VECTOR

Working in Interaction with DaVinci Configurator Classic 6 and DaVinci Developer Classic

User Manual

V1.00.00 | 2025-12-12

Document Information

History

Author	Date	Version	Remarks
Andreas Lachenschmidt Chris Pingel	2025-12-12	V1.00.00	Initial version

Contents

Illustrations

1 About this Document

This document describes the specific features of DaVinci Developer Classic for the interaction with DaVinci Configurator Classic 6. It focuses on workflows of dedicated use cases and software design aspects.

2 Introduction

The workflow defined for DaVinci Configurator Classic 6 will differ from the existing one in DaVinci Configurator Classic 5.

The important differences for working in interaction with DaVinci Developer Classic are about how the workflow steps are executed. To be precise, all workflow steps will be available to be executed detached, and the arrangement of the workflow steps is more aligned with the goal to avoid circular dependencies and working in an application-centric workflow.

Furthermore, DaVinci Configurator Classic 6 does not call DaVinci Developer Classic functionality. This significantly enhances flexibility and transparency in the execution of tool functionalities. In DaVinci projects with DaVinci Configurator Classic 5, project tasks included calls to DaVinci Developer Classic functionality, e.g. during input file processing or project creation process. Since this caused a lot of interaction in the background not seen by the user and inflexible workflow step execution, the cross-tool dependencies are removed.

The effects will be considered in more detail in subchapters of this document.

Additionally, the use cases are assigned slightly different between the tools. DaVinci Configurator Classic 6 focuses on supporting ECUC modeling and the associated service components. Any further modeling activities related to Software Component (short: SWC) design are in responsibility of DaVinci Developer Classic. The impact of this change will be particularly noticeable in the distribution of editing capabilities within SWC design, creating a clearer separation in the development journey of SWC design and Basic Software (short: BSW). Additional details are provided in the following subchapters.

Reference
Please find more details about concepts of DaVinci Configurator Classic 6 in our online help.

To provide a fast overview of changes in the way of creating SWC design with DaVinci Developer Classic in combination with DaVinci Configurator Classic 6, chapter 3 describes enhancements in modeling functionality. The intended changes for the interaction with DaVinci Configurator Classic 6 are described in chapter 4. Chapter 5 presents a selection of interaction workflows underlined with examples in a step-by-step description. Further information about additional topics is available in chapter 6.

3 Modeling Functionality Consolidated in DaVinci Developer Classic

3.1 General Interaction

Due to the distinct use cases of DaVinci Developer Classic and DaVinci Configurator Classic 6, there is now a clearer separation between SWC Design and BSW Config. That means, SWC Design modeling shall not be done in DaVinci Configurator Classic 6 anymore, and its responsibility is taken over by DaVinci Developer Classic.

However, DaVinci Configurator Classic 6 provides possibilities to create service software components and corresponding connections in GUI, as well as extensive possibilities in SWC Design via Automation Interface. Since some model parts are highly dependent on the BSW configuration (e.g. service components) it will not be moved completely into DaVinci Developer Classic.

Since both tools can provide modeling approaches to the user, it is important to think about file handling and file priorities. Therefore, the following conditions will apply:

- SWC Design created within DaVinci Configurator Classic 6 will be read-only in DaVinci Developer Classic
- SWC Design created within DaVinci Developer Classic will be read-only in DaVinci Configurator Classic 6

These principles are realized on file level where both tools have their own files which are known by the other tool.

Furthermore, it is important to shortly mention that both tools are now working on the same model level. That means, DaVinci Developer Classic and DaVinci Configurator Classic 6 are modeling on the Structured Extract. In contrast, the structured model (containing composition hierarchy) was flattened out by DaVinci Configurator Classic 5, and all modifications made within DaVinci Configurator Classic 5 have been populated on the Flat Extract without being noticed by DaVinci Developer Classic.

Since both tools are working on the same model level, new possibilities arise in DaVinci Developer Classic which will be explained in the next chapters.

3.2 Service Components

Previously, service components created in DaVinci Configurator Classic 5 were already visible in DaVinci Developer Classic. With the transition to the structured-extract-based configuration, service component prototypes and service connectors created in DaVinci Configurator Classic 6 are now also displayed in DaVinci Developer Classic. This enhancement is based on the previously mentioned product scope and thus provides a more comprehensive view of the SWC Design within DaVinci Developer Classic.

3.2.1 Service Component Prototypes and Connections

The service connectors and service component prototypes are shown in a new view, called **Service Connectors View**. This is a new view with editing possibilities that allows you to see already existing component prototypes and connections and create new ones.

It can be opened via ribbon menu “Service Connectors”.

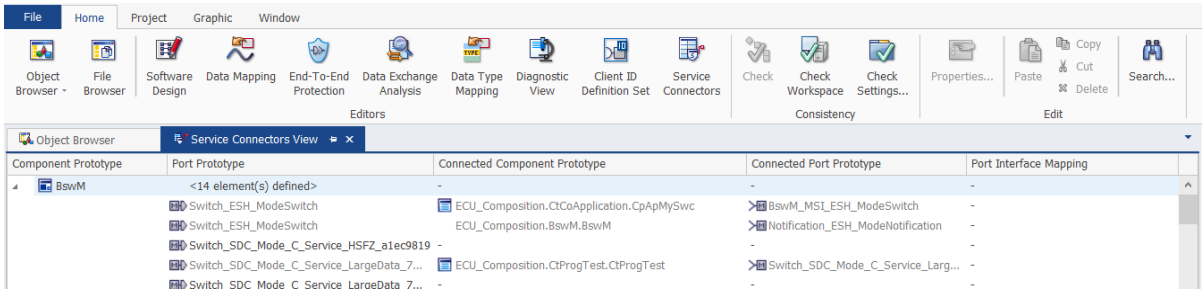


Figure 3-1: Service Connectors View

From this view, the context menu provides options to create new service component prototypes and service connectors.

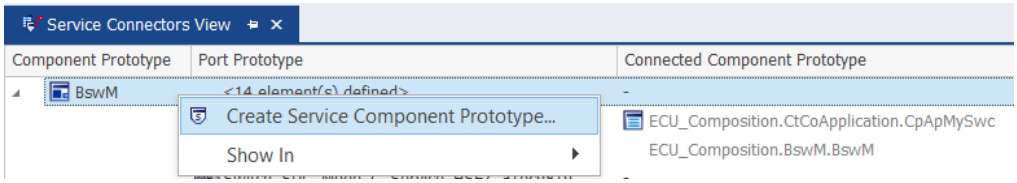


Figure 3-2: Create Service Component Prototype

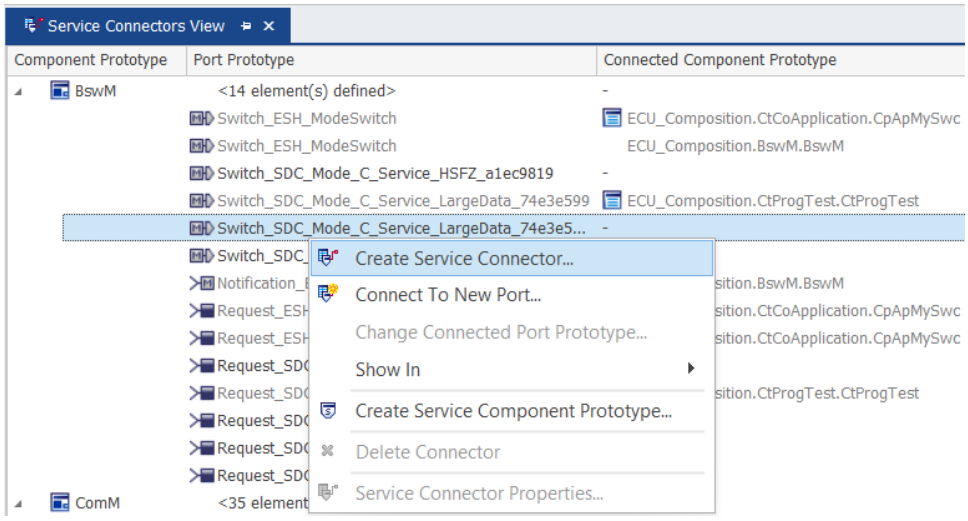


Figure 3-3: Create Service Connector

Within the dialog **Create Service Connecto**r, a port prototype of the entire software design (across hierarchies) can be selected to be connected to the context port. The selection includes two parts: **Component Prototypes** and **Compatible Port Prototypes**. A port is compatible if it has the opposite port direction to the context port.

In addition, the dialog provides two options that are intended to make it much easier to find the desired port to be connected:

- **Show compatible ports only**: It includes further compatibility checks in addition to the port direction.
- **Show unconnected ports only**: It includes all ports that are not connected to any other port.

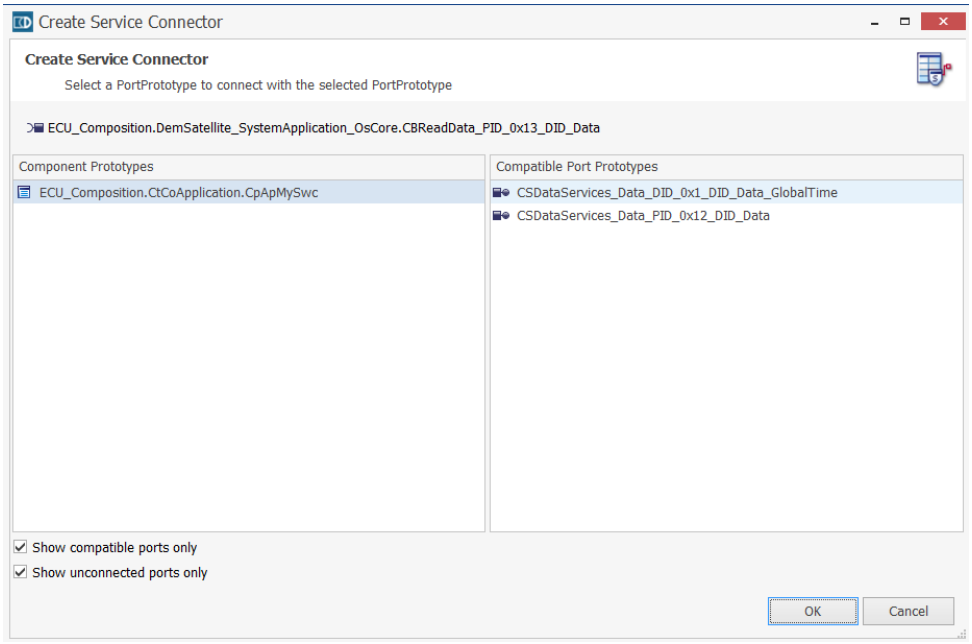


Figure 3-4: Create Service Connector dialog

The adaptation of existing connections, or even deletion is also possible but only for the once created in DaVinci Developer Classic.

