

Working with DCF

User Manual

Version 1.6

Authors:	Matthias Wernicke, Andreas Claus, Stefanie Kruse
Version:	1.6
Status:	released (in preparation/completed/inspected/released)

1 History

Author	Date	Version	Remarks
Wk	2008-02-07	0.1	Initial version
Wk	2008-02-08	0.2	Additional remark in ch. 5.1
Wk	2008-03-27	1.1	Ch. 6.2 "Handling of read-only objects" added. Ch. 5.1 "Creating a new DCF" updated. Ch. 7 "Limitations" added. Released.
Cs	2008-04-17	1.2	Ch. 4 "Relative path handling" added Ch. 7 "Limitations" updated
Ske	2009-02-08	1.3	Ch.7 "DCFUtility" added. "Limitations" removed.
Wk	2011-01-10	1.4	Harmonized with UserManual_DataImport.pdf
Wk	2011-08-12	1.5	Ch. 6.2 updated. Screenshots updated.
Wk	2012-06-15	1.6	Obsolete terms and limitations removed.

Contents

1	History	2
2	About this Document	5
2.1	Abbreviations and Items used in this Document	5
3	Introduction	6
3.1	What is a DaVinci Developer Workspace?.....	6
3.2	What is a DaVinci Configuration File?.....	6
4	DCF storage format	7
5	Working with DCF.....	11
5.1	Creating a new DCF	11
5.2	Opening an existing DCF	12
5.3	Exporting a DCF	13
5.4	Importing a DCF	14
5.5	Converting a DEV into a DCF (or vice versa).....	14
6	Using DCF as interface to a CM system.....	15
6.1	General concept	15
6.2	Handling of read-only objects	16
7	Working with DaVinci Configuration File Utility	18
7.1	Installation	18
7.1.1	Setup program.....	18
7.1.2	Licensing	18
7.2	Starting DaVinci Configuration File Utility.....	18
7.3	Open DCF workspace	20
7.4	DCF workspace	21
7.4.1	Content of a DCF workspace.....	21

7.4.2	Missing files	22
7.5	DCB files of a DCF workspace	23
7.5.1	Content of DCB files	23
7.5.2	Corrupt DCB files.....	23
7.6	Compare AUTOSAR files of DCF workspaces.....	25
7.7	Command line usage of comparison.....	26
8	Tips and Tricks	27
8.1	When to use DEV or DCF?.....	27
8.2	Working with temporary DEV.....	27
9	Contact.....	28

Illustrations

Figure 4-1: Granularity of the design objects in a DCF	8
Figure 4-2: Folder structure of a DCFP	8
Figure 4-3: Example DCF	10
Figure 6-1: Using DCF with a CM system	16

2 About this Document

Besides the traditional DaVinci Developer Workspace (DEV), DaVinci Developer supports an alternative storage format for the design data: the DaVinci Configuration File (DCF). This document describes when to use DCFs and how to work with DCFs.

2.1 Abbreviations and Items used in this Document

DEV	DaVinci Developer Workspace
DCF	DaVinci Configuration File
DCFP	DaVinci Configuration File Package
DCFE	DaVinci Configuration File Element
CM	Configuration Management

3 Introduction

Design data created with DaVinci Developer need to be persistently stored. You may choose among the following storage formats:

- DaVinci Developer Workspace (DEV)
- DaVinci Configuration File (DCF)

Both formats have their specific advantages. Depending on your use cases you should decide which format is appropriate for you.

3.1 What is a DaVinci Developer Workspace?

The DaVinci Developer Workspace is the native XML based storage format of DaVinci Developer. This storage format is optimized to achieve a fast loading and saving time of the design data.

The detailed storage format is proprietary and might be changed by Vector without notice in future DaVinci versions. Of course, loading of old workspaces is always possible with a newer DaVinci version.

3.2 What is a DaVinci Configuration File?

The DaVinci Configuration File is an alternative storage format of DaVinci Developer. This format is appropriate for cooperative working with DaVinci Developer and any external CM system with a file-based interface. You may use DaVinci Developer as authoring tool for the design data, and you may use the CM system to manage the design data in a repository. This gives you the maximum degree of freedom to use the CM system's features like versioning, branching or merging files. To ensure long term readability of old design data in the CM system, the DCF concept avoids proprietary formats where possible and uses standardized AUTOSAR XML formats instead.

4 DCF storage format

A DCF is a storage format, which allows for storing any AUTOSAR design data and related proprietary data as consistent set of files. This overall set of files is called DaVinci Configuration File Package (DCFP). A DCFP may consist of the following files:

- 1 DCF (DaVinci Configuration File)
Central file, which contains file references to a consistent set of ARXML, DCB or gen_attr.XML files
- 1..n DCFE (DaVinci Configuration File Element), each consisting of
 - 1 ARXML (AUTOSAR XML file)
AUTOSAR compliant XML file to store the AUTOSAR design data
 - 0..1 DCB (DaVinci Configuration Binary)
Optional binary file to store the proprietary data (e.g. graphics) of DaVinci, which are not covered by AUTOSAR XML formats. The DCB is optional and always associated (by file name) to an ARXML.
 - 0..1 gen_attr.XML (Generic Attribute XML file)
Optional XML file in proprietary (but published) format to store the generic attributes of the design data. You find the specification of the format in the Technical Reference “User-define attribute XML-export”. The gen_attr.XML is optional and always associated (by file name) to an ARXML.

When using DCF in combination with an external CM system, an important topic is the granularity of the data concerning versioning and branching. A trade-off needs to be found between too course grained (hard to merge) and too fine grained (huge number of files).

