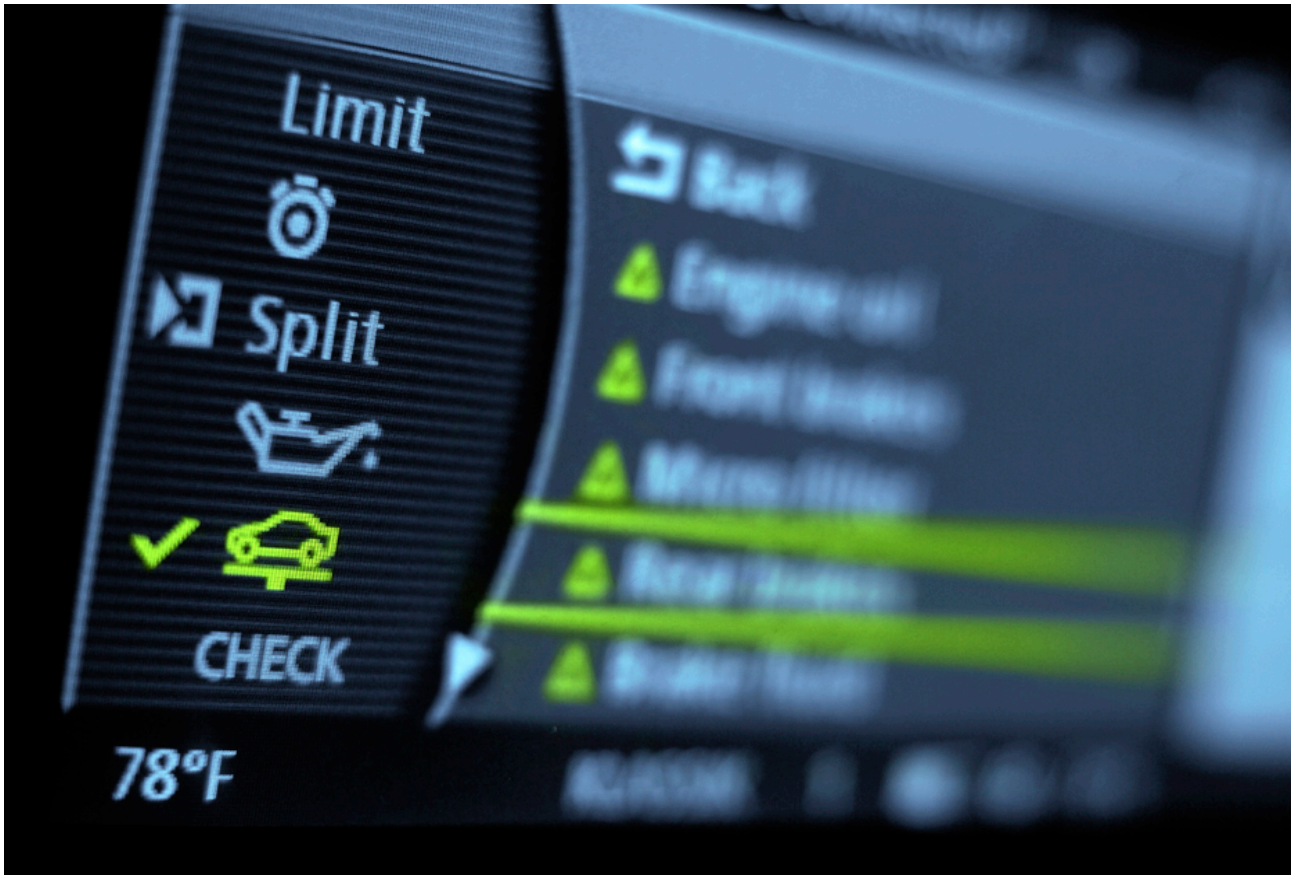
Elektrobit

EB tresos® Safety RTE user's guide

EB tresos Safety RTE

Date: 2022-11-16, Document version 2.1, Status: RELEASED



Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

Table of Contents

1. Document history

Version	Date	Author	State	Description
0.1.0	2015-06-01	Elektrobit Automotive GmbH	DRAFT	Initial version
0.2.0	2015-06-09	Elektrobit Automotive GmbH	PROPOSED	Added chapters *About this documentation* and *Safe and correct use*
0.2.1	2015-08-28	Elektrobit Automotive GmbH	PROPOSED	Added chapters *Background information* and *Rte generation manual*
0.2.2	2015-09-01	Elektrobit Automotive GmbH	PROPOSED	Added chapter *Begin here*
0.2.3	2015-09-14	Elektrobit Automotive GmbH	PROPOSED	Added appendix *Open source licenses*
0.2.4	2015-09-18	Elektrobit Automotive GmbH	PROPOSED	Reworked chapter *Using the EB tresos Safety RTE*
1.0.0	2015-12-04	Elektrobit Automotive GmbH	RELEASED	► Improved chapter *Preparing the test application*. ► Moved almost complete chapter *Using the EB tresos Safety RTE* to the *EB tresos Safety RTE Safety Manual* except from chapter *Generating the RTE test cases*. ► Renamed chapter *Generating the RTE test cases* to *Using the RteCV*. ► Updated all links to moved chapters.
1.0.1	2016-05-23	Elektrobit Automotive GmbH	RELEASED	Incorporated review findings from TE.
1.0.2	2016-07-20	Elektrobit Automotive GmbH	RELEASED	Updated bibliography entry for the *EB tresos Studio user's guide*.

