Elektrobit

EB tresos® AutoCore Generic 8 Scriptor documentation

Module release 1.0.9

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential and proprietary information

Table of Contents

1. Overview 4
2. Scriptor module release notes 5
2.1. Change log 5
2.2. New features 7
2.3. EB-specific enhancements 7
2.4. Deviations 7
2.5. Limitations 7
2.6. Open-source software 7
3. Scriptor user's guide 8
3.1. Overview 8
3.2. Usage 8
3.2.1. Background knowledge 8
3.2.1.1. XML, XPath, `EB tresos Studio` XPath API 8
3.2.1.2. XDM configuration data model 8
3.2.2. `Scriptor` module configuration 9
3.2.2.1. XML script locations 9
3.2.2.2. ConstantGroups 9
3.2.3. The `Scriptor` unattended wizard interface 10
3.2.4. `Scriptor` XML language 11
3.2.4.1. Basic structure 11
3.2.4.2. Available elements and operations 12
3.2.4.2.1. `ForEach` 12
3.2.4.2.2. `SetValue` 12
3.2.4.2.3. `SetEnabled` 13
3.2.4.2.4. `Add` 13
3.2.4.2.5. `Remove` 13
3.2.4.2.6. `Condition` 13
3.2.4.3. Script validation using XML schema 13
3.2.4.4. Scope of the Scriptor XML language 13
3.2.4.5. Examples 14
3.2.4.5.1. Working with boolean configuration values (`Condition`, `ForEach`, `SetValue`) 14
3.2.4.5.2. Set main function period for all FlexRay NM channels to 10 ms (`ForEach`, `SetValue`) 15
3.2.4.5.3. Calculate Com PDU handle IDs (`ForEach`, `SetEnabled`, `SetValue`) 15
4. Scriptor module references 17
4.1. Integration notes 17
4.1.1. Integration requirements 17

1. Overview

Welcome to the Scriptor product release notes and documentation.

This document provides:

- Chapter 2, “Scriptor module release notes“: details of changes and new features in the current release
- Chapter 3, “Scriptor user's guide“: concept information and configuration instructions
- Chapter 4, “Scriptor module references“: configuration parameters and the application programming interface

2. Scriptor module release notes

- Module version: 1.0.9.B300355
- Supplier: Elektrobit Automotive GmbH

2.1. Change log

This chapter lists the changes between different versions.

Module version 1.0.9

2019-07-22

Module version 1.0.8

2019-06-18

Module version 1.0.7

2019-04-08

Module version 1.0.6

2019-02-15

- Fix for missing Scriptor.xsd file issue (introduced in 1.0.5)

Module version 1.0.5

2018-12-20

- The sorting order of scripts for execution is now defined as the sorting order of their names.

- The execution of all sub-operations of Scriptor's Add operation is now documented.

Module version 1.0.4

2018-06-21

- Scriptor now sets the importer information (module name, unattended wizard instance, script name) for configuration values which it sets or adds.

Module version 1.0.3

2018-02-16

Module version 1.0.2

2017-12-22

- Fixed SetValue for CHOICE. Previously, the operation had failed silently.
- Fixed SetValue for containers. Previously, the operation had failed silently if the container was not element of a list.
- Extended user's guide (XPath pitfalls, operation descriptions, example on boolean values)
- Add operation: node type prefixes `REFERENCE:`, `CHOICE:` and `VALUE:` are now not needed any more and declared as deprectated. Support may be removed in a future version.
- Improved error handling for SetEnabled operation and other cases
- Adapted current version for ACG 6 compatibility (not guaranteed to persist for future releases although)

Module version 1.0.1

2017-09-14

- Added support for choice containers.

Module version 1.0.0

2017-05-17

- initial release

2.2. New features

2.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

2.4. Deviations

This module is not part of the AUTOSAR specification.

2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- Character ： can cause undefined behaviour when used for Operation of type Add on a Choice list

  Description:

  The usage of character ： in an Expression element results in undefined behaviour when all of the following conditions apply: The Operation type is Add AND the list to add an element to contains elements of type Choice AND the ： character is used in either the selection or value XPath expression of the Expression element, making the separation of the two parts ambiguous to the simple parsing mechanism.
- Wizard configuration page not shown if script name does not match ^[A-Za-z_][A-Za-z0-9 _--]*$

  Description:

  Additional internal constraints on IDs derived from the Name element of Scripts have revealed. If a Name does not match the aforementioned regex, the wizard configuration page is not shown (error message instead).
- Optional nodes which are already disabled will not be shown in results view, when disabled again.

