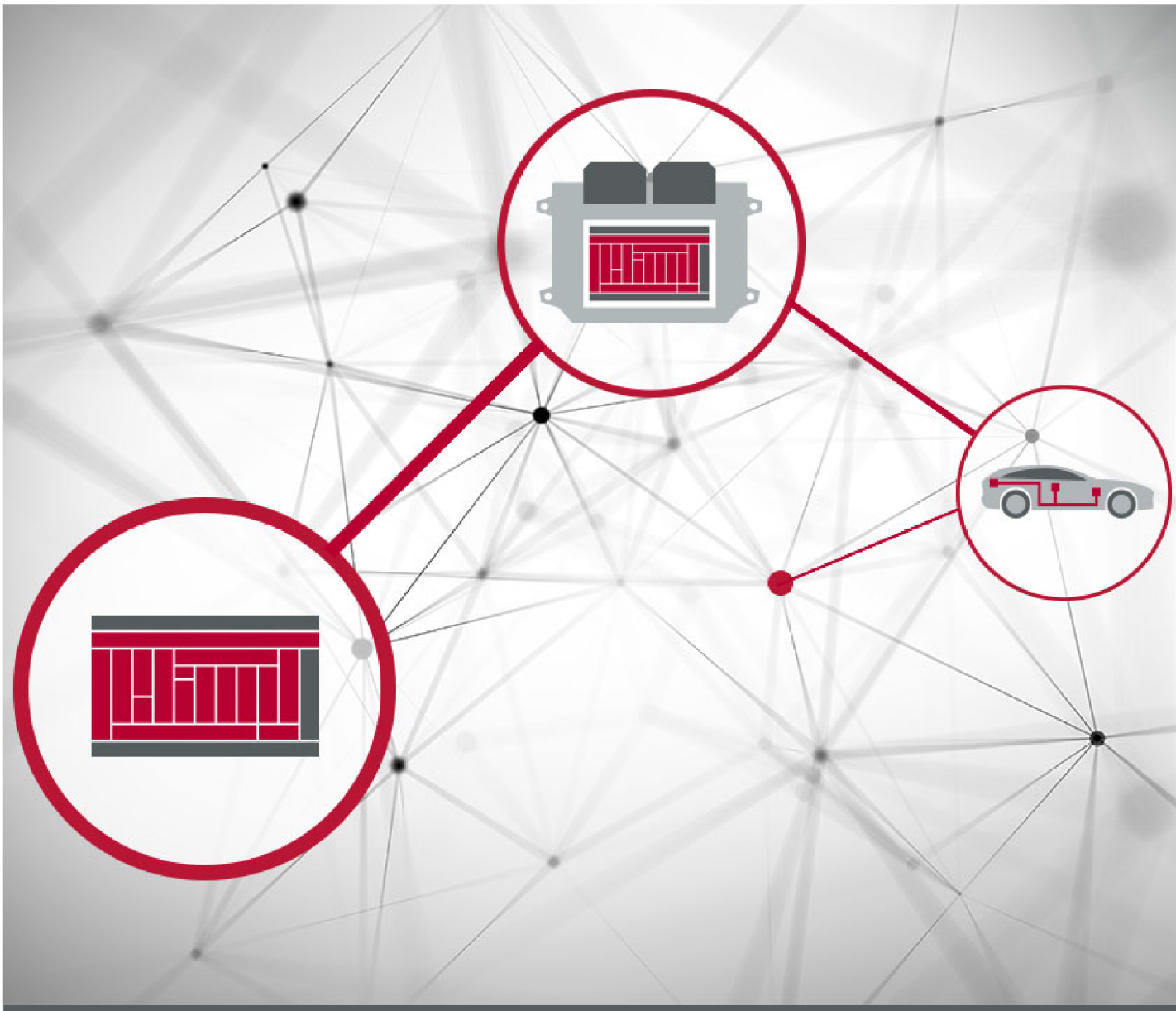
VECTOR



Startup with Vector SLP4

Version 13.0.1 for MICROSAR 4 Release 21

User Manual - A Step by Step Introduction

Content

1 About This Manual

1.1 History Information

Cross Reference

For detailed information about what's new, what's changed in current version, refer to section What's New, What's Changed? on page 193.

1.2 Finding Information Quickly

The user manual provides the following access help:

- You can see in the footer to which version the user manual refers,
- At the end of the user manual you will find an index to find information quickly,
- Also at the end of the user manual you will find a glossary of the used technical terms

1.3 Conventions

In the following two charts you will find the conventions used in the user manual regarding utilized spellings and symbols.

Style	Style Utilization
bold	Blocks, surface elements, window- and dialog names of the software. Accentuation of warnings and advices. **[OK]** Push buttons in brackets **File|Save** Notation for menus and menu entries
MICROSAR	Legally protected proper names and side notes.
`Source Code`	File name and source code.
Hyperlink	Hyperlinks and references.
<CTRL>+<S>	Notation for shortcuts.

Symbol	Symbol Utilization
	Here you can obtain supplemental information.
	This symbol calls your attention to warnings.
	Here you can find additional information.
	Here is an example that has been prepared for you.
	Instructions on editing files are found at these points.
	This symbol warns you not to edit the specified file.
	This icon indicates multimedia files like e.g. video clips.
	This icon indicates an introduction into a specific topic.
	This icon indicates text areas containing basic knowledge.
	This icon indicates text areas containing expert knowledge.
	This icon indicates that something has changed.

1.4 Certification

Vector Informatik GmbH has ISO 9001:2008 certification. The ISO standard is a globally recognized standard.

1.5 Warranty

We reserve the right to modify the contents of the documentation or the software without notice. Vector disclaims all liabilities for the completeness or correctness of the contents and for damages

which may result from the use of this documentation.

1.6 Support

<table>
<tr><td colspan="2">Germany
And all other countries, not listed here.</td></tr>
<tr><td colspan="2">☎+ 49 711 80670 3900
✉ support@de.vector.com</td></tr>
<tr><td>BR
Brazil, Argentina, Bolivia, Chile, Ecuador, Colombia, Paraguay, Peru, Uruguay, Venezuela</td><td>CN
China</td></tr>
<tr><td>☎ +55 11 5180 2350
✉ support@br.vector.com</td><td>☎ +86 21 6432 5353
✉ support@cn.vector.com</td></tr>
<tr><td>FR
France</td><td>IN
India</td></tr>
<tr><td>☎ +33 1 42 31 40 10
✉ support@fr.vector.com</td><td>☎ +91 20 6634 6634
✉ support@in.vector.com</td></tr>
<tr><td>IT
Italia</td><td>JP
Japan</td></tr>
<tr><td>☎ +39 2 678171 10
✉ support@de.vector.com</td><td>☎ TOKYO +81 3 5769 6972
☎ NAGOYA +81 52 238 5014
✉ support@jp.vector.com</td></tr>
<tr><td>KR
South Korea</td><td>Scandinavia
Sweden, Denmark, Finland, Iceland, Norway</td></tr>
<tr><td>☎ +82 2 807 0600
✉ support@kr.vector.com</td><td>☎ +46 31 764 76 00
✉ support@se.vector.com</td></tr>
<tr><td>UK
United Kingdom & Ireland</td><td>North America
USA, Canada, Mexico</td></tr>
<tr><td>☎ +44 121 7887 902
✉ support@uk.vector.com</td><td>☎ +1 248 449 9290, Option 2
✉ support@vectorcantech.zendesk.com</td></tr>
</table>

1.7 Trademarks

All brand names in this documentation are either registered or non registered trademarks of their respective owners.

1.8 Errata Sheet of Hardware Manufacturers

Caution! Vector only delivers software!

Your hardware manufacturer will provide you with the necessary errata sheets concerning your used hardware. In case of errata dealing with CAN, please provide us with the relevant erratas and we will examine whether this hardware problem is already known to us or whether to get a possible workaround.

Note

Because of many NDAs with different hardware manufacturers or because we are not informed about them, we are not able to provide you with information concerning hardware errata of the hardware manufacturers.

1.9 Example Code

Caution!

All application code in any of the Vector User Manuals is for training purposes only. They are slightly tested and designed only to understand the basic idea of using a certain component or set of components. Vector gives consent that you use it as a basis for developing your own code and modify it (Vector waives its copyright to forbid you the use), however, with regard to the fact that the code is designed for training purposes only and not for the use by you, usability of the code is not warranted and Vector's liability shall be expressly excluded in cases of normal negligence, to the extent admissible by law or statute. Of course you have to test the software developed on the basis of the code with diligent care before using the software.

1.10 What Do You Learn from This Manual

Use this document to

- Set up your project in just a few steps
- Learn how to use the Vector tools
- Learn how to work with the AUTOSAR means like software components, runnables, etc.
- Work with the Vector AUTOSAR Solution on a basic level
- Get a feeling of what you have to do when you start working for your actual project

The document is split in two parts, a **STEPbySTEP** description and an **Expert Knowledge (Concepts)** section.

- **STEPbySTEP**
  This section gives you a short overview how to set up your project in just a few steps. Use the step

by step description to start up with basic information and get your project basically running.

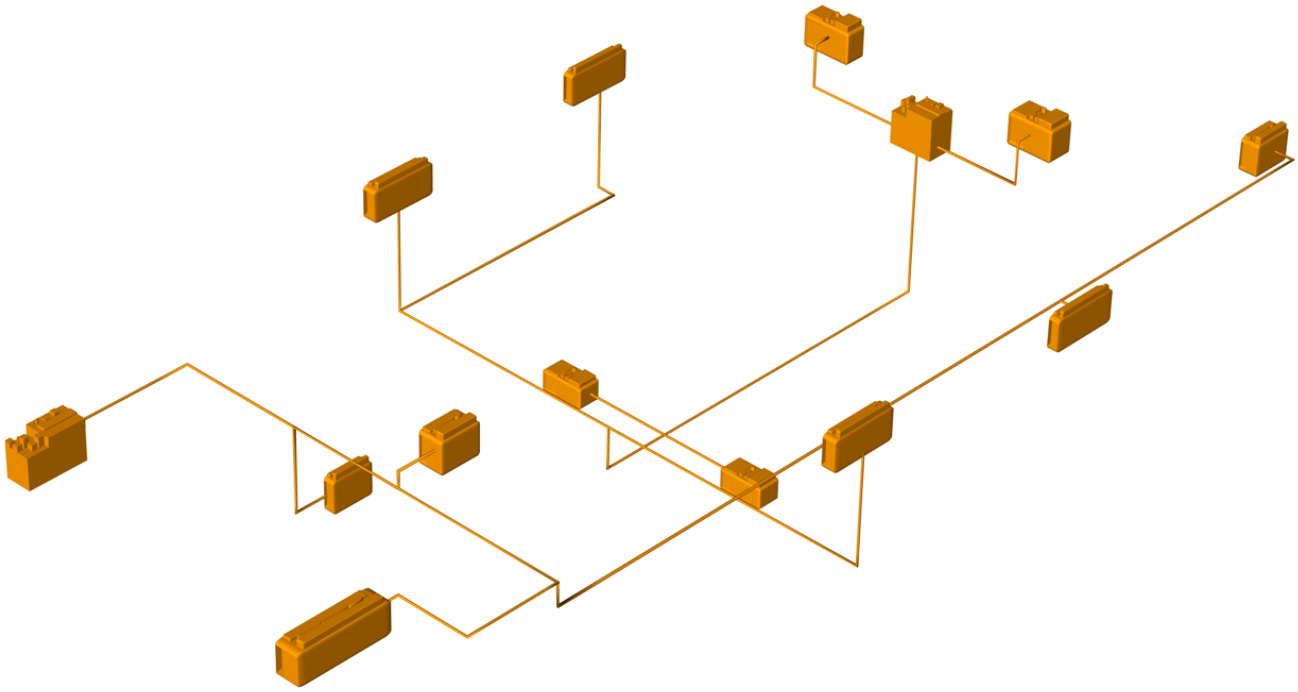
> **Expert Knowledge**
This section gives you a detailed information about working with Vector AUTOSAR Solution. It contains expert knowledge based on STEPbySTEP description sections and information about tool use and general AUTOSAR concepts.

2 Basics

2.1 An Overall View

What we are talking about is an ECU, a module to be built-in a vehicle. The ECUs have to communicate with other ECUs and therefore almost every ECU participates in a bus system like e.g. CAN, FlexRay, LIN, MOST or IP. Any ECU within one bus system has to provide an identical interface to this bus system, because all ECUs have to share information via this bus system.

The illustration below shows an example of a CAN network that connects the ECUs of a vehicle.



All ECUs are built-up in a very similar way. There is a software part to perform the main job (application) of the ECU, e.g. to control the engine or a door. The other part handles the network communication and diagnostics.

2.2 MICROSAR - Vector's AUTOSAR Solution

The Vector MICROSAR Solution provides efficient and scalable basic software modules and an RTE for AUTOSAR ECUs.

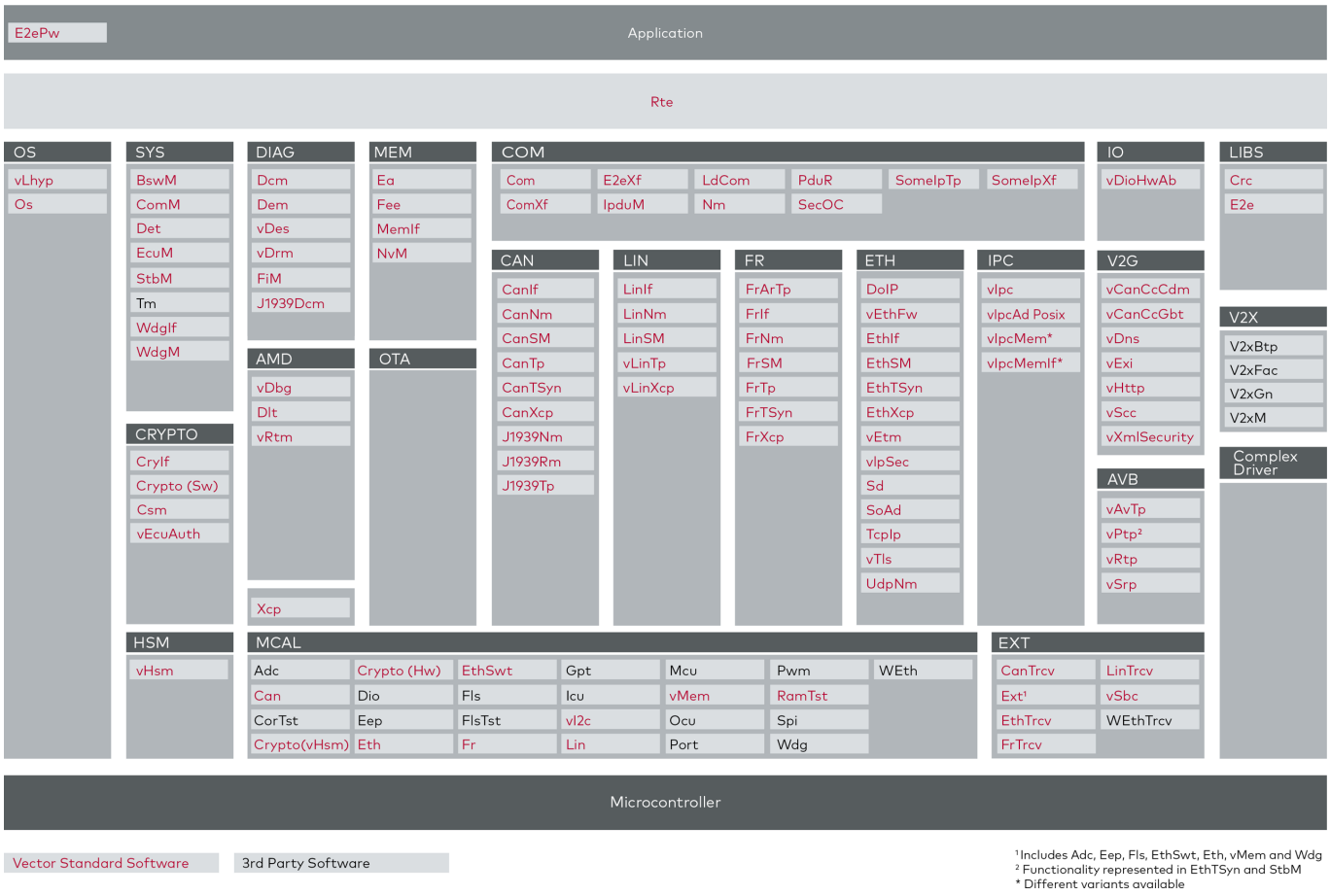
The individual AUTOSAR basic software modules are grouped into MICROSAR products that consists of related functions. The configuration of the MICROSAR basic software with DaVinci Configurator Pro is simple, user-friendly, and consistent. The included command-line-based generators create optimized code for your ECU based on the configuration.

Cross Reference

To get more information about the MICROSAR concept refer to General Overview on page 85.

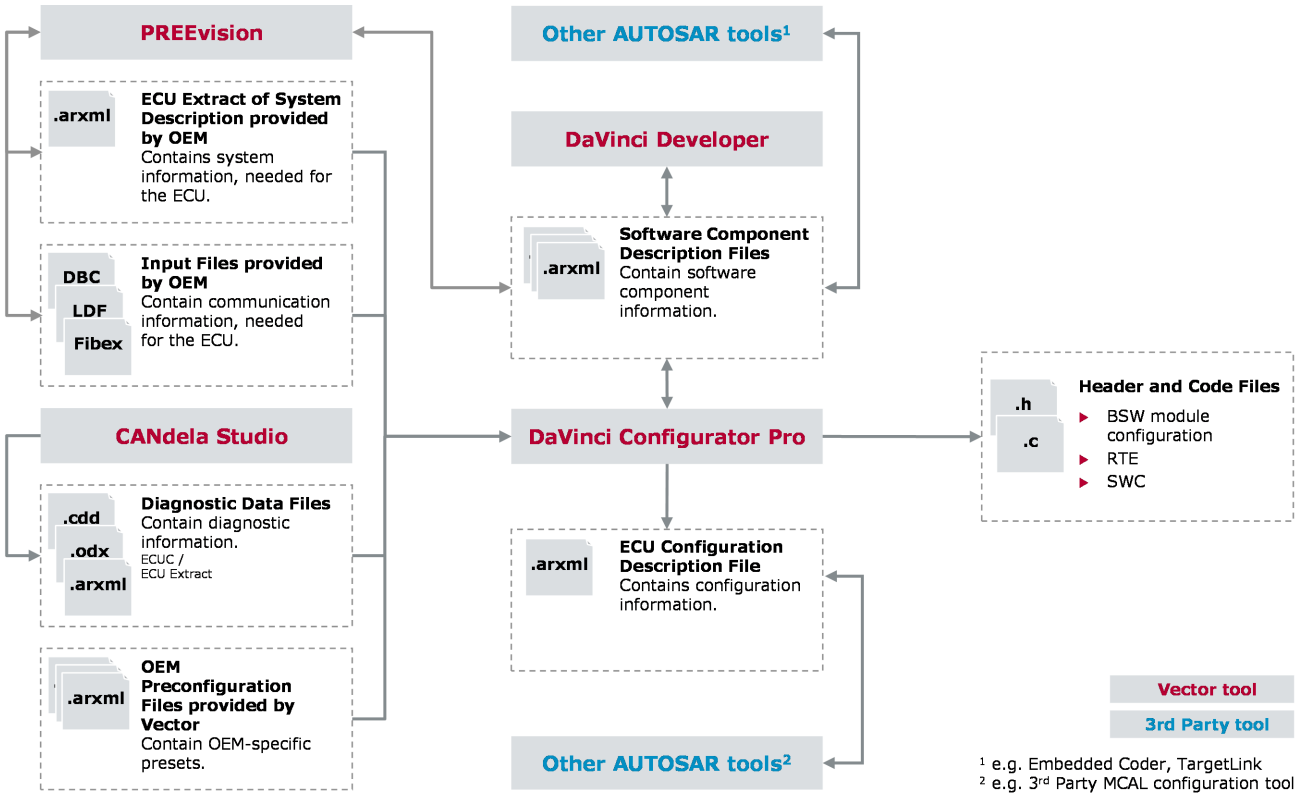
2.3 AUTOSAR Layer Model Vector SLP4

The following figure shows the AUTOSAR Layer model for Vector SLP4 .



2.4 Configuration Workflow Vector SLP4

The following figure shows the configuration workflow for Vector SLP4 :



I STEPbySTEP

This section shows you the few steps that are necessary to set up your project. Use it to start with basic information and get your project basically running.

1 STEP1 Setup Your Project

This section includes all information for setting up a new project. It starts with the situation after installation and provides you with all information about relevant tools.

Expert Knowledge

You need to know more? See section New Project on page 94.

1.1 Situation after Installation

- Before you start your first steps with Vector SLP4 , check whether you have installed all necessary tools and software.
- Have you read the **Installation Guide**? Have you received and installed all the necessary tools mentioned there?
- Have you installed the MICROSAR SIP?

NO? Then please read the installation guide carefully and follow the installation guidelines given there.

Cross Reference

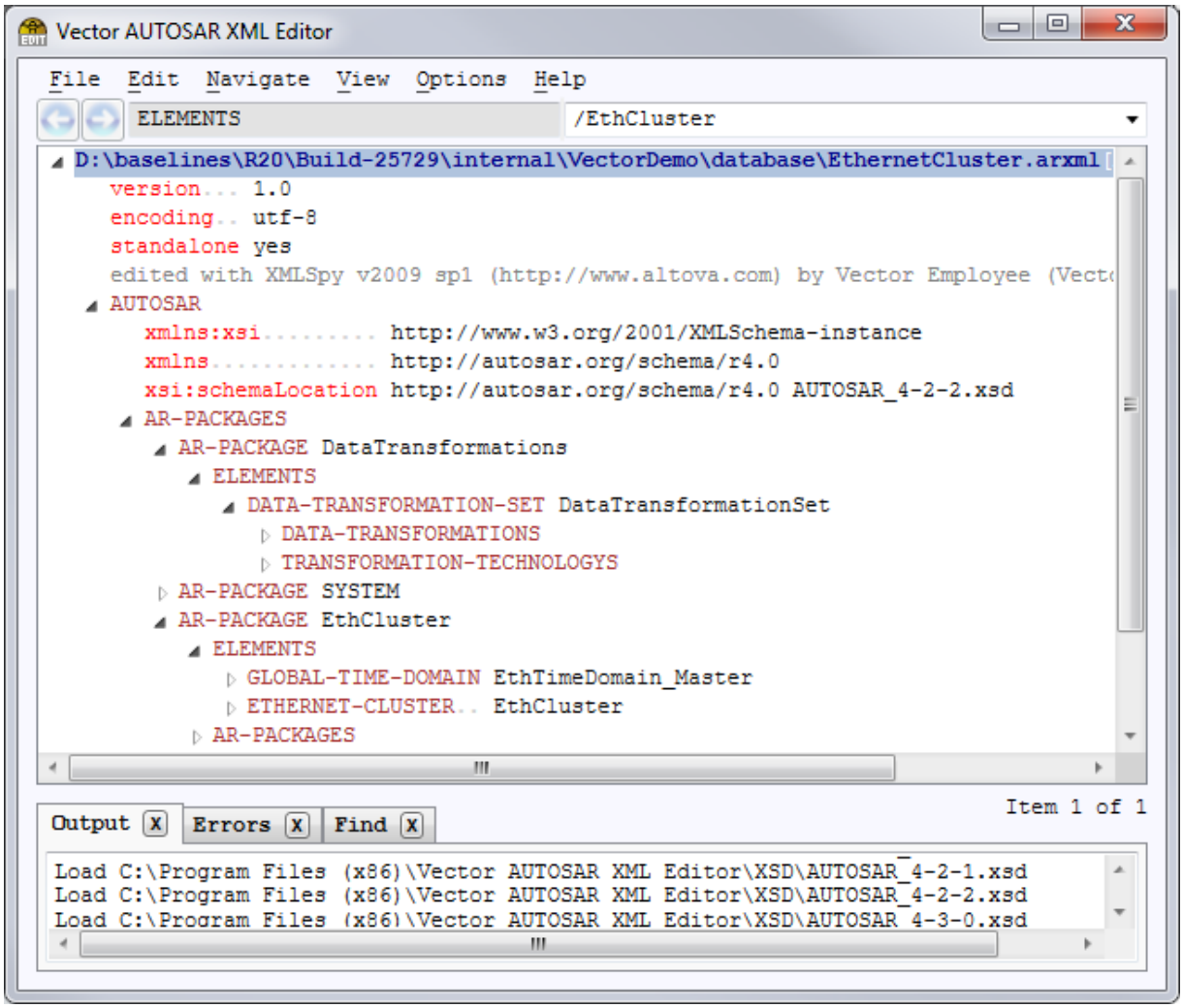
You will find the installation guide document at the Vector Knowledge Base:
https://vector.com/kbp/entry/640/

YES? You are ready to continue and to start with your first steps with the Vector SLP4 solution.

1.1.1 Vector AUTOSAR XML Editor - A Useful Tool

After the installation of the **ExternalComponentsSetup** you find the **Vector AUTOSAR XML Editor**, a low level XML editor in your program folder **C:\Programme(x86)\Vector AUTOSAR XML Editor**.

For more information about the tool, please read the user manual. You will find it via **Help|User Manual**.



1.2 Setup Project via DaVinci Configurator Pro

Start the DaVinci Configurator Pro to set up your project.

Note

The DaVinci Configurator is located in your SIP Folder:
<SIPFolder>\DaVinciConfigurator\Core\DaVinciCFG.exe

1. Open **File|New...**.
   A set of four configuration windows will open;
   General Settings,**Project Folder Structure**, **Target**, and **DaVinci Developer**.
2. Enter the necessary information for each window and click **[Next>]** to get to the next window.

Note for No-Communication Projects

For projects without communication modules like e.g. DEM only, just confirm the windows **Target** and **DaVinci Developer** as this information is not relevant.

Especially the **Target** window will show pull-down menus where **(undefined)** is preset and nothing else can be selected. This is not relevant for your use case and can be confirmed with **[Next]**.

1.2.1 General Settings

The following settings are required for project setup.

- **SIP Location**
  The SIP Location will be defined automatically based on the DaVinci Configurator Pro location and cannot be changed by the user!
  The DaVinci Configurator Pro is always located within the SIP.
- **Project Name**
  The name of your project (e.g. MyECU). It will be used as name of the generated ECUC files and for the **DaVinci Developer workspace**.
- **Project Folder**
  This is the root path of your project. Select an existing one or define a new one.
- **Author, Version and Description**
  Enter the name of the Author, the Version and a Description of the project.
- **Create Entries in the Start Menu**
  Set the checkbox **Create Entries in the Start Menu**. This is a comfortable feature that enables you to open your tools and project folder directly from the Windows start menu.

1.2.2 Project Folder Structure

Define your output paths here. You can use the given defaults or change paths to fit your project's needs.

Note

Default paths are based on the location of the defined project folder.

ECUC File Granularity

Additionally, you can specify whether the ECUC should be saved as one file (**Single File**), is split into several files (**One File per Module**) or is split into several files with a subfolder per file (**One File per Module (Subfolder per File)**.

Note

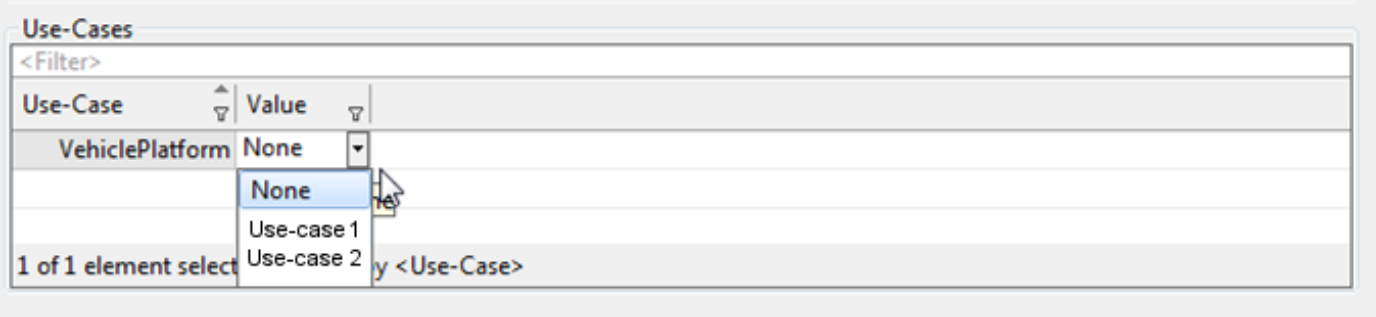
Split to several files is necessary to realize a multi user concept. Refer to the **AN-ISC-8-1208_DaVinciTeamAndPlatformSupport.pdf** located at <SIPFolder->\Doc\ApplicationNotes.

1.2.3 Target

Check **Derivative**, **Compiler** and **Pin Layout** (automatically set, based on SIP information).

If you want to use **Postbuild-Loadable** or **Postbuild-Selectable** for your project, activate the necessary support option.

If your delivery contains different use-cases, select the use-case that is suitable to your project.



1.2.4 DaVinci Developer

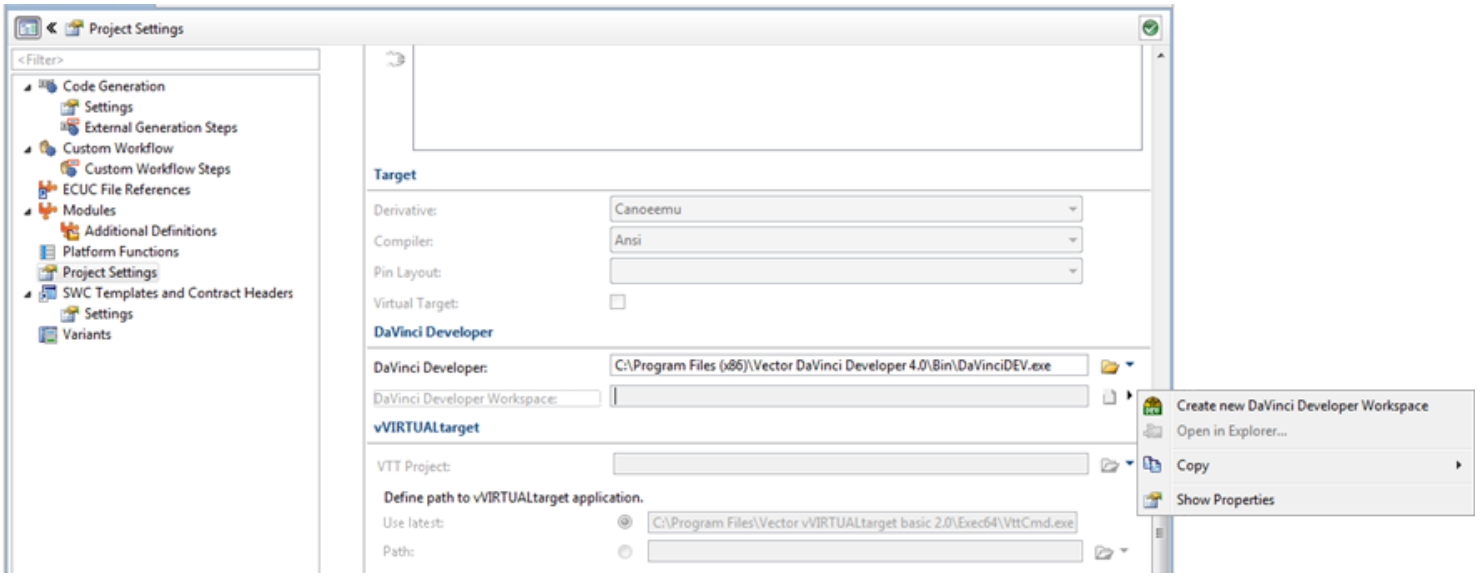
> **DaVinci Developer Workspace**
If you have a RTE and you use the DaVinci Developer select the checkbox **Create DaVinci Developer Workspace** and define the path to the DaVinci Developer executable and the path to the workspace.

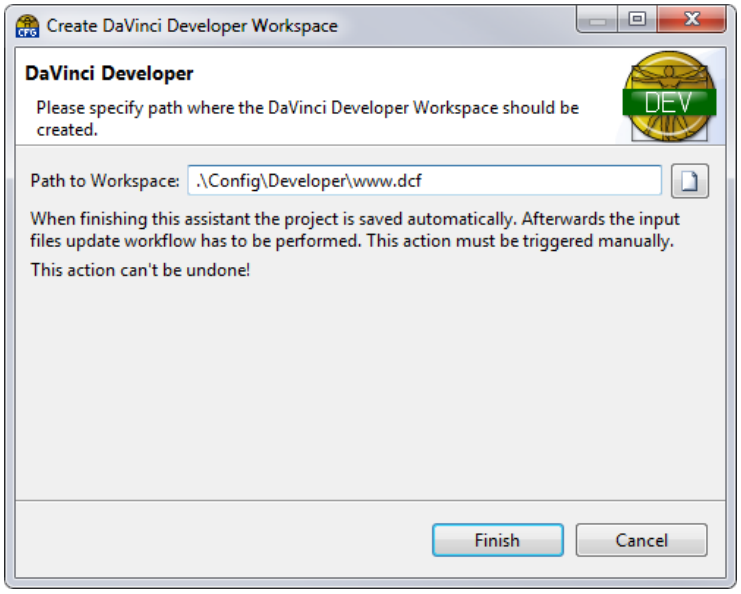
> **Automatic Update of DaVinci Developer Workspace**
Select **Enable Automatic Workspace Update** and define options that are used for updating the DaVinci Developer workspace during the project update process. Details see DaVinci Developer help.

1.2.5 Creating DaVinci Developer Workspace later

You can set up a project without having DaVinci Developer and in this case, you do not need a DaVinci Developer Workspace.

If you decide to use DaVinci Developer when the project is already created, DaVinci Configurator now can create the DaVinci Developer Workspace afterwards via **Project|Project Settings|Project Settings|DaVinci Developer**. Click the arrow to open the list of **DaVinci Developer Workspace** and click **Create new DaVinci Developer Workspace.** The window **Create DaVinci Developer Workspace** will open where you can enter the path for the workspace. Then click **[Finish]** and the workspace will be created.

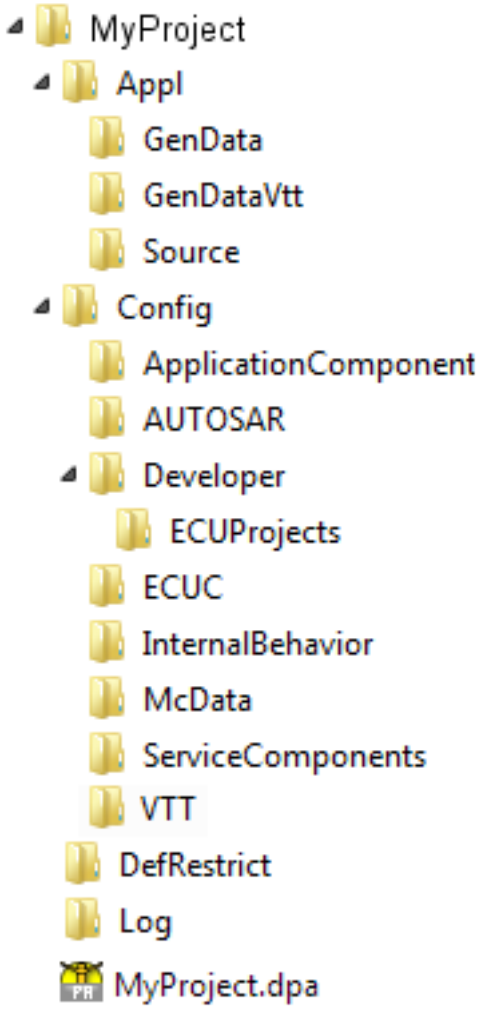




3. Click the button **[Finish]** to create your project.

1.3 Result Project Folder - Result of the Project Setup

The DaVinci Configurator Pro automatically creates and stores all relevant files to the project folder. The content of the folder looks like shown and described below.



1.3.1 Appl folder

The **Appl** folder contains the folders **GenData**, **GenDataVtt** and **Source**. In **GenData** the configuration tools generate their output files. The **Source** folder is meant to carry your source code files.

1.3.2 Config folder

The **Config** folder contains the following folders:

- **ApplicationComponents**
  This is a folder to put in additional SWC-Ts (Software Component Types). The DaVinci Configurator Pro reads in all *.arxml files of this folder.
- **AUTOSAR**
  The definition file of DataTypes (**PlatformTypes_AR4.arxml**) is centrally stored here.

- **Developer**
  All relevant project data for DaVinci Developer is located in this folder.
- **ECUC**
  This folder contains the created ECUC files. Before you add your input file(s), these files are only placeholders without content.

File Name	Description
<ProjectName>.ecuc.arxml	ECU Configuration Description
<ProjectName>.ecuc.Initial.arxml	Initial ECU Configuration Description for internal tool use only.

- **InternalBehavior**
  This folder contains the bswmd files for all activated modules in the DaVinci Configurator.
- **McData**
  Folder is used for storing measurement and calibration data.
- **ServiceComponents**
  The DaVinci Configurator stores all generated Service Component Prototypes to this folder.

Note

After you have added your Input file(s) and updated the configuration in the DaVinci Configurator, the **System** folder will be added to the **Config** folder.

- **System**
  The input file(s) and an additional created input file extract for communication use are stored to this folder.

Do not edit manually!

These files must not be modified manually!

- **VTT**
  In case of activated Virtual Target option, the vVIRTUALtarget project file is located here.

1.3.3 <ProjectName>.dpa

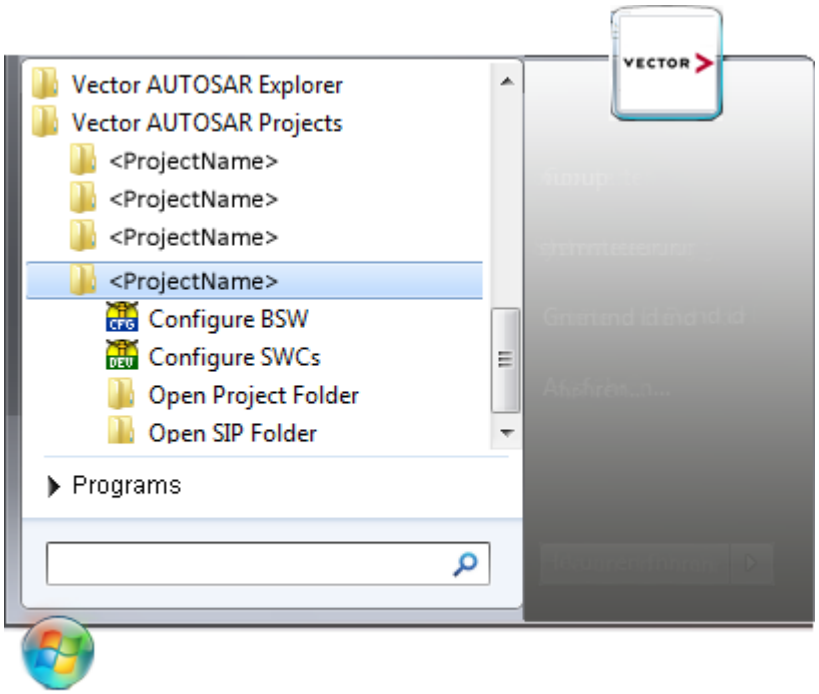
The Project File <ProjectName>.dpa includes all relevant project data. Additionally the context menu of this file is the control center to open your tools with already loaded projects.

1.3.4 Log Folder

Folder for the generated log files.

1.4 Start Menu - Result of the Project Setup

The DaVinci Configurator Pro creates a link to start the tools with already loaded configuration. You will find this link in **Start | Programs | Vector AUTOSAR Projects | <ProjectName>** if **Create entries in the startmenu** was selected while project setup.



Info

You can also use the ***.dpa** file to start your project with an already loaded configuration. Therefore right-click on your ***.dpa** file in the Windows Explorer and select **Configure BSW** to open the DaVinci Configurator Pro with an already loaded configuration or **Configure SWCs** to open the DaVinci Developer for Software Design and Data Mapping.

1.5 DaVinci Configurator Pro Project

After finishing project set-up, a new empty project is loaded within the DaVinci Configurator Pro.

Note

Editors to configure modules will be available after input file import (see Add Input Files on page 29) or after module activation (see Activate Your BSW Modules on page 40).

2 STEP2 Define Project Settings

Before you start with the configuration, it is necessary to define some project-specific data. This is the definition of project-specific data and the definition of all necessary input file(s) for communication and diagnostic purpose. Additional external generation steps and the activation of the necessary Basic Software Modules have to be done in the DaVinci Configurator Pro Project Settings view before.

Expert Knowledge

You need to know more? See section Project Settings on page 102.

Cross Reference

If you want to use variant handling within your project, you have to define your variants before adding input files. For detailed information about defining project settings for variant handling refer to Variant Handling on page 169.

Note

In case of OS only projects, skip the next sections **Add Input Files**, **External Generation Steps** and **SWC Templates and Contract Phase Headers** and go on with Activate Your BSW Modules on page 40.

2.1 Add Input Files

Open Input Files editor via **Project | Input Files** or use the **Input File** button of DaVinci Configurator Pro toolbar.

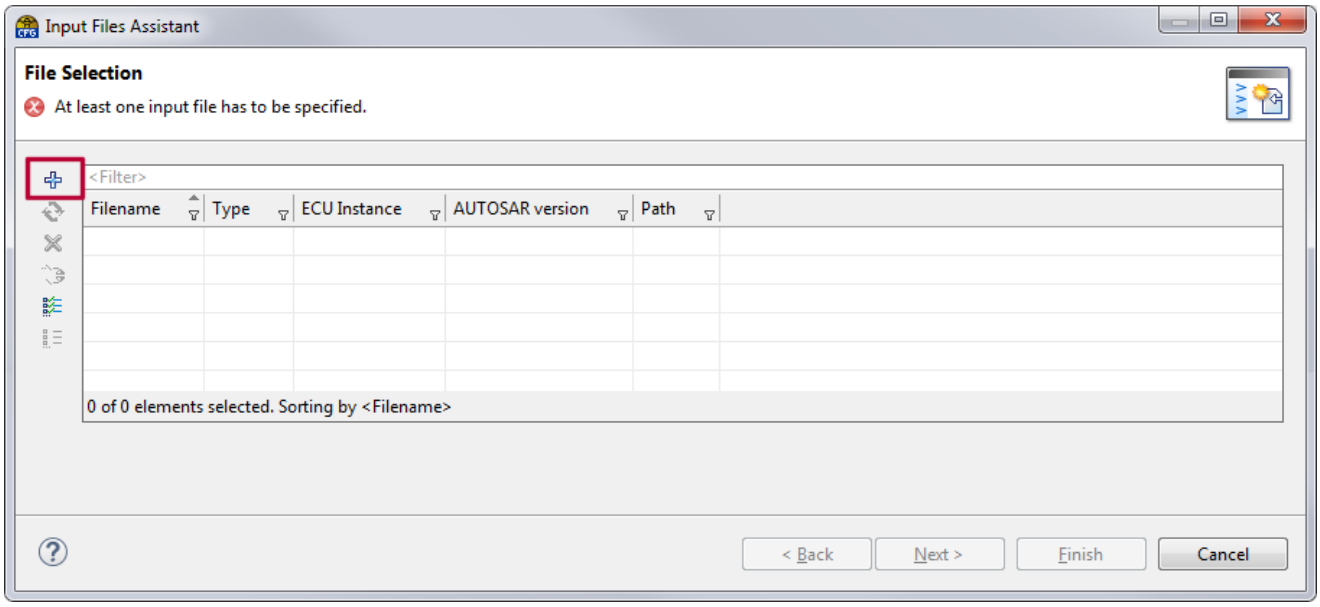
Note

Any change of the input files requires an update of the configuration. Add all relevant files before and finally start the file processing Start file preprocessing in the **File processing** view of the **Input File Assistant**.

2.1.1 Input File Assistant

Start the Input File Assistant Open the Input Files Assistant. to enter all types of input files:

- System Description files (ARXML)
- Legacy files like (DBC, Fibex, ...)
- Diagnostic data base files (CDD, ODX, ...).

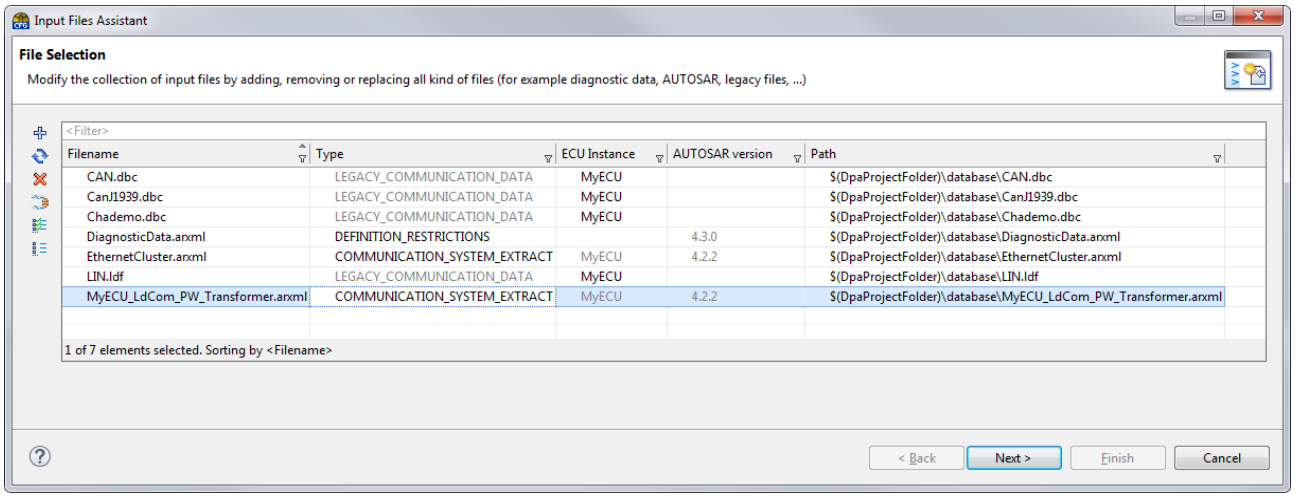


2.1.2 Add System Description Files

In the **File Selection** click ✚ to browse for your System Description File. Decide about **AUTOSAR Files (arxml)** or **Legacy Files (dbc, xml, ldf)** or both.

Note - Only necessary if DBC is used as input file

There is a document TechnicalReference_DbcRules_Vector.pdf that describes necessary DBC rules to help you creating your DBC file. You find this document in the TechnicalReference folder of your delivery.



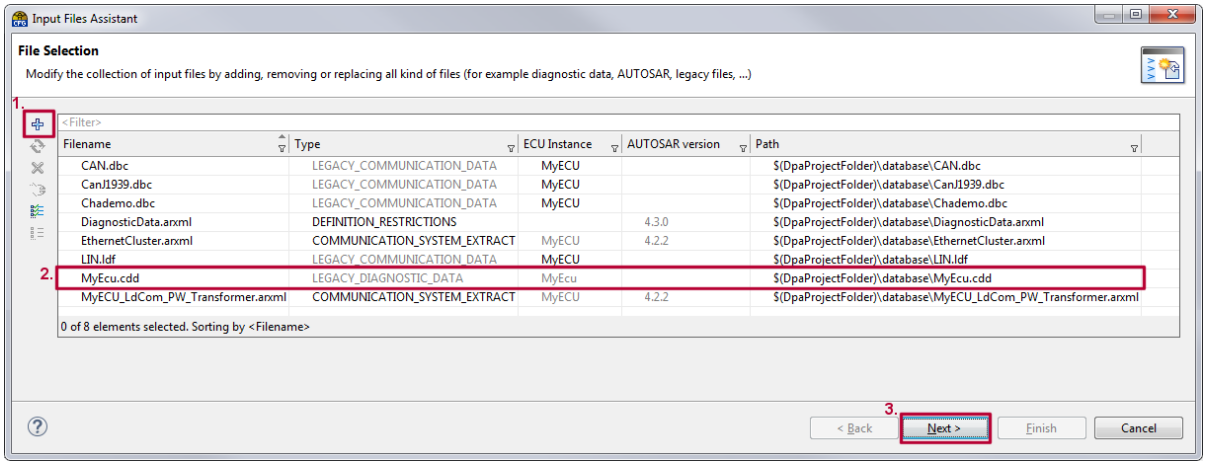
Click **[Next]** if you do not need any diagnostic input files and get to the **Overview configuration** of the Input File Assistant.

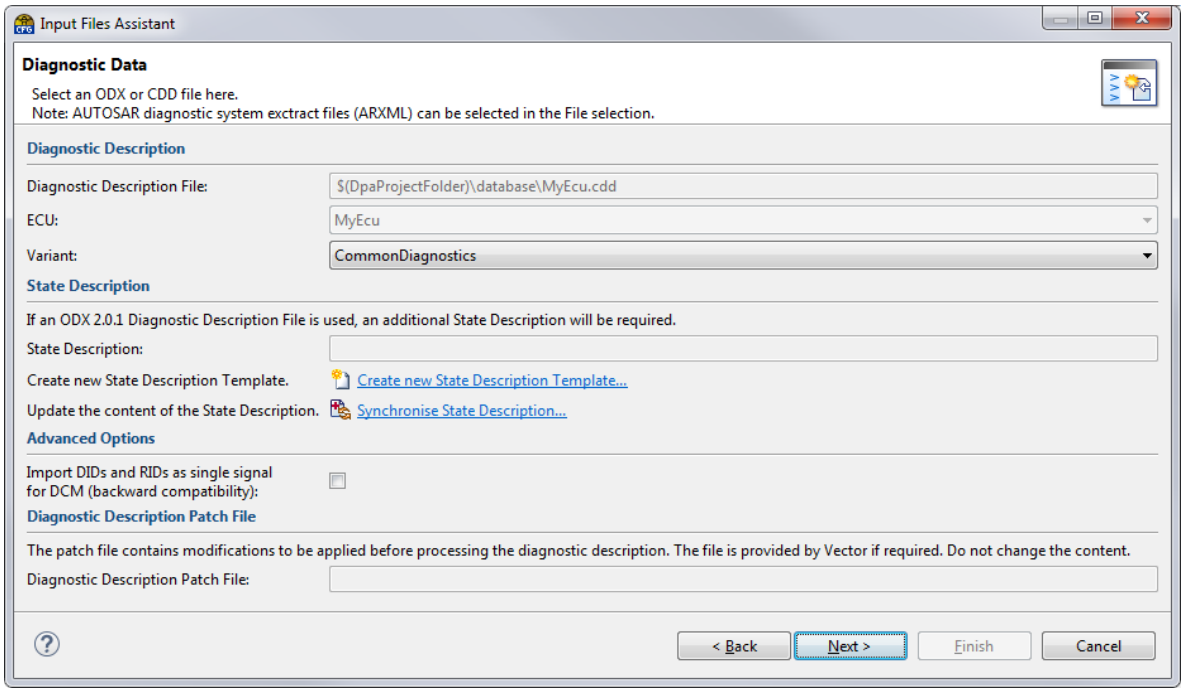
2.1.3 Add Diagnostic Data Files

If diagnostic is used, add the diagnostic data file(s) in the **File Selection** of the **Input Files Assistant**. If not, you can skip this section. There are different use cases to import diagnostic data, decide if you import your data from a diagnostic data file (use case 1 just below) or from an already existing configuration (use case 2).

Use Case 1: Import Diagnostic Data from ODX, PDX or CDD File
Use **File Selection** in the Input File Assistant

1. Click ✚ to browse for the **Diagnostic Description File** (*.cdd/*.pdx/*.odx).
2. Select the **ECU Instance**
3. Click **[Next]** and the **Diagnostic Data** view is shown
4. Select the **Variant**

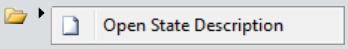




Note

If the **Diagnostic Description File** is ODX/PDX 2.0.1, you have to create a state description template (*.csv), because the dependencies between services and sessions must be described in an additional State Description file.

Therefore select **Create State Description Template...**, enter path to the state description file and confirm with **[OK]**.

Use **Open State Description File** 

to edit the created state description in standard editor (usually Microsoft Excel).

Edit the states of the defined sessions and security accesses.

Example

The following example shows the structure of the state description file.

	A	B	C	D	E	F	
1	StateGroup		Session	Session	Session	Session	Sessio
2	Prefix	Service	Default	Programming	Extended	EndOfLine	Develc
3	10 01	DiagnServi_DiagnSessiContrDefaultSessi	(yes)	Default	Default	Default	Defaul
4	10 02	DiagnServi_DiagnSessiContrECUProgrSessi	(no)	(yes)	Programming	(no)	Progra
5	10 03	DiagnServi_DiagnSessiContrExtenSessi	Extended	Extended	(yes)	Extended	Extenc
6	10 04	DiagnServi_DiagnSessiContrVWEndOfLineSessi	Session04	(no)	Session04	(yes)	Sessio
7	10 40	DiagnServi_DiagnSessiContrVWEndOfLineSessi	EndOfLine	(no)	EndOfLine	(yes)	EndOf
8	10 41	DiagnServi_DiagnSessiContrDevelSessi	Session41	(no)	Session41	Session41	(yes)
9	10 4F	DiagnServi_DiagnSessiContrDevelSessi	Development	(no)	Development	Development	(yes)
10	11 01	DiagnServi_ECUResetHardReset	(no)	Default	(no)	(no)	(no)
11	11 02	DiagnServi_ECUResetKeyOffOnReset	Default	(no)	Default	Default	Defaul
12	14	DiagnServi_ClearDiagnInfor	(yes)	(no)	(yes)	(yes)	(yes)
13	19 01	EventAndFaultMemoryManagement_ReportNum	(yes)	(yes)	(yes)	(yes)	(yes)
14	19 02	DiagnServi_ReadDTCInforReporDTCByStatuMask	(yes)	(no)	(yes)	(yes)	(yes)
15	19 04	EventAndFaultMemoryManagement_ReportDTCS	(yes)	(no)	(yes)	(yes)	(yes)
16	19 06	DiagnServi_ReadDTCInforReporDTCExtenDataRe	(yes)	(no)	(yes)	(yes)	(yes)
17	22 01 00	Combined_Measurement_ReadDataByIdent	(yes)	(no)	(yes)	(yes)	(yes)
18	22 01 02	Combined_Measurement_ReadDataByIdent	(yes)	(no)	(yes)	(yes)	(yes)
19	22 02 85	Combined_Measurement_ReadDataByIdent	(yes)	(no)	(yes)	(yes)	(yes)

Service Prefixes
The first column (denoted Prefix) contains the service prefixes.

The configuration values given in each line will be applied for all services with a matching prefix. Lines for which there is no matching service defined within the ODX data will be ignored.

State Name
The second row starting with the third column contains the names of the states. If necessary add your new states here (the names may only contain A-Z, a-z and underscores).

State Group
For each state the name of the corresponding state group is written in the first row (The names may only contain A-Z, a-z and underscores). The session state group must contain the**session** describing (e.g. **SessionStateGroup**, **MySessionGroup**) and the state group

describing. Security access must contain security in its name (e.g **Security_Access**, **Mysecurity_StateGroup**).

Never use "session" and "security", at the same time, in a state group name!

Diagnostic sessions will be extracted from the ODX file (if the respective SDGs are included) during the creation of the state description. For each diagnostic session defined in the ODX file there will be one column belonging to the session state group. Security levels cannot be extracted from the ODX file. You must either specify them manually or use the OEM specific template provided by Vector when initially creating the state description (see above). One security level which describes the default state of the security access state machine (e.g. "LOCKED") is always mandatory. For each instance of service 0x27 (SecurityAccess) there must exist one corresponding column belonging to the security state group.

States
State cells can contain a (yes), (no) or state name.

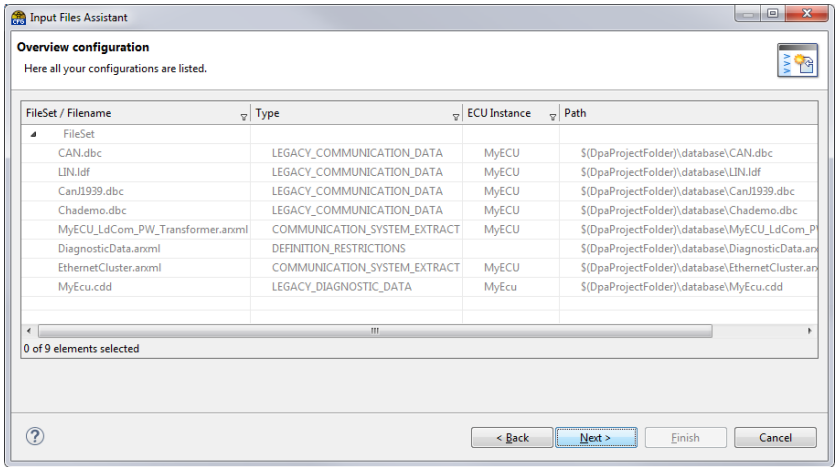
(yes) means that the service is executable in the current state, but there is no state transition.

(no) means that the service is not executable in the state.

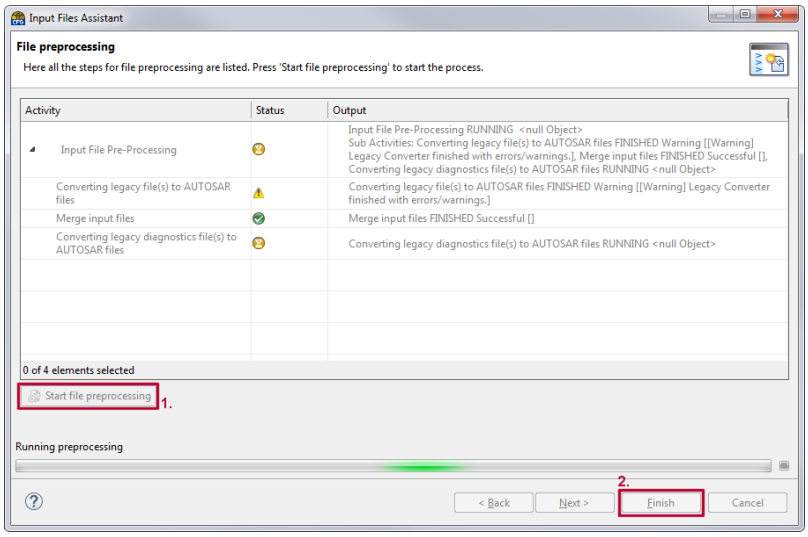
A state name indicates that the service is executable in the current state and that the ECU is in the named state after sending the request.

A target state always has to be contained in the same state group as the source state.

Click **[Next]** to get the **Overview configuration**.



Click **[Next]** to get the **File processing**.



Here click Start file preprocessing first to check all your Input Data Files. If the checks are all right, click **[Finish]**.

Use Case 2: Import Diagnostic Data From an Existing Configuration
Open your project in the DaVinci Configurator and go on with the following steps to import your diagnostic data from an existing Configuration.

1. Start Import Assistant via **File | Import**
2. Add ECUC File and go on with **[Next]**
3. Select modules which should be imported and close the **Modul Configuration Import** dialog via **[Finish]**.

2.1.4 Add Standard Configuration Files

In some cases an OEM-specific preconfiguration is necessary, therefore add the Standard Configuration File provided by your OEM. Add fragments of ECUC modules which will be used as project specific mandatory configuration.

2.1.5 Define Options for Input Files

Switch to the **Options** view of the Input Files via Input Files Overview | Options.

Object Identification during Update

You can ignore the function of identifying object bases on their UUID to enforce an object identification based on AUTOSAR path for System Description Files or Standard Configuration Files.