Version	Date	Author	State	Description
1.0.3	2016-09-13	Elektrobit Automotive GmbH	RELEASED	▶ Introduced a new command line parameter for plugin selection. ▶ Introduced the new generation mode `ALL`.
1.1.0	2017-03-24	Elektrobit Automotive GmbH	RELEASED	Created release for EB tresos Safety RTE 3.0.2.
1.1.1	2017-05-03	Elektrobit Automotive GmbH	PROPOSED	▶ Updated versions to ACG7. ▶ Added chapter "Performing RTE code checks with the RteCV".
1.1.2	2017-05-19	Elektrobit Automotive GmbH	RELEASED	▶ Incorporated review findings from TE. ▶ Set document state to released.
1.1.3	2017-08-25	Elektrobit Automotive GmbH	PROPOSED	▶ Corrected the usage of BaseTypePropertyFile option.
1.1.4	2017-11-09	Elektrobit Automotive GmbH	RELEASED	▶ Updated reporting, so it now reflects the new matched code highlighting feature.
1.1.5	2017-03-21	Elektrobit Automotive GmbH	RELEASED	▶ Added new section "Evaluating the results of the static code analysis".
1.1.6	2018-08-21	Elektrobit Automotive GmbH	RELEASED	▶ Added new section "Evaluating the results of the test application".
1.1.7	2018-11-02	Elektrobit Automotive GmbH	RELEASED	▶ Updated mode details to include USERDEFINED mode for pattern checks ▶ Added details for the new plugin UserPatternChecker
2.0.0	2019-04-18	Elektrobit Automotive GmbH	RELEASED	▶ Updated chapter "Using the RteCV" for the new RTE data model. ▶ Added RTE data model report plugin
2.0.1	2019-06-14	Elektrobit Automotive GmbH	PROPOSED	▶ Updated details for verification of data model report plugin. ▶ Added details of html data model report files. ▶ Fixed review findings for the new RTE data model report.

Version	Date	Author	State	Description
2.0.2	2019-08-14	Elektrobit Automotive GmbH	RELEASED	Set to released: ASCRTECV-1550
2.0.3	2019-09-06	Elektrobit Automotive GmbH	PROPOSED	▶ Changed mode command line parameter default value to ALL. ▶ Extended plugin selection description to describe the relation to the mode command line parameter.
2.0.4	2019-11-26	Elektrobit Automotive GmbH	PROPOSED	▶ Removed pattern extension restriction for user defined patterns. ▶ Adapted section for validating the RTE data model report. ▶ Added section for validating the RTE data model report to section for generated tests ▶ Adapted section for validating the RTE data model report after RTE data model report rework.
2.0.5	2019-11-27	Elektrobit Automotive GmbH	RELEASED	▶ Set to released: ASCRTECV-1562 ▶ Rephrase explanatory patterns to improve readability. ▶ Transform instructions into step-by-step task patterns.
2.0.6	2020-01-14	Elektrobit Automotive GmbH	RELEASED	▶ Update open source licenses in Appendix B. ▶ Update to Apache v2.0 ▶ Add MIT license
2.0.7	2020-04-14	Elektrobit Automotive GmbH	RELEASED	▶ Added details about optional pattern in UserDefinedPatterns. ▶ Set to released: ASCRTECV-1728
2.0.8	2020-08-28	Elektrobit Automotive GmbH	RELEASED	▶ replaced RTE data model by serialized RTE intermediate model ▶ Set to released: ASCRTECV-1921
2.0.9	2020-11-02	Elektrobit Automotive GmbH	PROPOSED	▶ Added source directories option to option file. ▶ Extended command line parameters for the code introspection checker.

Version	Date	Author	State	Description
				▶ Extended command line parameters for C pre-processor defines and include directories.
2.0.10	2020-12-17	Elektrobit Automotive GmbH	RELEASED	Set to released: ASCRTECV-2025
2.0.11	2021-05-10	Elektrobit Automotive GmbH	PROPOSED	Extended usage for introspection checker.
2.0.12	2021-05-19	Elektrobit Automotive GmbH	RELEASED	Set to released: ASCRTECV-2152
2.0.13	2022-08-21	Elektrobit Automotive GmbH	PROPOSED	▶ Added command line parameter "version" ▶ Updated command line details for introspection checker
2.0.14	2022-08-24	Elektrobit Automotive GmbH	RELEASED	Set to released: ASCRTECV-2436
2.1	2022-11-16	Elektrobit Automotive GmbH	RELEASED	▶ Removed Test Generator details ▶ Added command line parameter "disable-data-model-validation" ▶ The document version scheme was changed ▶ Removed Mass Production review process references ▶ Set to released: ASCRTECV-2503

Table 1.1. Document history

2. Begin here

The purpose of this document is to provide you with the information necessary to use *EB tresos Safety RTE*.

It is assumed that you are familiar with the `RTE` module. For more information on the `RTE`, see [AUTO-COREUSRDOC] chapter *User's guide → Run-Time Environment*.

Chapter 2, “Begin here“ (this chapter) describes the structure of the document.

Chapter 3, “About this documentation“ describes the style and typography of this document.

Chapter 4, “Safe and correct use“ describes how to use EB tresos Safety RTE safely.

Chapter 5, “EB tresos product line support“ lists support contact information.

Chapter 6, “Background information“ introduces the concept of EB tresos Safety RTE as well as the features of the AUTOSAR RTE.

Chapter 7, “Using the RteCV“ describes how to use the RteCV.

3. About this documentation

Welcome to the EB tresos Safety RTE user's guide.

This chapter provides you with typographical and style conventions used throughout this documentation. It also defines the usage of special fonts in the documentation.

3.1. Typography conventions

Throughout the documentation you see that words and phrases are displayed in bold or italic font, or in mono space font.

To find out what these conventions mean, consult the following table. All default text is written in font Arial Regular without any markup.

Convention	**Item is used**	**Example**
Arial italics	to define new terms	The *basic building blocks* of a configuration are module configurations.
Arial italics	to emphasize	If your project's release version is mixed, all content types are available. It is thus called *mixed version*.
Arial italics	to indicate that a term is explained in the glossary	...exchanges *protocol data unit*s (*PDU*s) with its peer instance of other ECUs.
Arial boldface	for pop-up windows names	The **Bulk Change** editor enables you to change data in bulk.
Arial boldface	for menus and submenus	Choose the **Options** menu.
Arial boldface	for buttons	Select **OK**.
Arial boldface	for keyboard keys	Press the **Enter** key
Arial boldface	for keyboard key combinations	Press **Ctrl**+**Alt**+**Delete**
Arial boldface	for commands	Convert the XDM file to the newer version by using the **legacy convert** command.
Monospace font (Courier)	for file and folder names, also for chapter names	Put your script in the `function_name\abc-folder`
Monospace font (Courier)	for code	`CC_FILES_TO_BUILD =(PROJECT_-PATH)\source\network\can_node.-c CC_FILES_TO_BUILD += $(PROJECT_-PATH)\source\network\can_config.c`