2.6. Open-source software

Open-source software information is not available for this module.

3. Scriptor user's guide

3.1. Overview

The `Scriptor` module allows the user to automate `EB tresos Studio` module configuration tasks using XML-based scripts. In order to achieve this, the user configures script locations on a per-directory base; every of name `*.xml` therein is interpreted as script. The scripts then can be enabled or disabled and then executed using `Scriptor`'s *unattended wizard* interface.

3.2. Usage

3.2.1. Background knowledge

3.2.1.1. XML, XPath, `EB tresos Studio` XPath API

The usage of `Scriptor` requires basic of knowledge of XML and XPath, which is not dealt with in this document. Moreover, in order to make work with and adaption of XPath expressions that are used in `Scriptor` XML scripts easier, `EB tresos Studio`'s XPath console and code template console are very helpful. They can be enabled in the preferences (*Window* > *Preferences* > *`EB tresos Studio`* > *Developer features* > *Show template path in Properties view* and *Show actions 'XPath console' and 'Codetemplate console' in Outline view*). See `EB tresos Studio` developer's guide sections 7.3 *XPath API* and 8.2.2 *Codetemplate console* for further usage information and XPath support in `EB tresos Studio`.

XPath API can have quite some pitfalls. The user is adviced to have a look at `EB tresos Studio` developer's guide section 7.3.4.1. *Pitfalls when using XPath* regarding code fragments which may behave unintuitively.

3.2.1.2. XDM configuration data model

The user should also be familiar with basics of `EB tresos Studio`'s configuration data model format XDM which is visualized in the configuration GUI. The format is discussed in detail in `EB tresos Studio`'s developer's guide section 6.1 *Concepts: The XDM format*, the representation as GUI elements is discussed in section 7.2.4 *GUI Elements*. Moreover, a short overview is provided below. Helpful information on the structure of a certain, currently-active configuration node can be looked up in *Properties* view. This can be enhanced

by activating *Show template path in Properties view* setting in *Window* > *Preferences* > *EB tresos Studio* > *Developer features*.

Configurable modules are shipped with an XDM file which contains the configuration schema. This is then the structure which the actual configuration data in a project are based on. The following configuration node types may be contained in an XDM configuration:

- **container**: Contains elements of any type (any number, any order)
- **variable**: Holds a value of one of the following types: boolean, integer, float, string, multi-line string, enumeration (string with defined list of allowed values).
- **reference**: References another configuration element, typically limited to certain target elements.
- **list**: A list of elements of a certain type. The two subtypes list (ordered list, can contain variables or references) and map (ordered list with unique entries, can contain containers or choices) are distinguished.
- **choice**: A set of containers. One of them is chosen as the *active* one.

Moreover, each node type may be **optional**. If this is the case, the node can be *enabled* or *disabled*.

3.2.2. `Scriptor` module configuration

3.2.2.1. XML script locations

The `Scriptor` module configuration (see `EB tresos Studio` user's guide section 7.8 *Editing parameters of a module configuration* for further information) allows the user to specify script locations on a per-directory base. Each file with suffix `.xml` inside theses configured directories is treated as a script file and can be used by the `Scriptor` unattended wizard, see Section 3.2.3, “The `Scriptor` unattended wizard interface”.

The script locations may contain a set of variables that allow to make them independent of the location of the project directory, workspace directory or `EB tresos Studio` installation directory. The following variables are supported in the form of `${variable}`:

- `workspace_loc`: the location of the currently opened `EB tresos Studio` workspace
- `project_loc`: the location of the currently opened project
- `tresos_loc`: the installation location of `EB tresos Studio`

3.2.2.2. ConstantGroups

Besides the script locations, so-called *ConstantGroups* can be configured. Such a *ConstantGroup* represents a generic pair of key/ID and value. The configuration of *ConstantGroups* helps to improve reusability of scripts.

The script is kept generic and the project-specific data used is obtained from the project-specific `Scriptor` module configuration. As an example, this can be useful for setting imported configuration nodes to certain values each time the importer has finished its job. As in this scenario, on every import, the value is overwritten again, hand-editing would be a tedious and error-prone task which can be automated by a `Scriptor` script.