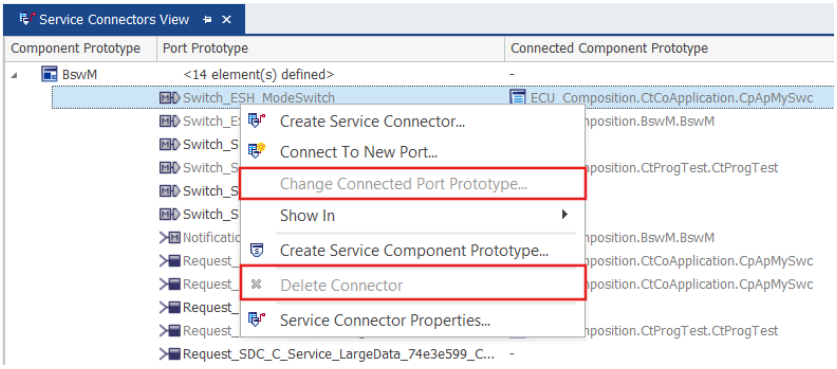


Figure 3-5: Modify or Delete Service Connector

Note
The connections that cannot be edited in Service Connectors View are grayed out. This could be the case if connections have been loaded from files managed by DaVinci Configurator Classic 6.

The service components and connections created within DaVinci Developer Classic will be loaded by DaVinci Configurator Classic 6.

Note
The integration of service component prototypes and service connectors into the graphical Software Design is not supported yet.

3.2.2 Connect to New Service Port

Within the Service Connectors View, it is also possible to create service ports and server runnables based on ports of service components. This introduces the functionality known as **Connect to New Port…** from DaVinci Configurator Classic 5, now available in DaVinci Developer Classic.

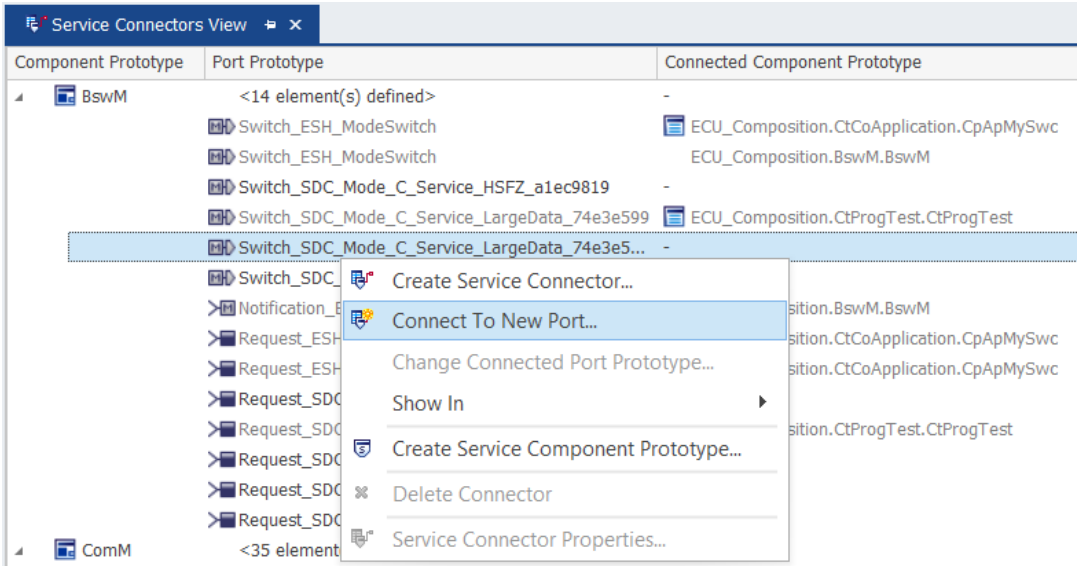


Figure 3-6: Connect To New Port

The **Connect To New Port…** option is available in the context menu of any port within Service Connectors View. Selecting this option opens **Create New Service Port** dialog.

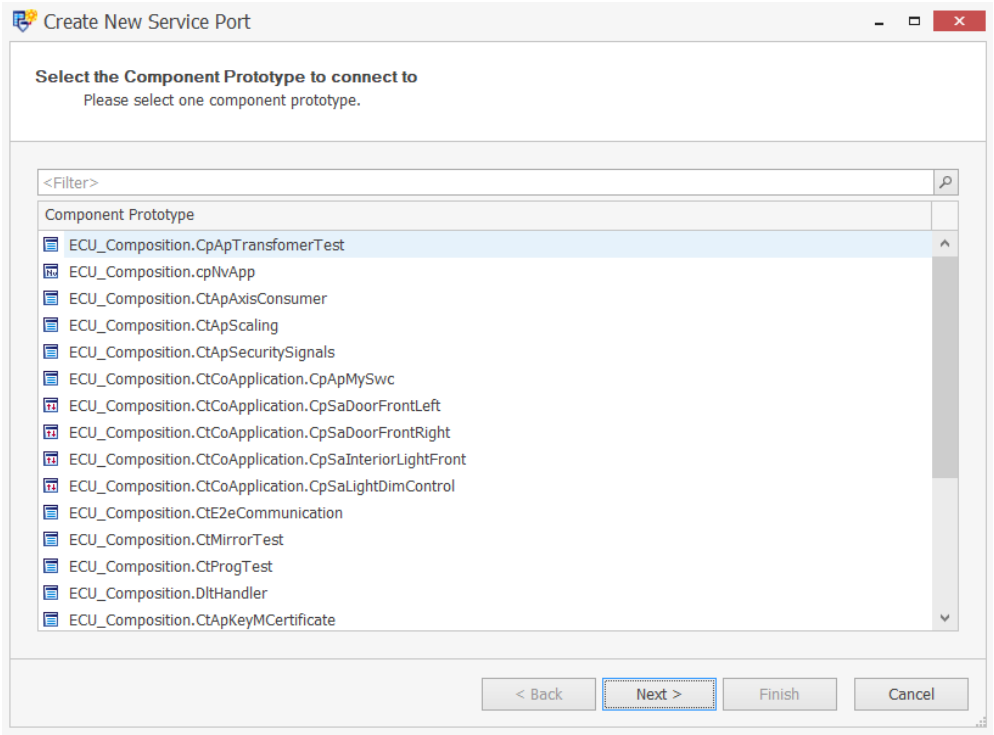


Figure 3-7: Create New Service Port dialog - Select the Component Prototype to connect to

The first page of the dialog displays all available software component prototypes. Choose the desired prototype to which the new port will be added, then click **[Next]**.

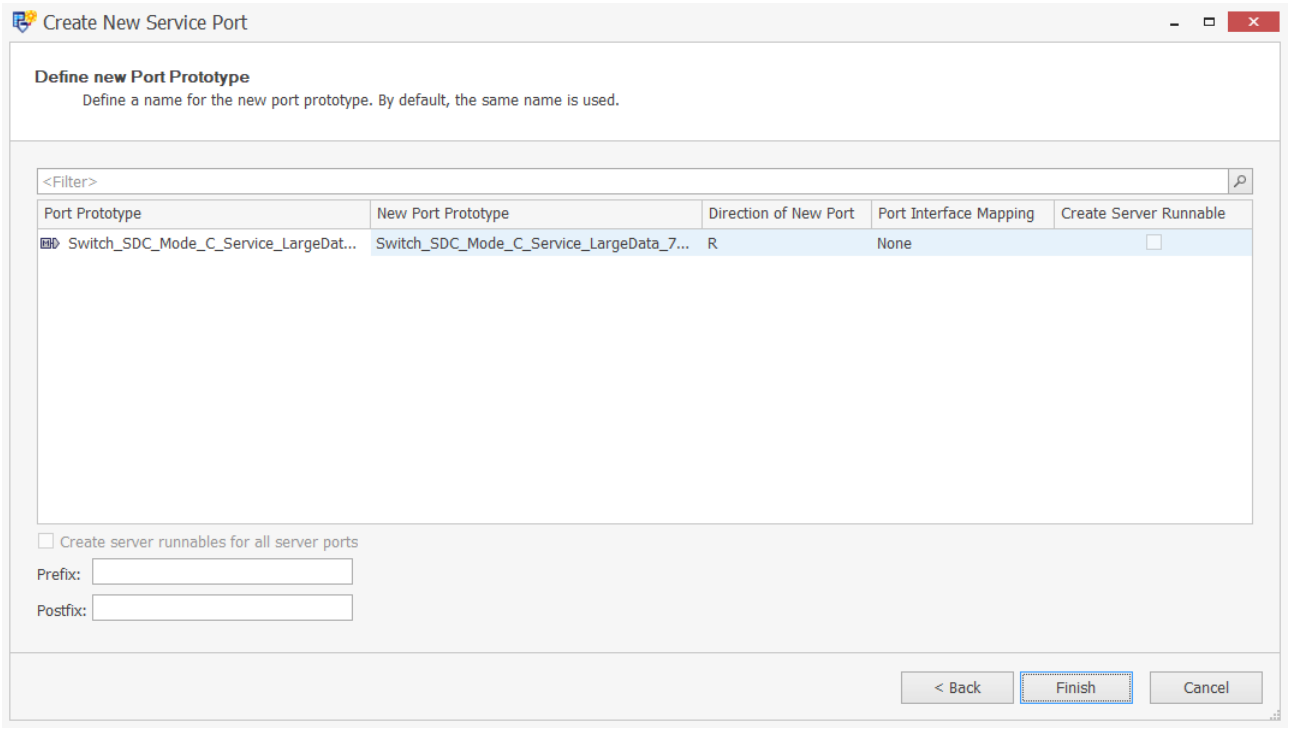


Figure 3-8: Create New Service Port dialog - Define new Port Prototype

The second page shows the details of the port to be created, organized in the following columns:

- **Port Prototype**: context port selected in the Service Connectors View
- **New Port Prototype**: name of the newly created port
- **Direction of New Port**: communication direction of the new port
- **Port Interface Mapping**: name of the selected port interface mapping for the port connection
- **Create Server Runnable**: option to create a sever runnable in addition to server port

All columns except **Port Prototype** can be edited.

