MICROSAR DB Update

Technical Reference

Description

Version 1.9

Authors	Benjamin Gottschalk, Mario Kunz, Matthias Bannholzer
Status	Released

Document Information

History

Author	Date	Version	Remarks
Benjamin Gottschalk	2009-12-10	1.0	Creation
Benjamin Gottschalk	2009-12-17	1.1	Adapted Reference Documents
Benjamin Gottschalk	2010-01-13	1.2	Added examples for validation and fix mode
Benjamin Gottschalk	2010-03-31	1.3	! ! !
Manuela Scheufele	2010-06-24	1.4	Add Appendix Supported Use Cases
Benjamin Gottschalk	2010-09-27	1.5	ESCAN00045603
Benjamin Gottschalk	2011-01-14	1.6	Added chapter 5
Mario Kunz	2011-10-31	1.7	ESCAN00054591, ESCAN00054592
Matthias Banholzer	2012-09-14	1.8	ESCAN00050371
Mario Kunz	2013-03-27	1.9	ESCAN00065153

Reference Documents

No.	Source	Title	Version
[1]	VECTOR	TechnicalReference_Asr_GenTool_CsAsrInitialEcuC.pdf	1.0
[2]	VECTOR	DbUpdateConsoleApplication.exe.config	

Please note

We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Introduction

The MICROSAR DB Update is the Vector solution for performing updates based on an existing ECU Configuration Description. An example workflow is shown below in Figure 1-1.

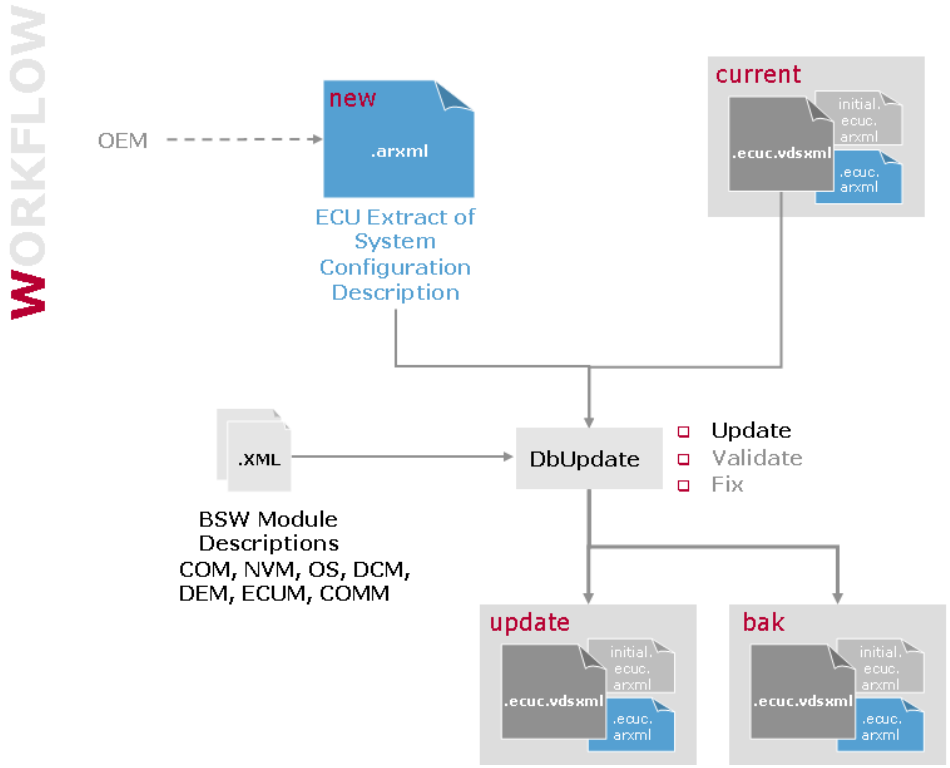

Figure 1-1 Workflow for updating an ECUC project

The ECUC project is generated by the MICROSAR Initial ECUC Generator. This project is modified by working.

If a new ECU Extract of System Configuration Description is released by the OEM this changes must find their way into the ECUC file you are currently working with. So an update of the existing (already partly or fully configured) ECUC project is necessary.

The MICROSAR DB Update needs to know the BSWMD files for matching vendor specific module configurations (with a DEFINITION- ! ! / ! 0 0 *! ! ! ! module configuration (with a DEFINITION- ! ! / ! 0 0 *!

The MICROSAR DB Update needs all definitions of the module configurations containing in the existing ECUC project; otherwise the update cannot be performed for theses module configurations / containers / parameters or references.

For this reason, the component directory (where the BSWMD files are stored) has to be specified for performing a complete update. Alternatively, the path to the components-directory may get read out of the *.vdsxml-file automatically by this tool, if that path really exists physically on the actual PC. This path will be written by GENy when saving the configuration the first time. However, if the configuration has been configured on one PC but will be updated on another, the path to the correct components-directory may have changed and must be specified explicitely.