DaVinci uses the following granularity within a DCFP:

DaVinci object	DCFP Granularity
CAN Bus	Stored within the DCFE of the ECU Project
Component Type	Each Component Type will be stored in as separate DCFE, regardless if it's a Composition or Atomic
Constant	All Constants will be stored together in one DCFE
Data Type	All Data Types will be stored together in one DCFE
ECU Project	Each ECU Project will be stored in a separate DCFE
FlexRay Cluster	Stored within the DCFE of the ECU Project

LIN Bus	Stored within the DCFE of the ECU Project
Port Interface	All Port Interfaces will be stored together in one DCFE
Signal	All Signals will be stored together in one DCFE
Signal Group	All Signal Groups will be stored in the Signals-DCFE
Signal Type	All Signal Types will be stored in the Signals-DCFE

Figure 4-1: Granularity of the design objects in a DCF

In the file system, a DCFP is represented by a folder, which has the following structure:

```

<DCFP folder>
  <DCF_name>.dcf
  Constants.arxml
  Constants_gen_attr.xml
  DataTypes.arxml
  DataTypes_gen_attr.xml
  PortInterfaces.arxml
  PortInterfaces_gen_attr.xml
  Signals.arxml
  Signals_gen_attr.xml
  ComponentTypes
    <Component Type name>.arxml
    <Component Type name>.dcb
    <Component Type name>_gen_attr.xml
  ECUProjects
    <ECU Project name>.arxml
    <ECU Project name>.dcb
    <ECU Project name>_gen_attr.xml

```

Figure 4-2: Folder structure of a DCFP

Storing a DCF into an empty directory always creates the standard directory structure, i.e. Constants, DataTypes, PortInterfaces, and Signals are stored in the DCFP folder, whereas ComponentTypes and ECUProjects get their own subdirectory.

The paths within the DCF are relative to the DCFP folder. If desired the paths in the DCF can be modified using an XML-Editor. This allows loading/saving of DCF parts from other directories.

The paths within the DCF can have a relative path format including any number of parent directories or subdirectories or an absolute path format. UNC notation is not supported.

Example:

```

..\..\MyFolder\MySubFolder\Signals.arxml
.\MySignals\Signals.arxml
E:\DCFPool\MyDCFPart\Signals.arxml

```


**Caution**

The file name itself is fixed and depends on the object type and name of the AUTOSAR ARXML file. The additional files (DCB, GEN_ATTR.XML) must always reside in the same directory as the ARXML file. It is not allowed to specify different paths within one single file reference in the DCF.

Saving a DCF in a directory that already contains the same DCF will keep the paths as specified in the existent DCF. In this case the standard directory structure is only applied to newly created objects.

Each file reference requires the definition of the ROOTITEM type that corresponds with the ARXML file. During DCF loading this ROOTITEM is evaluated to solve external object references between the various ARXML files. Wrong or missing ROOTITEM definitions causes invalid object references that will be reported during DCF loading.


An example DCF file looks like this:

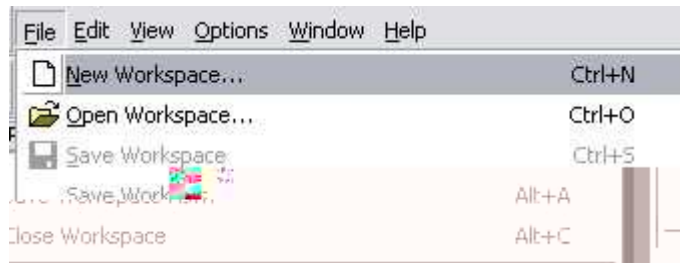
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE DCF [
  <!ELEMENT DCF ((NAME, FILEREF*))>
  <!ATTLIST DCF
    ARSCHEMA (20XSD49632 | 21XSDREV0017 | 30XSDREV0001) "21XSDREV0017">
    <!ELEMENT NAME (#PCDATA)>
    <!ELEMENT FILEREF (ARXML, DCB?, ECUC?, GENATTR?)>
    <!ELEMENT ARXML (#PCDATA)>
    <!ATTLIST ARXML
      TYPE CDATA ""
      ROOTITEM (CONSTANT | DATATYPE | PORTINTERFACE | SIGNAL | COMPONENTTYPE | ECUPROJECT | VEHICLEPROJECT) #REQUIRED>
    <!ELEMENT DCB (#PCDATA)>
    <!ELEMENT ECUC (#PCDATA)>
    <!ELEMENT GENATTR (#PCDATA)>
  ]>
<DCF ARSCHEMA="21XSDREV0017">
  <NAME>InteriorLight</NAME>
  <FILEREF>
    <ARXML ROOTITEM="COMPONENTTYPE" TYPE="">ComponentTypes\AP_Dimmer.xml</ARXML>
    <DCB>ComponentTypes\AP_Dimmer.dcb</DCB>
    <GENATTR>ComponentTypes\AP_Dimmer_gen_attr.xml</GENATTR>
  </FILEREF>
  <FILEREF>
    <ARXML ROOTITEM="COMPONENTTYPE" TYPE="">ComponentTypes\AP_DoorContacts.xml</ARXML>
    <DCB>ComponentTypes\AP_DoorContacts.dcb</DCB>
    <GENATTR>ComponentTypes\AP_DoorContacts_gen_attr.xml</GENATTR>
  </FILEREF>
  <FILEREF>
    <ARXML ROOTITEM="COMPONENTTYPE" TYPE="">ComponentTypes\CO_LightControl.xml</ARXML>
    <DCB>ComponentTypes\CO_LightControl.dcb</DCB>
    <GENATTR>ComponentTypes\CO_LightControl_gen_attr.xml</GENATTR>
  </FILEREF>
  <FILEREF>
    <ARXML ROOTITEM="CONSTANT" TYPE="">Constants.xml</ARXML>
    <GENATTR>Constants_gen_attr.xml</GENATTR>
  </FILEREF>
  <FILEREF>
    <ARXML ROOTITEM="DATATYPE" TYPE="">DataTypes.xml</ARXML>
    <GENATTR>DataTypes_gen_attr.xml</GENATTR>
  </FILEREF>
  <FILEREF>
    <ARXML ROOTITEM="PORTINTERFACE" TYPE="">PortInterfaces.xml</ARXML>
    <DCB>PortInterfaces.dcb</DCB>
    <GENATTR>PortInterfaces_gen_attr.xml</GENATTR>
  </FILEREF>
  <FILEREF>
    <ARXML ROOTITEM="SIGNAL" TYPE="">Signals.xml</ARXML>
    <DCB>Signals.dcb</DCB>
    <GENATTR>Signals_gen_attr.xml</GENATTR>
  </FILEREF>
  <FILEREF>
    <ARXML ROOTITEM="VEHICLEPROJECT" TYPE="">VehicleProjects\DemoVehicle.xml</ARXML>