Note
The **Create Server Runnable** option is only enabled if the new port is a server port (client-server communication).
When enabled, you can define a prefix and postfix for the server runnable.

Click **[Finish]** to create the new port prototype (and the server runnable, if selected) within the chosen software component prototype.

3.2.3 Auto-Connection of Service Ports

In certain scenarios, it may be necessary to create many service connections at once. The Service Connectors View supports this through manual creation, while DaVinci Configurator Classic 6 offers an automated approach. Please refer to the DaVinci Configurator Classic 6 documentation for more details.

Note
The **Auto-Connect Port Prototypes** feature within DaVinci Developer Classic does not include service connections.

3.2.4 Service Connections in Software Design Editor

Although service components appear in the **Connector Prototype Lists** of the **Software Design** editor, this view is not intended for managing or reviewing service connections. Service connections displayed here are limited to those created on the same hierarchical level while abstract connections between ports across different component hierarchies are not shown.

Furthermore, service components and their connections are not part of the graphical representation in the Software Design editor.

3.3 Port Interface Mapping

Port interface mappings have long been supported by DaVinci Developer Classic; however, those created in DaVinci Configurator Classic 5 were not previously loaded into the workspace of the DaVinci Developer Classic. With the recent changes, port interface mappings associated with service connectors are now displayed, providing improved visibility and consistency.

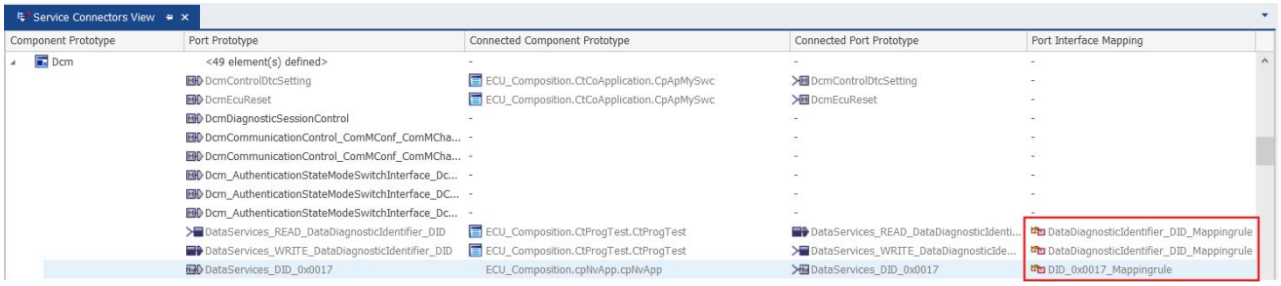


Figure 3-9: Port Interface Mapping in Service Connectors View

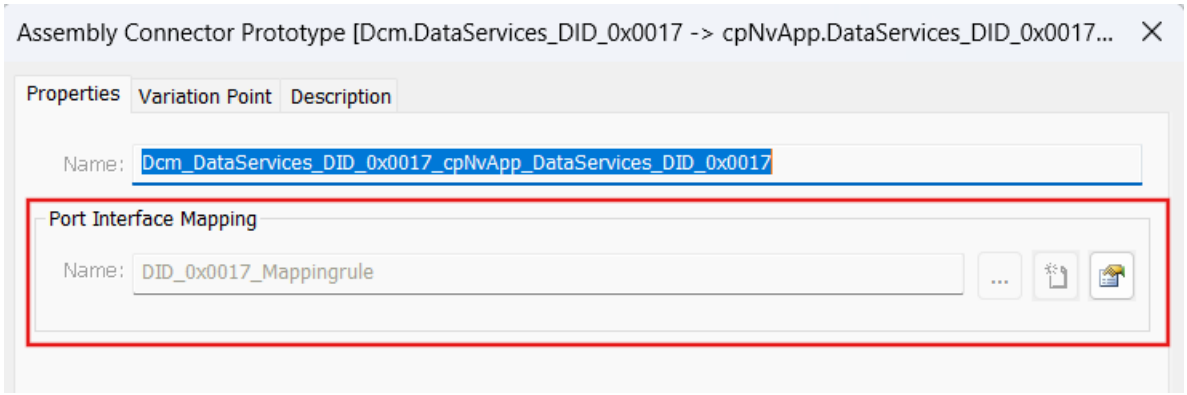


Figure 3-10: Port Interface Mapping in Service Connectors Properties

3.4 Port Termination

In the interaction between DaVinci Developer Classic and DaVinci Configurator Classic 5, both tools use different approaches to model port termination. These modeling methods will

continue to be supported, and DaVinci Configurator Classic 6 together with DaVinci Developer Classic will be able to read both formats. This ensures improved compatibility and enables you to see terminated ports from DaVinci Configurator Classic 6 in DaVinci Developer Classic workspace, and vice versa.

4 Supported Use Cases with DaVinci Configurator Classic 6

Note
Developer Classic supports complex use cases via commands an a Command Line Interface (CLI). The CLI tools are installed in DaVinci Developer Classic installation path in the "./Bin" subfolder.

4.1 Workspace Creation

For creating a DaVinci Developer Classic workspace the following CLI command can be used:

```
dvdevc project create --workspace-path <path of workspace
file>.dcf --model-version <model version> --autosar-version
<AUTOSAR version>[--davinci-project] <path to davinci project
file>.dvjson [--create-root-design]
```

- `project create`: creates workspace, if not existing
- `--workspace-path`: relative or absolute path to workspace file (command will create folders if they are not existing)
- `--model-version`: Model version in format <MAJOR>.<MINOR>
- `--autosar-version`: AUTOSAR version in format <YEAR>-<MONTH>

Options:

- `--davinci-project`: relative or absolute path to a DaVinci project file of DaVinci Configurator Classic 6 (".dvjson"; it will be checked by the tool if the given target project exists). The reference point for a relative path is the location of the workspace file. See 6.1 for further details about the path handling on the command line interface.
- `--create-root-design`: generate a Root Design consisting of the System and the Root Composition (see 4.3 Root Design Creation for more details)

The workspace will be created by the tool as "`<path of workspace file>.dcf`" with the specified AUTOSAR version and DCF model version. The subcommand will print "`Workspace created successfully: <path of workspace file>.dcf`" if the workspace was created successfully. In case of errors the following messages will be printed:

- "`Could not modify the workspace <path of workspace file>.dcf – Reason: <error code from system>`" → return code ACCNOK will be set
- "`Workspace <path of workspace file>.dcf not found`" → return code WSNOK will be set

Possible Return codes are:

- OK (0): workspace creation successful
- ACCNOK (3): could not create missing parts of the path (missing rights, maximum path length exceeded etc.)
- WSNOK (7): workspace not existing

Help content can be displayed by entering `dvdevc project create -h` or `dvdevc project create –help`.

4.2 Project Link

There is also the possibility to link a DaVinci Developer Classic workspace to a DaVinci Configurator Classic 6 project after a workspace creation via Command Line Interface:

```
dvdevc project link --workspace-path <path of workspace file>.dcf --davinci-project <path to davinci project file>.dvjson [--create-root-design]
```

- `project link`: links workspace to a DaVinci Configurator Classic 6 project, if the given workspace exists
- `--workspace-path`: path to workspace file
- `--davinci-project`: relative or absolute path to DaVinci project file of DaVinci Configurator Classic 6 (it will be checked by the tool if target project exists). The reference point for a relative path is the location of the DaVinci Developer Classic workspace file. See 6.1 for further details about the path handling on the command line interface.

Options:

- `--create-root-design`: generate a Root Design consisting of the System and the Root Composition (see 4.3 Root Design Creation for more details)

In the DCF file at location "`<path of workspace file>.dcf`" the tag "`<PROJECTASSISTANT>`" is created in the dcf file and the path to the DaVinci Project file is inserted if specified workspace exists.

The subcommand will print: "`Workspace linked successfully: "<path of workspace file>.dcf linked to <path to davinci project file>.dvjson`" when workspace was linked. In case of errors the following messages are printed:

- "`Could not modify the workspace <path of workspace>.dcf – Reason: <error code from system>`" → return code ACCNOK will be set
- "`Workspace <path of workspace file>.dcf not found`" → return code WSNOK will be set

Possible Return codes are:

- OK (0): workspace creation successful
- ACCNOK (3): could not create missing parts of the path (missing rights, maximum path length exceeded etc.)
- WSNOK (7): workspace not existing

Help content can be displayed by entering `dvdevc project link -h` or `dvdevc project link -help`.

4.3 Root Design Creation

To support the creation/instantiation of SWCs within a System context, DaVinci Developer Classic provides the capability to generate a Root Design consisting of the System (aggregating data mappings) and the Root Composition. These defaults enable the use of **Software Design**, **Data Mapping** as well as other System-dependent editors and views. In addition to the existing GUI functionality, this creation process can now also be performed via dvdevc.exe:

```
dvdevc model create-root-design --workspace-path <path of workspace file>.dcf
```

- `model create-root-design`: creates a default system with a root composition if none exists
- `--workspace-path`: path to workspace file

Note
The root design is only generated if none already exists.

In interaction with DaVinci Configurator Classic 5, the system was automatically created during the project creation process (triggered in DaVinci Configurator Classic 5) and updated whenever an external system was provided. With the decoupled project creation process, you now have the flexibility to trigger this process manually, eliminating any hidden background operations.

Note
Please note that the mechanism for updating from external sources is currently not supported.

Hence, it is important to note that creating a root design is only necessary when OEM input without root design is available. If an externally defined root design is expected, this must be used instead.

In the DCF workspace folder a subfolder “ECUProjects” is created (if not already existing) including an ARXML file that contains the default root design. This file is added to the DCF file to define it as a file to be loaded by the workspace.

If the command execution was successful, message "`Creating default RootComposition completed`" is printed.

