BTC Embedded Platform - RESTful API documentation

API Version: 23.1p0

This is the documentation for the BTC Embedded Platform RESTful API.

CONTACT

EMAIL: support@btc-embedded.com

INDEX

API
1. APPLICATION

Handle the EP application itself.

1.1 DELETE /ep/application

Exit EP and save the active profile
After calling the exit method, the current active profile will be saved, if there is one. If the profile has no save path, one is created at a temporary directory and a response containing the path is returned.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
force-quit	boolean	If set to 'true', the application will force quit without saving the active profile. If set to 'false', the profile is saved at the given location, or at a temporary location if provided none. Default is 'false'

RESPONSE

STATUS CODE - 200: OK

    RESPONSE MODEL - text/plain

string

STATUS CODE - 201: Created

    RESPONSE MODEL - text/plain

string

STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

2. ARCHITECTURES

Import and retrieve architectures.

2.1 POST /ep/architectures/ccode

Import a C-Code architecture

\Long running task\ Import a C-Code architecture. Settings are provided as an import info object.

REQUEST

REQUEST BODY - application/json
```
{
    modelFile*    string    File name of the C-Code XML file
    mappingFile   string    File name of the mapping XML file. The file must be provided if architectures are already available in the
                            profile. The file is used to map the first imported architecture to the new imported C-Code architecture.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the \Progress\ i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json
```
{
    jobID    string    READ-ONLY
                       The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

2.2 POST /ep/architectures/embedded-coder-wrapper-creation

Create an EmbeddedCoder™ wrapper model

\Long running task\ Creating an EmbeddedCoder™ wrapper model. Settings are provided as an import wrapper info object.

REQUEST

REQUEST BODY - application/json
```
{
    ecModelFile*   string    The absolute or relative path to the Embedded Coder model.
    ecInitScript   string    The absolute or relative path to the init script of the Embedded Coder model.
```

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID    string    READ-ONLY
                       The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

2.3 POST /ep/architectures/embedded-coder

Import an EmbeddedCoder™ architecture

Long running task Import an EmbeddedCoder™ architecture. Settings are provided as an import info object.

REQUEST

REQUEST BODY - application/json

```
{
    ecModelFile*          string    The absolute or relative path to the EmbeddedCoder model.<br>This file is mandatory
                                    and may not be undefined.
    ecInitScript          string    The absolute or relative path to the init script of the EmbeddedCoder model.<br>This
                                    attribute may be undefined.<br>If the specified file path is invalid, an error is reported.
    parameterHandling     enum      ALLOWED:OFF, EXPLICIT_PARAMETER
                                    Specifies how parameter variables are handled. Can be 'OFF' or
                                    'EXPLICIT_PARAMETER'.<br>'OFF': Only regular inputs in the interface of subsystems
                                    are observed.<br>'EXPLICIT_PARAMETER': Parameter variables are regarded as
                                    additional<br>inputs to subsystems.<br>Default is 'EXPLICIT_PARAMETER'.<br>If not
                                    specified (undefined or invalid), the default value is used.
    testMode              enum      ALLOWED:BLACK_BOX, GREY_BOX
                                    Specifies the test mode. Can be 'BLACK_BOX' or 'GREY_BOX'.<br>If set to 'GreyBox',
                                    local variables are regarded as additional outputs of subsystems.<br>For BlackBox-
                                    Testing only the regularoutputs in the interfaces of subsystems are
                                    observed.<br>Default is 'GreyBox'.<br>If not specified (undefined or invalid), the default
                                    value is used.
    fixedStepSolver       boolean   Handling when a non-fixed-step solver is ecountered: If 'true', the solver is automatically
                                    set to fixed-step.<br>Otherwise an error is issued.Default is 'false'.
    inDepthCcodeAnalysis  boolean   true: Simulink model and C-Code model are imported.<br>false: indicates that a SL SIL
                                    simulation is supported.<br>C-Code and mapping are omitted.<br>Default is 'true'
    subsystemMatcher      string    A whitelist of all subsystems you would like to import.<br>Uses the regular expression
                                    standard.<br>Each subsystem is identified by its virtual path inside the model.<br>If no
                                    value is defined, all subsystems are imported.<br>If the specified regular expression
                                    does not produce any match, an error is reported.
    parameterMatcher      string    A whitelist of all parameters you would like to import.<br>Uses the regular expression
```

standard.
Each parameter is identified by its name.
If no value is defined, all calibrations are imported.
If the specified regular expression does not produce any match, no calibration is imported.
The list cannot be applied, if parameterHandling is set to 'OFF'.
Please note, that model workspace parameters have their name extended by the model they
are located in and need to be adressed accordingly in the following way: 'modelname:paramname'

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID    string   READ-ONLY
                       The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

2.4 POST /ep/architectures/targetlink-ev

Import a TargetLink™ EV architecture
Long running task Import a TargetLink™ EV architecture. Settings are provided as an import info object.

REQUEST

REQUEST BODY - application/json

```
{
    tlModelFile*          string     The absolute or relativ path to the TargetLink model file (.mdl|.slx).
    tlInitScript          string     The absolute or relativ path to the script defining all parameters needed for initializing the TL-
                                     model. If not provided, the TL-model is assumed to be selfcontained.
    fixedStepSolver       boolean    'true': The analyzed models (TargetLink model and Simulink model) will be set to the fixed-step
                                     solver type automatically. The usage of the EmbeddedPlatform requires a fixed-step solver. If
                                     the model is open and the fixed-step solver is not already set, this might lead to a modified
                                     model. <br>'false': The analyzed models (TargetLink model and Simulink model) will not be set
                                     to the fixed-step solver automatically. If the fixed-step solver is not already set in the model, an
                                     exception is thrown. In order to proceed the user has to set the fixed-step solver manually in
                                     the simulation settings. <br>Note: The option is ignored if the model is currently not in an
                                     open/loaded state. In this case it has no visible side-effect. <br>Default is 'false'.
    tlSubsystem           string     Name of the Subsystem representing the TL toplevel subsystem for the analysis.<br>Note:
                                     Argument is obligatory if there is more than one toplevel system in the model.
    calibrationHandling   enum       ALLOWED:OFF, EXPLICIT_PARAMETER, LIMITED_BLOCKSET
                                     Determine how calibration variables are being handled.<br>'OFF': Only regular inputs in the
                                     interface of subsystems are observed. <br>'EXPLICIT_PARAMETER': Calibration variables are
                                     regarded as additional inputs to subsystems. Their value is set once during the initial phase of
                                     the simulation and is held constant thereafter. <br>'LIMITED_BLOCKSET': Calibration variables
                                     are regarded as additional inputs to subsystems. Their value is set once during the initial
                                     phase of the simulation and is held constant thereafter. Enable support for calibration within
```

		block properties, not using workspace calibrations. Note: The supported Set of calibratable TL-Blocks is different from 'EXPLICIT_PARAMETER'. Default is 'EXPLICIT_PARAMETER'.
subsystemMatcher	string	A whitelist of all subsystems you would like to import. Uses the regular expression standard. Each subsystem is identified by its virtual path inside the model. If no value is defined, all subsystems are imported. If the specified regular expression does not produce any match, only the toplevel subsystem is imported. The root subsystem is always imported. Please note that the subsystem-path has to include the Targetlink wrapper.
calibrationMatcher	string	A whitelist of all calibrations you would like to import. Uses the regular expression standard. Each calibration is identified by its name. If no value is defined, all calibrations are imported. If the specified regular expression does not produce any match, no calibration is imported. The list cannot be applied, if calibrationHandling is set to 'OFF'.

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

{
   jobID   string   **READ-ONLY**
                             The ID of a job.
}

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

2.5 POST /ep/architectures/mdf

Import an MDF™ architecture
Long running task Import an MDF™ architecture. Settings are provided as an import info object.

REQUEST

REQUEST BODY - application/json
{
   modelFile*   string   Absolute or relative path to the execution record (.mf4).
   sampleTime*   string   Sample time, in seconds.
}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                       The ID of a job.
}
```

STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

    RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not acceptable

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

2.6 POST /ep/architectures/sdf

Import a SDF™ architecture

Long running task Import a SDF™ architecture. Settings are provided as an import info object.

REQUEST

REQUEST BODY - application/json
```
{
    sdfFile*            string        The absolute path to the SDF file. This file is mandatory and may not be null! If the specified
                                      file path is invalid, an error is reported.
    trcFile*            string        The absolute path to the TRC file. This file is mandatory and may not be null! If the specified
                                      file path is invalid, an error is reported.
    sampleTime*         string        The SDF architecture sample time in seconds. The format can be only numbers (e.g. 3) or with
                                      the time unit specified (e.g. 3 s). Attempt to use any other format will be reported as an error.
    excludedSubsystems  [string]      Subsystems names that should not be imported, can be empty. If an invalid subsystem is
                                      requested, an error is reported.
    propagateSelection  boolean       This option allows to configure the excluded subsystems option. If set to true, also the child
                                      subsystems of the excluded subsystems are excluded from the import. Default setting is false.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

    RESPONSE MODEL - application/json
```
{
    jobID   string   READ-ONLY
                       The ID of a job.
}
```

STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

2.7 GET /ep/architectures/{architecture-uid}

Get a specific architecture

Get an existing architecture by UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*architecture-uid	string	The UID of the architecture to be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
    uid                 string   READ-ONLY
                                 The unique identifier (UID) of this object.
    architectureKind    string   READ-ONLY
                                 The string representation of the concrete architecture
                                 , e.g. 'Simulink', 'C-Code', 'TargetLink'.
    name                string   READ-ONLY
                                 The architecture name as specified by the model.
    propList {
    The Architecture property List
    }
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

2.8 GET /ep/architectures

Get all architectures

Search for all open architectures.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
architecture-kind	string	Specifies the specific architecture kind to be queried. (e.g. 'Simulink', 'C-Code', 'TargetLink')

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
architectureKind	string	**READ-ONLY**
		The string representation of the concrete architecture , e.g. 'Simulink', 'C-Code', 'TargetLink'.
name	string	**READ-ONLY**
		The architecture name as specified by the model.

```
    propList {
```
The Architecture property List

```
    }
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

```
string
```

2.9 PUT /ep/architectures

Update architectures

Long running task Perform an architecture update.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

`string`

2.10 PUT /ep/architectures/model-paths

Update model paths for architectures

Update the paths for imported architectures. If the architecture-uid is not specified, the model paths will be updated for the master architecture.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
`architecture-uid`	`string`	The UID of the architecture to be updated. If empty, the path will be updated for the master architecture.

REQUEST BODY - application/json

```
{
    slModelFile    string   The path to the Simulink model(.mdl|.slx).
    slInitScript   string   The path to the init script of the Simulink model.
    addModelInfo   string   The path to additional model information.
    tlModelFile    string   The path to the TargetLink model file (.mdl|.slx).
    tlInitScript   string   The path to the init script of the TargetLink model.
    environment    string   The path to XML file including information about
                            environmental files like additional code.
}
```

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

> **RESPONSE MODEL - text/plain**

string

2.11 POST /ep/architectures/simulink

Import a Simulink™ architecture

Long running task Import a Simulink™ architecture. Settings are provided as an import info object.

REQUEST

REQUEST BODY - application/json