System Description Modification

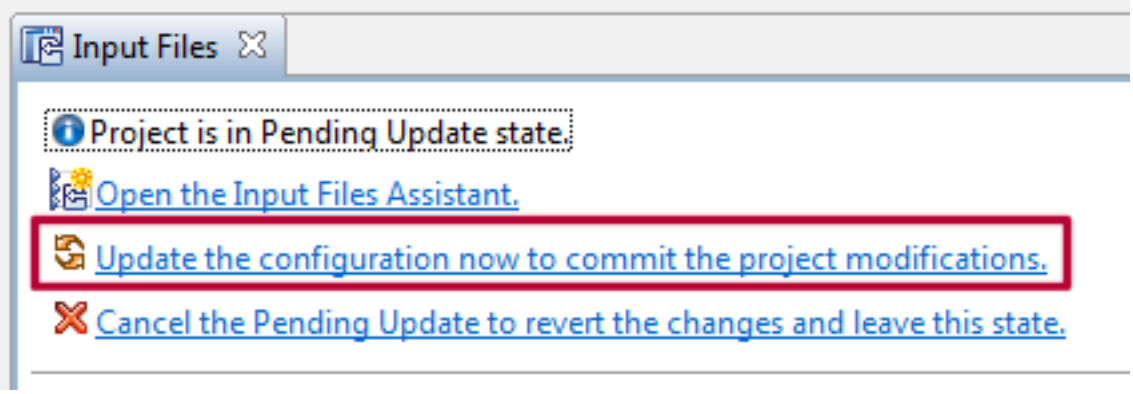
Sometimes the system description or the legacy formats are incomplete or some changes are necessary before the ECUC is generated. These kind of changes have to be done after every update of the underlying data base. You can automate this completion or modification by activating the VASE (Vector AUTOSAR Scripting Engine) scripts and by selecting a specific script that performs those modifications. The script will be run automatically during the project update process, just before the call of the BaseEcucGenerator.

For some cases Vector offers default VASE scripts within the SIP delivery. Check if VASE scripts are provided in **<SIP Location>/DaVinciConfigurator\VASE-Scripts**. Are these files necessary for your project? Then activate the **System Description Modification** check box and enter **Script File Path**.

2.1.6 Update Configuration

Any change of the input files lead to Pending Update state of the project, this requires an update of the configuration.

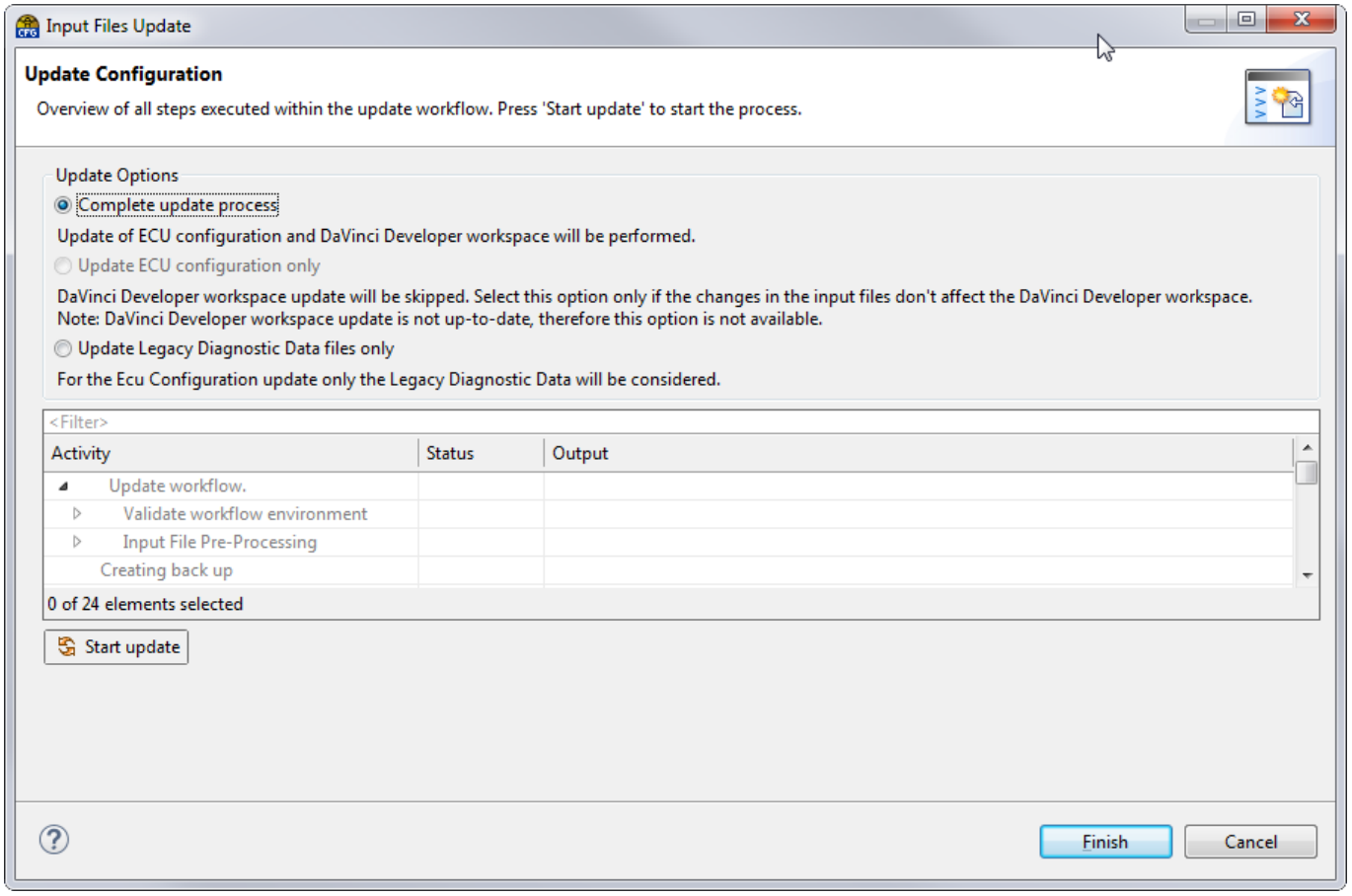
Update the configuration via in the **Input Files** view.



What happens during the configuration update process?

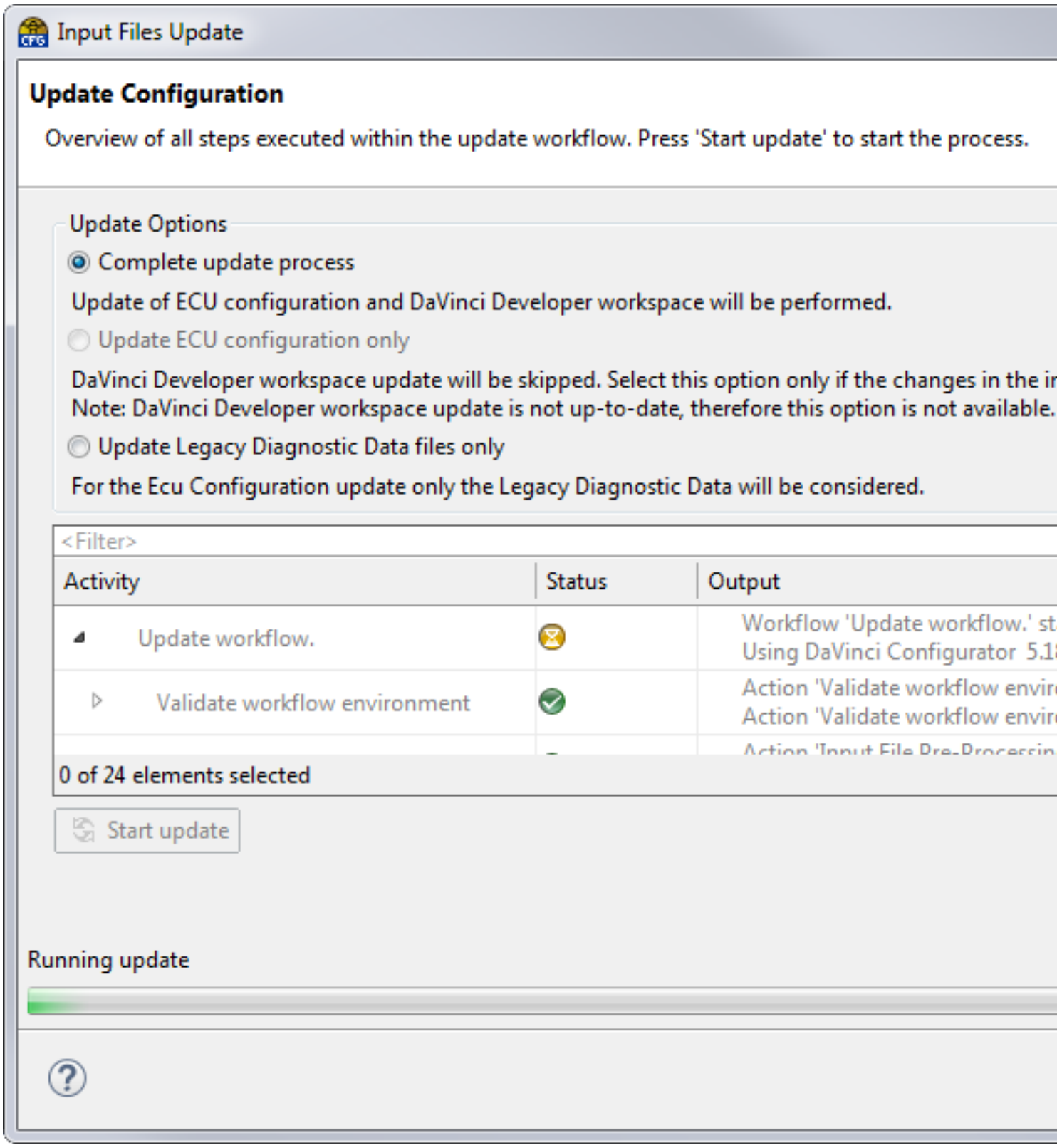
Before the update dialog opens, you will be informed, that the project will be closed for update configuration.

Before the update starts the **Update Configuration** message opens, select **Complete update process** and go on with **[Yes]**.



Note

If you do not know whether your changes will affect the DaVinci Developer workspace, always select **Complete update process**!



The project file will be updated based on loaded input/diagnostic files. Additionally the DaVinci Developer will be opened and updated manually while configuration update. If you selected **Complete update process**, the DaVinci Developer update will run in the background.

The result can be seen as comfortable HTLM report that is stored in your project **Log** path.

Note

After the first update/import of data bases all required modules are automatically enabled. This information is derived from the data base. If further modules are required or some shall be removed please refer to section Activate Your BSW Modules on page 40.

2.2 Define External Generation Steps and SWC Templates and Contract Phase Headers

To define external generation steps/SWC templates and contract phase headers, open **Project Settings Editor** via **Project | Project Settings** or use the **Project Settings** icon of DaVinci Configurator Pro toolbar.

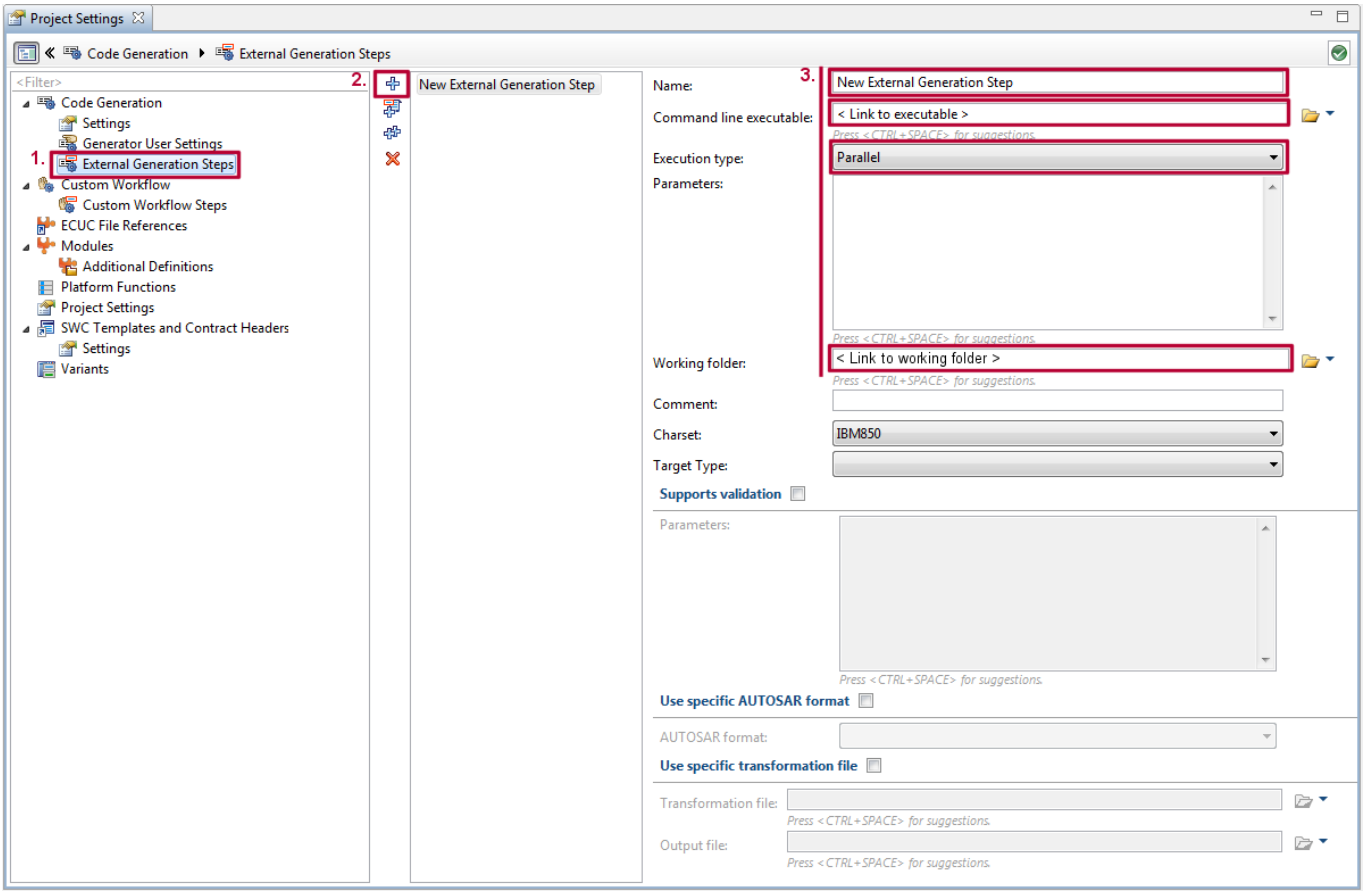
2.2.1 External Generation Steps

You have external software that needs to be started during the code generation of DaVinci Configurator Pro? Or just before? Or afterwards?

Then link your software to the **External Generation Steps** of DaVinci Configurator Pro.

How? Follow these steps.

1. Select **Code Generation** | **External Generation Steps**
2. **Add External Generation Step**
3. Define
   Name,
   Command line executable,
   Execution Type and
   Working folder.



2.2.2 SWC Templates and Contract Phase Headers

To generate software component templates and/or contract phase headers perform the following steps:

1. Select **SWC Templates and Contract Headers | Settings** and choose the **Generation Mode**
2. Go to **SWC Template and Contract Headers** and tick the checkbox of the software components for which you want to generate SWC templates and/or contract phase headers

Cross Reference

For the generation of the SWC templates and contract phase headers please refer to STEP7 Code Generation on page 78.

2.3 Activate Your BSW Modules

Before you start configuring your Basic Software, you have to activate all modules you need for your project. Therefore open **Project Settings Editor** via **Project** | **Project Settings** or use **Project Settings** icon at DaVinci Configurator Pro toolbar.

Select **Modules** to see all modules currently activated for your project.

Info

It depends on your Input File which modules are loaded initially.

You need additional modules?

1. Click and select the source of the module definition.
2. Select the module you would like to add to your configuration and confirm with **[OK]**.

Info

Multiple selection is possible for activating or deactivating modules.

Not all modules are needed?

To delete not used modules, select the module and delete it via .

Caution!

It is not recommended to delete initially activated modules from project!

You have a completely configured module and would use this for the current configuration?

1. Make sure, that the module you would like to import and use it, is deactivated in your current project.
2. Select **File** | **Import**
3. Add ECUC file including the configured module
4. Click **[Next]**
5. Select the module configurations to be imported

Note

It is not allowed to import BSW modules which are currently activated.

6. Click **[Finish]**

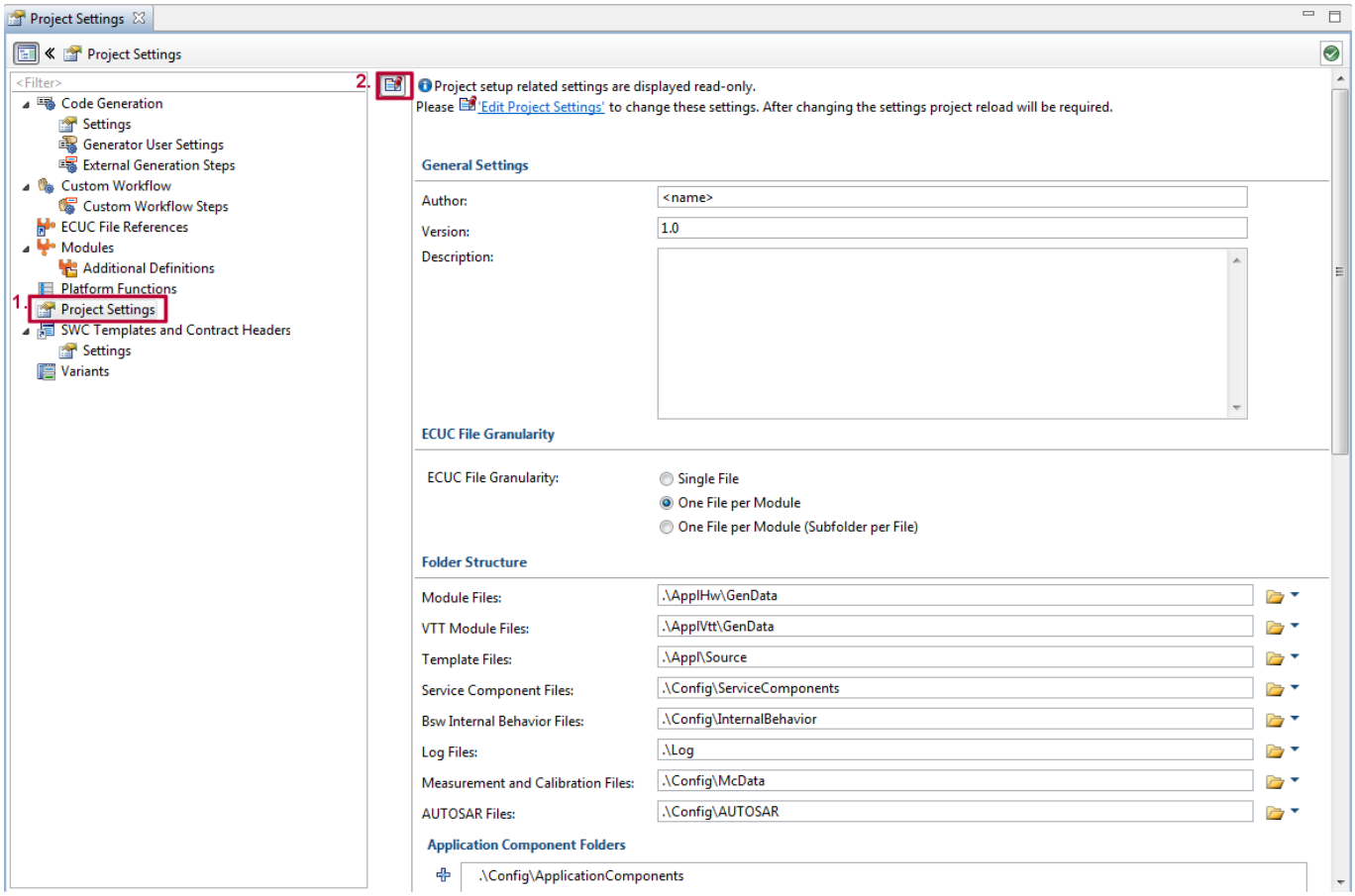
2.4 Add ECUC File References

In some cases it is necessary to add an ECUC reference file, to include mechanisms for module configuration from other projects.

1. Select **File** | **Add ECUC File Reference...**
2. Add the ECUC file of the module configuration you want to refer
3. Click **[Finish]**
4. Open **Project Settings Editor** and **ECUC File References** view, select reference and make path relative via **Make path relative to...** icon.

2.5 Change Project Settings

In some cases it is necessary to change project settings, which were defined while project setup. Therefore select **Project Settings** and activate editing via **Edit Project settings** icon.



2.5.1 Postbuild Support

Caution!

These parameters should be changed for compelling reasons only. Parameter changes require a lot of manual adjustment after project update!

Note

In case of project migration, the parameters will be set automatically.

3 STEP3 Validation

Start initial validation process to solve the first warnings and errors.

The DaVinci Configurator Pro provides validation routines that run in the background and are called Live Validation. At the beginning of the configuration, the project is incomplete and much information in the system is still missing. Therefore the DaVinci Configurator Pro offers a powerful solving algorithm.

Expert Knowledge

You need to know more? See section Validation on page 105.

3.1 Start Solve All Mechanism

First of all start solving mechanism via the **Solve All** button  at the validation view of the DaVinci Configurator.

Note

Not all errors can be fixed with **Solve All** mechanism.

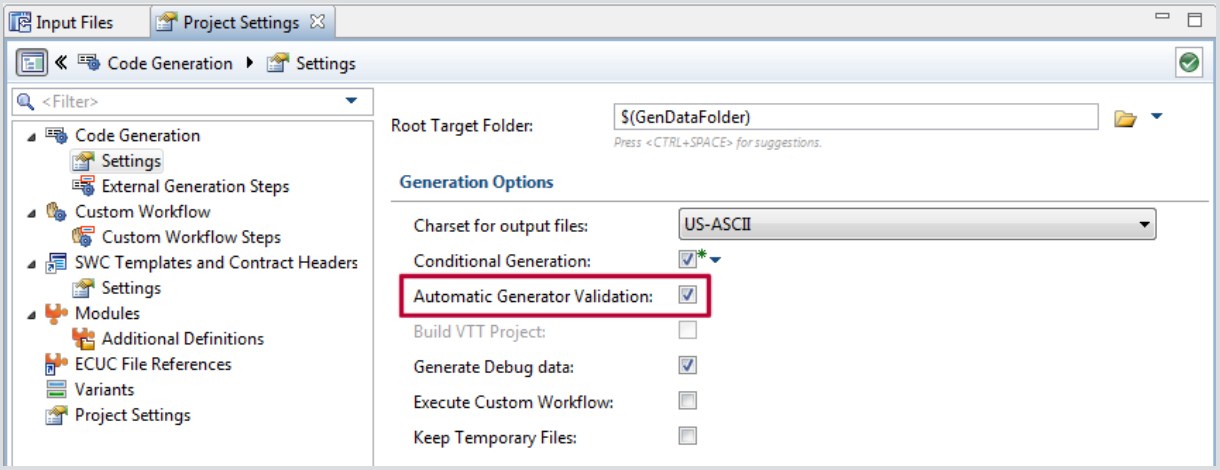
Some errors might have two or more solutions, these errors cannot be fixed automatically and need your decision.

3.2 Live Validation - Solving Actions

The DaVinci Configurator Pro provides a live validation during project configuration. For some parameter errors the live validation process offers possible solution(s) in the validation view of the DaVinci Configurator Pro. These errors are marked with a bulb.

Note 3rd Party Module Live Validation

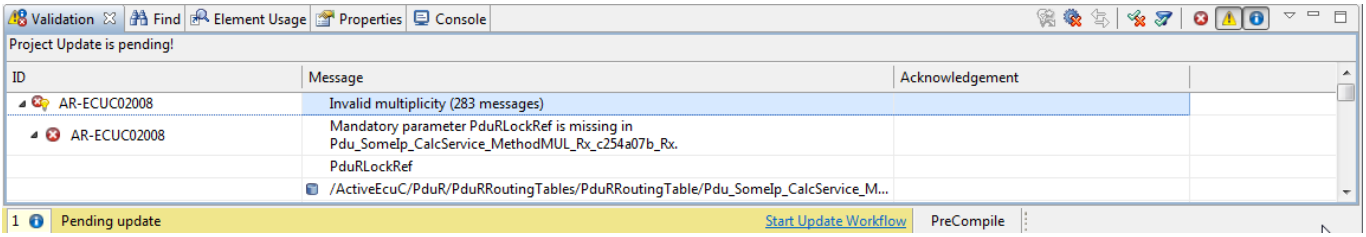
If **Automatic Generation Validation** is activated in **Project Setting Editor**, the configuration of 3rd party modules is also checked while live validation.



Check if the suggested solving action is correct. Double click the error message to open the configuration window of the parameter.

Solving Action is correct?

Start the solving action via double click on the solving action entry in the validation window.



Note

If the solving action is not correct, you can additionally use the link in the solving action to navigate to the parameter and configure the changes manually.

Cross Reference

You find more information about the feature Acknowledgment in the help of the DaVinci Configurator Pro.

4 STEP4 Start BSW Configuration

Start the configuration of the BSW modules now. The target of this first description is a basically working configuration, using a minimal necessary configuration.

Expert Knowledge

You need to know more? See section Configuration with Configurator Editors on page 106.

Note

Changes in Configurator Editors could lead to errors during live validation!

If these errors include solving actions, check and start solving action in the validation view

Cross Reference

If you use variant handling within your project, you have to refer to Variant Handling on page 169 for special configuration information.

4.1 Start Configuration with Configuration Editors

The configuration editors are use case-oriented and optimized for displaying or configuring those parts of the ECU configuration, which are related to the use case.

4.2 Base Services

The Base Services domain contains configuration editors for the configuration of the development error reporting, general purpose timer channels, microcontroller units and RAM test algorithms.

4.2.1 Default Error Tracer

Open **Default Error Tracer** | **Enable or Disable Error Tracing** and select necessary modules or domains to enable the development error tracing.

4.2.2 General Purpose Timer (GPT)

Use default settings.

Note

If the editor is deactivated (greyed out), you can activate the General Purpose Timer (GPT) module via click. The Modules Assistant opens to specify the short name of the module to be added. Click **[Finish]** to add the module.

4.2.3 RAM Test

Some hardware-specific settings are necessary.

Note

If the editor is deactivated (greyed out), you can activate the RAM Test module via click. The Modules Assistant opens to specify the short name of the module to be added. Click **[Finish]** to add the module.

4.3 Communication

The Communication domain contains configuration editors for the configuration of general parameters of the communication related modules, ECU Bus Controllers, protocol data units, data signals and the transport protocol.

Note

The following setting description are meant for demonstration purposes only and can differ from the settings, that are necessary for your project.

If there are modules describe, you do not have in your project, skip this chapter.

4.3.1 Communication General

Define central settings of the communication based modules.

CAN

1. Open **CAN** | **Miscellaneous**
2. Set **[...]** parameter **Counter Ref** to **SystemTimer**.
3. Set **Interrupt Category** to **CATEGORY 2**
4. Set **Interrupt Lock** to **DRIVER**

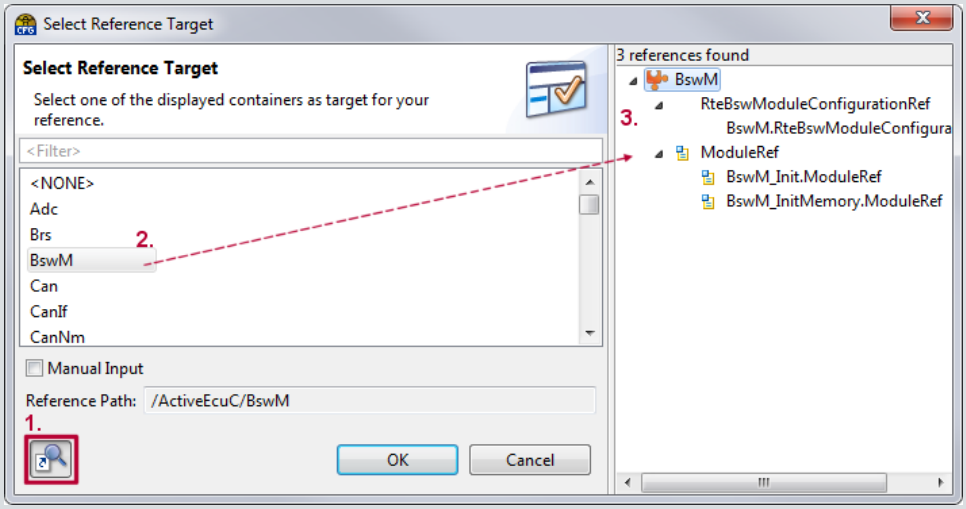
4.3.2 Bus Controller

1. Open the **CAN Controllers** | **<CANController>**
2. Define **Controller Default Baudrate**
3. Define **CPU Clock Ref**

Note

If no CPU Clock Ref is available, add a new reference via ✣ in the **Select Reference Target** view.

To get more reference information click on the icon **Element Usage** (1.) to activate the view with the found references (3.). Then select any list entry to get displayed the reference information.



4.3.3 PDUs

Configure the PDUs of the modules (depends on your project)

- CAN/LIN/FR
- CANIF/LINIF/FRIF
- CANTP/LINTP/FRISOTP
- PDUR
- IPDUM
- COM

4.3.4 Signals

Configure the communication signals of the ECU.

4.3.5 Socket Adapter Users

Use default settings.

Note

If the editor is deactivated (greyed out), you can activate the SOAD module via click. The Modules Assistant opens to specify the short name of the module to be added. Click **[Finish]** to add the module.

4.3.6 Transport Protocol

1. Open **FlexRay** | **Tx Pdu Pools** and **Rx Pdu Pools** to add a new **Tx Pdu Pool** and **Rx Pdu Pool**.
2. Open **FlexRay** | **Connections** | **FrTpConnection** and set **[...]**

- Connection Control,
- TxPduPool and
- RxPduPool.

4.4 Diagnostics

The Diagnostic domain contains comfort editors for the configuration of extended data records and snapshot records, diagnostic events and production error handling of the individual BSW modules and an automatic setup of the NV memory blocks.

4.4.1 Diagnostic Data Identifiers

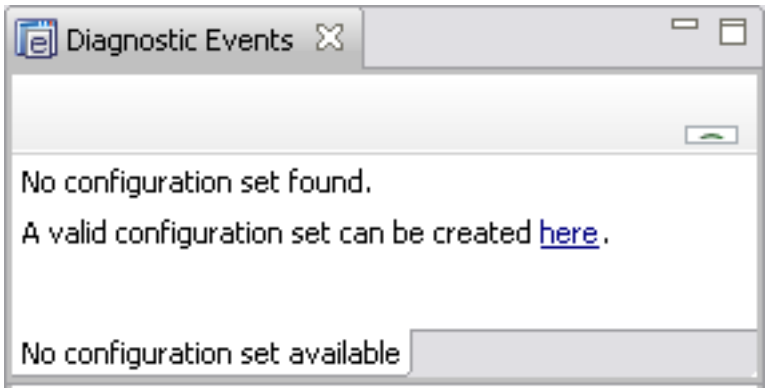
Use default settings as these have been derived from input files.

4.4.2 Diagnostic Event Data

Use default settings as these have been derived from input files.

4.4.3 Diagnostic Events

If no configuration set is found, use the hyperlink **here** to create a valid configuration set.

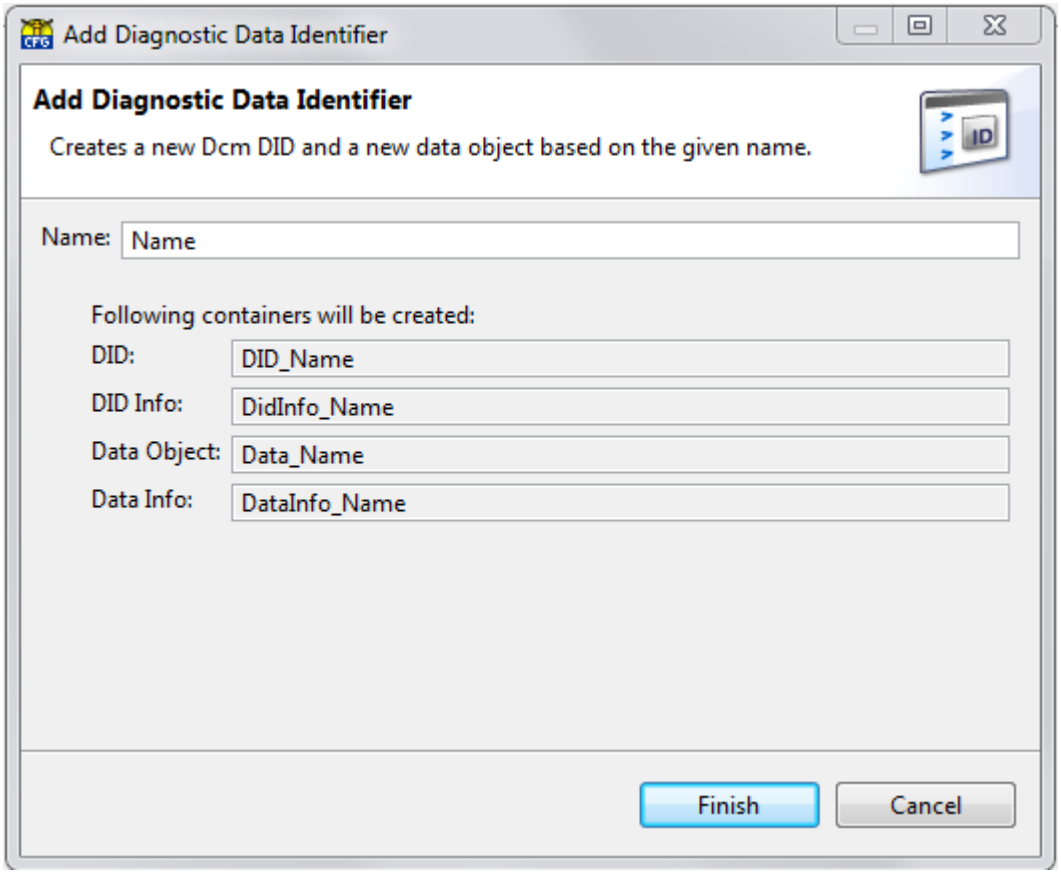


4.4.4 Production Error Handling

Use default settings.

4.4.5 Add Diagnostic Data ID Assistant

Start the assistant, enter a Name and confirm with **[Finish]**. The assistant creates a new DCM DID and a new Data Object based on the given name.



Note

The assistant cannot be used within a project with variants.

4.4.6 Automap Diagnostic Data Objects

This maps the data objects of the DCM DIDs to NV memory blocks. Therefore start the assistant, select the data objects, go on with **[Next >]**, select Data object to be mapped and confirm with **[Finish]**.

4.4.7 Setup Event Memory Blocks

The event memory block configuration needs to be updated. Start the Editor and confirm with **[Finish]**.

Note

The automatic setup of the NV memory blocks is required for diagnostic purpose.

4.5 I/O

4.5.1 IO Hardware Abstraction

Configure those I/O signals from peripheral devices, which need to be available for Software

Components operation.

4.6 Memory

The Memory domain contains comfort editors for the configuration of the general parameters of the memory related modules and the memory blocks.

4.6.1 Memory General

If FEE is used, define

- **BSS Thresholder Reserved**
  BSS - Background Sector Switch
- **FSS Thresholder Reserved**
  FSS - Foreground Sector Switch

4.6.2 Memory Blocks

Open **Memory Partitions** | **PartitionConfiguration** and define

- **Partition Device**
  Add the reference to their device of the partition

Note

If the editor is deactivated (greyed out), you can activate the EA/FEE module via click. The Modules Assistant opens to specify the short name of the module to be added. Click **[Finish]** to add the module.

4.6.3 Optimize Fee

If Fee is used, start FEE Optimization. The number of chunk instances (parameter FeeNumberOfChunkInstances) of the Fee blocks will be optimized. Please review the new value or select "skip" to leave the parameter.

Note

If the editor is deactivated (greyed out), you can activate the FEE module via click. The Modules Assistant opens to specify the short name of the module to be added. Click **[Finish]** to add the module.

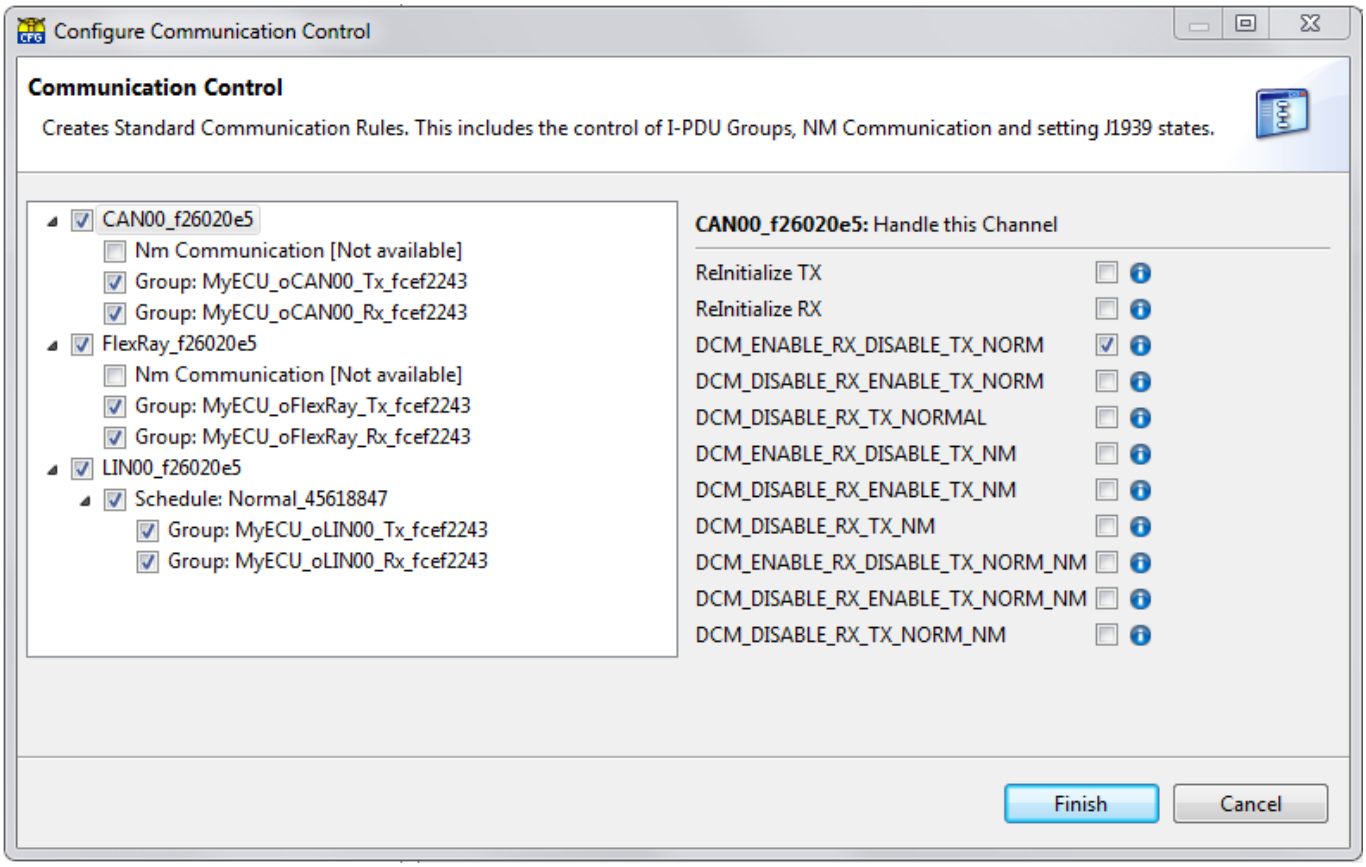
4.7 Mode Management Editors

The Mode Management domain contains comfort editors for the configuration of BSW Management, ECU Management, initialization and restart sequences of the BSW modules, sleep modes and wakeup sources, supervised entities and watchdog modes.

4.7.1 BSW Management

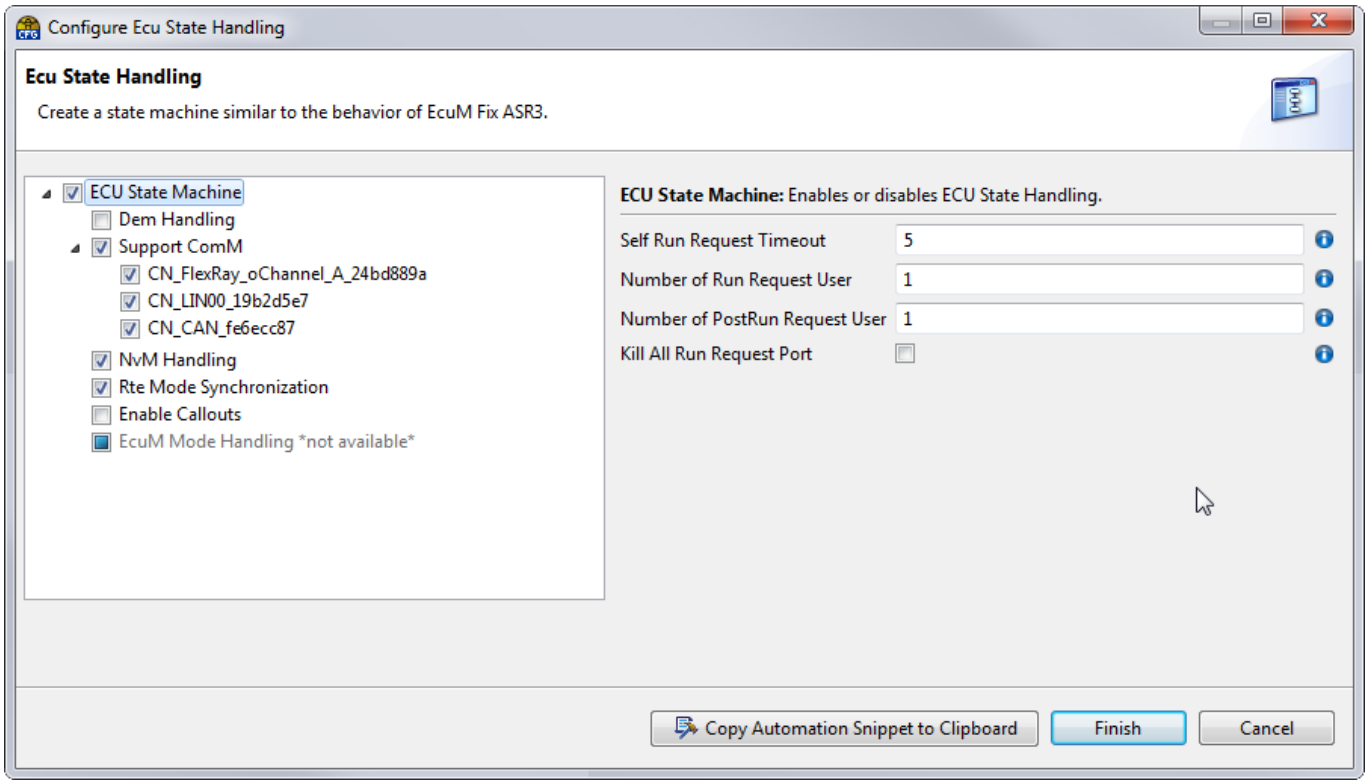
The BSW Management view has different tabs. Open **BswMConfig** tab.

Auto Configuration: Communication Control
Open Auto Configuration: Communication Control and configure Communication Control via **Configure Communication Control** hyperlink  Configure Communication Control. The Configure Communication Control dialog opens.



Select the bus systems for which you would like to create standard communication rules.

Auto Configuration: Ecu State Handling
Open Auto Configuration: Ecu State Handling and configure Ecu State Handling settings via **Configure Ecu State Handling** hyperlink  Configure Ecu State Handling. The Configure Ecu State Handling dialog opens.



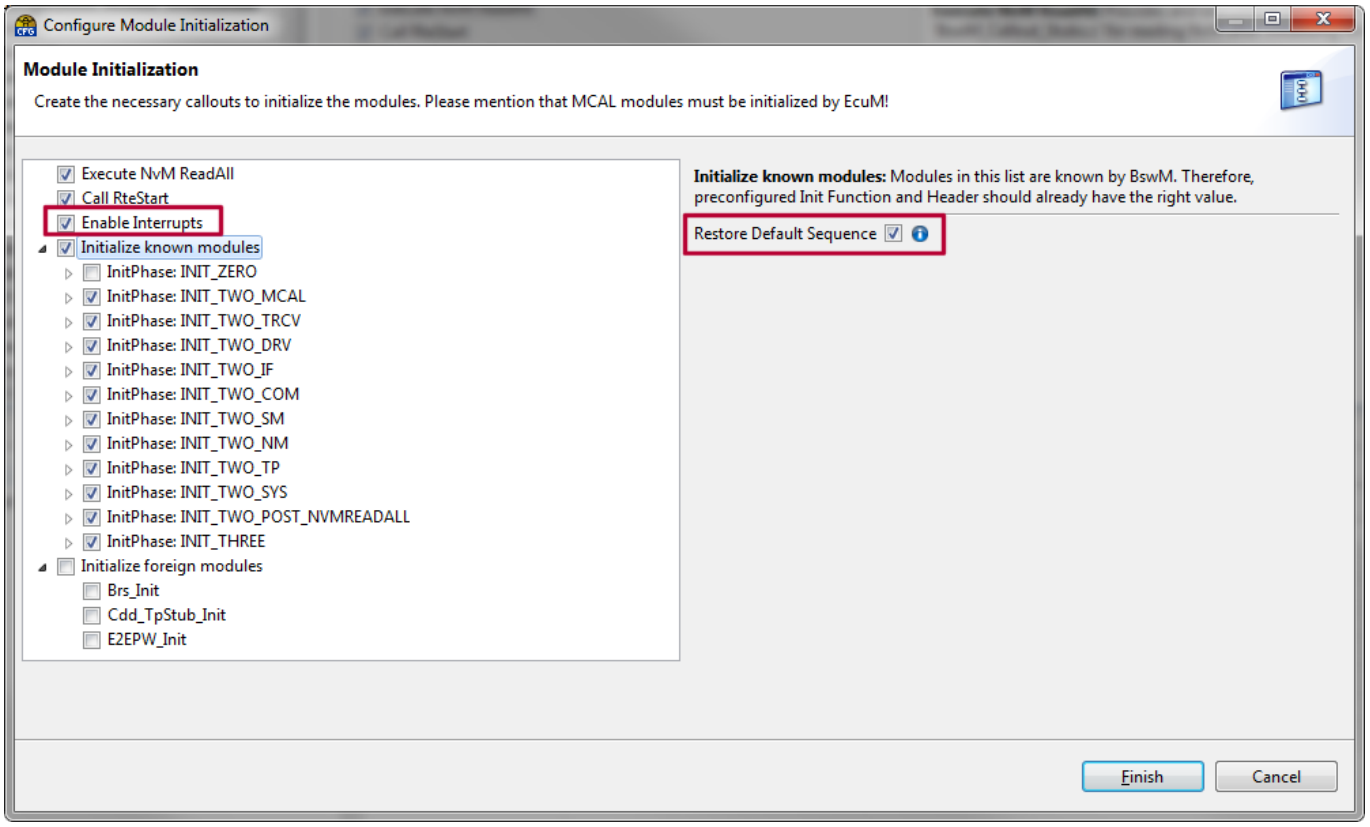
Select the necessary use case definitions to create a state machine similar to the behavior of EcuM fix in AUTOSAR3.

Note

If you select Ecu State Machine, you have to define the project-specific settings **Self Run Request Timeout** and **Number of Run Request User**.

Auto Configuration: Module Initialization

Open Auto Configuration: Module Initialization and configure Module Initialization via **Configure Module Initialization** hyperlink  Configure Module Initialization. The Configure Module Initialization dialog opens.



Make sure to set **Enable Interrupts** and **Restore Default Sequence**. For the latter setting you have to activate **Initialize known modules**.

Select module to create the necessary callouts to initialize the modules.

4.7.2 Activate Interrupts of Peripherals Devices

Activate **Enable Interrupts** to get the callout `BswM_AL_SetProgrammableInterrupts`. Use this callout to activate interrupts of the peripherial devices. Exceptions from this rule are explained in the technical reference of the drivers.

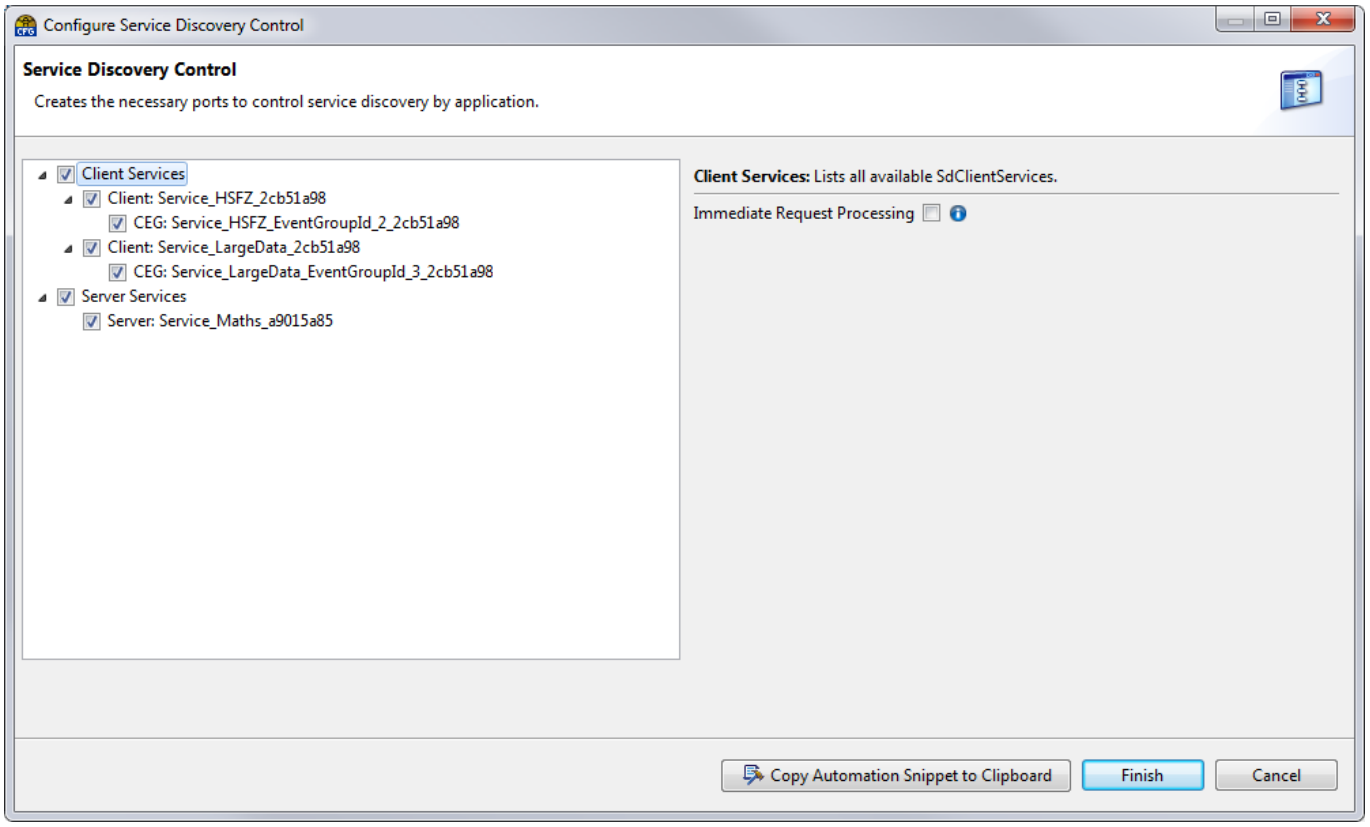
Note

Configure module initialization at initialization view.

Note

If **Restore Default Sequence** is activated, the lists will be sorted automatically, all new entries will be located at the end of the list.

Auto Configuration: Service Discovery Control
Open Auto Configuration: Service Discovery Control and configure Service Discovery Control via **Configure Service Discovery Control** hyperlink Configure Service Discovery Control. The Configure Service Discovery Control dialog opens.



Activate **Client Services** and **Server Services** and close the auto configuration view via **[Finish]**.

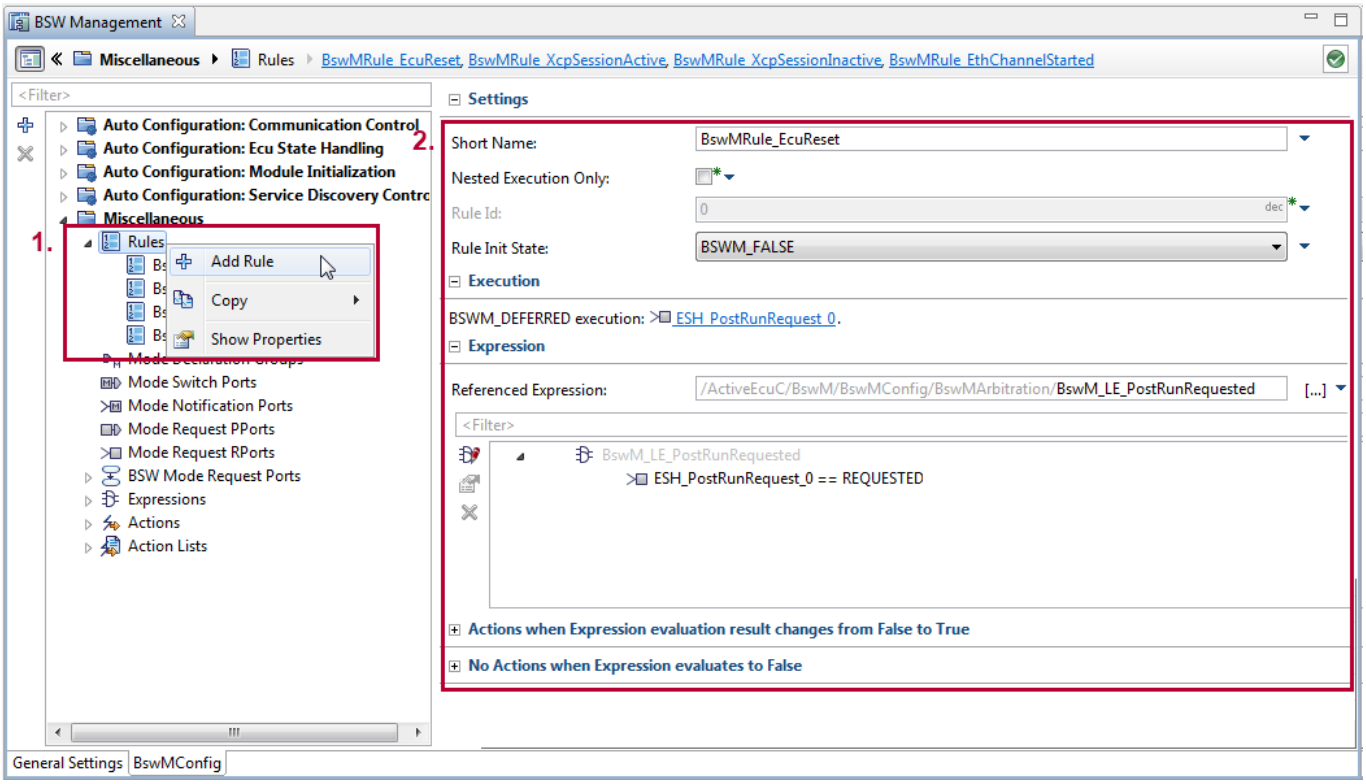
Miscellaneous - Custom Configuration
If necessary you can define additional Custom BswM Configuration information.

Note

To create a rule, at least one **logical expression** and one **action list** is necessary. If they are not available, you can additionally configure them via **Create Rule** dialog.

1. Open **Miscellaneous**, right click **Rules** and select **Add Rule**.
2. Configure added Rule
   Therefore select created rule and define

> Short Name
> Rule Init State
> Referenced Expression
> True Action List and actions which will be executed if the rule's condition evaluates to true
> False Action List and actions which will be executed if the rule's condition evaluates to false



Note

If no actions are configured, no action list will be created.

Note

It is possible to create groups subdivide the BSWM configuration into logical sections.

The elements can be organized in groups using drag and drop in the editor tree.

The groups are only for organization reason, functional usage is not possible. You can't move elements from auto configuration to the created BSWM Group.

4.7.3 ECU Management

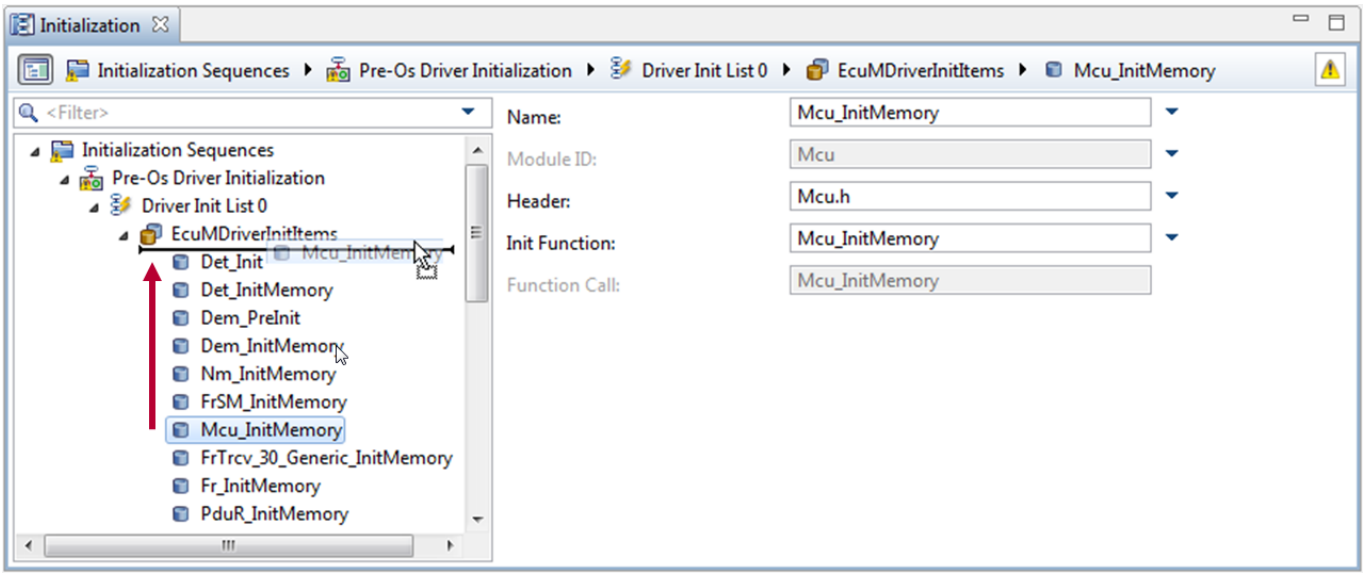
1. Open **EcuMCommonConfiguration** and define the **Default App Mode** which is loaded when the ECU comes out of reset.
2. Open **EcuMCommonConfiguration** and define the **OS Resource** which is used to bring the ECU into sleep mode.
3. Open **EcuMCommonConfiguration** | **EcuMSleepModes** and add **[+]** EcuMSleepMode. The Add EcuMSleepMode assistant opens, select **EcuMWakeupSources** and **McuModeSettingConf** and confirm with **[Finish]**.
4. Open **EcuMFlexConfiguration** and define **Normal Mcu Mode Ref** which is being restored after a sleep.

