Firmware. So this is typically not a short term process.

The IEC-Project has different versions during commissioning a machine. After the machine is released, the IEC-Project is only changed for service or maintenance work.

Here are the general compatibility aspects:

- Compatibility CODESYS - Runtime System
- Compatibility IEC-Project - Runtime System
- Compatibility DeviceDescriptions - Runtime System
- Compatibility External Libraries - Runtime System
- Compatibility Bootproject and Retain-Variables - Runtime System
- Compatiblilty CODESYS Projects
- Compatibility CODESYS Automation Platform Interfaces
- Compatibility CODESYS Compiler Version
- Compatibility CODESYS Compiler Version and Libraries

2 Compatibility CODESYS - Runtime System

The typical use case is that CODESYS is updated more often as the Runtime System and so CODESYS has a newer version as the Runtime System. This is no problem for existing projects, if neither the DeviceDescription nor the CompilerVersion are changed!

3 Compatibility IEC-Project - Runtime System

Existing projects must run unchanged on existing Runtime Systems, also with newer CODESYS versions. The following aspects are of interest:

- New features in CODESYS are activated typically in the DeviceDescription. So they are disabled for older Runtime Systems respectively older DeviceDescriptions.
- The versions of all external Libraries are resolved by the DeviceDescription. So they are matching to the Runtime System.
- Internal libraries are resolved by the library profile matching to the CODESYS version. This has no influence on the Runtime System is described in chapter .
- With a newer CODESYS version we provide typically a newer CompilerVersion. But for existing Projects, the selected CompilerVersion is not changed! ATTENTION: If you update the CompilerVersion to a newer version the generated IEC code can be changed (e.g. implicit generated code from PlugIns like visualization dependent on the selected CompilerVersion)! So this can lead to a misbehaviour of the Project e.g. different timing, fixed bugs which changes the behaviour, etc.
- The communication services are built according to a common tagged format. Unknown tags are ignored by the runtime system. A change in the communication service is therefore possible but the programming system will not expect the runtime system to interpret the new information.

4 Compatibility DeviceDescriptions - Runtime System

Existing projects must run unchanged on existing Runtime Systems, also with newer CODESYS versions. This can be fulfilled, because the DeviceDescription in the project is not changed an has the existing version, that matched to the runtime system version. Additionally you can login to the runtime system, if the version of the DeviceDescription and the runtime system are not too far away or incompatible:

- An older DeviceDescription version and a newer Runtime Systems are accepted at login. Newer DeviceDescription version and an older Runtime Systems is not accepted so the login is denied.
- the DeviceDescription contains a list of external libraries, which external functions are implemented in the runtime system. The corresponding version of the library is specified here in the so called library placeholder list. NOTE: The placeholder list should only contain the libraries, which corresponding runtime components are available in the runtime system!
- There is a possibility to configure the compatibility range between the DeviceDescription and the Runtime System. That means you cannot login into the controller if the versions are not matching! This range can be selected by the OEM with the following settings in the Runtime System (see SysTargetItf.h of the runtime system):
  - SYSTARGETKEY_INT_TARGET_VERSION_MASK “TargetVersionMask”

    Setting to specify a mask to check the target version for compatibility with the device description. In the mask only the significant digits are checked.
  - SYSTARGETKEY_INT_TARGET_VERSION_COMPATIBILITY_MASK “TargetVersionCompatibilityMask”

    Setting to specify a compatibility mask to check the target version for compatibility with the device description. A device description that is lower or equal the target version is accepted. A higher device description version is denied.

For details on creating own target packages and device descriptions, see Manuals/CODESYSControlV3_Manual.pdf, chapter 10.1.