```
{
```

`slModelFile*`	string	The absolute or relative path to the Simulink model. This file is mandatory and may not be undefined!
`slInitScriptFile`	string	The absolute or relative path to the init script of the Simulink model. This attribute may be undefined. If the specified file path is invalid, an error is reported.
`parameterHandling`	enum	**ALLOWED:** OFF, EXPLICIT_PARAMETER Specifies how parameter variables are handled. Can be 'OFF' or 'EXPLICIT_PARAMETER'. 'OFF': Only regular inputs in the interface of subsystems are observed. 'EXPLICIT_PARAMETER': Parameter variables are regarded as additional inputs to subsystems. Default is 'EXPLICIT_PARAMETER'. If not specified (undefined or invalid), the default value is used.
`testMode`	enum	**ALLOWED:** BLACK_BOX, GREY_BOX Specifies the test mode. Can be 'BLACK_BOX' or 'GREY_BOX'. If set to 'GREY_BOX', local variables are regarded as additional outputs of subsystems. For Black Box-Testing only the regular outputs in the interfaces of subsystems are observed. Default is 'GREY_BOX'. If not specified (undefined or invalid), the default value is used.
`fixedStepSolver`	boolean	Defines, if the fixed step solver will be set or not. Can be true or false. true: The analyzed Simulink model will be set to the fixed-step solver type automatically. The usage of the EmbeddedPlatform requires a fixed-step solver. If the model is open and the fixed-step solver is not already set, this might lead to a modified model. false: The analyzed Simulink model will not be set to the fixed-step solver automatically. If the fixed-step solver is not already set in the model, the method return with state of a non fixed-step solver. In order to proceed the user has to set the fixed-step solver manually in the Simulink simulation settings. Default is 'false'. Note: The option is ignored if the model is currently not in an open/loaded state. In this case it has no visible side-effect.
`mappingFile`	string	File name of the mapping XML file. The file must be provided if architectures are already available in the profile. The file is used to map the first imported architecture to the new imported Simulink architecture. This attribute may be undefined. If the specified file path is invalid, an error is reported.
`subsystemMatcher`	string	A whitelist of all subsystems you would like to import. Uses the regular expression standard. Each subsystem is identified by its virtual path inside the model. If no value is defined, all subsystems are imported. If the specified regular expression does not produce any match, an error is reported. Keep in mind that having multiple Toplevel-Subsystem will result in an error.
`parameterMatcher`	string	A whitelist of all parameters you would like to import. Uses the regular expression standard. Each parameter is identified by its name. If no value is defined, all parameters are imported. If the specified regular expression does not produce any match, no parameter is imported. The list cannot be applied, if parameterHandling is set to 'OFF'. Please note, that model workspace parameters have their name extended by the model they are located in and need to be adressed accordingly in the following way: 'modelname:paramname'

```
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

> **RESPONSE MODEL - application/json**

```
{
    jobID   string   READ-ONLY
            The ID of a job.
```

}

    STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

string

    STATUS CODE - 403: Forbidden

    RESPONSE MODEL - text/plain

string

    STATUS CODE - 500: Internal server error.

    RESPONSE MODEL - text/plain

string

2.12 POST /ep/architectures/targetlink

Import a TargetLink™ architecture

Long running task Import a TargetLink™ architecture. Settings are provided as an import info object.

REQUEST

REQUEST BODY - application/json
{

`tlModelFile*`	`string`	The absolute or relative path to the TargetLink model file (.mdl|.slx).
`tlInitScript`	`string`	The absolute or relative path to the script defining all parameters needed for initializing the TL-model. If not provided, the TL-model is assumed to be selfcontained.
`slModelFile`	`string`	The absolute or relative path to the Simulink model file (.mdl|.slx). Note: If a TargetLink model is given, the SL-model is assumed to be equivalent to the TL-model.
`slInitScript`	`string`	The absolute or relative path to the script defining all parameters needed for initializing the SL-model. If not provided, the SL-model is assumed to be selfcontained.
`environment`	`string`	The absolute or relative path to XML file including information about environmental files like additional code. (see specifications in CodeGeneration.dtd).
`useExistingCode`	`boolean`	Determine if code generation needs to be done. 'true': Code generation is not explicitly performed. Instead it is assumed that the required code files are already generated and that the corresponding DataDictionary of the model includes information about the code generation. 'false': Code generation is explicitly performed during the analysis. The resulting code is used for further steps. Default is 'false'.
`activateModelLinking`	`boolean`	Determine if the source code needs to be linked to the TargetLink model. 'true': A link between the source code and the TargetLink model will be established. This setting may lead to a modified TargetLink model. To accomplish this link relation, the TargetLink option 'ExtendedBlockComments' needs to be enabled with a subsequent TargetLink code generation. 'false': The source code model linking is not explicitly set by EmbeddedPlatform. Default is 'false'.
`closedLoopModel`	`boolean`	Determine if the SUT envoronment is analyzed during extraction 'true': The environment of the SUT is also analyzed during the model extraction. Important for analyzing closed-loop models. 'false': Only the SUT is considered during the model extraction. Default is 'false'.
`fixedStepSolver`	`boolean`	Defines, if the fixed step solver will be set or not. 'true': The analyzed models (TargetLink model and Simulink model) will be set to the fixed-step solver type automatically. The usage of the EmbeddedPlatform requires a fixed-step solver. If the model is open and the fixed-step solver is not already set, this might lead to a modified model. 'false': The analyzed models (TargetLink model and Simulink model) will not be set to the fixed-step solver automatically. If the fixed-step solver is not already set in the model, an exception is thrown. In order to proceed the user has to set the fixed-step solver manually in the simulation settings. Note: The option is ignored if the model is currently not in an open/loaded state. In this case it has no visible side-effect. Default is 'false'.
`tlSubsystem`	`string`	Name of the Subsystem representing the TL toplevel subsystem for the analysis. Note: Argument is obligatory if there is more than one toplevel system in the model.
`calibrationHandling`	`enum`	**ALLOWED:** OFF, EXPLICIT_PARAMETER, LIMITED_BLOCKSET
		Determine how calibration variables are being handled. 'OFF': Only regular inputs in the interface of subsystems are observed. 'EXPLICIT_PARAMETER': Calibration

		variables are regarded as additional inputs to subsystems. Their value is set once during the initial phase of the simulation and is held constant thereafter. 'LIMITED_BLOCKSET': Calibration variables are regarded as additional inputs to subsystems. Their value is set once during the initial phase of the simulation and is held constant thereafter. Enable support for calibration within block properties, not using workspace calibrations. Note: The supported Set of calibratable TL-Blocks is different from 'EXPLICIT_PARAMETER'.Default is 'EXPLICIT_PARAMETER'.
testMode	enum	**ALLOWED:** BLACK_BOX, GREY_BOX
		Specifies the test mode. Can be 'BLACK_BOX' or 'GREY_BOX'. If set to 'GREY_BOX', local variables are regarded as additional outputs of subsystems. For Black Box-Testing only the regular outputs in the interfaces of subsystems are observed. Default is 'GREY_BOX'. If not specified (undefined or invalid), the default value is used.
mappingFileForTLArch	string	The absolute or relative path to the mapping file for TargetLink architecture mapping. At least one mapping file must be provided if architectures are already available in the profile. The file is used to map the first imported architecture to the new imported TargetLink architecture. If this file is not provided, the mapping to the first imported architecture is derived from the auto-generated mapping between TargetLink, C-Code, and Simulink (if available). For the derivation to work, at least one of these architectures must be mapped to the first imported architecture via a specified mapping file
mappingFileForCCodeArch	string	The absolute or relative path to the mapping file for C-Code architecture mapping. At least one mapping must be provided if architectures are already available in the profile. The file is used to map the first imported architecture to the new imported C-Code architecture. If this file is not provided, the mapping to the first imported architecture is derived from the auto-generated mapping between TargetLink, C-Code, and Simulink (if available). For the derivation to work, at least one of these architectures must be mapped to the first imported architecture via a specified mapping file
mappingFileForSLArch	string	The absolute or relative path to the mapping file for Simulink architecture mapping. At least one mapping must be provided if a Simulink architecture is additionally imported and<vr> architectures are already available in the profile. The file is used to map the first imported architecture to the new imported Simulink architecture. If this file is not provided, the mapping to the first imported architecture is derived from the auto-generated mapping between TargetLink, C-Code, and Simulink (if available). For the derivation to work, at least one of these architectures must be mapped to the first imported architecture via a specified mapping file
subsystemMatcher	string	A whitelist of all subsystems you would like to import. Uses the regular expression standard. Each subsystem is identified by its virtual path inside the model. If no value is defined, all subsystems are imported. If the specified regular expression does not produce any match, only the toplevel subsystem is imported. The root subsystem is always imported. Please note that the subsystem-path has to include the Targetlink wrapper.
calibrationMatcher	string	A whitelist of all calibrations you would like to import. Uses the regular expression standard. Each calibration is identified by its name. If no value is defined, all calibrations are imported. If the specified regular expression does not produce any match, no calibration is imported. The list cannot be applied, if calibrationHandling is set to 'OFF'.
cfileMatcher	string	A whitelist of all c-code files you would like to import. Uses the regular expression standard. Each file is identified by its name. If no value is defined, all files are imported. If the specified regular expression does not produce any match, no file is imported.

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

{
   jobID   string   **READ-ONLY**
                   The ID of a job.
}

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

2.13 POST /ep/architectures/userdefined

Import a User-defined architecture

Long running task Import a User-defined architecture from a XML file that describes the interface of the architecture.
It is only possible to have one User-defined architecture imported at all times. Importing a second User-defined architecture will be rejected.

REQUEST

REQUEST BODY - application/json
{
   modelFile*   string   Filename (and path) of the xml file that describes the interface of the User-defined architecture to import.
   mappingFile   string   Filename (and path) of the mapping xml file that describes the mapping between the User-defined architecture and the master architecture.
This file is only required (and mandatory), if the User-defined architecture is imported into a profile that already contains another architecture.

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json
{
   jobID   string   **READ-ONLY**
                    The ID of a job.
}

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

3. BACK-TO-BACK TEST REPORTS

3.1 POST /ep/b2b/{b2b-test-uid}/b2b-reports

Create a B2B test report on a B2B test

Creates a Back-to-Back test report on a Back-to-Back test by providing its UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*b2b-test-uid	string	The Back-to-Back test UID for which the Back-to-Back test report is created.

RESPONSE

STATUS CODE - 201: Created

RESPONSE MODEL - application/json

```
{
    uid          string    READ-ONLY
                           The unique identifier (UID) of this object.
    reportName   string    Name of the report.
    reportType   string    Type of the report.
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

3.2 GET /ep/b2b-reports

Get all B2B test reports

Retrieve all the Back-to-Back test reports from the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:
```
    uid          string    READ-ONLY
                           The unique identifier (UID) of this object.
    reportName   string    Name of the report.
```

```
    reportType  string   Type of the report.
}]
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

```
string
```

4. BACK-TO-BACK TESTS

Create and retrieve Back-to-Back tests.

4.1 GET /ep/b2b/{b2b-uid}

Get a B2B test
Get a Back-to-Back test with the provided UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*b2b-uid	string	The UID of the Back-to-Back test to be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
    uid                 string      READ-ONLY
                                    The unique identifier (UID) of this object.
    referenceMode       string      The reference execution config type.
    comparisonMode      string      The comparison execution config type.
    referenceFolderUIDs [string]    Reference folder UIDs
    comparisonFolderUID string      Comparison folder UID
    executionDate       string      Execution Date
    verdictStatus       enum        ALLOWED:PASSED, FAILED, ERROR, FAILED_ACCEPTED
                                    The verdict status
    verdictState        enum        ALLOWED:VALID, OUTDATED_TOLERANCE_UPDATE,
                                    OUTDATED_MISSING_EXECUTIONS
                                    Verdict State
    failed              integer     Number of failed comparisons.
    failedAccepted      integer     Number of failed accepted comparisons.
    passed              integer     Number of passed comparisons.
    error               integer     Number of comparisons with error.
    total               integer     Total number of comparisons.
    comparisons [{
    Array of object: All comparisons.

        uid                         string      READ-ONLY
                                                The unique identifier (UID) of this object.
        name                        string      The name of Test Case / Stimuli Vector used in Comparison.
        verdictStatus               enum        ALLOWED:PASSED, FAILED, ERROR, FAILED_ACCEPTED
                                                The verdict status
        referenceExecutionRecordUID string      UID of reference execution record.
        comparisonExecutionRecordUID string     UID of compared to execution record.
        comment                     string      Added comment for Comparison.
    }]
    name                string      The name of the Back-To-Back Test.
    stimuliFolderUIDs   [string]    Folder UIDs for which Back-To-Back Test was generated.
    stimuliScopeUIDs    [string]    Scope UIDs for which Back-To-Back Test was generated.
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

4.2 PATCH /ep/b2b/{b2b-uid}

Change a verdict status for a comparison

Changes verdict status for a comparison. If accept is true, the comparison verdict status is changed from 'failed' to 'failed (accepted)'. If accept is false, the comparison verdict status is changed from 'failed (accepted)'to 'failed'.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*b2b-uid	string	The Back-to-Back test UID for which to change the comparison verdict.