Convention	Item is used	Example
Monospace font (Courier)	for function names, methods, or routines	The cos function finds the cosine of each array element. Syntax line example is `MLGetVar ML_var_name`
Monospace font (Courier)	for user input/indicates variable text	Enter a `three-digit prefix` in the menu line.
Square brackets [ ]	to denote optional parameters; for command syntax with optional parameters	`insertBefore [<opt>]`
Curly brackets {}	to denote mandatory parameters; for command syntax with mandatory parameters (in curly brackets)	`insertBefore {<file>}`
Three dots …	to indicate further parameters; for command syntax, indicates further parameters	`insertBefore [<opt>…]`
A vertical bar |	to separate parameters in a list from which one parameters must be chosen or used; for command syntax, indicates a choice of parameters	`allowinvalidmarkup {on|off}`
Warning	to show information vital for the success of your configuration	**WARNING**  **This is a warning** This is what a warning looks like.
Note	to give additional important information on the subject	**NOTE**  **This is a note** This is what a note looks like.
Tip	to provide helpful hints and tips	**TIP**  **This is a tip** This is what a tip looks like.
Example	to demonstrate or illustrate information	 **Example 3.1. This is an example** This is what an example looks like.

4. Safe and correct use

4.1. Intended usage of EB tresos Safety RTE

EB tresos Safety RTE is intended to be used in automotive projects based on AUTOSAR. For more information about the AUTOSAR consortium, see www.autosar.org.

4.2. Possible misuse of EB tresos Safety RTE

This section provides you with information and examples about what can be made wrong in the usage of EB tresos Safety RTE. If EB tresos Safety RTE is used in a wrong way, Elektrobit Automotive GmbH is not liable for this misuse.

- Use of this product without taking the appropriate risk-reduction measures throughout the entire development phase can result in unexpected behavior. Elektrobit Automotive GmbH is not liable for this misuse.
- If you use the product in architectures that are not defined by the AUTOSAR consortium, the product and its technology may not conform to the requirements of your architecture.
- If you use a wrong version of the `RTE` or EB tresos Studio, it is likely that the *Rte Code Verifier (RteCV)* does not ensure that the generated `RTE` fulfills the assumed safety requirements. For more information about the supported `RTE` and EB tresos Studio, see the EB tresos Safety RTE Safety Manual [SAFETYMAN] chapter *Using EB tresos Safety RTE safely → Getting started → Prerequisites*.
- If you do not use the product according to the instructions of this document and to the EB tresos Safety RTE Safety Manual [SAFETYMAN], it is not ensured that the generated `RTE` fulfills the assumed safety requirements.

  Some examples for possible misuses are listed in the following list:

  - The serialized RTE intermediate model provided to the RteCV is outdated, for example by disabling the `/Rte/RteGenaration/RteDataModelExport` configuration parameter. For more information about how to use the RteCV, see Chapter 7, “Using the RteCV“.
  - The `RTE` contains unsupported features and you do not validate them according to the EB tresos Safety RTE Safety Manual [SAFETYMAN] chapter *Using EB tresos Safety RTE safely → Validating your generated RTE → Validating unsupported features of the RTE*.
  - The system application does not fulfill the assumptions of EB tresos Safety RTE specified in the EB tresos Safety RTE Safety Manual [SAFETYMAN] chapter *Application constraints and requirements → Assumptions*.

WARNING **Possible misuse and liability**

You may use the software only as in accordance with the intended usage and as permitted in the applicable license terms and agreements. Elektrobit Automotive GmbH assumes no liability and cannot be held responsible for any use of the software that is not in compliance with the applicable license terms and agreements.

4.3. Target audience and required knowledge

The intended audiences of this document are the following:

- Application developers
- Software *integrators*

The audiences should at least fulfill the following requirements:

- Programming skills in C
- Experience in programming AUTOSAR-compliant ECUs

4.4. Suitability for mass production

Use of this product without taking the appropriate risk-reduction measures throughout the entire development phase can result in unexpected behavior.Elektrobit Automotive GmbH is not liable for this misuse.

5. EB tresos product line support

https://www.elektrobit.com/support

6. Background information

EB tresos Safety RTE implements an AUTOSAR RTE and is based on *EB tresos AutoCore Generic 8 RTE*. In addition to that, it provides means to validate the generated RTE code to allow its usage in ASIL-D environments for a specific subset of features.

For general information about EB tresos Safety RTE, see chapter *About EB tresos Safety RTE* of the [SAFETYMAN]. For specific information, see section *Description of EB tresos Safety RTE* of the chapter *About EB tresos Safety RTE*.

For specific information about the features of the AUTOSAR RTE, see chapter *User's guide → Run-Time Environment → Background information* of the [AUTOCOREUSRDOC].

NOTE

Restricted feature set

EB tresos Safety RTE supports only a subset of the features described in the EB tresos AutoCore Generic 8 RTE user's guide for the usage in system applications with a safety assignment. For more information on the subset of supported features, see the [SAFETYMAN].

7. Using the RteCV

7.1. Overview

This chapter describes how to use the RteCV to verify the generated `RTE` according to the configuration. The intended target audience of this chapter are application developers and software integrators.

Section 7.2, “Introduction to RteCV” describes how to use the RteCV and its command line parameters.

Section 7.3, “Performing RTE static code checks with the RteCV” describes how to perform a static code analysis.

Section 7.4, “RteCV static code check examples” shows examples on how to execute the static code checks with the RteCV.

Section 7.5, “Verifying RTE features with user defined patterns” describes how to verify user defined patterns.

Section 7.6, “Created output folders and generated files” describes the folders and files generated by the RteCV.

For more information about the EB tresos Safety RTE workflow, see the [SAFETYMAN], chapter *Using EB tresos Safety RTE safely → EB tresos Safety RTE workflow*.

7.2. Introduction to RteCV

The RteCV is a tool to verify the generated `RTE` code and to ensure that the generated code can be used in system applications with a safety allocation.