4.7.4 Initialization

Open **Initialization** to configure initialization and restart sequence of the basic software.

Pre-OS Initialization Sequence

1. Open **Initialization Sequences** | **Pre-Os Driver Initialization** and choose the **Driver Init List** that you want to configure.
2. Open **Driver Init List** | **EcuMDriverInitItems**. Each module in the list will be called for initialization in the list order. Modify the processing order via drag and drop in the tree:



3. Select all MCAL modules and the DET to call them on ECU startup.
4. Select **EcuMDriverInitItems** and delete via the **Delete containers button** those modules from the list that shall not be initialized before the OS initialization.

4.7.5 Watchdogs

Use default settings.

Note

If the Editor is deactivated (greyed out), you can activate the Wdg module via click. The Modules Assistant opens to specify the short name of the module to be added. Click **[Finish]** to add the module.

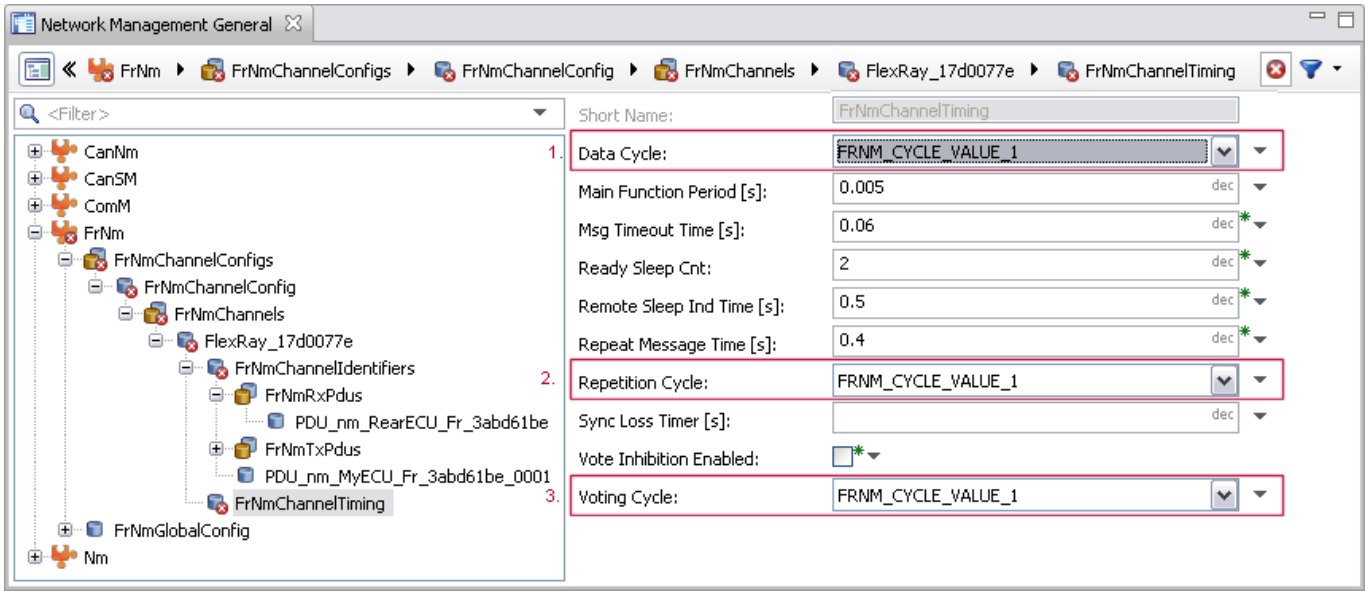
4.8 Network Management

The Network Management domain contains editors for the configuration of the general network management parameters and users which are relevant for the ECU communication management.

4.8.1 Network Management General

Open **FrNm** | **FrNmChannelConfigs** | **FrNmChannels** | **<FRNMChannelIdentifiers>** | **FrNmChannelTiming** and define

1. Data Cycle,
2. Repetition Cycle and
3. Voting Cycle.



Note

For all other NM modules, use default settings.

4.8.2 Communication Users

Use default settings.

4.8.3 Partial Networking

Use default settings.

4.9 Runtime System

The Runtime System domain contains comfort editors for the configuration of general RTE and OS parameters, operating system and mapping assistants.

4.9.1 Runtime System General

1. Open **Runtime System General** | **OS** and activate/deactivate the following settings if necessary.
   - > Stack Monitoring
   - > Use GetServiceId
   - > Use Parameter Access
2. Define **Scalability Class**.
3. Open **OS** | **Hook Routines** and activate/deactivate the following settings if necessary.
   - > Error Hook
   - > Panic Hook
   - > Post-Task Hook
   - > Pre-Task Hook
   - > Protection Hook
   - > Shutdown Hook
   - > Startup Hook

Note

The **Shutdown Hook** is required for shutdown of the ECU to power off.

The **Error Hook** is recommended to inform the application on any error events in the OS.

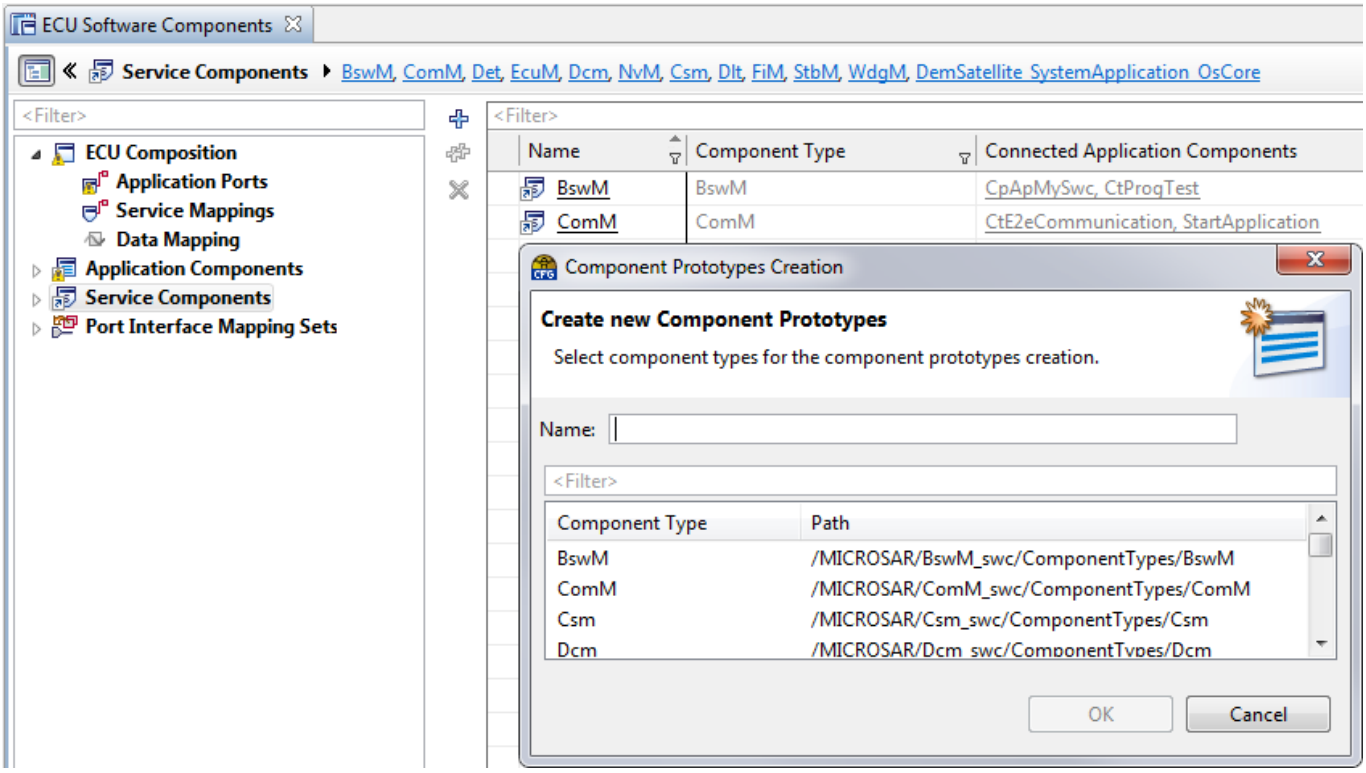
4.9.2 ECU Software Components

Service Components
Open **ECU Software Components** | **Service Components** and check if all Service Components are added to your project. If not create new component prototypes, based on component types created by the DaVinci Configurator Pro.

Note

The component types are stored in **<ProjectFolder>\Config\ServiceComponents**.

If necessary add ✚ all available service components which are currently not added.



4.9.3 Module Internal Behavior

Use default settings.

4.9.4 OS Configuration

Create Tasks

Open **OS Configuration** | **Task**. Add the following tasks, define Schedule, Priority and Type.

- Init_Task (Autostart has to be enabled!)
- Rte_Task
- Main_Task

Note

All events and alarms which are required are created by the RTE automatically.

Derived from the settings for the runnables and their activation (triggers) some OS events an OS alarms will be created automatically for RTE by the DaVinci Developer. The names start with **Rte_**.

4.9.5 Task Mapping

Use default settings.

4.10 Go on with Basic Editor

Open the Basic Editor and configure additional settings, currently not supported by specific configuration editors.

Note: Bottom Up Approach - from hardware-dependent to hardware-independent modules

Start configuration with hardware dependent modules and continue with the hardware independent modules.

4.11 Start Solving Actions

Changes in Configurator Editors/Basic Editor could lead to errors during live validation!

If these errors include solving actions, check and start solving action in the validation view.

4.12 Start On-demand Validation

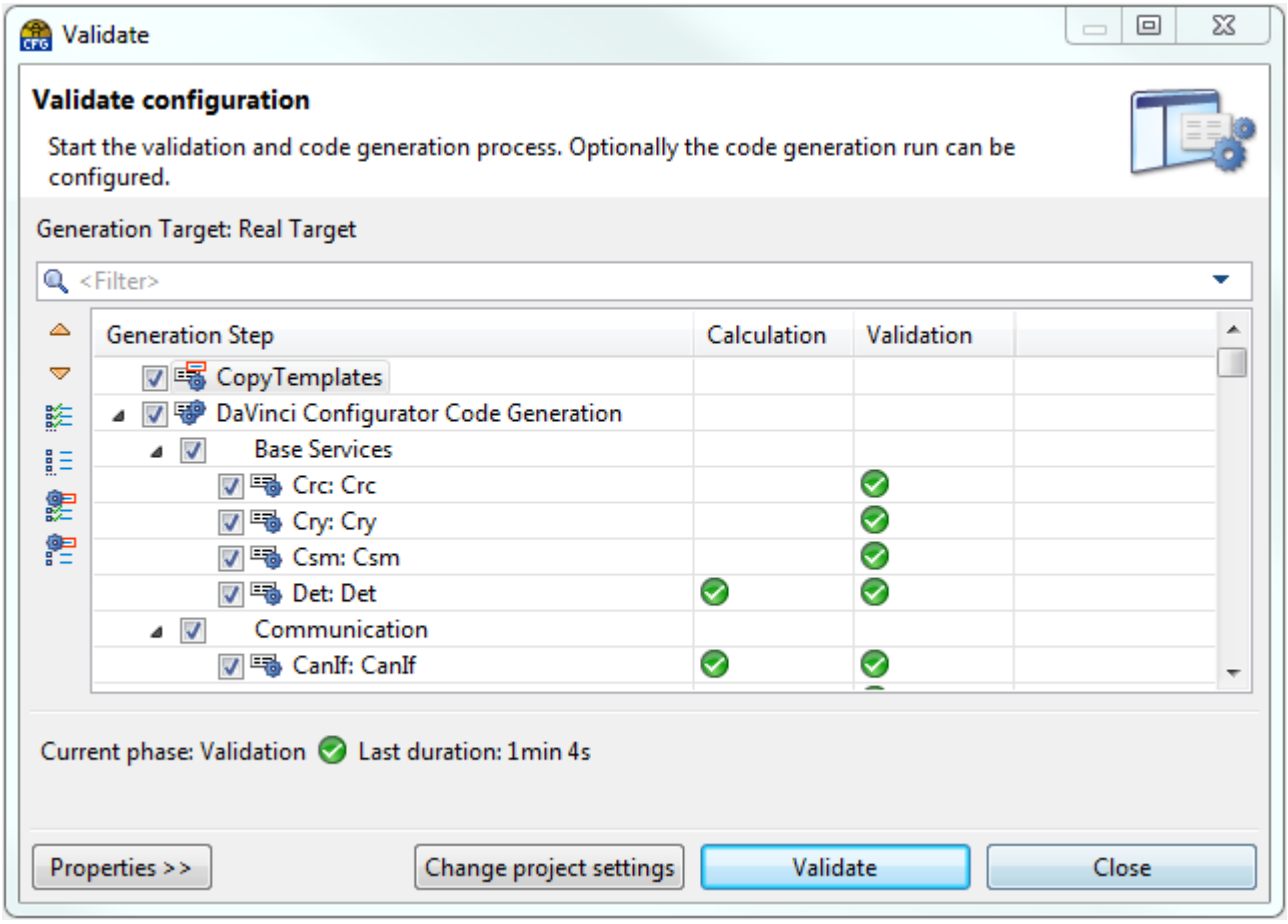
The on-demand validation is performed in addition to the automatic validation before starting the code generation or on explicit user request. The on-demand validation calls the specific validation function of the code generators.

1. Open **On-demand Validation** dialog via **On-Demand Generator Validation** button  or press **<F8>**.
2. Deactivate **RTE**

Note

Currently the RTE is not configured completely, this causes errors while validation. The RTE configuration will be finished after SWC Design in the DaVinci Developer.

3. Start the Validation via **[Validate]**



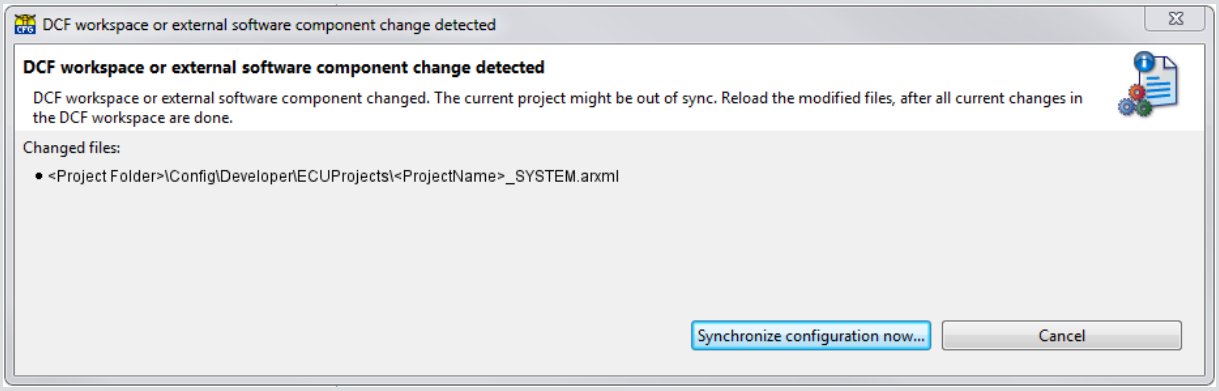
Validation finished successfully?

Save and close the DaVinci Configurator Pro and go on with configuring SWCs in the DaVinci Developer.

Save the DaVinci Configurator Pro project and go on with configuring SWCs in the DaVinci Developer.

Note

It is possible to work with both tools parallel. An update message within the DaVinci Configurator Pro will inform you about the DaVinci Developer workspace changes and vice versa.



Validation finished NOT successfully?

See error message in validation view and fix the configuration.

4.13 BSW Configuration finished

You have finished your BSW Configuration?

Now it depends on your project, how to go on. You have created a Developer workspace while setting up the project? Then you have to go on with the STEP5 Design Software Components on page 63 using the DaVinci Developer. If you have no Developer workspace, you can skip the following chapter an go on with STEP6 Mappings on page 65.

5 STEP5 Design Software Components

Switch to the DaVinci Developer to create, configure or modify software components, create or add ports and data elements. Connect software components and define runnables and their activation and interfaces (data access).

Expert Knowledge

You need to know more? See section Software Component Design on page 107.

5.1 Switch to DaVinci Developer

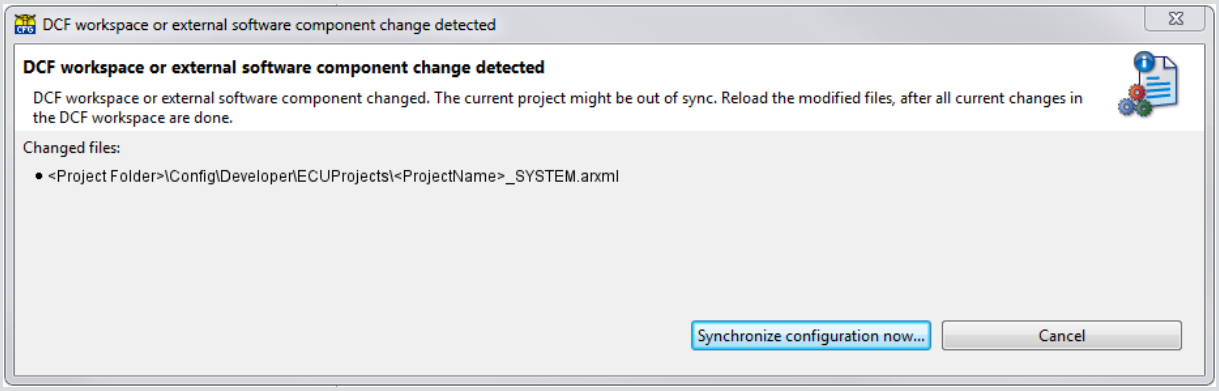
You can start DaVinci Developer with already loaded project in two different ways:

- **Start** | **Programs** | **Vector AUTOSAR Projects** | **<your project name>** | **Configure SWCs** (Only if Create entries in the start menu has been selected)
- Right-click on your **<projectname>.dpa** and select **DaVinci Project Assistant** | **Configure SWCs**

Note

For faster resynchronization, the DaVinci Configurator Pro may remain open while working with the DaVinci Developer.

An update message within the DaVinci Configurator Pro will inform you about the DaVinci Developer workspace changes.



5.2 Design Software Components

If the software components are not delivered by the OEM, use the DaVinci Developer to design your software components. How to do this is described more detailed in the expert knowledge part of this manual. Use the link above.

Everything correct? Finally check your settings via **Check workspace**  .

6 STEP6 Mappings

Start project mappings via assistants offered by the DaVinci Configurator Pro. It is also possible to perform data mapping within the DaVinci Developer.

Expert Knowledge

You need to know more? See section Mappings on page 129.

6.1 Perform Data Mapping within DaVinci Developer or DaVinci Configurator?

Via the data mapping, you connect data elements with network signals. Network signals have been loaded via import of ECU Extract of System Description.

Data Mapping can be done in DaVinci Developer or DaVinci Configurator Pro. Decide which tool should be used. In the following step, both variants are described.

Note

Data Mapping in DaVinci Developer has a higher priority than Data Mapping in the DaVinci Configurator Pro.

Data Mapping performed in DaVinci Developer cannot be overwritten in DaVinci Configurator Pro.

Cross Reference

To perform data mapping within the DaVinci Configurator Pro, skip this section and go on with section Switch (back) to DaVinci Configurator.

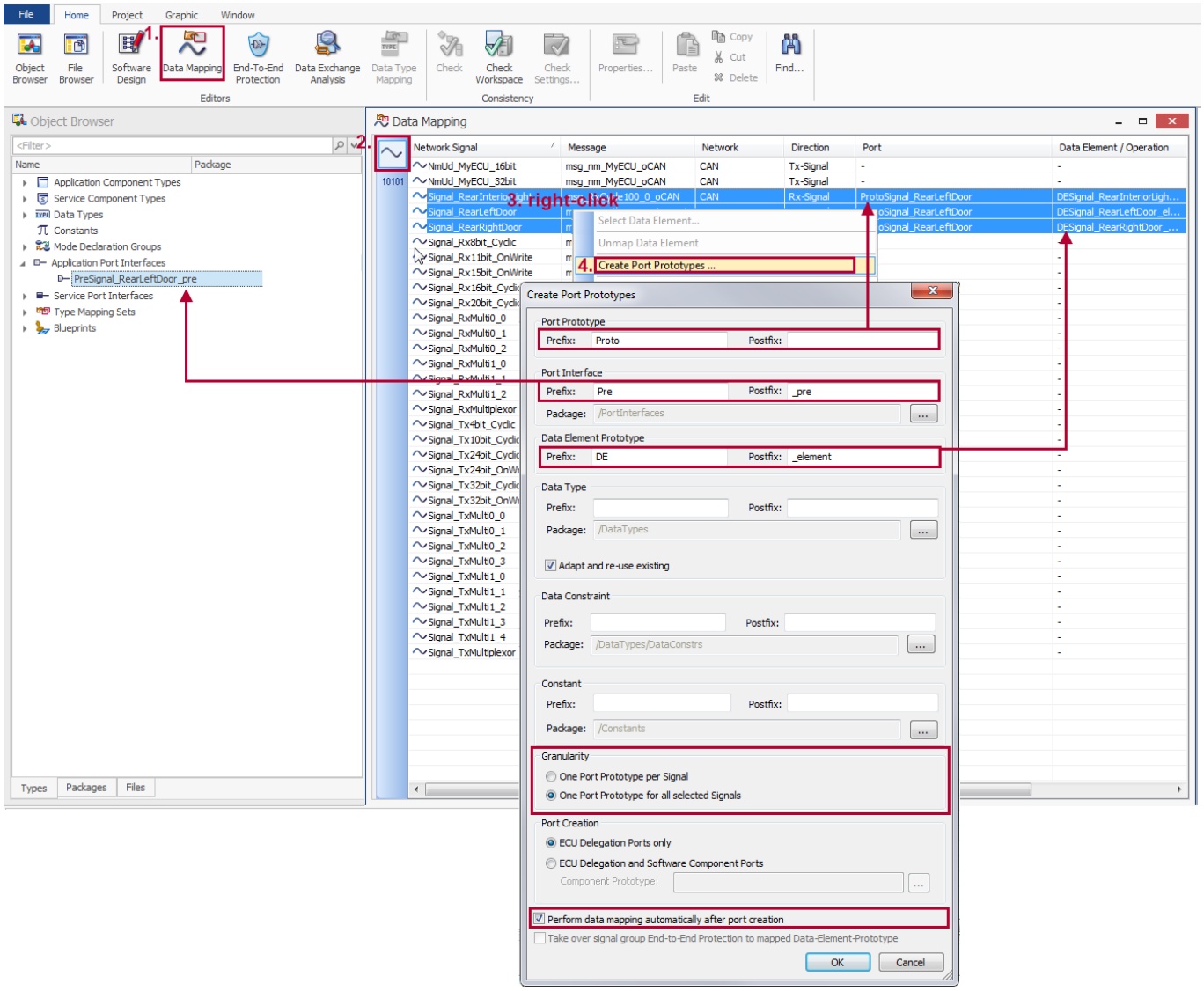
6.2 Data Mapping within the DaVinci Developer

The DaVinci Developer offers two ways of data mapping - **automatically** OR **manually**.

6.2.1 Data Mapping Automatically - DaVinci Developer

Network signal oriented approach
Select bus signals and let DaVinci Developer create all necessary data elements, ports and the mapping of the signals to the data elements.

Finally, you only have to drag'n'drop the port interfaces to your software components and create the connections. Here is how this works.



1. Click **Data Mapping** on the **Home** tab
2. Select the **Signal view mode**
3. Select signals to create Data Elements and Port(s)
4. Right-Click selected signals and select **Create Port Prototypes ...** to open the **Create Port Prototypes** window.
5. Define prefixes and postfixes for the Port Prototype Name, Port Interface Name and Data Element Prototype Name.
6. Select whether you want One Port Prototype per Signal or One Port Prototype for all selected Signals.

Note

If you select One Port Prototype for all selected Signals, of course the signal must be of equal direction, only Tx signals or only Rx signals.

7. Confirm your settings with **[OK]**.
8. You see that one Port Prototype, three Data Elements and one Application Port Interface are generated. The pre- and postfixes help to easily find them in the list.
9. Select the Software Design view and see the delegation port that has also been created.
10. Now drag'n'drop the Application Port Interface from the Object Browser to your software component (same direction as the delegation port) and connect them.

6.2.2 Data Mapping Manually - DaVinci Developer

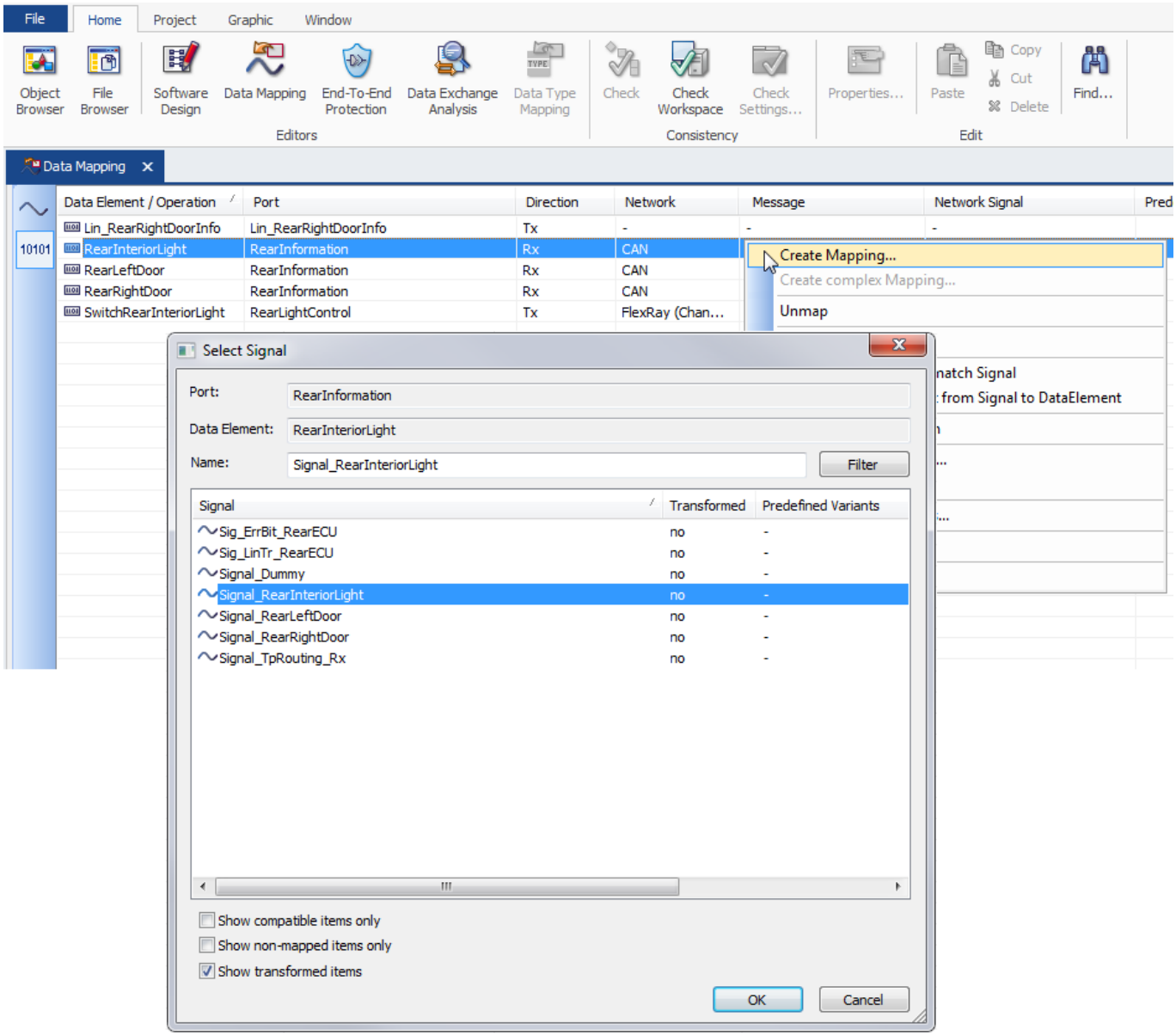
You can also connect the data elements with real bus signals manually.

Note

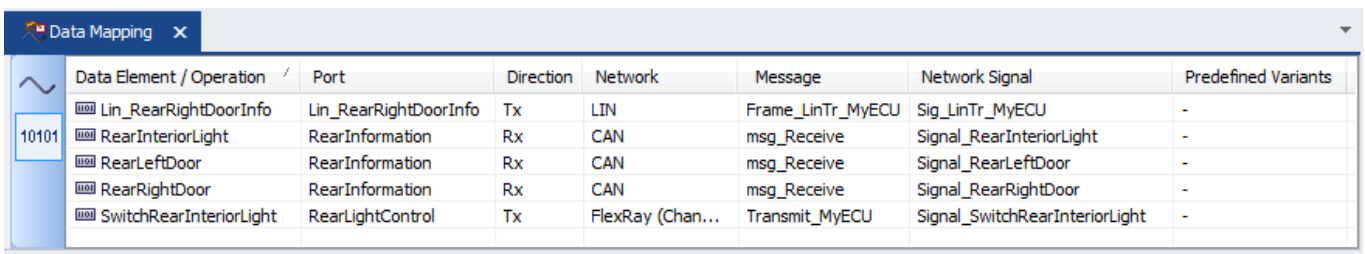
A prerequisite is that you have to create the necessary delegation ports before!

Select Data Mapping in the ECU Projects view and select e.g. the **Data Element View Mode**  and see a list of data elements. The dash (-) on the right side below Network, Messages and Network Signals shows that there is no assignment between data elements and messages/signals from the ECU Extract of System Description.

Right-click the **Data Element column** and click **Select Signal...**. Select the suitable signal in the **Select Signal** window and confirm with [OK]. Repeat this until every signal is mapped.



The result of the data mapping should look like this.



6.2.3 DaVinci Developer - Save and close

You have imported your Service Components, configured your Software Components and finished Data Mapping?

Check your workspace. If there are no messages in the **Output** view, save your project and close the DaVinci Developer.

6.3 Switch (back) to DaVinci Configurator

Start the DaVinci Configurator Pro (again) with already loaded configuration using one of the two possibilities.

> **Start** | **Programs** | **Vector AUTOSAR Projects** | **<your project name>** | **Configure BSW**
(Only if Create entries in the start menu is selected)

> Right-click on your **<projectname>.dpa** and select **DaVinci Project Assistant** | **Configure BSW**

6.4 Synchronize System Description

Changes by the DaVinci Developer require a synchronization of the System Description in the DaVinci Configurator Pro.

The necessity of a synchronization is detected by the validation process of DaVinci Configurator Pro and displayed at the footer of the DaVinci Configurator - **System or SWC description out-of-sync**. Start System Description synchronization via **Synchronize configuration now...**.

6.5 Add Component Connection

Data elements can only be transported from one application component to another if their ports are connected via connectors. Create these connections using the **Component Connection Assistant** Add Component Connection (below **Runtime System**).

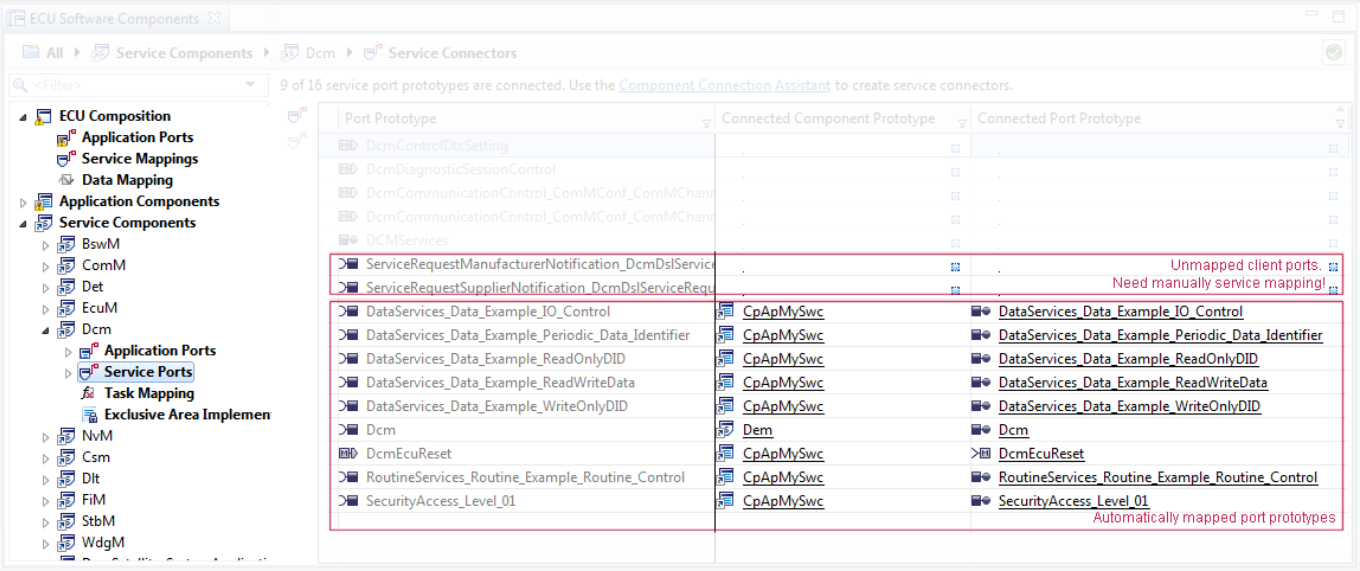
1. Start Assistant, select connector type and go on with **[Next]**
Select **Application Connector Prototypes** to connect software components or **Service Connector Prototypes** to connect service components
2. Select a **Component Prototype** and go on with **[Next]**
3. Select **Port Prototypes** and go on with **[Next]**
4. If necessary select additional search option and go on with **[Next]**
Initially the Component Connection Assistant will search matching port prototypes based on the port prototype name automatically
5. Check mapping. All Connector Prototypes are mapped with the right ports? Confirm with **[Finish]**

6.6 Service Mapping

6.6.1 Service Mapping via Service Component

If you have connected your service components via **Add Component Connection** in the previous step, the service mapping is basically done.

To define additional mappings manually you have to configure it in the service component directly. Therefore open **ECU Software Components** | **Service Components** and select the service component you want to map. Open service component and select **Service Connectors**.

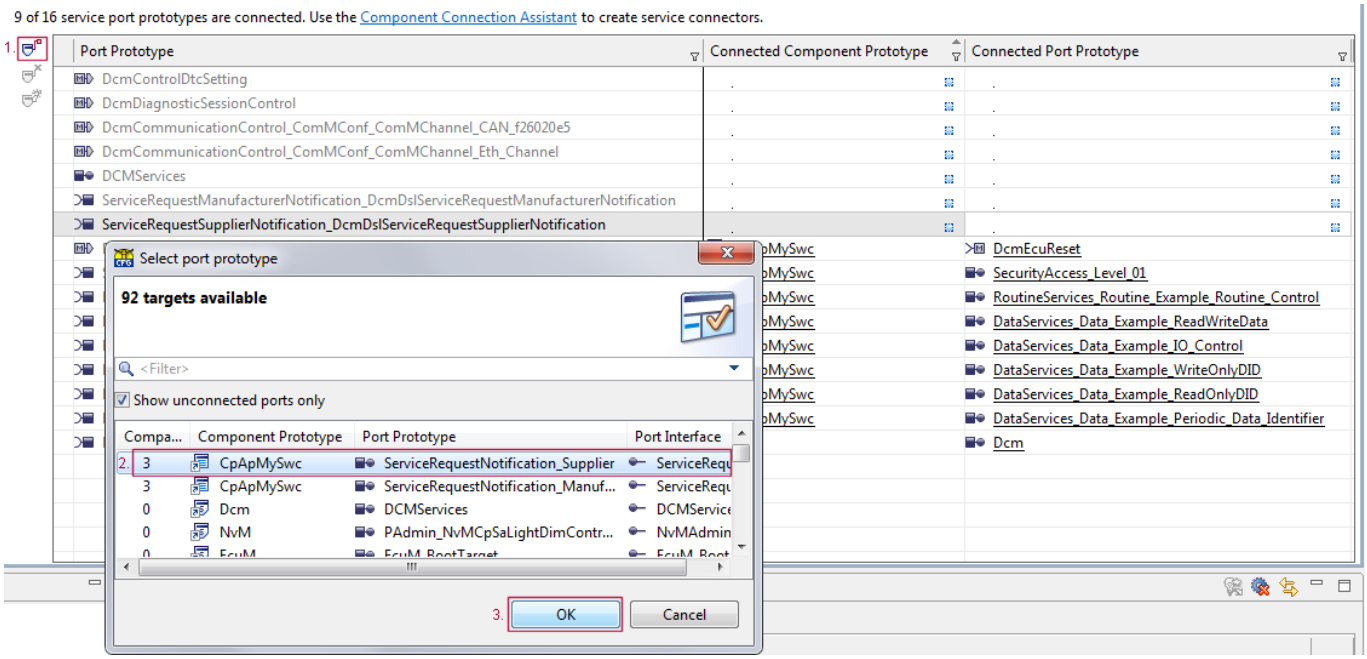


Now you can see all ports, the automatically matched connections and the unmapped ports. **Is manually service mapping necessary?**

Note

Each **client port** should be mapped to a **server port**! For all unconnected client ports the RTE will generate a **dummy** function returning RTE_E_UNCONNECTED.

1. Select the service component port and use the **Connect** button .
2. Assign the appropriate port prototype to the selected port of the service component.
3. Finish the mapping with **OK**.



The appropriate port prototype not exists? You like to connect the port to a new port prototype?

Therefore select the service component port and use the button to open the **Connect To New Port Prototype** Assistant.

1. Select the **Component** to be connected with the selected port prototypes.
   Only those components are displayed which component type is part of the DaVinci Developer workspace.
2. Define the name of the **Port Prototype to be created**. By default the same name is used. Therefore click in the **Port Prototype to be created** column.
   If a server port is selected, you can additionally select, if server runnables should created.
3. Click **Finish**

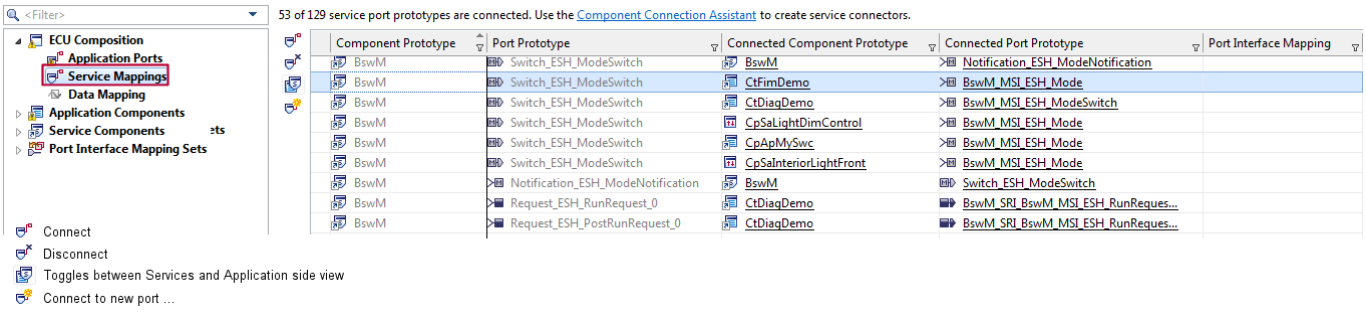
What happens?

The DaVinci Developer will open in background, a new port prototype (Application and Service port prototypes are possible) will be created within selected application component. The workspace will be saved and the DaVinci Configurator project will be reloaded. The new created port is now available within the DaVinci Configurator project and will be mapped automatically to the service component.

6.6.2 Service Mapping (overview)

Here you can do the complete service mapping. Connect or disconnect services to applications, toggle between services view and application view or connect to new port.

Here you also get a complete overview of all connections.



6.7 Add Data Mapping

You have already performed Data Mapping within the DaVinci Developer?

Yes? Skip this section and go on with Add Memory Mapping on page 74.

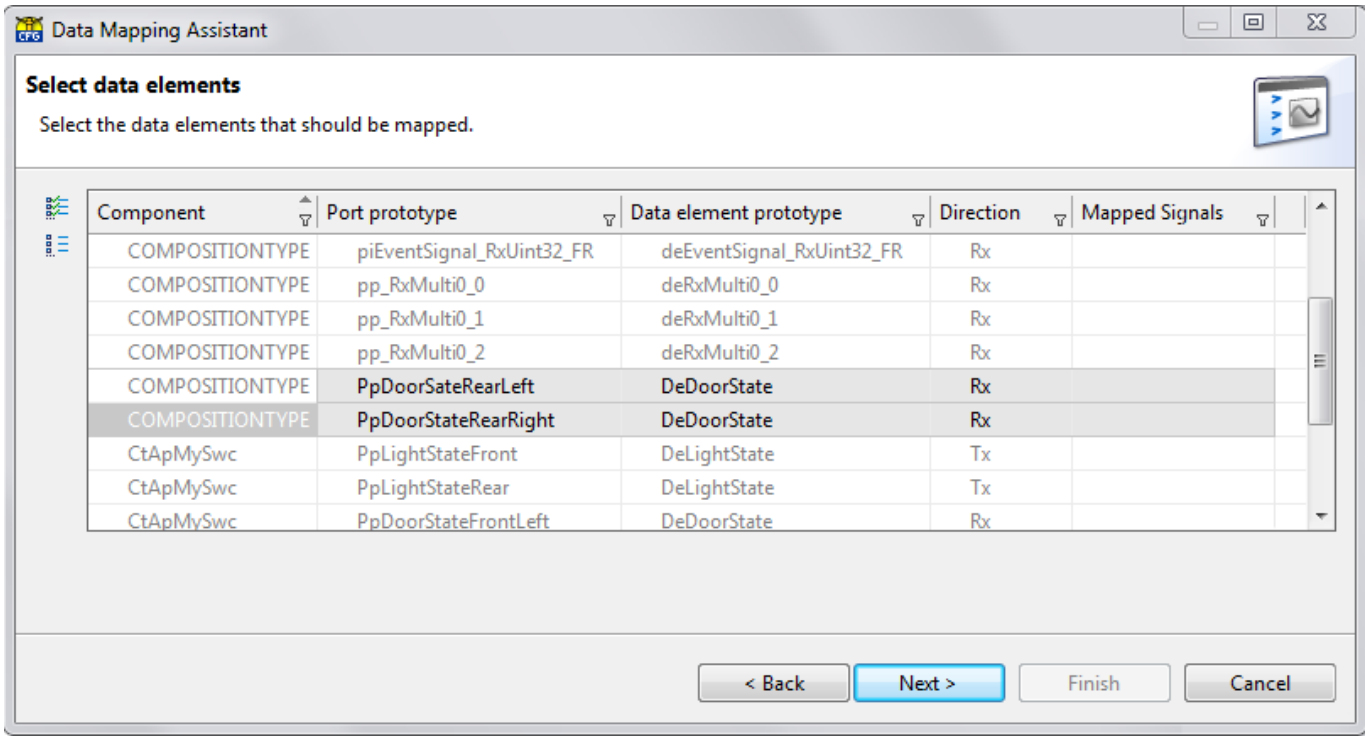
Cross Reference

See section Perform Data Mapping within DaVinci Developer or DaVinci Configurator? on page 65

No? Start Assistant via **Add Data Mapping** Add Data Mapping and select mapping direction, i.e. choose whether **Signals** or **Data Elements** should be mapped.

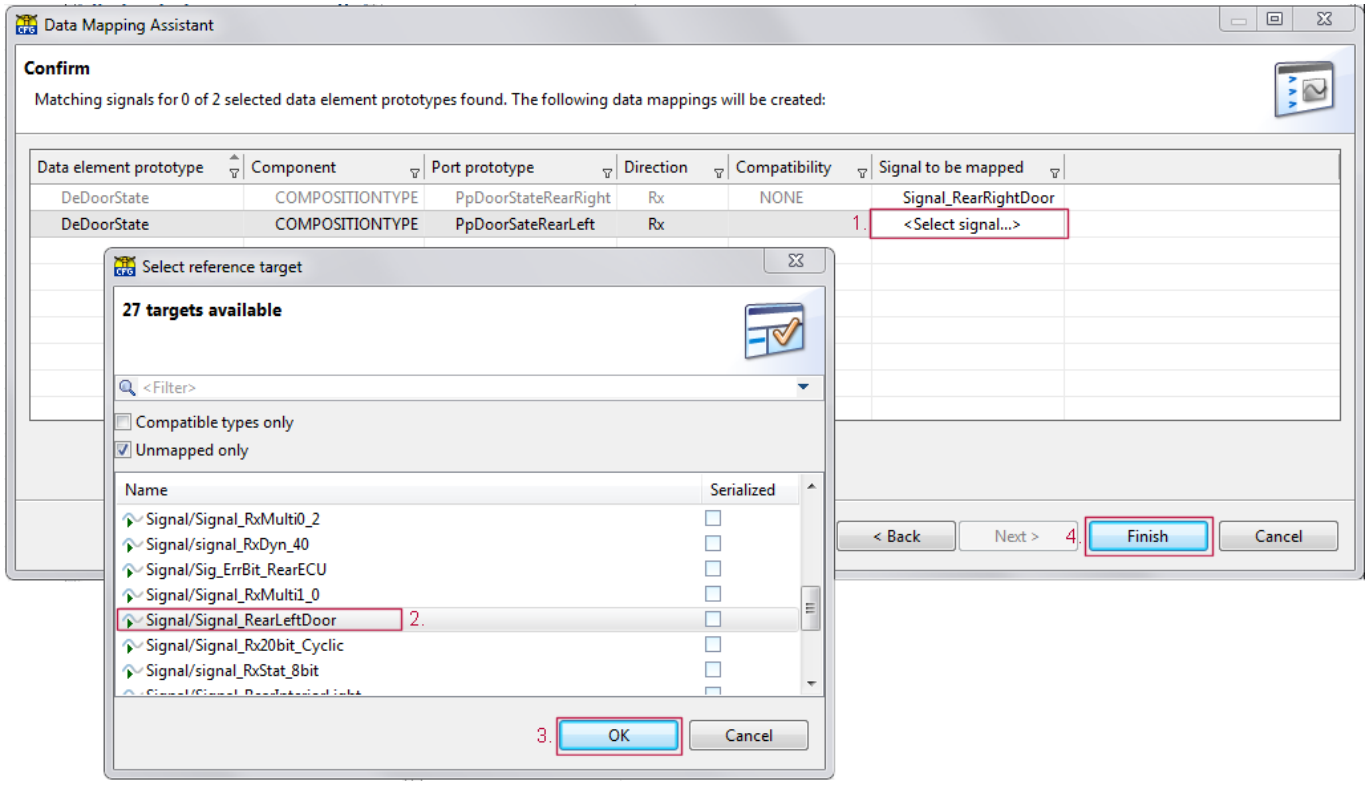
Choose **Find matching signals for the data elements** to assign your signals based on a selected data element and go on with **[Next]**

Now you can see a list of data elements. The empty row **Mapped Signals** shows that there is no assignment between data elements and messages/signals from the ECU Extract of System Description.



Select the Data element prototypes you want to map and go on with **[Next]**.

Now you can see if matched signals are found automatically. If there are no entries in row **Signal to be mapped**, you have to define the signal manually.



1. Click **<Select signal...>** to open the **Select reference target** window.
2. Select the signal you want to map.
3. Confirm with **[OK]**.

Repeat these steps until all data elements you have selected for mapping are mapped to a signal.

4. Close Data Mapping with **[Finish]**.

6.8 Add Memory Mapping

You have to map all per-instance memory objects of the application components to NV memory blocks. Therefore use the **Memory Mapping Assistant**  Add Memory Mapping .

Note

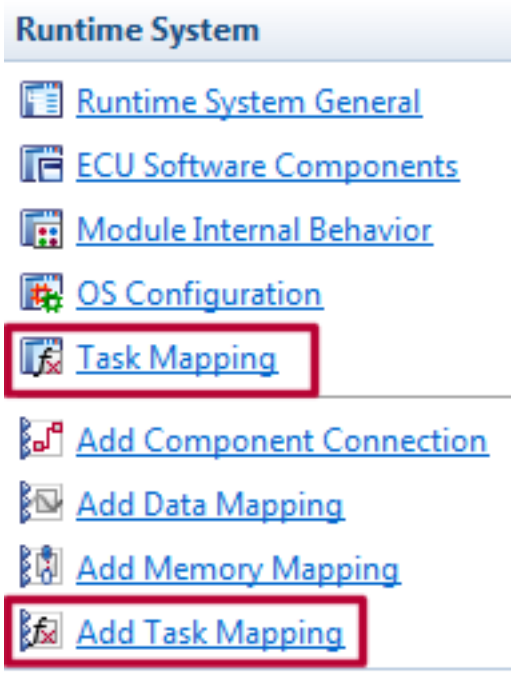
A per-instance memory object is mappable if it is referenced by a service need object. The definition of per-instance memory objects or service needs is part of the component type definition and not yet editable by DaVinci Configurator Pro.

1. Start Assistant, choose **Use Case** and go on with **[Next]**
2. Select component prototype and go on with **[Next]**
3. Select per-instance memory objects
   Choosing Use Case Find exisiting NV memory blocks?
   Go on with **[Next]**, Check the memory mappings and confirm with **[Finish]**
   Choosing Use Case Create new NV memory blocks?
   Confirm with **[Finish]**

6.9 Add Task Mapping

You have to map all runnable entities and schedulable entities (e.g. Main Functions of the BSW modules) to a task. There are two ways to perform the task mapping.

Where to activate the two ways of task mapping



Either the **Task Mapping Assistant** Add Task Mapping or the **Task Mapping** view .

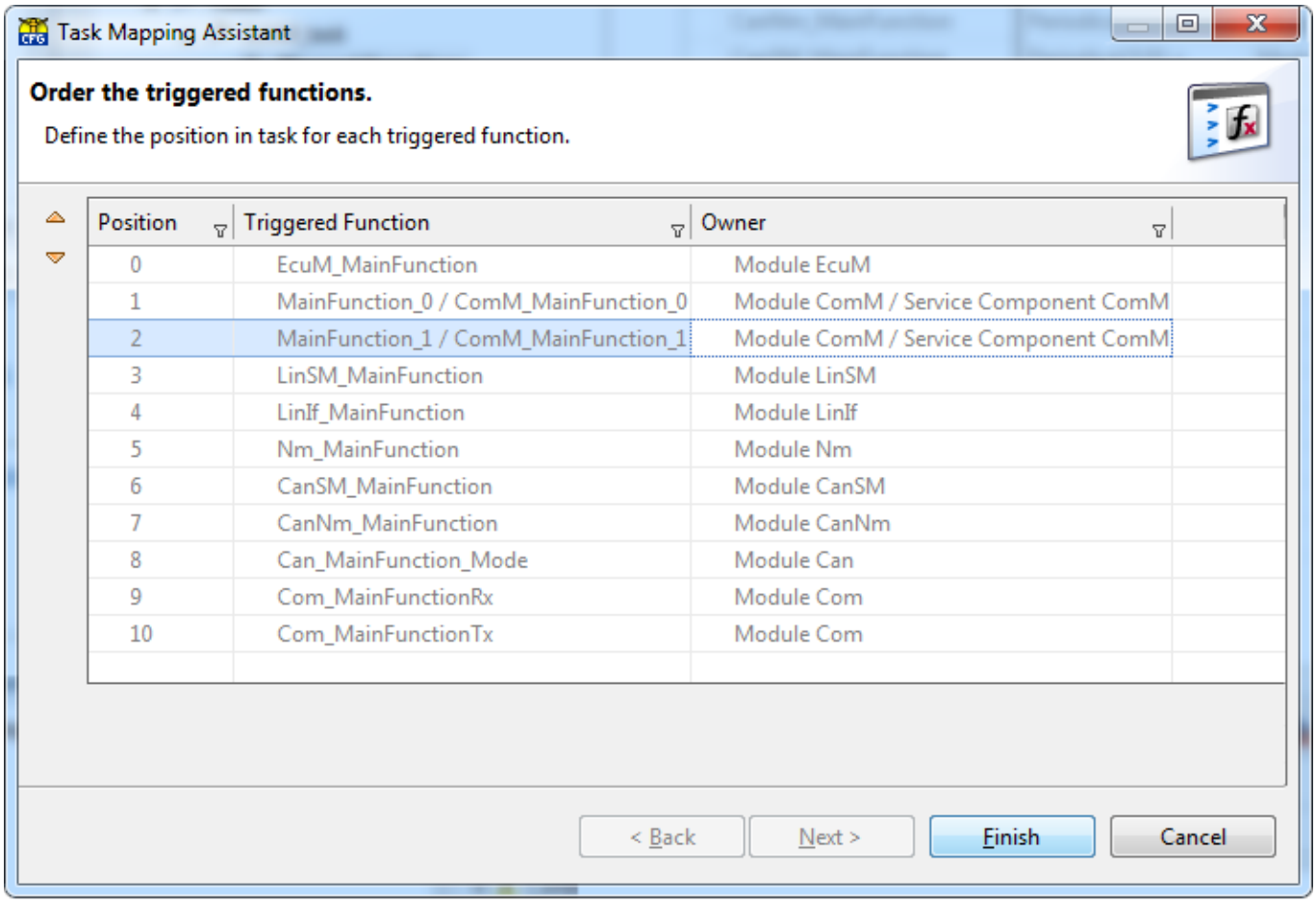
Our advice is to use the assistant for the first mapping and then switch over to the **Task Mapping** view to do the fine tuning.

Note

A runnable has to be mapped to a task if it is not re-entrant but could be called re-entrantly during system operation.

6.9.1 Task Mapping Assistant

1. Start Assistant, select all Triggered Functions with Function Trigger On Init and go on with **[Next]**
2. Select Init_Task and go on with **[Next]**
3. Define the position in task for each triggered function and confirm with **[Finish]**



4. Repeat the steps and select all **Main Functions** of **Service Components** and **Modules** and map them to **Main_Task**.
5. Repeat the steps and select all **Runnables** of your **Application Components** and map them to **Rte_Task**.

Note

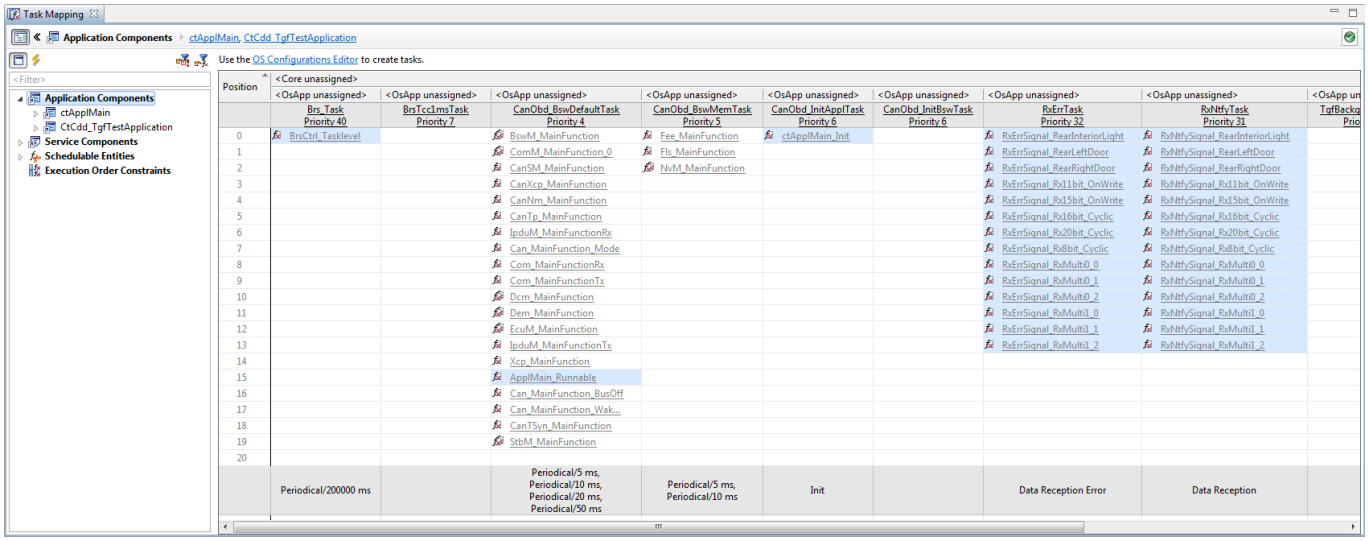
Runnables with Function Trigger **On Operation Invocation** are normally not needed to be mapped to a task, as they result in direct function calls.

6.9.2 Task Mapping

The task mapping is one view where you can

- map runnable entities per drag and drop,
- move mapped runnable entities from one task to another,
- filter the list on the left side for mapped mandatory or unmapped runnable entities,
- get information of the mapped runnable entities concerning their triggers, etc.

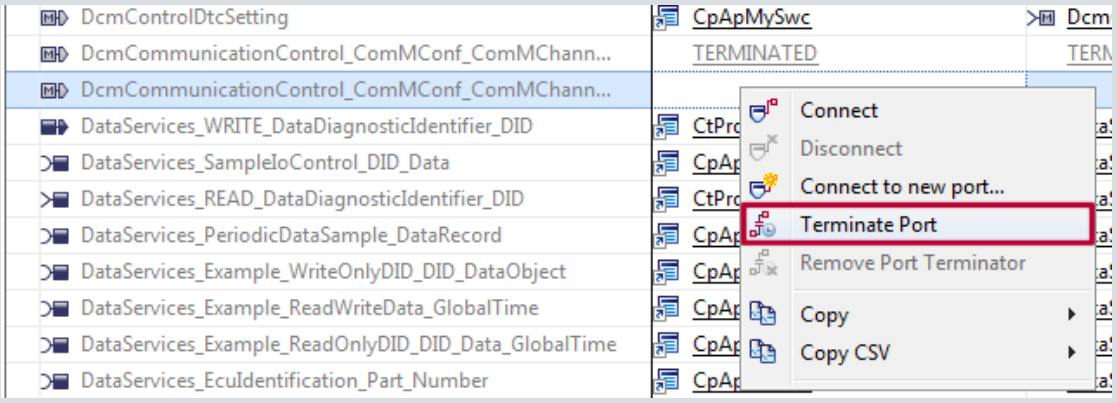
More details about the features of this dialog you find in the **Online Help** of the DaVinci Configurator Pro.



Mappings finished? Validation successful? Then go on with project generation!

Note - Terminate Port

A port you do not use in your project because you do not have to use it, can be left open or unconnected, but then warnings will be given. Using **Terminate Port** you can put a note on the port , that you intentionally left this port open.



7 STEP7 Code Generation

Now you can go on with code generation.

Expert Knowledge

You need to know more? See section Generation on page 136.

7.1 Generate SWC Templates and Contract Phase Headers

Start the generation of the SWC templates and/or contract phase headers via  from DaVinci Configurator Pro toolbar.

Cross Reference

How to select your settings for the generation of the software components templates and contract phase headers? See section STEP2 Define Project Settings on page 39.