REQUEST BODY - application/json
```
{
    comparisonUID*    string    UID of the Comparison
    accept*           boolean   If accept is true, the comparison verdict status is changed from 'failed' to 'failed (accepted)'. If accept is
                                false, the comparison verdict status is changed from 'failed (accepted)'to 'failed'
}
```

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

4.3 POST /ep/folders/{folder-uid}/b2b

Create a B2B test on a folder

Long running task Generates a Back-to-Back test on a given folder UID for the specified reference and comparison execution kinds. Optionally, the execution can be forced to simulate all contained requirements-based test cases/stimuli-vectors. Optionally, only stimuli vectors can be set to be executed if force execution is activated. Otherwise if there are requirements-based test cases with comparison verdicts, the verdicts will be deleted in order to run the Back-to-Back test.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The folder UID for which the Back-to-Back test is generated.

REQUEST BODY - application/json

```
{
    refMode*                           string   Reference execution mode (e.g. 'SL MIL', 'TL MIL', 'TL MIL (EV)', 'SIL', 'PIL', 'SL SIL'). Not
                                                case-sensitive.
    compMode*                          string   Comparison execution mode (e.g. 'SL MIL', 'TL MIL', 'TL MIL (EV)', 'SIL', 'PIL', 'SL SIL').
                                                Not case-sensitive.
    forceExecute                       boolean  (Optional) If true, simulates all contained test cases/stimuli-vectors, replacing
                                                existing execution records. Default is false.
    simulateStimuliVectorsOnly  boolean  (Optional) This option can be used only when forceExecution is set to true. If true,
                                                simulates only stimuli-vectors. If false, and there are Requirements-based Test Cases
                                                with comparison verdicts, the verdicts will be deleted in order to run the Back-To-Back
                                                Test. Default value is false.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

4.4 POST /ep/folders/b2b

Create a B2B test on a list of folders

Long running task Generates a Back-to-Back test on a given list of folder UIDs for the specified reference and comparison execution kinds. Optionally, the execution can be forced to simulate all contained requirements-based test cases/stimuli-vectors. Optionally, only stimuli vectors can be set to be executed if force execution is activated. Otherwise if there are requirements-based test cases with comparison verdicts, the verdicts will be deleted in order to run the Back-to-

Back test.

REQUEST

REQUEST BODY - application/json

```
{
    refMode*                      string      Reference execution mode (e.g. 'SL MIL', 'TL MIL', 'TL MIL (EV)', 'SIL', 'PIL', 'SL SIL').
                                              Not case-sensitive.
    compMode*                     string      Comparison execution mode (e.g. 'SL MIL', 'TL MIL', 'TL MIL (EV)', 'SIL', 'PIL', 'SL SIL').
                                              Not case-sensitive.
    forceExecute                  boolean     (Optional) If true, simulates all contained test cases/stimuli-vectors, replacing
                                              existing execution records. Default is false.
    simulateStimuliVectorsOnly    boolean     (Optional) This option can be used only when forceExecution is set to true. If true,
                                              simulates only stimuli-vectors. If false, and there are Requirements-based Test
                                              Cases with comparison verdicts, the verdicts will be deleted in order to run the
                                              Back-To-Back Test. Default value is false.
    UIDs*                         [string]    UIDs list (e.g. scopes, folders)
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

4.5 POST /ep/scopes/{scope-uid}/b2b

Create a B2B test on a scope

Long running task Generates a Back-to-Back test on a given scope UID for the specified reference and comparison execution kinds. Optionally, the execution can be forced to simulate all contained requirements-based test cases/stimuli-vectors. Optionally, only stimuli vectors can be set to be executed if force execution is activated. Otherwise if there are requirements-based test cases with comparison verdicts, the verdicts will be deleted in order to run the Back-to-Back test.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The scope UID for which the Back-to-Back test is generated.

REQUEST BODY - application/json

{

refMode*	string	Reference execution mode (e.g. 'SL MIL', 'TL MIL', 'TL MIL (EV)', 'SIL', 'PIL', 'SL SIL'). Not case-sensitive.
compMode*	string	Comparison execution mode (e.g. 'SL MIL', 'TL MIL', 'TL MIL (EV)', 'SIL', 'PIL', 'SL SIL'). Not case-sensitive.
forceExecute	boolean	(Optional) If true, simulates all contained test cases/stimuli-vectors, replacing existing execution records. Default is false.
simulateStimuliVectorsOnly	boolean	(Optional) This option can be used only when forceExecution is set to true. If true, simulates only stimuli-vectors. If false, and there are Requirements-based Test Cases with comparison verdicts, the verdicts will be deleted in order to run the Back-To-Back Test. Default value is false.

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

{

jobID	string	READ-ONLY
		The ID of a job.

}

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

4.6 POST /ep/scopes/b2b

Create a B2B test on a list of scopes

Long running task Generates a Back-to-Back test on a given list of scope UIDs for the specified reference and comparison execution kinds. Optionally, the execution can be forced to simulate all contained requirements-based test cases/stimuli-vectors. Optionally, only stimuli vectors can be set to be executed if force execution is activated. Otherwise if there are requirements-based test cases with comparison verdicts, the verdicts will be deleted in order to run the Back-to-Back test.

REQUEST

REQUEST BODY - application/json

{

refMode*	string	Reference execution mode (e.g. 'SL MIL', 'TL MIL', 'TL MIL (EV)', 'SIL', 'PIL', 'SL SIL'). Not case-sensitive.

compMode*	string	Comparison execution mode (e.g. 'SL MIL', 'TL MIL', 'TL MIL (EV)', 'SIL', 'PIL', 'SL SIL'). Not case-sensitive.
forceExecute	boolean	(Optional) If true, simulates all contained test cases/stimuli-vectors, replacing existing execution records. Default is false.
simulateStimuliVectorsOnly	boolean	(Optional) This option can be used only when forceExecution is set to true. If true, simulates only stimuli-vectors. If false, and there are Requirements-based Test Cases with comparison verdicts, the verdicts will be deleted in order to run the Back-To-Back Test. Default value is false.
UIDs*	[string]	UIDs list (e.g. scopes, folders)

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

{

   jobID   string   **READ-ONLY**

                      The ID of a job.

}

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

4.7 GET /ep/b2b

Get all B2B tests

Get all Back-to-Back tests from active profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

[{

Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
referenceMode	string	The reference execution config type.
comparisonMode	string	The comparison execution config type.
referenceFolderUIDs	[string]	Reference folder UIDs
comparisonFolderUID	string	Comparison folder UID
executionDate	string	Execution Date

verdictStatus	enum	ALLOWED:PASSED, FAILED, ERROR, FAILED_ACCEPTED
		The verdict status
verdictState	enum	ALLOWED:VALID, OUTDATED_TOLERANCE_UPDATE, OUTDATED_MISSING_EXECUTIONS
		Verdict State
failed	integer	Number of failed comparisons.
failedAccepted	integer	Number of failed accepted comparisons.
passed	integer	Number of passed comparisons.
error	integer	Number of comparisons with error.
total	integer	Total number of comparisons.

comparisons [{

Array of object: All comparisons.

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
name	string	The name of Test Case / Stimuli Vector used in Comparison.
verdictStatus	enum	ALLOWED:PASSED, FAILED, ERROR, FAILED_ACCEPTED
		The verdict status
referenceExecutionRecordUID	string	UID of reference execution record.
comparisonExecutionRecordUID	string	UID of compared to execution record.
comment	string	Added comment for Comparison.

}]

name	string	The name of the Back-To-Back Test.
stimuliFolderUIDs	[string]	Folder UIDs for which Back-To-Back Test was generated.
stimuliScopeUIDs	[string]	Scope UIDs for which Back-To-Back Test was generated.

}]

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

5. CODE ANALYSIS REPORTS B2B

Handle Back-To-Back code analysis reports.

5.1 POST /ep/folders/{folder-uid}/code-analysis-reports-b2b

Create a report on a folder
Create a B2B code analysis report on given folder.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The folder UID for which to create the B2B code analysis report.

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

5.2 POST /ep/folders/code-analysis-reports-b2b

Create a report on a list of folders
Create B2B Code Analysis Report on given folders.

REQUEST

REQUEST BODY - application/json

```
{
    UIDs* [string]  List with unique identifiers of the objects.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                        The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

5.3 POST /ep/scopes/{scope-uid}/code-analysis-reports-b2b

Create a report on a scope

Create a B2B code analysis report on given scope.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The scope UID for which to create the B2B code analysis report.

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                        The ID of a job.
```

}

    STATUS CODE - 400: Bad request

        RESPONSE MODEL - text/plain

    string

    STATUS CODE - 403: Forbidden

        RESPONSE MODEL - text/plain

    string

    STATUS CODE - 404: Not found

        RESPONSE MODEL - text/plain

    string

    STATUS CODE - 500: Internal server error

        RESPONSE MODEL - text/plain

    string

5.4 GET /ep/code-analysis-reports-b2b

Get all reports

Retrieve all B2B code analysis reports from profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

    RESPONSE MODEL - application/json

    [{
    Array of object:

    uid   string   READ-ONLY
                               The unique identifier (UID) of this object.
    reportName string   Name of the report.
    reportType string   Type of the report.
    }]

STATUS CODE - 500: Internal server error.

    RESPONSE MODEL - text/plain

    string

---

6. CODE ANALYSIS REPORTS RBT

Creates requirement-based code analysis reports.

6.1 POST /ep/folders/{folder-uid}/code-analysis-reports-rbt

Create a report on a folder

Long running task Create an RBT code analysis report on a given folder.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The folder UID for which to create an RBT code analysis report.

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID    string    READ-ONLY
                       The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

6.2 POST /ep/folders/code-analysis-reports-rbt

Create a report on a list of folders

Create an RBT code analysis report on given folders.

REQUEST

REQUEST BODY - application/json

```
{
    UIDs* [string]  List with unique identifiers of the objects.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                        The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

6.3 POST /ep/requirements-sources/{requirements-source-uid}/code-analysis-reports-rbt

Create a report on a requirements source

Create an RBT code analysis report on a given requirements source UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*requirements-source-uid	string	The requirements source UID for which to create RBT code analysis report.

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
```

jobID string **READ-ONLY**

                               The ID of a job.

    }

STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

    RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

6.4 POST /ep/requirements-sources/code-analysis-reports-rbt

Create a report on requirements sources

Create an RBT code analysis report on given requirements sources.

REQUEST

REQUEST BODY - application/json
{
    UIDs* [string] List with unique identifiers of the objects.
}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

    RESPONSE MODEL - application/json

    {
        jobID string **READ-ONLY**

                               The ID of a job.

    }

STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

    RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

6.5 POST /ep/scopes/{scope-uid}/code-analysis-reports-rbt

Create a report on scope

Create an RBT code analysis report on a given scope.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The scope UID for which to create RBT code analysis report.