The RteCV is a command line tool that expects command line parameters to get the necessary information for its execution. You can find a list with command line parameters in the Section 7.2.3, “Command line parameters”. Additionally, the RteCV works with an option file to replace some command line parameters. For more information on the option file, see Section 7.2.4, “Using an option file”.

After the RTE has generated the RTE data model, you can import this data model to the RteCV. This RTE data model is used by all code verifier plugins as a basis for their checks. Ensure, that you import the correct data to the RteCV. The RteCV generates an RTE data model report that shows all relevant data and helps you to verify the correctness of the data.

The RteCV uses different code verifier plugins. These plugins are implementing the checks, that are loaded at run-time. You can either execute one code verifier plugin to run one specific test or execute all available plugins to run all tests. For more information about the available code verifier plugins, see Section 7.2.5, “Code verifier plugins”.

7.2.1. Exporting the serialized RTE intermediate model

The exported serialized RTE intermediate model is a specific view of the RteCV on the module configuration files and *system description* of your *system application*. You can enable the export via the following configuration parameter in you RTE module configuration:

`/Rte/RteGenaration/RteDataModelExport`

Set this parameter to `true` then the RTE generator generates the RTE data model automatically before it generates of the RTE code. You can find the generated `RteDataModel.xml` file in the `generated/doc` folder in your specified EB tresos Studio output directory.

7.2.2. Using the command line

To start the RteCV, execute the batch file `RteCV.bat`, which is installed as part of EB tresos Safety RTE. You can find `RteCV.bat` in the folder `tools` in the installation folder of EB tresos Safety RTE in your EB tresos Studio installation.

The command line to execute the RteCV is as follows:

```
RteCV.bat [PARAMETER]...
```

7.2.3. Command line parameters

The following table lists all command line parameters supported by the RteCV.

Command line parameter	Default value	Description
`-?`, `-h`, `--help`	`---`	Prints a help message.
`--dm`, `--data-model FILE`	`---`	Imports the RTE intermediate model file FILE. The file shall contain the serialized RTE intermediate model exported during the RTE generation phase.
`-sdir`, `--source-dir DIRECTORY [,DIRECTORY]...`	`---`	Imports all files from `DIRECTORY` or from a comma-separated list of directories. The RteCV assumes that all these files are source files to be statically verified, including those that are generated by the `RTE` and `OS`. Make sure that there are no other file types. All hidden files in the directory are ignored.
`-sfile`, `--source-files FILE [,FILE]...`	`---`	Imports the source file `FILE` or a comma- separated list of source files. These files shall contain the

Command line parameter	Default value	Description
		source code to be statically verified, including the `RTE` and `OS` source files.
`-m, --mode [ALL|CHECKER|INTROSPECT|USERDEFINED]`	`ALL`	Sets the mode of the *RteCV*. ► `ALL`: all the modes are triggered ► `CHECKER`: the code checker is triggered ► `INTROSPECT`: only the code introspection checks are triggered ► `USERDEFINED`: only the pattern checks are triggered
`-o , --output DIRECTORY`	`output/`	Sets the output directory `DIRECTORY` for the results, e.g. the `RteCV Verification Reports`.
`--opt, --option-file FILE`	`---`	Parses the provided file `FILE` for additional parameters.
`-p, --plugins PLUGIN-NAME [,PLUGIN-NAME]...`	`---`	This optional parameter selects zero or more code verifier plugins that are used to verify the generated RTE code. For more information on the available code verifier plugins, see Table 7.3, “Code verifier plugins”. If this parameter is not set, all installed plugins are used.
`-u, --user-defined-patterns FILE [,FILE]...`	`---`	This optional parameter provides a list of user-defined patterns that should be checked against RTE generated files. See detailed description Section 7.5, “Verifying RTE features with user defined patterns”
`-d, --cpp-defines DEFINE[=VALUE] [,DEFINE[=VALUE]]...`	`---`	This optional parameter provides a comma separated list of C preprocessor defines to the static code checker.
`-i, --include-dir DIRECTORY[,DIRECTORY]...`	`---`	This optional parameter provides a comma separated list of C preprocessor include directories to the static code checker. The C preprocessor include directories are searched for header files so that include statements in the C code can be resolved.
`-v, --version`	`---`	Shows the RteCV and the supported RTE version.

Command line parameter	Default value	Description
`--disable-data-model-validation`	`---`	This optional parameter deactivates the default validation during loading the data model.

Table 7.1. Command line parameters supported by the RteCV

WARNING

Provide all defines, include directories and source files to use the code introspection checker.

If you want to analyze your code with the code introspection checker, then you have to provide all your C preprocessor defines, include directories, all source and header files generated by the RTE including the source files containing the runnable implementations which you are using to build the project to the RteCV. Otherwise the results of the code introspection checker could be wrong or incomplete.

7.2.4. Using an option file

In addition to the command line parameters, the RteCV also accepts an option file based on the Java property file format, see command line option `--opt`. This file stores properties which are equivalent to the command line parameters. The following table lists the supported properties and shows their corresponding command line option.

Command line parameter	Option file property
`--dm, --data-model FILE`	`RteCV.Option.DataModelFile = FILE`
`-sfile, --source-files FILE [,FILE]`	`RteCV.Option.SourceFiles = FILE [,FILE]`
`-sdir, --source-dir DIRECTORY [,DIRECTORY]`	`RteCV.Option.SourceDirectories = DIRECTORY [,DIRECTORY]`
`-m, --mode [ALL|CHECKER|INTROSPECT|USERDEFINED]`	`RteCV.Option.Mode = [ALL|CHECKER|INTROSPECT|USERDEFINED]`
`-o, --output DIRECTORY`	`RteCV.Option.OutputDirectory = DIRECTORY`
`-p, --plugins PLUGIN-NAME [,PLUGIN-NAME]...`	`RteCV.Option.Plugins = PLUGIN-NAME [,PLUGIN-NAME]...`
`-u, --user-defined-patterns FILE [,Files]*`	`RteCV.Option.UserDefinedPatterns = FILE [,FILE]`
`-d, --cpp-defines DEFINE[=VALUE] [,DEFINE[=VALUE]]...`	`RteCV.Option.Defines = DEFINE[=VALUE] [,DEFINE[=VALUE]]...`
`-i, --include-dir DIRECTORY [,DIRECTORY]...`	`RteCV.Option.IncludeDirectories = DIRECTORY [,DIRECTORY]...`

Command line parameter	Option file property
`--disable-data-model-validation`	`RteCV.Option.DataModelDisableValidation`

Table 7.2. Option file properties supported by the RteCV

NOTE

Multiple properties and command line parameters

Unexpected behavior can occur if you combine command line parameters with corresponding option file properties.