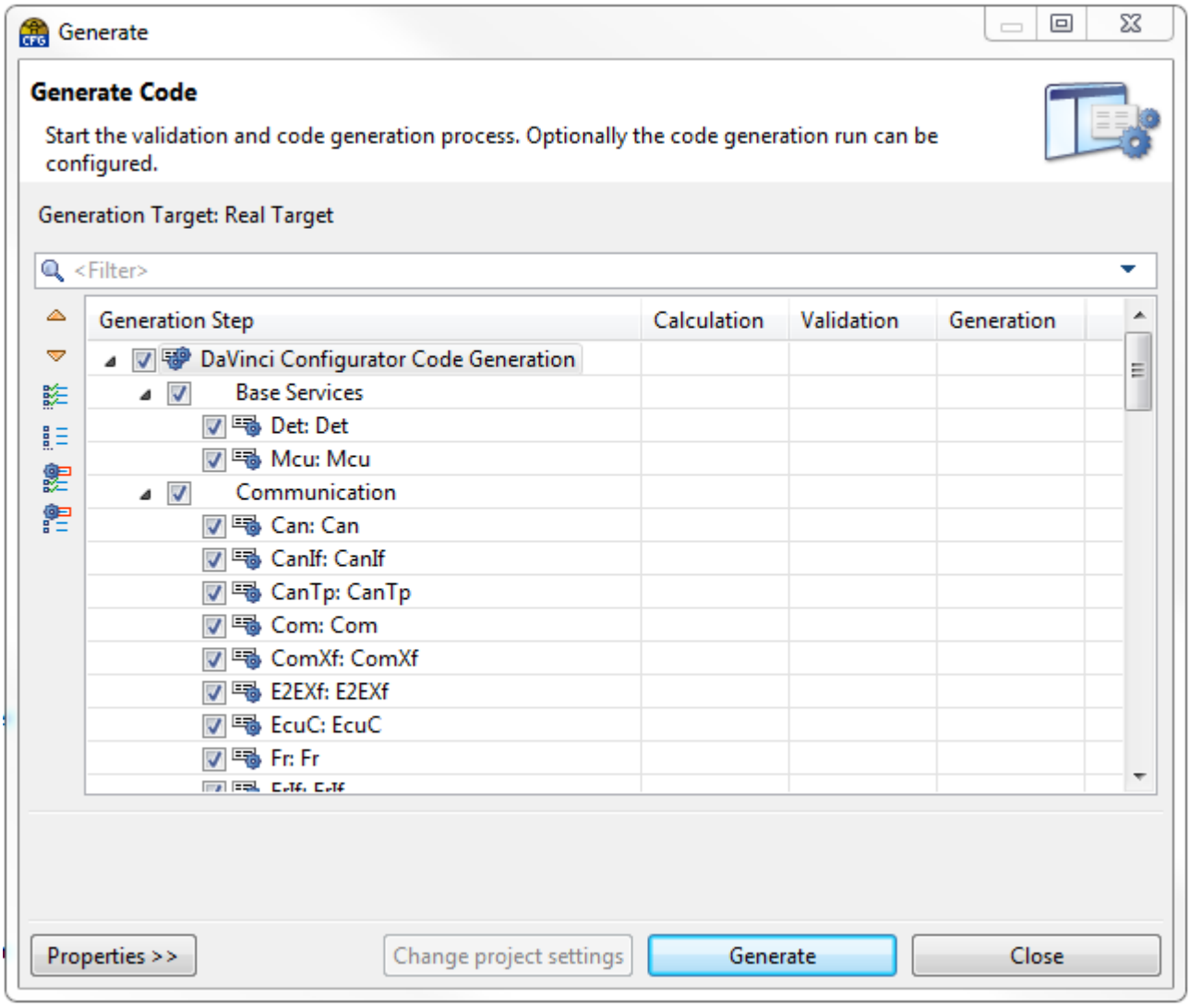
7.2 Start Code Generation

1. Open the Generation dialog via **Generate** button  .

Note

The activated generation steps depend on the project settings. If you want to take over your selection to the project settings press **[Change project settings]**.

2. Start Generation Process via **[Generate]**
   If you select **Start immediately** (see **Properties >>**) the generation starts immediately after opening this dialog.



Note

Before the **Generation** process starts, **Calculation** and **Validation** has to be finished successfully.
If code generation canceled because of calculation and validation errors, start solving actions at the validation window of the DaVinci Configurator Pro (see section STEP3 Validation on page 43).

7.3 Generation Process finished!

Generation Process finished and all generation steps are marked with Phase finished successfully ?

Note

The DaVinci Configurator Pro stores all generated *.c and *.h file in the modules files folder, given during project setup (the default is **<ProjectFolder>\Appl\GenData**).

Go on with implementing your runnables using the runnable skeletons created during the generation process.

8 STEP8 Add Runnable Code

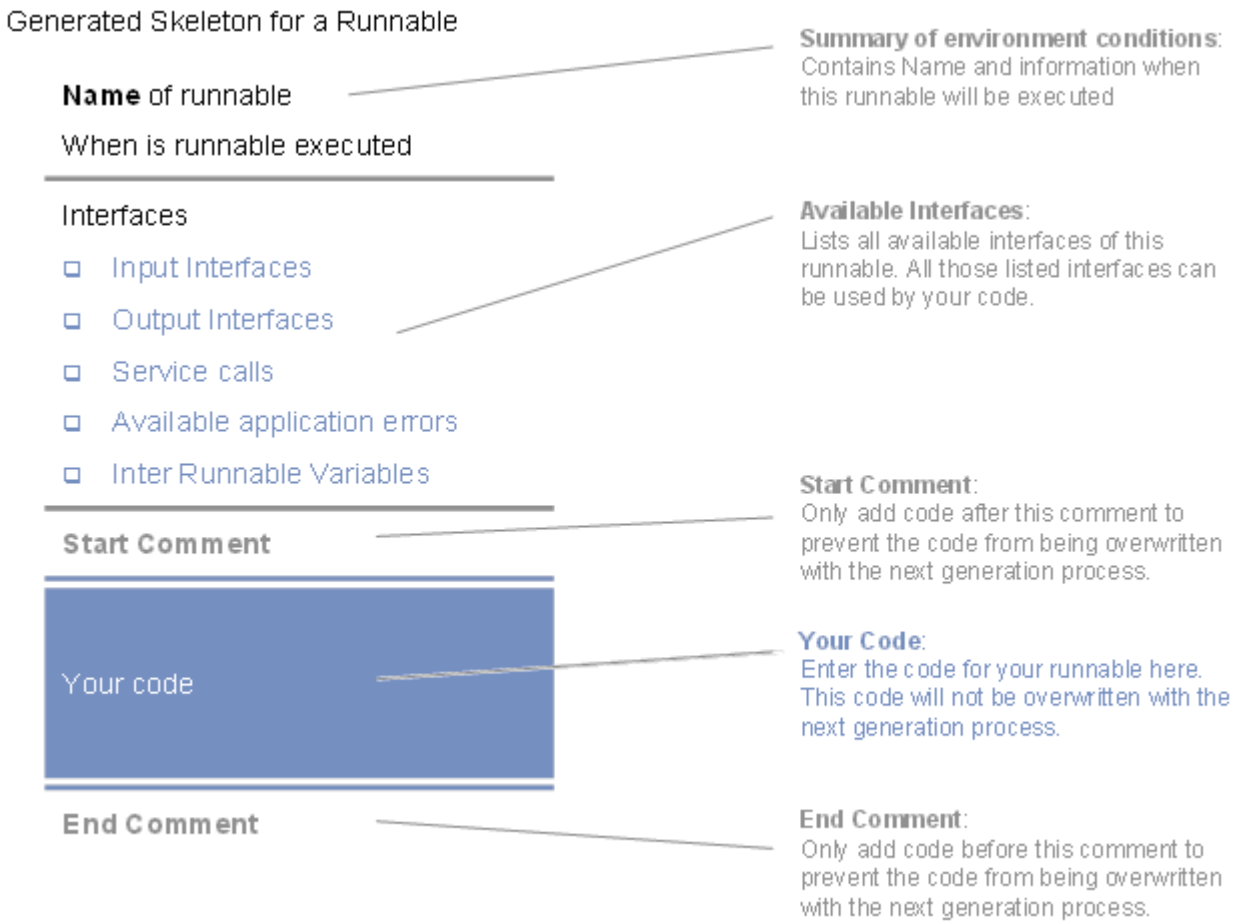
If configured, the DaVinci Configurator Pro code generator generates the skeletons for your runnables into the software component template files. Now you have to implement your code to the runnable skeletons.

Expert Knowledge

You need to know more? See section Runnable Code on page 137

8.1 Component Template

The software component templates contain all runnables of the according software components. The RTE Template Generator generates skeletons for the runnables. The structure of those skeletons looks as shown in the illustration below.



8.2 Implement Code

Implement your code using the runnable skeletons. Make sure to enter your code between the start and end comment. Only this section is protected against modification by a further generation process.

```
FUNC(void, RTE_AP_MYSWC_APPL_CODE) MySCW_Code(void)
{
/**********************************************************************************************************************
 * DO NOT CHANGE THIS COMMENT!          << Start of runnable implementation >>          DO NOT CHANGE THIS COMMENT!
 * Symbol: MySCW_Code
 *********************************************************************************************************************/
Boolean dataLeft, dataRight, dataLeftRear, dataRightRear;

Std_ReturnType value;


Rte_Read_FrontLeftDoorState_OpenClose(&dataLeft);
Rte_Read_FrontRightDoorState_OpenClose(&dataRight);
Rte_Read_RearInformation_RearLeftDoor(&dataLeftRear);
Rte_Read_RearInformation_RearRightDoor(&dataRightRear);


if (dataLeft || dataRight || dataLeftRear || dataRightRear)
{
    Rte_Call_UR_ComMUser_CAN_GetRequestedComMode(&value);
    if (value != COMM_FULL_COMMUNICATION)
    {
        Rte_Call_UR_ComMUser_CAN_RequestComMode(COMM_FULL_COMMUNICATION);
    }

    Rte_Write_FrontLightState_OnOff(TRUE);
    Rte_Write_RearLightControl_SwitchRearInteriorLight(TRUE);
}
else
{
    Rte_Write_FrontLightState_OnOff(FALSE);
    Rte_Write_RearLightControl_SwitchRearInteriorLight(FALSE);

    Rte_Call_UR_ComMUser_CAN_GetRequestedComMode(&value);
    if ((value != COMM_NO_COMMUNICATION) && (lightDimControl == 0))
    {
        Rte_Call_UR_ComMUser_CAN_RequestComMode(COMM_NO_COMMUNICATION);
    }
}
/**********************************************************************************************************************
 * DO NOT CHANGE THIS COMMENT!          << End of runnable implementation >>          DO NOT CHANGE THIS COMMENT!
 *********************************************************************************************************************/
}
```

All other sections like execution information or interfaces can be changed depending on your settings in the DaVinci Developer. If access to e.g. a port interface is missing, go back to the DaVinci Developer and adapt your configuration, regenerate and then you can use this port interface. There is no other way to do it properly.

Info

A backup (*.bak) file will be generated before a component template C file is modified to make sure that your previous version is not deleted in case of any generation problem caused by non-predictable reasons.

And there is a backup section at the end of the template file to rescue the code of runnables that have been deleted by the DaVinci Developer.

9 STEP9 Compile, Link And Test Your Project

Now you can compile, link and test to get your executable for the download to your hardware.

Expert Knowledge

You need to know more? See section Compile and Link on page 139.

9.1 Finish your project with compiling and linking

This step is highly dependent on your compiler, linker and project settings.

1. Compile and Link your code.
2. Test your code
   A proper compile and link process is just the beginning. The final step is to test what you have done until now. The ideal tester would be CANoe from Vector Informatik.

To test your application with CANoe, the following steps are necessary:

> Create a new CANoe configuration
> Import the communication matrix (e.g. *.dbc file)
> Right-click on the ECU to be simulated, and click on **Configuration**
> Open the **Components** tab, add another node layer DLL and select the *.dll file, you have created with the Microsoft Visual Studio solution in the previous steps.
  This completes the basic setup for testing your application. See CANoe references for further information.
> Start the simulation.
> Have a look at the trace window and observe bus communication on your virtual ECU.

3. Debugging the application
   When the simulation in CANoe is started, switch to Microsoft Visual Studio and select **Debug | Attach to process ...**. Select **RuntimeKernel.exe** and make sure that the code type (Attach to) is set to Native code. Now you can debug your code. For example, try to set a breakpoint in a cyclic runnable!

9.2 No error frames? Congratulations, that's it!

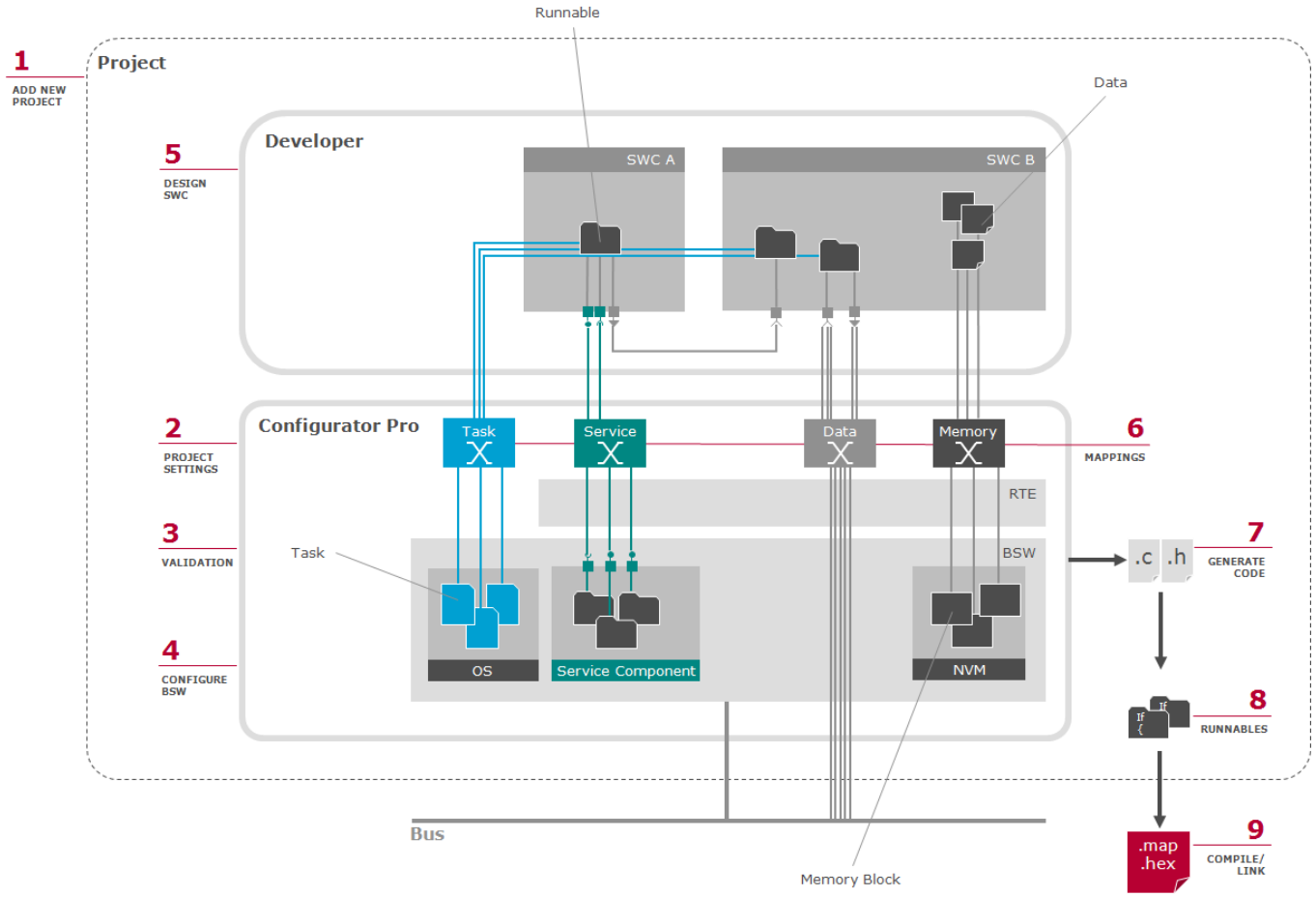
The basic step is done, the Vector SLP4 software is basically working together with your application.

That is the base you can start from to optimize your system. Refer to Technical References of the BSW modules for more detailed information.

Cross Reference

You need additional information? See section Additional Information on page 140.

II Concept


Runnable
1
ADD NEW PROJECT
Project
Data
5
DESIGN SWC
Developer
SWC A
SWC B
2
PROJECT SETTINGS
Configurator Pro
Task
Service
Data
Memory
6
MAPPINGS
RTE
3
VALIDATION
Task
BSW
.c
.h
7
GENERATE CODE
4
CONFIGURE BSW
OS
Service Component
NVM
8
RUNNABLES
Bus
.map
.hex
9
COMPILE/ LINK
Memory Block

1 General Overview

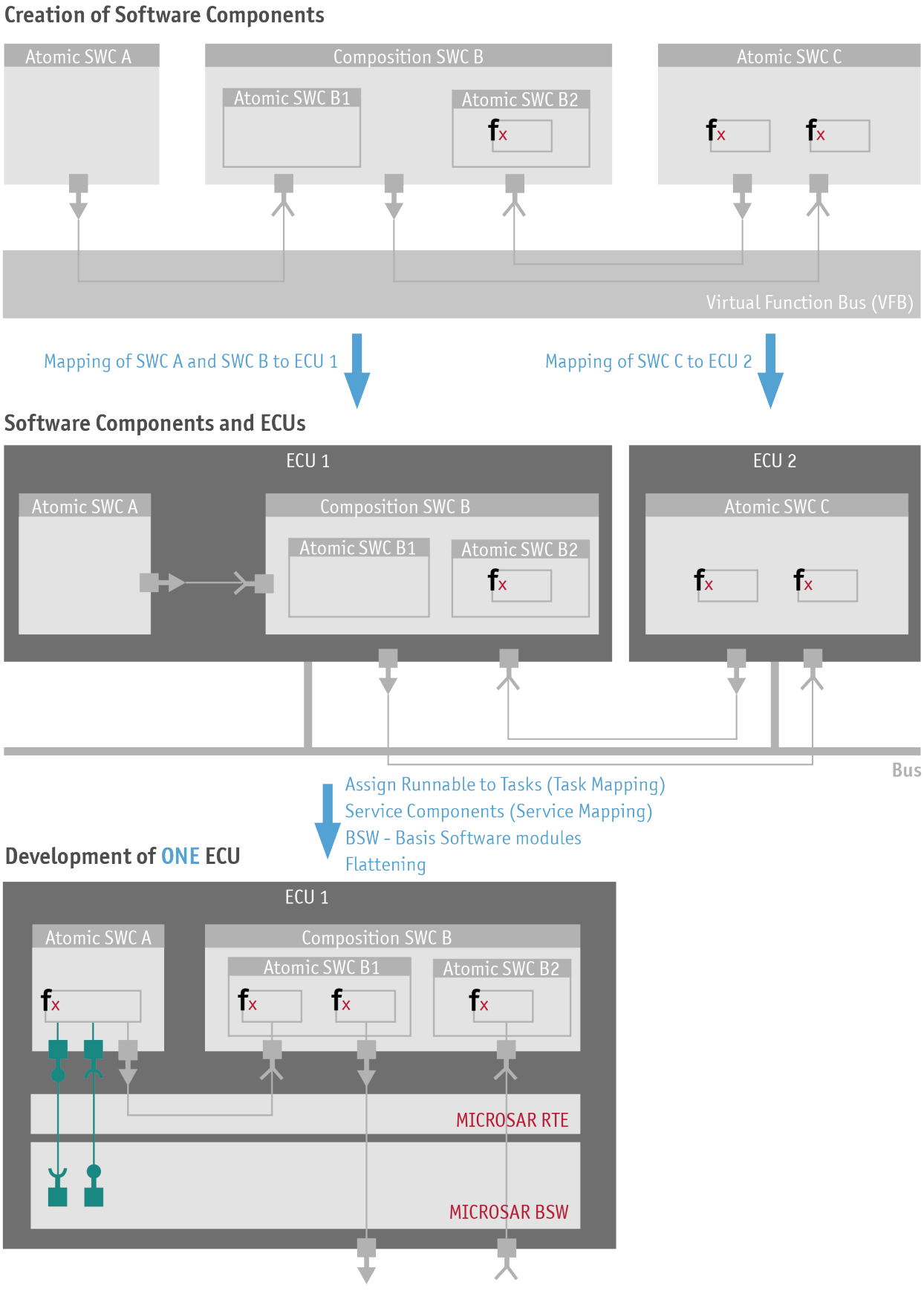
The following illustration shows the basic idea behind AUTOSAR and can be divided up into 3 parts, upper, middle and lower part, named as

- Creation of Software Components
- Software Components and ECUs
- Development of ONE ECU

The upper part deals with software components, runnables, ports and connections. It is the design level of functions and applications.

During the transition to the middle part, the software components are assigned to ECUs. Software components that are assigned to the same ECU can communicate internally, via so-called internal connections. Software components, that communicate with each other and that are mapped to different ECUs communicate via so-called external connection, i.e. via a real bus system.

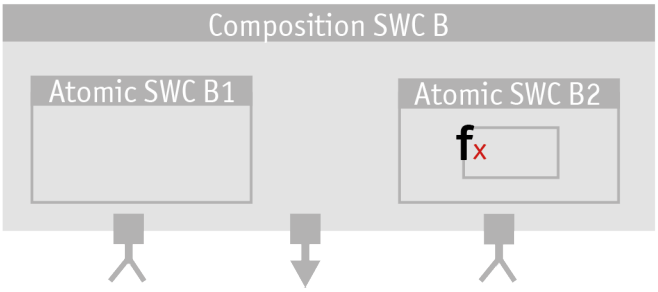
In the last transition towards the ECU sight, the BSW is put in-between the software components and the hardware. And this is the starting point for the ECU development.


Creation of Software Components
Atomic SWC A
Composition SWC B
Atomic SWC B1
Atomic SWC B2
Atomic SWC C
Virtual Function Bus (VFB)
Mapping of SWC A and SWC B to ECU 1
Mapping of SWC C to ECU 2
Software Components and ECUs
ECU 1
ECU 2
Bus
Assign Runnable to Tasks (Task Mapping)
Service Components (Service Mapping)
BSW - Basis Software modules
Flattening
Development of ONE ECU
MICROSAR RTE
MICROSAR BSW

The following sub chapters explain all the means that belong to AUTOSAR and that make it work.

1.1 Software Component

Software Components are described by their SWC file - a file in XML format. By definition SWC files are independent from any ECU except for the sensor and actuator software components. Those software components are dependent on the sensors and actuators attached to a specific ECU.



There are several kinds of software components

1.1.1 Atomic components

- Application
- Sensor Actuator
- Calibration
- I/O Hardware Abstraction
- Complex Driver
- Application (End-to-end Protection)
- Non-Volatile Memory Block
- Service components

1.1.2 Compositions

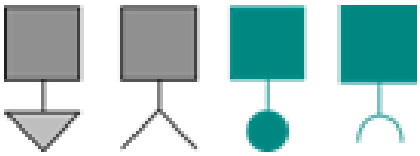
Use compositions to group software components.

1.2 Runnables

A runnable entity, runnable for short, is a C function that carries your code. It depends on the configuration of the runnable when it is called and what data your code is allowed to access within the runnable. Runnables are triggered by the RTE.

1.3 Ports

Via ports, a software component can communicate with its outside world. This outside world can be another software component within the same ECU or a software component within another ECU. There are several kinds of ports:



1.3.1 Application Port Interfaces

To communicate between different SWCs

- Sender port
- Receiver port
- Client port
- Server port
- Calibration port
- Mode port
- Interface to non-volatile data

1.3.2 Service Port Interfaces

To communicate between BSW and SWCs

- Sender port
- Receiver port
- Client port
- Server port
- Calibration port
- Mode port
- Interface to non-volatile data
- NV Data Interface

1.4 Data Element Types

Data element types determine the kind of data that can pass a port. There are predefined data element types and you can define data elements on your own.

1.5 Connections

To define which software components communicate with each other, their ports have to be connected via so-called connectors. The connectors realize the sender/receiver and client/server communication

between the software components. A connector can only be drawn between two ports, if their port interfaces are compatible.

1.6 RTE

The RTE is the interface between the SWCs and the BSW. In essence the RTE

- controls the execution of the Runnables (your code),
- controls access of Runnables to the basic software modules,
- controls the access of Runnables to Services of the BSW,
- knows about external and internal transmission of data and
- provides data consistency

1.7 BSW – Basic Software Modules

The BSW contains configurable modules that offer services to the SWCs dealing with:

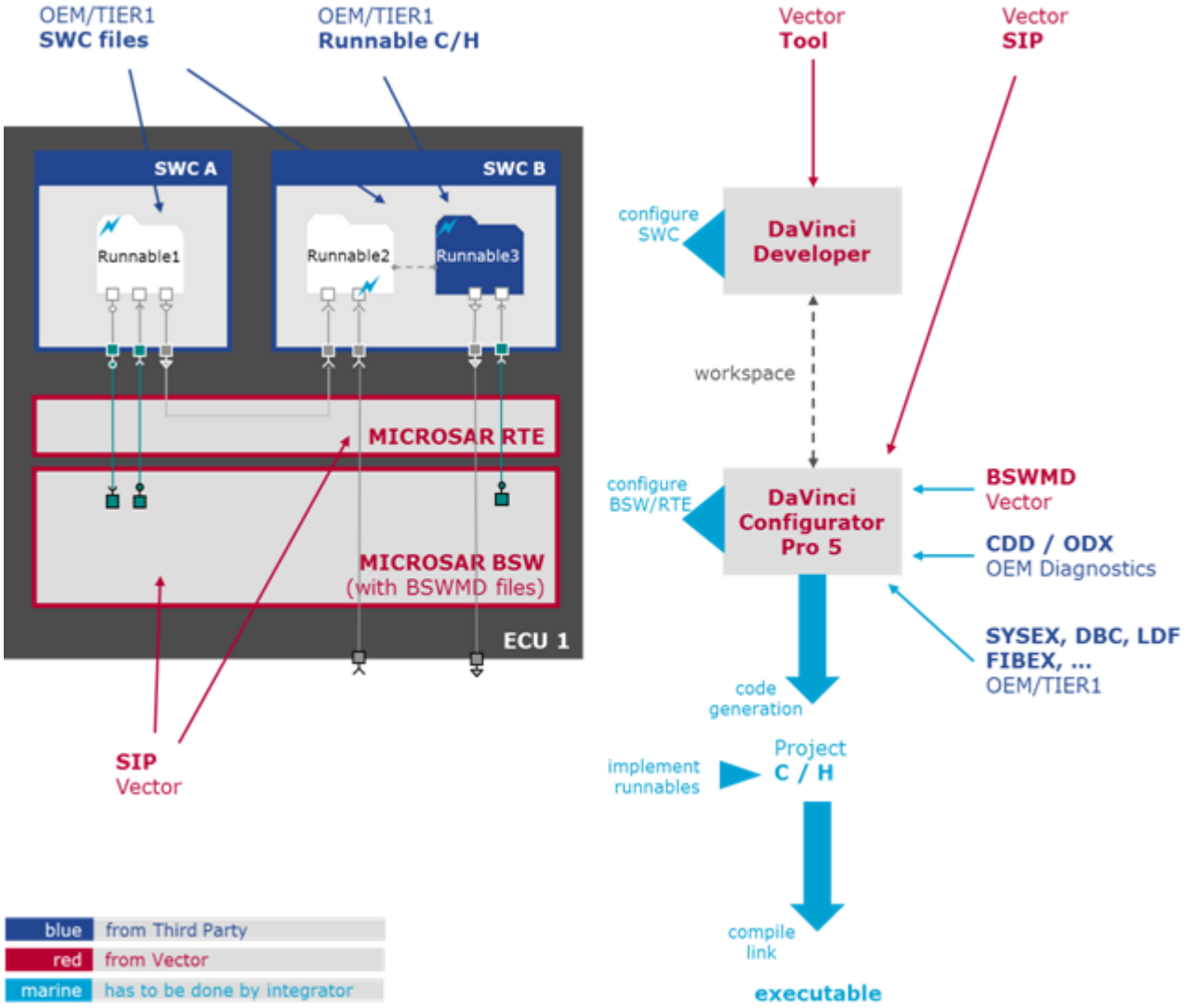
- Communication
- Diagnostics
- Mode management
- Memory management
- Network management

1.8 Software, Tools and Files

When setting up a new project using the DaVinci Configurator Pro, many things that are not necessary to know for your basic work with the tool happen in the background. But sometimes it might be necessary to know more. Then this part of the document is the place to get a deeper insight.

This chapter gives you an overview what kind of files you need to set up a new project, where the files normally come from and how to work with the DaVinci Configurator Pro and, if necessary for software component design, with DaVinci Developer. Get familiar with what is installed where and how to work with it and to know, how to update if necessary.

Starting point is the development of a single ECU. The focus is set on the MICROSAR Solution and where all the necessary software and files do come from.



To build up a project according AUTOSAR with MICROSAR from Vector you need the following tool(s), software and files:

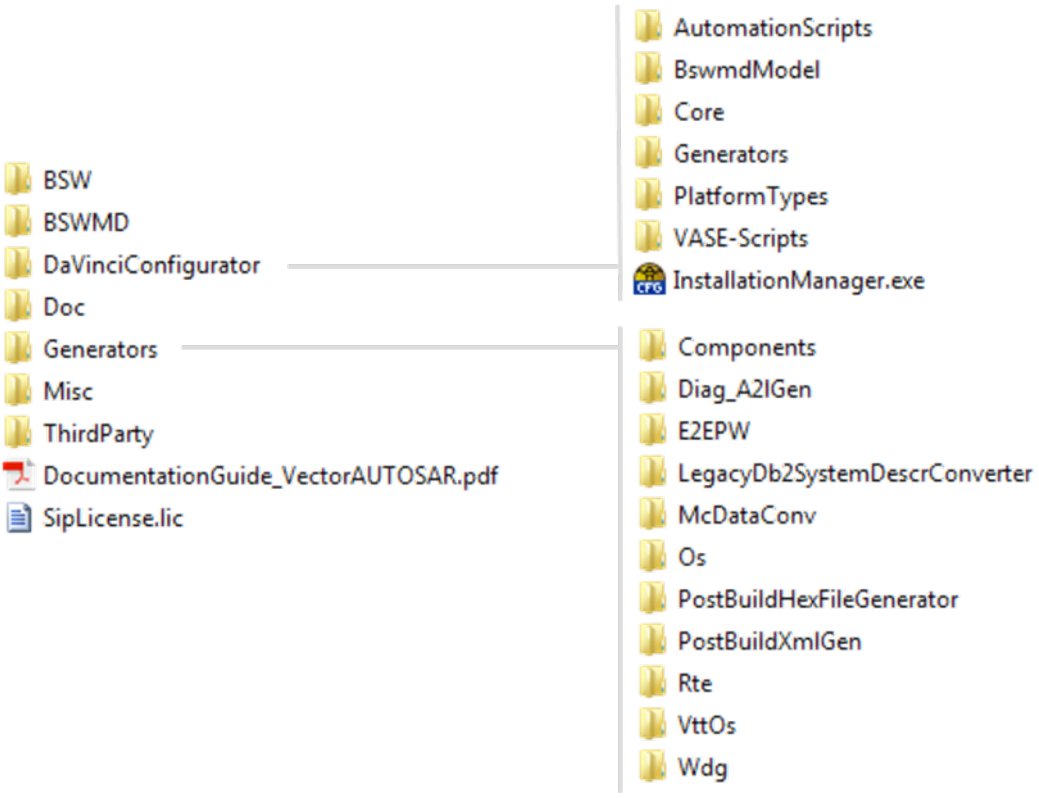
- **Software Components (SWC) and Runnables**
  Software components are created by the OEM or the TIER1. A software component can be an SWC file, where you have to perform the task mapping and decide about runnables and the code they should contain. Or the software component is delivered ready-made, together with C and H files. Then you only have to map it to a task and compile and link the files together with your project.
- **DaVinci Developer (Vector Tool, has to be ordered separately - NOT included within the SIP)**
  The DaVinci Developer is the Vector Tool to create and configure Software Components. Via a graphical view you can draw software components, add ports, draw connections, etc. DaVinci Developer and DaVinci Configurator Pro exchange information via the DaVinci Developer Workspace.

> **BSW and RTE (included in SIP)**
The BSW modules together with their BSWMD files (dependent on your order) and the RTE are part of the Vector delivery that is called SIP. **SIP** stands for **Software Integration Package**. A SIP is a ready-made software from Vector, already tailored and tested for your hardware, derivative, compiler, compiler version (as you filled-out the questionnaire) and can be put to work within a few steps (see step by step description).

> **DaVinci Configurator Pro (included in SIP)**
The DaVinci Configurator Pro is the Vector Tool to configure the BSW and the RTE. It is part of the SIP delivery. It contains a comfort editor to configure the BSW cluster-oriented, a powerful validation concept to check your settings against constraints and it also offers to directly edit AUTOSAR parameters.

1.9 Structure of the SIP Folder

The screenshot below shows the file structure that is created by installation of the SIP.



> **BSW**
Contains the code files (*.C and *.H) files for each Basic Software Module (BSW).

> **BSWMD**
Contains the BSWMD files for each Basic Software Module, which include necessary information for DaVinci Configurator Pro.

- **DaVinciConfigurator**
  Contains the DaVinci Configurator Pro tool itself. This is also the place where your tool is updated when an update is available.
- **Doc**
  Contains SIP relevant documentation e.g. Technical References, User Manuals, Startups (containing Release Notes) and Application Notes.
- **Generators**
  Contains Generators and additional tools
- **Misc**
  Contains SIP-specific additional data
- **ThirdParty**
  Contains MCAL-specific data

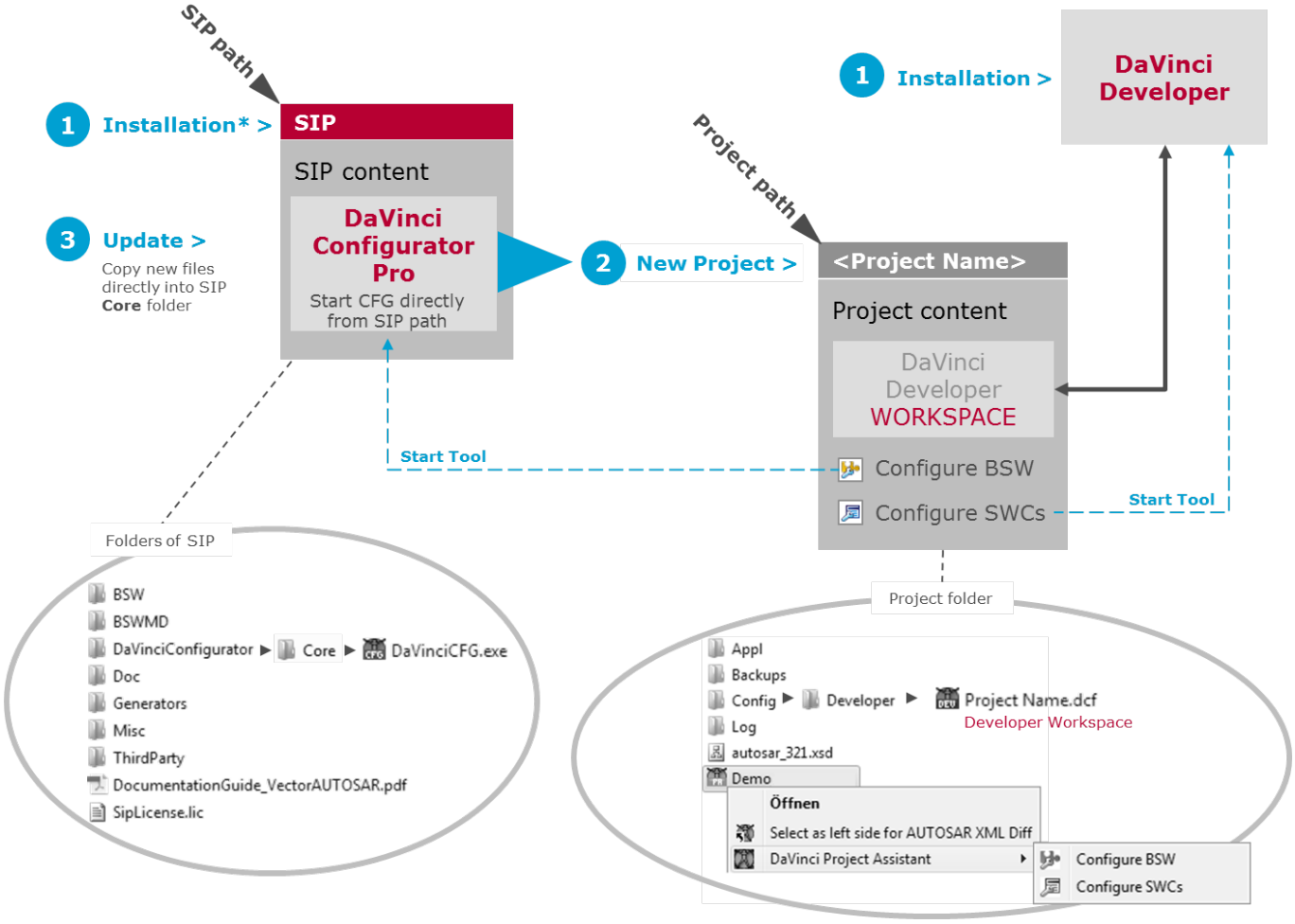
Note

Each MCAL is listed within a separate folder (<MCAL Short Name>).

This folder includes:

- **\Supply**
  Location of the 3rd party MCAL delivery in its original structure
- **\VectorIntegration** (optional)
  Location of scripts for the integration into the MICROSAR SIP
- Legal notes (optional)

See **TechnicalReference_3rdParty-MCAL-Integration.pdf** for detailed information.



*** DVCfgExternalComponentSetup.exe**

Installation of external components for the DaVinci Configurator Pro are only necessary for the first installation of a SIP and in some seldom cases together with an update. See more about this in the InstallationGuide.pdf

2 Set-Up New Project

DaVinci Configurator Pro offers a Project Assistant that helps you setting up a new project file in just a few steps. The assistant guides you through several windows to enter necessary project information.

The DaVinci Configurator Pro creates your new project to the defined folder. The project folder contains:

- **Appl** folder for generated files and source files.
- **Backup** folder to store older DaVinci Developer workspaces
- **Config** folder for software components, BSWMD files, ECUC files, System description and DaVinci Developer workspace
- **Log** folder
- Access to DaVinci Developer and DaVinci Configurator Pro via context menu on the **DPA file**.

Your development project consists of a folder structure on your disk, where all the configuration files, template files, etc. are generated to and where you implement your runnables using the empty runnable skeletons. Then you compile and link and get your executable that you can download to your hardware.

The DaVinci project file (DPA) is the central file which references all the related folders and files. It also references the SIP, which is used for the project.

2.1 DaVinci Configurator

As already mentioned the DaVinci Configurator Pro is the major tool for Vector MICROSAR for AUTOSAR.

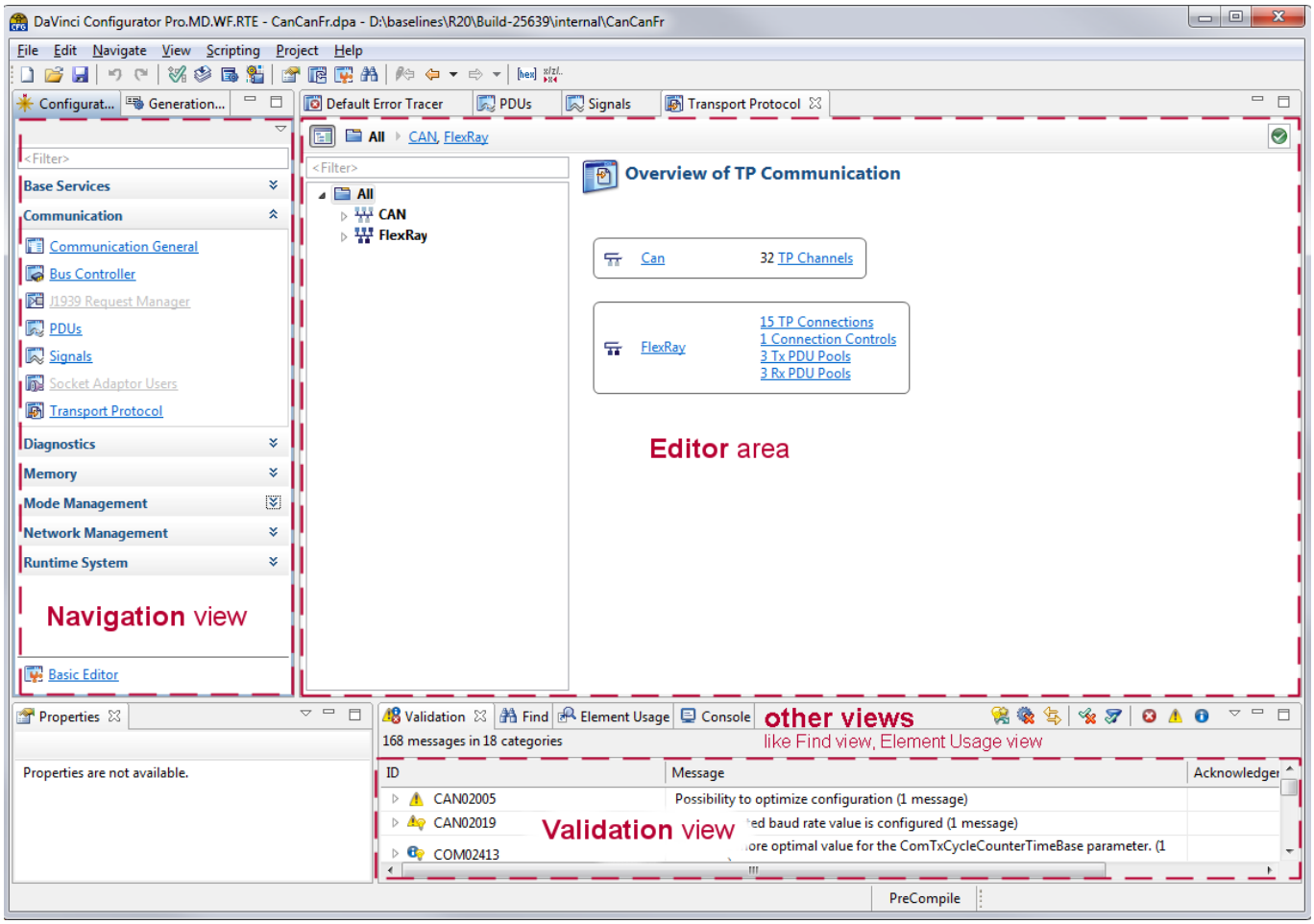
2.1.1 The Main Window of DaVinci Configurator Pro

The illustration below shows the main appearance of the DaVinci Configurator Pro. It consists of 3 main areas or views. The positions of the views can be defined manually and depend on your habits in handling with tools.

- Navigation view
- Editor Area
- Validation, Output and Properties view

Cross Reference

For more information about simple tool handling and icons refer to the help of the DaVinci Configurator Pro.



Navigation view
The Navigation view displays all available editors and assistants sorted by domain. What you select in the Navigation view will be displayed in the Editor view.

Editor area
The Editor area is the place to configure the modules or whatever you have selected in the Navigation view.

Validation view
The Validation view displays the overall list of validation messages. The messages are grouped by their ID. You may expand such a group to display all message of the same ID. By expanding an individual message, the related items are displayed as well as the proposed solutions, the so-called solving actions.

Using according commands in the shortcut menu you can navigate to the editors displaying the related items, or execute one of the proposed solutions for solving the problem.

You can use the Validation view with the following buttons:

> **Solve All**
Calls the default solving action of all messages that provide a default solving action.

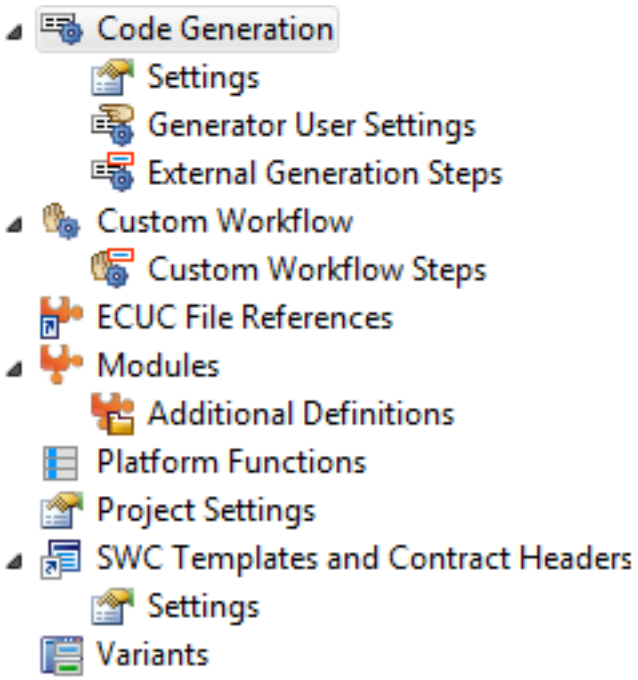
- **Clear On-Demand Validation Messages**
  Deletes all validation messages issued by the external generators during the on-demand validation.
- **Link with Editor**
  Filters the Validation View to display only messages related to the currently focused editor.
- **Filter Validation Display**
  Use the Icons **Show/Hide ...** to filter the display of your validation results.
  - Show/Hide Error Results
  - Show/Hide Warning Results
  - Show/Hide Info Results

Properties view
The Properties view displays details of the object (parameter or container) selected in an editor. The view contains the following tabs:

- **Definition**: Displays the definition of the selected object
- **Status**: Displays information about the parameter state
- **Description**: Displays the description of the parameter definition

Project Settings

This Overview shows all necessary project data and enables the configuration of customer workflow steps, the definition of external configuration steps and the activation of additional BSW modules.



Code Generation: Configure the code generation sequence

Settings: The code generation settings can be changed within this section

External Generation Steps: Define and execute a sequence of custom workflow step

Customer Workflow: Configure and execute a sequence of **custom workflow steps**

SWC Template and Contract Headers:Definition of additional pathes to BSWMD files relevant for the project.

Modules: Activate or deactivate modules

Additional Definitions: Definition of additional paths to BSWMD files relevant for the project.

ECUC File Reference: Include mechanism for external ECU configuration files from other projects.

Variants: Handling of Variants when used.

Project Settings: The project settings can be changed within this section.

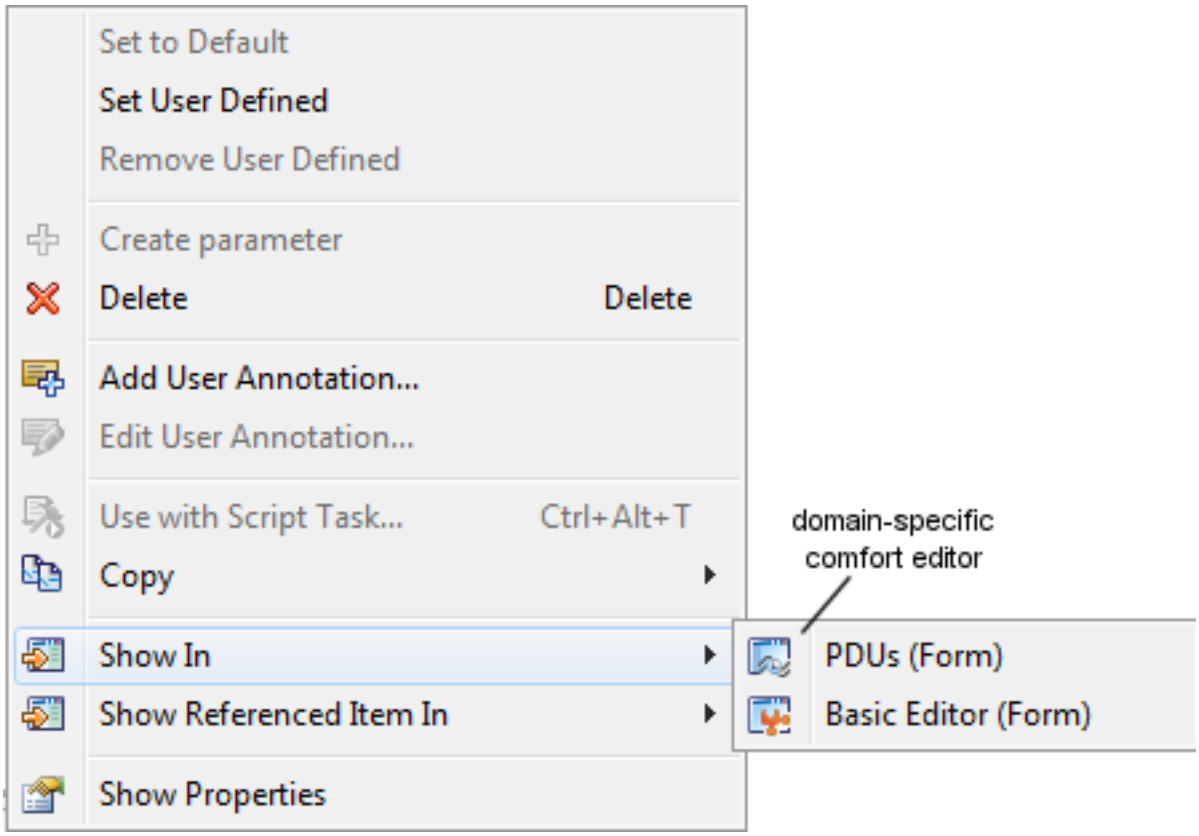
2.1.2 Editors and Assistants

Generally spoken there are two different configuration editor concepts in the DaVinci Configurator Pro.

- **Domain-specific** configuration editors, which provide an optimized view on the ECUC
- **Basic** Editor, which provides a native view on the ECUC

Assistants
Assistants guides you through special tasks like connections, data mapping, memory mapping or task mapping and more.

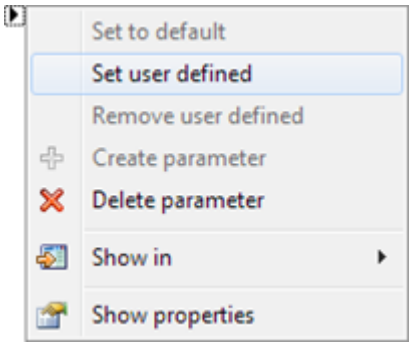
Switch Between the Two Editors
You can switch from the basic editor to the domain-specific comfort editor and back. Move to a setting or a value and click the little triangle to open the context menu. If there is a pendant in the other configuration editor, you can switch from one to the other via **Show in**.



Set User Defined

It is very important to understand that you are responsible for these modifications!!

Move to a setting or a value and click the little triangle to open the context menu.

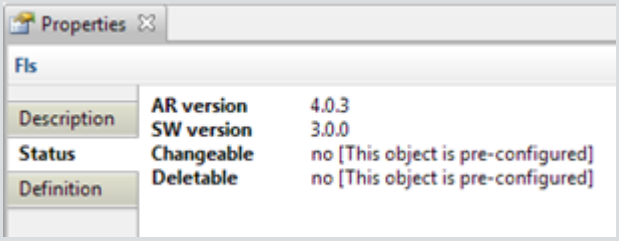


With **Set user defined** you can change values even when they are grayed because their values are derived from the input files or locked by a configuration dependency.

When you use **Set user defined**, the parameter will be marked with a little blue pin on the left side. In the example illustration below the upper check box is marked as user defined, the lower one is not.

Note

Set user defined will not work for pre-configured parameters. The setting in the context menu is grayed and cannot be used. The information about predefined or not can be seen in the **Properties** view. Select **Status** and look at **Changeable** or **Deletable**.



Remove user defined

To set back the user defined status, use the context menu and select **Remove user defined.**

Note

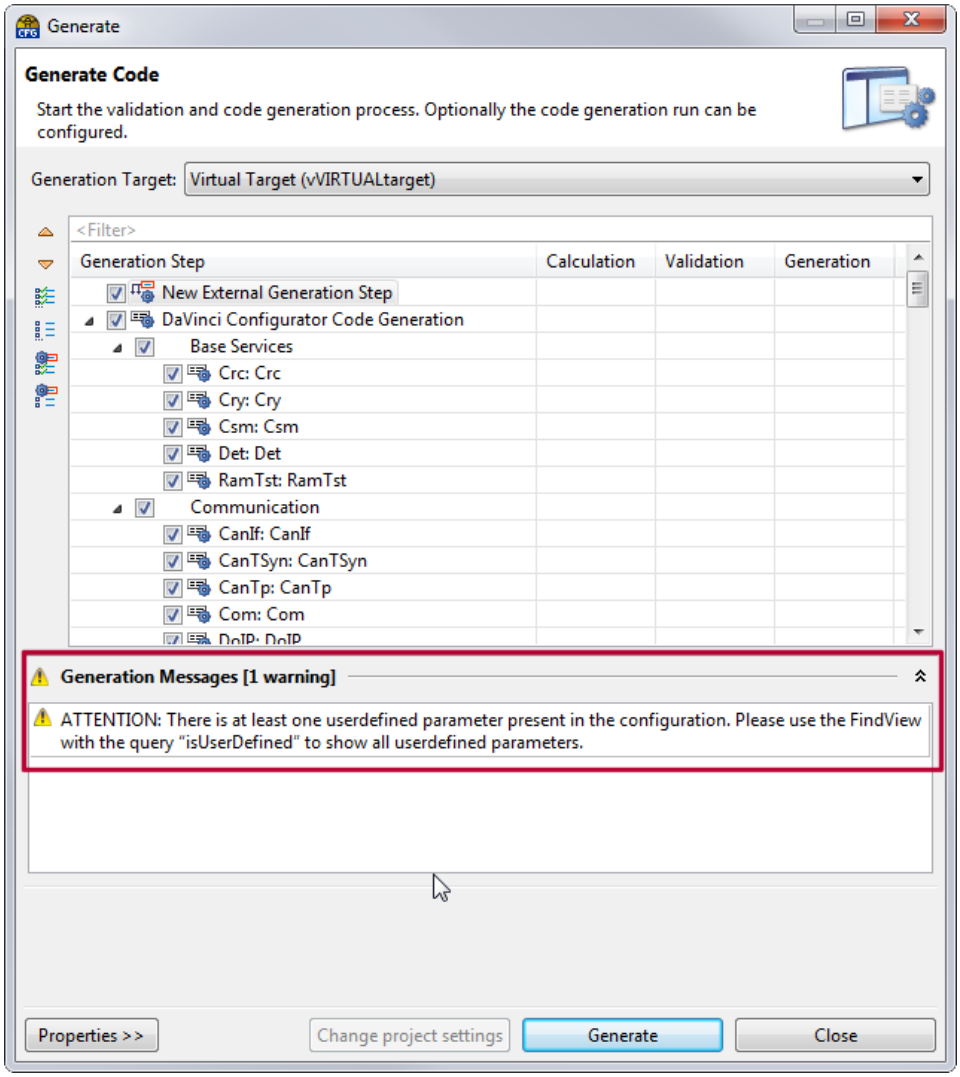
Every parameter set to user defined will provoke a warning during validation with the information, that this parameter was set to be user defined.

Caution!

The feature **set user defined** enables settings that might be terribly wrong. As soon as you use this feature, you have to be sure that you know what you do and you have the complete responsibility.

Generate Code and User Defined Values

If you have any parameter into your project set to user defined, this results in a warning before the code generation. To find all parameters, set to **user defined**, use the **Find View** with the query **isUserDefined**.

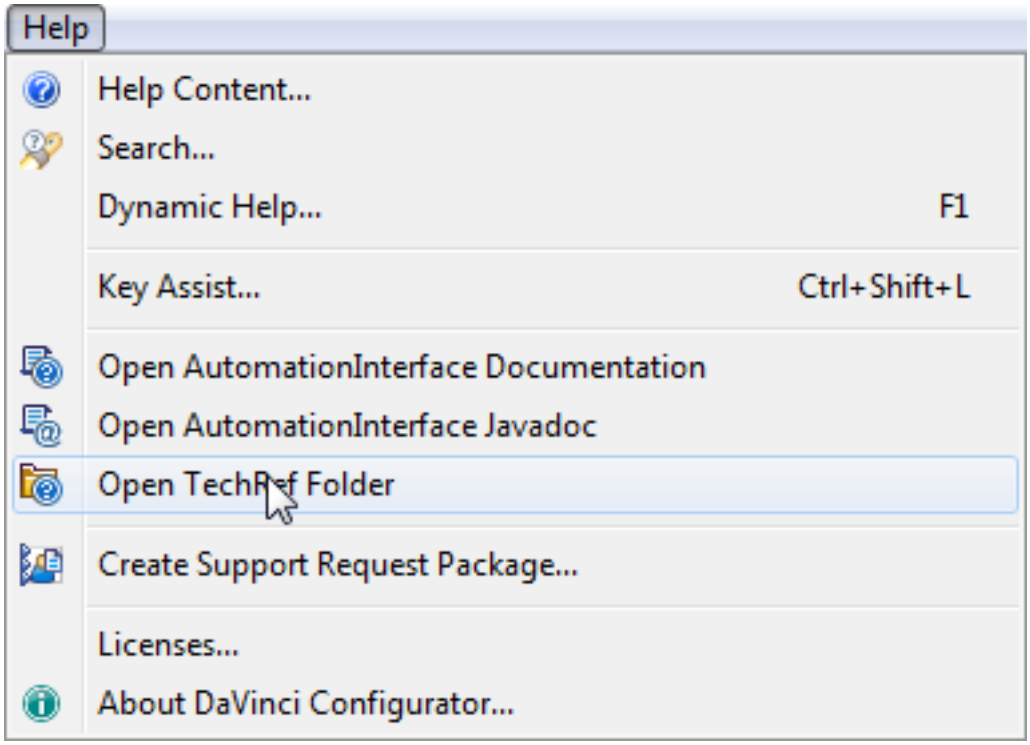


Cross Reference

ReleaseNote_MICROSAR4_Vector_*.pdf- **FEAT-3426**

2.2 Open Technical Reference Folder

The Technical References are documents that describe the MICROSAR modules in details. You find the Technical References for your delivery via this button **Help|Open TechRef Folder**



Cross Reference

ReleaseNote_MICROSAR4_Vector_*.pdf - **FEAT-3345**

3 Define Project Settings

Your first steps after the new project is set up, are:

- Define Input files
- Define your specific workflow steps and external generation steps (if needed)
- Activate all necessary BSWs

3.1 Input files

The DaVinci Configurator Pro needs information about your system, the bus communication, etc. This information is stored in the SYSEX or when using legacy file formats dependent on the bus system, in the **DBC**,**LDF**or **FIBEX** files. One or more (e.g. for systems with multiple channels) of these files must be given to the DaVinci Configurator as input files.

The necessary diagnostic information is stored in **CDD** or **ODX** files and is also needed by the DaVinci Configurator Pro.

3.1.1 System Description Files

You need one of the following description files:

3.1.2 SYSEX

ECU-specific extract of the System Description. This file is typically provided to the TIER1 by the OEM. It contains

- ECU composition
- Atomic SWCs
- Compositions
- Communication
- Data mapping
- typically NO service SWCs

3.1.3 ECUEX

Base for the ECU development is the ECU Extract of System Description, ECUEX for short. It is created from the SYSEX by flattening the SWC architecture. This means that SWC compositions have been removed, only the atomic SWCs are left. And they now communicate directly with each other. Additionally to the SYSEX, the ECUEX contains:

- Service SWCs
- Service mapping, i.e. service connections between the ports of the SWCS and the ports of the service SWCs.

3.1.4 Legacy Data Base files (DBC, LDF, FIBEX, ...)

The legacy data base files are normally not in ARXML format. They also contain information about ECUs, messages, signals, attributes, etc. Good AUTOSAR tools can read them and convert their information into AUTOSAR-conform notation. The missing information when using legacy file formats must

be given to the configuration tool by the software developer.