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

6.6 GET /ep/code-analysis-reports-rbt

Get all reports

Retrieve all RBT code analysis reports from the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
reportName	string	Name of the report.
reportType	string	Type of the report.

```
}]
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

7. CODE COVERAGE/ROBUSTNESS CHECK B2B

Retrieve code coverage/robustness checks details and results in B2B.

7.1 GET /ep/scopes/{scope-uid}/coverage-details-b2b

Get coverage details for a scope

Get all code coverage/robustness checks details for a scope. Some filters can be applied.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
statuses	array of string ALLOWED: COVERED, UNKNOWN, UNREACHABLE_INF, UNREACHABLE_N, INCONSISTENT, UNKNOWN_JUSTIFIED, UNREACHABLE_N_JUSTIFIED, UNREACHABLE_INF_JUSTIFIED	The status filter for code coverage/robustness check goals. Possible values: COVERED, UNKNOWN, UNREACHABLE_INF, UNREACHABLE_N, INCONSISTENT, UNKNOWN_JUSTIFIED, UNREACHABLE_INF_JUSTIFIED, UNREACHABLE_N_JUSTIFIED. Can also specify multiple options. Can also be empty, in which case the results are shown for all statuses.
goal-types	array of string ALLOWED: STM, D, C, MCDC, F, FC, SC, RO, CDC, DZ, CA	The goal type filter for code coverage/robustness check goals. Possible values for code coverage goals: STM(Statement), C(Condition), D(Decision/Branch), CDC(C/DC), MCDC(C/DC and MC/DC), F(Function), SC(Switch-Case), RO(Relational Operator), FC(Function Call). Possible values for robustness check goals are: DZ(Division by Zero), CA(Downcast). Can also specify multiple options. Can also be empty, in which case the results are shown for all goal types.
files	array of string	List of file filters for code coverage/robustness check goals. If list is empty the results are shown for all files.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

pll	string	PLL string of the coverage goal.
type	string	Goal type of the coverage goal.
line	integer	The line number of the location where the coverage goal is located in the file.
file	string	The file name where the coverage property can be located.

```
  properties [{
```
Array of object: A list with coverage goal properties.

pll	string	PLL string of the coverage property.
status	string	Status of the coverage property.
coveringVectors	[string]	List of string vector names that cover the property.

```
  }]
```

```
      expression  string      Expression of the coverage goal.
      blocks      [string]    The TargetLink blocks
      comment     string      The comment.
      justified   boolean     If this goal is justified
  }]
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

7.2 GET /ep/scopes/{scope-uid}/coverage-results-b2b

Get coverage results for a scope

Get the code coverage and robustness checks results for a scope. Goal type filters can be applied.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
goal-types	array of string ALLOWED: STM, D, C, MCDC, F, FC, SC, RO, CDC, DZ, CA	The goal type filter for code coverage/robustness check goals. Possible values for code coverage goals: STM(Statement), C(Condition), D(Decision/Branch), CDC(C/DC), MCDC(C/DC and MC/DC), F(Function), SC(Switch-Case), RO(Relational Operator), FC(Function Call). Possible values for robustness check goals are: DZ(Division by Zero), CA(Downcast). Can also specify multiple options. Can also be empty, in which case the results are shown for all goal types.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
  CDCCoverage {
  CDC goal coverage information.

      coverageGoal                       string    Name of the goal
      coveredCompletelyCount             integer   Coverage complete count
      coveredCompletelyPercentage        number    Coverage complete percentage
      coveredPartiallyCount              integer   Coverage partial count
      coveredPartiallyPercentage         number    Coverage partial percentage
                                                   Handled complete count
```

```
        handledCompletelyCount              integer
        handledCompletelyPercentage         number    Handled complete percentage
        handledPartiallyCount               integer   Handled partial count
        handledPartiallyPercentage          number    Handled partial percentage
        unhandledCount                      integer   Unhandled count
        unhandledPercentage                 number    Unhandled percentage
        uncoveredCount                      integer   Uncovered count
        uncoveredPercentage                 number    Uncovered percentage
        justifiedCompletelyCount            integer   Covered completely (justified) count
        justifiedCompletelyPercentage       number    Covered completely (justified) percentage
        justifiedPartiallyCount             integer   Covered partially (justified) count
        justifiedPartiallyPercentage        number    Covered partially (justified) percentage
        totalCount                          integer   Total count
    }
    CDCPropertyCoverage {
    CDC property coverage information.

        coverageGoal                              string    Name of the goal
        coveredCount                              integer   Covered count
        coveredPercentage                         number    Covered percentage
        unreachableInfiniteCount                  integer   Unreachable Infinite count
        unreachableInfinitePercentage             number    Unreachable Infinite percentage
        unreachableNCount                         integer   Unreachable N count
        unreachableNPercentage                    number    Unreachable N percentage
        unreachableInfiniteJustifiedCount         integer   Unreachable Infinite (justified) count
        unreachableInfiniteJustifiedPercentage    number    Unreachable Infinite (justified) percentage
        unreachableNJustifiedCount                integer   Unreachable N (justified) count
        unreachableNJustifiedPercentage           number    Unreachable N (justified) percentage
        unknownJustifiedCount                     integer   Unknown (justified) count
        unknownJustifiedPercentage                number    Unknown (justified) percentage
        unknownCount                              integer   Unknown count
        unknownPercentage                         number    Unknown percentage
        handledCount                              integer   Handled count
        handledPercentage                         number    Handled percentage
        inconsistentCount                         integer   Inconsistent count
        inconsistentPercentage                    number    Inconsistent percentage
        unreachableCount                          integer   Unreachable count
        unreachablePercentage                     number    Unreachable percentage
        totalCount                                integer   Total count
        comment                                   string    The comment
    }
    ConditionCoverage {
    Condition goal coverage information.

        coverageGoal                        string    Name of the goal
        coveredCompletelyCount              integer   Coverage complete count
        coveredCompletelyPercentage         number    Coverage complete percentage
        coveredPartiallyCount               integer   Coverage partial count
        coveredPartiallyPercentage          number    Coverage partial percentage
        handledCompletelyCount              integer   Handled complete count
        handledCompletelyPercentage         number    Handled complete percentage
        handledPartiallyCount               integer   Handled partial count
        handledPartiallyPercentage          number    Handled partial percentage
        unhandledCount                      integer   Unhandled count
        unhandledPercentage                 number    Unhandled percentage
        uncoveredCount                      integer   Uncovered count
        uncoveredPercentage                 number    Uncovered percentage
        justifiedCompletelyCount            integer   Covered completely (justified) count
        justifiedCompletelyPercentage       number    Covered completely (justified) percentage
```

```
        justifiedPartiallyCount              integer   Covered partially (justified) count
        justifiedPartiallyPercentage         number    Covered partially (justified) percentage
        totalCount                           integer   Total count
    }
    ConditionPropertyCoverage {
    Condition property coverage information.

        coverageGoal                                  string    Name of the goal
        coveredCount                                  integer   Covered count
        coveredPercentage                             number    Covered percentage
        unreachableInfiniteCount                      integer   Unreachable Infinite count
        unreachableInfinitePercentage                 number    Unreachable Infinite percentage
        unreachableNCount                             integer   Unreachable N count
        unreachableNPercentage                        number    Unreachable N percentage
        unreachableInfiniteJustifiedCount             integer   Unreachable Infinite (justified) count
        unreachableInfiniteJustifiedPercentage        number    Unreachable Infinite (justified) percentage
        unreachableNJustifiedCount                    integer   Unreachable N (justified) count
        unreachableNJustifiedPercentage               number    Unreachable N (justified) percentage
        unknownJustifiedCount                         integer   Unknown (justified) count
        unknownJustifiedPercentage                    number    Unknown (justified) percentage
        unknownCount                                  integer   Unknown count
        unknownPercentage                             number    Unknown percentage
        handledCount                                  integer   Handled count
        handledPercentage                             number    Handled percentage
        inconsistentCount                             integer   Inconsistent count
        inconsistentPercentage                        number    Inconsistent percentage
        unreachableCount                              integer   Unreachable count
        unreachablePercentage                         number    Unreachable percentage
        totalCount                                    integer   Total count
        comment                                       string    The comment
    }
    DecisionCoverage {
    Decision goal coverage information.

        coverageGoal                         string    Name of the goal
        coveredCompletelyCount               integer   Coverage complete count
        coveredCompletelyPercentage          number    Coverage complete percentage
        coveredPartiallyCount                integer   Coverage partial count
        coveredPartiallyPercentage           number    Coverage partial percentage
        handledCompletelyCount               integer   Handled complete count
        handledCompletelyPercentage          number    Handled complete percentage
        handledPartiallyCount                integer   Handled partial count
        handledPartiallyPercentage           number    Handled partial percentage
        unhandledCount                       integer   Unhandled count
        unhandledPercentage                  number    Unhandled percentage
        uncoveredCount                       integer   Uncovered count
        uncoveredPercentage                  number    Uncovered percentage
        justifiedCompletelyCount             integer   Covered completely (justified) count
        justifiedCompletelyPercentage        number    Covered completely (justified) percentage
        justifiedPartiallyCount              integer   Covered partially (justified) count
        justifiedPartiallyPercentage         number    Covered partially (justified) percentage
        totalCount                           integer   Total count
    }
    DecisionPropertyCoverage {
    Decision property coverage information.

        coverageGoal                                  string    Name of the goal
        coveredCount                                  integer   Covered count
        coveredPercentage                             number    Covered percentage
        unreachableInfiniteCount                      integer   Unreachable Infinite count
```

unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}
FunctionCoverage {
Function goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}
FunctionPropertyCoverage {
Function property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count

unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}
FunctionCallCoverage {
Function call goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}
FunctionCallPropertyCoverage {
Function call property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

```
MCDCCoverage {
  MCDC goal coverage information.

  coverageGoal                          string    Name of the goal
  coveredCompletelyCount                integer   Coverage complete count
  coveredCompletelyPercentage           number    Coverage complete percentage
  coveredPartiallyCount                 integer   Coverage partial count
  coveredPartiallyPercentage            number    Coverage partial percentage
  handledCompletelyCount                integer   Handled complete count
  handledCompletelyPercentage           number    Handled complete percentage
  handledPartiallyCount                 integer   Handled partial count
  handledPartiallyPercentage            number    Handled partial percentage
  unhandledCount                        integer   Unhandled count
  unhandledPercentage                   number    Unhandled percentage
  uncoveredCount                        integer   Uncovered count
  uncoveredPercentage                   number    Uncovered percentage
  justifiedCompletelyCount              integer   Covered completely (justified) count
  justifiedCompletelyPercentage         number    Covered completely (justified) percentage
  justifiedPartiallyCount               integer   Covered partially (justified) count
  justifiedPartiallyPercentage          number    Covered partially (justified) percentage
  totalCount                            integer   Total count
}
MCDCPropertyCoverage {
  MCDC property coverage information.

  coverageGoal                             string    Name of the goal
  coveredCount                             integer   Covered count
  coveredPercentage                        number    Covered percentage
  unreachableInfiniteCount                 integer   Unreachable Infinite count
  unreachableInfinitePercentage            number    Unreachable Infinite percentage
  unreachableNCount                        integer   Unreachable N count
  unreachableNPercentage                   number    Unreachable N percentage
  unreachableInfiniteJustifiedCount        integer   Unreachable Infinite (justified) count
  unreachableInfiniteJustifiedPercentage   number    Unreachable Infinite (justified) percentage
  unreachableNJustifiedCount               integer   Unreachable N (justified) count
  unreachableNJustifiedPercentage          number    Unreachable N (justified) percentage
  unknownJustifiedCount                    integer   Unknown (justified) count
  unknownJustifiedPercentage               number    Unknown (justified) percentage
  unknownCount                             integer   Unknown count
  unknownPercentage                        number    Unknown percentage
  handledCount                             integer   Handled count
  handledPercentage                        number    Handled percentage
  inconsistentCount                        integer   Inconsistent count
  inconsistentPercentage                   number    Inconsistent percentage
  unreachableCount                         integer   Unreachable count
  unreachablePercentage                    number    Unreachable percentage
  totalCount                               integer   Total count
  comment                                  string    The comment
}
RelationalOperatorCoverage {
  Relational operation goal coverage information.