Never combine command line parameters and option file properties for the same purpose.

You must choose if you want to use command line parameters or option file properties.

7.2.5. Code verifier plugins

The RteCV can load plugins at run-time. The table below lists all available code verifier plugins and their related modes. To select a subset of the installed plugins, pass the names of the required plugins to the command line parameter `--plugins` and set the mode appropriately for the required plugins via the command line parameter `--mode`. For more information, see Section 7.2.3, “Command line parameters”.

Name	Description	Modes
RteCVExt.CodeChecker_TS_TxDxM5I5R0	Executes the pattern based static code checker for the generated code of the configured RTE that will run on the target. For more information on how to run this plugin, see Section 7.3, “Performing RTE static code checks with the RteCV”	[ALL|CHECKER]
RteCVExt.UserPatternChecker_TS_TxDxM5I5R0	Executes the user defined pattern checker to check for matching patterns. For more information on how to run this plugin, see Section 7.5, “Verifying RTE features with user defined patterns”	[ALL|USERDEFINED]
RteCVExt.DataModelReport_TS_TxDxM5I5R0	Generates an XML based report of the imported RTE data model.	[ALL|CHECKER|INTROSPECT]
RteCVExt.Introspection_TS_TxDxM5I5R0	Executes the code introspection checker for the generated code of the configured RTE that will run on the target. run on the target. The AST based static code checker verifies the sender-receiver communication. Therefore it parses the provided C source code, the provided C preprocessor defines and	[ALL|INTROSPECT]

Name	Description	Modes
	the C preprocessor include directories to verify the following: ▶ It checks whether the correct buffers are written or read. ▶ It checks whether only the expected functions are accessing the buffers. ▶ It checks whether readers and writers can not interrupt each other during the access of the buffer.	

Table 7.3. Code verifier plugins

7.3. Performing RTE static code checks with the RteCV

To perform static code checks on the generated `RTE`, you must perform the following steps:

1. Execute the RteCV to generate the RTE data model reports and to perform the static code checks. For more information, see Section 7.3.1, “Executing the RteCV”.
2. Verify the RteCV log output to ensure that no problems were found. For more information, see Section 7.3.2, “Verifying the RteCV log output”.
3. Verify the generated RTE data model reports. For more information, see Section 7.3.3, “Verifying the RTE data model report”.
4. Evaluate the test results. For more information, see Section 7.3.4, “Evaluating the results of the static code checks”.

7.3.1. Executing the RteCV

To execute the RteCV, both pattern based and abstract syntax tree code checkers must be called with the batch file `RteCV.bat` in *ALL* mode as described in Section 7.2, “Introduction to RteCV”.

The RteCV needs the following parameters:

- The serialized RTE intermediate model provided via the `--dm` parameter.
- The generated RTE source files via the `-sdir` and `-sfile` parameter.
- The C preprocessor defines via the `--cpp-defines` parameter.
- The C preprocessor include directories via the `--include-dir` parameter.

- The `RteCVExt.DataModelReport_TS_TxDxM5I5R0`, `RteCVExt.CodeChecker_TS_TxDxM5I5R0` and `RteCVExt.Introspection_TS_TxDxM5I5R0` plugins via the `--plugins` parameter.

Optionally you can use other parameters like the output directory or an option file instead of command line parameters.

For information on example calls to perform the code checks, see Section 7.4, “RteCV static code check examples”

NOTE **RTE intermediate model import - not all file formats are supported**

- The serialized RTE intermediate model is the model which the RTE has exported during the RTE generation phase.
- It is not sufficient to use a well-formed XML file. You must use an XML file valid against the RTE schema. Other formats like *AUTOSAR XML (ARXML)* or *XDM* configuration files as they are used by EB tresos® are not supported.

The RTE intermediate model file `RteDataModel.xml` is automatically created by the RTE before the source code is generated. You can find the file in the `generated/doc` folder in your specified EB tresos Studio output directory.

7.3.2. Verifying the RteCV log output

You must check manually that the static code analysis was performed successfully after the RteCV has been executed.

Check the tool output by verifying the following items:

Step 1
Verify that the serialized RTE intermediate model file was imported.

Step 2
Verify that all generated source files, C preprocessor defines and C preprocessor include directories that are relevant for the RTE were imported.

Step 3
Verify that other command line parameters or option file properties were correctly read and logged to the output.

Step 4
Verify that there are no errors in the tool output. You must correct these errors in any case.

Step 5
Verify that the only warnings in the tool output are about ignoring unsupported features. If there are other warnings, make sure that they do not affect the static code analysis.

TIP **Log output and content**

The RteCV log output is written to:

- the console output (`STDOUT`)
- the log file `log.txt` in the current working directory.

7.3.3. Verifying the RTE data model report

The RTE data model is the basis for code checks. It has been created via serialized RTE intermediate model exported by the RTE during the generation. You must validate the RTE data model first, before an evaluation of the code check results is possible. Validate the RTE data model by using the reports that are generated from the RteCV via the data model report plugin. Find the link to the reports by navigating to your specified output directory and open the `reports.html` file in the reports folder. The resulting reports are attached under the heading RTE Data Model Reports in the `reports.html`.

For more information on how to validate the RTE data model, see [SAFETYMAN] chapter *Using EB tresos Safety RTE safely → Validating your generated RTE → Validating the RTE data model report*.

WARNING **Unverified or incomplete RTE data model**

Unverified or incomplete data models will lead to unexpected behavior of the RteCV.

The imported RTE data model is the basis for the static code checker.

Always verify the correctness and completeness of the displayed elements in the report manually.