    <GENATTR>VehicleProjects\DemoVehicle_gen_attr.xml</GENATTR>
  </FILEREF>
</DCF>
```

Figure 4-3: Example DCF

5 Working with DCF

5.1 Creating a new DCF

Select **File | New workspace...** or use the shortcut .

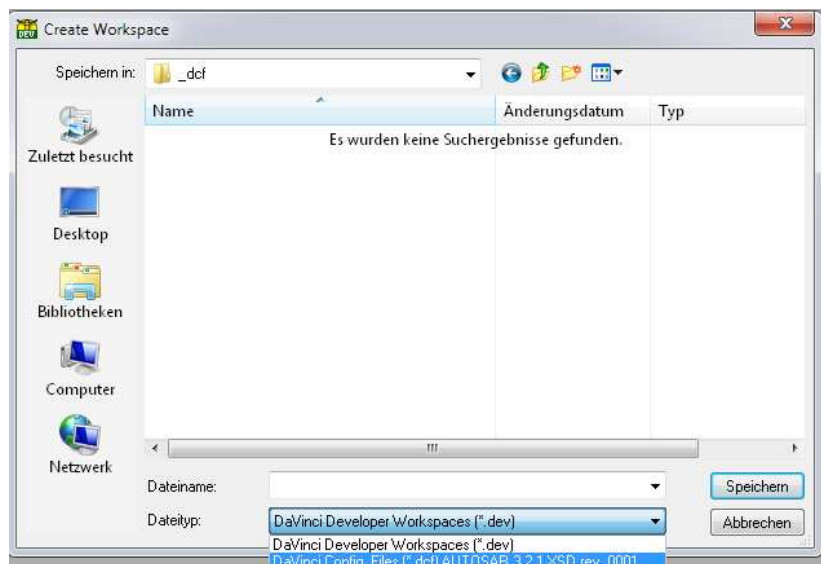


Select “DaVinci Configuration Files (*.dcf)” as file type. Browse for a suitable folder and set a file name for the DCF file.

You may now work as usual in with DaVinci Developer to define new objects.

To save the data select

File | Save Workspace... or use the shortcut .



Please observe the following conditions, which apply to the folder of a DCF file:

- The folder, which contains the DCF file, must not contain other DCF files
- Within the folder of the DCF file and within any sub-folder, the following file types are “reserved” for DaVinci
 - *.arxml
 - *.dcf
 - *.dcb

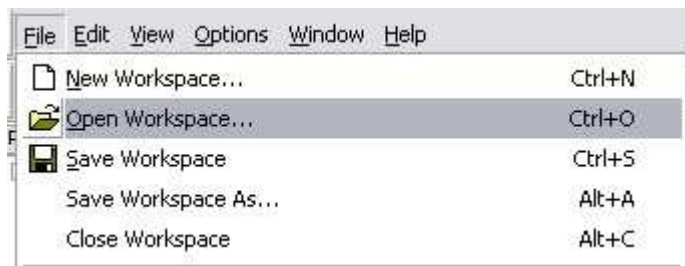
- *_gen_attr.xml

These files are created and deleted by the DaVinci when saving the DCF workspace. Other files are not touched by DaVinci.

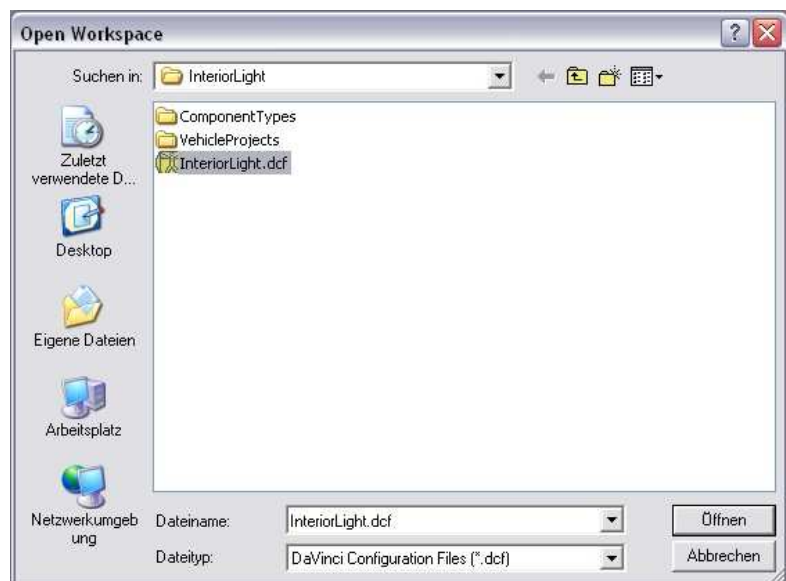
- Sub-folders of the DCF folder are not deleted by DaVinci, even if the sub-folder has become obsolete (e.g. since the according object has been deleted in DaVinci)

5.2 Opening an existing DCF

Select **File | Open...** or use the shortcut 



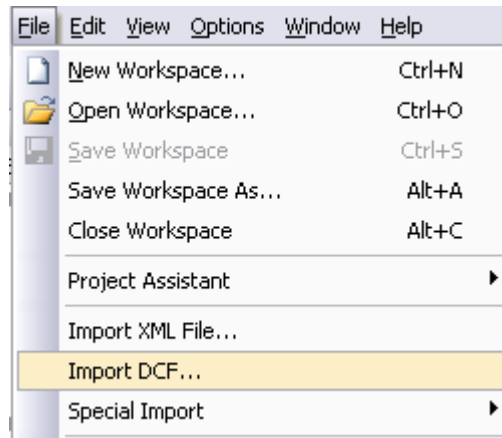
Select "DaVinci Configuration Files (*.dcf)" as file type. Browse for the DCF file.



5.4 Importing a DCF

You may import a DCF into an already opened workspace or DCF.

Select **File | Import DCF...**



5.5 Converting a DEV into a DCF (or vice versa)

You may convert an opened DEV into a DCF.

Select **File | Save Workspace As....**



Select "DaVinci Configuration File (*.dcf)" as file type. Browse for a suitable folder and set a file name for the DCF file.

In a similar way you may convert an opened DCF into a DEV.

6 Using DCF as interface to a CM system

6.1 General concept

The DCF may serve as interface between DaVinci Developer and your CM system.