  coverageGoal                          string    Name of the goal
  coveredCompletelyCount                integer   Coverage complete count
  coveredCompletelyPercentage           number    Coverage complete percentage
  coveredPartiallyCount                 integer   Coverage partial count
  coveredPartiallyPercentage            number    Coverage partial percentage
  handledCompletelyCount                integer   Handled complete count
  handledCompletelyPercentage           number    Handled complete percentage
  handledPartiallyCount                 integer   Handled partial count
```

handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

RelationalOperatorPropertyCoverage {

Relational operation property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

StatementCoverage {

Statement goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

```
}
StatementPropertyCoverage {
Statement property coverage information.
    coverageGoal                                    string    Name of the goal
    coveredCount                                    integer   Covered count
    coveredPercentage                               number    Covered percentage
    unreachableInfiniteCount                        integer   Unreachable Infinite count
    unreachableInfinitePercentage                   number    Unreachable Infinite percentage
    unreachableNCount                               integer   Unreachable N count
    unreachableNPercentage                          number    Unreachable N percentage
    unreachableInfiniteJustifiedCount               integer   Unreachable Infinite (justified) count
    unreachableInfiniteJustifiedPercentage          number    Unreachable Infinite (justified) percentage
    unreachableNJustifiedCount                      integer   Unreachable N (justified) count
    unreachableNJustifiedPercentage                 number    Unreachable N (justified) percentage
    unknownJustifiedCount                           integer   Unknown (justified) count
    unknownJustifiedPercentage                      number    Unknown (justified) percentage
    unknownCount                                    integer   Unknown count
    unknownPercentage                               number    Unknown percentage
    handledCount                                    integer   Handled count
    handledPercentage                               number    Handled percentage
    inconsistentCount                               integer   Inconsistent count
    inconsistentPercentage                          number    Inconsistent percentage
    unreachableCount                                integer   Unreachable count
    unreachablePercentage                           number    Unreachable percentage
    totalCount                                      integer   Total count
    comment                                         string    The comment
}
SwitchCaseCoverage {
Switch Case goal coverage information.
    coverageGoal                         string    Name of the goal
    coveredCompletelyCount               integer   Coverage complete count
    coveredCompletelyPercentage          number    Coverage complete percentage
    coveredPartiallyCount                integer   Coverage partial count
    coveredPartiallyPercentage           number    Coverage partial percentage
    handledCompletelyCount               integer   Handled complete count
    handledCompletelyPercentage          number    Handled complete percentage
    handledPartiallyCount                integer   Handled partial count
    handledPartiallyPercentage           number    Handled partial percentage
    unhandledCount                       integer   Unhandled count
    unhandledPercentage                  number    Unhandled percentage
    uncoveredCount                       integer   Uncovered count
    uncoveredPercentage                  number    Uncovered percentage
    justifiedCompletelyCount             integer   Covered completely (justified) count
    justifiedCompletelyPercentage        number    Covered completely (justified) percentage
    justifiedPartiallyCount              integer   Covered partially (justified) count
    justifiedPartiallyPercentage         number    Covered partially (justified) percentage
    totalCount                           integer   Total count
}
SwitchCasePropertyCoverage {
Switch Case property coverage information.
    coverageGoal                                    string    Name of the goal
    coveredCount                                    integer   Covered count
    coveredPercentage                               number    Covered percentage
    unreachableInfiniteCount                        integer   Unreachable Infinite count
    unreachableInfinitePercentage                   number    Unreachable Infinite percentage
    unreachableNCount                               integer   Unreachable N count
    unreachableNPercentage                          number    Unreachable N percentage
```

unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

DivisionByZeroCoverage {
Division By Zero goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

DivisionByZeroPropertyCoverage {
Division By Zero property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage

inconsistentCount		integer	Inconsistent count
inconsistentPercentage		number	Inconsistent percentage
unreachableCount		integer	Unreachable count
unreachablePercentage		number	Unreachable percentage
totalCount		integer	Total count
comment		string	The comment

}

DownCastCoverage {
DownCast goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

DownCastPropertyCoverage {
DownCast property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

codeCoverageComment	string	The code coverage overview comment.
robustnessCoverageComment	string	The robustness coverage overview comment.

}

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

7.3 POST /ep/coverage-goals/set-comment-b2b

Set goal comments

Set comments of code coverage and robustness check goals.

REQUEST

REQUEST BODY - application/json

```
[{
```
Array of object:

    pll*   string   The PLL of the code coverage or robustness goal for which to set the comment.

    comment* string   The comment to set on the goal with the given PLL.

```
}]
```

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

7.4 POST /ep/coverage-goals/set-justified-b2b

Set goal justified status

Set justified status of code coverage and robustness check goals.

REQUEST

REQUEST BODY - application/json

```
{
```
    justified boolean   The desired state of justify for the supplied goals.

    plls   [string] The Plls of the goals

```
}
```

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

7.5 POST /ep/scopes/{scope-uid}/set-coverage-overview-comment-b2b

Set overview comments

Set the comments of code coverage overview sections.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope

REQUEST BODY - application/json

[{

Array of object:

    type*   enum   **ALLOWED:**CC_STAT, STM, D, C, MCDC, F, FC, SC, RO, RC_STAT, DZ, CA

    The type of overview for which to set the comment. Possible values for code coverage goals: CC_STAT(Code Coverage Statistics), STM(Statement), D(Decision/Branch), C(Condition), MCDC(C/DC and MC/DC), F(Function), FC(Function Call). SC(Switch-Case), RO(Relational Operator), Possible values for robustness check goals are: RC_STAT(Robustness Check Statistics) ,DZ(Division by Zero), CA(Downcast). Can also specify multiple options.

    comment* string The comment to set for the overview type.

}]

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

8. CODE COVERAGE/ROBUSTNESS CHECK RBT

Retrieve code coverage/robustness checks details and results in RBT.

8.1 GET /ep/requirements-sources/{requirement-source-uid}/coverage-details-rbt

Get coverage details for a requirement source

Get code coverage/robustness checks details for a requirement source. Some filters can be applied.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*requirement-source-uid	string	The UID of the requirement source

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
statuses	array of string ALLOWED: COVERED, UNKNOWN, UNREACHABLE_INF, UNREACHABLE_N, INCONSISTENT, UNKNOWN_JUSTIFIED, UNREACHABLE_N_JUSTIFIED, UNREACHABLE_INF_JUSTIFIED	The status filter for code coverage/robustness check goals. Possible values: COVERED, UNKNOWN, UNREACHABLE_INF, UNREACHABLE_N, INCONSISTENT, UNKNOWN_JUSTIFIED, UNREACHABLE_INF_JUSTIFIED, UNREACHABLE_N_JUSTIFIED. Can also specify multiple options. Can also be empty, in which case the results are shown for all statuses.
goal-types	array of string ALLOWED: STM, D, C, MCDC, F, FC, SC, RO, CDC, DZ, CA	The goal type filter for code coverage/robustness check goals. Possible values for code coverage goals: STM(Statement), C(Condition), D(Decision/Branch), CDC(C/DC), MCDC(C/DC and MC/DC), F(Function), SC(Switch-Case), RO(Relational Operator), FC(Function Call). Possible values for robustness check goals are: DZ(Division by Zero), CA(Downcast). Can also specify multiple options. Can also be empty, in which case the results are shown for all goal types.
files	array of string	List of file filters for code coverage/robustness check goals. If list is empty the results are shown for all files.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

pll	string	PLL string of the coverage goal.
type	string	Goal type of the coverage goal.
line	integer	The line number of the location where the coverage goal is located in the file.
file	string	The file name where the coverage property can be located.

```
properties [{
```
Array of object: A list with coverage goal properties.

pll	string	PLL string of the coverage property.
status	string	Status of the coverage property.
coveringVectors	[string]	List of string vector names that cover the property.

```
    }]
    expression  string      Expression of the coverage goal.
    blocks      [string]    The TargetLink blocks
    comment     string      The comment.
    justified   boolean     If this goal is justified
  }]
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

8.2 GET /ep/scopes/{scope-uid}/coverage-details-rbt

Get coverage details for a scope

Get all code coverage/robustness checks details for a scope. Some filters can be applied.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
statuses	array of string ALLOWED: COVERED, UNKNOWN, UNREACHABLE_INF, UNREACHABLE_N, INCONSISTENT, UNKNOWN_JUSTIFIED, UNREACHABLE_N_JUSTIFIED, UNREACHABLE_INF_JUSTIFIED	The status filter for code coverage/robustness check goals. Possible values: COVERED, UNKNOWN, UNREACHABLE_INF, UNREACHABLE_N, INCONSISTENT, UNKNOWN_JUSTIFIED, UNREACHABLE_INF_JUSTIFIED, UNREACHABLE_N_JUSTIFIED. Can also specify multiple options. Can also be empty, in which case the results are shown for all statuses.
goal-types	array of string ALLOWED: STM, D, C, MCDC, F, FC, SC, RO, CDC, DZ, CA	The goal type filter for code coverage/robustness check goals. Possible values for code coverage goals: STM(Statement), C(Condition), D(Decision/Branch), CDC(C/DC), MCDC(C/DC and MC/DC), F(Function), SC(Switch-Case), RO(Relational Operator), FC(Function Call). Possible values for robustness check goals are: DZ(Division by Zero), CA(Downcast). Can also specify multiple options. Can also be empty, in which case the results are shown for all goal types.
files	array of string	List of file filters for code coverage/robustness check goals. If list is empty the results are shown for all files.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

pll	string	PLL string of the coverage goal.
type	string	Goal type of the coverage goal.
line	integer	The line number of the location where the coverage goal is located in the file.
file	string	The file name where the coverage property can be located.

```
  properties [{
```
Array of object: A list with coverage goal properties.

pll	string	PLL string of the coverage property.
status	string	Status of the coverage property.
coveringVectors	[string]	List of string vector names that cover the property.

```
  }]
```