3.1.5 Diagnostic Data Files

Your Project uses Diagnostic? Therefore you need one of the following diagnostic description files:

3.1.6 CDD / ODX

These files contain the diagnostic description for the ECU.

- ***.CDD**
  The CANdelaStudio Document (*.cdd) format is specified by Vector for describing the diagnostic feature set of an ECU.
- ***.ODX/*.PDX**
  ODX (Open Diagnostic Data Exchange) is an ASAM standard which defines a unique, open XML exchange format for diagnostic data. A packaged ODX file (*.PDX) comprises all files of an ODX project.

3.1.7 State Description

Diagnostic session and security level management is modeled as a state machine which describes the transitions between sessions and states triggered by the execution of diagnostic services.

If a CANdela Document (*.CDD) is used as diagnostic description file, all diagnostic session and security level related information will be imported from the input file.

If a Packaged ODX file (*.PDX) is used as diagnostic description file, it will depend on the used ODX version if an additional state description is required.

- **ODX 2.2.0**
  For ODX 2.2.0, the state machines are defined within the ODX file, using the data model intended for this purpose.
- **ODX 2.0.1**
  As ODX 2.0.1 does not provide means to explicitly model diagnostic sessions, security levels and their corresponding transitions, dedicated SDGs (special data groups) are used to annotate the ODX data with supplemental information about the diagnostic session behavior. This information is not sufficient in all cases and an additional state description will be mandatory to augment the ODX data with the required information during the import of the diagnostic description.

A state description is a *.CSV (comma separated values) file which defines the transition matrices of the diagnostic session and security level state machines.

3.1.8 Standard Configuration Files

In some cases a OEM-specific preconfiguration is necessary, therefore additional Standard Configuration Files provided by your OEM are necessary. Add fragments of ECUC modules which will be used as project specific mandatory configuration.

3.2 External Generation Steps

In STEP7 Code Generation on page 78, after the configuration, DaVinci Configurator Pro generates code. You can additionally let DaVinci Configurator Pro call any external tool during this step in a free configurable order. If something is to be done before code generation, if you use some command line

tools or whatever, add these tools to the list and define the call order. With STEP7 Code Generation on page 78, all your mentioned tools will be also called.

3.3 Activate BSW

Which modules are activated and which are not depends on the information in the input files. Activate or deactivate the modules.

4 Validation

Nothing done but imported some files and already many mistakes in the configuration? Why is that?

The DaVinci Configurator Pro provides powerful validation routines that run in the background named **Live Validation**. At start-up of your project, there is only the information from the input files of STEP2 Define Project Settings on page 29. The content of these input files normally has leaks, sometimes errors or the information of the system is still missing and has to be accomplished during configuration. This all leads to a number of errors detected by the validator of DaVinci Configurator Pro. But with the powerful solving algorithms and assistants, it is very easy to solve off the first warnings and errors.

4.1 Validation Concept

The validation concept comprises:

- Detailed validation of the ECU configuration
- Comprehensive set of domain-specific validation rules
- Navigation from validation message to the editors
- Tool makes proposal for solving an error (solving actions)
- Automatic consistency (auto-solving action) after changing the configuration via Comfort Editors or via the Basic Editor

Live Validation

The Live Validation is performed in the background after loading a project or after changing the configuration. The Live Validation directly gives feedback after entering wrong values.

On-Demand Validation

This validation is performed when you start a generation process. You can start the validation via **Project | on-demand validation**. The on demand validation will also launch external generators or execute more complex validation tasks that cannot be executed live.

Note

The result of a validation is displayed in the **Validation View**.

5 BSW Configuration With Configuration Editors

5.1 DaVinci Configurator Pro Editors

To configure the BSW modules you can use the **Basic Editor** or the more comfortable and cluster-specific **Configuration Editors**.

Basic Editor 

The Basic Editor is a generic configuration editor (GCE) and includes the native view of the ECU configuration. It is based on the basic software module description (BSWMD) format and displays all modules of the ECU configuration.

Configuration Editors 

The Configuration Editors offer a use-case oriented view of the ECU configuration. They are optimized for displaying or configuring those parts of the ECU configuration, which are related to the use case.

Use the Configuration Editor first. To get the necessary settings for your fist step in dependency of your OEM and its use cases refer to Start Configuration with Configuration Editors of the step by step description.

6 Software Component (SWC) Design

6.1 DaVinci Developer Workspace - DCF

DaVinci Developer operates on a workspace (DCF file).

6.1.1 Standalone Workspace

If you don't work in combination with DaVinci Configurator Pro you start with creating a new workspace via the File menu. It is recommended to create the new workspace in a blank folder on your disk.

Show Illustration

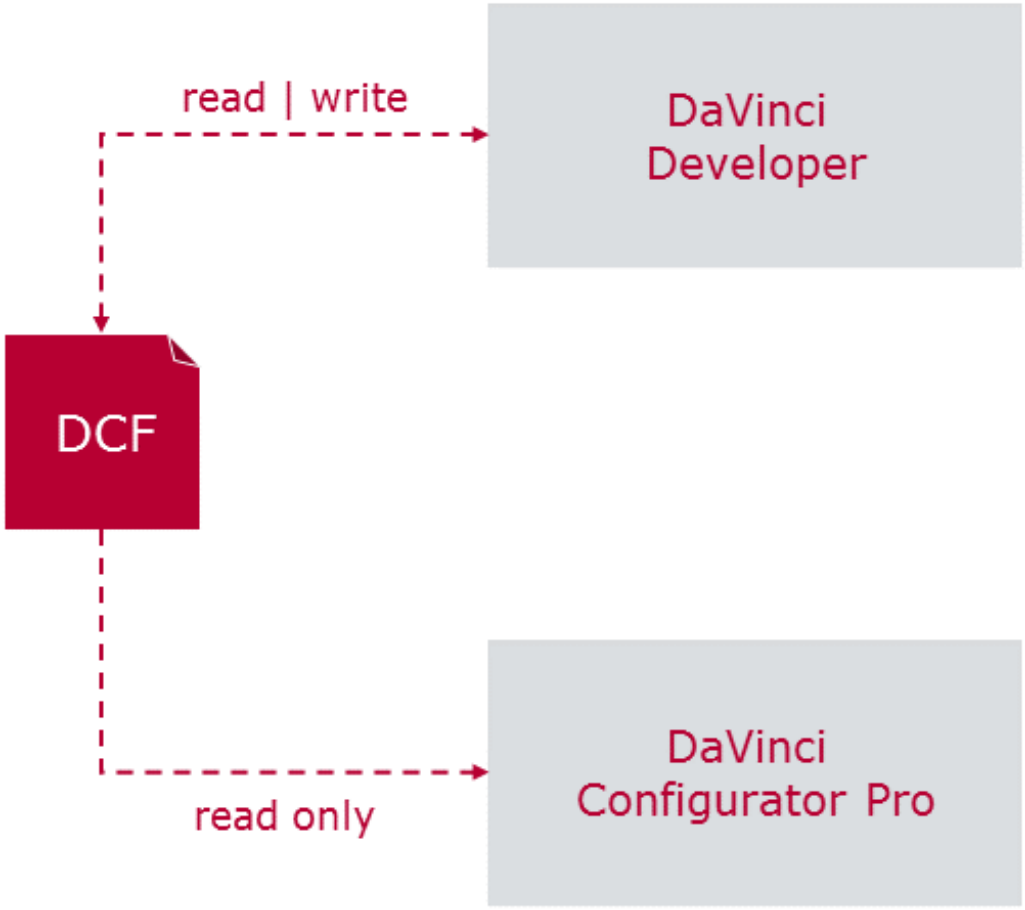
Note

Some editors of the DaVinci Developer, e.g. the Data Mapping Editor, will not be available in standalone workspaces

6.1.2 Workspace of the DaVinci Project

If you work in combination with DaVinci Configurator Pro, the DaVinci project created with DaVinci Configurator Pro already contains a DaVinci Developer workspace. You don't need to create it separately.

You edit the content of the workspace with DaVinci Developer. After saving the workspace, DaVinci Configurator Pro will detect the change and reload the data automatically. This also works vice-versa.

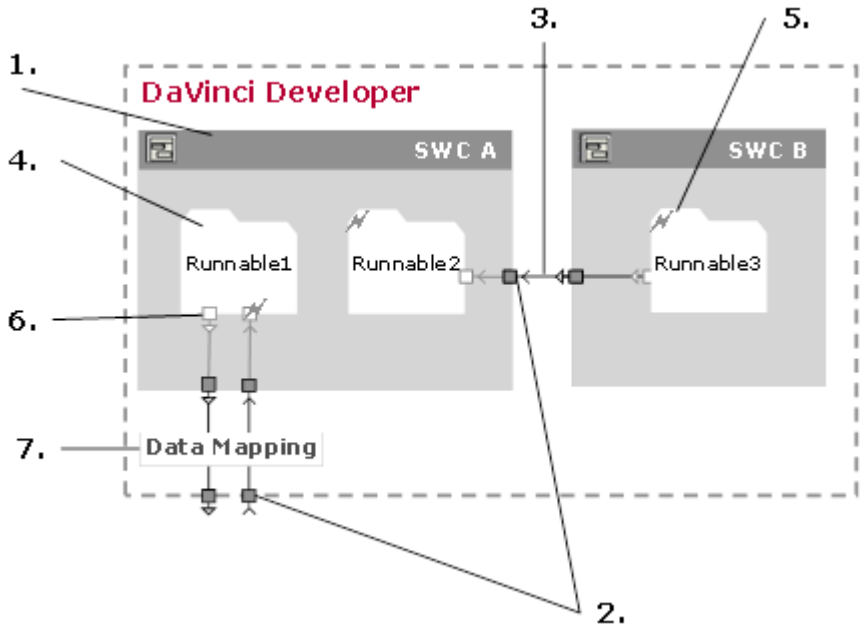


Caution!

Keep in mind that the DaVinci Configurator Pro is not able to write to DCF file.

6.2 About Application Components, Ports, Connections, Runnables and More...

Learn all about application components, ports, connection and runnables, how they work and how to exchange information between application components.



1. Application Components
2. Ports, Port Init Values and Data Elements
3. Connections
4. Runnables
5. Triggers
6. Port Access
7. Data Mapping

Cross Reference

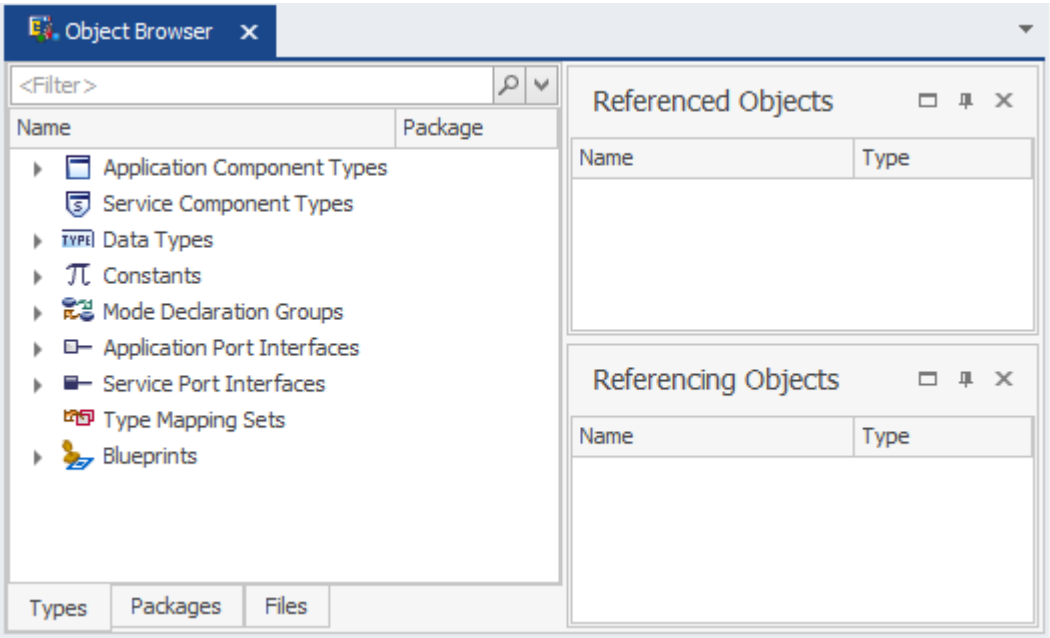
To get basic tool handling information, refer to the online help of the DaVinci Developer.

6.3 Application Components

To create application components use the DaVinci Developer.

6.3.1 The Object Browser – Types, Packages and Files

The Object Browser is the central view of all design objects in the workspace. Open it via the toolbar.



Types, Package or Files?

The Object Browser you use in the following steps can basically be used in different ways.

- Type-oriented
- Package-oriented
- File-oriented

The type-oriented workflow is shown in the following steps. But you can also choose the package-oriented or file-oriented one. This concept will now be described briefly.

Packages

With Packages you can pack elements together like:

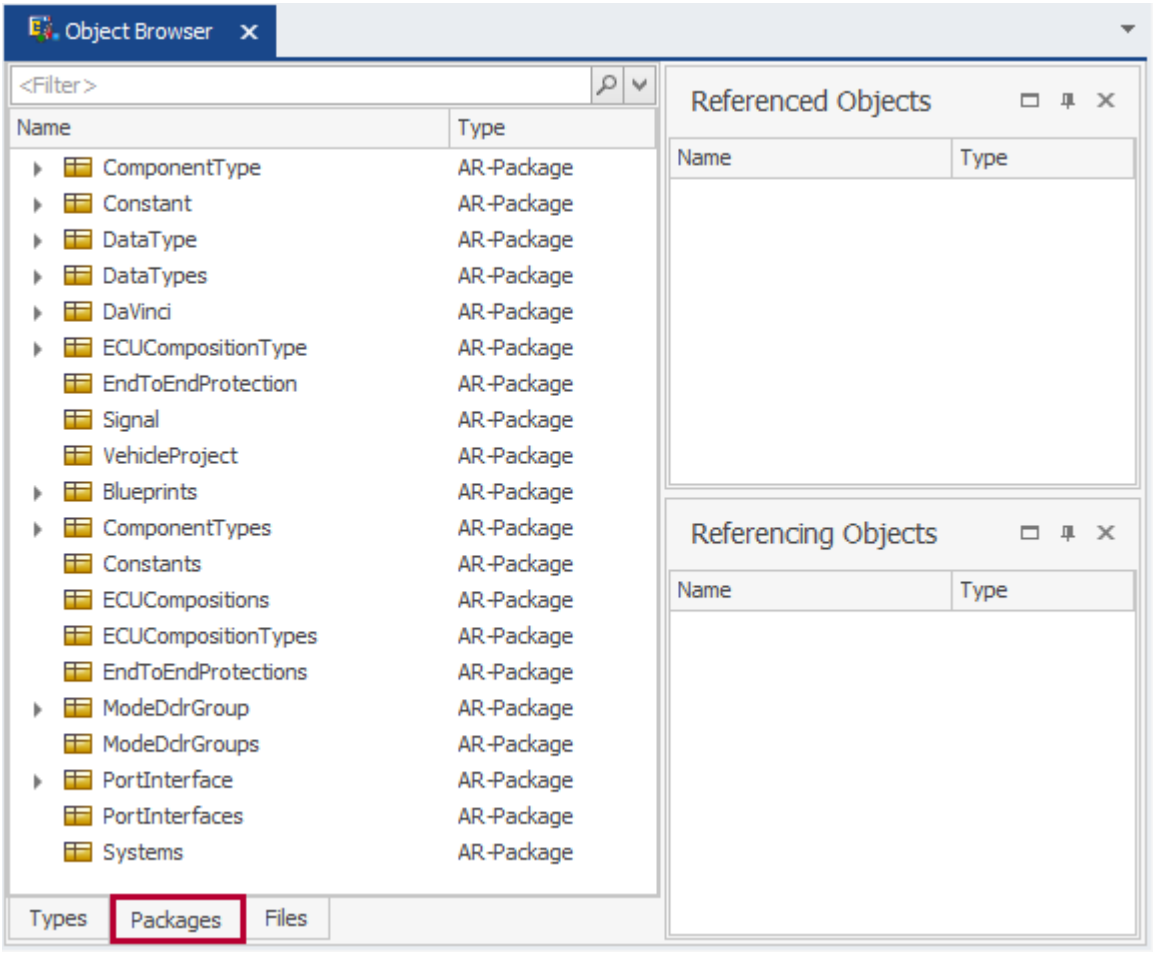
- Application Component Types
- Service Component Types
- Data Types
- Constants
- Mode Declaration Groups
- Application Port Interfaces
- Service Port Interfaces

It is like a kind of grouping with its own name space. A name of an object has to be unique within a package but can be used more than once within the project.

These packages can be exported and imported for round-tripping with other tools also supporting the packages.

Switch to package view

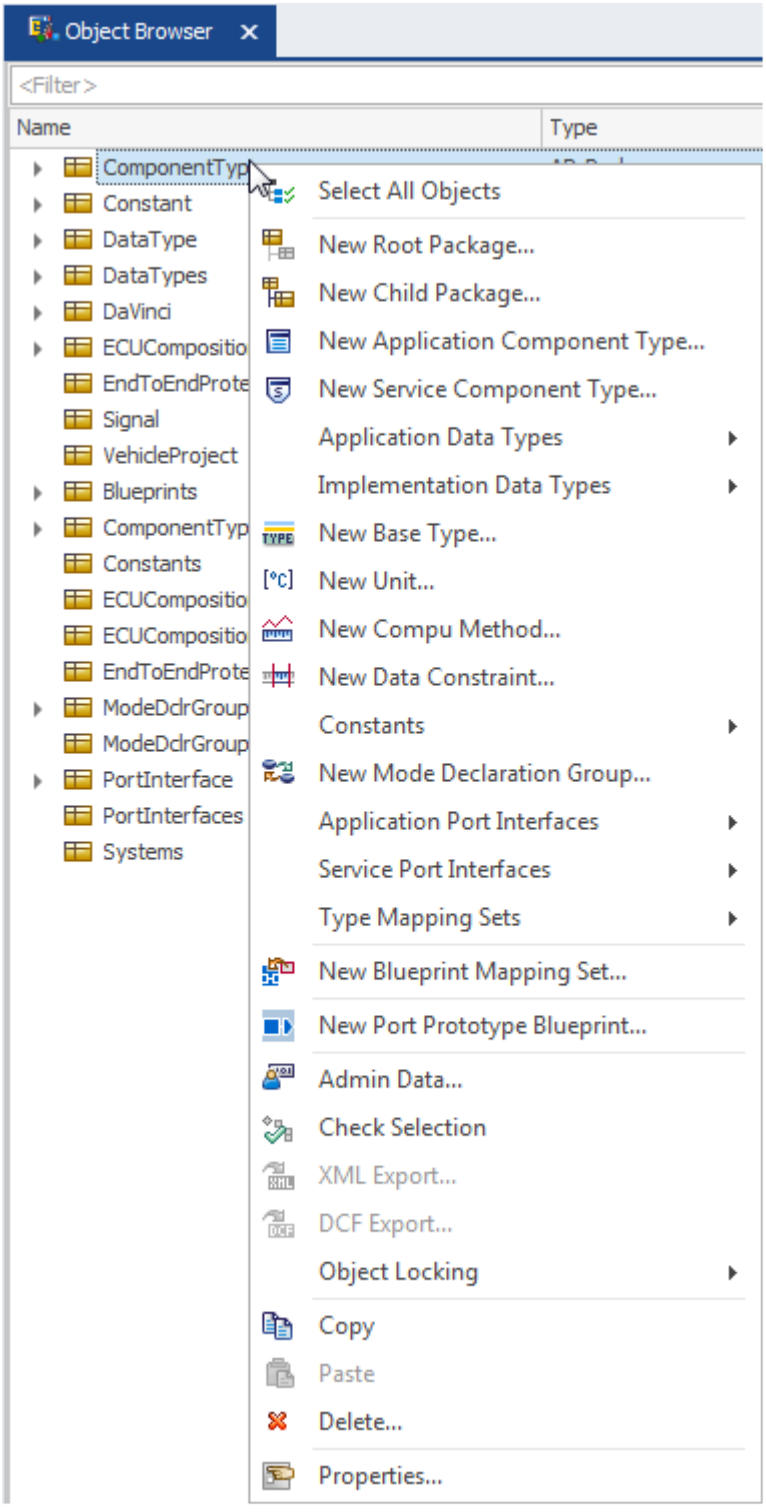
At the bottom of the Object Browser you can select Types view, Packages view or Files view. Select the Package.
You need a Root Package first? Create it via right-click in the Packages view.



New Child Package...

Then right-click the root (e.g. MyECU_PartA) and you will get to the context menu. Use New Child Package to nest the packages or you put the objects plain below the root package.

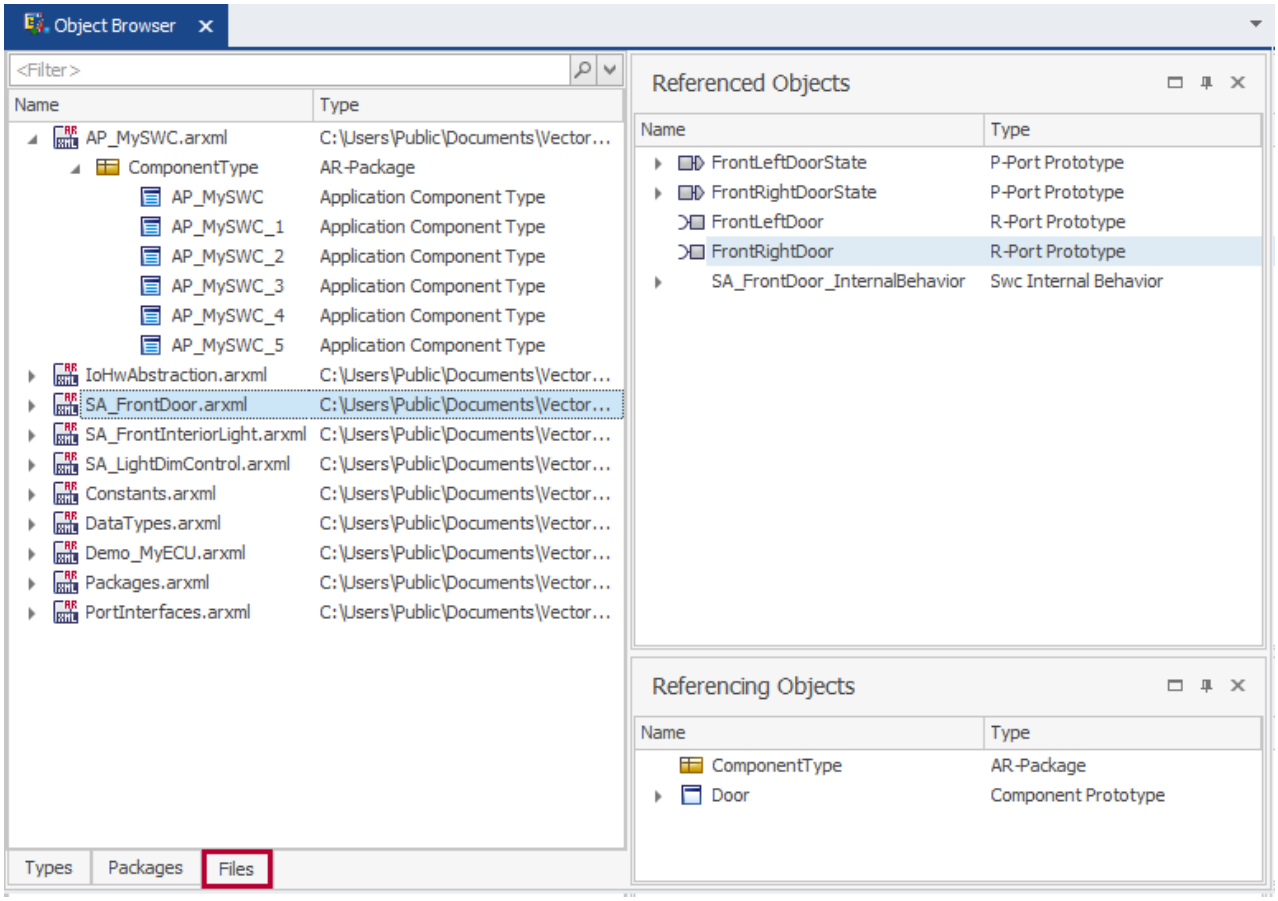
Context Menu for the Packages



Files

Within the **Files** tab you

> can see all AUTOSAR design objects, which are loaded from the particular file.
> move objects to other files.

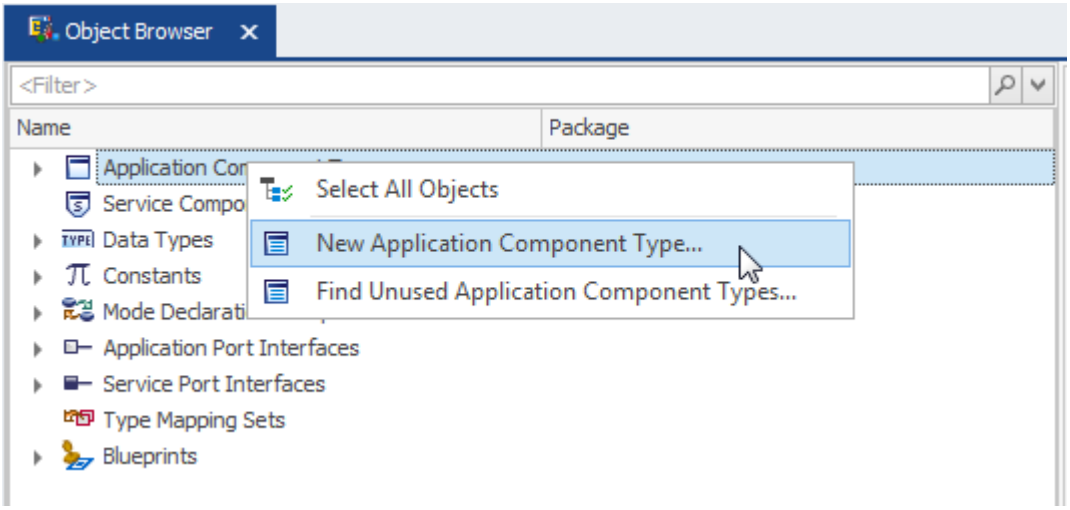


Note

The **Files** tab is not editable, if you want to add or delete files you have to move to the **File Browser**.

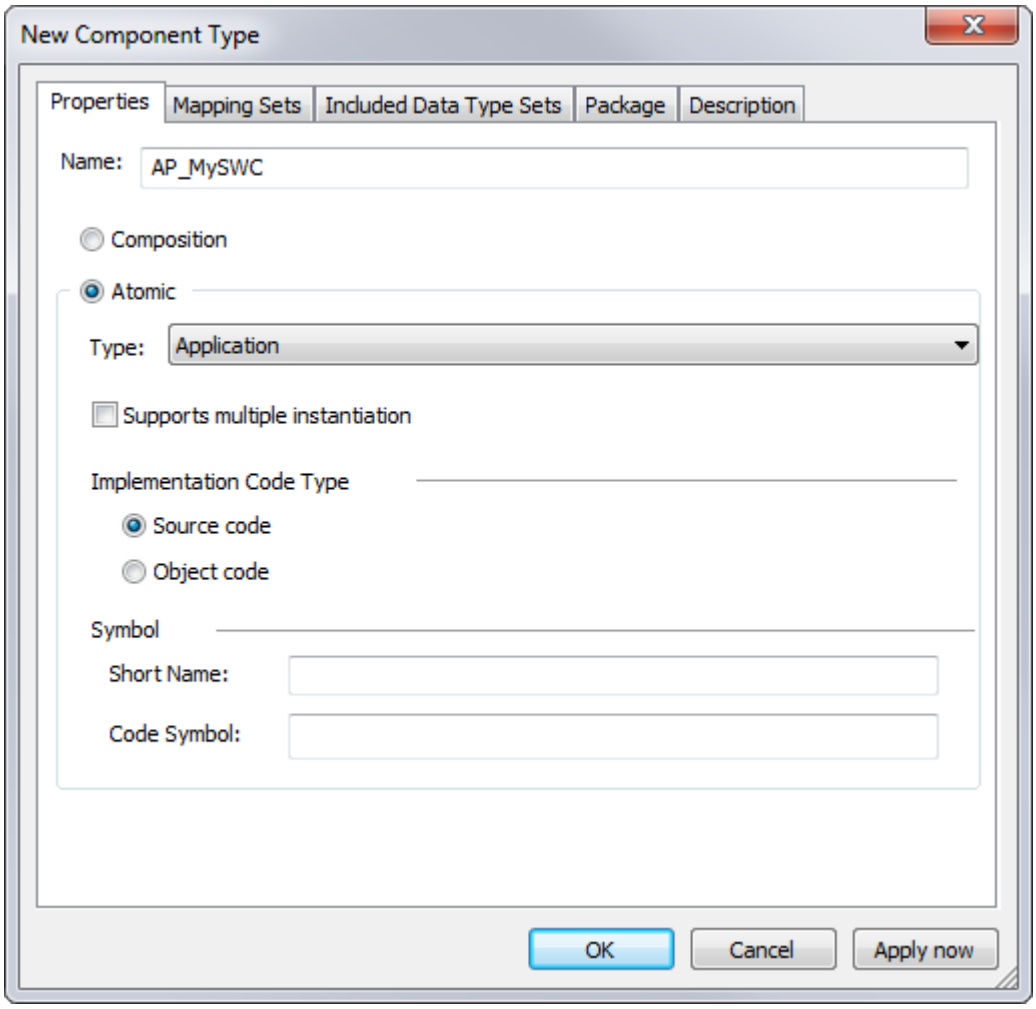
6.3.2 New Application Components

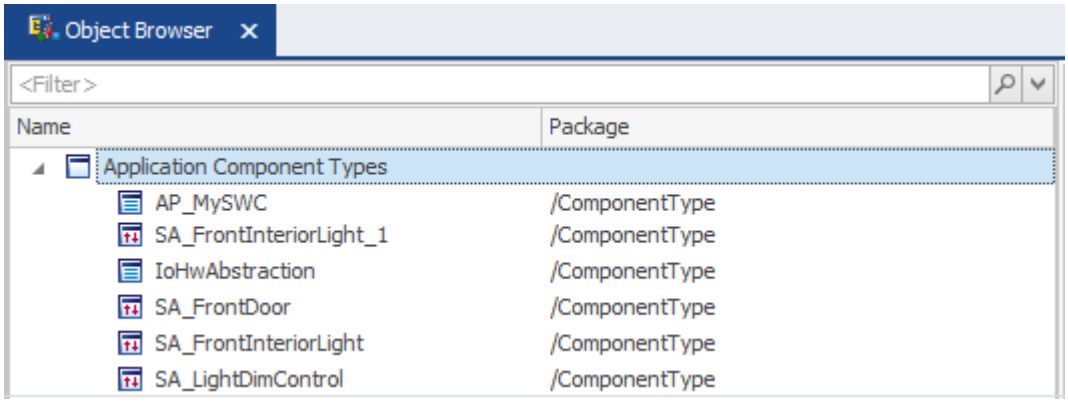
Right-click on Application Component Types in the Object Browser and select **New Application Component Type...**.



It is good to give a speaking name to see what kind of component it is. e.g. AP – Application, SA – Sensor / Actuator

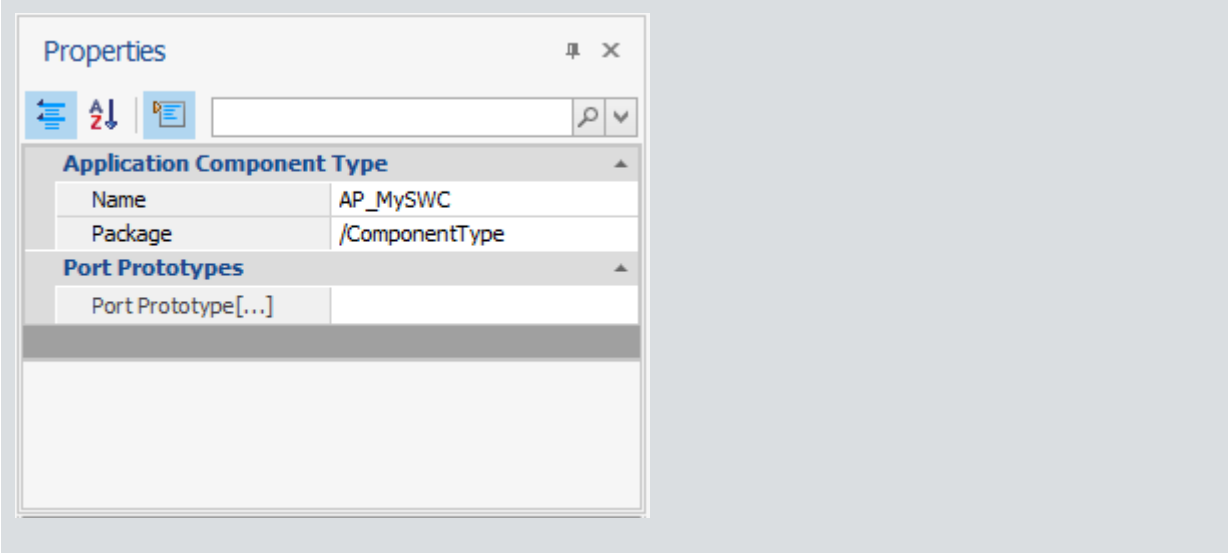
Enter a Name for the Application Component Type. Select Composition if the SWC should contain other SWC or select Atomic. Select its Type and confirm with **[OK]**. All application software component types are displayed in the Object Browser.





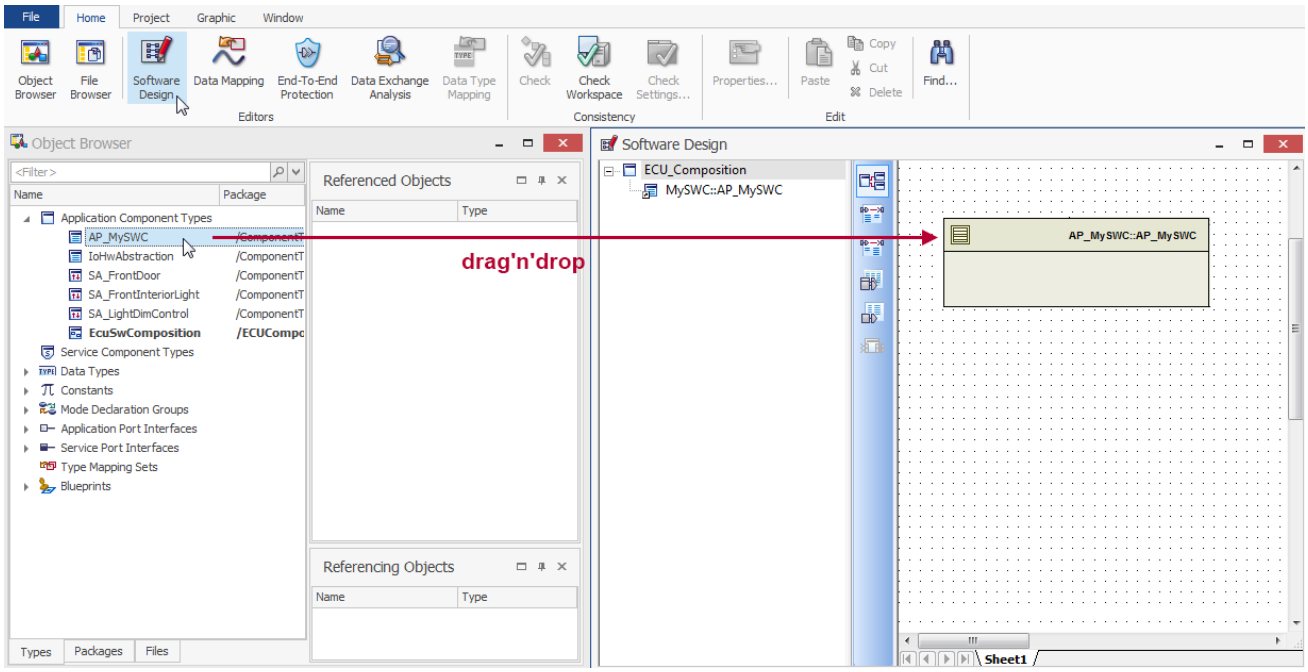
Note

To get fast information about any element in the Object Browser just select one and see its Properties, Port Prototypes (if available) and Description in the Properties view (right side).



Component types become component prototypes by being used.

Double-click the SWC in the Object Browser or click **Software Design** to open the Software Design view for your ECU project (e.g. MyECU).



Note

In the graphical representation of an application component prototype you can change its size using the little squares shown at the angles and in the middle of the sides.

An application component type becomes an application component prototype when it is used. It also needs a name. Using drag’n’drop, both names are the same. Open the properties window from the context menu for a component prototype to change the prototype name.

Note

Drag'n'drop is not possible within **Tabbed Layout**.

Define and use as many application software components as you need for your ECU.

Note

The notation of the software component starts with the software component prototype followed by the software component type.

6.3.3 Understand Types, Prototypes and Interfaces

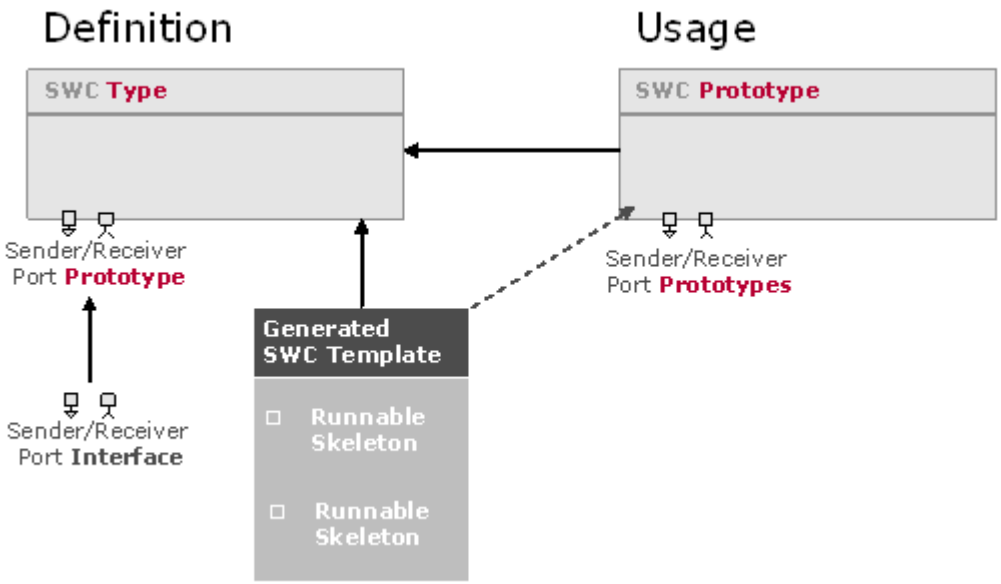
This illustration will help you to understand working with the DaVinci Developer. You have to deal with

- Prototype
- Type
- Interface

When is a software component a software component type, when a software component prototype? When is a port a port interface, when a port prototype?

In the Object Browser, the software components are types, the ports are interfaces. As soon as you use them, they become prototypes.

- Port Interface used by a component type ➔ Port Prototype
- Component Type in Object Browser used in software design view ➔Component Prototype



Runnables are always connected to the component type.

6.4 Ports, Port Init Values and Data Elements

To communicate and to exchange information the components need so-called ports (S/R ports).

For communication between software component ports have to be defined.

There are different kinds of application/service ports,

- **Sender Ports**  to provide information
- **Receiver Ports**  to receive information
- **Sender/Receiver Ports**  to provide and receive information within one port
- **Server Ports**  to provide services (operations)
- **Client Ports**  to use services (operations)

The following ports are listed here for completeness.

- Calibration Ports to hand over calibration parameters
- Mode Ports to e.g. trigger or not trigger runnables within certain modes

Before you can use application ports you have to define application port interfaces. To completely define the port interfaces you have to define data types first, if you don't want to use the predefined ones.

Predefined data types in Object Browser

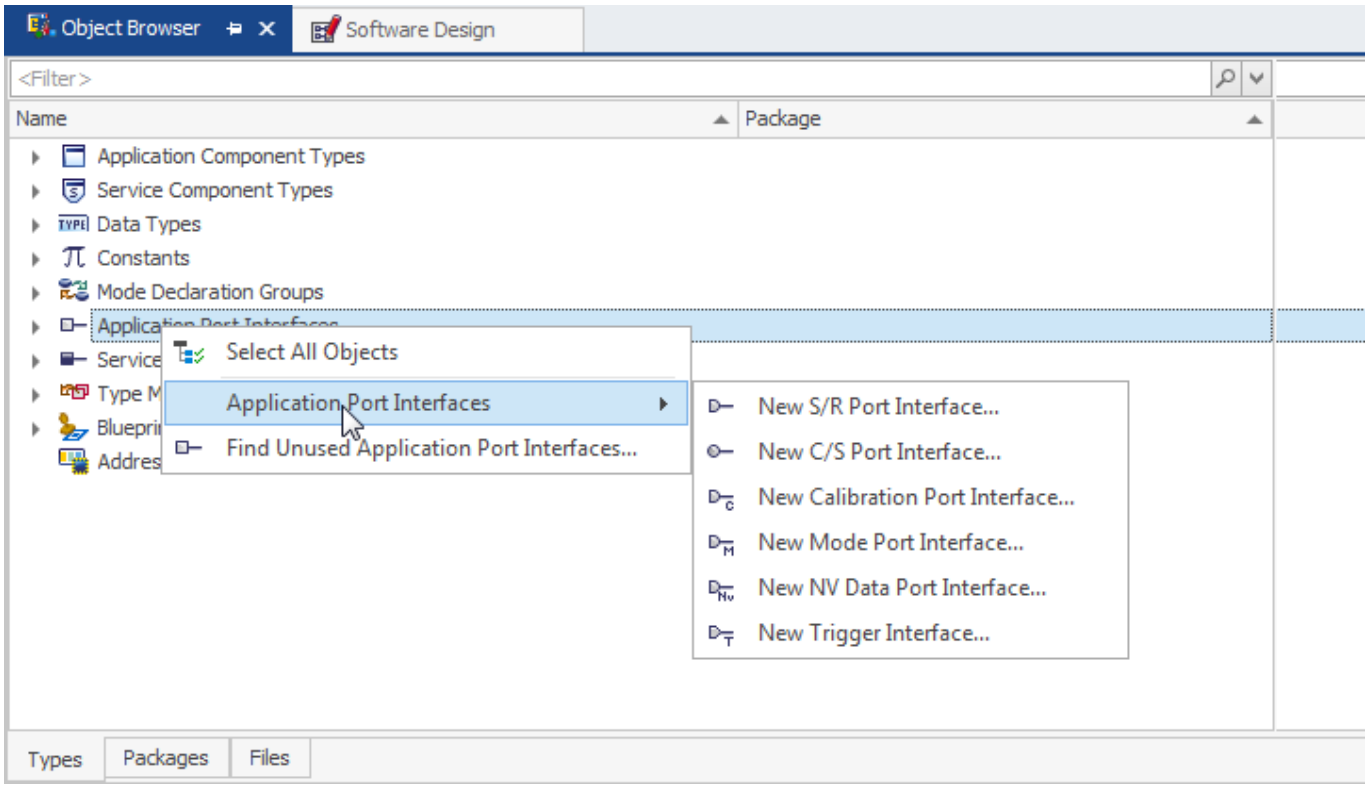
Just right-click and select from the list.

Caution!

You can handle application port interfaces like application component types. Define the application port interfaces in the Object Browser and then use them as application port prototypes for each application component. The same applies for data types and data elements.

Create Port Interface

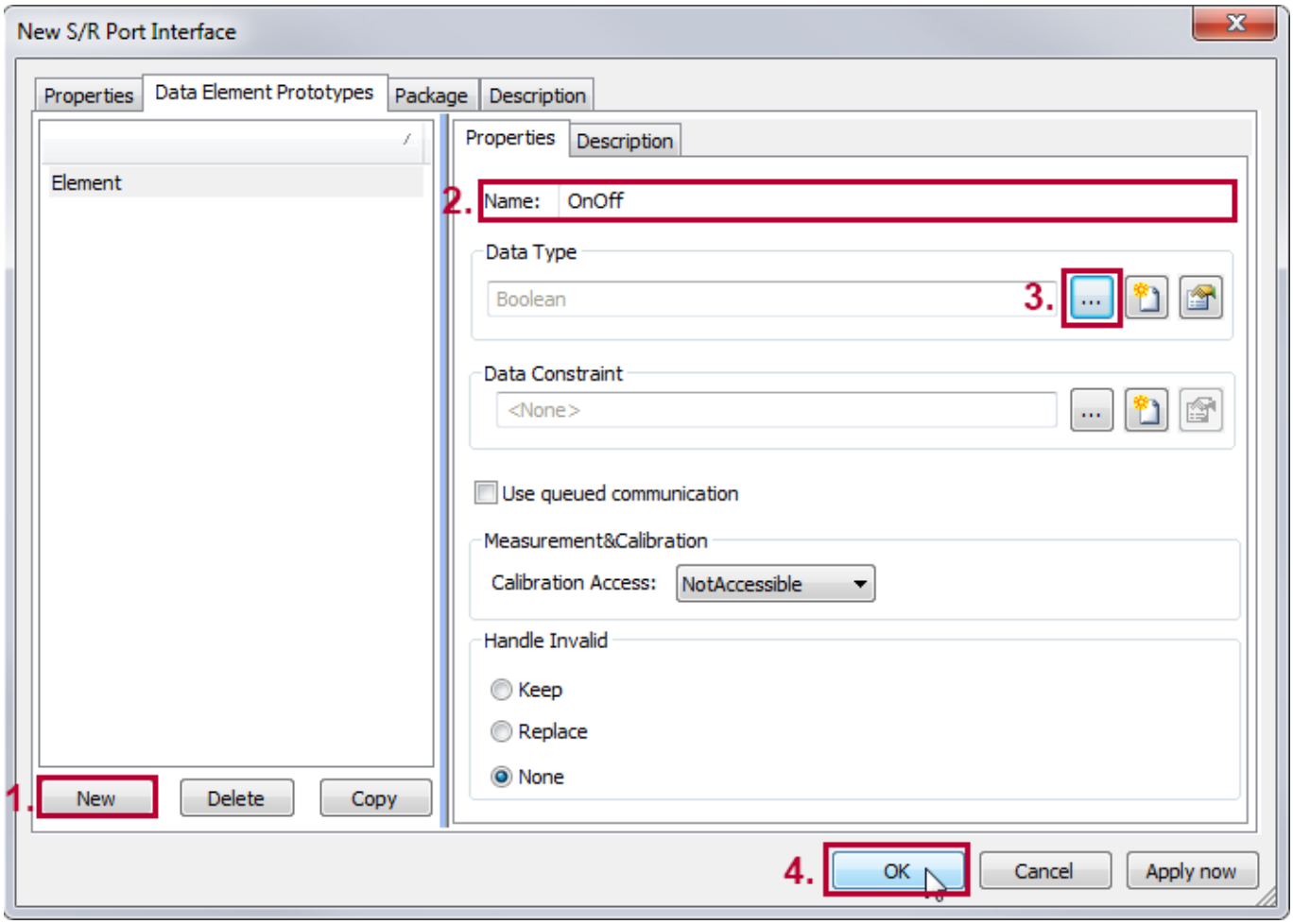
To define a new application port interface, right-click the Application Port Interface in the Object Browser and select **New S/R Port Interface...**.



Then the window for the S/R Port Interface opens. Enter a name, select the tab **Data Element Prototypes** and define the content of the port. In this case it is just one data element called OnOff with the Data Type Boolean, use the button [...] to select Data Type.

Note

Data elements are the contents of ports.



Note

A port interface can carry many data elements of different data types.

Provide software components with ports

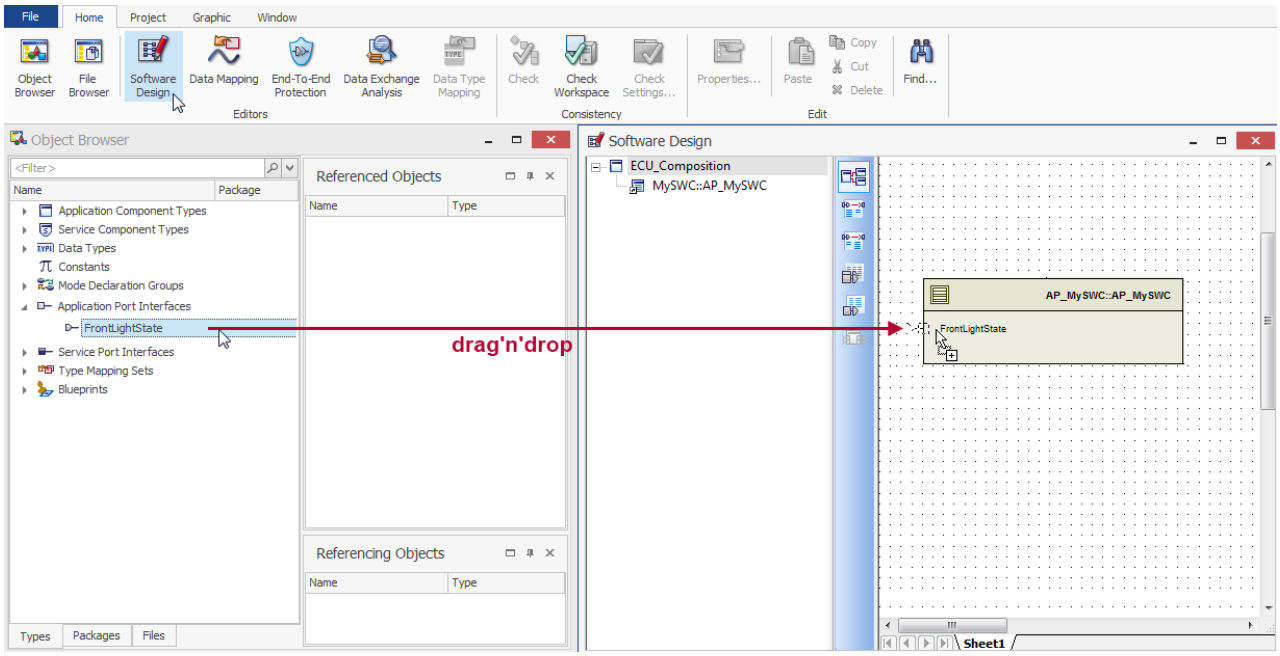
> You have created your application software components.

> You have created necessary application ports interfaces and their content (data element prototypes).

Now you have to define which application component needs which application ports.

Add ports to the application components via drag'n'drop

Select an application port interface from the Object Browser and place it onto the application component via drag'n'drop. This transfers a port interface into a port prototype.



Note

Press <Ctrl> while drag'n'drop to change between sender or receiver port.

Interfaces

The ports are added to the components as graphical element together with the name of the port.

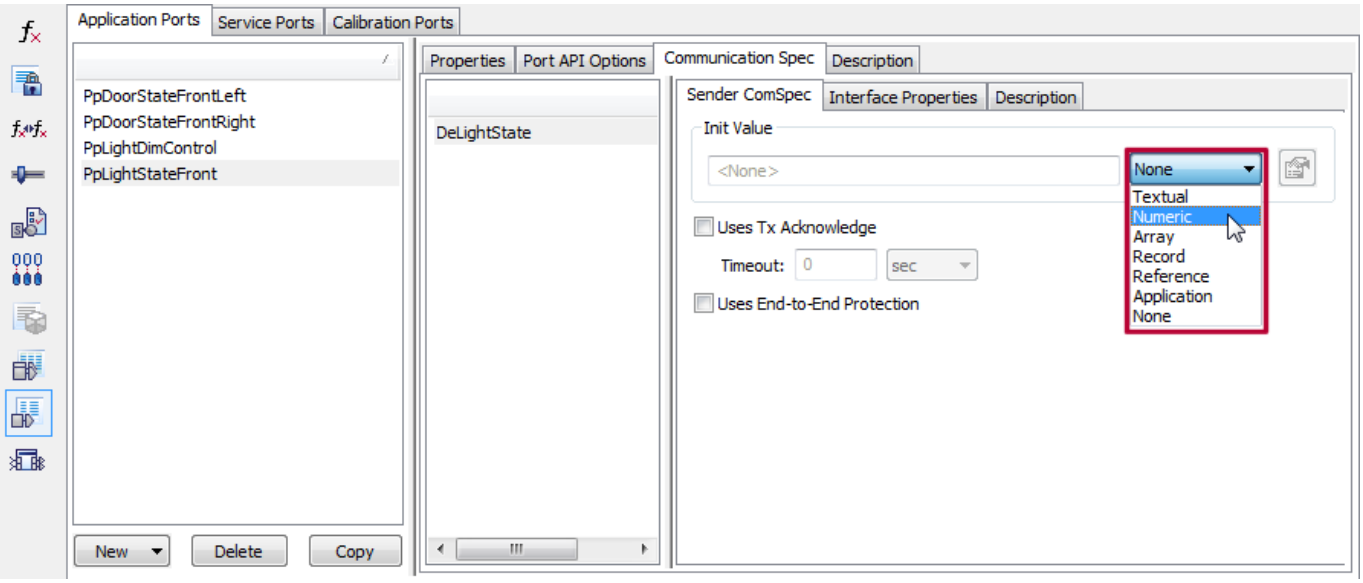
- **Sender port**
- **Receiver port**

Note

Ports need Init Values.

Port Init Values

You have to assign an initial value to every used application port.



6.5 Configure Service Ports within your Application Components

Your Service Components will be loaded automatically to the DaVinci Developer workspace .

Note

The DaVinci Configurator stores the service components within the folder **<Project Folder->\Config\ServiceComponents**.

The content of the service components are not part of the DaVinci Developer workspace.

Service Port Interface name and content depend on the BSW configuration and version of the service component within the DaVinci Configurator, so it is not stable.

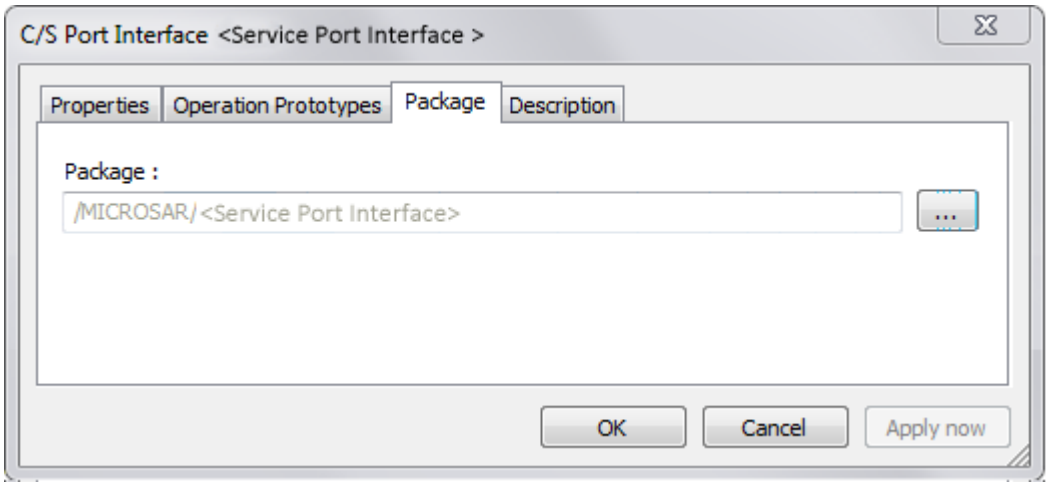
If you want to connect one of your Application Component to a Service Component, you have to define a Service Port Prototype based on a Service Port Interface.

To avoid implicit modification of the Application Components, they should not use these Service Port Interfaces directly. Instead, they should use an own copy of the Service Port Interfaces.

The Service Port Interface definition from the loaded Service Component is always in read-only status. These Service Port Interfaces are marked with a **read-only** icon within the Object Browser.

You have currently not defined your Service Port Interfaces?

Then copy and paste the definition from the Service Component. Assign the definition to an own package, so it is possible to use the same name as before. Therefore open the Service Port Interface, select **Package** tab and choose a package via **[...]**.



Assign your copied Service Port Interface to your Application Component. Therefore open your Application Component Prototype, select **Port Prototype List** , open **Service Ports** tab and add your Service Port via **[New]** | **From Port Interface**. Now the Service Port used by your Application Component is independent from the Service Component description.

Note

If incompatible changes apply these will be reported by RTE checks.

6.6 Define your Runnables

Now configure runnables that will carry your code.

There are four important topics for a runnable:

- **SYMBOL** and **NAME** – what the runnable skeleton is called in the template file
- **TRIGGER** – when is the runnable executed
- **PORT ACCESS** – what data can the runnable access
- **MAPPING** – in which task context does the runnables work?

Caution!

Runnables can only be defined for atomic application components types.

Note

The runnable skeletons are generated by the DaVinci Configurator Pro/the RTE and can then be accomplished with your code. Find more details follow in the next chapters.

Check also later hints to template generation.

1. Open a software component via the Object Browser or the Software Design view.
2. Click on the runnable icon **[fx]**.
3. Click **[New]** and select Runnable.
4. Enter Name and Symbol for the new runnable on the Properties tab.

If you leave the Symbol field empty, the runnables (functions) will be named according to the entry in the **Name** field.

Minimum Start Interval
Define the minimum time interval between the successive triggering of this runnable.

Can Be Invoked Concurrently

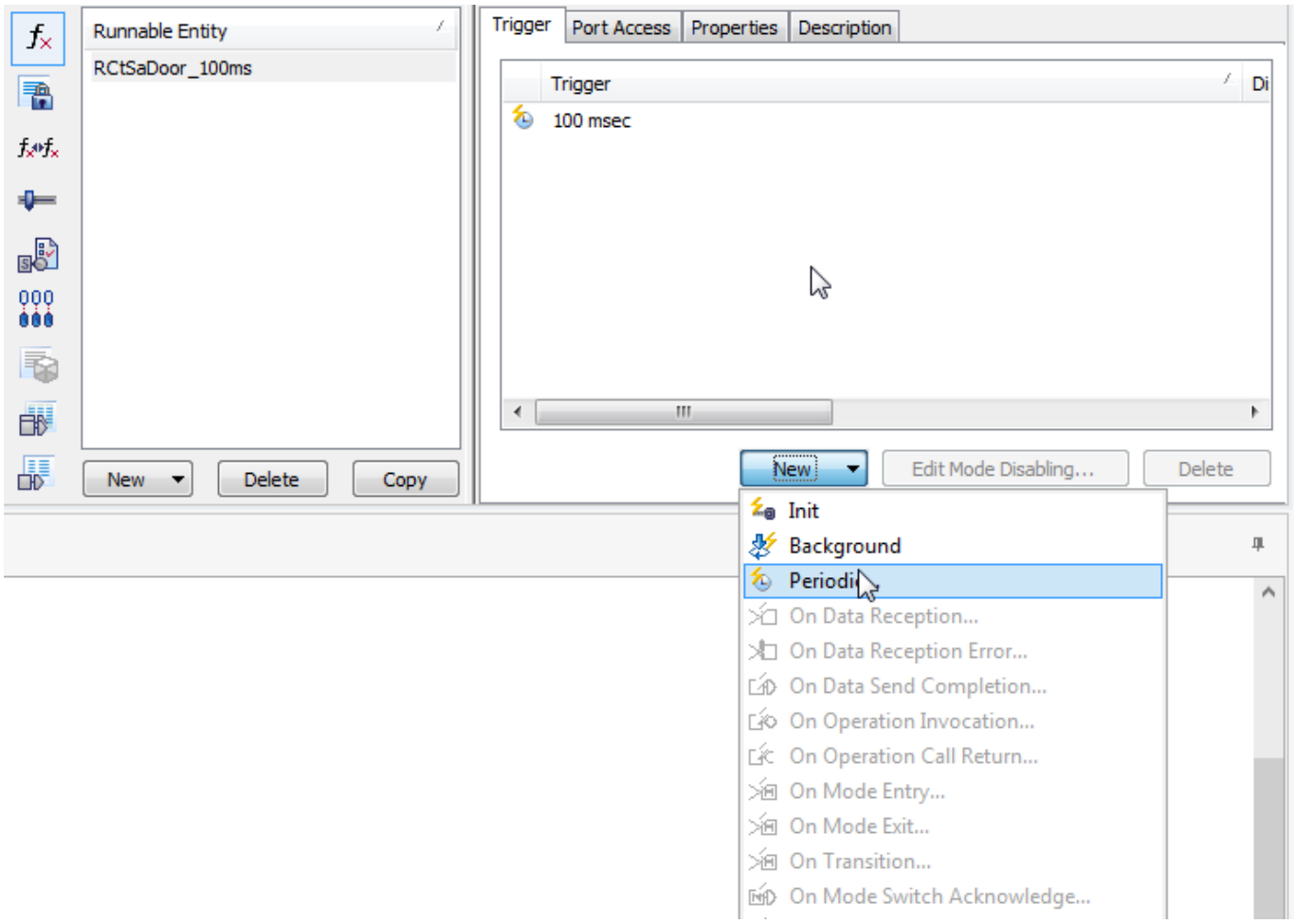
A runnable can be marked as **Can be invoked concurrently** if it can be executed while it is already running. In other words it can be safely executed concurrently (re-entrant). There are a few criteria for your implementation code:

- No static (or global) non-constant data
- No return of address to static (or global) non-constant data
- Only work on data provided by caller
- No modification of own code at runtime
- No call of non-re-entrant runnables

6.7 Triggers for the Runnables

The trigger decides when a runnable is executed.