- Using DaVinci Developer you create or edit the DCFP in your working copy of the CM repository.
- Your working copy may contain several DCFP, e.g. one DCFP per ECU project you are currently developing, or some DCFP for particular software components. Depending on the capabilities of your CM system you may even have several variants (branches) of e.g. the same ECU project as separate DCFP in your working copy.
- Using the CM system's user interface you may check-in (commit) the DCFP into the repository, or get an already existing DCFP.
- The read-only state of files within the DCFP (e.g. because they are checked-in) is reflected within DaVinci Developer (see chapter 6.2).
- Optionally you may use external utility tools e.g. to merge the ARXML files within a DCFP.

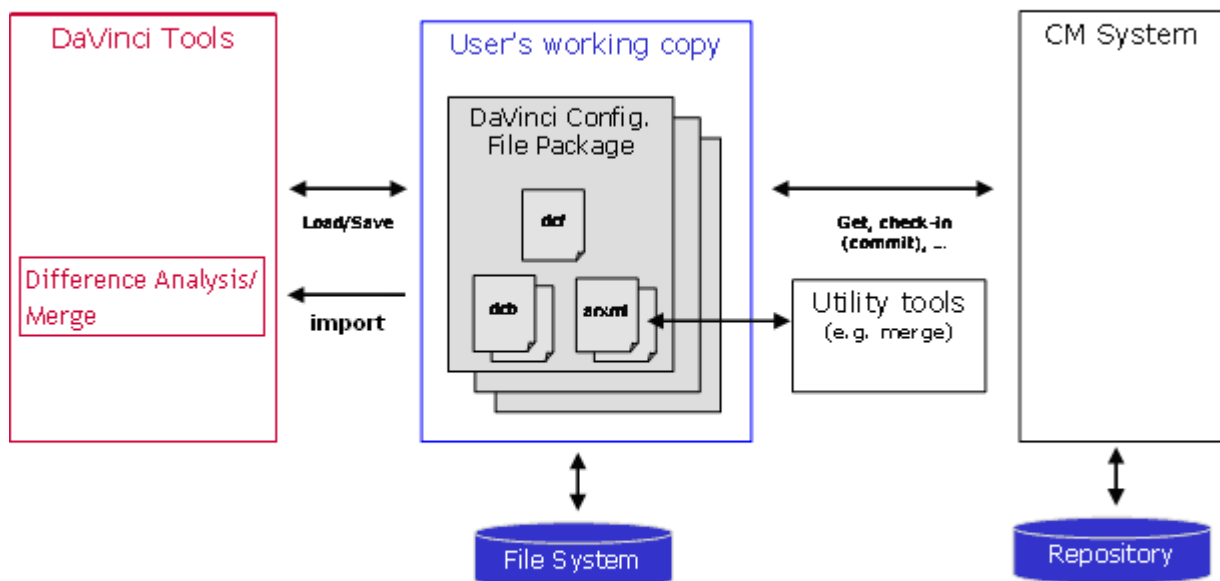
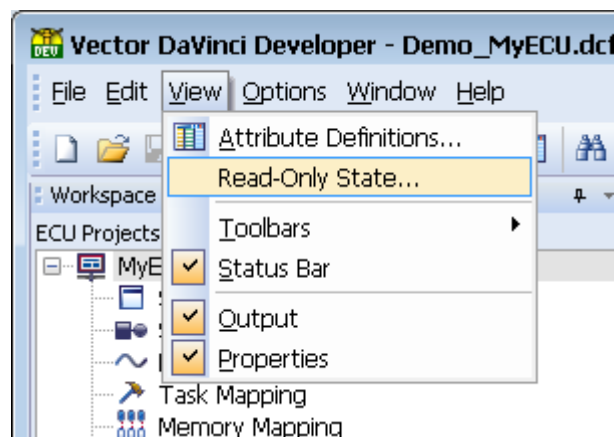


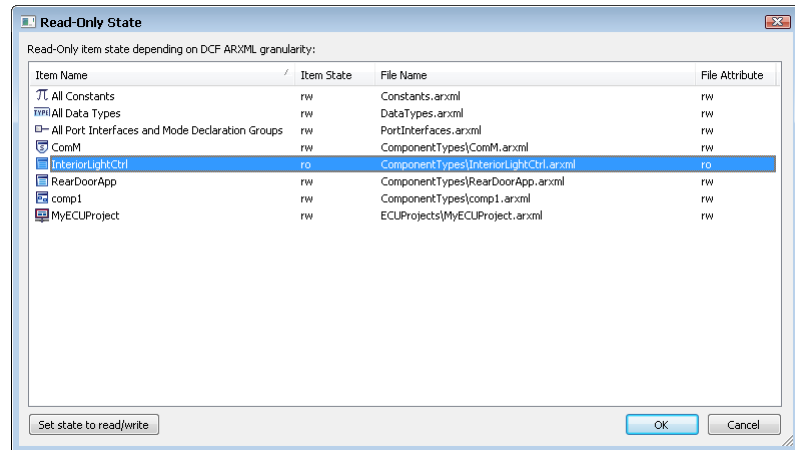
Figure 6-1: Using DCF with a CM system

6.2 Handling of read-only objects

After opening a DCF workspace, you may display the read-only state of the objects by selecting **View | Read-Only State...**

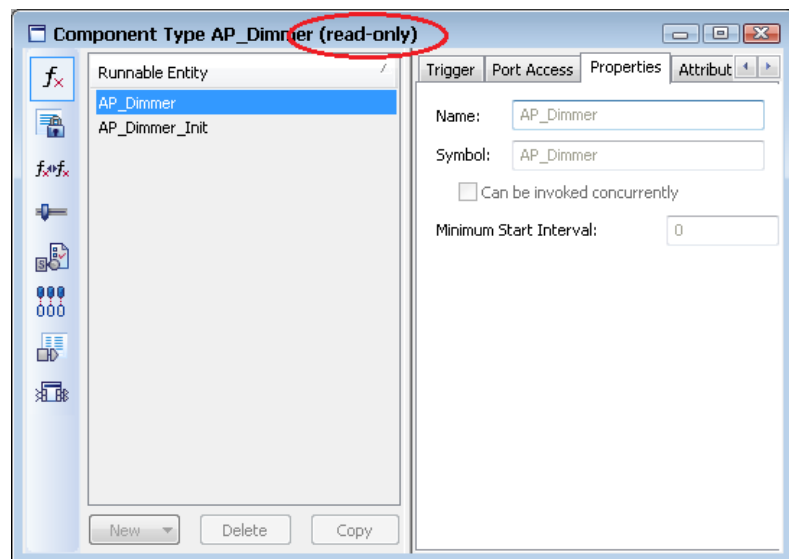


The Read-Only State dialog opens. This dialog displays all DCFEs of the workspace according to the DCFP granularity (see chapter 4). For each DCFE, the ARXML file name is displayed as well as the file attribute (“ro” read-only or “rw” read-write).



If a DCFE is read-only, all objects of the DCFE are assumed to be read-only. Any editing of the objects via dialogs or editors is blocked in DaVinci Developer.

If you want to allow editing the object anyway, you may select the DCFE and push “Set state to read/write”. This will make the objects in the DCFE read-write, while the files remains read-only. Consequently, you can edit the object, but you cannot save the workspace unless you make the files “rw”.



Note: If you have set a DCFE to read-write, it is not possible to set it to read-only again.

7 Working with DaVinci Configuration File Utility