3.2.3. The `Scriptor` unattended wizard interface

The execution of `Scriptor` scripts is controlled and done by the corresponding `Scriptor` *unattended wizard* (for further information on *unattended wizards*, see section 7.8.9 *Autoconfiguring routine jobs* of `EB tresos Studio`'s user's guide). All available scripts inside the configured script locations are listed in the wizard configuration, see figure Figure 3.1, "`Scriptor` unattended wizard configuration.". The configuration is available via the *unattended wizard* button below `EB tresos Studio`'s menu bar or via the menu at *Project* > *Unattended wizards* > *Unattended wizard configuration...* > *Scriptor*. For every script, it is possible to enable or disable it during wizard execution.

After configuration, the wizard can be run via the configuration menu, via the menu, via the *unattended wizard* button below the menu bar, via an *execute multiple tasks unattended wizard* or even from command line. When done so, the *results view* is displayed, see figure Figure 3.2, "Results view after running the `Scriptor` unattended wizard.". The *results view* lists all nodes that were changed by the execution of the `Scriptor` *unattended wizard*. Double clicking onto a leaf node jumps to that node in the configuration window.

Note that during a single `Scriptor` *unattended wizard* run, a single node may be changed multiple times by different scripts as these are executed consecutively (ordered by script name). The results view only displays the final state of the touched configuration node.

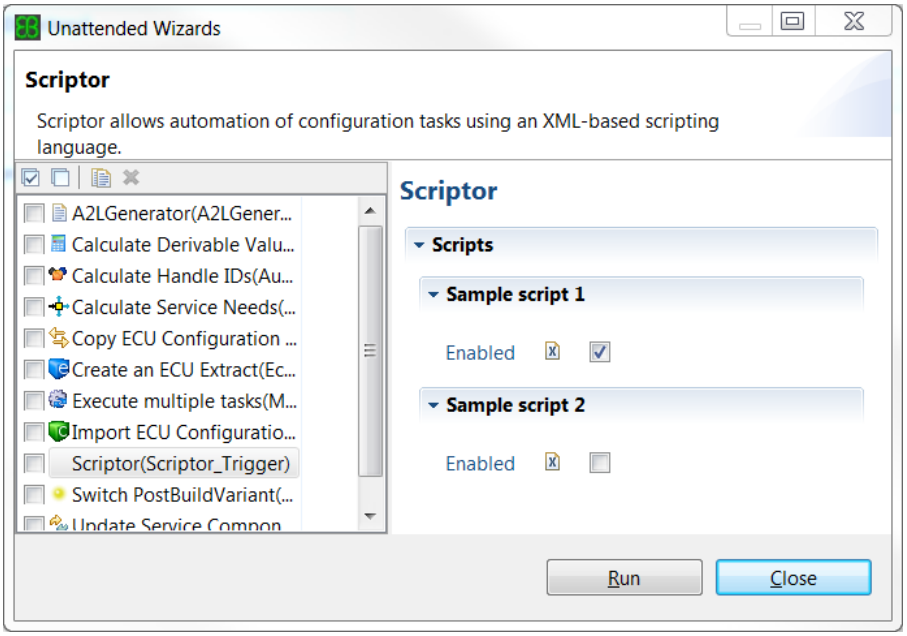

Figure 3.1. `Scriptor` unattended wizard configuration.

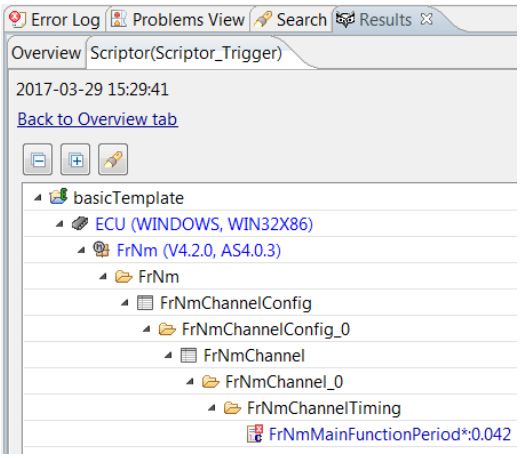

Figure 3.2. Results view after running the `Scriptor` unattended wizard.