Help content can be displayed by entering `dvdevc model create-root-design -h` or `dvdevc model create-root-design –help`.

4.4 Input File Processing

As described in previous chapters, DaVinci Configurator Classic 6 no longer invokes DaVinci Developer Classic tool functionality, unlike in DaVinci Configurator Classic 5 environments. This change also affects **Input File Processing**.

The interaction with DaVinci Configurator Classic 5 was based on the principle that OEM input data is editable in DaVinci Developer Classic. However, this had disadvantages, as deleted or changed data provided by the OEM was overwritten during the subsequent import. With Import Mode Preset, a mechanism was available that allowed partial intervention in this behavior, which meant that fewer changes had to be repeated. However, this functionality is limited to a few applicable model objects.

With the stricter separation of the tools, the input file processing steps have been revised so that **Extendable External** is now provided as a mechanism to load the OEM inputs into the DaVinci Developer Classic. As a consequence, the file content of the OEM input file is extendable, but modification and removal of model objects is not allowed. The use of Extendable External brings decisive advantages over the procedure from the interaction with DaVinci Configurator Classic 5:

- OEM input will be available in DaVinci Developer Classic after `derive-ecuc` has been executed in DaVinci Configurator Classic 6 (no additional action, despite workspace reload required)
- Performance advantages due to removed internal model update

Note
If a change to the OEM input is necessary, a pre-shift to the **ecuxpro** is intended. See DaVinci Configurator Classic 6 documentation for more information.

The feature Extendable External may already be known from the Interface Agreement and should now be used to see the OEM input as a strict requirement (read-only) in the DaVinci Developer Classic and only extensions (such as adding ports) may be made to the software design.

Note
The use case of importing SWCs is **not** affected. The solution approach Extendable External refers to input file processing (OEM input) only.

If an already existing and migrated DaVinci Configurator Classic 5 project shall be applied to Extendable External, the DaVinci Developer Classic workspace needs to be converted. Please find more information about it in chapter 6.4.

4.4.1 Limitations and Alternative Approach

Although Extendable External is the recommended way of handling the OEM input, it represents a fundamentally different approach of how to model the SWC design. In particular, the ability to modify OEM input is intentionally restricted when using Extendable External.

Since switching to Extendable External also affects your workflow, you can use the ARXML import mechanism in DaVinci Developer Classic to update the OEM input (in a migrated project). This ensures that editing data from the OEM input remains possible within DaVinci Developer Classic. As already stated, this process is not triggered by DaVinci Configurator Classic 6, which means additional steps in DaVinci Developer Classic are required.

The **ARXML Import…** in GUI, or **DVImEx** in command line interface (short: CLI) shall be used to import the OEM input explicitly in DaVinci Developer Classic.

Expert Knowledge
The import mechanism called via DaVinci Configurator Classic 5 Input File Processing works slightly differently compared to calling the import mechanism directly in DaVinci Developer Classic. There are some objects which are handled differently:

- System and SystemMapping
- EndToEndProtectionSets

The OEM input used to call the import in DaVinci Developer Classic workspace should be the ECU Extract that has been used to call `derive-ecuc` in DaVinci Configurator Classic 6.

Note
The closest way to achieve the same import behavior as known from DaVinci Configurator Classic 5 workflow, the DaVinci Developer Classic workspace should be decoupled from the DaVinci Configurator Classic 6 project for the import step. That means, the <PROJECTASSISTANT> tag of the workspace file (.dcf) needs to be removed for this action.

Since the import mechanism transfers external data into the internal model, it is possible to modify/delete/add model objects as it is known from the DaVinci Configurator Classic 5 workflow. To avoid conflicts and duplicate objects in the corresponding DaVinci Configurator Classic 6 project, it is necessary to remove SWC design data (imported into DaVinci Developer Classic workspace) from the loaded data from DaVinci Configurator Classic 6.

In other words: SWC design from the OEM input that has been imported in DaVinci Developer Classic workspace should not be loaded a second time from the ECU Extract by DaVinci Configurator Classic 6.

Therefore, a command line exporter of DaVinci Configurator Classic 6 can be used that exports communication (incl. diagnostic and timing extensions) from a project model.

```
dvcfg-b export run --exporter communication --project <dvjson-path>
--bsw-package <package-path> --output <output-folder>
```

It will basically reproduce the Communication.arxml known from the DaVinci Configurator Classic 5 interaction.

Note

Due to the fact that the exporter is called on a project model, it will most likely contain additional data compared to artefact produced by the DaVinci Configurator Classic 5 input file pre-processing.

Especially COM and MICROSAR paths (which are usually not part of the OEM input) can be removed by a simple post-processing step (e.g. execute via PowerShell script).

Example:

```
[xml]$xml = Get-Content $InputFile
$nodesToRemove = @()
foreach ($node in $xml.SelectNodes('//*')) {
    foreach ($child in $node.ChildNodes) {
        if ($child.InnerText -eq 'COM' -or $child.InnerText -eq
'MICROSAR') {
            $nodesToRemove += $node
            break
        }
    }
}
foreach ($node in $nodesToRemove) {
    $node.ParentNode.RemoveChild($node)
}
$xml.Save($OutputFile)
```

If the communication parts have been exported, this file should be added to the “ifp/harmonizedExtract” list of ArxmlProjectContent.json (settings file of DaVinci Configurator Classic 6). The old (potentially migrated Communication.arxml) and the ECU Extract (added during `derive-ecuc`) should be removed.

Caution

Using the import mechanism to update DaVinci Developer Classic workspace should be a temporary way of working since it leads to various limitations and requires several manual steps whenever a new OEM input is provided.

4.4.2 Extendable External for New Projects

For new projects, it is recommended to work with Extendable External.

Reference
Please refer to chapter 5.1 for more details about project creation from scratch.

4.5 Compare Workspace via Comparison Mode

Comparison **Mode** allows the user to compare one to two workspace states or ARXML files with the context DaVinci Developer Classic workspace. It is provided via GUI and CLI. The following subchapters will introduce both ways.

4.5.1 Comparison Mode in GUI

The GUI functionality has not changed much compared to DaVinci Configurator Classic 5 interaction, with one exception: the ability to read *.dvjson files. See picture below:

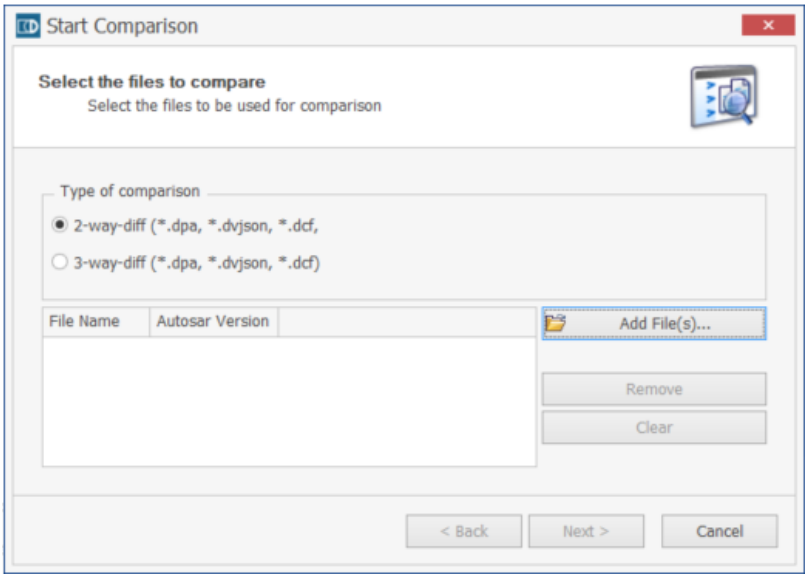


Figure 4-1: Start Comparison dialog

4.5.2 Comparison Mode without GUI

The non-GUI Comparison Mode is a new feature, which will allow the user to merge the DaVinci Developer Classic workspace without having to open the tool. It can be called the following way:

```
dvdevc project compare --mine-project <mine-project> --other-
project <other-project> [--base-project] <base-project> [--
automerge] [--conflict-resolution] <conflict-resolution> [--
report-file] [--disable-uuids]
```

- `project compare`: domain for processing a workspace and subcommand for comparing workspaces
- `--mine-project`: path to project MINE
- `--other-project`: path to project OTHER

Options:

- `--base-project`: path to project BASE
- `--automerge`: execute auto-merge (default conflict resolution is USE_MINE)
- `--conflict-resolution`: possible values are: USE_MINE or USE_OTHER
- `--report-file`: path to .html report file
- `--disable-uuids`: do not use UUIDs for comparison

Example
This is an example on Windows of how to use it:

```
dvdevc project compare --mine-project "c:\davinci\project-
mine\mineproject.dvjson" --other-project
"c:\davinci\project-other\otherproject.dvjson" --base-
project "c:\davinci\project-base\baseproject.dvjson" --
automerge --conflict-resolution USE_OTHER --report-file
"c:\davinci\project\reports\diffreport.html" --disable-
uuids
```

Help content can be displayed by entering `dvdevc project compare -h` or `dvdevc project compare -help`.

4.6 Variant Handling: Variant Update

If an updated Evaluated Variant Set (short: EVS) is loaded with a DaVinci Developer Classic workspace, three use cases can occur relative to the content of the last loaded EVS file version:

- variants are added,
- variants are deleted, or
- values are changed.

In the case of an **added** variant, the user will be able to choose the new variant for variation points. A **deleted** variant which was already used in the model can cause unwanted behavior and variation points can be marked as invariant.