Select the tab **Trigger**. On this tab you select when your runnable should be executed.



Note

A runnable can be triggered periodically or via an event.

Periodical: Select the checkbox and enter the cycle time.

On Data Reception: As soon as data is received at the appropriate port the runnable is activated. (Indication)

The trigger **On Operation Invocation** belongs to service ports and will be explained later.

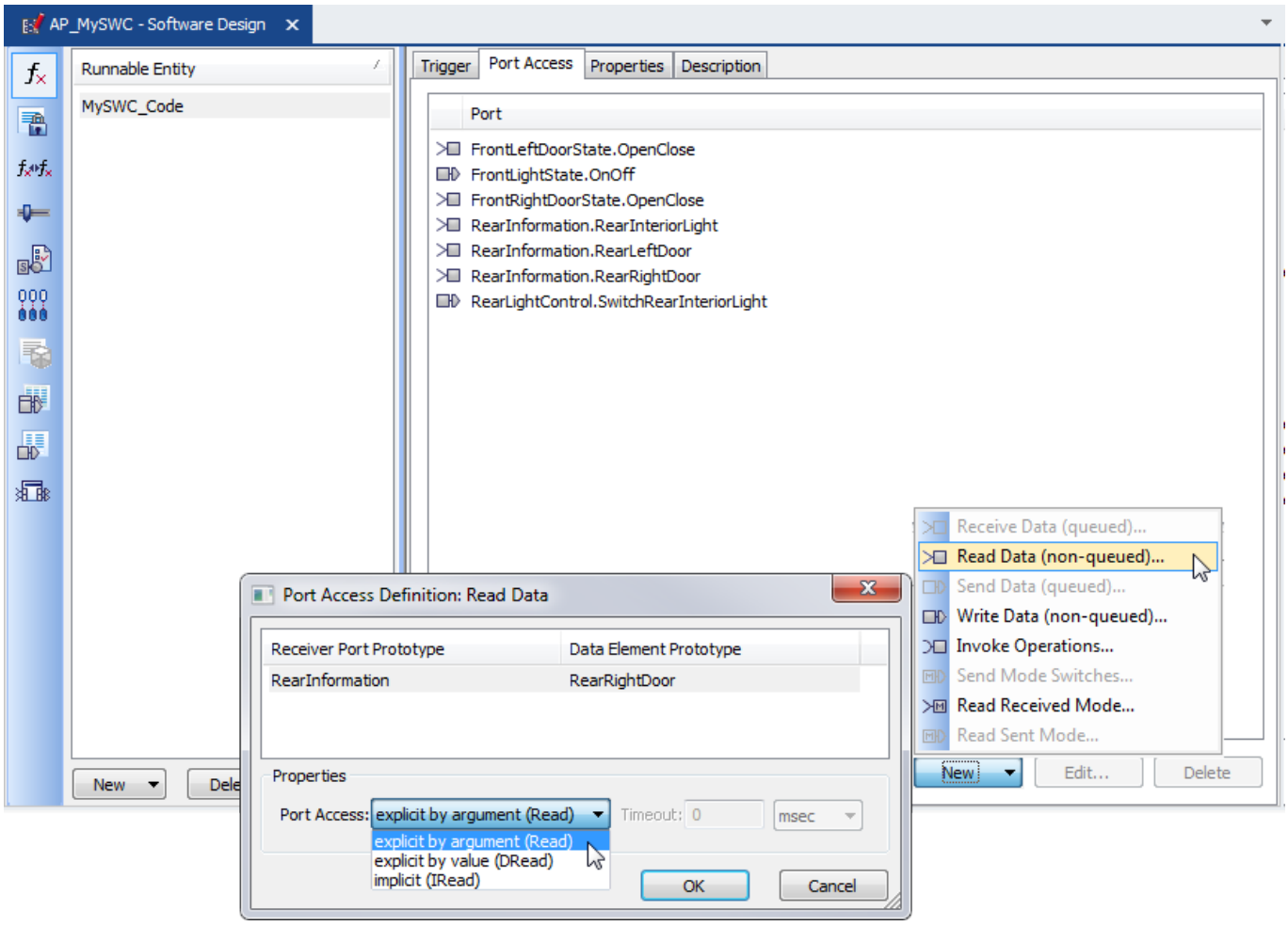
The runnable of the demo application only needs to be called when the state of the doors is changed. This can be realized via a cyclic polling or directly by a trigger from the doors. In the demo, **Periodical**... is chosen.

Note

When experimenting with the demo, replace the cyclic trigger with two triggers On Data Reception..., one for the left and one for the right front door.

6.8 Port Access of the Runnables

Define which port information each runnable should be able to read or write. Select the tab **Port Access**. You only get displayed accessible ports.



You can define access to

- Read Data… and
- Write Data…
- and later when using client server port also to operations (Invoke Operations).

Click e.g. **Read Data**… and the Port **Access Definition: Read Data** window will open.

6.9 Multi-Edit

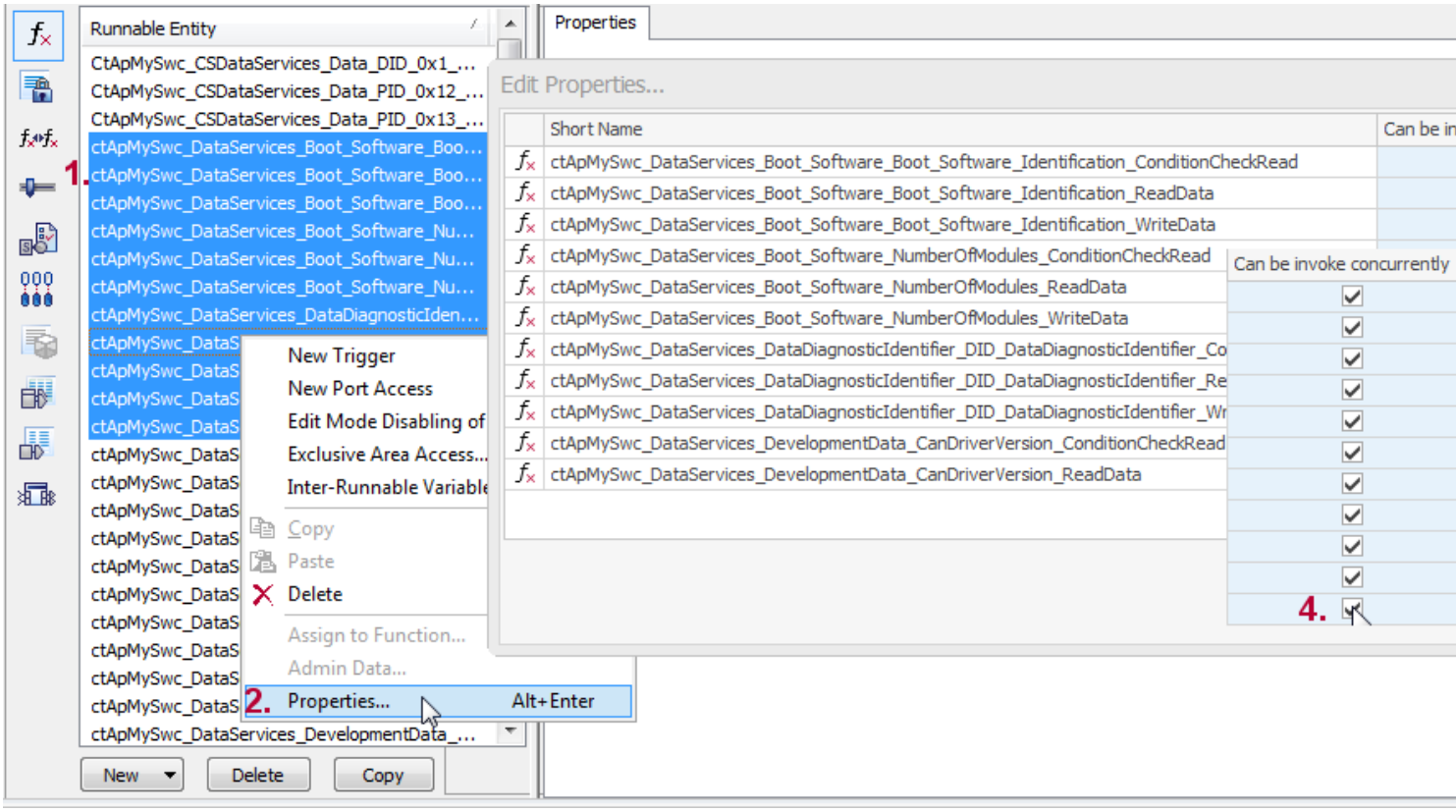
A very know task is to set e.g. the checkbox **Can be invoked concurrently** for many Runnable Entities. This can now be done very comfortable.

1. Select your Runnable Entities and

2. Open the **Edit Properties...** window via the context menu.

3. Mark the cells you want to activate the checkboxes

4. Activate one checkbox and all will be activated. Same works to deactivate the checkboxes.



Cross Reference

ReleaseNote_MICROSAR4_Vector_*.pdf - **FEAT-3484**

Summary

Summary of the settings for your runnable MySWC_Code:

- The runnable is called MySWC_Code
- Its functional representation is called: MySWC_Code.
- It is triggered periodically
- The runnable has access to the state of the left and right door and to the data of the front interior light. It also has access to the LightDimControl.

6.10 Templates and Contract Phase Headers

The design of the SWC including ports, runnables and port access influences the SWC Templates and Contract Header Files you generate with DaVinci Configurator Pro.

In the header of the runnable's skeleton you will find a list with all available API functions for this runnable. If any access is missing, go back to the DaVinci Developer and check whether the Port Access is set correctly.

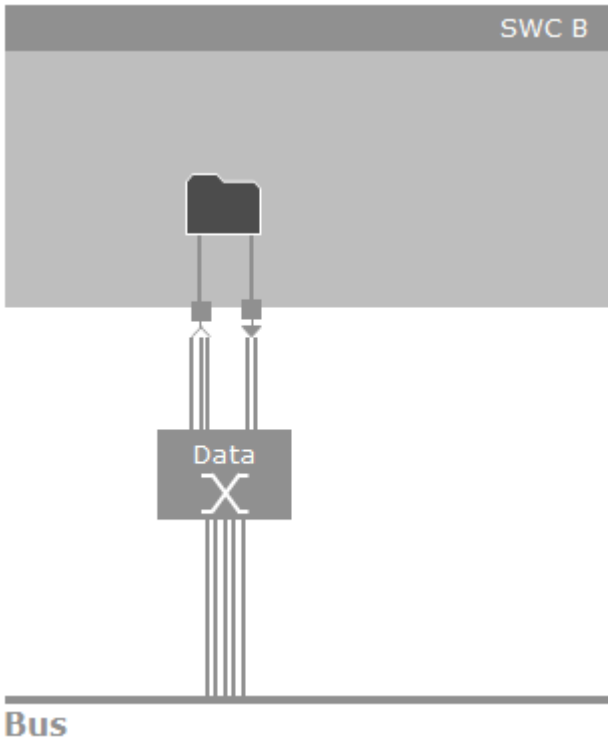
7 Mappings

Generally spoken Mapping stands for assignment of one object to another. There are different mappings in the context of AUTOSAR. What is meant here is:

- Data Mapping
- Task Mapping
- Service Mapping
- Memory Mapping

7.1 Data Mapping

Your ECU has to communicate with other ECUs, the external communication. Signals have to be sent and received via the connected bus system. The signals and the types of data transported via the signals is defined in the data base like DBC, LDF, FIBEX, and also in the SYSEX.



From the view of software components, communication information is exchanged via ports that carry so-called data elements. The definition of a port contains the assignment of data elements that can pass the port. Via the Data Mapping you define, which data element belongs to which signal. For short, you assign data element to bus signals. That’s called Data Mapping.

Note

Data Mapping can also be done in DaVinci Configurator Pro!

Data Mapping in DaVinci Developer has a higher priority than Data Mapping in the DaVinci Configurator Pro.

Data Mappings done via DaVinci Developer cannot be overwritten by the ones done via DaVinci Configurator Pro.

7.2 Task Mapping

Tasks are means of the operating system. You can define as many tasks as you like and need. You define the name of the tasks, its priority and its type like **Auto**, **Basic** or **Extended**. You can also define the task as non- or full preemptive.

With Task Mapping, you have to map all runnables to tasks, expect **On operation invocation** triggered runnables. Those are normally called only from one task and have no problem with reentrance.

A runnable has to be mapped to a task if it is not re-entrant but could be called re-entrantly during system operation.

What happens if you do not map a runnable that should be mapped?

The Vector AUTOSAR Solution provides an intelligent RTE generator.The RTE generator checks necessary conditions and will warn you if there are unmapped runnables that have to be mapped.

What happens if you map a runnable that does not have to be mapped?

Nothing will happen. It will still work. But the code efficiency and RAM consumption could be less good.

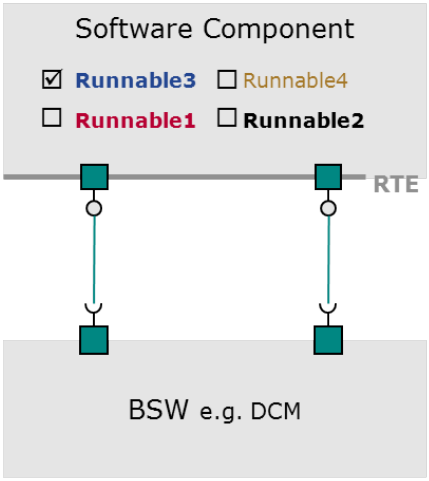
Is there a disadvantage of the RTE generator automatism?

It could be less comfortable to figure out the call context of the runnables and the stack consumption could increase.

7.2.1 Information about Interaction between Runnable, Re-entrance and Task Mapping

Your goal is a configuration of the system that results in a highly runtime-optimized and memory consumption-optimized code. Here is a little information to estimate what the RTE generator does.

Software Design



Task Mapping

Task A	Task B
Runnable1	Runnable2
Runnable5	Dcm_MainFunction
Runnable9	Runnable11
	Dem_MainFunction

In context of Task A

RTE

tmp = **A**
SetEvent **R1**
...

TASK
{
tmp = A
waitEvent
if Event = **R1**
Runnable1(**A**)
...
}

In context of Task B

Rte_Call **Runnable3**
Rte_Call **Runnable4**

This illustration shows a software component with four runnables. Only runnable 3 is set to **Can be invoked concurrently**. On the right-hand side you see which runnable is mapped to which task and the result of this mapping for the generated code. Here is the details.

Can be invoked concurrently is set and the runnable is not mapped to a task

Example: Runnable 3

Runnable 3 is called directly in the context of the calling BSW (DCM in this example – DCM_MainFunction, TASK B).

Caution

Make sure that your runnable is really re-entrant (see section **Can Be Invoked Concurrently**). If not, errors can occur that are difficult to debug.

Can be invoked concurrently is NOT set and the runnable is not mapped to a task

Example Runnable 4

If the RTE generator finds out that the runnable is not called multiple times in parallel, Runnable 4 is called directly in the context of the caller BSW (DCM). Otherwise an error message will be shown.

Can be invoked concurrently is NOT set and the runnable is mapped to a task different from the task where the main function of the caller (e.g. Dcm_MainFunction) is mapped.

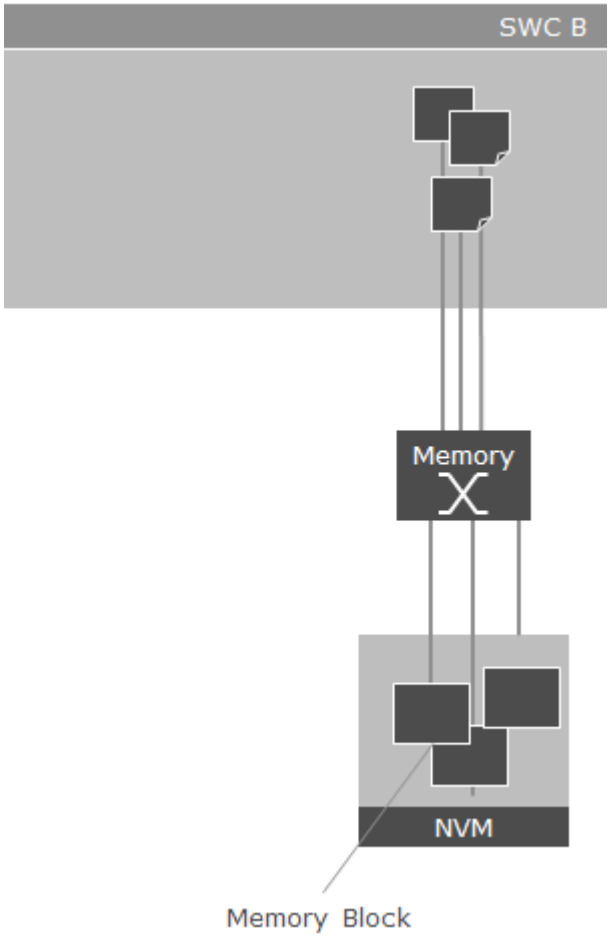
Example: Runnable 1

When the caller BSW now wants to activate the runnable via the RTE call, an event is set and the temporary variables of the runnables are stored to RAM (context of calling BSW, e.g. Task B). When the point in time has come, the WaitEvent in Task A is triggered by the event and starts the Runnable1(A) and hands over the parameters previously stored to RAM.

As you see, runtime is longer than using direct call and the RAM consumption is higher. E.g. for diagnostics: DCM runnables always have array types that are completely stored to RAM. For e.g. 100 DIDs à 4 bytes you need 400 bytes!

7.3 Memory Mapping

Within ECUs there is data that has to be persistent during power-off. Therefore this data cannot be stored simply to RAM, it must be stored to EEPROM or Flash, i.e. to non-volatile memory. As an example for this data, think of the DEM information.



There are two possible ways how this non-volatile data will be used. Either your application always accesses a RAM copy of the NV data where the data is copied at start-up. Or data can be read/written directly from/to NV memory on request.

The Memory Mapping assigns your defined PIM, per instance memory to the memory blocks of the NVM.

A per-instance memory object is mappable if it is referenced by a service need object. The definition of per-instance memory objects or service needs is part of the component type definition and is not yet editable by DaVinci Configurator Pro.

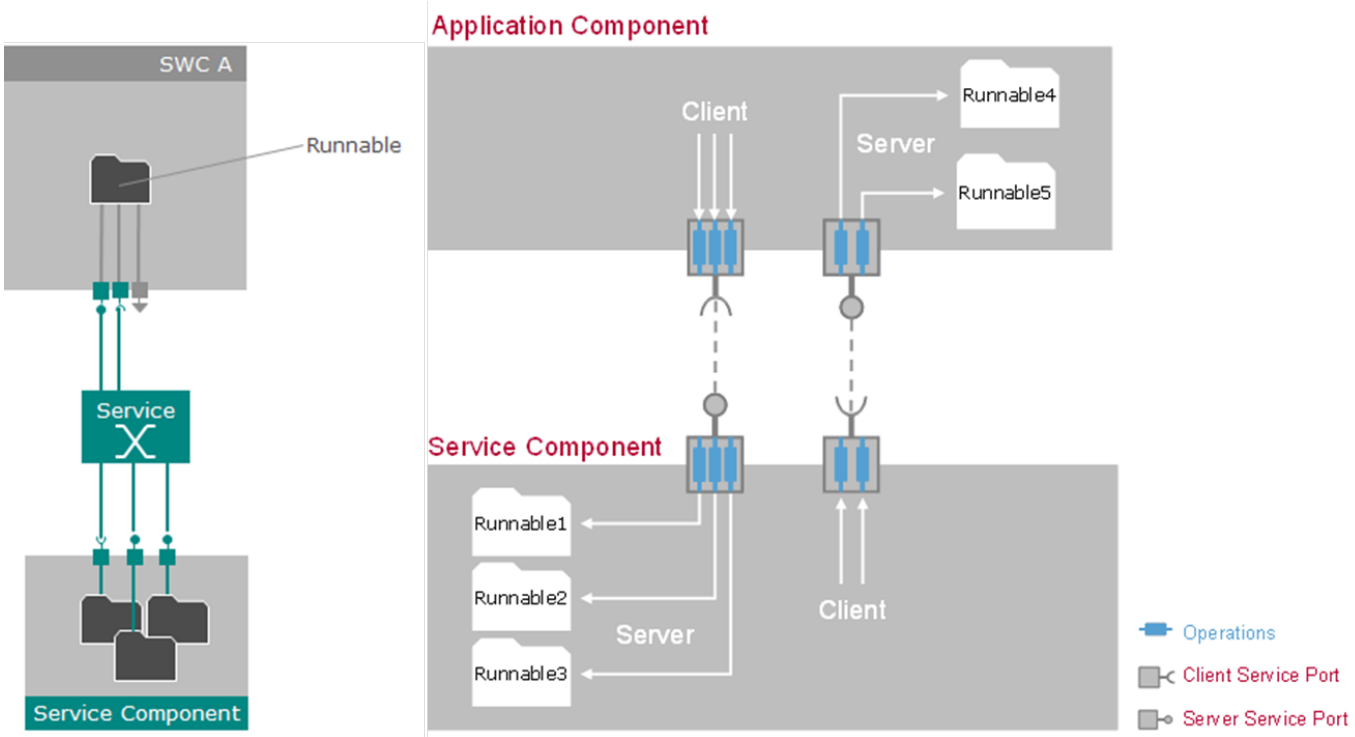
7.4 Service Mapping

The base for the service mapping is the theory about clients and servers. The client uses a service of the server. The server itself provides the operation to the client. The client server communication between your application software component and a service component is done via service ports – client ports and server ports. A server port provides services (one or more operations) and a client port uses these services.

A service component can have server ports and client ports.

- **Service port of the service component is a server port**
  The service is provided by the service component and your application software component just uses the operations (services, functions)
- **Service port of the service component is a client port**
  Your application software component must be server and has to provide the operation (service, function). In this case, you have to program code, i.e. a runnable. This runnable must be created by you and it is triggered by the request of the service.

Besides the sender and receiver ports, there are also client and server ports. Sender and receiver ports carry data elements. Via client ports you can access so-called operations (or functions) of the servers.



Within one service port there could be n operations. Every single operation has to be assigned to a runnable.

The servers provide the runnables that contain the code. Here Runnable1, Runnable 2, Runnable 3 of the service component and Runnable 4 and Runnable 5 of application component.

When performing the service mapping, it is almost the same as the data mapping – but now you deal with services instead. To be able to access the services of a service component, you have to add this service component to your software component.

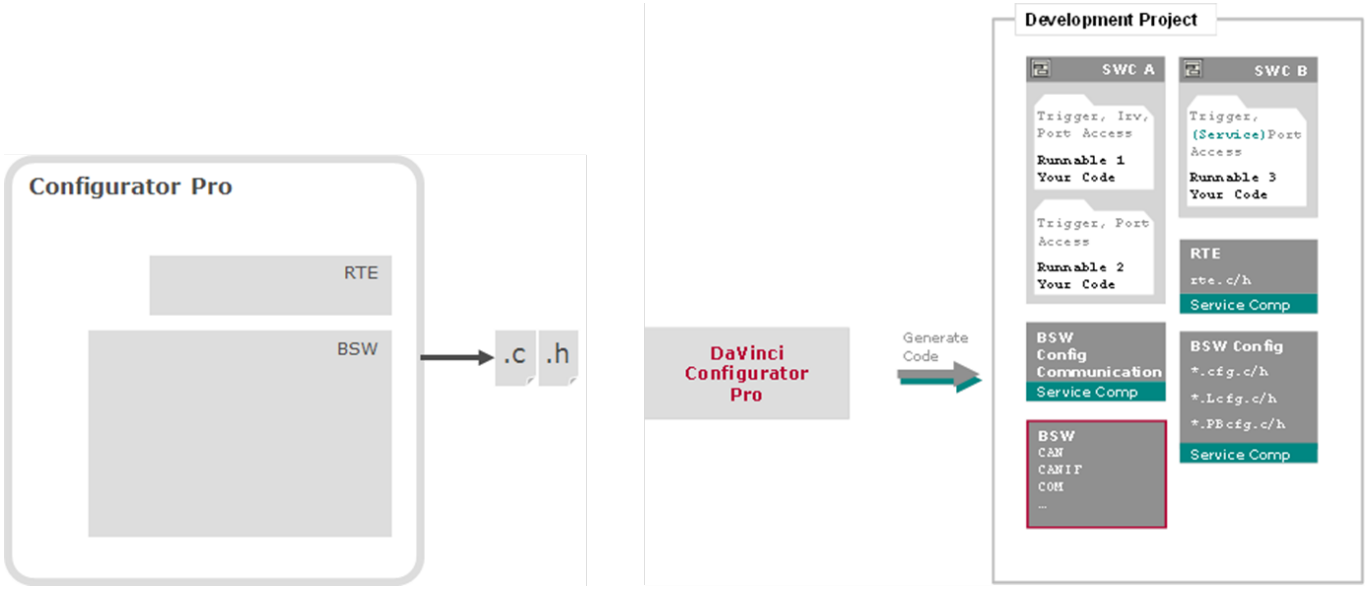
Then you can use all services that are provided by the service component. In this case your application is the client and the service component is the server.

But using a service component not only brings benefit – it also costs. Almost every service component does not only provide services – there can also be client ports on the service component side. And if you add this service component to your software component, you have to provide all the services where the software component provides a client port interface.

8 Generation

With the generation step DaVinci Configurator Pro generates the code for your project and also concerns your settings in STEP2 Define Project Settings on page 29. DaVinci Configurator Pro clings to your settings and works off the "to be generated" list.

The DaVinci Configurator Pro provides an automation interface for remote-controlled validation and code generation.



8.1 MICROSAR Rte Gen

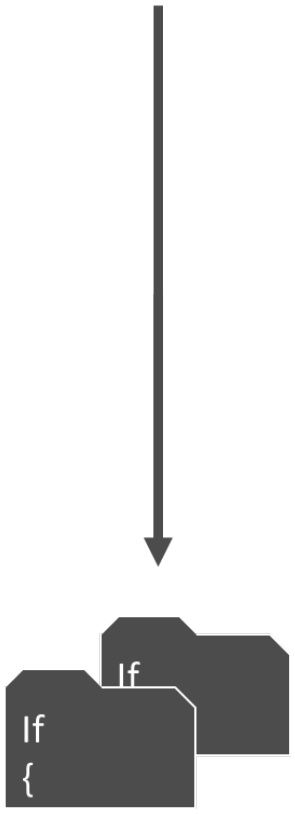
The RTE is generated by the **MICROSAR Rte Gen**. Its generation depends on many settings in the configuration tools and how runnables are mapped to tasks. The files start with **Rte_** or **SchM_**.

9 Runnable Code

Until now you don't have written one line of C code. And that's typical for the AUTOSAR concept of software development. There is much more configuring work with tools than programming or implementing as you are normally used to from previous ECU software projects with e.g. CANbedded.

Example

This is an example showing the runnable called MySWC_Code with the function name (Symbol) MySWC_Code




```
/*****************************************************************************************************************
 *
 * Runnable Entity Name: SwitchFrontLight
 *
 *----------------------------------------------------------------------------------------------------------------
 *
 * Executed if at least one of the following trigger conditions occurred:
 *   - triggered on DataReceivedEvent for DataElementPrototype <OnOff> of PortPrototype <FrontLightState>
 *
 *****************************************************************************************************************
 *
 * Input Interfaces:
 * =================
 *   Explicit S/R API:
 *   -----------------
 *   Std_ReturnType Rte_Read_FrontLightState_OnOff(Boolean *data)
 *
 * Client/Server Interfaces:
 * =========================
 *   Server Invocation:
 *   ------------------
 *   Std_ReturnType Rte_Call_FrontInteriorLight_DEFECT_OP_GET(IoHwAb_BoolType *signal)
 *     Synchronous Server Invocation. Timeout: None
 *     Returned Application Errors: RTE_E_env_FrontInteriorLight_DEFECT_E_NOT_OK
 *
 * Service Calls:
 * ==============
 *   Service Invocation:
 *   -------------------
 *   Std_ReturnType Rte_Call_Event_DTC_0x000002_SetEventStatus(Dem_EventStatusType EventStatus)
 *     Synchronous Service Invocation. Timeout: None
 *     Returned Application Errors: RTE_E_DiagnosticMonitor_DEM_E_QUEUE_OVERFLOW, RTE_E_DiagnosticMonitor_E_NOT_OK
 *
 *****************************************************************************************************************/

FUNC(void, RTE_SA_FRONTINTERIORLIGHT_APPL_CODE) SwitchFrontLight(void)
{
/*****************************************************************************************************************
 * DO NOT CHANGE THIS COMMENT!           << Start of runnable implementation >>            DO NOT CHANGE THIS COMMENT!
 * Symbol: SwitchFrontLight
 *****************************************************************************************************************/

IoHwAb_BoolType lightDefect;


Rte_Call_FrontInteriorLight_DEFECT_OP_GET(&lightDefect);

if (lightDefect)
{
  /*Light is detected to be defect. So any light information from MySWC_Code is turned to light off.*/
  Rte_Call_Event_DTC_0x000002_SetEventStatus(DEM_EVENT_STATUS_FAILED);  /*set event*/
}
else
{
  /*Light is working well. No special treatment necessary*/
  Rte_Call_Event_DTC_0x000002_SetEventStatus(DEM_EVENT_STATUS_PASSED);  /*set event*/
}
```

This is the header of a runnable showing all necessary information about the runnable like:

- When it is triggered
- Input and output interfaces
- Service interfaces

Note

If some access is missing, go back to the DaVinci Developer, add the necessary port access for your runnable, go back to DaVinci Configurator Pro, synchronize the system description and generate the component templates again.

Then the missing interface should be there and can be used.

10 Compile And Link

10.1 Using your "real" hardware

This step is highly dependent on your compiler / linker / project settings. Compile and link everything together and create a file that can be downloaded to your hardware.

III Additional Information

This section includes additional information dealing with special topics like e.g. update your input file or support request package.

1 Update Input Files

Open Input Files editor via **Project** | **Input Files** or **Input Files** icon at DaVinci Configurator Pro toolbar.

Note

Any change of the input files requires an update of the configuration. Update configuration updates all input files, whether they are changed or not.

1.1 System Description Files

If necessary, replace your File at Input Files | System Description File.

Select the Input File, click **Replace** button and choose the new **Input File**.

Note

Any change of the input files requires an update of the configuration. Update the configuration via **Update Configuration** icon .

1.2 Diagnostic Data Files

If necessary, update your Diagnostic Description file and select ECU and Variant.

In case of **ODX 2.0.1** as Diagnostic Description file, it is necessary to update the state description. Therefore click **[Synchronize State Description...]**.

Note

Any change of the input files requires an update of the configuration. Update the configuration via **Update Configuration** icon .

2 Update Project Settings

Open Project Settings Editor via **Project** | **Settings** or **Project Setting** icon  at DaVinci Configurator Pro toolbar. Select **Project Settings** to open the view.

3 Support Request Via DaVinci Configurator Pro

If you request support for Vector products, you have to provide project and environment information. Therefore, use the **Support Request Package** function of the DaVinci Configurator Pro.

The **Support Request Package** compiles the necessary project and environment information in a ZIP-file. Send this file to the Vector Support via E-mail. Just follow the description below.

Note

The DaVinci Configurator Pro does not send any data automatically!

1. Open the Support Request Editor via **Help** | **Create Support Request Package...**

Note

The entry **Create Support Request Package**... is only enabled if a project is loaded.

2. Add your First Name, Last Name and E-Mail address.
3. Click **[Next >]**
4. Select project information which has to be added to the support request ZIP-file.

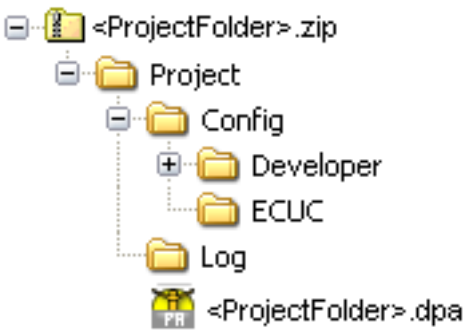
Note

PC Info, Tool Version Info and SIP Info are default and can not be deselected.

5. Click **[Next >]**
6. Enter the path where the created ZIP-file has to be stored.
7. Click **[Finish]**

3.1 Result

In addition to the **Zip-File** <Target Path>.zip , the Project Assistant creates the following folders and files.

Folder/File	Description
	The Project folder includes all defined project files
	The file includes all necessary project and system information (e.g. version of tools)
	The file includes all necessary project and system information (e.g. version of tools)

Send **<Target Path>.zip** via e-mail to the Vector Support Address embeddedSupport@de.vector.com.

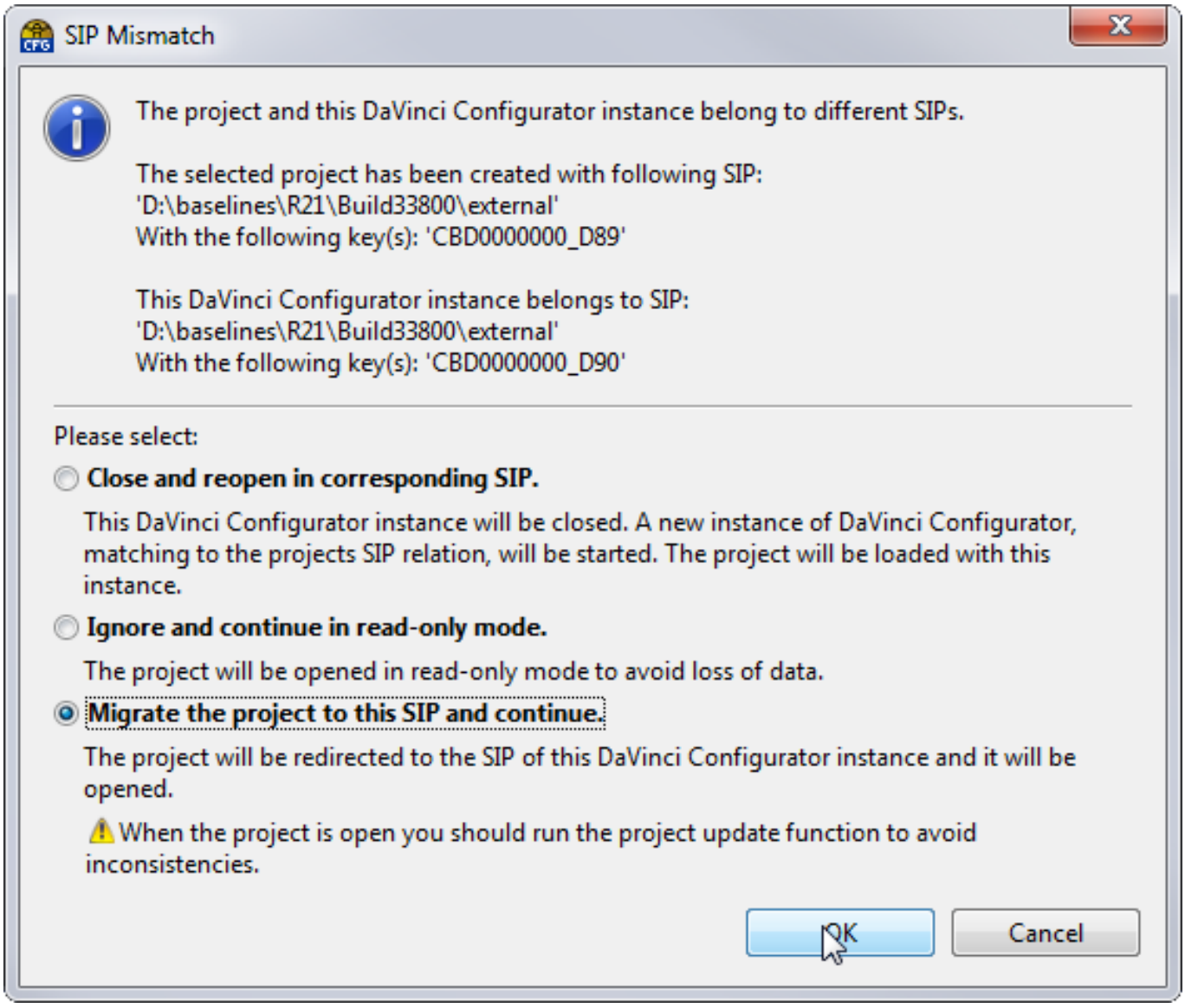
4 SIP Update And Project Migration

If you get a new delivery with Vector BSW Release x, it is necessary to update some parameter values of your configuration to the new release. There are some steps, that are necessary for most of the updates and there are special steps, that are only necessary for a certain update.

First there are described the necessary steps for any update followed by the special update steps dependent on the release version.

4.1 Release x-1 to Release x (necessary steps for any update)

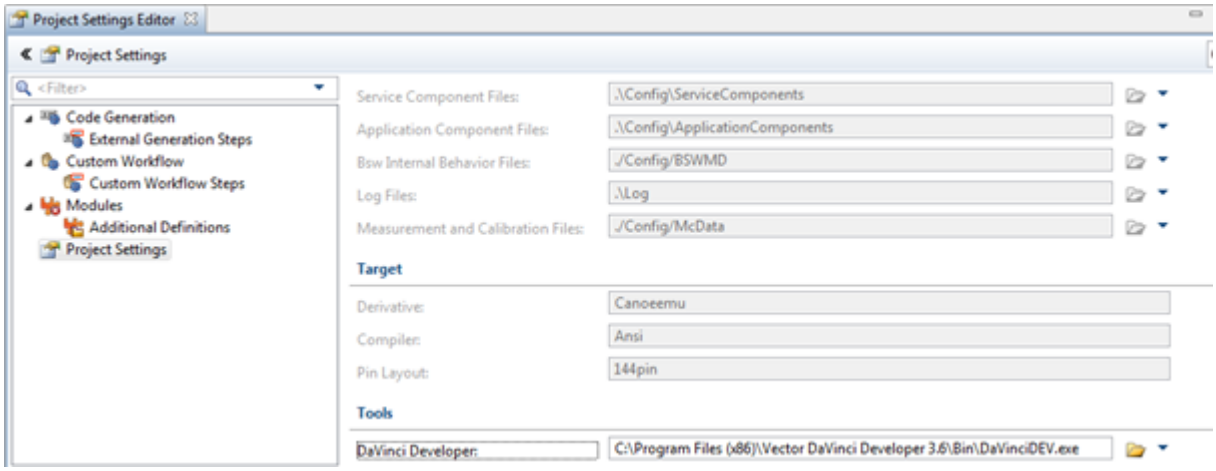
1. Install the DaVinci Developer version required for the new Release (check DaVinci Compatibility Matrix for compatibility)
2. Open your configuration with the DaVinci Configurator of the new SIP .\DaVinciConfigurator\Core\DaVinciCFG.exe and select the following option:



Note

The path within the *.DPA file will be updated to the new SIP.

3. Change the path to the new DaVinci Developer if it was installed to a different location than the former DaVinci Developer version.



Note

C:\Program Files (x86)\Vector DaVinci Developer **<version>**\Bin\DaVinciDEV.exe

4. Handle remaining errors and warning. The following might occur for any update

Errors Invalide/Incorrect Definitions	
> AR-ECUC02008 Invalid multiplicity > AR-ECUC03019 Incorrect definition of configuration element	
Solution	Delete parameter in ECUC file not existing in BSWMD file any more.
Example	Some parameters has been replaced. E.g. **/MICROSAR/Dem/DemGeneral/DemEventStorageTrigger** has been replaced by /**MICROSAR/Dem/DemGeneral/DemEventMemoryEntryStorageTrigger**

Errors Inconsistent Connector Prototypes	
> RTE51027 Connector prototype inconsistent. (1 message) > RTE51031 Connector prototype inconsistent. (1 message)	
Solution	Due to changes within the Service Components the ports can become incompatible Application Software Components. Adapt your Application Software Components with the DaVinci Developer to match the new port definitions. The changes are normally resulting from > Correction (the old definition was wrong) > Changed implementation based on AUTOSAR Bugzilla entries / RfC > Changed implementation due to support of newer AUTOSAR Version (e.g. AUTOSAR 4.0.3 to AUTOSAR 4.1.2)

5. Process all further Errors and Warnings given within the Validation View of the DaVinci Configurator
6. Start project update
   Execute a project update using the same set of input files.

Note

Even though the input files have not changed, the new DaVinci Configurator version might have a different algorithm for deriving the parameters. It is essential to run this "neutral" update before updating the project to a new version of the input files.

Cross Reference

Additional information about necessary configuration update steps are described within the Release Notes.

The following describes what to do when updating especially from one release to another.

4.2 Migration Steps from Release 20 SIP to Release 21

In addition to the standard migration from x-1 to x, please find necessary information in the folder **ReleaseNotes** of your delivery.

4.2.1 New checks in DaVinci Developer / Rte

Because of new checks, errors may occur when checking the DaVinci Developer Workspace or during the generation of the Rte with a configuration that was accepted before. Follow the error messages to correct the configuration.

Example

Invalid port for port defined argument value.

The port <portname> of type <client/server> (direction: R-Port) has port defined value(s) (PDAVs) assigned. PDAVs are only valid for server ports. [constr_1386]

4.2.2 vSecPrim

If you use crypto functionality, you have to add the module vSecPrim. Use Project Settings|Modules and add the module via **Select from Software Integration Package (SIP)**.

Cross Reference

ReleaseNote_MICROSAR4_Vector_*.pdf - **FEAT-3330**

4.2.3 v_cfg.h

With Release 21 the file v_cfg.h in the GenData and GenDataVtt folder is not generated anymore. Its content is transferred to the module-specifc cfg files.

If you use v_cfg.h in your application, please use the one from a Release 20 project or adapt your application accordingly.

Cross Reference

ReleaseNote_MICROSAR4_Vector_*.pdf - **FEAT-2949**

4.3 Migration Steps from Release 19 SIP to Release 20

In addition to the standard migration from x-1 to x, please find necessary information in the folder **ReleaseNotes** of your delivery.

Cross Reference

ReleaseNote_MICROSAR4_Vector_*.pdf

Please read carefully the listed features below:

- FEAT-2745
- FEAT-3109
- FEAT-3133
- FEAT-3173

4.4 Migration Steps from Release 18 SIP to Release 19

Open your existing configuration with the new SIP environment, i.e. Configurator 5.16.xx and execute **[SolveAll]**  . After that, solve the remaining validation messages according to your knowledge.

4.4.1 Migration of DEM

The Dem has been reworked to support multiple partitions (refer also to Release Notes). Thus the following steps have to be done.

Cross Reference

For further information refer also to the technical reference of the DEM.

- Delete old DEM Service Component using the RTE Solving Actions.
- DEM runnables have to be mapped again where **Dem_MainFunction** has been before.
  For best performance map **Dem_SatelliteMainFunction** before **Dem_MasterMainFunction**, if they are mapped to the same task.
- Make sure the DCM has AR 4.3 API to the DEM (see parameter **DcmDemApiVersion**).

> Each Service Port Connection to the former Service Component DEM have to be redone to **DemMaster_*** and **DemSatellite_*** Service Components. Think about using the auto connect feature of the DaVinci Configurator Pro.

> The DEM now has an AR4.3 API to the Application SWCs. Thus some Service Interface Types have been changed by signature and/or timing behavior. Adapt your Application SWCs to this.

> **Dem_MemMap.h** has to be provided in the build environment (refer to Technical Reference of DEM).

4.4.2 MemMap Handling

The checking regarding correct MemMap handling has been improved (refer also to Release Notes).This might lead to new reported incorrect MemMap handling, which were not detected before.

4.4.3 Migration of STBM

The StbM now has an AR4.3 API to the application SWCs (refer also to Release Notes). Some Service Interface Types have been changed - so you have to adapt your application SWCs to this. This might be reported by RTE51017, if the Port Interfaces provided by the StbM have been used by the application. Remap the Port Interfaces in the DaVinci Developer - so there will be one Port Interface created for each Time Domain.

4.4.4 New DaVinci Developer Version - Workspace conversion necessary!

In case your DCF file was created with an older DaVinci Developer version (DaVinci Develoer 3.x) open the DCF file with the new DaVinci Developer 4.0. You get a message window, informing you about the necessary conversion of the workspace.

What happens while workspace conversion?

> The ***.dcb** file will be obsolete and replaced by the ***.dvg** which now includes graphical information.

> The ***.dcf** file will be adapted and references the new files.

Caution!

Converted workspaces can NOT be opened with a previous version of the DaVinci Developer 3.x!

Save workspace and go back to the DaVinci Configurator. Open Project settings and adapt DaVinci Developer tool location.

4.5 Migration Steps from Release 17 SIP to Release 18

Open your existing configuration with the new SIP environment, i.e. Configurator 5.15.xx and execute **[SolveAll]**. After that, solve the remaining validation messages according to your knowledge and make sure all unmapped schedulable entities are mapped (RTE01055).

4.6 Migration Steps from Release 16 SIP to Release 17

Open your existing configuration with the new SIP environment, i.e. Configurator 5.14.xx and execute **[SolveAll]**. After that, solve the remaining validation messages according to your knowledge.

4.7 Migration Steps from Release 15 SIP to Release 16

Open your existing configuration with the new SIP environment, i.e. Configurator 5.13.xx and execute **SolveAll**. After that, solve the remaining validation messages according to your knowledge. The following validation messages require further explanation:

1) **AR-ECUC02008 EthTSynGeneral contains a bigger number of parameter EthTSynGmCapable than upperMultiplicity specifies.**

This is a known issue appearing during SIP migration, which is easy to solve. Delete one of the two parameters.

2) **AR-ECUC02039 The target of the reference value StbMEthGlobalTimeDomainRef(value=<SomeEthTSynGlobalTimeDomainContainer>) must be a container of type DefinitionRef: /AUTOSAR/EcucDefs/EthTSyn/EthTSynGlobalTimeDomain.**

This issue occurs, because of the introduced support of AR-4.2.x, which brings along the TimeSync modules. The manually configured reference of the previous SIP needs to be re-configured, i.e. select the appropriate container again from the list of available EthTSyn domains.

4.8 Migration Steps to or over Release 15 according to Feature 1657

4.8.1 PDUs

These steps are necessary whenever you migrate from a release lower than 15 to a release greater or equal to Release 15.

All PDUs that are not derived automatically from the system description have to be mapped again manually. The names of those PDUs are now longer, because they got an additional name decoration. By searching for the former base name, the PDUs can be easily found in the PDU list and remapped again.

4.8.2 DEM

There are minor changes at Dem's interface to NvM, i.e. the Dem has changed some symbol names when accessing its non-volatile data. Please make sure that you have executed **Setup Event Memory Blocks** in the diagnostics domain. Otherwise you may run into compiler errors in `NvM_Cfg.c` due to undefined symbol names.

4.9 Migration Steps Release 12 Projects to Release 13

Former modules IpV4, IpV6 and TcpIp have been merged into one single module TcpIp

After you have loaded the configuration into DaVinci Configurator delivered with Release 13, execute

all preferred solving actions by clicking **SolveAll**.

For all remaining validator messages choose the appropriate solving action individually, e.g.

> Delete all parameters, which are not available anymore (AR-ECUC03019 Incorrect definition of configuration element)

Note

Parameter **SoAdSocketUdpRetryEnabled** might give a hint about how to configure **SoAdSocketXxxAddressResolutionRetryEnabled**, but is not necessarily the same.

> Go through remaining messages issued by SoAd and TcpIp and solve them. If in doubt, leave features enabled, rather than disabled.

Derivation of Periodic Dcm connection has changed

After you have solved all validation errors you should execute an update of your input files.

Caution!

To avoid two update processes you should delete (before updating your configuration) the derived container with definition **/MICROSAR/Dcm/DcmConfigSet/DcmDsl/DcmDslProtocol/DcmDslProtocolRow/DcmDslConnection/DcmDslPeriodicTransmission**.

After having updated your configuration, right-click on the container with definition **/MICROSAR/Dcm/DcmConfigSet/DcmDsl/DcmDslProtocol/DcmDslProtocolRow/DcmDslConnection** and restore the derived sub-containers.

Derivation of CDD input files changes Dcm port interface names (Loosing service connections of SWC to Dcm and even danger of damaging Application SWC).

After you have updated your configuration you will see validation messages like **RTE51017 Unresolved port interface reference '/MICROSAR/Dcm_swc/...' for PPort prototype '...'**

> Open **Runtimesystem** | **ECU Software Components** | **Service Components** | **Dcm** | **Service Connectors** and mark all unconnected client ports of the Dcm.

> Create server ports at the Application SWC mentioned in the validation messages. Do NOT create server runnables.

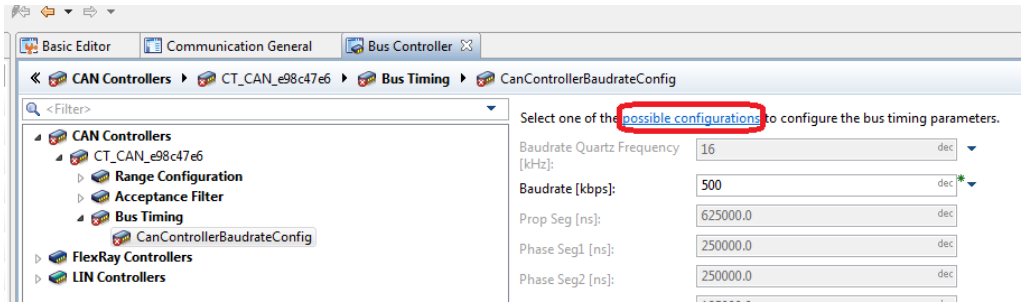
In the DaVinci Developer workspace associated with your configuration you will find many untriggered runnables, these triggers are lost during the update process.

> Create a trigger of type **on operation invocation** for each of these runnables and select the matching operation of the appropriate port.

Note

Only port interface of type **DataServices** and **RoutineServices** might be affected.

4.10 Migration Steps from Release 8 to Release 9

Errors Invalid Baud Rate	
> CAN02012 An invalid baud rate value is configured > CAN02012 invalid baudrate settings: [MANUAL ACTION]: change clock or baudrate. /ActiveEcuC/Can/CanConfigSet/CT_CAN_e98c47e6/CanControllerBaudrateConfig	
Solution	The tool now checks the values of the baud rate configuration, they might have been wrong before. Check valid settings of McuClockReferencePointFrequency referenced by **/MICROSAR/Can_Cano-eemuCanoe/Can/CanConfigSet/CanController/CanCpuClockRef** Then select one of the offered baud rate configurations by clicking on 'possible configurations'. 

4.11 Migration Steps from Release 7 to Release 8

If you get a new delivery with Vector BSW Release 8, it is necessary to update some parameter values of your configuration to the new release.

After you have updated your project with the new SIP the following steps are necessary:

1. Select ECUC parameter which is not defined by a BSWMD Parameter definition.
   (Error number AR-ECUC03019)
2. Check whether additional parameters exist.
   (Parameters without value definition or with default value)
3. Define value of the new parameter with value from the existing parameter
4. Delete existing parameter

Use the same steps for container updates.

Note

Select all **VectorCommonData** containers, which are marked with an error, use multi-selection in the DaVinci Configurator Basic Editor and delete them via **Delete Container**.

All **Use RTE** switches and **Production Error Detection** switches can be deleted without check.

5 Update DaVinci Tools

Cross Reference

The DaVinci service packs are located at the **Vector Download Center**.
https://vector.com/vi_downloadcenter_en.html

The Update of the DaVinci Tools is described within the installation guide.

Cross Reference

You will find the installation guide document at the Vector Knowledge Base:
https://vector.com/kbp/entry/640/

Note

Please note that compatibility is only granted for service packs provided for the same major and minor version.

5.1 DaVinci Configurator Pro

If necessary, update DaVinci Configurator Pro service packs, therefore you have to download them from the Vector Download Center and install them via DaVinci Configuration Service Pack Installer. Select the installed SIPs which should be updated.

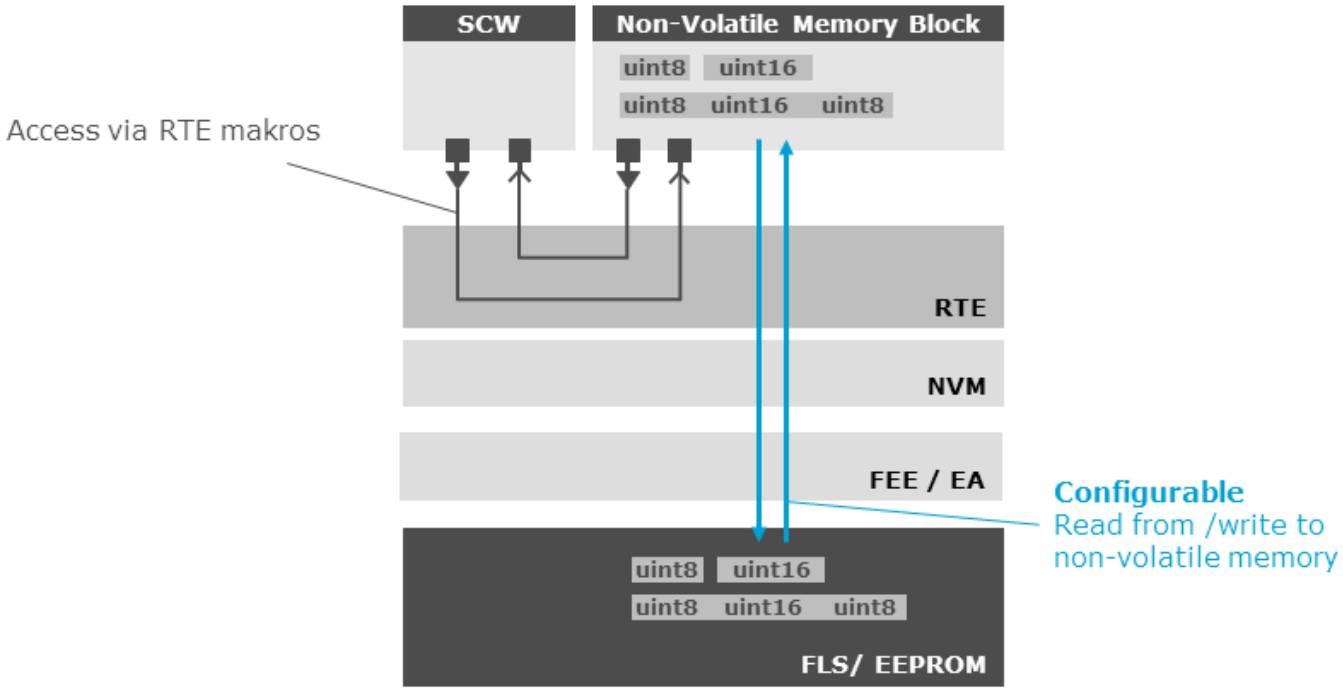
5.2 DaVinci Developer

Service Packs of DaVinci Developer are also available in the Vector Download Center. The service packs include an installer to update your DaVinci Developer installation.

6 Non-Volatile Memory Block

This is a description how to handle data that has to be stored to non-volatile memory like FLASH or EEPROM.

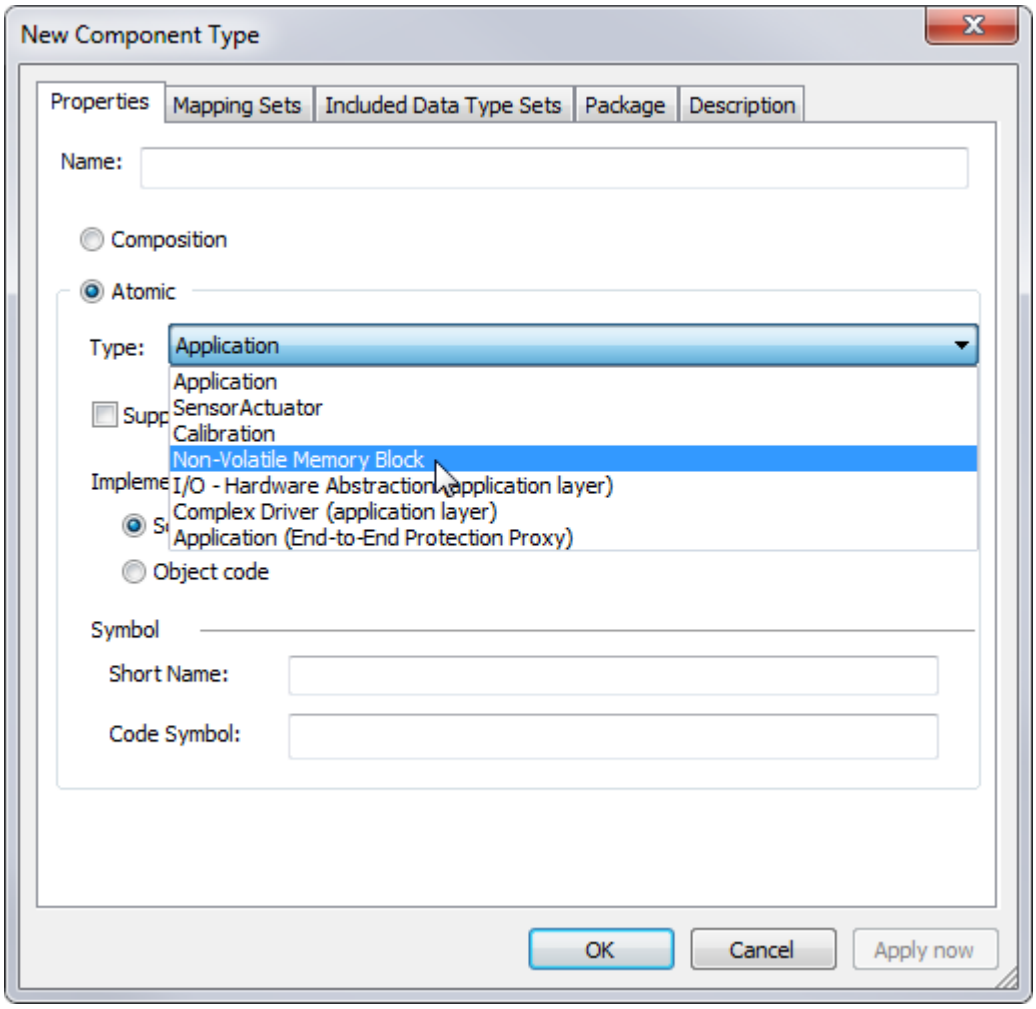
The illustration below shows briefly the concept behind. Data is defined in a Non-Volatile Memory Block. The application SWC has access to this data via the RTE. The data is stored and addressed in RAM. At configuration time you can define which data in which format you need and when and how this data is written to and read from the non-volatile memory. You can also trigger the immediate writing of this data to the FLS or EEPROM.