3.2.4. `Scriptor` XML language

This section deals with the available features of `Scriptor`'s custom XML language. Section 3.2.4.1, “Basic structure” gives an introduction to the basic structure of XML script documents. Section 3.2.4.2, “Available elements and operations” then gives an overview of the available operations. Section 3.2.4.3, “Script validation using XML schema” shortly gives some hints regarding syntax validation of the XML scripts. Section 3.2.4.5, “Examples” finally gives some usage examples of the `Scriptor` XML language.

3.2.4.1. Basic structure

The minimum structure of a `Scriptor` script contains the root node `Script`, a `Name` and `Description` for the script, and an XPath `Expression` as a starting context for the following `Operation` list in the top-level `Operations` node:

```
<Script xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:noNamespaceSchemaLocation='Scriptor.xsd'>
  <Name>Sample script</Name>
  <Decription>Sample script that works on the configuration of module "Com"
              as start context.</Decription>
  <Expression>as:modconf('Com')[1]</Expression>
  <Operations>
    ...
  </Operations>
</Script>
```

Each `Operation` moreover has the following structure with the operation type as `Type` attribute and an XPath `Expression` and `Operations` node:

```
<Operation Type="(Type)">
  <Expression>root selection expression</Expression>
  <Operations>
    ...
  </Operations>
</Operation>
```

3.2.4.2. Available elements and operations

`Scriptor` currently allows the usage of the operations described in following sections.

3.2.4.2.1. `ForEach`

The `ForEach` operation executes all sub-`Operation`s for each of the items specified by its `Expression`. Before execution of the sub-`Operation`, the execution context is switched to the next item selected by `Expression`.

The `Expression` should always target configuration nodes. If an invalid input such as a string `'string'` is used, this behaves as an empty selection.

3.2.4.2.2. `SetValue`

The `SetValue` operation sets its context node to the value given by its `Expression`:

- For node type *variable* (nodes containing a single value), its value is set to the evaluation result of `Expression`.

  For boolean configuration values (checkboxes), strings `'true'` or `'false'` can be used, but also boolean expressions such as `true()`.

  Due to internal conversion in XPath, strings should be used for configuration values of types integer and float. Otherwise, e. g. an integer value may be set to a floating point value which makes the configuration value invalid.
- For node type *reference*, an AUTOSAR path needs to be used as value. For example, `'ASPath:/Os/Os/Rte_Event_Task'` is to be used instead of `'/Os/Os/Rte_Event_Task'` which is displayed identical in GUI but leads to a warning for the modified node.
- For node type *choice*, the selection of the active container is changed to the container with the given name.
- For node type *container*, the container name is set.

3.2.4.2.3. `SetEnabled`

The `SetEnabled` operation enables or disables its context node according to the `Expression`. This operation works on *optional* nodes (green/red box next to the node). For boolean values (checkboxes), the `SetValue` operation needs to be used.

3.2.4.2.4. `Add`

The `Add` operation inserts new data nodes into its context node of type *list* or *map*. `Expression` contains the reference, value, container name or choice (as `<type_selection>:<name>`). The `Add` operation can contain sub-`Operation`s which are executed on the newly created node as context.

3.2.4.2.5. `Remove`

The `Remove` operation removes its context node from its parent list or map. `Expression` is ignored.

3.2.4.2.6. `Condition`

The `Condition` operation executes all sub-`Operation`s if the condition provided by `Expression` evaluates to `true`.

3.2.4.3. Script validation using XML schema

`Scriptor` ships with an XML schema file included that can be used for validation of a user's XML scripts using special validation tools or XML editors. Note that using a dedicated XML editor may lead to best productivity when writing `Scriptor` scripts anyway.

3.2.4.4. Scope of the Scriptor XML language

`Scriptor`'s XML language provides a simple interface for common use cases for automated setting of configuration parameter values. However, it is not designed to handle all possible use cases or provide the power that a general purpose programming language has. Therefore, everything beyond that scope should better be handled e. g. by using `EB tresos Studio`'s public Java API that is described in its developer's guide. Examples for such limitations:

- There is no language construct such as a variable definition that would allow to store information from an XPath query and refer to that from subsequent XPath queries.
- There is no language construct that allows to copy list containers.