expression	string	Expression of the coverage goal.
blocks	[string]	The TargetLink blocks
comment	string	The comment.
justified	boolean	If this goal is justified

```
}]
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

8.3 GET /ep/requirements-sources/{requirements-source-uid}/coverage-results-rbt

Get coverage results for a requirement source

Get the code coverage and robustness checks results for a requirement source. Goal type filters can be applied.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*requirements-source-uid	string	The UID of the requirement source

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
goal-types	array of string ALLOWED: STM, D, C, MCDC, F, FC, SC, RO, CDC, DZ, CA	The goal type filter for code coverage/robustness check goals. Possible values for code coverage goals: STM(Statement), C(Condition), D(Decision/Branch), CDC(C/DC), MCDC(C/DC and MC/DC), F(Function), SC(Switch-Case), RO(Relational Operator), FC(Function Call). Possible values for robustness check goals are: DZ(Division by Zero), CA(Downcast). Can also specify multiple options. Can also be empty, in which case the results are shown for all goal types.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
  CDCCoverage {
```
CDC goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

```
  }
  CDCPropertyCoverage {
```
CDC property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage

```
        unreachableCount                                      integer   Unreachable count
        unreachablePercentage                                 number    Unreachable percentage
        totalCount                                            integer   Total count
        comment                                               string    The comment
    }
    ConditionCoverage {
    Condition goal coverage information.

        coverageGoal                          string    Name of the goal
        coveredCompletelyCount                integer   Coverage complete count
        coveredCompletelyPercentage           number    Coverage complete percentage
        coveredPartiallyCount                 integer   Coverage partial count
        coveredPartiallyPercentage            number    Coverage partial percentage
        handledCompletelyCount                integer   Handled complete count
        handledCompletelyPercentage           number    Handled complete percentage
        handledPartiallyCount                 integer   Handled partial count
        handledPartiallyPercentage            number    Handled partial percentage
        unhandledCount                        integer   Unhandled count
        unhandledPercentage                   number    Unhandled percentage
        uncoveredCount                        integer   Uncovered count
        uncoveredPercentage                   number    Uncovered percentage
        justifiedCompletelyCount              integer   Covered completely (justified) count
        justifiedCompletelyPercentage         number    Covered completely (justified) percentage
        justifiedPartiallyCount               integer   Covered partially (justified) count
        justifiedPartiallyPercentage          number    Covered partially (justified) percentage
        totalCount                            integer   Total count
    }
    ConditionPropertyCoverage {
    Condition property coverage information.

        coverageGoal                                  string    Name of the goal
        coveredCount                                  integer   Covered count
        coveredPercentage                             number    Covered percentage
        unreachableInfiniteCount                      integer   Unreachable Infinite count
        unreachableInfinitePercentage                 number    Unreachable Infinite percentage
        unreachableNCount                             integer   Unreachable N count
        unreachableNPercentage                        number    Unreachable N percentage
        unreachableInfiniteJustifiedCount             integer   Unreachable Infinite (justified) count
        unreachableInfiniteJustifiedPercentage        number    Unreachable Infinite (justified) percentage
        unreachableNJustifiedCount                    integer   Unreachable N (justified) count
        unreachableNJustifiedPercentage               number    Unreachable N (justified) percentage
        unknownJustifiedCount                         integer   Unknown (justified) count
        unknownJustifiedPercentage                    number    Unknown (justified) percentage
        unknownCount                                  integer   Unknown count
        unknownPercentage                             number    Unknown percentage
        handledCount                                  integer   Handled count
        handledPercentage                             number    Handled percentage
        inconsistentCount                             integer   Inconsistent count
        inconsistentPercentage                        number    Inconsistent percentage
        unreachableCount                              integer   Unreachable count
        unreachablePercentage                         number    Unreachable percentage
        totalCount                                    integer   Total count
        comment                                       string    The comment
    }
    DecisionCoverage {
    Decision goal coverage information.

        coverageGoal                          string    Name of the goal
        coveredCompletelyCount                integer   Coverage complete count
        coveredCompletelyPercentage           number    Coverage complete percentage
```

coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

```
DecisionPropertyCoverage {
```
Decision property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

```
FunctionCoverage {
```
Function goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage

```
      justifiedCompletelyCount            integer  Covered completely (justified) count
      justifiedCompletelyPercentage       number   Covered completely (justified) percentage
      justifiedPartiallyCount             integer  Covered partially (justified) count
      justifiedPartiallyPercentage        number   Covered partially (justified) percentage
      totalCount                          integer  Total count
  }
  FunctionPropertyCoverage {
  Function property coverage information.

      coverageGoal                                    string   Name of the goal
      coveredCount                                    integer  Covered count
      coveredPercentage                               number   Covered percentage
      unreachableInfiniteCount                        integer  Unreachable Infinite count
      unreachableInfinitePercentage                   number   Unreachable Infinite percentage
      unreachableNCount                               integer  Unreachable N count
      unreachableNPercentage                          number   Unreachable N percentage
      unreachableInfiniteJustifiedCount               integer  Unreachable Infinite (justified) count
      unreachableInfiniteJustifiedPercentage          number   Unreachable Infinite (justified) percentage
      unreachableNJustifiedCount                      integer  Unreachable N (justified) count
      unreachableNJustifiedPercentage                 number   Unreachable N (justified) percentage
      unknownJustifiedCount                           integer  Unknown (justified) count
      unknownJustifiedPercentage                      number   Unknown (justified) percentage
      unknownCount                                    integer  Unknown count
      unknownPercentage                               number   Unknown percentage
      handledCount                                    integer  Handled count
      handledPercentage                               number   Handled percentage
      inconsistentCount                               integer  Inconsistent count
      inconsistentPercentage                          number   Inconsistent percentage
      unreachableCount                                integer  Unreachable count
      unreachablePercentage                           number   Unreachable percentage
      totalCount                                      integer  Total count
      comment                                         string   The comment
  }
  FunctionCallCoverage {
  Function call goal coverage information.

      coverageGoal                        string   Name of the goal
      coveredCompletelyCount              integer  Coverage complete count
      coveredCompletelyPercentage         number   Coverage complete percentage
      coveredPartiallyCount               integer  Coverage partial count
      coveredPartiallyPercentage          number   Coverage partial percentage
      handledCompletelyCount              integer  Handled complete count
      handledCompletelyPercentage         number   Handled complete percentage
      handledPartiallyCount               integer  Handled partial count
      handledPartiallyPercentage          number   Handled partial percentage
      unhandledCount                      integer  Unhandled count
      unhandledPercentage                 number   Unhandled percentage
      uncoveredCount                      integer  Uncovered count
      uncoveredPercentage                 number   Uncovered percentage
      justifiedCompletelyCount            integer  Covered completely (justified) count
      justifiedCompletelyPercentage       number   Covered completely (justified) percentage
      justifiedPartiallyCount             integer  Covered partially (justified) count
      justifiedPartiallyPercentage        number   Covered partially (justified) percentage
      totalCount                          integer  Total count
  }
  FunctionCallPropertyCoverage {
  Function call property coverage information.

      coverageGoal                                    string   Name of the goal
      coveredCount                                    integer  Covered count
```

coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

MCDCCoverage {
MCDC goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

MCDCPropertyCoverage {
MCDC property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count

unknownJustifiedPercentage	number	Unknown (justified) percentage	
unknownCount	integer	Unknown count	
unknownPercentage	number	Unknown percentage	
handledCount	integer	Handled count	
handledPercentage	number	Handled percentage	
inconsistentCount	integer	Inconsistent count	
inconsistentPercentage	number	Inconsistent percentage	
unreachableCount	integer	Unreachable count	
unreachablePercentage	number	Unreachable percentage	
totalCount	integer	Total count	
comment	string	The comment	

}

RelationalOperatorCoverage {

Relational operation goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

RelationalOperatorPropertyCoverage {

Relational operation property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count

```
        comment                                        string    The comment
    }
    StatementCoverage {
    Statement goal coverage information.
        coverageGoal                              string    Name of the goal
        coveredCompletelyCount                    integer   Coverage complete count
        coveredCompletelyPercentage               number    Coverage complete percentage
        coveredPartiallyCount                     integer   Coverage partial count
        coveredPartiallyPercentage                number    Coverage partial percentage
        handledCompletelyCount                    integer   Handled complete count
        handledCompletelyPercentage               number    Handled complete percentage
        handledPartiallyCount                     integer   Handled partial count
        handledPartiallyPercentage                number    Handled partial percentage
        unhandledCount                            integer   Unhandled count
        unhandledPercentage                       number    Unhandled percentage
        uncoveredCount                            integer   Uncovered count
        uncoveredPercentage                       number    Uncovered percentage
        justifiedCompletelyCount                  integer   Covered completely (justified) count
        justifiedCompletelyPercentage             number    Covered completely (justified) percentage
        justifiedPartiallyCount                   integer   Covered partially (justified) count
        justifiedPartiallyPercentage              number    Covered partially (justified) percentage
        totalCount                                integer   Total count
    }
    StatementPropertyCoverage {
    Statement property coverage information.
        coverageGoal                                        string    Name of the goal
        coveredCount                                        integer   Covered count
        coveredPercentage                                   number    Covered percentage
        unreachableInfiniteCount                            integer   Unreachable Infinite count
        unreachableInfinitePercentage                       number    Unreachable Infinite percentage
        unreachableNCount                                   integer   Unreachable N count
        unreachableNPercentage                              number    Unreachable N percentage
        unreachableInfiniteJustifiedCount                   integer   Unreachable Infinite (justified) count
        unreachableInfiniteJustifiedPercentage              number    Unreachable Infinite (justified) percentage
        unreachableNJustifiedCount                          integer   Unreachable N (justified) count
        unreachableNJustifiedPercentage                     number    Unreachable N (justified) percentage
        unknownJustifiedCount                               integer   Unknown (justified) count
        unknownJustifiedPercentage                          number    Unknown (justified) percentage
        unknownCount                                        integer   Unknown count
        unknownPercentage                                   number    Unknown percentage
        handledCount                                        integer   Handled count
        handledPercentage                                   number    Handled percentage
        inconsistentCount                                   integer   Inconsistent count
        inconsistentPercentage                              number    Inconsistent percentage
        unreachableCount                                    integer   Unreachable count
        unreachablePercentage                               number    Unreachable percentage
        totalCount                                          integer   Total count
        comment                                             string    The comment
    }
    SwitchCaseCoverage {
    Switch Case goal coverage information.
        coverageGoal                              string    Name of the goal
        coveredCompletelyCount                    integer   Coverage complete count
        coveredCompletelyPercentage               number    Coverage complete percentage
        coveredPartiallyCount                     integer   Coverage partial count
        coveredPartiallyPercentage                number    Coverage partial percentage
        handledCompletelyCount                    integer   Handled complete count
```

handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

SwitchCasePropertyCoverage {
Switch Case property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

DivisionByZeroCoverage {
Division By Zero goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count

```
      justifiedPartiallyPercentage          number    Covered partially (justified) percentage
      totalCount                            integer   Total count
  }
  DivisionByZeroPropertyCoverage {
  Division By Zero property coverage information.
      coverageGoal                              string    Name of the goal
      coveredCount                              integer   Covered count
      coveredPercentage                         number    Covered percentage
      unreachableInfiniteCount                  integer   Unreachable Infinite count
      unreachableInfinitePercentage             number    Unreachable Infinite percentage
      unreachableNCount                         integer   Unreachable N count
      unreachableNPercentage                    number    Unreachable N percentage
      unreachableInfiniteJustifiedCount         integer   Unreachable Infinite (justified) count
      unreachableInfiniteJustifiedPercentage    number    Unreachable Infinite (justified) percentage
      unreachableNJustifiedCount                integer   Unreachable N (justified) count
      unreachableNJustifiedPercentage           number    Unreachable N (justified) percentage
      unknownJustifiedCount                     integer   Unknown (justified) count
      unknownJustifiedPercentage                number    Unknown (justified) percentage
      unknownCount                              integer   Unknown count
      unknownPercentage                         number    Unknown percentage
      handledCount                              integer   Handled count
      handledPercentage                         number    Handled percentage
      inconsistentCount                         integer   Inconsistent count
      inconsistentPercentage                    number    Inconsistent percentage
      unreachableCount                          integer   Unreachable count
      unreachablePercentage                     number    Unreachable percentage
      totalCount                                integer   Total count
      comment                                   string    The comment
  }
  DownCastCoverage {
  DownCast goal coverage information.
      coverageGoal                      string    Name of the goal
      coveredCompletelyCount            integer   Coverage complete count
      coveredCompletelyPercentage       number    Coverage complete percentage
      coveredPartiallyCount             integer   Coverage partial count
      coveredPartiallyPercentage        number    Coverage partial percentage
      handledCompletelyCount            integer   Handled complete count
      handledCompletelyPercentage       number    Handled complete percentage
      handledPartiallyCount             integer   Handled partial count
      handledPartiallyPercentage        number    Handled partial percentage
      unhandledCount                    integer   Unhandled count
      unhandledPercentage               number    Unhandled percentage
      uncoveredCount                    integer   Uncovered count
      uncoveredPercentage               number    Uncovered percentage
      justifiedCompletelyCount          integer   Covered completely (justified) count
      justifiedCompletelyPercentage     number    Covered completely (justified) percentage
      justifiedPartiallyCount           integer   Covered partially (justified) count
      justifiedPartiallyPercentage      number    Covered partially (justified) percentage
      totalCount                        integer   Total count
  }
  DownCastPropertyCoverage {
  DownCast property coverage information.
      coverageGoal                      string    Name of the goal
      coveredCount                      integer   Covered count
      coveredPercentage                 number    Covered percentage
      unreachableInfiniteCount          integer   Unreachable Infinite count
      unreachableInfinitePercentage     number    Unreachable Infinite percentage
```

```
            unreachableNCount                        integer   Unreachable N count
            unreachableNPercentage                   number    Unreachable N percentage
            unreachableInfiniteJustifiedCount        integer   Unreachable Infinite (justified) count
            unreachableInfiniteJustifiedPercentage   number    Unreachable Infinite (justified) percentage
            unreachableNJustifiedCount               integer   Unreachable N (justified) count
            unreachableNJustifiedPercentage          number    Unreachable N (justified) percentage
            unknownJustifiedCount                    integer   Unknown (justified) count
            unknownJustifiedPercentage               number    Unknown (justified) percentage
            unknownCount                             integer   Unknown count
            unknownPercentage                        number    Unknown percentage
            handledCount                             integer   Handled count
            handledPercentage                        number    Handled percentage
            inconsistentCount                        integer   Inconsistent count
            inconsistentPercentage                   number    Inconsistent percentage
            unreachableCount                         integer   Unreachable count
            unreachablePercentage                    number    Unreachable percentage
            totalCount                               integer   Total count
            comment                                  string    The comment
        }
        codeCoverageComment             string        The code coverage overview comment.
        robustnessCoverageComment       string        The robustness coverage overview comment.
    }
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

8.4 GET /ep/scopes/{scope-uid}/coverage-results-rbt

Get coverage results for a scope

Get the code coverage and robustness checks results for a scope. Goal type filters can be applied.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
goal-types	array of string ALLOWED: STM, D, C, MCDC, F, FC, SC, RO, CDC, DZ, CA	The goal type filter for code coverage/robustness check goals. Possible values for code coverage goals: STM(Statement), C(Condition), D(Decision/Branch), CDC(C/DC), MCDC(C/DC and MC/DC), F(Function), SC(Switch-Case), RO(Relational Operator), FC(Function Call). Possible values for robustness check goals are: DZ(Division by Zero), CA(Downcast). Can also specify multiple options. Can also be empty, in which case the results are shown for all goal types.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
  CDCCoverage {
```
CDC goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

```
  }
  CDCPropertyCoverage {
```
CDC property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage

unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

ConditionCoverage {
Condition goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

ConditionPropertyCoverage {
Condition property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

DecisionCoverage {
Decision goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage

coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

DecisionPropertyCoverage {
Decision property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

FunctionCoverage {
Function goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage

```
        justifiedCompletelyCount              integer   Covered completely (justified) count
        justifiedCompletelyPercentage         number    Covered completely (justified) percentage
        justifiedPartiallyCount               integer   Covered partially (justified) count
        justifiedPartiallyPercentage          number    Covered partially (justified) percentage
        totalCount                            integer   Total count
    }
    FunctionPropertyCoverage {
    Function property coverage information.

        coverageGoal                                       string    Name of the goal
        coveredCount                                       integer   Covered count
        coveredPercentage                                  number    Covered percentage
        unreachableInfiniteCount                           integer   Unreachable Infinite count
        unreachableInfinitePercentage                      number    Unreachable Infinite percentage
        unreachableNCount                                  integer   Unreachable N count
        unreachableNPercentage                             number    Unreachable N percentage
        unreachableInfiniteJustifiedCount                  integer   Unreachable Infinite (justified) count
        unreachableInfiniteJustifiedPercentage             number    Unreachable Infinite (justified) percentage
        unreachableNJustifiedCount                         integer   Unreachable N (justified) count
        unreachableNJustifiedPercentage                    number    Unreachable N (justified) percentage
        unknownJustifiedCount                              integer   Unknown (justified) count
        unknownJustifiedPercentage                         number    Unknown (justified) percentage
        unknownCount                                       integer   Unknown count
        unknownPercentage                                  number    Unknown percentage
        handledCount                                       integer   Handled count
        handledPercentage                                  number    Handled percentage
        inconsistentCount                                  integer   Inconsistent count
        inconsistentPercentage                             number    Inconsistent percentage
        unreachableCount                                   integer   Unreachable count
        unreachablePercentage                              number    Unreachable percentage
        totalCount                                         integer   Total count
        comment                                            string    The comment
    }
    FunctionCallCoverage {
    Function call goal coverage information.

        coverageGoal                          string    Name of the goal
        coveredCompletelyCount                integer   Coverage complete count
        coveredCompletelyPercentage           number    Coverage complete percentage
        coveredPartiallyCount                 integer   Coverage partial count
        coveredPartiallyPercentage            number    Coverage partial percentage
        handledCompletelyCount                integer   Handled complete count
        handledCompletelyPercentage           number    Handled complete percentage
        handledPartiallyCount                 integer   Handled partial count
        handledPartiallyPercentage            number    Handled partial percentage
        unhandledCount                        integer   Unhandled count
        unhandledPercentage                   number    Unhandled percentage
        uncoveredCount                        integer   Uncovered count
        uncoveredPercentage                   number    Uncovered percentage
        justifiedCompletelyCount              integer   Covered completely (justified) count
        justifiedCompletelyPercentage         number    Covered completely (justified) percentage
        justifiedPartiallyCount               integer   Covered partially (justified) count
        justifiedPartiallyPercentage          number    Covered partially (justified) percentage
        totalCount                            integer   Total count
    }
    FunctionCallPropertyCoverage {
    Function call property coverage information.

        coverageGoal                                       string    Name of the goal
        coveredCount                                       integer   Covered count
```

coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

MCDCCoverage {
MCDC goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

MCDCPropertyCoverage {
MCDC property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count

unknownJustifiedPercentage	number	Unknown (justified) percentage	
unknownCount	integer	Unknown count	
unknownPercentage	number	Unknown percentage	
handledCount	integer	Handled count	
handledPercentage	number	Handled percentage	
inconsistentCount	integer	Inconsistent count	
inconsistentPercentage	number	Inconsistent percentage	
unreachableCount	integer	Unreachable count	
unreachablePercentage	number	Unreachable percentage	
totalCount	integer	Total count	
comment	string	The comment	

}

RelationalOperatorCoverage {
Relational operation goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

RelationalOperatorPropertyCoverage {
Relational operation property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count

```
        comment                                    string    The comment
}
StatementCoverage {
Statement goal coverage information.

        coverageGoal                               string    Name of the goal
        coveredCompletelyCount                     integer   Coverage complete count
        coveredCompletelyPercentage                number    Coverage complete percentage
        coveredPartiallyCount                      integer   Coverage partial count
        coveredPartiallyPercentage                 number    Coverage partial percentage
        handledCompletelyCount                     integer   Handled complete count
        handledCompletelyPercentage                number    Handled complete percentage
        handledPartiallyCount                      integer   Handled partial count
        handledPartiallyPercentage                 number    Handled partial percentage
        unhandledCount                             integer   Unhandled count
        unhandledPercentage                        number    Unhandled percentage
        uncoveredCount                             integer   Uncovered count
        uncoveredPercentage                        number    Uncovered percentage
        justifiedCompletelyCount                   integer   Covered completely (justified) count
        justifiedCompletelyPercentage              number    Covered completely (justified) percentage
        justifiedPartiallyCount                    integer   Covered partially (justified) count
        justifiedPartiallyPercentage               number    Covered partially (justified) percentage
        totalCount                                 integer   Total count
}
StatementPropertyCoverage {
Statement property coverage information.

        coverageGoal                               string    Name of the goal
        coveredCount                               integer   Covered count
        coveredPercentage                          number    Covered percentage
        unreachableInfiniteCount                   integer   Unreachable Infinite count
        unreachableInfinitePercentage              number    Unreachable Infinite percentage
        unreachableNCount                          integer   Unreachable N count
        unreachableNPercentage                     number    Unreachable N percentage
        unreachableInfiniteJustifiedCount          integer   Unreachable Infinite (justified) count
        unreachableInfiniteJustifiedPercentage     number    Unreachable Infinite (justified) percentage
        unreachableNJustifiedCount                 integer   Unreachable N (justified) count
        unreachableNJustifiedPercentage            number    Unreachable N (justified) percentage
        unknownJustifiedCount                      integer   Unknown (justified) count
        unknownJustifiedPercentage                 number    Unknown (justified) percentage
        unknownCount                               integer   Unknown count
        unknownPercentage                          number    Unknown percentage
        handledCount                               integer   Handled count
        handledPercentage                          number    Handled percentage
        inconsistentCount                          integer   Inconsistent count
        inconsistentPercentage                     number    Inconsistent percentage
        unreachableCount                           integer   Unreachable count
        unreachablePercentage                      number    Unreachable percentage
        totalCount                                 integer   Total count
        comment                                    string    The comment
}
SwitchCaseCoverage {
Switch Case goal coverage information.

        coverageGoal                               string    Name of the goal
        coveredCompletelyCount                     integer   Coverage complete count
        coveredCompletelyPercentage                number    Coverage complete percentage
        coveredPartiallyCount                      integer   Coverage partial count
        coveredPartiallyPercentage                 number    Coverage partial percentage
        handledCompletelyCount                     integer   Handled complete count
```

handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count
justifiedPartiallyPercentage	number	Covered partially (justified) percentage
totalCount	integer	Total count

}

SwitchCasePropertyCoverage {
Switch Case property coverage information.

coverageGoal	string	Name of the goal
coveredCount	integer	Covered count
coveredPercentage	number	Covered percentage
unreachableInfiniteCount	integer	Unreachable Infinite count
unreachableInfinitePercentage	number	Unreachable Infinite percentage
unreachableNCount	integer	Unreachable N count
unreachableNPercentage	number	Unreachable N percentage
unreachableInfiniteJustifiedCount	integer	Unreachable Infinite (justified) count
unreachableInfiniteJustifiedPercentage	number	Unreachable Infinite (justified) percentage
unreachableNJustifiedCount	integer	Unreachable N (justified) count
unreachableNJustifiedPercentage	number	Unreachable N (justified) percentage
unknownJustifiedCount	integer	Unknown (justified) count
unknownJustifiedPercentage	number	Unknown (justified) percentage
unknownCount	integer	Unknown count
unknownPercentage	number	Unknown percentage
handledCount	integer	Handled count
handledPercentage	number	Handled percentage
inconsistentCount	integer	Inconsistent count
inconsistentPercentage	number	Inconsistent percentage
unreachableCount	integer	Unreachable count
unreachablePercentage	number	Unreachable percentage
totalCount	integer	Total count
comment	string	The comment

}

DivisionByZeroCoverage {
Division By Zero goal coverage information.