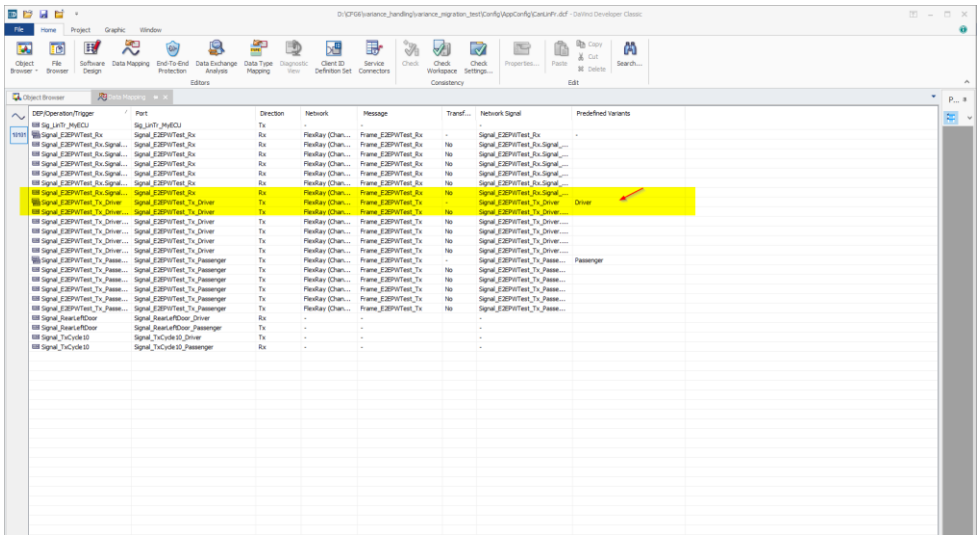


Figure 4-2: Faulty behavior after variant deletion: before EVS update

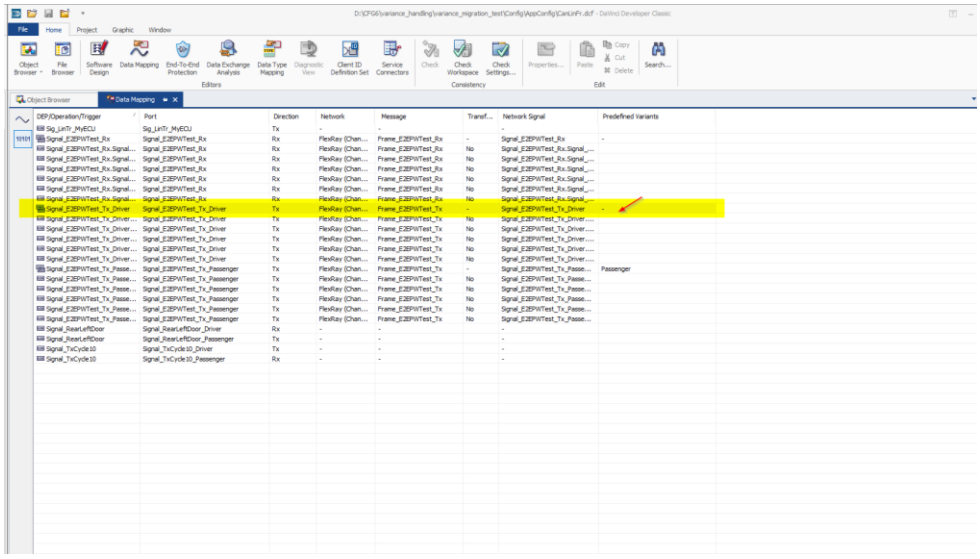


Figure 4-3: Faulty behavior after variant deletion: after EVS update

For bringing the model in a consistent state again the following steps must be done:

- Optional: Compare the old and new EVS files to find differences and to have a better understanding of the changes
- Copy the new EVS file version to the location where the old EVS file version was loaded from by DaVinci Developer Classic
- Open the workspace in DaVinci Developer Classic
- Perform a model consistency check with **Check Workspace** 

- There will be “ERROR 31002” messages for deleted variants

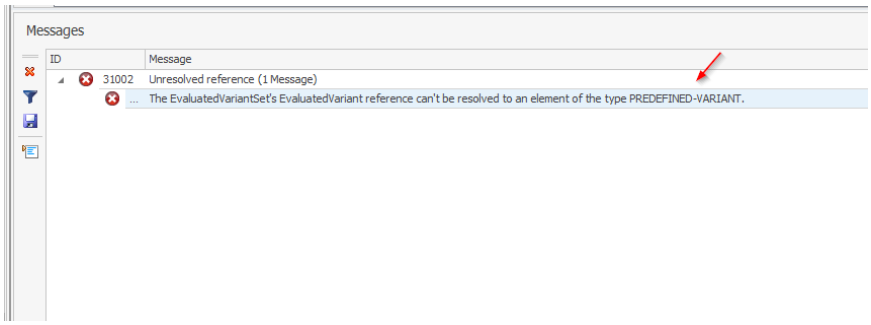


Figure 4-4: Error message 31002

> Choose another variant for the mapping if necessary or un-map signals which were just mapped to the deleted variant(s) to make it invariant.

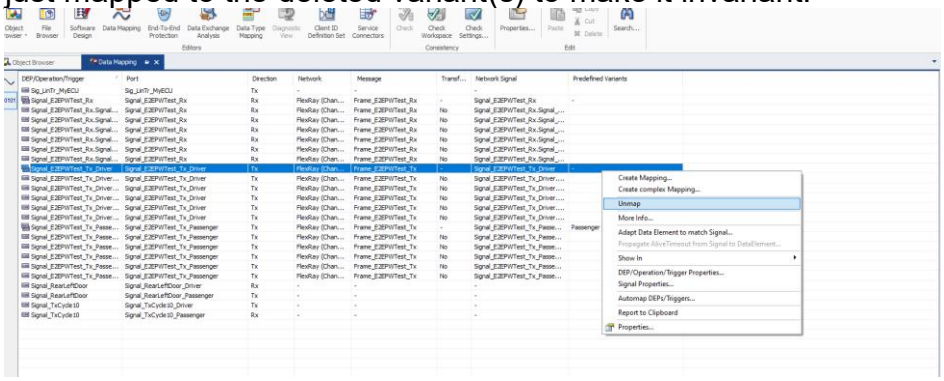


Figure 4-5: Un-mapping signals

Note
Unmap will result in an invariant signal or connection.

> Run a model consistency check, if the model is consistent again

4.7 Model Consistency Checks

To validate your model and check if there are inconsistencies or other model issues you can use the command `dvdevc project validate`:

```
dvdevc project validate --workspace-path <path of workspace file>.dcf [--model-objects] <model-objects> [--object-types] <object-types> [--remove-unused-objects] [--include-locked-objects] [--omit-references] [--remove-unused-objects-iterative] [--remove-types] <remove-types> [--remove-elements] <remove-elements> [--ensure-references] [--output] <path to output> [--verbose]
```

> `project validate`: validates the workspace

> `--workspace-path`: path to workspace file (or DaVinci Configurator Classic 6 project)

Options:

> `--model-objects`: select objects to be validated

- `--object-types`: select object types to be validated
- `--remove-unused-objects`: remove unused objects from workspace
- `--include-locked-objects`: include locked objects for model consistency checks
- `--omit-references`: omit references from data mapping sets
- `--remove-unused-objects-iterative`: remove unused objects iteratively from workspace
- `--remove-types`: included object types when removing unused objects
- `--remove-elements`: excluded object types when removing unused objects
- `--ensure-references`: ensure all references are resolved
- `--output`: specifies output path and output format (.html/.xml)
- `--verbose`: enable verbose mode

Help content can be displayed by entering `dvdevc project validate -h` or `dvdevc project validate --help`.

The command is a replacement of the DVWspChecker tool of previous releases. The behavior is the same, but parameter names have a new format of “--long-name-parameter“. For further information about the checks please refer to the help on CLI, or the documentation of DVWspChecker.

Caution
On Linux the following parameters are implemented so far: --workspace-path, --model-objects, --object-types, --rte-generation and --verbose. All others cannot be used yet.

As a result, you get findings of the model consistency as output on the console and optionally in a logfile:

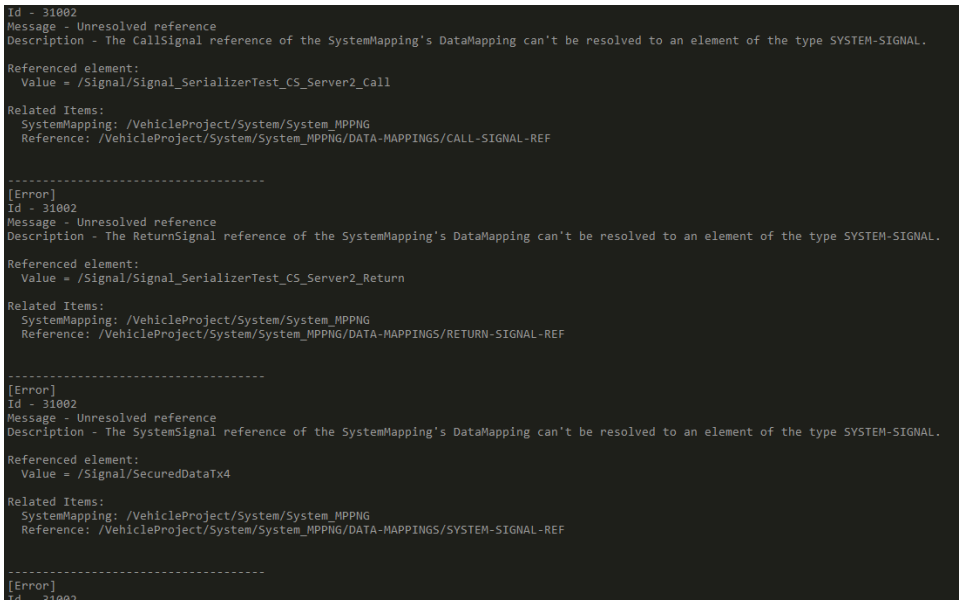


Figure 4-6: Console output of validation results

Caution
It is important to note that the model consistency rules under Linux only support a subset of the rules under Windows. Migration is an ongoing task. For a complete model consistency check, execution under Windows is recommended.

4.8 Contract Header and Implementation Template Generation

Even if the Contract Header und Implementation Template Generation is already supported by DaVinci Configurator Classic 6, the current versions of DaVinci Developer Classic will still contain DaVinci Configurator Classic 5 components as part of the installation.

4.8.1 SWC Generation via CLI

Use DVSwcGen.exe located in DaVinci Developer Classic installation path in the “/bin” subfolder:

```
DVSwcGen -d <path to dcf> -gi/gc -m <component type> -o <path to output folder>
```

- `-d:` path to workspace
- `-gi` and `-gc:` used to distinguish generation of **i**mplementation templates and **c**ontract header files
- `-m:` optional parameter to select one or more component types to be generated
- `-o:` path to output folder

Note
If `-m` is not used, the complete ECU project will be generated.