2 Basic concepts

The updater has to deal with a quadruplet of ECUC files:

-
- Existing Active ECUC File
- Existing Initial ECUC File
- New Active ECUC File
- New Initial ECUC File

! ! ! ! ! ! ! ! ! ! ! ! ! - details at all. The tool is operating on fundamental ECUC objects, like Containers, Parameters and References, not on semantically objects like Channels or Identities. So, during update, decisions are based on a strict algorithm which evaluates the ECUC objects in the four mentioned ECUC files.

When starting the tool in the recommended use-case, at first the MICROSAR Initial ECUC Generator will be called, which creates a temporary new ECUC project out of the new ECU-Extract of System Configuration Description of the OEM. This project will be deleted at the end of the update- ! ! ! ! ! /

The second step is the real update of the existing ECUC /! ! ! ! ! ! newly created Initial ECUC File to the original working directory, so the existing Initial ECUC File ! ! /! ! ! ! ! ! ! ! ! ! ! restrictions. All ECUC objects which are located in the existing Active ECUC File will be adapted to their pendants in the new Initial ECUC File. All other ECUC objects in the existing Active ECUC File are user- ! ! ! ! /! ! ! ! ! ! ! ! all ECUC objects which only exist in the new Active ECUC File. These objects will just be copied to the existing (updated) Active ECUC File.

The matching of ECUC objects to each other in the different ECUC files can only be done through their definitions and ! ! ables through their ShortName.

3 Usage

The default usage help on the command line appears by calling the executable without parameters or with parameter **-h**.

The MICROSAR DB Update calls the MICROSAR Initial ECUC Generator internally for generating a temporary ECUC Configuration based on the new ECU Extract of System Configuration Description. This temporary ECUC Configuration will be updated independent of the existing one.

3.1 Use Cases

This tool has been implemented for the purpose of updating existing ECUC projects based on a first version of an ECU Extract of System Configuration Description of an OEM to a second version (which includes just minor changes, like some new / removed messages or signals).

This tool is **NOT** intended for the following use cases and does **NOT support** them:

- Complete change of the ECU Extract of System Configuration Description (complete change of communication matrix)
- Update of a standard use case project to a multiple ECU use case project
- Adding a new identity to an existing configuration
- Removing of identities in a multiple identity configuration project.
- Update of an existing ECUC project for ECU1 to a new ECU Extract of System Configuration Description of the OEM which is specified for another ECU2
- Update from "single node" to a "multiple node". To change the configuration during the development will not work.

3.2 Arguments

The arguments of the MICROSAR DB Update are listed in the following table.

Argument	Description
-c, --componentsDir=PATH	Makes the following parameter PATH being interpreted as the component directory where important BSWMD files are stored.
-f, --fix=FILE	Makes the following parameter FILE being interpreted as an ECUC project, which will be validated and updated.
-h, --help	Display the help.
-i, --initialEcuCGen=PATH	Makes the following parameter PATH being interpreted as the directory, where the MICROSAR Initial ECUC Generator is stored. If this option is not specified, the MICROSAR DB Update assumes that the MICROSAR Initial ECUC Generator is placed in the same directory.
-l, --logFile=FILE	Makes the following parameter FILE being interpreted as a log file. If **-l** is not specified, a default log file will be generated.
-n, --noBackup	Creates no backup of the existing ECUC project.
-u, --updateWithProject=FILE	Makes the following parameter FILE being interpreted as VDSXML file of the new ECUC project.
-v, --verbose	Display additional descriptions and help.
-V, --Validate=FILE	Makes the following parameter FILE being interpreted as an ECUC project, which will be validated.
-w, --workingProject=FILE	Makes the following parameter FILE being interpreted as VDSXML file of the existing ECUC project.

Table 3-1 Command line arguments

The following arguments are possible for calling the MICROSAR Initial ECUC Generator. These parameters will only be forwarded to the generator and are not needed by the MICROSAR DB Update itself. For detailed information about the MICROSAR Initial ECUC Generator see [1].

Argument	Description
-a, --asrVersion=VERSION	Makes the following parameter VERSION being interpreted as the AsrVersion, the ECUC files should be generated in. The VERSION should be the XML namespace version of the AsrVersion and should be specified in the following manner: e.g. 3.0.2.
-b, --bswmd=FILE	Makes the following parameter FILE being interpreted as a BSWMD file.
-m, --merge	Performs a merge of all specified ECU Extract of System Configuration Description files to one output ECUC project. For this option, you have to specify at least two valid ECU Extract of System Configuration Description files. This parameter should be only used, when using the Identity Manager feature. Merging of ECU Extract of System Configuration Description files is just supported in this context.
-o, --overlayConfig=FILE	Makes the following parameter FILE being interpreted as the XML configuration file for buffer overlay in Multiple ECU use case. This parameter should be only used, when using the Identity Manager feature.
-p, --projectName=NAME	Makes the following parameter NAME being interpreted as project name.