Examples:

CODESYS Version	CODESYS Control Version	Device Description Version	Access possible	Comments
3.5.9.0	3.5.9.0	3.5.9.0	yes	This is the most ideal combination
3.5.9.0	3.5.4.0	3.5.4.0	yes	This the typcial compatibility use case
3.5.4.0	3.5.9.0	3.5.9.0	perhaps, but not recommended!	In CODESYS the set of all external libraries must be available and perhaps there are newer IEC language resource are used that leads to compile errors!
3.5.9.0	3.5.4.0	3.5.2.0	yes	This is possible because the DevDesc is in the compatible range of 3.5.x.x
3.5.9.0	3.5.4.0	3.5.9.0	no	This is denied by default because of an incompatible mismatch between DescDesc and the runtime system
3.5.9.0	3.5.4.0	3.4.3.0	no	This is denied by default because of an incompatible mismatch between DescDesc and the runtime system (out of compatibility range)

5 Compatibility External Libraries - Runtime System

Within a Main Version (e.g. 3.5.x.x) we are compatible in the Runtime System regarding external Libraries. So interface functions in external Libraries are not changed or removed within a Main Version. This is checked in every Release Test for a ServicePack against the previous ServicePack version. But new interface functions or structure definitions could be added in newer library versions without any influence on existing libraries.

6 Compatibility Bootproject and Retain-Variables - Runtime System

Existing bootprojects must be loadable within a main version by the Runtime System! Within a main version (e.g. v3.5.x.x) we must be backward compatible regarding the bootproject and the external Libraries! This means a newer runtime system must be able to load older bootprojects, but we must not be able to load newer bootprojects that maches only to newer runtime systems! Especially the backward compatibility of external Libraries are checked explicitly at every release of a ServicePack (see chapter CompatibilityExternalLibraries). Over main versions it could work, but we cannot guarantee the compatibility! We check an existing bootproject before loading against the type label of a target (VendorID, DeviceID and since v3.5.8.0 the DeviceVersion).

Retain-Variables out of a stored retain file (<Application>.ret) or SRAM must always be compatible. Per Retain Area a checksum is saved in a separate file, the same checksum is saved in the bootproject. This checksum is generated by the compiler out of the retain data and is an identification of all variables in the retain area with their respective types. If the checksum of the Retain data and the checksum in the bootproject do not match, there are several options in the Runtime System to select the behaviour. These optionas can be specified as settings in the cfg-File in the [CmpApp] section:

1. Bootproject.RetainMismatch.Init = 1: Bootproject is loaded and retains are initialized
2. Bootproject.RetainMismatch.Exception = 1: Bootproject is loaded, but the application will remain in stop and is set to an exception state. You can manually do a reset to heal this state.
3. No setting [DEFAULT]: Bootproject is not loaded and an error message is added to the logger.

Before Runtime Version 3.5.7.0 the Checksum was calculated out of the whole data of the application, with newer versions, the checksum is calculated only out of the data in the retain area. I.e. with newer versions, the retain data can be loaded to a boot project even if the project has changed, as long as these changes do not include changes in retain data.

7 Compatibility CODESYS Projects

First of all, opening existing projects in newer CODESYS versions must be possible without any data loss with any version combination.

Existing projects that are opened with a newer CODESYS version are kept in a “compatibility mode” as long as possible. That means, as long as the user only makes changes to the project which do not require new functionality, the storage format remains at the same version, so that the original CODESYS version is still able to open and edit that project without any loss of data. If the user makes changes which requires new data to be stored in the project, s/he will be interactively informed about that situation, giving him/her the possibility to undo the recent changes if a compatibility break would not be acceptable.

Opening newer project storage versions in older CODESYS versions is often - not always, this depends on the availability of plug-in types in the old version - possible, but strictly discouraged. A corresponding message will be displayed while loading such a project.

The user is also explicitly able to save a project for an older version of CODESYS using the “File :: Save As...” command. In that case, information will be dumped to the Messages view about which objects are suffering from data loss.

8 Compatibility CODESYS Automation Platform Interfaces

We guarantee that public AP interfaces, classes, delegates, structs, and enums are not changed any more after they have been released. We also guarantee that every AP SDK release does not contain any un-released types. That means, existing customer plug-ins can surely be executed in the context of a newer AP version without the need of re-compilation.