This section describes working with DaVinci Configuration File Utility. (DCFUtility)
DCFUtility is a tool to handle DCF files.

The functionality of DCFUtility contains:

- Visualize all files referenced within a DCF workspace.
- Detect missing files of a DCF workspace
- Show content of DCB files
- Detect corrupt DCB file
- Compare AUTOSAR files of two DCF workspaces

7.1 Installation

7.1.1 Setup program

The DCFUtility can be installed by starting the setup program setup.exe. The tool is installed during the installation procedure of DaVinci Developer automatically.

The DCFUtility is installed into the folder of your DaVinci Developer installation.
The executable “DVDCFUtility.exe” is installed in the sub-folder “bin”.

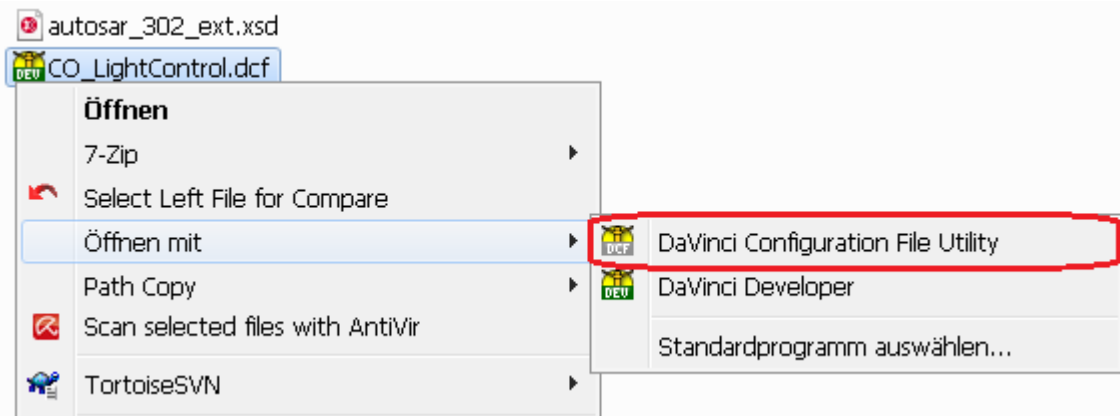
7.1.2 Licensing

In order to run DCFUtility a license of DaVinci Developer is required on the PC.


7.2 Starting DaVinci Configuration File Utility

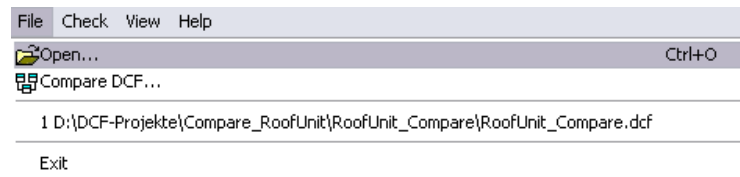
The DCFUtility can be started via one of the following options

- Via the Program menu
Using Program menu | DaVinci Developer <Version> | **DaVinci Configuration File Utility** the DCFUtility is started. Using **File | Open** you can open a dialog for selecting a DCF file files to be shown. After confirming this dialog, the DCF file is opened
- Via the Windows Explorer
After selecting a file with extension .dcf in the Windows Explorer, you can open the file using the context menu entry **Open with | DaVinci Configuration File Utility**

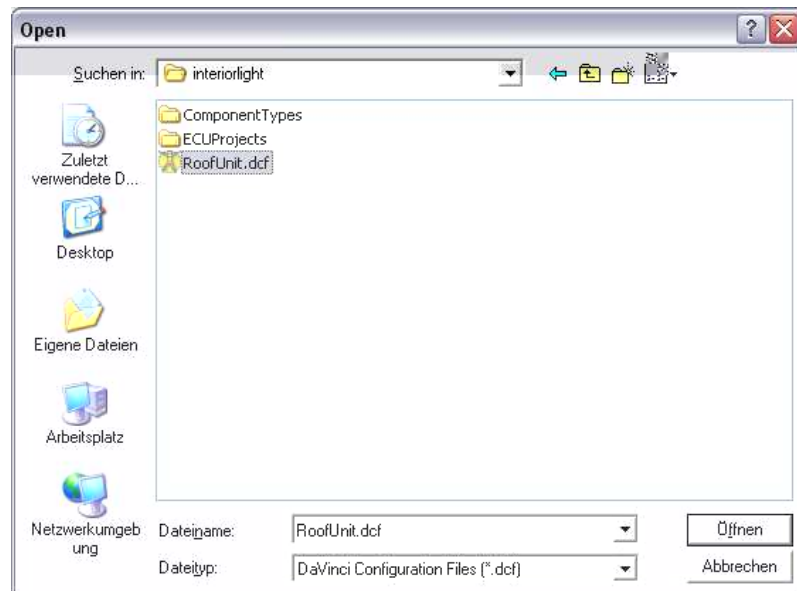


7.3 Open DCF workspace

Select **File | Open...** or use the shortcut 



Select “DaVinci Configuration Files (*.dcf)” as file type. Browse for the DCF file.

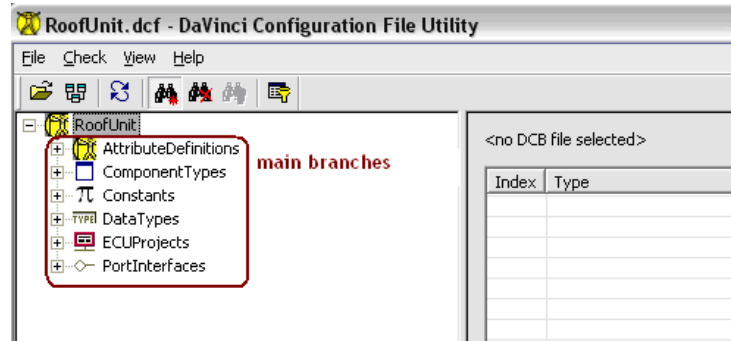


7.4 DCF workspace

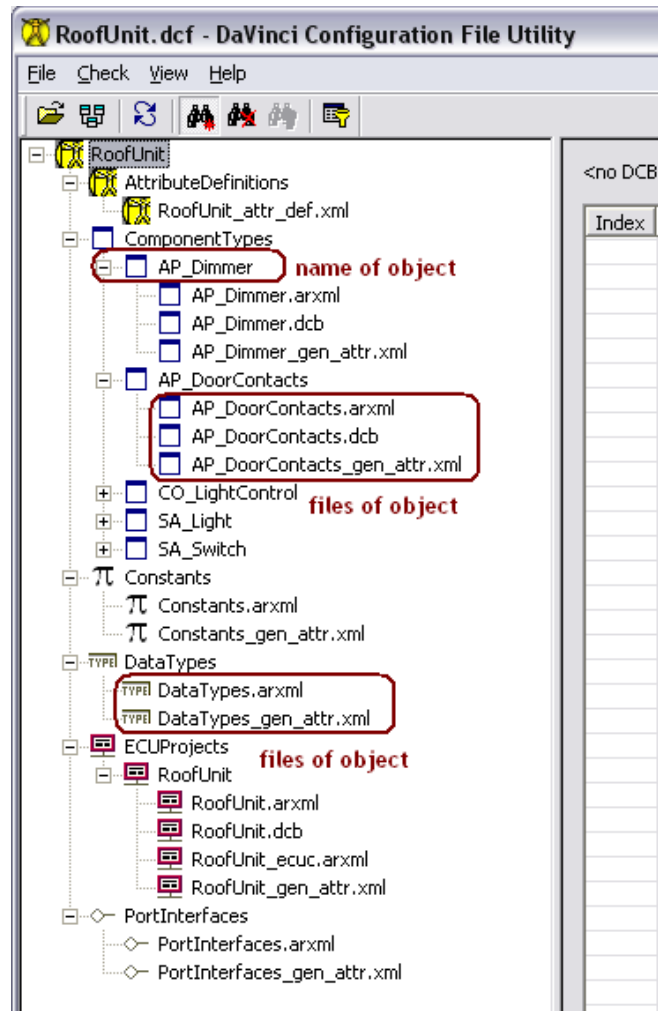
7.4.1 Content of a DCF workspace

The tree on the left hand shows the files referenced within a DCF file.

The files are sorted within the main branches shown in the picture on the right side.




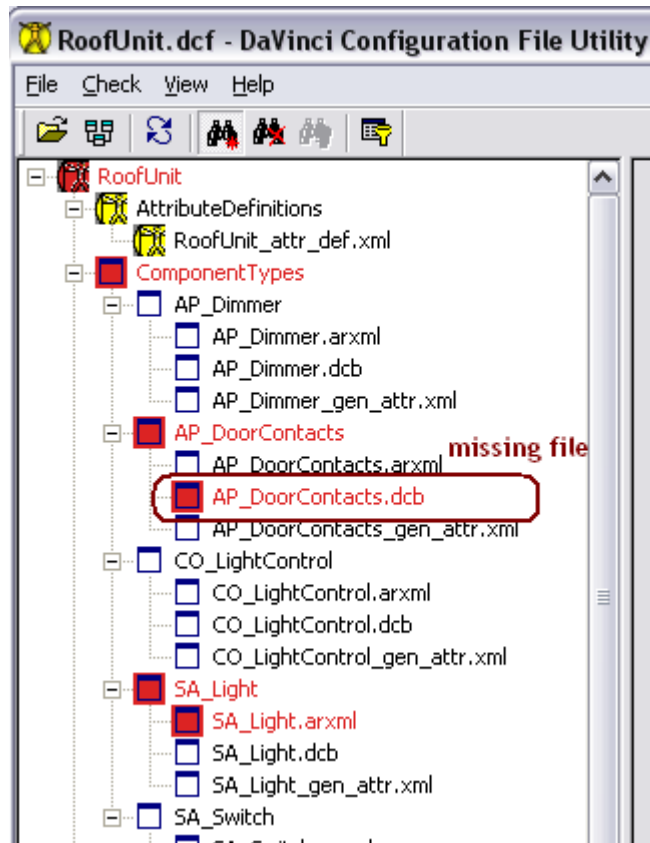