Generated reports

The RTE data model report generates the following XML files to the `rtedatamodelreport` folder in your specified output directory:

- `Structural_View_Report.xml`: The structural view report contains the structural components of the system model like software components and their ports.
- `Functional_View_Report.xml`: The functional view report contains elements of the system model that are related to the functionality of the system like tasks, events, and runnables.
- `Detailed_View_Report.xml`: The detailed view report contains a consolidated list of data used by the RteCV.

The displayed elements show their configured names and the XML tag names are derived from the according system model or module configuration elements

Verify the correctness and completeness of the RTE data model:

Step 1
Make sure that no element is missing in the report.

Step 2
Make sure that no additional element is shown in the report.

7.3.4. Evaluating the results of the static code checks

Each static code checker verifies an `RTE` feature and executes one or more test suites. Each test suite can have one or more test cases that verify an aspect of the `RTE` feature. For example the API mapping or function implementation. Each test case can have one or more test steps that verify the fragments in the code.

- A *static code checker* contains the following structure:
  - 1...n *test suites*
    - 1...n *test cases*
      - 1...n *test steps*

The generated RteCV verification report are evaluated after all static code checks are executed. The test report `reports.html` is located in the `report` folder in your specified output directory.

The RteCV `reports.html` verification report provides indexes and links to the input files and to the test suites for the RTE features that are verified. The matched pattern is highlighted in the source file. You can navigate to the corresponding test case by a click on this highlighted pattern. The pattern highlighting helps the user to identify the parts of the RTE source files. These RTE source files are verified by the code checker. Each test suite chapter begins with a table with the overall test results of the executed test cases of a code checker test. The results of a test case can have two different status:

- *passed*: the test case was successful and all contained code patterns match to the generated RTE code.
- *failed*: the test case was not successful and one or more contained code patterns did not match to the generated code.

Additionally, the RteCV verification report shows a table with the results of all verification steps and their test steps. Each verification step has these two test steps:

- *Pattern generation*: in this step a code pattern is generated according to your `RTE` configuration. That means that the code pattern is filled with the expected attributes, e.g. function names and variables.
- *Pattern search*: in this step the RteCV tries to match the generated code pattern to a specific part of your generated `RTE` code.

If both verification steps are executed successfully the following information are included in the RteCV verification report:

- a link to the generated code pattern
- the matched part of the code
- a link to the source file

No links are generated if the generated code pattern does not match any part of the generated `RTE` code or the code pattern could not be generated at all.

The verification step and test step can have two different status:

- *passed*: if all test steps of a verification step or the test step itself succeed.
- *failed*: if one or more test steps of a verification step or the test step itself failed.

A test suite is successful, if all included test cases and all including verification steps with their test steps were executed successfully. That means that their status is *passed*. Otherwise, the test suite is *failed*.

7.4. RteCV static code check examples

This section contains some example calls to the `RteCV.bat` batch file to illustrate how to perform the RTE static code checks.

Input for code check examples

All examples assume the following situation:

- The generated RTE source files are located in the following two directories:
  - `output/generated/include`
  - `output/generated/src`
- The serialized RTE intermediate model is provided in file `output/generated/doc/RteDataModel.xml`
- The following C preprocessor defines are provided to the compiler:
  - `DEFINE_A=VALUE_A`
  - `DEFINE_B`
- The following C preprocessor include directories are provided to the compiler:
  - `output/generated/include`

- `<TRESOS>/plugins/TestPlugin/include`

- The reports for the code checks shall be generated to the directory `output/generated/rtecv`

All paths are relative to your EB tresos® project directory.

7.4.1. RteCV examples with command line parameters

7.4.1.1. Performing all static code checks

The following call represents a variant to run both the pattern code checks and abstract syntax tree based code checks in which all information is passed via command line parameters:

Example 7.1. Call RteCV with command line parameters

```
<TRESOS>/tools/RteCVExt_TS_TxDxM5I5R0/RteCV.bat
    --data-model output/generated/doc/RteDataModel.xml
    --mode ALL
    --source-dir output/generated/include, output/generated/src
    --cpp-defines DEFINE_A=VALUE_A,DEFINE_B
    --include-dir <TRESOS>/plugins/TestPlugin/include, output/generated/include
    --plugins RteCVExt.CodeChecker_TS_TxDxM5I5R0,
    RteCVExt.Introspection_TS_TxDxM5I5R0, RteCVExt.DataModelReport_TS_TxDxM5I5R0
    --output output/generated/rtecv
```

7.4.1.2. Performing only pattern based code checks

The following call represents a variant in which all information is passed via command line parameters:

Example 7.2. Call only pattern based code checks

```
<TRESOS>/tools/RteCVExt_TS_TxDxM5I5R0/RteCV.bat
    --data-model output/generated/doc/RteDataModel.xml
    --source-dir output/generated/include, output/generated/src
    --plugins RteCVExt.CodeChecker_TS_TxDxM5I5R0,
    RteCVExt.DataModelReport_TS_TxDxM5I5R0
    --output output/generated/rtecv
```

Alternatively,the check could be performed by removing `--plugins` parameter and instead adding `--mode` parameter with the value *CHECKER*.

7.4.1.3. Performing only abstract syntax tree based checks

The following call represents a variant to run only the introspection checker in which all information is passed via command line parameters:

Example 7.3. Call only abstract syntax tree based checks

```
<TRESOS>/tools/RteCVExt_TS_TxDxM5I5R0/RteCV.bat
    --data-model output/generated/doc/RteDataModel.xml
    --source-dir output/generated/include, output/generated/src
    --cpp-defines DEFINE_A=VALUE_A,DEFINE_B
    --include-dir <TRESOS>/plugins/TestPlugin/include, output/generated/include
    --plugins RteCVExt.Introspection_TS_TxDxM5I5R0,
    RteCVExt.DataModelReport_TS_TxDxM5I5R0
    --output output/generated/rtecv
```

Alternatively,the check could be performed by removing `--plugins` parameter and instead adding `--mode` parameter with the value *INTROSPECT*.