3.2.4.5. Examples

3.2.4.5.1. Working with boolean configuration values (`Condition`, `ForEach`, `SetValue`)

Note that boolean values (checkboxes) provide their value as strings `'true'` or `'false'`. In order to check the value of such a configuration node, the correct syntax is `BooleanConfigParam = 'true'`. Other similar formulations may behave unexpected, e. g.

- `boolean(BooleanConfigParam)` evaluates to `'false'` independent of the configuration value held by `BooleanConfigParam`
- `BooleanConfigParam` evaluates to `'false'`, but also `not(BooleanConfigParam)` evaluates to `'false'` (for checking whether a node exists, the `node:exists()` function needs to be used)

Please see `EB tresos Studio` developer's guide section 7.3.4.1. *Pitfalls when using XPath* for a collection of general hints when using XPath API.

The following script works on the configuration of a module `MyModule`. If the boolean value `BooleanConfigParam` is set to `'true'`, then integer configuration value `IntegerValueOne` is set to value `42`, otherwise `IntegerValueTwo` is set to `4711`

```
<Script xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation='Scriptor.xsd'>
 <Name>Set integer configuration value</Name>
 <Decription>Sets two integer values dependent on a boolean.</Decription>
 <Expression>as:modconf('MyModule')</Expression>
 <Operations>
  <Operation Type="Condition">
   <Expression>BooleanConfigParam = 'true'</Expression>
   <Operations>
    <Operation Type="ForEach">
     <Expression>IntegerValueOne</Expression>
     <Operations>
      <Operation Type="SetValue">
       <Expression>'42'</Expression>
      </Operation>
     </Operations>
    </Operation>
   </Operations>
  </Operation>
  <Operation Type="Condition">
   <Expression>BooleanConfigParam = 'false'</Expression>
   <Operations>
    <Operation Type="ForEach">
     <Expression>IntegerValueTwo</Expression>
```

```
     <Operations>
      <Operation Type="SetValue">
       <Expression>'4711'</Expression>
      </Operation>
     </Operations>
    </Operation>
   </Operations>
  </Operation>
 </Operations>
</Script>
```

3.2.4.5.2. Set main function period for all FlexRay NM channels to 10 ms (`ForEach`, `SetValue`)

The following script iterates over all main function periods in configuration of module FrNm and sets their value to 10 ms.

```
<Script xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation='Scriptor.xsd'>
  <Name>Set FrNm main period</Name>
  <Decription>Sets main function period for all FlexRay NM channels to 10 ms.</Decription>
  <Expression>as:modconf('FrNm')</Expression>
  <Operations>
    <Operation Type="ForEach">
      <Expression>FrNmChannelConfig/*/FrNmChannel/*/FrNmChannelTiming/FrNmMainFunctionPeriod \
</Expression>
      <Operations>
        <Operation Type="SetValue">
          <Expression>0.01</Expression>
        </Operation>
      </Operations>
    </Operation>
  </Operations>
</Script>
```

3.2.4.5.3. Calculate Com PDU handle IDs (`ForEach`, `SetEnabled`, `SetValue`)

The following script iterates over all Com PDU handle IDs. The configuration nodes are optional and may be disabled. They are therefore enabled, and then assigned a zero-based consecutive value (consecutive per each communication direction).

```
<Script xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation='Scriptor.xsd'>
  <Name>COM handle IDs</Name>
  <Decription>Calculates all COM handle IDs</Decription>
  <Expression>as:modconf('Com')/ComConfig/*</Expression>
  <Operations>
    <Operation Type="ForEach">
      <Expression>ComIPdu/*/ComIPduHandleId</Expression>
      <Operations>
        <Operation Type="SetEnabled">
          <Expression>boolean(1)</Expression>
        </Operation>
        <Operation Type="SetValue">
          <Expression>num:i(count(../preceding-sibling::*[ComIPduDirection=node:current() \
/../ComIPduDirection]))</Expression>
        </Operation>
      </Operations>
    </Operation>
  </Operations>
</Script>
```

4. Scriptor module references

Scriptor configuration parameter reference is not available.

Scriptor API reference is not available.

4.1. Integration notes

4.1.1. Integration requirements

WARNING 	**Integration requirements list is not exhaustive** The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the Scriptor module.