For each DaVinci object there is one sub node within the according main branch. The sub branches name is the name of the DaVinci object. The children of the objects nodes contain the names of the files which belong to one object.




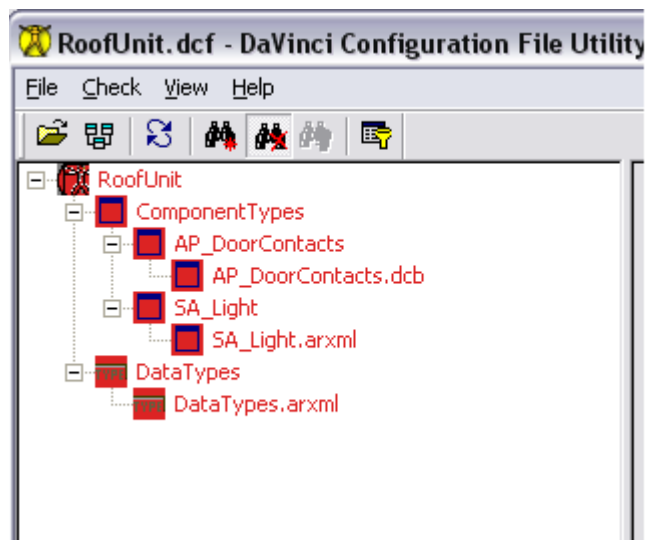
7.4.2 Missing files

When opening a DCF file referenced files in the DCF file are checked for existence automatically. Missing files and their parent nodes are marked red in the tree view of the DCF file.

Use short cut  to start the check again manually and to update the DCF file's tree view. (e.g. after copying missing files to the location specified in the DCF)



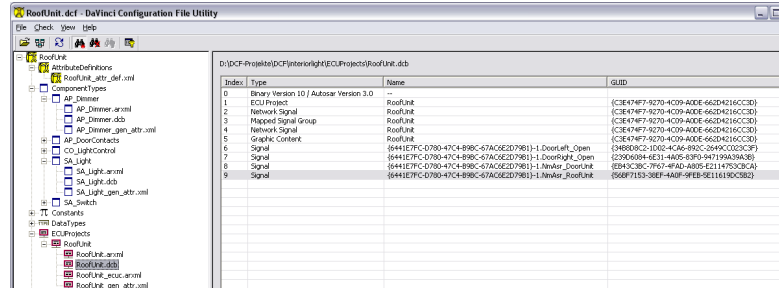
Use short cut  to show missing files and their parents only.



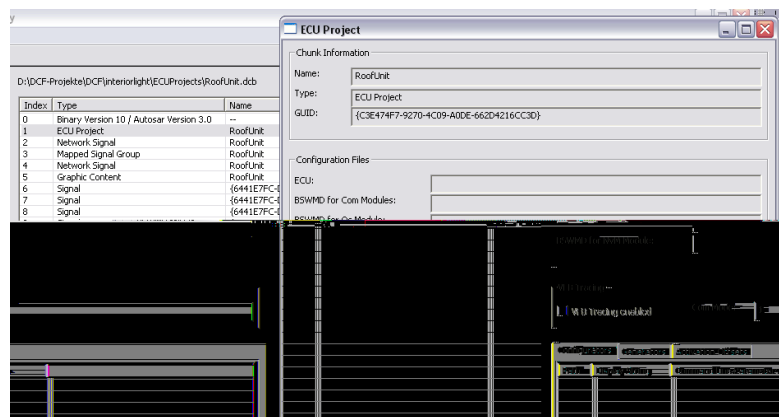
7.5 DCB files of a DCF workspace

7.5.1 Content of DCB files


Select a DCB file in tree view to show its content in the table on the right side.

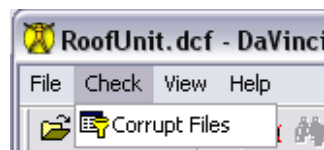


Double click on a row in the table to show the object's values.

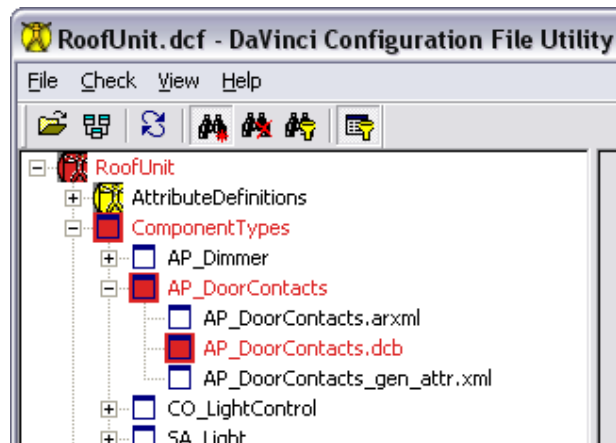



7.5.2 Corrupt DCB files

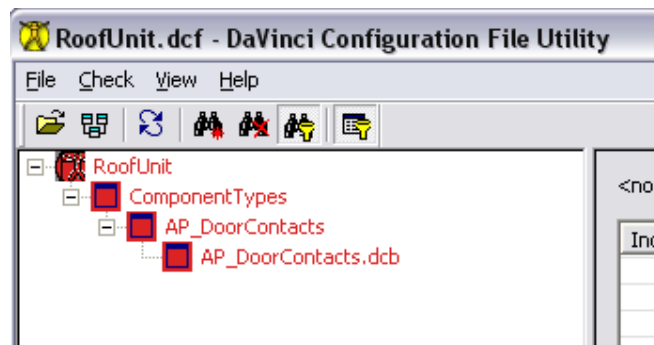
Select **Check | Corrupt Files** or use short cut  to analyze all DCB files for corruptness.




Nodes of corrupt DCB files and their parent nodes are marked red in the DCF file's tree view.

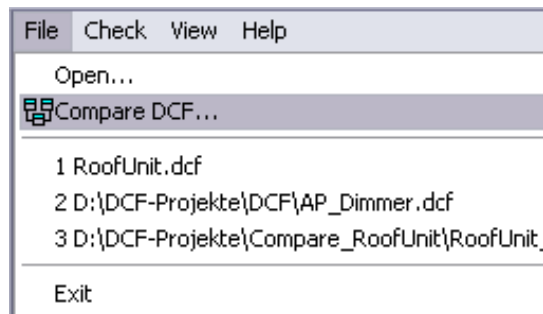


Use short cut  to show corrupt DCB files only. The short cut is enabled after checking the DCF file for corrupt DCB files.



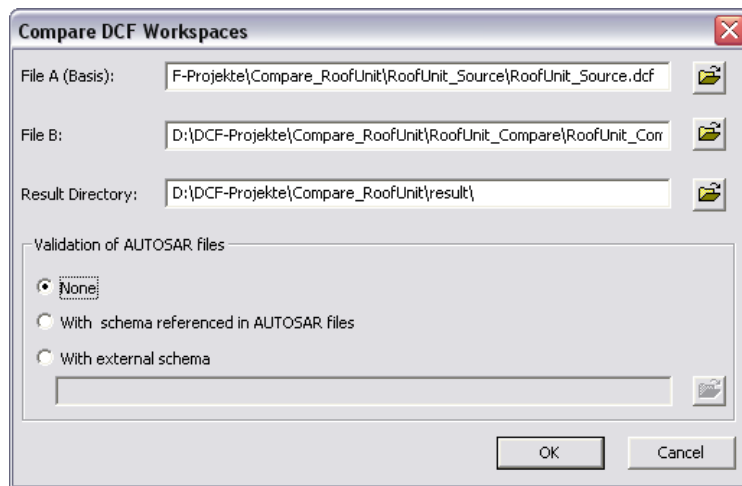
7.6 Compare AUTOSAR files of DCF workspaces

Select **File | Compare DCF** or use short cut  to compare all same AUTOSAR files which are referenced in two DCF files.



Select two existing DCF files and define a directory in which the result files of type “dvdiff” may be written.

Select whether the AUTOSAR files are to be validated or not. If you choose validate “With external schema” you have to define a XSD file to validate against.