7.4.2. RteCV examples with option file

If the example option file(shown below) `rtecv_option_file.opt` is located in the current directory, the call would look like this:

Example 7.4. Call RteCV with the option file `rtecv_option_file.opt`

```
<TRESOS>/tools/RteCVExt_TS_TxDxM5I5R0/RteCV.bat
    --opt rtecv_option_file.opt
    --source-dir output/generated/include, output/generated/src
```

7.4.2.1. Performing all static code checks

The following call represents a variant to run both the pattern code checks and abstract syntax tree based code checks in which all information is passed via option file:

Example 7.5. Example option file `rtecv_option_file.opt`

```
RteCV.Option.DataModelFile       = output/generated/doc/RteDataModel.xml
RteCV.Option.SourceDirectories   = output/generated/include, output/generated/src
RteCV.Option.Defines             = DEFINE_A=VALUE_A,DEFINE_B
```

```
RteCV.Option.IncludeDirectories  = <TRESOS>/plugins/TestPlugin/include,
                                   output/generated/include
RteCV.Option.OutputDirectory     = output/generated/rtecv
RteCV.Option.Plugins             = RteCVExt.CodeChecker_TS_TxDxM5I5R0,
                                   RteCVExt.Introspection_TS_TxDxM5I5R0,
                                   RteCVExt.DataModelReport_TS_TxDxM5I5R0
RteCV.Option.Mode                = ALL
```

7.4.2.2. Performing only pattern based code checks

The following call represents a variant in which all information is passed via option file:

Example 7.6. Example option file `rtecv_option_file.opt`

```
RteCV.Option.DataModelFile       = output/generated/doc/RteDataModel.xml
RteCV.Option.OutputDirectory     = output/generated/rtecv
RteCV.Option.Plugins             = RteCVExt.CodeChecker_TS_TxDxM5I5R0,
                                   RteCVExt.DataModelReport_TS_TxDxM5I5R0
```

Alternatively,the check could be performed by removing the option `RteCV.Option.Plugins` and instead adding the option `RteCV.Option.Mode` with the value *CHECKER*.

7.4.2.3. Performing only abstract syntax tree based checks

The following call represents a variant to run only the introspection checker in which all information is passed via option file:

Example 7.7. Example option file `rtecv_option_file.opt`

```
RteCV.Option.DataModelFile       = output/generated/doc/RteDataModel.xml
RteCV.Option.SourceDirectories   = output/generated/include, output/generated/src
RteCV.Option.Defines             = DEFINE_A=VALUE_A,DEFINE_B
RteCV.Option.IncludeDirectories  = <TRESOS>/plugins/TestPlugin/include,
                                   output/generated/include
RteCV.Option.OutputDirectory     = output/generated/rtecv
RteCV.Option.Plugins             = RteCVExt.Introspection_TS_TxDxM5I5R0,
                                   RteCVExt.DataModelReport_TS_TxDxM5I5R0
```

Alternatively,the check could be performed by removing the option `RteCV.Option.Plugins` and instead adding the option `RteCV.Option.Mode` with the value *INTROSPECT*.

7.5. Verifying RTE features with user defined patterns

The plugin *user pattern checker* is used to verify the generated RTE files. The user defined pattern files describe a code fragment to be matched. The RteCV provides support for special syntax that can be used in the patterns to make them more flexible. The supported keywords are the following (without quotes):

- `'#####'`: matches multiple lines of any symbols.
- `'####'`: matches a single line of any symbols.
- `'###'`: matches a single word.
- `'###=CaptureName=###'`: declares a named-capturing group that can be referenced in other parts of the pattern.
- `'###&CaptureName&###'`: references a previously declared named-capturing group.
- `###?OptionalString?###`: matches the given string once or not at all

NOTE

Allowed characters for the name of a named-capturing

The name of a named-capturing group is composed of the following characters. The first character must be a letter, followed by:

- Uppercase letters `'A'` through `'Z'`.
- Lowercase letters `'a'` through `'z'`.
- Digits `'0'` through `'9'`.
- Underscores `'_'`.

For example, two fictive RTE api functions could look like this:

```
FUNC(void, RTE_CODE) Rte_ApiFunction_OsApplication_Asil(void)
{
    Rte_State_OsApplication_Asil = RTE_PARTITION_TERMINATED;
    (void)Rte_IP_Write_Rte_State_OsApplication_Asil(RTE_PARTITION_TERMINATED);
}



FUNC(void, RTE_CODE) Rte_ApiFunction_OsApplication_Qm(void)
{
    Rte_Optional_OsApplication_Qm();
    Rte_State_OsApplication_Qm = RTE_PARTITION_TERMINATED;
    (void)Rte_IP_Write_Rte_State_OsApplication_Qm(RTE_PARTITION_TERMINATED);
}
```

To match all similar `Rte_PartitionTerminated` methods, user can provide a pattern:

```
FUNC(void, RTE_CODE)
    Rte_ApiFunction_OsApplication_###=PartitionName=###(void)
{
    ###?Rte_Optional_OsApplication_###();?###
    Rte_State_OsApplication_###&PartitionName&### = ###=TerminatedVariable=###;
    (void)Rte_IP_Write_Rte_State_OsApplication_###&PartitionName&###
                    (###&TerminatedVariable&###);
}
```

This pattern will ensure that correct partition variables are being handled and the right method call was generated, subsequently matching both the above examples.

7.5.1. Executing the RteCV user-defined pattern checker

To execute the RteCV, you must call the batch file `RteCV.bat` in *USERDEFINED* mode as described in the example below:

Example 7.8. Call RteCV user-defined pattern checker with command line parameters

```
<TRESOS>/tools/RteCVExt_TS_TxDxM5I5R0/RteCV.bat
    --source-dir output/generated/include, output/generated/src
    --user-defined-patterns patterns/Rte_Read.pattern
    --mode USERDEFINED
    --output output/generated/rtecv
```

Alternatively, users can pass user-defined patterns with option file by using parameter "RteCV.Option.UserDefinedPatterns" as described in the example below:

Example 7.9. Example option file `rtecv_option_file.opt`

```
RteCV.Option.SourceDirectories   = output/generated/include, output/generated/src
RteCV.Option.UserDefinedPatterns = patterns/Rte_Read.pattern
RteCV.Option.Mode                = USERDEFINED
RteCV.Option.OutputDirectory     = output/generated/rtecv
```

For each pattern file a test case is created and the results will be added to the test report. The test case status will be marked as passed if at least one match was found across all of the generated RTE files. If no matches

are found for user-defined pattern, then the test case is marked as failed. For more details about test report, see Section 7.3.4, “Evaluating the results of the static code checks”.