4.8.2 SWC Generation via GUI

Use DaVinci Developer Classic SWC Template Generator in Generation Settings dialog.

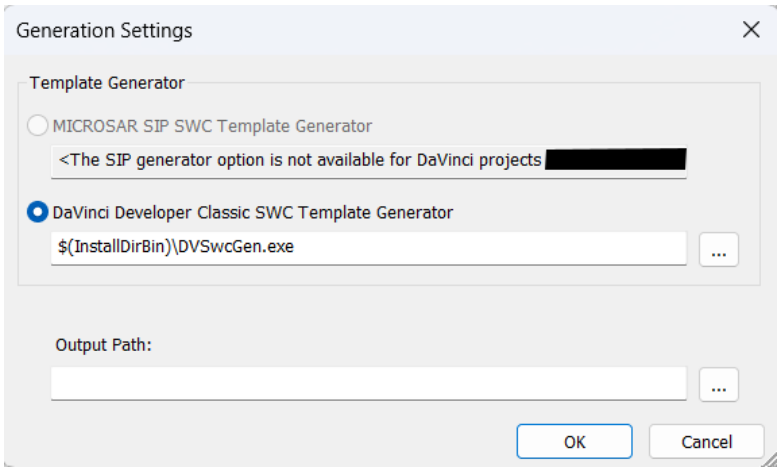


Figure 4-7: Contract Header and Implementation Template in GUI

It is automatically selected since the first option MICROSAR SIP SWC Template Generator is blocked by the following message: “The SIP generator option is not available for DaVinci projects (<dvjson-path>)”.

Note
Until further notice, the BSW Package Generator is not available for use in this process.

4.9 Support Request Package Creation

The creation of a Support Request Package (short: SRP) is currently not implemented as a tool functionality within DaVinci Developer Classic. If providing the workspace data is required, the files must be manually packaged within the file system.

Note
Please note that files referenced in the DCF may reside outside the workspace folder (containing the DCF file).

5 Example Workflows from Scratch

The following subchapters will describe crucial workflows of DaVinci Developer Classic in interaction with DaVinci Configurator Classic 6, illustrated with examples.

5.1 Project Creation with ECU Extract

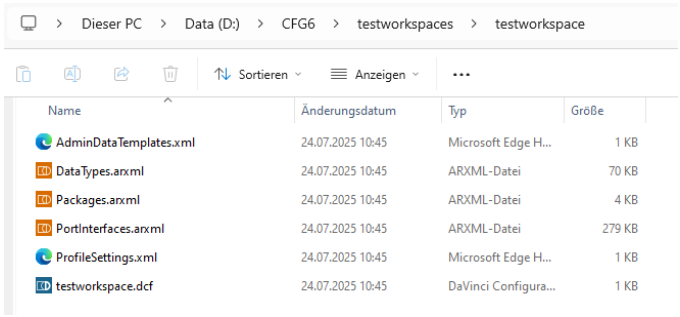
We use a **NON**-App Package workflow on **Windows** as an example and use **CLI** commands as far as possible. The subchapters will provide more details about the following intended workflow:

5.1.1 Step 1: Create DaVinci Developer Classic Workspace

The first step to create a new DaVinci Developer Classic workspace is to call `project create` via dvdevc:

```
dvdevc create --workspace-path
"D:\CFG6\testworkspace\testworkspace.dcf" --model-version 2.0 --
autosar-version 24-11
```

After successful creation the following workspace files should be created:



Reference
Refer to chapter Workspace Creation 4.1 for more detailed information about `project create` and corresponding command line options.

5.1.2 Step 2: Create DaVinci Configurator Classic 6 Project

The creation of a DaVinci Configurator Classic 6 project is the second step.

Note
Even if it is described as second step in the workflow, it is possible to create a DaVinci Configurator Classic 6 project in parallel or beforehand. While steps 1 and 2 are interchangeable, the subsequent steps must remain in sequence.

It can be created by calling `project create` via the DaVinci Configurator Classic 6 command line tool **dvcfg-b**:

```
dvcfg-b project create --bsw-package=D:\anyFolder\bswPackage --param-file=D:\anyFolder\ProjectCreationSettings.json
```

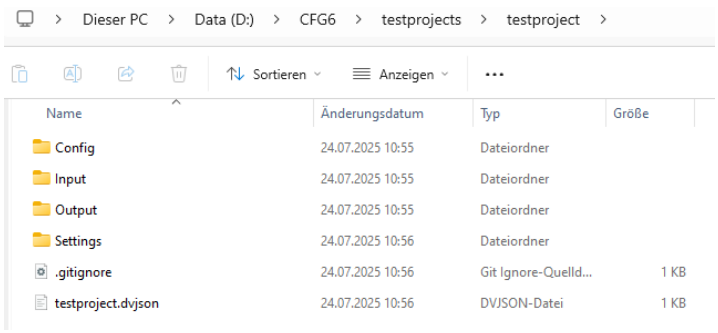
As a precondition, it is required to provide a parameter file and integrate it into the command line call.

The parameter file contains information to set up a project with individual settings. A link to a DaVinci Developer Classic workspace is created if the key “references.dvDeveloperWorkspace” is set.

D: > CFG6 > param-files > {} ProjectCreation2.json > ...

```json
{
    "compatibilityVersion": "1.0",
    "general": {
        "name": "mynewproject",
        "version": "1.0",
        "author": "theuser",
        "projectFolder": "./temp/ProjectCreation"
    },
    "ecucSplitter": {
        "separateFiles": true,
        "ownFolderForEachSplitter": true,
        "ownFileForEachInstance": true
    },
    "folders": {
        "applicationComponents": "./Config/AppComponentsCustom",
        "timingExtension": "./Config/TimingExtensionsCustom",
        "autosarFiles": "./Config/AUTOSARCustom",
        "logFolder": "./Output/LogCustom",
        "mcDataFolder": "./Output/Source/McDataCustom",
        "serviceComponents": "./Output/Config/SoftwareComponentsCustom",
        "templates": "./Output/Source/TemplatesCustom",
        "genData": "./Output/Source/GenDataCustom",
        "genDataVtt": "./Output/Source/GenDataVttCustom"
    },
    "environment": {
        "derivative": "CoolDerivative",
        "compiler": "Renesas",
        "pinLayout": "pin234",
        "projectType": {
            "type": "SoftwareClusterConnection",
            "domain": "SystemExtractProject"
        },
        "targetType": "VIRTUAL",
        "useCases": {
            "MyPreUseCase": "MyPreUseCaseValue1c",
            "MyRecUseCase": "None"
        },
        "postBuildLoadableSupport": true,
        "postBuildSelectableSupport": true,
        "syncJobRole": "SyncJobRoleValue"
    },
    "references": {
        "dvDeveloperWorkspace": "./DvDeveloperWorkspace",
        "vttProjectFile": "./VttProjectFile"
    }
}
```

After successful creation, the following project folder and files should be created:



Reference
Refer to DaVinci Configurator Classic 6 documentation for more detailed information about `dvcfg-b project create` and corresponding command line options.

5.1.3 Step 3: Link a DaVinci Project

The DaVinci Developer Classic can link a DaVinci project via the workspace file (.dcf). During loading of the workspace referenced files of the linked project will be loaded as external file references. The content of the files can be used for modelling software components.

5.1.3.1 Use “project link” command

The following step-by-step instructions will give you some information about the `project link` command of dvdevc:

1 Open your command line tool

2 Navigate to the /bin folder of your DaVinci Developer Classic installation (e.g. on Windows: “cd C:\Program Files\Vector DaVinci Developer Classic <version>\Bin”)

3 Use the following command to link the DaVinci Developer Classic workspace to your DaVinci Configurator Classic 6 project:

```
dvdevc project link --workspace-path <path to workspace file>.dcf --davinci-project <path to davinci project file>.dvjson
```

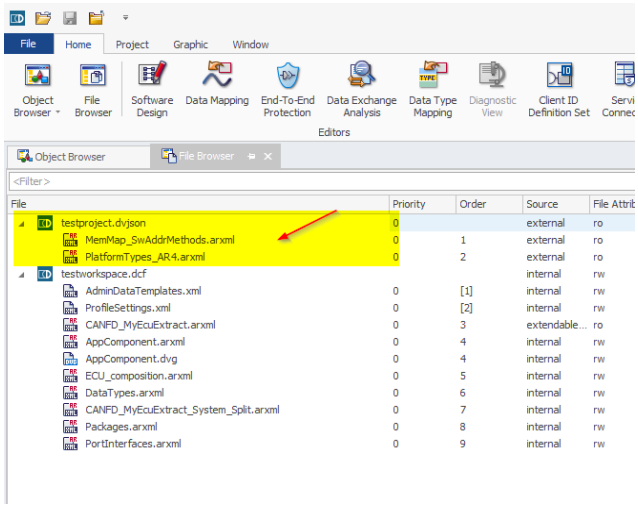
Note
A DaVinci project can also be linked at project creation time with parameter `--davinci-project` of command `project create`.

As a result, the tag “<PROJECTASSISTANT>” is now added to your workspace file (.dcf) and the linkage is done.

For Example, on Windows:

<PROJECTASSISTANT>D:\CFG6\testprojects\testproject.dvjson</PROJECTASSISTANT>

If you load the workspace in the GUI the referenced files of the DaVinci (DaVinci Configurator Classic 6) project will be loaded. You can check this in the **Object Browser** and see the loaded files:



Caution
The referenced DaVinci project must exist at the specified location at link time, because the command line tool dvdevc checks, if the specified ".dvjson" file exists.

5.1.3.2 Backlink to DaVinci Developer Classic Workspace

Note
It is possible to link the DaVinci Developer Classic workspace in two ways:

> as part of the parameter file of step 2 as described, or

> manually adding the workspace path in General.json.

The details can be found in the DaVinci Configurator Classic 6 documentation.

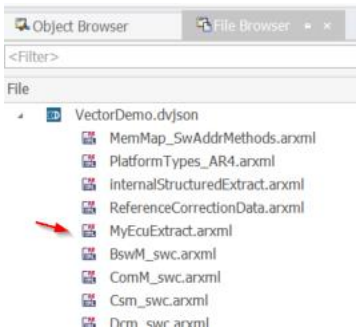
5.1.4 Step 4: Automatic Load of ECU Extract

Note
The following description is based on the Extendable External approach. Please make sure to read chapter 4.4 before applying the steps here.