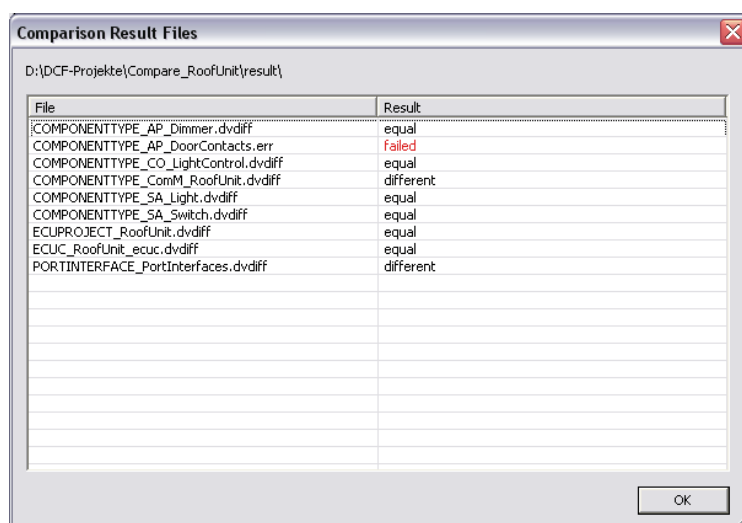
In “Comparison Result Files” dialog a list of result files and their result status is shown.

Note: Files at same location are not compared.

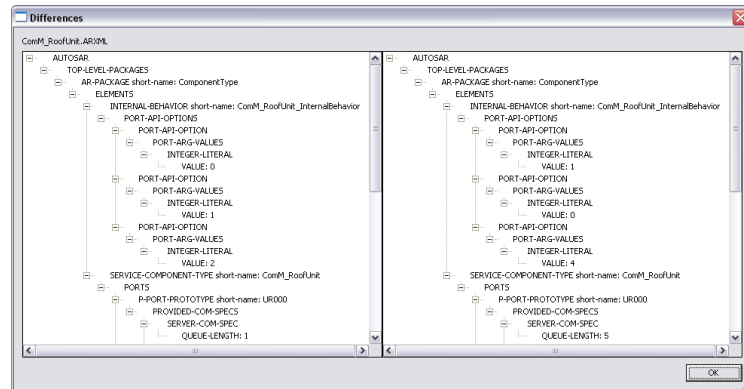
Result “equal”: No Changes

Result “different”: Changed

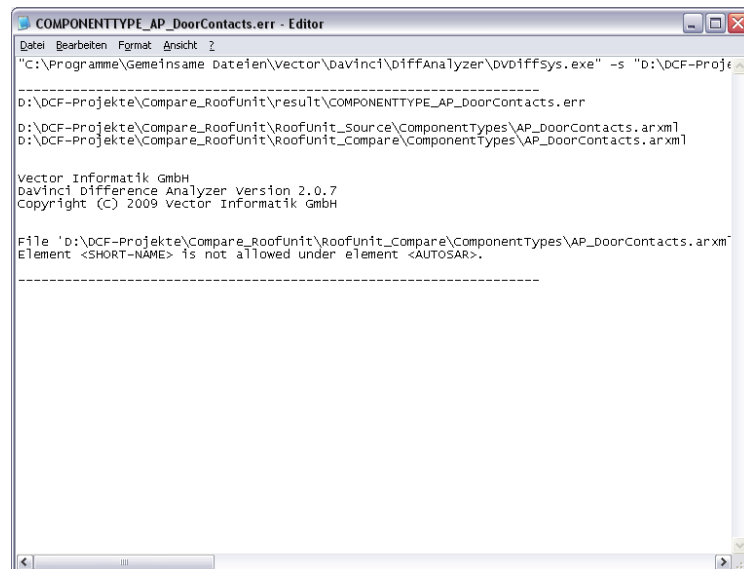
Result “failed”: Comparison failed. The results are written into an error file.



Double click a table row of the “Comparison Result Files” dialog to show the result file of the comparison in “AUTOSAR XML DifferenceView”.



or the error file of a failed comparison.



7.7 Command line usage of comparison

You can run the DCFUtility comparison of DCF files by the following command:
 DVDUtility.exe -s <dcffile1> -c <dcffile2> -o <resultdir>

■ parameters

- -s <dcffile1> source file.
- -c <dcffile2> DCF file which is compared with the source file.
- o <resultdir> path of result directory which contains the difference files of all AUTOSAR files contained in both <dcffile1> and <dcffile2>

8 Tips and Tricks

8.1 When to use DEV or DCF?

You should use the DaVinci Developer Workspace in case you need a fast, local saving/loading of your design data.

You should use the DaVinci Configuration File in case you want to manage the design data as AUTOSAR compliant files in an external CM system.

8.2 Working with temporary DEV

Saving and loading of a DCF usually takes more time than saving and loading of a DEV of the same content. If you e.g. work for several days on your design data without having to check-in/commit your work into the CM system, it may be useful to work like this:

- Get the DCF from the CM system into your local working copy (e.g.
`c:\work\mySystem.dcf`)
- Save the DCF as temporary DEV (e.g. `c:\tmp\mySystem.DEV`)
- Work on the temporary DEV
- Save the temporary DEV back into the DCF in the working copy (e.g.
`c:\work\mySystem.dcf`)

9 Contact

Visit our website for more information on

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

www.vector.com