Furthermore, we do our best that the semantics behind the interfaces does not change in an incompatible way. As one could imagine, this is sometimes a tightrope walk, because even a bugfix could be considered a semantic change. However, due to our experience with typical customer requirements, we mostly succeed in balancing out an implementation change.

9 Compatibility CODESYS Compiler Version

With any version of the CODESYS Programming System you can select older compiler versions in the Project Settings - Compiler Settings (beginning with version 3.4.0.0). If you create a new Project, the compiler version will be set per default to the newest available compiler version. If an older project is opened with a newer Programming system, it is prompted to the programmer, that a newer compiler version is available, but the version will not be changed automatically.

We guarantee that the generated code will be the same with any newer version of CODESYS, if the compiler version is not changed. That includes compiler bugs resulting in wrong code. We guarantee that login without online change is possible, if a download was done with an

older version of CODESYS and the project is opened with a newer version of CODESYS, and the project is not changed. We guarantee that online changes are possible with a newer version of CODESYS, if the project contains changes, as long as an online change is possible at all. (I.e: if not the task configuration or the IO-Configuration is changed).

That means in reverse: if the compiler version is changed, we cannot guarantee that the generated code is exactly the same. If you update the compiler version, you have to test your application again. New error messages and new warnings may occur after updating the compiler version. Login on a device containing the generated code is not possible. Online Change is not possible. A new download will be necessary.

Note that the compiler version not only affects the compilation process itself, but also the language model that is produced by the objects. For example, since the IO-Configuration produces code, this code is produced according to the compiler version. Any automation platform plugin that produces code needs to consider the compiler version.

You should keep the compiler version for maintaining applications on existing machines. You should use the newest version available for creating new applications.

10 Compatibility CODESYS Compiler Version - AP Developers

Maintaining the compiler compatibility is not only a task for the compiler itself. It is a requirement for all Plugins in the automation platform. Every code change in an AP-Plugin that may have an effect on the generated code has to be done depending on the current compiler version. This is necessary in particular for all objects implementing the ILanguageModelProvider-interface and related interfaces.

11 Compatibility CODESYS Compiler Version and Libraries

If you write libraries for CODESYS, you should consider, that the library may be used with an older version of CODESYS. You then should use this older compiler version for checking and saving the library. Thereby you avoid using new language features that are not available in older versions (like for example the operator AND_THEN or ANY_TYPE). On the other hand, with newer compiler versions your library may produce new errors and new warnings. Therefore it may be useful to check the library also with newer Compiler versions. A compiled-library will use the compiler version as the project version when saved. That means for example, if you have the CODESYS installation 3.5.7.0 and the compiler version 3.5.5.0 then the compiled-library will be saved with all information that is available in CODESYS 3.5.5.0.

We guarantee that a library will successfully compile if used in a project, if the library was successfully checked with the same CODESYS compiler version. We guarantee that a compiled-library can be loaded with any CODESYS-Version equal or newer to the compiler version with which it was saved.

Note that the last statement is not valid for source libraries.

12 Compatibility CODESYS Visualization Profile

The so called Visualization Profile in CODESYS is used to harmonize all visualization related components (Libraries, Codegeneration, etc.) to one consistent version that works prefectly together. The visualization profile is nearly independent of the adjusted compiler version in CODESYS. If the visualization profile is not compatible to this compiler versions, you get a compiler error with a hint to switch to a newer compiler version.

Version History

Version	Description	Author	Date
0.1	Creation	AH	13.05.2015
0.2	Corrections after review	AH, KK, BW	30.10.2015
1.0	Release	AH	29.06.2015
1.1	Release	AH	30.10.2015
1.2	Release	AH	08.01.2016
1.3	Chapter 4 settings corrected Some small corrections	AH	14.06.2016
2.0	Release	AH	13.07.2016
2.1	Added legal note	TZ	23.08.2016
3.0	Release	MaH	23.08.2016
3.1	Legal note removed	GeH	24.10.2016
4.0	Release	MN	28.11.2016