Let's assume that the file "MyEcuExtract.arxml" is an OEM Extract that shall be considered as input for the DaVinci Developer Classic workspace.

If the DaVinci Developer Classic workspace has a link to a DaVinci project and is therefore not used stand-alone, the OEM input is loaded from the DaVinci project settings (./settings/ArxmlProjectContent.json). The first file which is defined in the file list of the key "harmonizedExtract" will be loaded.

The result can also be verified by checking the **File Browser**.



5.1.5 Step 5: Start Software Design

After the projects have been created und an OEM input has been added, the DaVinci Developer Classic workspace is ready, and the software design can be started. If the provided ECU Extract does not contain a root composition (or even the file is not provided), it should be defined explicitly.

If the root composition is not defined, some editors are not available because the context root is missing, e.g. **Data Mapping**.

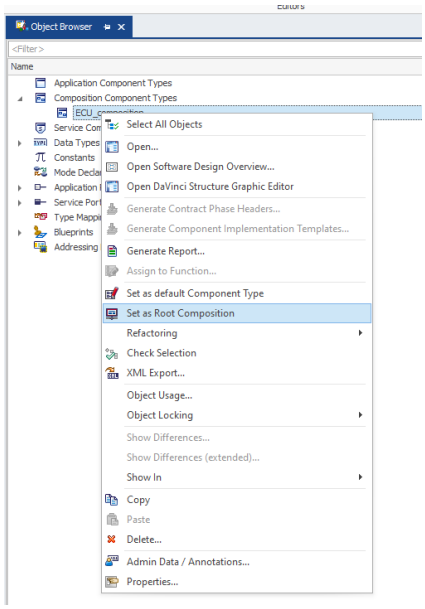
Note
The creation of the root composition is not needed if you want to set up an AppPackage project.

The CLI operation described in chapter 4.3 is available for this process. Besides this command, a root composition can already be created with the `project create` command described in chapter 4.1 and `project link` command described in chapter 4.2.

This can also be done in the GUI by following these steps:

> Open context menu of intended composition component type

> Click **[Set as Root Composition]**



When the assignment succeeds, selecting **[Software Design]** from the ribbon menu opens the chosen composition component type.

Caution
The use case of defining the root composition in the DaVinci Developer Classic workspace and then overwriting or merging it with an OEM input is not supported. A subsequent merge can have risks. Therefore, the use case is strongly discouraged.

Instead, it is recommended to use the existing OEM input before creating a root composition manually. If this is not possible, please refer to chapter 6.6.

5.1.6 Step 6: Check Workspace (validation)

The workspace model can be checked for inconsistencies by using the `dvdevc project validate` command:

```
dvdevc project validate --workspace-path "<path to workspace file>"
```

To have the outputs additionally written to a logfile, use the `--output` argument as follows:

```
dvdevc project validate --workspace-path "<path to workspace file>"
--output "<path to logfile>"
```

As a result, a log file is created at the given path that contains all results of the check.

Reference
See chapter 4.7 for more details.

5.1.7 Step 7: Generate Implementation Templates

The implementation templates can be generated by calling the DVSwcGen that is part of the DaVinci Developer Classic installation.

Either an ARXML file or a DCF workspace can be used as a basis.

For example:

```
DVSwcGen -d
"D:\CFG6\testworkspaces\testworkspace\testworkspace.dcf" -gi -o
"D:\CFG6\testworkspaces\testworkspace\templates"
```

Reference
Please refer to chapter 4.8 for more detailed information about Contract Header and Implementation Template Generation.

5.2 Input File Update

We use a **NON**-App Package workflow on **Windows** as an example and use **CLI** commands as far as possible. The subchapters will provide more details about the intended workflow.

5.2.1 Step 1: Derive ECUC

Use updated OEM extract in DaVinci Configurator Classic 6 CLI to call `dvcfg-b project derive-ecuc`.

5.2.2 Step 2: Fix Model Consistency and Update Service Components

Use DaVinci Configurator Classic 6 CLI to call `dvcfg-b project update –apply-ifp-changes`.

5.2.3 Step 3: Start Software Design

The updated OEM extract is automatically loaded by DaVinci Developer Classic as long as it is referenced in DaVinci Configurator Classic 6 project settings “harmonizedExtract” (ArxmlProjectContent.json).

5.2.4 Step 4: Update RTE Configuration

After software design has been synchronized, use DaVinci Configurator Classic 6 CLI to call `dvcfg-b project update –rte-config`.

6 Miscellaneous

6.1 Path Handling on CLI

6.1.1 Absolute paths

Absolute paths can be defined according to the standards of the operating system used. Double quotation marks can be used on Windows for paths which contain blank spaces. On Linux single quotation marks can be used instead. Slashes and Backslashes or a mixture is supported on Windows.

6.1.2 Relative path

Relative paths are defined relative to the current working directory. An explicit specification of the current working directory is possible (“./path”). The use of slashes and backslashes or a mixture is possible on Windows.

6.2 Tool Version Selection

DaVinci Developer Classic does not provide any information about the relationship between a specific tool version and the associated stand-alone workspaces or DaVinci Configurator Classic 6 projects. Likewise, the project settings in DaVinci Configurator Classic 6 do not include any details regarding the corresponding DaVinci Developer Classic version.

6.2.1 Tool version selection on CLI

When DaVinci Developer Classic is called, the context specifies which version is used for the execution of the commands. This means that the call to dvdevc already represents a specific version, as the command line tool is delivered as part of a version.

6.2.2 Tool version selection on GUI

Opening a DaVinci Developer Classic project often starts with double-clicking the project file (.dcf). Per default, the latest installed version is used to open the project.

If any other version than the latest installed one shall be used to open the project, DaVinci Developer Classic needs to be opened first before loading the project file (drag and drop, or Open Project…) in a second step.

To configure and launch a project with a designated DaVinci Developer Classic version, a Windows batch script provides a simple solution.

To facilitate understanding, the following example demonstrates the approach:

Example

```
@echo off
setlocal
set version="4.17 (SP2)"
set dvdevcPath="C:\Program Files\Vector DaVinci Developer Classic %version%\Bin\DaVinciDEV.exe"
set workspacePath="C:\dev\project\Config\AppConfig\devc-project.dcf"
cmd /c "%dvdevcPath% %workspacePath%"
endlocal
```

Note
Linux is not considered here.

6.3 CLI usage on Linux

The following prerequisites must be fulfilled in order to use the CLI under Linux.

- Install Linux command line tools
- Install Vector License Client for Linux or External Components for Linux if not already installed
  - Important libs are (4.17 SP2):
    - codemeter: version 8.30 and up necessary
    - axprotector: version 11.60 and up necessary
    - Please check the versions on your Linux environment (dpkg -l |grep codemeter, dpkg -l |grep axprotector)

- Check if a valid Developer license is activated in the license client on linux (see installation guide: Vector KnowledgeBase)

Reference
All above can be found on our official download center: Download-Center | Vector

6.4 Workspace Conversion

The workspace conversion is necessary to apply the Extendable External workflow.

Caution
Please make sure that you have clarified your need to switch to the Extendable External workflow with chapter 4.4 before using it.

It converts the DaVinci Developer Classic workspace so that model objects from the OEM input will be part of the ECU Extract exclusively and will not be present as duplicates in the DaVinci Developer Classic model.

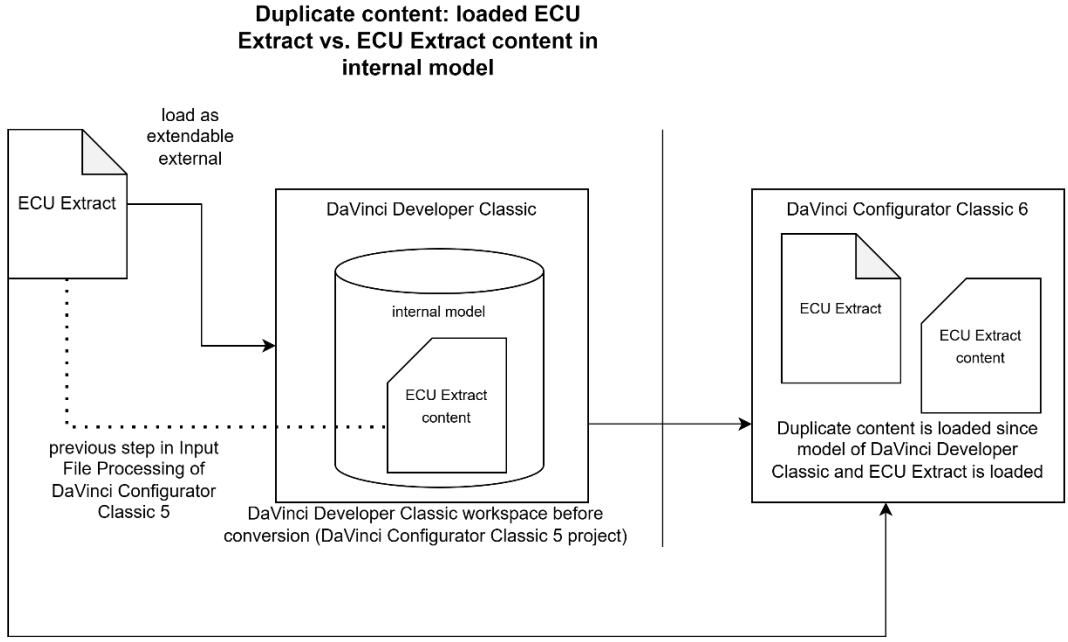


Figure 6-1: Duplicate content in DaVinci Configurator Classic 6

Before the conversion can take place, it is necessary to update the **<PROJECTASSISTANT>** tag in the DCF file. After migrating from a DaVinci Configurator Classic 5 project to DaVinci Configurator Classic 6, this reference still points to the DPA by default. For proper operation within the DaVinci Configurator Classic 6 project context, the reference must be changed to the newly created **dvjson** settings file.

Further details on this procedure can be found in chapter 4.2.