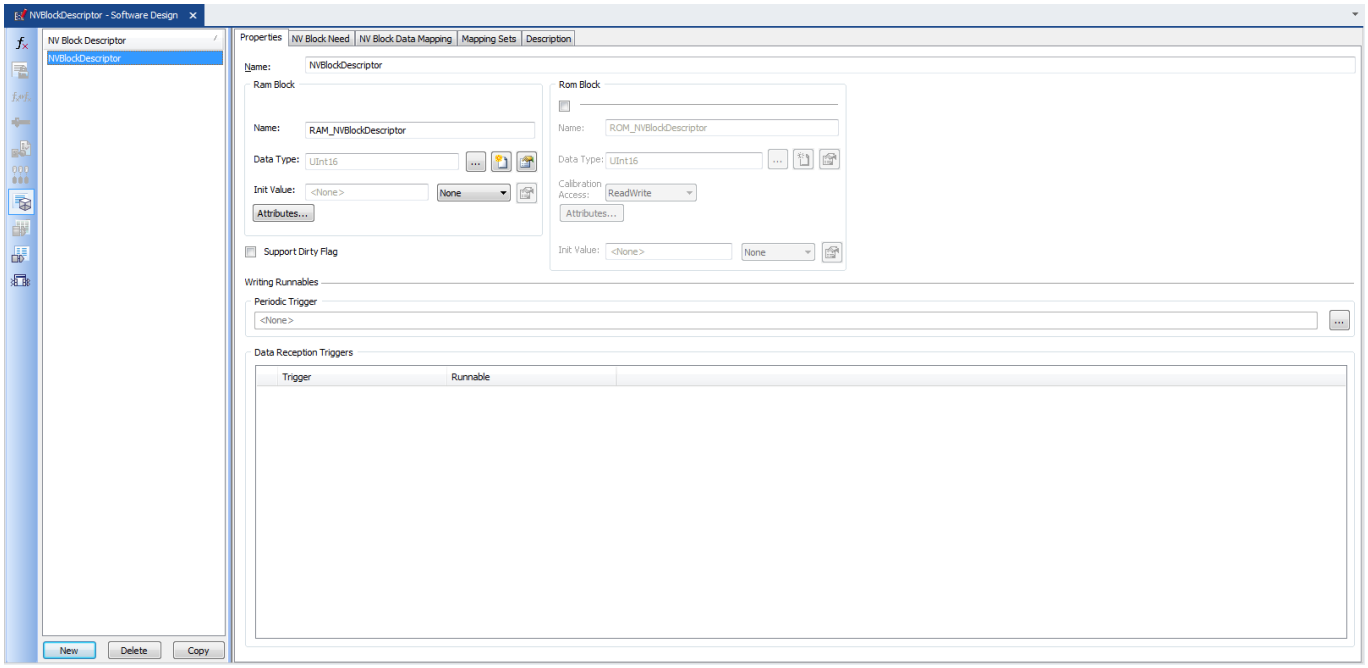
6.1 Configure and use Non-Volatile Memory Block

The following description shows how the Non-Volatile Memory Block is configured and used.

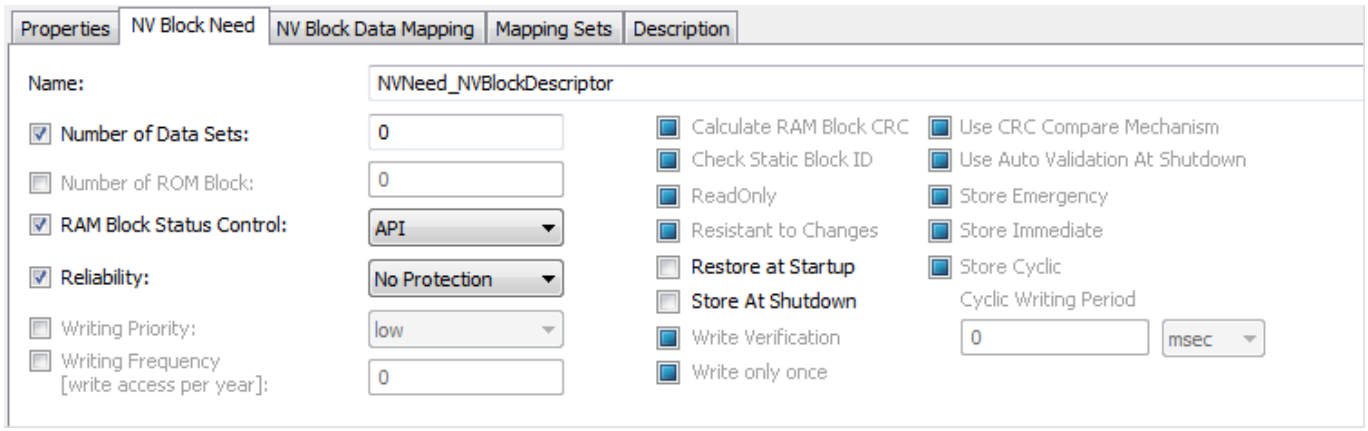
In the Object Browser of DaVinci Developer, add a **New Application Component Type**, select **Non-Volatile Memory Block** as type and give a **Name** to this new component type. Double-click the new **Non-Volative Memory Block** in the Object Browser and select **NV Block Descriptors**



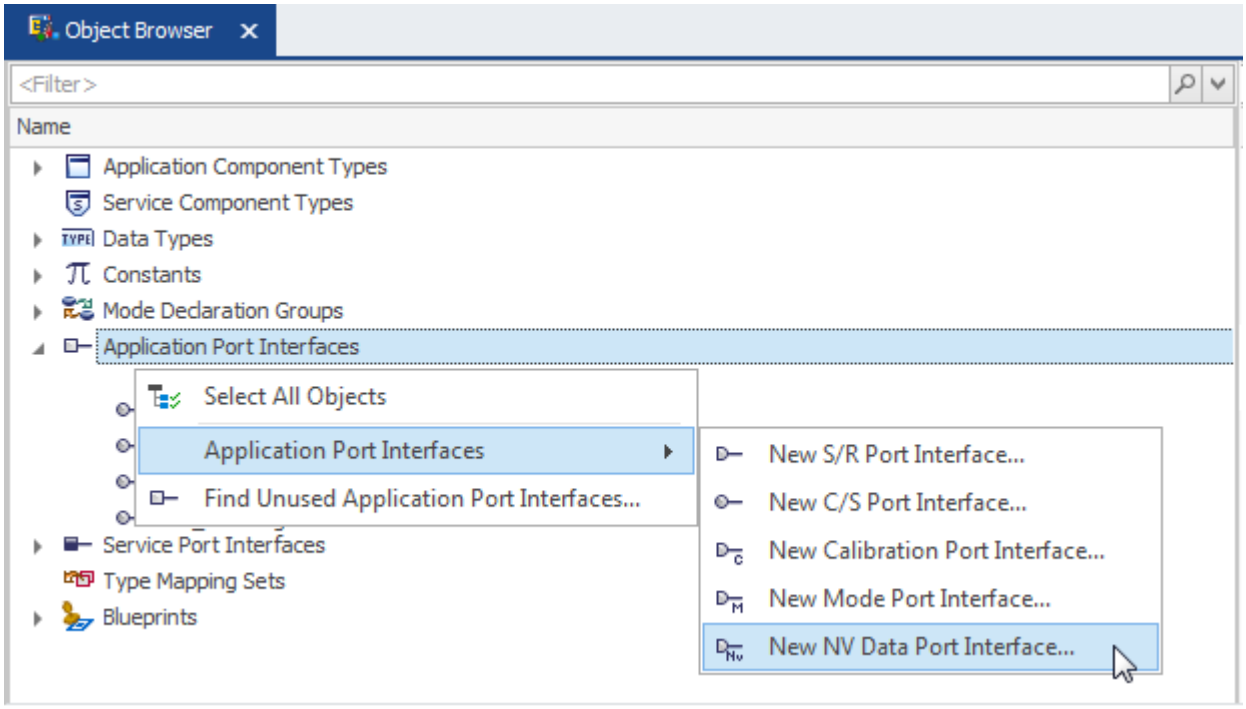
Use **[New]** to define a new NV Block Descriptor. On the **Properties** tab define the data you want to store to non-volatile memory and its type. This can be a single variable like integer or boolean. Also complex data types like records or arrays can be defined.



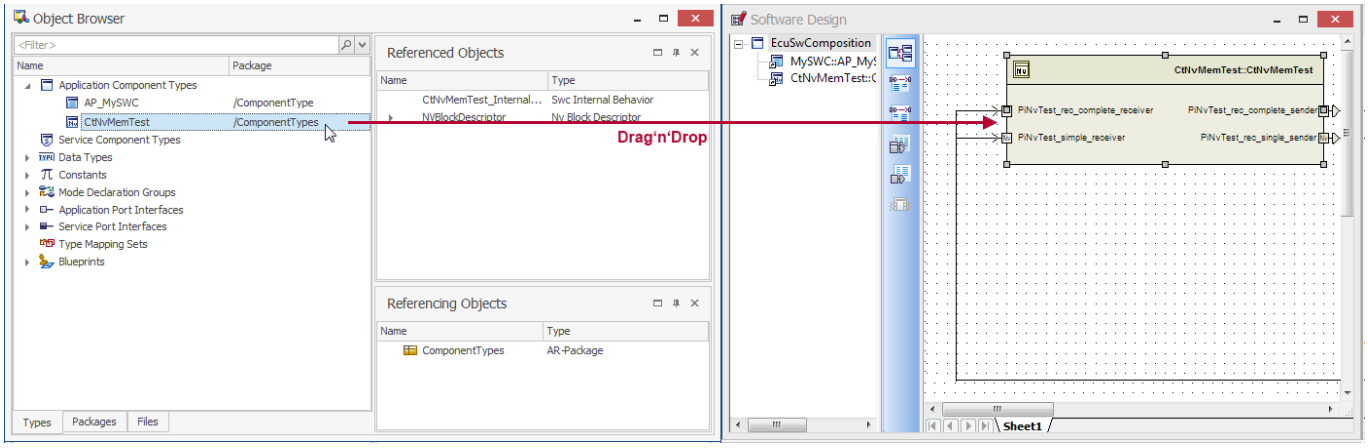
Select the tab **NV Block Need** and define, when and how your data should be finally written to NV memory (FLS or EEPROM).



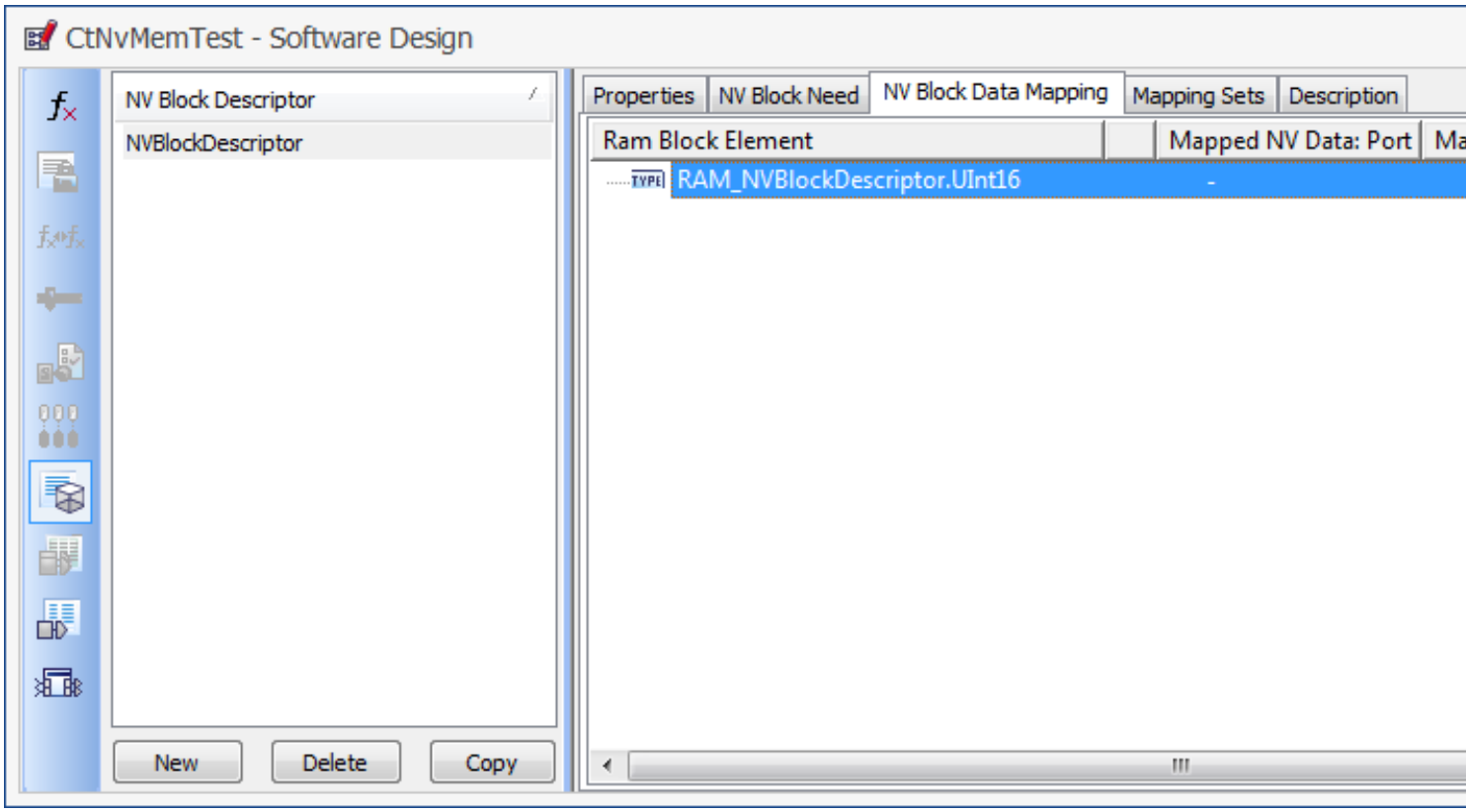
Now add New NV Data Interface… to define a port to access the data defined before. Select suitable Data Element Prototypes.



Add the NV Component Type to your Software Design (drag'n'drop from Object Browser), assign the NV Data Interface to the NV Component Prototype and another NV Data Interface to your application SWC. Draw the necessary connectors by hand. Define the Init Values of all ports.



Double-click NV Component Prototype and perform the internal mapping on the NV Block Data Mapping tab (right-click to open context menu).



6.2 Port Access of your Runnables

What is left now – the access of your application runnable to the **NV Data Interface**(s). Open your application software component, select a runnable and then the tab **Port Access**. Use the **[New]** button to add a read or write data access to the **Non-Volatile Memory Block**.

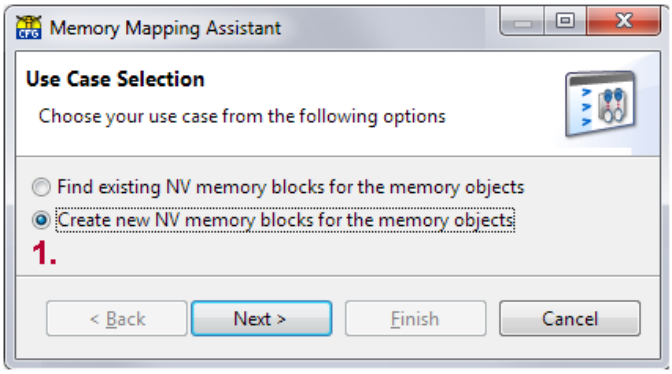
Save your project and switch to DaVinci Configurator Pro.

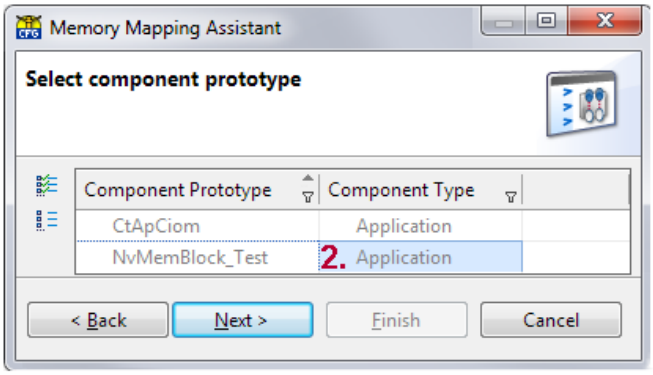
6.3 Memory Mapping in DaVinci Configurator Pro

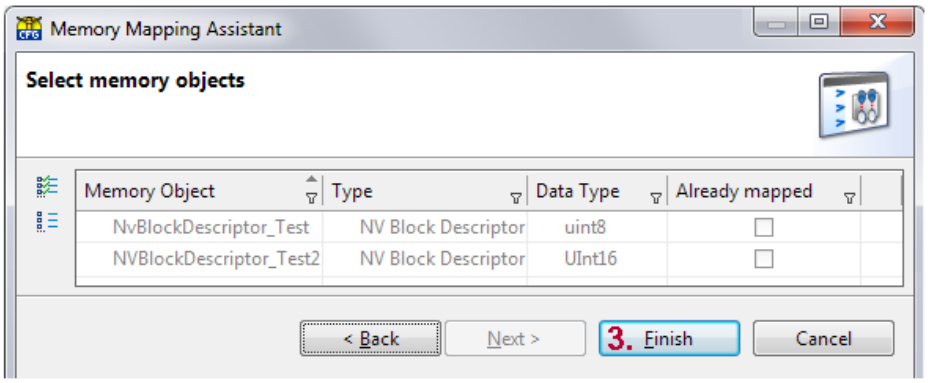
Synchronize the system using the synchronization message in the **Validation** area.

Open **Runtime System**. Click **Add Memory Mapping**.

1. Select **Create new NV memory blocks for the memory objects**.
2. Select the block created before in the DaVinci Developer
3. Get information about unmapped blocks and click **[Finish]**.

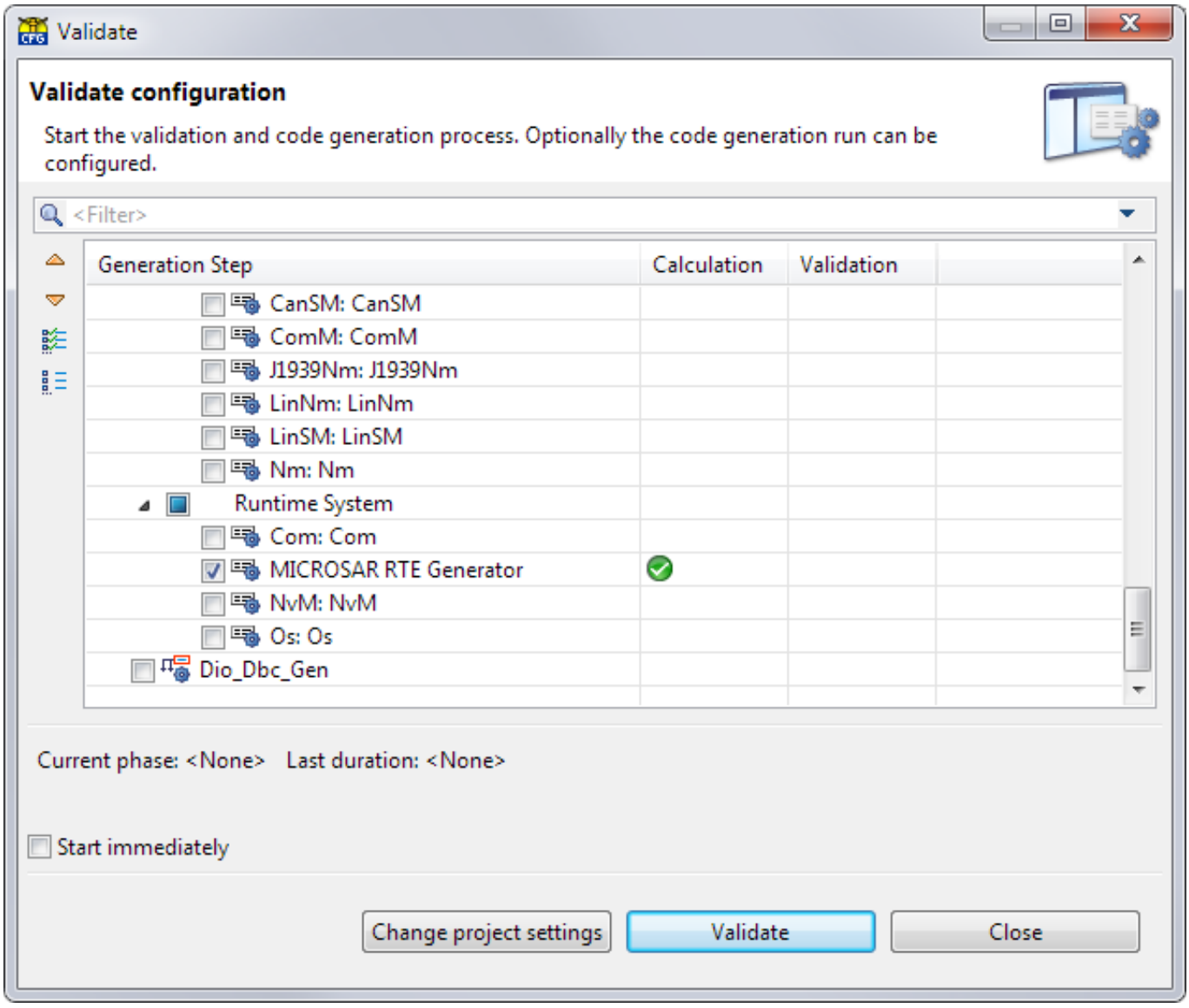

Memory Mapping Assistant
Use Case Selection
Choose your use case from the following options
Find existing NV memory blocks for the memory objects
Create new NV memory blocks for the memory objects
1.
< Back
Next >
Finish
Cancel


Memory Mapping Assistant
Select component prototype
Component Prototype
Component Type
CtApCiom
Application
NvMemBlock_Test
2.
Application
< Back
Next >
Finish
Cancel


Memory Mapping Assistant
Select memory objects
Memory Object
Type
Data Type
Already mapped
NvBlockDescriptor_Test
NV Block Descriptor
uint8
NVBlockDescriptor_Test2
NV Block Descriptor
UInt16
< Back
Next >
3.
Finish
Cancel

6.4 Validate the RTE

1. Open the validation window via
2. Deactivate all boxes via
3. Select RTE and click **[Validate]**.



When implementing you runnable code, you can now access the defined variables by using RTE read and write macros.

7 CANoe OsCAN Library

Note

The created library has to be part of the CANoe configuration.

Note

With CANoe 8.x it is not sufficient to start CANoe from Visual Studio, as the execution of the DLL is done in the process **RunTimeKernel**. Therefore the Visual Studio environment has to be attached to this process.

7.1 Emulate your project with CANoe

You can use CANoe and the CANoe osCAN Library to simulate your project without any hardware but your PC.

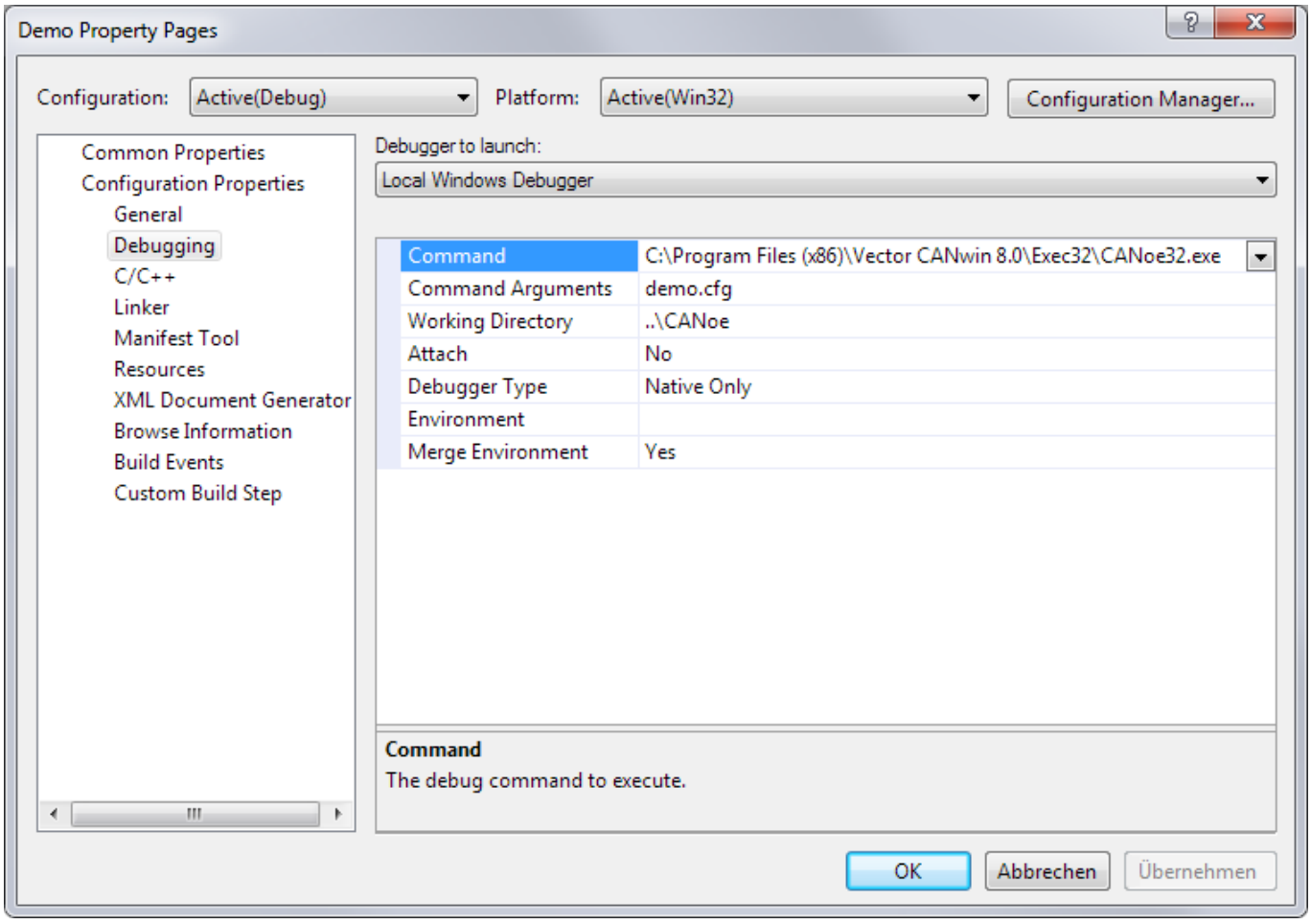
Make sure your project includes all files from:

- the BSW delivery (CAN, CANIF, CANNM, OS…)
- The configuration files of DaVinci Configurator Pro
- The generated RTE (RTE_DEV)
- And the folder sources with the component templates filled with your code.

All generated code has to be compiled and linked

Right click your project and select Properties. Then click Debugging and fill in the following information:

- Give the path to your CANoe in the Command line
- Select the CANoe project, here demo.cfg as Command Argument
- Set the Working Directory to ..\CANoe (i.e. to the path, where your demo.cfg is located)
- Set the Debugger Type to Native Only



Now you can start the simulation via <F5>.

You can run your ECU together with other ECUs, programmed like this one and represented as DLL or as remaining bus simulation. You can debug the system using breakpoints in Microsoft Visual Studio™.

8 Basic Software Modules

This chapter deals with BSW modules, Basic Software Modules. First they are introduced theoretically then configured with the DaVinci Configurator Pro.

8.1 Generic BSW Modules

Before you start to configure the BSW module with DaVinci Configurator Pro you have to know something about BSW modules.

This chapter introduces a generic BSW module and wants you to get a basic understanding of it:

- How it looks like
- Of what it is formed
- How it is configured
- How it works
- What is fix
- What is generated and

some special topics like

- Memory sections and
- Exclusive areas

Cross Reference

Find the detailed description of the BSW module in the folder **Doc|TechnicalReferences** of your delivery.

8.2 What is a BSW Module?

A Basis Software Module (BSW module) consists of:

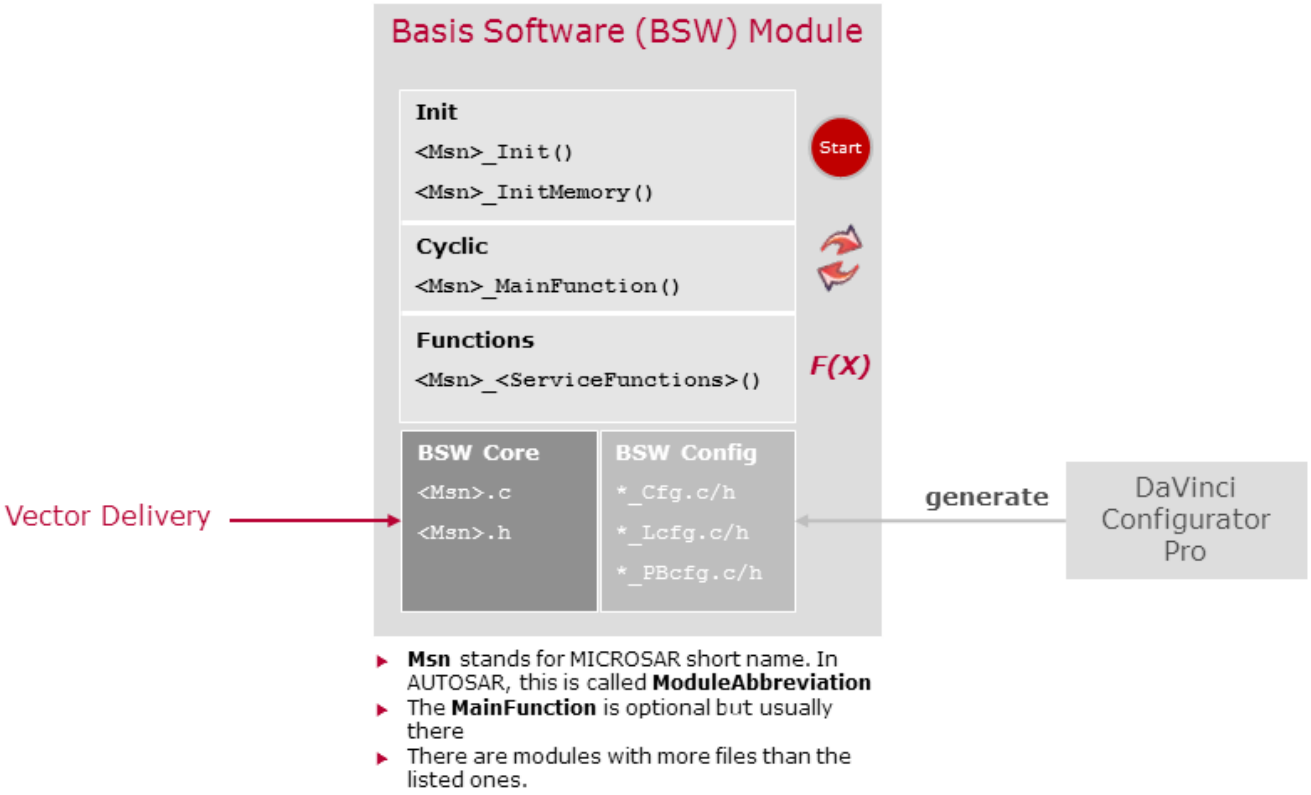
- Static BSW files (algorithms and functions)
- Generated files (data and sometime algorithms and functions)

To get it to work it has to be

- Included
- Initialized
- Its main function has to be called cyclically (in most of the cases)

The mapping of its code and variables can be configured as well as their exclusive areas, to protect the module's internal data.

- Memory Mapping
- Exclusive Areas



Example

Examples for the short names: Dem, Dcm, EcuM, etc.

8.3 BSW Module Configuration

The amount and properties of configurable parameters is defined within the <Msn>.bswmd file. BSWMD stands for Basic Software Module Description file. This file is read by the BSW configuration tool (DaVinci Configurator Pro) and used to create the Basic Editor. The Basic Editor is the tool expression of all configurable parameters of the BSW module, displayed in a generic way.

Note

In addition to the Basic Editor, the Vector BSW configuration tool DaVinci Configurator Pro provides comfortable views that make configuration of BSW modules very comfortable.

The DaVinci Configurator Pro generates the configuration files for every used BSW module. These are files like

- <Msn>_cfg.c/h,
- <Msn>_Lcfg.c/h,
- <Msn>_PBcfg.c/h

8.4 BSW Initialization

8.4.1 <Msn>_InitMemory()

Most BSW module needs variables that have to be initialized before the call of `<Msn>_Init()`. This can be global or static variables. According to AUTOSAR concept, all necessary variables will be initialized by the start-up code. Where this is not done or not possible the function `<MSN>_InitMemory` has to be called as an alternative.

8.4.2 <Msn>_Init()

Every BSW module has to be initialized at start-up. This is done with the start-up of the ECUM. The function to initialize a BSW module is called: `<Msn>_Init()`

Note

Our modules and functions are protected against multiple initializations.

8.5 BSW Module Version (xxx_GetVersionInfo)

Each BSW module provides an API **<MSN>_GetVersionInfo**.This API returns the BSW published information

- BSW version
- Module ID
- Vendor ID

The BSW versions are BCD-coded (binary coded decimal). The availability of this API can be individually configured for each module.

8.6 Cyclic Calls

8.6.1 <Msn>_MainFunction()

To run and to work, most of the BSW modules (except for those, that are triggered by interrupt), have to be called cyclically with a configured call cycle. The module derives its time base from those cyclic calls. This cyclically called function is called main function and its name is formed like: `<MSN>_MainFunction()`

Note

In most of the cases it is mapped to a cyclic task that is called cyclically triggered by an Alarm of the Operating System.

8.7 Service Functions

Cross Reference

For information about **Client Server Theory**, refer to Data Mapping (section **Service Mapping**).

8.8 Critical Sections - Exclusive Areas

BSW modules use so-called exclusive areas to protect their resources (module internal data) from concurrency. Each module defines its own exclusive areas, up to 5 can be defined per AUTOSAR definition.

Whether a critical section needs to be handled or not depends on the possibility of a concurrent access onto a given resource of the BSW. It therefore depends on e.g. OS or hardware configuration constraints. If a critical section needs to be handled, different measurements are possible, ranging from a global interrupt lock to the usage of OS semaphores.

Example

As an example, a definition of an exclusive area could look like:

```
SchM_Enter_Com_COM_EXCLUSIVE_AREA_2();

com_TxModeHdlr_DelayTimeCnt[i]--; /* Protected object */

SchM_Exit_Com_COM_EXCLUSIVE_AREA_2();
```

There are different locks available i.e. different interrupt sources could be locked. Global interrupt, peripheral interrupt like CAN or no lock at all.

Cross Reference

Refer to the technical reference document of the BSW module to get information how exclusive areas are defined and used.

8.8.1 Memory Section

Every generated code and variable of the <MSN> modules is enclosed by #defines. Via the template file `compiler_memMap.h` you can define the mapping of each single section of a <MSN> module.

Cross Reference

See more about the memory mapping via `memMap.h` in the technical references of each <MSN> module.

8.8.2 Switch <MSN> Modules Off

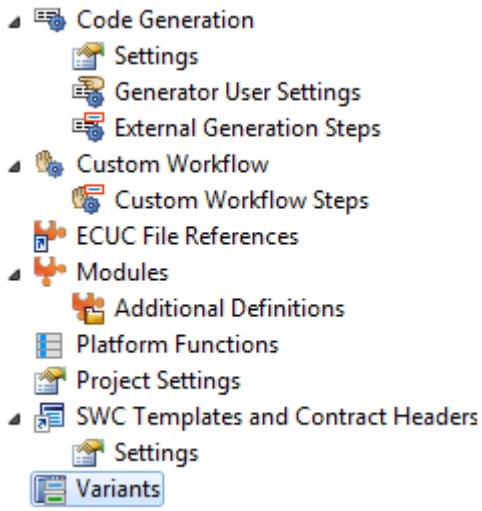
Most of the modules can be switched off via: `<Msn>_Shutdown`

9 Variant Handling

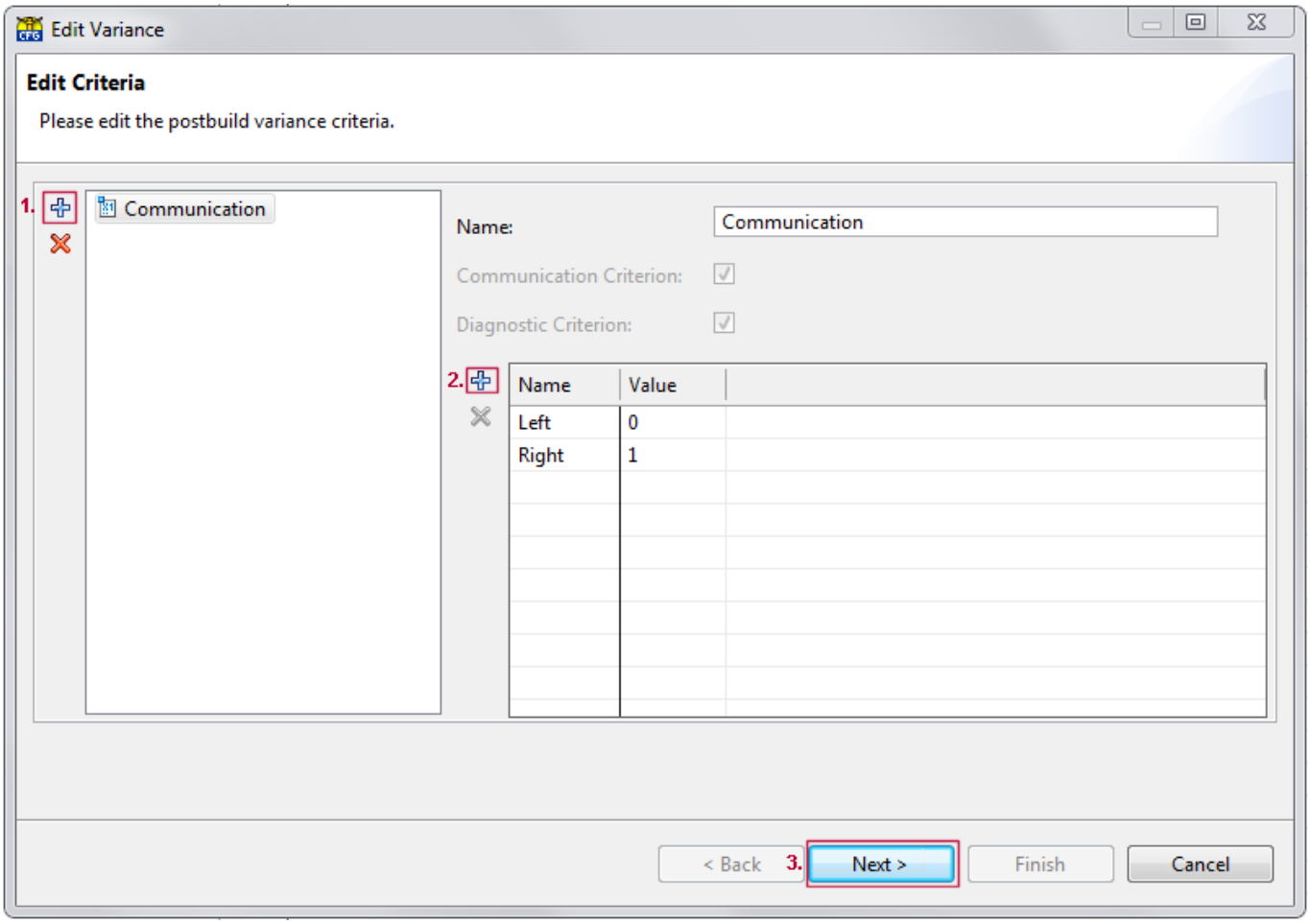
9.1 Define Criterion and Variants

If you use variant handling within your project you have to define your variants first.

After you setup your project, open the **Project Settings Editor** via **Settings** icon and select **Variants**.



Select the **Edit Variance** icon to add criterion, define criterion values and map criterion values to variants.

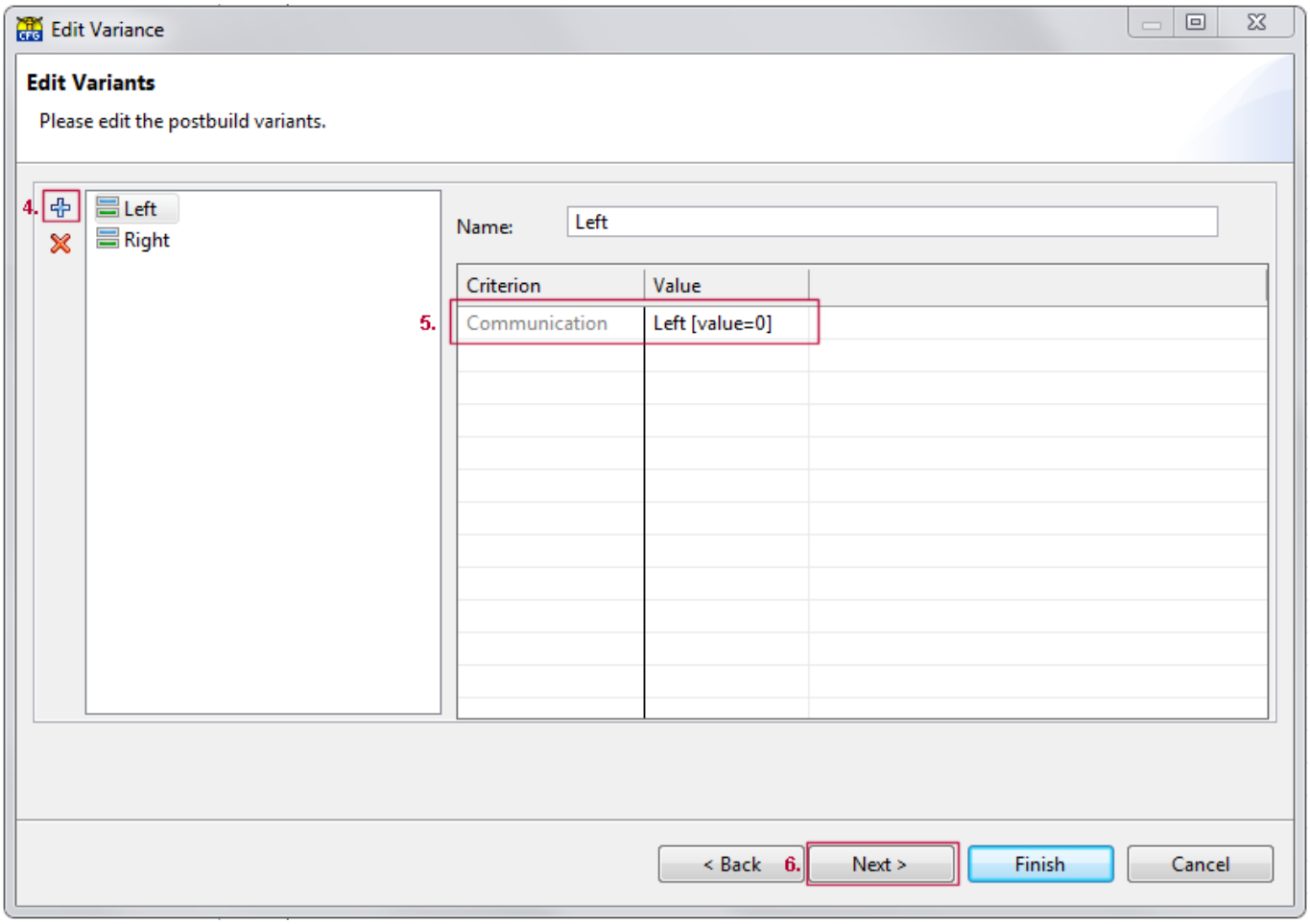


1. Add ✚ new Criterion.

Note

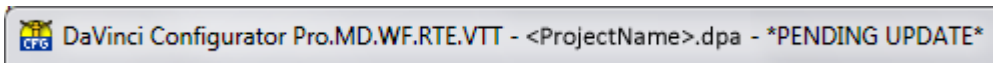
The DaVinci Configurator Pro is currently limited to a single criterion. This criterion can be used for **diagnostic** and **communication** variance.

2. Add ✚ one criterion value for each variant.
3. Go on with **[Next >]**



4. Add ✚ your Variants
5. Define Criteria Value for your Variants
6. Go on with **[Next>]** to see a summary of your definitions or **[Finish]** to come back to Variants view.

After finishing variant definition, the tool enters to **PENDING UPDATE** project state.



Note

The DaVinci Configurator Pro is currently limited to a single criterion. This criterion can be used for **diagnostic** and **communication** variance.

9.2 Add and Assign Input Files to Variants

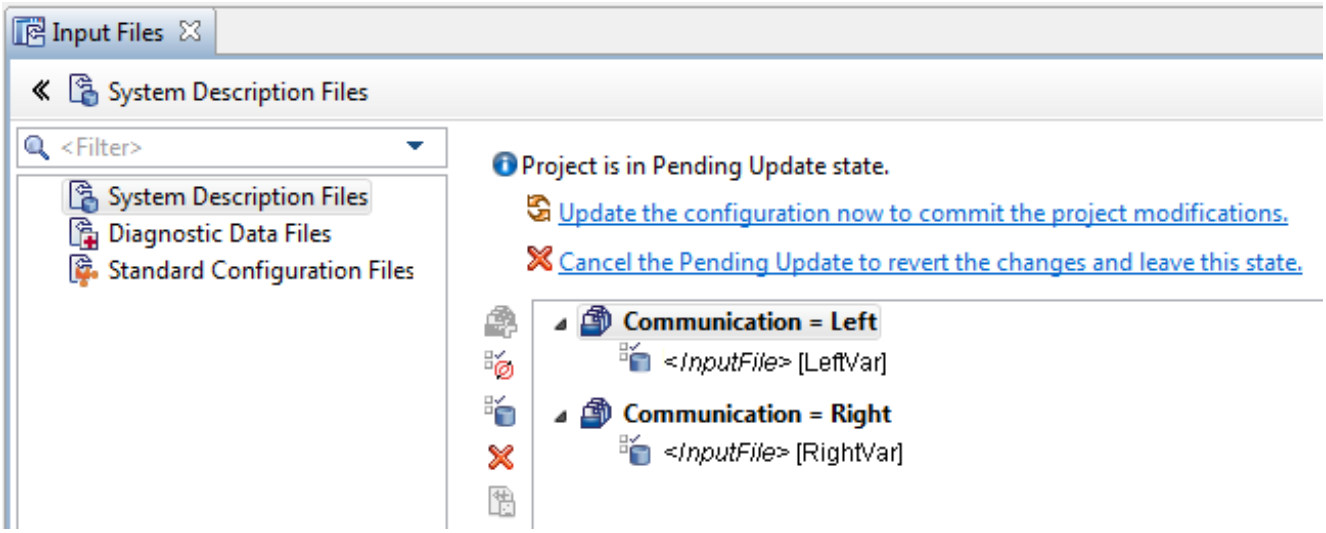
Open Input Files editor via **Project | Input Files** or use the **input file button** of DaVinci Configurator Pro toolbar.

Select **System Description Files** and add your Input Files via **Add File Sets icon** . The Add File Set Assistant opens.

Note

The following step is necessary for each variant.

Select the **Post-Build Criterion Value** (Variant) your input file will be valid and go on with **[Next >]**. Add your input file(s), select your ECU Instance and confirm with **[Finish]**.



After adding and assigning your input files our project is still in PENDING UPDATE state.

Select **Update the configuration now to commit the project modifications** to start update process.

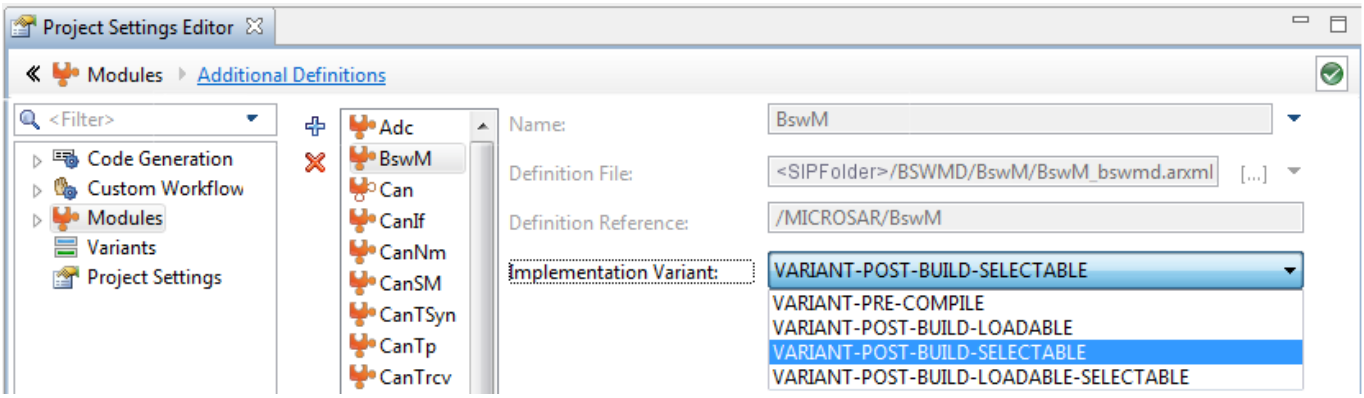
Cross Reference

What happens while Update Process? See section STEP2 Define Project Settings on page 35.

9.3 Define Variance for BSW Modules

Open Project Settings Editor via **Project** | **Project Settings** or use **Settings** icon at DaVinci Configurator Pro toolbar.

Select **Modules** to see all modules currently activated for your project.



For each BSW module define if it supports variance or not. Therefore choose **Implementation Variant**.

- **VARIANT-POST-BUILD-SELECTABLE**
  No post-build loadable update of the configuration at post-build time. All variants are configured at pre-compile time.
- **VARIANT-POST-BUILD-LOADABLE-SELECTABLE**
  post-build loadable update of the configuration data is supported using MICROSAR Post-Build Loadable. This feature requires dedicated licensing.

Note

Multi selection is possible within module view of the DaVinci Configurator Pro, this enables you to set variance for several BSW modules at the same time.

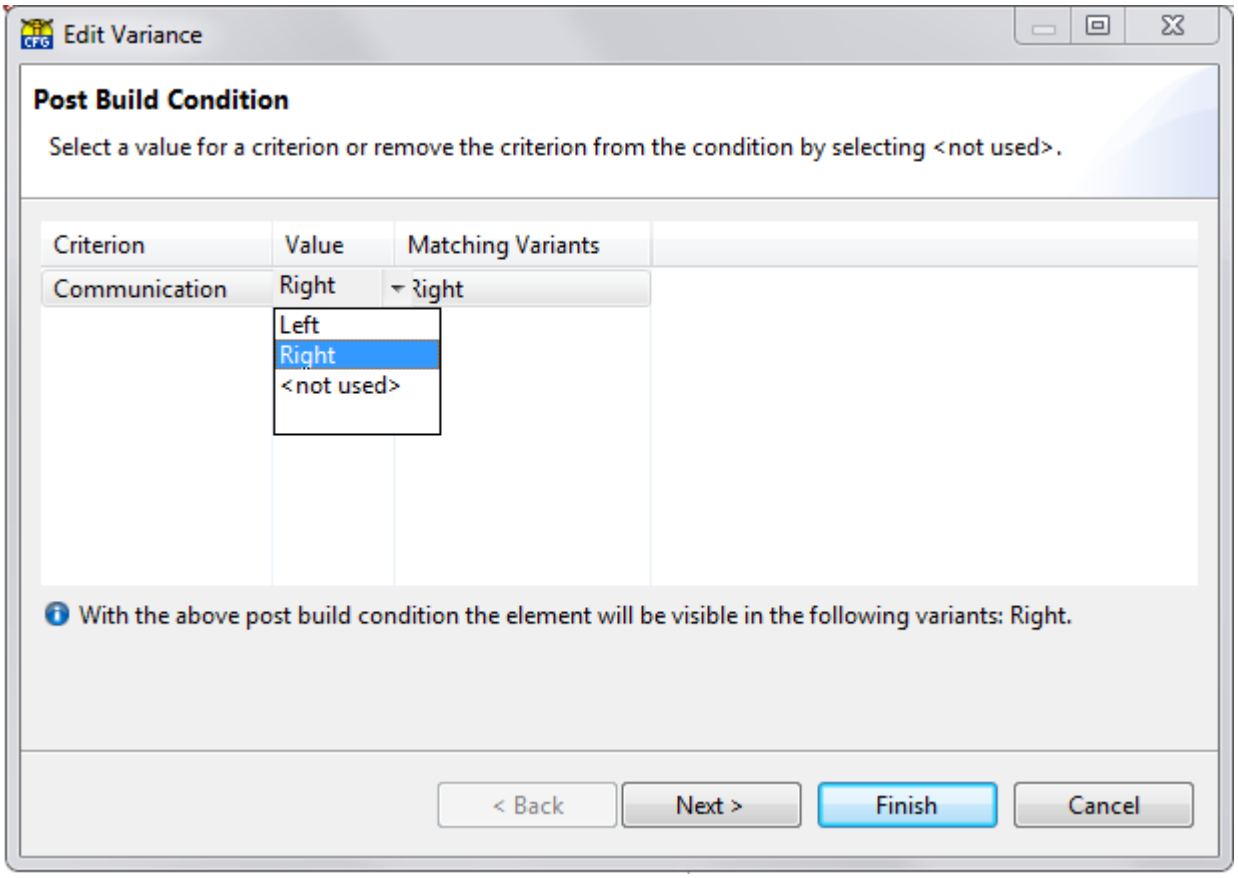
9.4 Configure and Validate BSW

The DaVinci Configurator Pro highlights variant parameter and container using a brown [V]. When changing the configuration it has to be considered whether the change shall be applied to all variants or if the change shall be done only for one variant.

Note

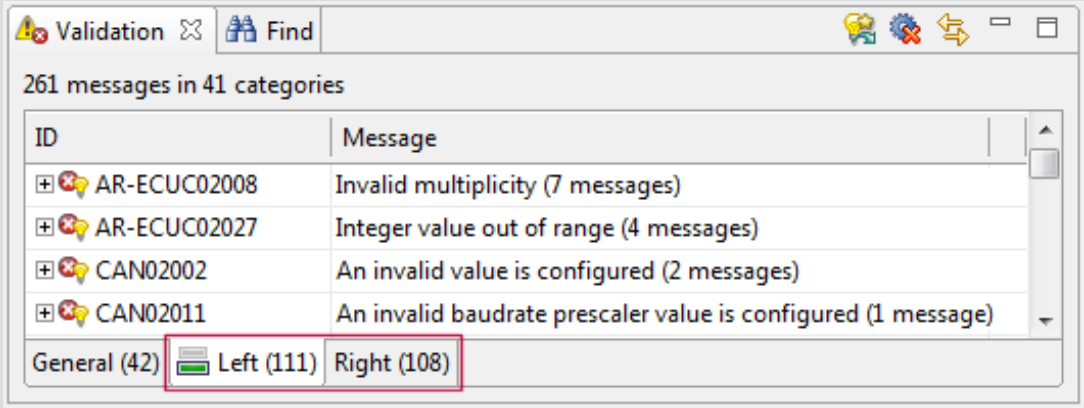
You can select your **Active Variant** at the DaVinci Configurator toolbar.

To define a value for a single variant only, right-click on parameter, select **Edit variance** and define variant (**Value**).



Note

The validation view includes views for each variant.

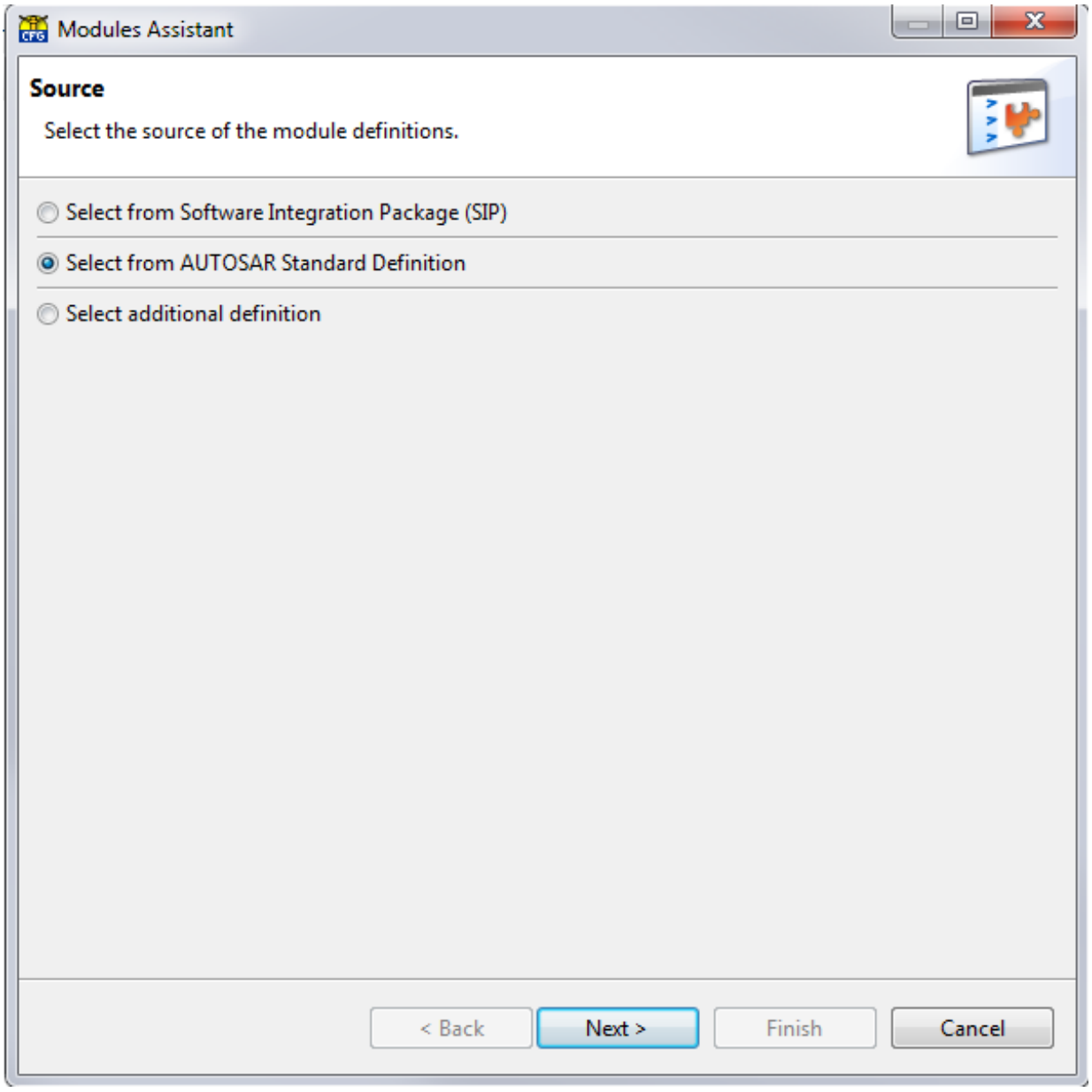


10 Add Module Stubs From AUTOSAR Definition

In some cases it is necessary to add a module, which is not delivered within your SIP, e.g. a delivered module (DCM) needs references and interfaces to a not delivered module (NVM), otherwise the validation fails.