Table 3-2 Command line arguments for calling the MICROSAR Initial ECUC Generator

Info
All arguments which describe a file or a path can be specified relatively or absolutely.

If you do not use parameter **-n**, the MICROSAR DB Update creates a backup of the existing ECUC project in a new generated directory. (Directory of the existing VDSXML file). The backup folder has the following naming convention:

backup_<hh>-<mm>-<ss>_<YYYY>-<MM>-<DD>, e.g.: backup_13-22-35_2009-12-07

After the backup has been performed successfully the existing ECUC project will be overwritten with the updated ECUC project. This is due to the fact that many tools may refer to exactly this VDSXML file at this location.

3.2.1 Application Configuration File

The MICROSAR DB Update is supplied with an application configuration file called **DbUpdateConsoleApplication.exe.config** (see [2]). This file is written as XML file and stored in the same directory as the executable.

The file contains a configuration section for specifying the relative path to the directory, where the MICROSAR Initial ECUC Generator is located. After the installation of your delivery package this path is already set correctly.

Edit
If the directory structure changed, this file has to be edit for adapting the path to the MICROSAR Initial ECUC Generator.

If command line parameter **-i** is defined, the path is taken from this parameter and not from the configuration file.

3.2.2 Standard Use Case

This is the default use case for updating an existing ECUC project based on a new ECU Extract of System Configuration Description.

In that case the MICROSAR Initial ECUC Generator is called automatically by the MICROSAR DB Update, to generate a new ECUC Project based on the new ECU Extract of System Configuration Description.

Command line:

DbUpdateConsoleApplication.exe w <vdsxml-file of the working EcuC project> -c <components-directory> <NewEcuExtractOfSystemConfigurationDescription>

Example
DbUpdateConsoleApplication.exe w D:\test\workingProject.ecuc.vdsxml c D:\test\components D:\test\myNewSystemExtract.arxml

Argument	Description
w D:\test\workingProject.ecuc.vdsxml	This is the current ECUC project.
-c D:\test\components	Specifies the component directory (including the BSWMD files, which are important for updating the current ECUC project).

D:\test\myNewSystemExtract.arxml	This is the new ECU Extract of System Configuration Description. This file is taken as input for updating the current ECUC project.

Table 3-3 Arguments standard use case

3.2.3 Updating project based on two ECUC projects

This use case makes an updated ECUC project based on an existing ECUC project and a new ECUC project possible.

Caution

This use case is NOT recommended. In most cases, the new ECU Extract of System Configuration Description is available, so the standard use case is given (see chapter 2.1.2 for details).

In some cases, the new ECU Extract of System Configuration Description may not be available, but you may have a new ECUC project. For these cases, you should use this use case.

Command line:

DbUpdateConsoleApplication.exe w <vdsxml-file of the working EcuC project> -c <components-directory> -u <new EcuC project>

Example

DbUpdateConsoleApplication.exe w D:\test\workingProject.ecuc.vdsxml c D:\test\components u D:\test\newProject.ecuc.vdsxml

Argument	Description
w D:\test\workingProject.ecuc.vdsxml	This is the current ECUC project.
-c D:\test\components	Specifies the component directory (including the BSWMD files, which are important for updating the current ECUC project).
u D:\test\newProject.ecuc.vdsxml	This is the new ECU project (the output of the MICROSAR Initial ECUC Generator).

Table 3-4 Arguments updating project based on two ECUC projects

4 Consistency of an existing ECUC project

4.1 Validation mode

When calling the MICROSAR DB Update with parameter **-V**, the

5 What to do after an update

There are some steps after an update, which has to be checked (and perhaps configured) manually.

5.1 Dcm

A DcmConnection refers to GlobalPdus. These GlobalPdus are derived from the corresponding DcmIPdus in SystemExtract.

If there was a configured DcmConnection in the ecuc-project before update AND the ShortName of a DcmIPdu has changed in the new SystemExtract, the DcmConnection is not fully described anymore after update (due to lack of the GlobalPdus, which have changed, too) and will be deleted during first load of the updatet ecu configuration in GENy. These DcmConnections must be added manually again after update in GENy.