Furthermore, it is important that the initial input file processing after project migration has been executed in DaVinci Configurator Classic 6 project before workspace conversion is triggered. You can confirm it by checking the “harmonizedExtract” project setting in ArxmlProjectContent.json. If the first file in the list refers to a file called “EcuExtract.arxml”, workspace conversion can be started. Otherwise, a file called “*_Communication.arxml” is referenced in first position, which is the project state after DaVinci Configurator Classic 6 project migration. This file is not appropriate to execute workspace conversion.

6.4.1 Conversion Process

The workspace conversion consists of several steps that shall be shown here.

6.4.2 Workspace Converter Handling

The workspace conversion is called via dvdevc command line tool located in DaVinci Developer Classic installation path in the "./Bin" subfolder:

```
dvdevc project convert --workspace-path <path to workspace file> [--extract-file] <path to extract file> [--disable-uuids] [--analysis-only] [--backup-path] <path to backup folder> [--no-backup]
```

- `project convert`: converts workspace to Extendable External workflow
- `--workspace-path`: path to the workspace file

Options:

- `--extract-file`: path to the extract file
- `--disable-uuids`: UUIDs will not be used for comparison between workspace and ECU Extract
- `--analysis-only`: workspace will not be changed, but changes are analyzed
- `--backup-path`: path to the backup folder to be intended if default (<workspace-path>\conversionartifacts\backup) shall not be used
- `--no-backup`: backup will be disabled

For the conversion, a workspace must be specified with the `--workspace-path` parameter. This can be a *.dcf file but also be a *.dpa or *.dvjson file if a *.dcf file is referenced in it.

- If a *.dcf file is specified that references a *.dpa file, the extract file can be determined from the *.dpa file. In this case, the SystemExtract.arxml file is used.
- If a *.dpa file is specified that references a *.dcf file, the extract file can be determined from the *.dpa file. In this case, the SystemExtract.arxml file is used.

In both cases, the `--extract-file` parameter is optional. It can also be used when the SystemExtract.arxml from the *.dpa file is not intended to be used.

But usually, the SystemExtract.arxml should be the correct file for the conversion process for a *.dpa file.

Caution
If a *.dvjson file is specified, the parameter `--extract-file` must be used to specify the path to the extract file that is used for the conversion, as this cannot yet be determined from the dvjson file by DaVinci Developer Classic.

During the conversion, the workspace is compared with the extract file. In this step, elements contained in both the workspace and the extract file are identified.

Elements that are present in both the workspace and the extract file are deleted from the workspace.

However, there are a few things to note here:

- Identical elements are deleted from the workspace without further ado.
- Elements that have differences between workspace and extract file must be examined more closely:
  - Certain differences caused by a previous import (during the DaVinci Configurator Classic 5 input file update) can be ignored and filtered out. (see chapter 6.4.3)
  - Elements that only have differences that can be ignored are also deleted.
  - Elements that have conflicts are removed from the workspace, while the differences are extracted and preserved.

The differences are treated as model conflicts and are therefore removed from the workspace but preserved for conflict resolution.

As a result, the duplicated content has been resolved.

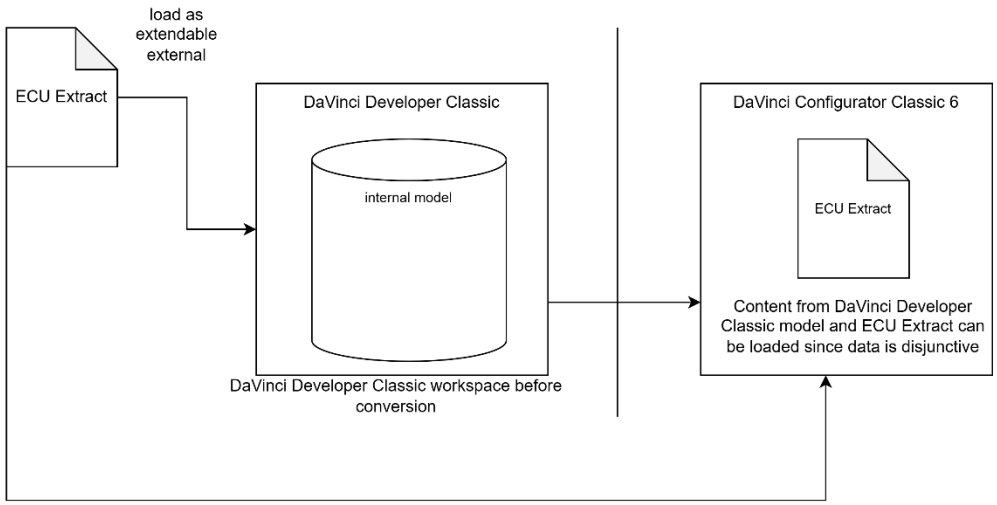


Figure 6-2: Complementary content in DaVinci Configurator Classic 6

When comparing the project state before and after conversion, you may notice that the software components within the root composition are organized differently in the Software Design editor (GUI). This occurs because the graphic file (.dvg) of the root composition is removed during the conversion process.

To restore the original graphical design, you can manually copy the graphic file from the (conversion) backup folder in two steps:

> Copy the .dvg file from the backup folder to the ECUProjects folder.

> Add the .dvg file to the corresponding ARXML file (same name).

After completing these steps, the result in the DCF file should look like the following:

```
<FILEREF>
   <ARXML ROOTITEM="ECUPROJECT"
TYPE="LOCAL">ECUProjects\anyFile.arxml</ARXML>
   <DVG>ECUProjects\anyFile.dvg</DVG>
</FILEREF>
```

6.4.3 Filter

There are filters for default values (created by previous import executions). Those default values would cause differences and will be ignored. The behavior is described in more detail in the following subchapters.

6.4.3.1 AdminData: DocRevision

DocRevisions that may be contained in AdminData and differ in date are evaluated as irrelevant.

The date format may have changed during import.

6.4.3.2 CompuScale: Desc

In CompuScale, the description (DESC) was deleted during import. As this is no longer present in the workspace, but in the extract file, such differences are ignored.

6.4.3.3 ARObject: Timestamp

Differences in timestamps that are stored with the T attribute on an AUTOSAR element are considered irrelevant.

The date format may have changed during import.

6.4.3.4 SenderComSpec/ReceiverComSpec: UsesEndToEndProtection

For SenderComSpec and ReceiverComSpec, the default value of UsesEndToEndProtection was deleted during import. If this is not present, this default value is still used, so it is irrelevant.

6.4.4 Conversion Artifacts

During the conversion, a subfolder "./conversionartifacts" is created in the workspace directory.

6.4.4.1 Conversion Log File

The conversion creates a file called "conversion.log" which contains the console output of the conversion process.

The console output contains information about which individual steps are executed.

It also shows which AUTOSAR elements were found to have differences between the workspace and the extract file.

> These elements will be deleted, but the enhancements are retained in the workspace.

It also shows which AUTOSAR elements were recognized as identical or with filtered differences.

> These elements will be completely deleted from the workspace.

6.4.4.2 Backup

By default, a backup of the workspace is created in a subdirectory called "./backup".

The parameter `--backup-path` can also be used to specify a different path for the backup.

The `--no-backup` parameter can be used to switch off the creation of the backup.

6.4.4.3 Export and Patch File

The workspace conversion produces two output files which are also stored in the backup folder:

> export.arxml: includes all elements with differences and enhancements compared to the ECU Extract were detected

> patch.arxml: includes conflicts only compared to the ECU Extract

The export file is used to bring back changes/extensions of deleted elements during the conversion process.

The patch.arxml file does not play a significant role in the workspace conversion process. However, it is intended to help trace conflicts with the ECU Extract back to the extract creation process, so that the necessary changes can be introduced in the source.

6.5 Edit DaVinci Developer Classic-external SWC Design

DaVinci Developer Classic provides the capability to edit self-managed objects. Objects created in other tools or environments are not writable within DaVinci Developer Classic without applying further steps. As a result, data mappings, software components and/or connections that were originally created in DaVinci Configurator Classic 5 may no longer be editable in the GUI after migration.

Since these objects cannot be edited in DaVinci Developer Classic, and DaVinci Configurator Classic 6 no longer provides a GUI for the mentioned content, the question arises as to what editing options remain available.

Basically, the following options are feasible:

> Manual modifications on ARXML file level

> Scripting via Automation Interface

Note
The above-mentioned solutions are not applicable for OEM input because it is read-only. However, scripting can be applied as a frontloading step.

The migration of application components (incl. connections) to the DaVinci Developer Classic workspace is currently not supported by any tool functionality.

6.6 Resolve Multiple System Conflict

When working with DaVinci Developer Classic, a situation may occur where the workspace already contains a root design, and an additional, but different, root design is provided through OEM input. Since DaVinci Developer Classic does not allow the existence of more than one System, this conflict must be resolved.

If this situation occurs, it can be identified by a pop-up window displaying the message “The workspace is in an inconsistent state, please see the Action Log for details.” After confirming the dialog by clicking **[OK]**, further details can be reviewed in the Action Log, where explicit information about the existence of multiple systems is provided, see “[Error] AppModel10001 - The loaded workspace contains multiple Systems”.

The following options are available to address this issue and resolve the conflict:

> Adapt internal path manually according to external one

In the end, the available solution shall ensure that only one System will exist in the complete workspace.

The manual adjustment of internal package paths begins by locating the file in the DaVinci Developer Classic workspace where the internal System node is defined. The DCF file can help narrow down the eligible files. Typically, the relevant file is located in the “ECUProjects” subfolder. Within this file, both the package paths of the System and the referenced composition (aggregated under the root composition prototype) must be updated. The target paths can be derived from the ECU Extract.

In addition to the two package paths mentioned, other references that include the composition as context must also be updated. These include, among others, data mappings, port connections, and client ID definitions. The **Unresolved References Editor** in DaVinci Developer Classic can assist in identifying and correcting these context-related references.

After these adjustments, the workspace should load without any error messages related to multiple systems and should include content from both the internal location and the ECU Extract.

7 Abbreviations

7.1 Abbreviations

Abbreviation	Description
.dcf	File format of DaVinci Developer Classic workspace
.dvjson	File format for DaVinci Configurator Classic 6 projects
BSW	Basic software
CLI	Command line interface
EVS	Evaluated Variant Set
OEM	Original Equipment Manufacturer
SRP	Support Request Package
SWC	Software Component

8 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com