7.6. Created output folders and generated files

In `CHECKER`, `INTROSPECT`and `USERDEFINED` mode, the RteCV generates a report for each performed test and it generates an `HTML` file for each source file, where one of the patterns of the tests was successfully matched. You can find an index `HTML` file called `reports.html` that contains links to all other `HTML` files in the `report` directory in the output folder. You can find the individual `HTML` files in the `report\html` directory in the output folder.

Examples:

- `generated\rtecv\report\reports.html`
- `generated\rtecv\report\html\SafetyRTE.VS.Check.ClientServer.RteCallApis.html`

Glossary

EB tresos AutoCore Generic RTE	EB tresos AutoCore Generic RTE is the AUTOSAR RTE implementation. It provides the configuration interface and the code generator to generate the RTE code. EB tresos AutoCore Generic RTE generates the RTE target implementation based on a predefined input model. See *Runtime Environment (RTE)*.
Application software	The application software is the software implemented within the AUTOSAR application layer. Among others this includes the *AUTOSAR application software components*.
AUTOSAR XML (ARXML)	An extensible markup language (XML) file format which is used in the AUTOSAR environment for storing *system descriptions* and AUTOSAR module configurations.
AUTOSAR model	An AUTOSAR model is an M1 AUTOSAR model, compare [ASRGENST], chapter 2.2. It can be e.g. an *AUTOSAR system description*, a *software component* description, or a *BSW module* configuration.
Basic software module (BSWM)	A basic software module is an AUTOSAR basic software module.
EB tresos Safety RTE	The EB tresos Safety RTE is the implementation of an AUTOSAR *Runtime Environment*. It consists the *EB tresos AutoCore Generic RTE* to generate the RTE code. It consists also the *RteCV* with the safety manual to verify the generated RTE code.
Electronic control unit (ECU)	An electronic control unit is an AUTOSAR ECU (compare [ASRGLOSSARY]).
Integrator	An integrator is a person who integrates the software on the ECU.
Runtime Environment (RTE)	The Runtime Environment (RTE) or also AUTOSAR Runtime Environment is an AUTOSAR module. It is specified by [ASRRTE422].
Rte Code Verifier (RteCV)	The Rte Code Verifier (RteCV) is a development tool to verify the generated code from *EB tresos AutoCore Generic RTE*. In this way the confidence in the generated code is increased. The RteCV performs static code checks on the generated RTE code to verify that the generated code correctly implements the provided model.
Software component (SWC)	A software component (SWC) is an AUTOSAR software component.

System application	A system application is the software implementing the functionality of an *ECU*. In context of the EB tresos Safety RTE this includes the *application software*, the *AUTOSAR RTE*, the AUTOSAR basic software and the AUTOSAR OS.
System description	A system description is an AUTOSAR system description.
XDM	The XDM-format is an EB tresos Studio specific XML data-format that allows storing the node structure including all data (attributes, node names) of a data model or a module configuration in a file.

Bibliography

[ASRGENST] *Generic Structure Template*, Version 3.2.0 Final, 2011-10-31

[ASRGEN-TRANSFORM422] *General Specification of Transformers*, Version 4.2.2 Final

[ASRGLOSSARY] *Glossary*, Version 4.2.2 Final

[ASRRTE422] *Specification of RTE*, Version 4.2.2 Final

[ASRSYST] *System Template*, Version 4.2.0 Final, 2011-11-08

[AUTO-COREUSRDOC] *EB tresos® AutoCore Generic documentation*, product release 7.7

[ISO26262_1ST] *INTERNATIONAL STANDARD ISO 26262*: Road vehicles - Functional safety, 2011

[ISO26262-10_1ST] *INTERNATIONAL STANDARD ISO 26262-10*: Road vehicles - Functional safety - Part 10: Guideline on ISO 26262, 2012

[RMP] *Requirement Management Plan*: EB tresos Safety RTE

[SAFETYMAN] *EB tresos Safety RTE Safety Manual*

Safety_Rte_safety_manual.pdf

Appendix A. Document configuration information

This document was created by the DocBook engine using the source files and revisions listed below. All paths are relative to the directory https://subversion.ebgroup.elektrobit.com/svn/autosar/asc_RteCV/trunk/doc/public/userguides.

Filename	Revision / Hash
../fragments/bibliography/Bibliography.xml	13947
../fragments/glossary/Glossary.xml	13938
About.xml	10822
appendix/OpenSourceLicenses.xml	9950
BackgroundInformation.xml	4671
BeginHere.xml	3091
BookInfo.xml	4244
document.ent	1314
History.xml	13942
SafeAndCorrectUse.xml	13915
Safety_RTE_documentation.xml	8383
UsingTheRteCV/CodeCheckerExamples.xml	13855
UsingTheRteCV/ExecuteStaticChecks.xml	13855
UsingTheRteCV/GeneratedFoldersAndFiles.xml	13855
UsingTheRteCV/IntroductionToRteCV.xml	13894
UsingTheRteCV/LogOutputTip.xml	9782
UsingTheRteCV/Main.xml	13855
UsingTheRteCV/Overview.xml	13855
UsingTheRteCV/RteDataModelImportNote.xml	10822
UsingTheRteCV/UserDefinedPatterns.xml	11972

Appendix B. Open source licenses

This product includes third party components that require the following notices. For respective license terms refer to the subfolder `licenses`.

- Google Inject 3.0.0. This product includes software developed by the Apache Software Foundation. Version 2.0, January 2004. (http://www.apache.org/licenses/).

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions: "License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

   "Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized

to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an

addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

APPENDIX: How to apply the Apache License to your work. To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party

archives. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

- The MIT License Copyright (c) 2004-2015 Paul R. Holser, Jr. for JOpt Simple 4.6 (command line parser).

  1. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
  2. The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
  3. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Appendix C. Potential export restrictions

The product is not classified as a “dual-use item” in accordance with Council Regulation (EC) 428/2009 (as of September 2015). The Licensee is responsible for compliance with any applicable export or import regulations or obligations. Elektrobit Automotive GmbH provides assistance on request.