6 Appendix

6.1 supported Use Cases

Use Cases
Signal groups CAN
> Add signal to signal group
> Remove signal from signal group
> Move signals between signal groups
> Change init values of signals within signal groups
> Change position of signal within signal groups
> Remove signal group
> Change signal group direction from RX to TX
> Change signal group direction from TX to RX
> Move signal group between PDUs
> Move signal group between channels
> Move signal group between channels
> Add signal group
PDUs CAN
> Delete PDU
> Change direction of PDU from RX to TX
> Change direction of PDU from TX to RX
> Move PDU between channels
> Change PDU timing from cyclic to direct
> Change PDU timing from direct to cyclic
> Change PDU timing from Mixed (direct, cyclic) to direct
> Add PDU
PDUs LIN
> Change direction of PDU from Rx to Tx
> Change direction of PDU from Tx to Rx
> Move PDU between channels (Chan1 to Chan2)
> Add sporadic PDU

Use Cases
> Add event-triggered PDU
> Add unconditional PDU
> Delete sporadic PDU
> Delete event-triggered PDU
> Delete unconditional PDU
> Change order index of associated frames in a sporadic frame
PDUs FlexRay
> Delete PDU
> Change direction of PDU from RX to TX
> Change direction of PDU from TX to RX
> Move PDU between channel A and B
> Move PDU between channel B and A
> Add PDU
> Move PDU between frames
Signals CAN
> Add signal
> Remove signal
> Change init value of signal
> Change bit position of signal
> Move signals between PDUs
> Move signals between PDUs and channels
Signals LIN
> Add signal
> Remove signal
> Change init value of signal
> Change bit position of signal
> Move signals between PDUs
> Move signals between PDUs and channels

Use Cases
Signals FlexRay
> Add signal
> Remove signal
> Change init value of signal
> Change bit position of signal
> Move signals between PDUs within different frames
> Move signals between PDUs and channels (A/B)
Frames CAN
> Change ID
Frames LIN
> Change ID of Unconditional Frame
Frames FlexRay
> Change Slot ID of dynamic frame
> Change Slot ID of static frame
> Add dynamic frame
> Add static frame
> Delete dynamic frame
> Delete static frame
> Change payload of frame
Gateway Signal Mapping CAN
> Add new signal mapping
> Remove signal mapping
> Change signal mapping
> Signal move between I-Pdus
Gateway Signal Mapping LIN
> Add new signal mapping
> Remove signal mapping
> Change signal mapping
> Signal move between I-Pdus
Gateway Signal Mapping FlexRay
> Add new signal mapping

Use Cases
> Remove signal mapping
> Change signal mapping
> Signal move between I-Pdus
Gateway I-Pdu Mapping CAN
> Add new I-Pdu Mapping
> Remove I-Pdu Mapping
> Change I-Pdu Mapping
Gateway I-Pdu Mapping LIN
> Add new I-Pdu Mapping
> Remove I-Pdu Mapping
> Change I-Pdu Mapping
Gateway I-Pdu Mapping FlexRay
> Add new I-Pdu Mapping
> Remove I-Pdu Mapping
> Change I-Pdu Mapping
Network parameters LIN
> Change Baud Rate for LINNetwork1
> Change Baud Rate for LINNetwork2
Network parameters FlexRay
> Change Baud Rate for FRNetwork1
> Increase number of static slots by changing cycle length and/or macrotick duration (high level parameters)
Channels CAN
> Change channel name CANNetwork0
> Change channel name CANNetwork1
> Add channel CANNetwork3
> Delete channel CANNetwork4
Channels LIN
> Change channel name LINNetwork0
> Change channel name LINNetwork1
> Add channel LINNetwork3
> Delete channel LINNetwork4

Use Cases
Cluster FlexRay
> Change cluster name FRNetwork0
I-PDU-Groups CAN
> Add I-PDU-Group for Node ECUUpdateTest
> Delete I-PDU-Group for Node ECUUpdateTest
> Add hierarchical I-PDU-Group
> Delete hierarchical I-PDU-Group
> Move IPDUs between I-PDU-Groups
I-PDU-Groups LIN
> Add I-PDU-Group for Node ECUUpdateTest
> Delete I-PDU-Group for Node ECUUpdateTest
> Add hierarchical I-PDU-Group
> Delete hierarchical I-PDU-Group
> Move IPDUs between I-PDU-Groups
I-PDU-Groups FlexRay
> Add hierarchical I-PDU-Group
> Delete hierarchical I-PDU-Group
LIN Schedule tables
> Add Schedule Table
> Delete Schedule Table
> Change slot position of frames within schedule table
> Move frames between schedule tables
> Change schedule table name
> Change table id (priority) of schedule table
> Change delay of frames within schedule table
> Change Run Mode of schedule table

Table 5-1 Supported Use Cases

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
ECU Extract of System Configuration Description	The ECU Extract of System Configuration Description is a subset of the System Configuration Description. It only contains information relevant for one specific ECU.

7.2 Abbreviations

Abbreviation	Description
ECU	Electronic Control Unit
ECUC	ECU Configuration

8 Contact

Visit our website for more information on

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

www.vector.com