To avoid validation and generation errors you have to add the required module according to AUTOSAR standard definition. Therefore open **Project Settings Editor**, select **Modules** and open Modules Assistant via add .

Choose **Select from AUTOSAR Standard Definition** and go on with **[Next]**, select the required module and confirm with **[Finish]**.



Now the module is available at the **Basic Editor**. The module based on AUTOSAR Standard definition needs to be configured basically with all necessary definitions required by the MICROSAR modules within your project.

11 TCP/IP Stack Migration Of Projects Based On MICROSAR R11 And Earlier (due To FEAT-261)

The API between SOAD and TCPIP is now realized according to AUTOSAR 4.2.1.

11.1 SD – Service Discovery

The Service Discovery module delivered with MSR4-R12 is conforming to the AUTOSAR Release 4.2.1. Due to changes of the configuration containers introduced with this AUTOSAR release, a legacy SD configuration has to be adapted.

Main changes

- Removed possibility to configure a SdEventHandler with the "Auto Available" feature. The corresponding parameter can be removed without any impact to the configuration.
- Additional references SdMulticastEventSoConRef and SdConsumedEventGoupMulticastGroupRef. These references simplify the validation and improve the traceability of the configuration.
- Each SdServerService and SdClientService contains additional references to SoAdSocketConnectionGroups (SdClientServiceXxxRef and SdServerServiceXxxRef). These references specify the UDP and TCP SoAdSocketConnections which shall be used for the communication of the corresponding service. This implies, that in case of a SdServerService, the entire communication of the service uses a single SoAdSocketConnectionGroup per communication protocol. The previously required additional SoAdSocketConnectionGroup to support provided methods can be removed.
- The legacy SdClientServiceXxxRef and SdServerServiceXxxRef within the SdConsumedMethods and SdProvidedMethods can be removed.
- Changed multiplicity of the SdEventHandlerXxx container. Due to additional APIs and configuration possibilities within the SoAd module, the configuration of the SdEventHandler is simplified. In contrast to the legacy configuration, each SdEventHandler requires exactly one SoAdRoutingGroup per communication protocol in order to enable and disable the transmission-paths of the corresponding SomeIp messages. The configuration of independent SoAdRoutingGroups per used SoAdSocketConnection is not required anymore.

11.2 TCPIP - Transmission Control Protocol / Internet Protocol

The API of the module TCPIP has been adapted to be conform to AUTOSAR release 4.2.1. Also the configuration interface between TCPIP and its upper layer (e.g. SOAD) has been adapted.

Main changes

- Sockets are not preconfigured to be used for a specific user or port / network-address combination anymore. The sockets now have to be requested and bound during runtime, and rx and tx buffers have to be requested (via TcpIp_ChangeParameter) before they can be used.

- TCPIP now implements the socket users according to AUTOSAR 4.2.1. In earlier AUTOSAR versions the only user of TCPIP has been the SOAD, while the Vector implementation of TCPIP always supported a multi user concept. Socket users are now called socket owners (renamed according to AUTOSAR)
- TCP server sockets now use a more BSD-like (Berkeley Software Distribution) behavior than before. To accept multiple connections on one listen socket, the maximum number of accepted connections is now configured during runtime when calling TcpIp_TcpListen.

Configuration changes

- Most elements from TcpIpSockUser are transferred to TcpIpSocketOwnerConfig
  - All existing TcpIpSockUser containers will be removed during the SIP update
  - The new TcpIpSocketOwnerConfig container for the SOAD will be created automatically
  - For each additional socket owner a TcpIpSocketOwnerConfig container has to be created manually
- TcpIpSockConfig is removed, so all old references to socket configurations have to be deleted
- The container TcpIpTcpSocketBuffer is added, here the required tx and rx buffers have to be configured
- The number of listen sockets (server sockets) a socket owner wants to use has to be configured using the parameter TcpIpSocketOwnerTcpListenSocketMax

11.3 SOAD - Socket Adapter

The API to the lower layer module TCPIP has been adapted to be conform to AUTOSAR release 4.2.1.

Now all sockets and their resources are assigned dynamically at runtime.

Configuration changes

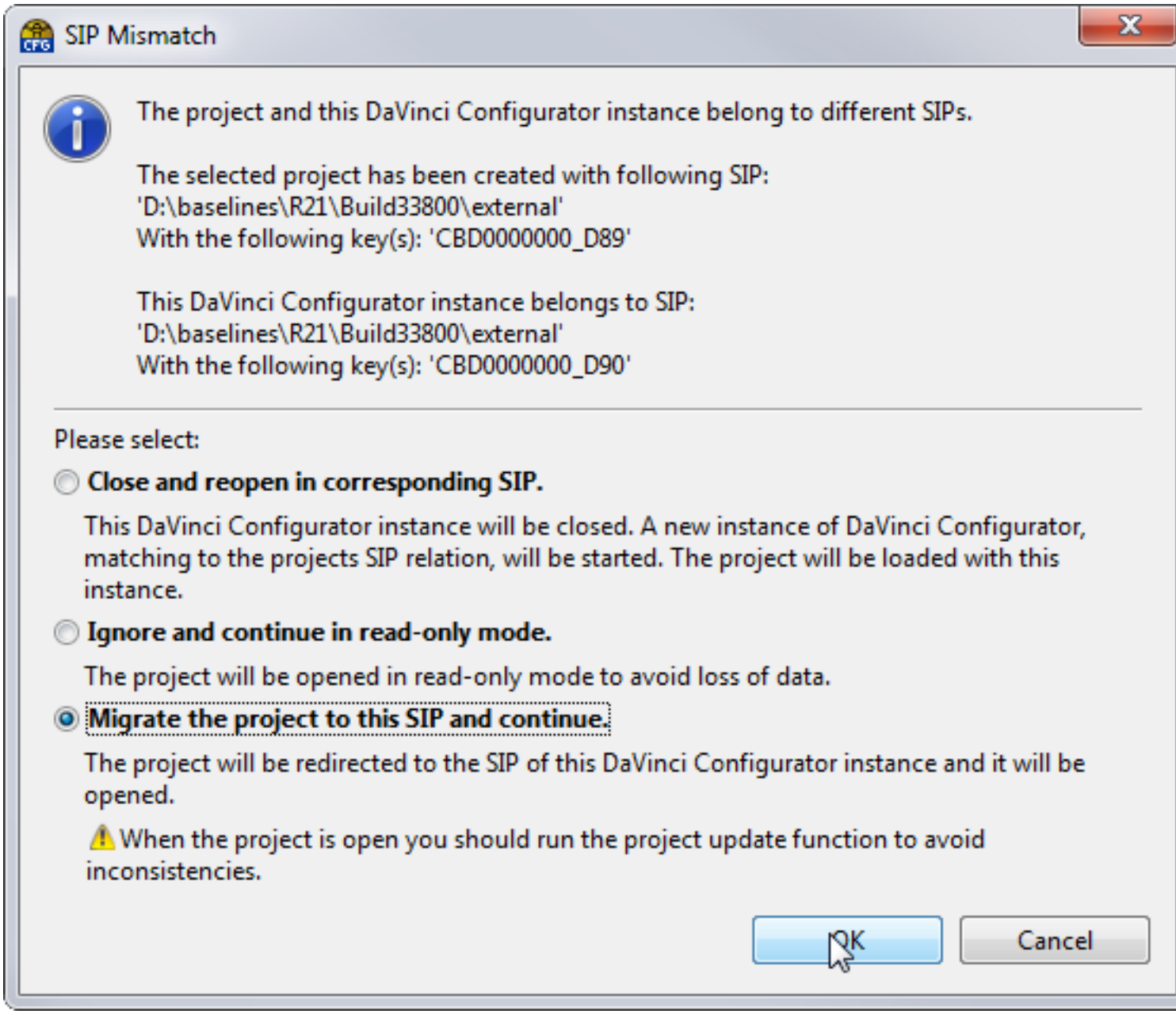
- Container TcpIpSockConfig does not exist anymore. Tx and Rx buffer configuration for a TCP socket is now located in SoAd. For UDP no adaptions are needed.
  - TcpIp/TcpIpConfigSet/TcpIpSockConfig/TcpIpSockTxBufSize →SoAd/SoAdConfig/SoAdSocketConnectionGroup/SoAdSocketProtocol/SoAdSocketTcp/SoAdSocketTcpTxBufferMin
  - TcpIp/TcpIpConfigSet/TcpIpSockConfig/TcpIpSockRxBufSize → SoAd/SoAdConfig/SoAdSocketConnectionGroup/SoAdSocketTpRxBufferMin
- Ressources for Tx and Rx buffer for TCP sockets are still located in the TcpIp module (container TcpIpTcpSocketBuffer). It must be ensured that enough buffers with corresponding sizes are configured for all sockets which are active at the same time at runtime. It must exist an exact value match of the following parameter set:
  - TcpIp/TcpIpConfigSet/TcpIpTcpSocketBuffer/TcpIpTcpSocketTxBufferSize ↔ SoAd/SoAdConfig/SoAdSocketConnectionGroup/SoAdSocketProtocol/SoAdSocketTcp/SoAdSocketTcpTxBufferMin
  - TcpIp/TcpIpConfigSet/TcpIpTcpSocketBuffer/TcpIpTcpSocketRxBufferSize ↔ SoAd/SoAdConfig/SoAdSocketConnectionGroup/SoAdSocketTpRxBufferMin

All other changes and adaptions are done by the configuration tool automatically or additional solving actions are provided to adapt the configuration manually.

12 SIP Update

If you received a newer version of your SIP, an update of your current project is required. Therefore open the project with the new version of the DaVinci Configurator Pro. The **DaVinciCFG.exe** is located within the folder *<UpdateSIP>\DaVinciConfigurator\Core*.

The DaVinci Configurator Pro identifies that the project was created with an older SIP and displays the following message. Select **Update the project by opening the project within the SIP of this DaVinci Configurator instance** and click **OK**.



The update will be performed. Wait until the DaVinci Developer project opens. Click **[Accept]** when the warning message appears. Click **[Import]** in the next dialog.

The Signal Import Mode dialog opens, select **Use import mode for all remaining objects** and click **[OK]**. The dialog proposing to **Save and close now** opens, confirm with **[Yes]**.

The DaVinci Developer project is closed and the update of the DaVinci Configurator Pro project continues. Once the update finishes click **[Ok]**.

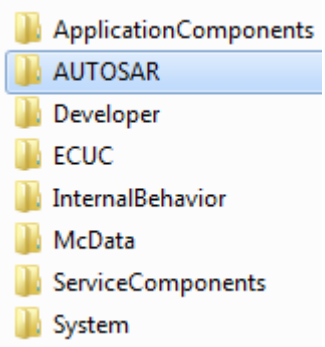
The project is now migrated to the new SIP.

13 Platform Types

From Release 12 on DaVinci Configurator and DaVinci Developer use a common definition of Platform Types as defined by AUTOSAR. These types are defined in the package defined by AUTOSAR: */AUTOSAR_Platform/*.

13.1 Location of the common definition

The **/Config** folder of the project contains a new folder named **/AUTOSAR**



This folder contains the file **PlatformTypes_AR4.arxml**, which defines the AUTOSAR Platform Types. This file is shared between DaVinci Configurator project and DaVinci Developer project. This file shall not be modified manually.

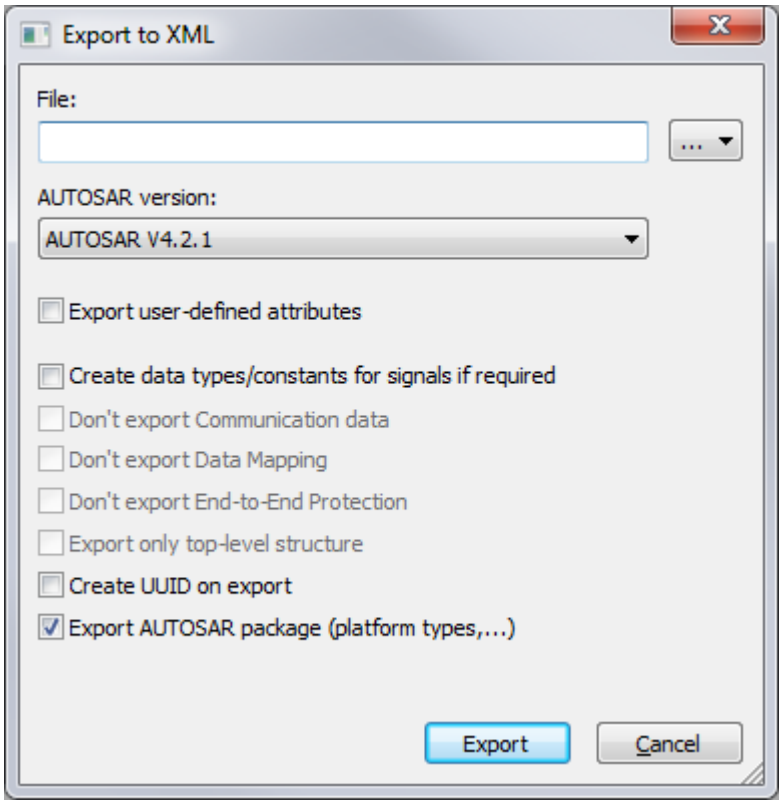
13.2 Import/Export behavior

DaVinci Configurator and DaVinci Developer provide the definition of the Platform Types as seen above. That means, the SWCs or Service Components do not have to provide this definition again, but shall only reference the types via the package defined by AUTOSAR (*/AUTOSAR_Platform/*).

When a file is imported in the project, the references to the provided Platform Types (via */AUTOSAR_Platform/* package) are automatically solved by DaVinci Configurator and DaVinci Developer.

In case an imported file contains the definition of a Platform Type already provided by DaVinci Configurator and DaVinci Developer, this definition is not imported, and the one provided by DaVinci Configurator and DaVinci Developer is used.

The **Export Editor** provides a new option **Export AUTOSAR package (platform types, ...)**



When this option is selected, the objects from the */AUTOSAR_Platform/* package are exported from the target file.

When this option is not selected, the objects from the */AUTOSAR_Platform/* package are not exported, only the references are exported.

13.3 Service components use the new Platform Types

In the SWC description of the Service Components generated by DaVinci Configurator, the new Platform Types are used instead of the ones generated in the previous SIP.

13.3.1 Old types

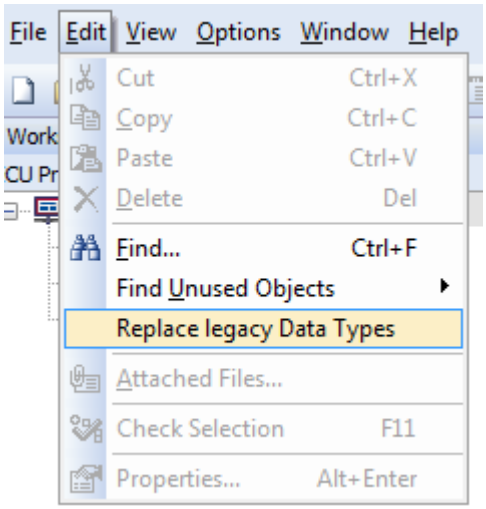
13.3.2 New types

13.4 Replacement of old developer standard types (Replace legacy Data Types)

Standard Types were provided in previous versions of DaVinci Developer projects under the */DataTypes/PlatformTypes/* package. These standard types are not provided anymore by DaVinci Developer, but they may remain in the project after migration to the new SIP.

An option is available in DaVinci Developer to replace references to these old types by references to the new provided Platform Types.

This option is available at **Edit | Replace legacy Data Types**



By clicking **Replace legacy Data Types**, the references are replaced.

This option is currently unavailable if the Object Browser is selected.

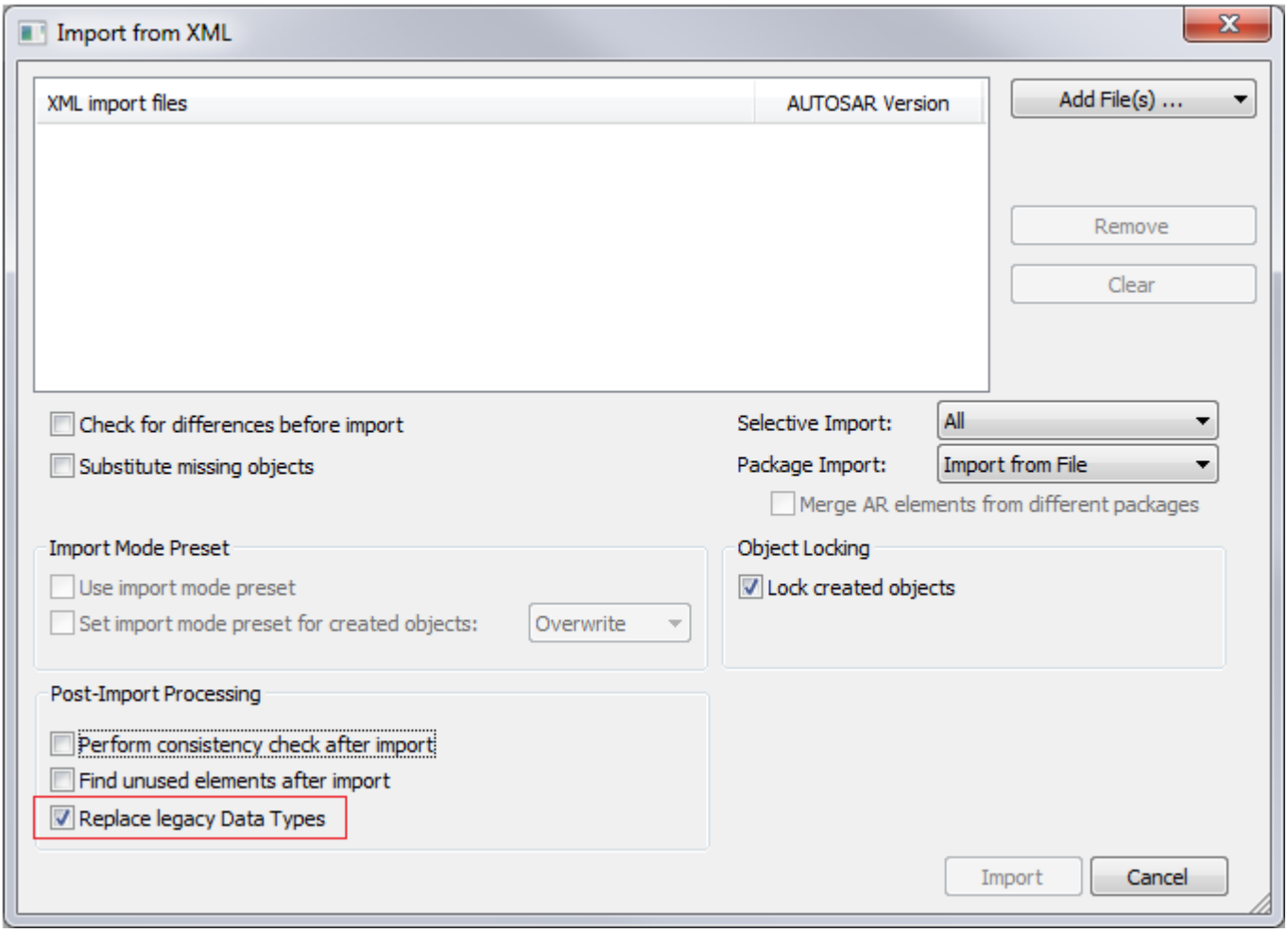
Note

The Data Types within the package /DataTypes/PlatformTypes/ are not removed from the Object Browser. This action can be performed manually.

Example with uint8

> References to */DataTypes/PlatformTypes/uint8* are replaced by */AUTOSAR_Platform/ImplementationDataTypes/uint8*
> */DataTypes/PlatformTypes/uint8* is kept in the project.

The same option is available when a file is imported in the project. The **Import from XML** editor provides the option **Replace legacy Data Types**"



When the box is checked, all references to the DataTypes within the */DataTypes/PlatformTypes/* package are replaced by the corresponding Platform Type from */AUTOSAR_Platform/* package.

14 MCAL Integration And Update

For new deliveries, MCALs are not longer part of the MICROSAR SIP delivery (exception: Vector is MCAL reseller).

Now you have to order the necessary MCALs directely from the Hardware Manufacturer.

The MICROSAR SIP includes a script, which will include the MCAL. You can find the **Script_MCAL_Prepare.bat** within the following path:
ThirdParty\<MCAL_µC>\VectorIntegration

Cross Reference

For detailed information see **TechnicalReference_3rdParty-MCAL-Integration.pdf**. (This document is only available in deliveries without MCAL)

Note

If you want to update your MCAL, you have to execute the **Script_MCAL_Prepare.bat** once again.

Check the delivered SIP, if **.\Doc\ReleaseNotes\ReleaseNotes_3rdPartyMCAL_VectorIntegration.pdf** exists. If the file exists, read the MCAL-specific information. If the file not exist ignore this section.

14 Multicore / Partition

Cross Reference

The document **UserManual_Multi-core.docm** in the folder **Doc|UserManual** of your delivery is a guideline and knowledge base which helps you during the project setup of a MICROSAR Multi-core project.

ReleaseNote_MICROSAR4_Vector_*.pdf - **FEAT-3063**

14 Reports

With **FEAT-3545** all reports are changed to support all browsers and the data is supported as HTML file.

This concerns the following reports

- **Update Report**
  Update your configuration via **Input Files** and see the result as HTML report stored in the **Log** folder of your project
- **Diff&Merge Report**
  Perform a diff & merge between two DPA files using **File|Start Comparison Mode...** and find the result as HTML report in the **Log** folder of your project.

IV Appendix

1 Frequently Asked Questions

You have a certain question? You just want to know how to do e.g. a certain setting without reading the whole document again? Then go on reading the following list and use the links to get at the place in the document where your question will be answered. This chapter will be extended continuously.

1.1 Problems with using two different DaVinci Configurator Versions on the same PC

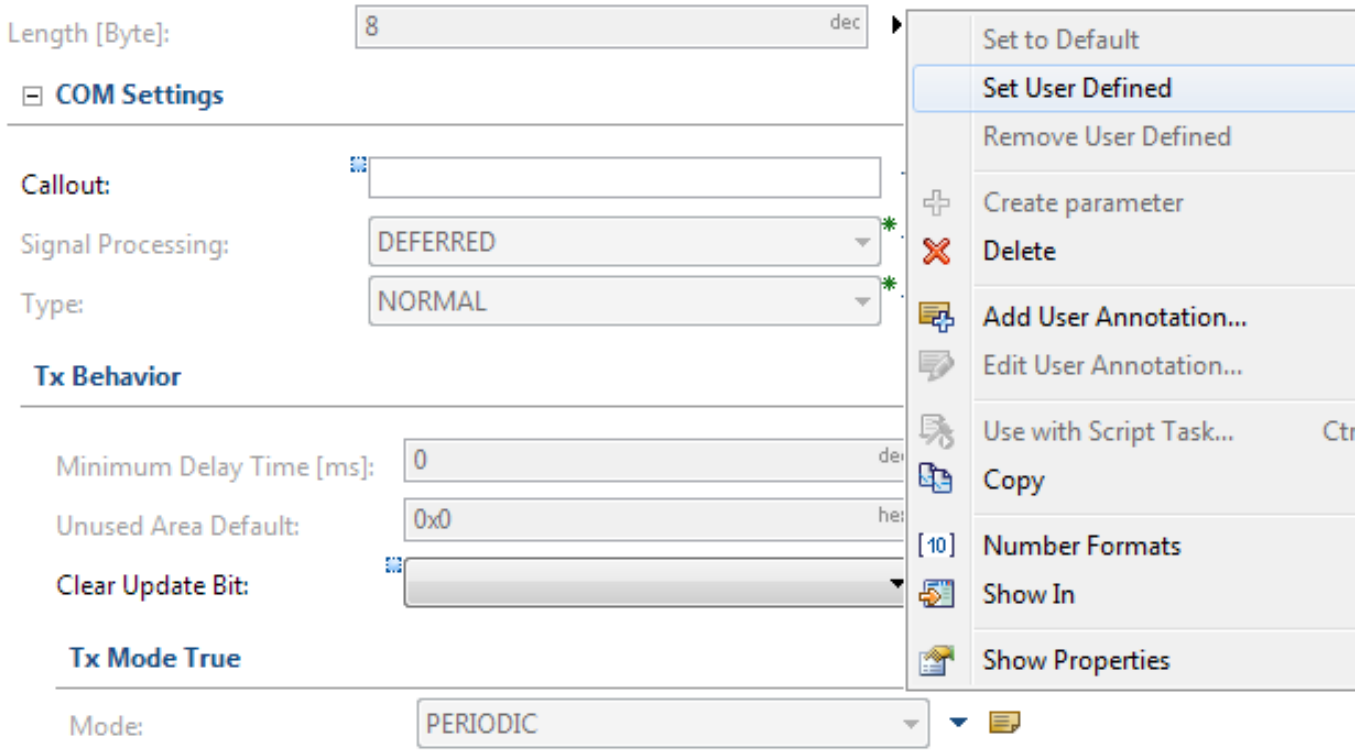
You get problems with using DaVinci Configurator versions 5.10.xx and DaVinci Configurator version 5.11.xx on the same PC?

Start your project with command line option **-clean** for solving Eclipse caching problems when DaVinci Configurator versions 5.10.xx and older are used on the same PC with DaVinci Configurator version 5.11.xx and younger.

1.2 Annotations for any parameter

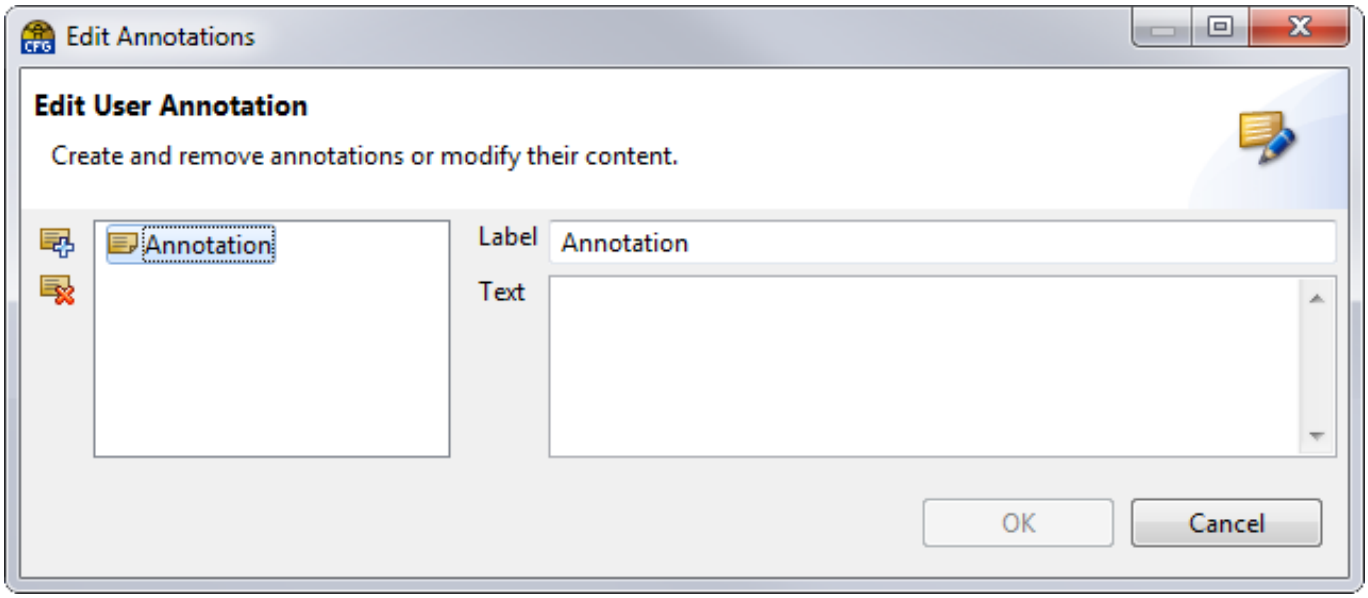
How to append an annotation to a parameter

For almost any parameter you can add individual annotations regardless whether you are in the Basic Editor or in the Configuration Editor. Open the context menu of a parameter by clicking the little right-pointing arrow and click **Add annotation**.



The annotation editor will open. You can enter one or more annotations each with a headline and floating text.

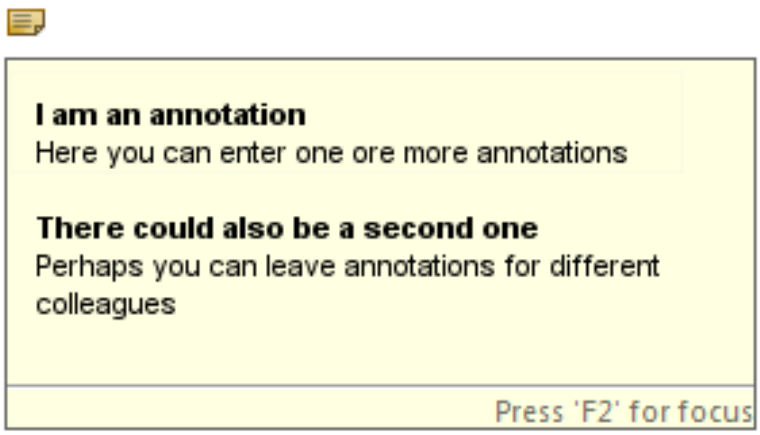
Via **Edit annotations** you can open and edit an already written annotation.



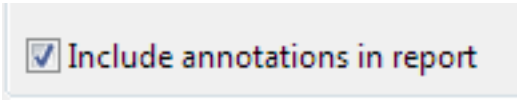
To read your annotations you can either open it via the editor or just by touching the little letter icon with the mouse, its content will be displayed like shown below as an example.

Note

It is also possible to use the **Properties** view. If an annotation exists, an additional tab will be shown to view this annotation.



Via a report (**Project|Report...**) it is possible to see all your annotations. Make sure to activate the checkbox **Include annotations in report**



1.3 Find Reference Container

1.3.1 How to find references using the [Find] dialog

Open the **Find** dialog, type in **value==""** and give a container name between the quotes.

2 History

What's new and what's changed?

This section explains the changes within this document from the previous version to the one mentioned in the headline.

Version 14.x.x

Author	Date
Klaus Emmert	2018-09-12

What's new?

- Update Configuration (SIP Update and Project Migration) - Migration steps from Release 20 to Release 21
- Direct link to TechRef folder from DaVinci Configurator Pro
- User defined parameters lead to warning in code generation window
- Multi-Edit feature in DaVinci Developer
- Information about MICROSAR Multi-Core
- For OEMs with preconfiguration, you can now select a Use Case
- Improved report files

What's changed?

- Various screenshots updated according to new tool appearance
- Layer illustrations updated

Version 13.x.x

Author	Date
Manuela Huber, Klaus Emmert	2018-03-28

What's new?

- Update Configuration (SIP Update and Project Migration) - Migration steps from Release 19 to Release 20

What's changed?

- Create new DaVinci Developer Workspace later(STEP1 Setup Your Project on page 24) when project is already set up without DaVinci Developer Workspace
- Completely new Input File Assistant
- Element Usage information in Select Reference Target window
- Filtering of Validation results
- Hint for Vector AUTOSAR XML Editor
- Terminate Port to intentionally leave ports unconnected

Version 12.x.x

Author	Date
Manuela Huber	2017-09-29

What's new?

- Update Configuration (SIP Update and Project Migration) - Migration steps from Release 18 to Release 19
- Add information for System Description Modification - see section STEP2 Define Project Settings on page 34.

What's changed?

- Minor changes, typos, etc.
- In case of the new DaVinci Developer 4.0, all DaVinci Developer sections are updated

3 Glossary

Adc
(Analogue Digital Converter Driver) The Adc driver abstracts hardware access to the analog-digital converter. For every input, the conversion parameters are configured (e.g. resolution, trigger source and trigger conditions).

Bfx
(Bitfield functions for fixed point) Library for bit handling in fixed point arithmetic functions.

BswM
(BSW Mode Manager) The BswM module contains vehicle mode management and application mode management. It processes mode requests from SWCs or other BSW modules and performs actions based on the arbitration, such as control of deadline monitoring, switching schedule tables, and handling of IPDU groups. In conjunction with the EcuM, the BswM module is responsible for starting up and shutting down the ECU. The BswM also coordinates the multi-core partitions.

Cal (Cpl)
(Crypto Abstraction Library) The Cal library offers SWCs and other BSW modules access to basic cryptographic functions. The individual cryptographic functions are implemented in software via the Cpl module. With AR 4.3 is set to obsolete.

Can
(CAN Driver) The Can driver abstracts access to the CAN hardware for sending and receiving messages and for switching between controller states (sleep, stop, etc.).

CanIf
(CAN Interface) The module CanIf offers abstracted (PDU-based) access to the CAN Driver. It controls the CAN Driver ((Can) as well as the transceiver driver ((CanTrcv).

CanNm
(CAN Network Management) Within a CAN network, the module CanNm is responsible for coordinated transitions between the wake up and sleep state.

CanSM
(CAN State Manager) The module CanSM is responsible for the bus-specific error handling.

CanTp
(CAN Transport Layer) The CanTp module conforms to ISO standard 15765-2. As the transport protocol for CAN, it is responsible for segmenting the data in the Tx direction, collecting data in the Rx direction and monitoring the data stream.

CanTrcv
(CAN Transceiver Driver) This driver is responsible for controlling the operating states of an external CAN transceiver. It contains control of wake up and sleep functions.

CanTSyn
(Time Synchronization over CAN) This module CanTSyn realizes the CAN specific time synchronization protocol. An access to the synchronized time base by the SWCs requires the

Synchronized Time-Base Manager ((StbM).

CanXcp
(CAN XCP-Module) The CanXcp module contains CAN-specific contents of the XCP module ((Xcp).

CDD
(Complex Drivers) The Complex Drivers are software modules that are not standardized by AUTOSAR. They have access to other BSW modules, the RTE and direct hardware access. For example, these modules are not standardized communication drivers for SPI or legacy software.

Com
(Communication) The Com module provides a signal-based data interface for the RTE. It places signals in messages and sends them according to the defined send type. The module contains various notification mechanisms for receiving signals. It also manages initial values, update bits and timeouts on the signal level. In multi-channel ECUs, the integrated signal gateway offers the ability to route signals between the communication buses.

ComM
(Communication Manager) The ComM module coordinates the communication channels and Partial Network Cluster with the communication requirements of the application.

ComXf
(COM Based Transformer) The module ComXf optimizes for signal groups the interaction between RTE and the COM module, because you can avoid the shadow buffer in the COM module. It de-/serializes the data identical to the COM module.

CorTst
(Core Test Driver) The CorTst module contains configuration and control of the test capabilities included in the microcontroller core. In addition, it offers a framework for extending these test capabilities. Specifically for safety-related software, the CorTst module tests critical units such as the ALU and the registers.

Crc
(CRC Routines) The Crc library calculates CRC checksums.

CryIf
(Crypto Interface) The module CryIf abstracts the access to the different crypto drivers.

Crypto
(Crypto Driver) The Crypto Driver offers standardized cryptographic primitives and a secure key- and certificate storage. The access to crypto hardware (HSM, SHE) or as software-based substitution is abstracted. It is possible to use multiple drivers with limited capabilities.

Crypto(Hw)
(Hardware-based Crypto Driver) This is the hardware-based abstraction on 3rd party crypto drivers (Crypto.

Crypto(Sw)
(Software-based Crypto Driver) This is the software-based abstraction of (Crypto.

CRYPTO(vHSM)
(CRYPTO Vector vHSM) This is the hardware-based abstraction of (Crypto of Vector's HSM solution (vHSM.

Csm
(Crypto Service Manager) The Csm module offers SWCs access to standardized cryptographic primitives and a secure key- and certificate storage. Each job can be prioritized and mapped to independent crypto drivers for parallel execution.

Dbg
(Debugging) The Dbg module enables external access to internal information of the basic software. It is also possible to modify memory data.

Dcm
(Diagnostic Communication Manager) The Dcm module implements diagnostic communication according to ISO 14229-1:2013 (UDS). Some diagnostic requests are processed directly in the Dcm (management of diagnostic sessions, reading of error codes, etc.) and some are routed to the SWCs via port interfaces (reading, writing and controlling of data elements within a data identifier, execution of routines, etc.). Legal requirements of OBDII / WWH-OBD are also supported.

Dem
(Diagnostic Event Manager) The Dem module implements a fault memory. The standardized interface for "DiagnosticMonitors" enables uniform development of manufacturer-independent SWCs. The Dem module is responsible for administering the DiagnosticTroubleCode states, environmental data, and for storing the data in NVRAM. The legal requirements of OBDII / SAE J1979 / WWH-OBD are also supported.

Det
(Default Error Tracer) The Det module supports error debugging during software development. It provides an interface for error notification, which is called by the individual BSW modules in case of error. Since R4.2 the Development Error Tracer is renamed to Default Error Tracer to cover additional error types i.e. Runtime- and Transient (hardware) faults.

Dio
(Digital Input Output Driver) The Dio Driver provides read and write services for the DIO channels (pins), DIO ports and DIO channel groups.

Dlt
(Diagnostic Log and Trace) The Dlt module provides generic "Logging and Tracing" functionality for SWCs and for the BSW modules (Rte, (Det and (Dem.

DoIP
(Diagnostics over IP) The DoIP module contains diagnostic functionality according to ISO 13400-2 like vehicle discovery. Up to and including AR 4.0.3, this functionality is part of the Socket Adaptor ((SoAd).

E2E
(End-to-End Communication Protection Library) Library for secure data exchange according to ISO 26262 for safety-related ECUs. It is responsible for calculating the checksum and providing the message counter.

E2EPW
(End-to-End Protection Wrapper) The E2EPW extends the RTE by adding verification of safety-related signals according to ISO 26262.

E2EXf
(E2E Transformer) With the module E2EXf, you integrate the protection of safety-relevant signals according to ISO 26262 in the RTE interface. In contrast to the E2E protection wrapper, the standard interfaces of the RTE are used.

Ea
(EEPROM Abstraction) The Ea module offers a hardware-independent interface for accessing EEPROM data and uses an EEPROM driver ((Eep) for this. In addition to reading, writing and clearing data, the Ea module also distributes write accesses to different areas of the EEPROM, so that all EEPROM cells are uniformly stressed, which increases their life-time.

EcuM
(ECU State Manager) The module EcuM is responsible for starting up and shutting down the ECU as well as waking it up. It is available in two variants: flexible and fixed. The EcuM Fix manages a number of defined, fixed operating states. Using the EcuM Flex, the user defines all operating states flexibly in the (BswM module. This lets you implement special energy-saving states and various types of startup. In a multi-core system, the EcuM coordinates the various cores.

Eep
(EEPROM Driver) The Eep driver enables hardware-independent access to internal respectively external EEPROM memory. It provides services for reading, writing and comparing data.

Efx
(Extended Fixed Point Routines) Library with extended mathematical functions for fixed-point values.

Eth
(Ethernet Driver) The Eth Driver abstracts access to the Ethernet hardware for sending and receiving data and for switching between controller states.

EthIf
(Ethernet Interface) The module EthIf enables bus-independent control of the Ethernet driver ((Eth) and Ethernet transceiver driver ((EthTrcv). Since AR 4.1, this module is also responsible for VLAN handling.

EthSM
(Ethernet State Manager) The module EthSM provides the Communication Manager ((ComM) with an abstract interface for starting up or shutting down communications in Ethernet clusters. EthSM accesses the Ethernet hardware via (EthIf.

EthSwt
(Ethernet Switch Driver) The module EthSwt provides a uniform and hardware independent interface for controlling and configuring Ethernet switches. It also coordinates the MAC learning when using multiple identical ECUs like cameras for the surround view.

EthTrcv
(Ethernet Transceiver Driver) EthTrcv offers a uniform and hardware-independent interface for driving multiple transceivers of the same kind. Configuration of EthTrcv is transceiver-specific and considers the properties of the physical network that is used.

EthTSyn
(Time Sync Over Ethernet) This module EthTSyn realizes the Ethernet specific time synchronization protocol and references to IEEE Standard 802.1AS (PTP). An access to the synchronized time base by the SWCs requires the Synchronized Time-Base Manager ((StbM).

EthXcp
(Ethernet XCP-Module) The EthXCP module contains Ethernet-specific contents of the XCP module ((Xcp).

Fee
(Flash EEPROM Emulation) The Fee module offers a hardware-independent interface for accessing flash data and uses a flash driver ((Fls) for this. In addition to reading, writing and clearing data, the Fee module also distributes write accesses to different areas of flash memory, so that all flash cells are uniformly stressed, which increases their life-time.

FiM
(Function Inhibition Manager) Based on the active errors managed by the (Dem module, the FiM offers the ability to prevent execution of functionalities in SWCs.

Fls
(Flash Driver) The Fls driver enables hardware-independent and uniform access to flash memory. It provides services for reading, writing and comparing data, and for deleting blocks (sectors).

FlsTst
(Flash Test) The module FlsTst offers algorithms for testing nonvolatile memory such as data or program flash, SRAM and protected cache.

Fr
(FlexRay Driver) The Fr driver abstracts access to the FlexRay hardware for sending and receiving data and for switching between controller states.

FrArTp
(FlexRay AUTOSAR Transport Layer) FrArTp is a FlexRay transport protocol. Based on ISO 15765-2 ((CanTp), it contains a frame compatibility with the CAN bus.

FrIf
(FlexRay Interface) The module FrIf offers abstracted (PDU-based) access to the FlexRay hardware. In addition, it offers support for synchronization with the global FlexRay time.

FrNm
(FlexRay Network Management) The module FrNm is responsible for network management in FlexRay. It synchronizes the transition to the bus sleep state.

FrSM
(FlexRay State Manager) The module FrSM controls and monitors the wake up and startup of nodes in the FlexRay cluster.

FrTp
(FlexRay ISO Transport Layer) FrTp is a FlexRay transport protocol and is based on the ISO 10681-2 standard.

FrTrcv
(FlexRay Transceiver Driver) The FrTrcv driver for an external FlexRay transceiver is responsible for switching the tranavsceiver on and off.

FrTSyn
(Time Sync Over FlexRay) The module FrTSyn realizes the FlexRay specific time synchronization protocol. An access to the synchronized time base by the SWCs requires the Synchronized Time-Base Manager ((StbM).

FrXcp
(FlexRay XCP-Module) The FrXcp module contains FlexRay-specific contents of the XCP module ((Xcp).

Gpt
(General Purpose Timer Driver) The Gpt driver provides an interface for accessing internal timers of the microcontroller. It is used for controlling periodic and singular events, for example.

Icu
(Input Capture Unit Driver) The Icu driver provides services for edge detection, measurement of periodic signals, assignment of edge time stamps and control of wake up interrupts.

Ifl
(Interpolation Floating Point) Library with interpolation functions for floating point values.

Ifx
(Interpolation Fixed Point) Library with interpolation functions for fixed point values.

IoHwAb
(I/O Hardware Abstraction) The module IoHwAb represents the connection between the RTE and the'ECU's I/O channels. It encapsulates access to the I/O drivers such as (Adc, (Dio or (Pwm and thereby makes the I/O signals of the ECU available to the SWCs.

IpduM
(I-PDU Multiplexer) The module IpduM supports multiple layouts which can be chosen by a selector field within an I-PDU. Alternatively it supports to dynamically combine PDUs into a Container PDU.

J1939Dcm
(SAE J1939 Diagnostic Communication Manager) The module J1939Dcm implements Diagnostic Messages of the SAE J1939-73 protocol, e.g. for reading out the fault memory.

J1939Nm
(SAE J1939 Network Management) J1939 supports adding ECUs to networks on-the-fly. The module J1939Nm is responsible for negotiating a unique ECU addres ("AddressClaim") and unlike other NM modules for handling the wake-up or going to sleep of the bus.

J1939Rm
(SAE J1939 Request Manager) The module J1939Rm implements requesting of data via Request Handling that is defined in the SAE J1939 protocol.

J1939Tp
(SAE J1939 Transport Layer) The J1939Tp module contains the transport protocols BAM (Broadcast Announce Message) and CMDT (Connection Mode Data Transfer) of the SAE J1939 standard.

LdCom
(Large Data COM) The LdCom module is optimized for routing large signals. Thereby it avoids an unnecessary copying of data. The LdCom is typically used together with (SomeIpXf as a serialization transformer.

Lin
(LIN Driver) The Lin driver provides services for initiating frame transmission (header, response, sleep-mode and wake up), and for receiving responses, checking the current state and validating wake up events.

LinIf
(LIN Interface) The module LinIf offers abstracted (PDU-based) access to the LIN hardware. It also handles schedule table processing and contains the LIN transport protocol ((vLINTP).

LinNm
(LIN Network Management) The LinNm module contains a hardware-independent protocol, which coordinates the transition between normal operation and the bus sleep mode of the LIN network.

LinSM
(LIN State Manager) The module LinSM switches between schedule tables and PDU groups in the (Com module and services the LIN interface with regard to sleep and wake up.

LinTrcv
(LIN Transceiver Driver) The LinTrcv module for an external LIN transceiver is responsible for monitoring and driving the wake up and sleep functions.

Mcu
(Micro Controller Unit Driver) The Mcu driver provides the services for: - A microcontroller reset triggered by software - Selecting microcontroller states (STOP, SLEEP, HALT, etc.) - Configuring wake up behavior - Managing the internal PLL clock unit - Initializing RAM areas with predefined values.

MemIf
(Memory Abstraction Interface) The module MemIf provides uniform access to the services of (Ea and (Fee. This lets you use multiple instances of these modules.

Mfl
(Mathematical Floating Point) Library with arithmetic functions for floating point values.

Mfx
(Mathematical Fixed Point) Library with arithmetic functions for fixed point values.

Nm
(Generic Network Management Interface) The Nm module offers a general and network-independent interface for accessing the bus-dependent network management modules ((CanNm, (LinNm, (UdpNm and (FrNm). In addition, the module handles synchronous, inter-network shutdown of the communication system in coordination with the other ECUs.

NvM
(Non Volatile RAM Manager) The NvM module manages, reads and writes data to a nonvolatile memory ((Ea or (Fee). At system start and at shutdown, it synchronizes the data in the RAM areas of the application. The module provides services such as saving of redundant blocks for a higher level of data protection. Since AR 4.0.3, the (RTE also provides a simpler and more flexible interface to Nv data (NvDataInterfaces).

Ocu
(Output Compare Unit Driver) The Ocu driver standardizes initialization and access to the Output Compare Unit.

Os
(Operating System) The module Os is the operating system of an AUTOSAR ECU. It is actually an extended OSEK operating system. Extensions are divided into so-called Scalability Classes (SC1-SC4). They cover the following functionalities: - SC1: schedule tables - SC2: timing protection + schedule tables - SC3: memory protection + schedule tables - SC4: memory protection + timing protection + schedule tables In safety-related ECUs, the OS is one of the safety-related modules. The operating system also supports (asymmetric) multi-core microcontrollers.

PduR
(PDU Router) The module PduR handles distribution of the communication packets (PDUs) between the bus systems and the various BSW modules. In addition it offers gateway mechanisms for routing IF-PDUs (last-is-best or FIFO) and TP-PDUs (last-is-best or on-the-fly) between the bus systems.

Port
(Port Driver) The Port driver is responsible for initialization of the entire port structure of the microcontroller.

Pwm
(Pulse Width Modulation Driver) The Pwm driver provides services for initialization and control of the PWM (pulse-width modulation) channels of the microcontroller.

RamTst
(Ram Test) The module RamTst tests internal microcontroller RAM cells. A complete test is triggered during startup and shutdown of the ECU or by a diagnostic command. During normal operation, a periodic test is performed (block-by-block or cell-by-cell).

Rte
(Runtime Environment) The Rte implements the Virtual Functional Bus and the execution of the SWCs. It also ensures consistent data exchange between the SWCs themselves and between the SWCs and the basic software. The execution of the basic software is realized by the integrated (SchM. The Rte also supports communication beyond partition boundaries (multi-core/trusted/untrusted). In addition, it offers simplified access to NVRAM data and calibration data. In safety-related ECUs, the RTE is one of the safety-related modules.

SchM
(BSW Scheduler) The SchM module is integrated in the (Rte, calls the periodic "main" function of the individual BSW modules, and provides functions for critical sections. For the distribution of the BSW (master satellite concept) via partitions and core boundaries, the SchM provides nearly identical communication interfaces like the (Rte.

Sd
(Service Discovery) The Sd module realizes the SOME/IP SD protocol. An ECU communicates the availability of its services to communication partners via a publish-subscribe protocol. In addition, ECUs can register to receive automatic notifications, e.g. on a signal update.

SecOC
(Secure Onboard Communication) The SecOC module is used to send or receive authenticated messages. Unauthorized, repeated or manipulated messages are detected. The SecOC is part of the AUTOSAR security solution.

SoAd
(Socket Adaptor) The module SoAd converts the communication via PDUs as it is defined in AUTOSAR into a socket-based communication. In AR 4.0, the SoAd also contains the diagnostic functionality defined in ISO 13400-2 ((DoIP). Since AR 4.1, this plug-in is removed and specified as a stand-alone module ((DoIP).

SomeIpTp
(SOME/IP Transport Protocol) The module SomeIpTp extend SOME/IP, so it is possible to transfer a larger amount of data (>1500 Byte) per UDP.

SomeIpXf
(SOME/IP Transformer) The module SomeIpXf is an RPC and serialization protocol. It is used to call methods on other ECUs, which for example were made known in the system beforehand via Service Discovery ((Sd).

Spi
(SPI Handler/ Driver) The Spi Driver handles data exchange over the Serial Peripheral Interface. It is primarily used in conjunction with external hardware such as EEPROM, Watchdog, etc.

StbM
(Synchronized Time-Base Manager) The module StbM enables time synchronization. Since AR 4.2 a time base prescribed by the Time Master can be synchronized with other ECUs via bus systems.

TcpIp
(TCP/IP Stack) The TcpIp module contains all protocols for UDP and TCP-based communication. It supports the versions IPv4 and Ipv6 as well as parallel operation of Ipv4 and Ipv6 in one ECU. It contains the following protocols: - Ipv4, ICMPv4 and ARP - Ipv6, ICMPv6 and NDP - UDP, TCP, DHCPv4 (client) and DHCPv6 (client)

Tm
(Time Services) The Tm module is used for such tasks as measuring execution DDmplementing active waiting. It offers a resolution from 1µs to 4.9 days.

Ttcan
(TTCAN Driver) The Ttcan driver offers the same functionality for a TTCAN controller (ISO 11898-4) as the CAN Driver ((Can) does for a CAN controller.

TtcanIf
(TTCAN Interface) The module TtcanIf offers the same functionality for a TTCAN controller (ISO 11898-4) as the CAN Interface ((CanIf) does for a CAN controller.

UdpNm
(UDP Network Management) The module UdpNm implements synchronous transition to sleep mode for Ethernet ECUs.

V2xBtp
(Vehicle-2-X Basic Transport) The module V2xBtp is part of the ETSI standard and required for the European market. The Vehicle-2-X Basic Transport module enables protocol entities at the Facilities to access services (data and control information) of the GeoNetworking protocol and to pass protocol control information between the Facilities and the GeoNetworking protocol.

V2xFac
(Vehicle-2-X Facilities) The module V2xFac is part of the ETSI standard and required for the European market. The Vehicle-2-X Facilities module provides the - basic services (BS), - Cooperative Awareness (CA), - Decentralized Environmental Notification (DEN).

V2xGn
(Vehicle-2-X GeoNetworking) The module V2xGn is part of the ETSI standard of ITS and required for the European market. The V2xGn module is responsible for the routing and forwarding of V2X messages.

V2xM
(Vehicle-2-X Management) The module V2xM is part of the ETSI standard of ITS and required for the European market. The V2xM module is responsible for: - Position and Time management (POTI), - Identity / Security Taking care on pseudonym certificates, - Decentralized Congestion Control (DCC) - Adjustment of communication due to environmental conditions (e.g. lower transmission frequency for traffic congestion).

vAVTP
(Vector Audio/Video Transport Protocol) The module vAVTP is specified in IEEE 1722. In AVB networks it is responsible for the transport of audio/video data, including the Presentation Time.

vCRY (HW)
(Vector Secure Hardware Extension) The CRY (HW) driver abstracts the access to the cryptographic hardware functions of a Secure Hardware Extension (SHE). This functionality is used by the (Csm module, for instance.

vDBG
(Vector Debugging (via XCP)) The module vDGB enables external access via XCP to internal information of the basic software. It is also possible to modify memory data.

vDES
(Vector Diagnostic Event Synchronizer) The module vDES enables diagnostic monitoring over multiple MCUs and implicates multi-controller DEM functionality. The diagnostic master collects qualified events communicated from the diagnostic slaves, which receive event messages locally.

vDIOHWAB
(Vector Digital Input Output Hardware Abstraction) The module vDIOHWAB offers a quick and easy access to the (Dio signals from the MCAL.

vDNS
(Vector Domain Name System Resolver) The module vDNS contains a DNS resolver. It is responsible for resolving a domain, e.g. vector.com, into a valid IP address.

vDRM
(Vector Diagnostic Request Manager) The vDRM module is an onboard diagnostic client. It allows to sends diagnostic requests to the same or other ECUs and receives their response messages. Additional features like managing parallel connections, identifying other ECUs in the vehicle and a firewall to block potentially dangerous diagnostic requests are included.

vETHFW
(Vector Ethernet Firewall) The module vETHFW is a firewall blocking unwanted traffic which is received or transmitted in order to increase the security of the entire network.

vETM
(Vector Ethernet Testability Module) The module vETM implements a standardized counterpart for protocol conformity tests. The module enables an externally connected test environment to trigger defined actions, e.g. sending UDP packages or creating a TCP-connection. The Vector solution based on AUTOSAR Acceptance Tests "Specification of Testability Protocol and Service Primitives".

vEXI
(Vector Efficient XML Interchange) The vEXI module is used to interpret XML document and to convert them to a binary format. This makes it more efficient to process the files and to transmit them, in order to economize on communication bandwidth. It is used in the Vehicle2Grid environment.

vHSM
(Vector Hardware Security Module) The vHSM is running in the secure core of the MCU. The offered cryptographic primitives and a secure key- and certificate storage capabilities can be adapted to the individual needs (e.g. like the usage of hardware accelerators for dedicated cryptographic primitives).

vHTTP
(Vector Hypertext Transfer Protocol) The module vHTTP, for instance, routes browser requests to a server. The module contains a HTTP client. It is used in the Vehicle2Grid environment.

vIICDRV
(Vector I²C Driver) The vIICDRV driver provides services for communication with external I2C chips.

vLINTP
(LIN Transport Layer) The module vLINTP is responsible for segmenting data in the Tx direction, collecting data in the Rx direction and monitoring the data stream. According to the AUTOSAR specification, LinTp is part of (LinIf.

vLINXCP
(Vector LIN XCP-Module) The vLINXCP module contains LIN-specific contents of the XCP module ((Xcp).

vPSI5
(Vector Peripheral Sensor Interface 5) The vPSI5 module realizes the "Peripheral Sensor Interface 5" (PSI5) protocol. This is an open standard for bidirectional, digital sensor data communication in vehicles.

vPTP
(Vector Precision Time Protocol) The Precision Time Protocol (IEEE 802.1AS) is in AUTOSAR realized as EthTSyn and StbM module.

vRTM
(Vector Runtime Measurement) You use the vRTMmodule to determine the runtime and CPU load of BSW modules and of application code. The module is typically used during development.

vRTP
(Vector Real-Time Transport Protocol) The module vRTP is a real time capable streaming protocol with uni-/multicast support. The RealTime Control Protocol (RTCP) for negotiation and observation of Quality-of-Service-Parameter (QoS) is also included. The following profiles are supported: - IETF RFC 6184 RTP Payload Format for H.264 Video - IEEE 1733 Layer 3 Transport Protocol for Time-Sensitive Applications in Local Area Networks.

vSBC
(Vector System Basis Chip) The vSBC module provides an abstraction layer for an externally connected System Basis Chip. The module provides a hardware independent interface that can be used by upper layers such as (CanTrcv, (LinTrcv or (Wdg to control the SBC hardware.

vSCC
(Vector Smart Charging Communication) The module vSCC is responsible for smart charging communication according to ISO 15118 or DIN SPEC 70121. It supports DC and AC charging and the associated profiles Plug and Charge (PnC) and External Identification Means (EIM).

vSENT
(Vector Single Edge Nibble Transmission) The vSENT module allows access to sensors which are implemented according to the SENT protocol based on SAE J2716. The driver is hardware independent through the interface connection with the (Icu module. It offers the following functionalities: - Realization of SENT master node functionality, - Multiple parallel SENT interfaces/channels, - Slow- and fast channel, - Enhanced serial message format on slow channel.

vSRP
(Vector Stream Registration Protocol) The module vSRP enables the registration of data streams with identification and access control, based on IEEE 802.1Qat for end stations.

vTLS
(Vector Transport Layer SecurityClient) The vTLS client encrypts TCP-based communication. The encryption algorithm can be selected.

vXMLSecurity
(Vector XML Security) The module vXMLSecurity is used to generate and validate XML signatures which are attached to EXI-encoded data and based on the W3C XML security standard. It is used in the Vehicle2Grid environment.

Wdg
(Watchdog Driver) The Wdg module provides services for controlling and triggering the watchdog hardware. The trigger routine is called by the Watchdog Interface ((WdgIf). For safety-related ECUs, the Wdg module must be developed according to ISO 26262.

WdgIf
(Watchdog Interface) The module WdgIf enables uniform access to services of the Watchdog Driver ((Wdg), such as mode switching and triggering. For safety-related ECUs, the WdgIf module must be developed according to ISO 26262.

WdgM
(Watchdog Manager) The module WdgM monitors the reliability and functional safety of the applications in an ECU. This includes monitoring for correct execution of SWCs and BSW modules and triggering of the watchdogs at the required time intervals. The WdgM module reacts to potential faulty behavior with multiple escalation stages. An important fact for safety-related functions according to ISO 26262 is the monitoring of correct flow sequences of critical tasks (logical supervision). For safety-related ECUs, the WdgM must be developed according to ISO 26262.

WEth
(Wireless Ethernet Driver) The WEth driver abstracts access to the Wireless Ethernet hardware for sending and receiving data.

WEthTrcv

(Wireless Ethernet Transceiver Driver) The WEthTrcv driver offers a uniform and hardware-independent interface for driving multiple transceivers of the same kind. Multiple radios with individual configurations are also supported.

Xcp

(Universal Measurement and Calibration Protocol) Xcp is a protocol for communication between a master (PC tool) and a slave (ECU). It is standardized by ASAM and is used primarily for measuring, calibrating, flashing and testing ECUs. XCP supports the bus systems CAN ((CanXcp), FlexRay ((FrXcp), Ethernet ((EthXcp) and LIN ((vLINXCP).

4 Index

Get more Information!

Visit our Website for:

- News
- Products
- Demo Software
- Support
- Training Classes
- Addresses

www.vector.com