coverageGoal	string	Name of the goal
coveredCompletelyCount	integer	Coverage complete count
coveredCompletelyPercentage	number	Coverage complete percentage
coveredPartiallyCount	integer	Coverage partial count
coveredPartiallyPercentage	number	Coverage partial percentage
handledCompletelyCount	integer	Handled complete count
handledCompletelyPercentage	number	Handled complete percentage
handledPartiallyCount	integer	Handled partial count
handledPartiallyPercentage	number	Handled partial percentage
unhandledCount	integer	Unhandled count
unhandledPercentage	number	Unhandled percentage
uncoveredCount	integer	Uncovered count
uncoveredPercentage	number	Uncovered percentage
justifiedCompletelyCount	integer	Covered completely (justified) count
justifiedCompletelyPercentage	number	Covered completely (justified) percentage
justifiedPartiallyCount	integer	Covered partially (justified) count

```
        justifiedPartiallyPercentage          number    Covered partially (justified) percentage
        totalCount                            integer   Total count
    }
    DivisionByZeroPropertyCoverage {
    Division By Zero property coverage information.

        coverageGoal                                    string    Name of the goal
        coveredCount                                    integer   Covered count
        coveredPercentage                               number    Covered percentage
        unreachableInfiniteCount                        integer   Unreachable Infinite count
        unreachableInfinitePercentage                   number    Unreachable Infinite percentage
        unreachableNCount                               integer   Unreachable N count
        unreachableNPercentage                          number    Unreachable N percentage
        unreachableInfiniteJustifiedCount               integer   Unreachable Infinite (justified) count
        unreachableInfiniteJustifiedPercentage          number    Unreachable Infinite (justified) percentage
        unreachableNJustifiedCount                      integer   Unreachable N (justified) count
        unreachableNJustifiedPercentage                 number    Unreachable N (justified) percentage
        unknownJustifiedCount                           integer   Unknown (justified) count
        unknownJustifiedPercentage                      number    Unknown (justified) percentage
        unknownCount                                    integer   Unknown count
        unknownPercentage                               number    Unknown percentage
        handledCount                                    integer   Handled count
        handledPercentage                               number    Handled percentage
        inconsistentCount                               integer   Inconsistent count
        inconsistentPercentage                          number    Inconsistent percentage
        unreachableCount                                integer   Unreachable count
        unreachablePercentage                           number    Unreachable percentage
        totalCount                                      integer   Total count
        comment                                         string    The comment
    }
    DownCastCoverage {
    DownCast goal coverage information.

        coverageGoal                          string    Name of the goal
        coveredCompletelyCount                integer   Coverage complete count
        coveredCompletelyPercentage           number    Coverage complete percentage
        coveredPartiallyCount                 integer   Coverage partial count
        coveredPartiallyPercentage            number    Coverage partial percentage
        handledCompletelyCount                integer   Handled complete count
        handledCompletelyPercentage           number    Handled complete percentage
        handledPartiallyCount                 integer   Handled partial count
        handledPartiallyPercentage            number    Handled partial percentage
        unhandledCount                        integer   Unhandled count
        unhandledPercentage                   number    Unhandled percentage
        uncoveredCount                        integer   Uncovered count
        uncoveredPercentage                   number    Uncovered percentage
        justifiedCompletelyCount              integer   Covered completely (justified) count
        justifiedCompletelyPercentage         number    Covered completely (justified) percentage
        justifiedPartiallyCount               integer   Covered partially (justified) count
        justifiedPartiallyPercentage          number    Covered partially (justified) percentage
        totalCount                            integer   Total count
    }
    DownCastPropertyCoverage {
    DownCast property coverage information.

        coverageGoal                                    string    Name of the goal
        coveredCount                                    integer   Covered count
        coveredPercentage                               number    Covered percentage
        unreachableInfiniteCount                        integer   Unreachable Infinite count
        unreachableInfinitePercentage                   number    Unreachable Infinite percentage
```

```
            unreachableNCount                        integer   Unreachable N count
            unreachableNPercentage                   number    Unreachable N percentage
            unreachableInfiniteJustifiedCount        integer   Unreachable Infinite (justified) count
            unreachableInfiniteJustifiedPercentage   number    Unreachable Infinite (justified) percentage
            unreachableNJustifiedCount               integer   Unreachable N (justified) count
            unreachableNJustifiedPercentage          number    Unreachable N (justified) percentage
            unknownJustifiedCount                    integer   Unknown (justified) count
            unknownJustifiedPercentage               number    Unknown (justified) percentage
            unknownCount                             integer   Unknown count
            unknownPercentage                        number    Unknown percentage
            handledCount                             integer   Handled count
            handledPercentage                        number    Handled percentage
            inconsistentCount                        integer   Inconsistent count
            inconsistentPercentage                   number    Inconsistent percentage
            unreachableCount                         integer   Unreachable count
            unreachablePercentage                    number    Unreachable percentage
            totalCount                               integer   Total count
            comment                                  string    The comment
        }
        codeCoverageComment              string          The code coverage overview comment.
        robustnessCoverageComment        string          The robustness coverage overview comment.
    }
```

STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

```
string
```

8.5 POST /ep/coverage-goals/set-comment-rbt

Set goal comments

Set comments of code coverage and robustness check goals.

REQUEST

REQUEST BODY - application/json

```
[{
```
Array of object:

```
    pll*       string   The PLL of the code coverage or robustness goal for which to set the comment.
    comment*   string   The comment to set on the goal with the given PLL.
}]
```

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

8.6 POST /ep/coverage-goals/set-justified-rbt

Set goal justified status

Set justified status of code coverage and robustness check goals.

REQUEST

REQUEST BODY - application/json
```
{
    justified  boolean    The desired state of justify for the supplied goals.
    plls       [string]   The Plls of the goals
}
```

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

8.7 POST /ep/scopes/{scope-uid}/set-coverage-overview-comment-rbt

Set overview comments

Set the comments of code coverage overview sections.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope

REQUEST BODY - application/json
```
[{
```
Array of object:

```
    type*      enum    ALLOWED:CC_STAT, STM, D, C, MCDC, F, FC, SC, RO, RC_STAT, DZ, CA
```
The type of overview for which to set the comment. Possible values for code coverage goals: CC_STAT(Code

Coverage Statistics), STM(Statement), D(Decision/Branch), C(Condition), MCDC(C/DC and MC/DC), F(Function), FC(Function Call). SC(Switch-Case), RO(Relational Operator), Possible values for robustness check goals are: RC_STAT(Robustness Check Statistics) ,DZ(Division by Zero), CA(Downcast). Can also specify multiple options.

comment*   string   The comment to set for the overview type.

}]

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: Bad request

   RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

   RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

   RESPONSE MODEL - text/plain

string

9. COVERAGE GENERATION

For a C-Code function, create stimuli vectors which cover the code function, or mark coverage properties that are unreachable.

9.1 POST /ep/coverage-generation

Execute coverage generation

Long running task Using the provided configuration, execute the coverage generation / stimuli vector generation.

REQUEST

REQUEST BODY - application/json

{

`isSubscopesGoalsConsidered`	boolean	Whether or not goals from sub scopes should be considered. Default is 'true'

`targetDefinitions [{`

Array of object: The target definitions to use for this run.
If no target definitions are provided, and no PLL is given, the default target definitions are used

`label*`	string	The label of the target definition.

`}]`

`folderName`	string	Name of the folder to store Stimuli Vectors in. If the specified folder doesn't exists, it will be created with the given name. If no folder is specified the default folder 'Default Stimuli Vectors' will be used.
`checkUnreachableProperties`	boolean	Whether or not unreachable properties should be (re-)checked. Default is 'false'
`pllString`	string	PLL String for specifc goals to be reached. Default is '::' which matches all goals. Use e.g. 'STM;D;CDC' to find stimuli vectors for statement, decision, and condition/decision coverage. Individual PLLs can be addressed by their specific label (e.g. 'D:4:1'). Multiple PLLs can be concatenated using semicolon, e.g. 'D:4:1;C:2'. See the user guide for more information about the property location labels. If this is null or empty, only the selected target definitions will be used. If in that case, the target definitions are be empty, the default target definitions are used.

`engineSettings {`

The engine settings to use for this run.

`timeoutSeconds`	integer	Global timeout (seconds) for the execution Default is '-1' (unlimited)
`handlingRateThreshold`	integer	After each scope is analyzed, the 'handled rate' of the entry scope (potentially including goals from subscopes) is checked against this threshold and the stimuli vector generation is stopped when it is reached. Allowed range are integers from [1, 100] (percent of handled goals). Default: 100
`analyseSubScopesHierarchically`	boolean	Enables / disables recursive analysis of subscopes. Default is 'true'

`engineAtg {`

The ATG engine.
Note: If neither ATG, nor CV engine is provided, both, ATG and CV are used together, using their default settings!

`name`	string	The name of the engine (heuristic). Currently, only 'ATG' is allowed. Default is 'ATG'
`searchDepthSteps`	integer	The search depth (number of SUT iterations) Default is '20'
`executionMode`	enum	**ALLOWED:** TOP_DOWN, BOTTOM_UP The search direction, bottom up or top down Default is 'TOP_DOWN'
`mutateExistingVectors`	boolean	Defines whether or not the Mutation Based ATG engine shall be used. MATG requires existing vectors to produce new results. Default is 'false'
`timeoutSecondsPerSubsystem`	integer	Timeout (seconds) per scope Default is '300'

`}`

`engineCv {`

The CV engine.
Note: If neither ATG, nor CV engine is provided, both, ATG and CV are used together, using their default settings!

`name`	string	The name of the engine (heuristic). Currently, only 'CV' is allowed. Default is 'CV'
`searchDepthSteps`	integer	The search depth (number of SUT iterations) Default is '10'
`timeoutSecondsPerSubsystem`	integer	Timeout (seconds) per scope Default is '-1' (unlimited)
`timeoutSecondsPerProperty`	integer	Timeout (seconds) per coverage property Default is '60'

memoryLimitMb   `integer`   The maximum amount of system memory to use (MB)
Default is '-1' (unlimited)

loopUnroll   `integer`   The number of internal loop unwindings for potentially unbounded loops within each SUT iteration.
Default is '50'

coreEngines [{

Array of object: The core engines to use
Note: If no core engine is provided, all core engines are used by default!

    name*   `enum`   **ALLOWED:** SMIBMC, VIS, AUTOFXP, CBMC, ISAT

       The name of the core engine.If no core engine is provided, all core engines are used by default!

}]

assumptionCheckEnabled   `boolean`   Whether or not core engines are allowed to explicitly checkthe satisfiability of the selected assumptions.
Default is 'true'

searchFocus   `enum`   **ALLOWED:** BALANCED, REACHABLE, UNREACHABLE

     The search focus of the core engines.
Default is 'BALANCED'

parallelExecutionMode   `enum`   **ALLOWED:** BALANCED, ENGINES, GOALS

     The mode used for the parallel engine execution.If maximum number of threads used is 1, the value of this parameter is not used (instead the default value BALANCED will be used).

maximumNumberOfThreads   `integer`   The maximum number of threads available for parallel engine execution for core engines. Valid values are between 1 and available number of cores. It is possible to set this parameter to a value of -1, which will compute the number of threads automatically as half of the available number of cores.
Default is '1'

}

allowDenormalizedFloats   `boolean`   Whether or not the engine may produce denormalized floats.
Default is 'true'

}

scopeUid*   `string`   The UID of the entry scope to use for this run.

assumptions [{

Array of object: The environmental assumptions to use for this run.

    id*   `string`   The Assumption UID.

}]

drivers [{

Array of object: The drivers to use for this run.

    id*   `string`   The Driver source UID.

}]

initializationVectorUID   `string`   The UID of the RequirementBasedTestCase or B2BStimuliVector which shall be used to initialize the engine

}

RESPONSE

STATUS CODE - 202: Accepted

RESPONSE MODEL - application/json

{

    jobID   `string`   **READ-ONLY**

         The ID of a job.

}

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

10. DOMAIN CHECKS

Handle domain checks.

10.1 POST /ep/domain-checks-ranges

Create domain checks ranges

Create the domain checks ranges for a scope and a list of signals. If the list of signals is not given, ranges for all signals of the scope will be created by default. The ranges can be created with some convenience functions applied (partitioned by a percent, with boundaries checks included or with invalid ranges checks included). Please note ALL domain checks ranges created via this service will overwrite any existing ranges for the given list of signals or for all signals (if no signal is specified).

REQUEST

REQUEST BODY - application/json

```
{
```

scopeUid*	string	The scope for which to create the domain checks ranges.
signalUids	[string]	The list of signals for which to create the domain checks ranges. If no signal is provided, ranges for all signals from the scope are created by default.
applyBoundaryChecks	boolean	Used for applying the boundary checks when creating the range.Can only be used for applying boundary checks on a defined range, so it must be used only together with one of the other options (either apply invalid ranges checks or partition ranges). Default is 'false'
applyInvalidRangesChecks	boolean	Used for applying the invalid ranges checks when creating the range. Default is 'false'
percentage	integer	Percentage used for partioning the range interval when creating the range. If no value is provided the domain check ranges will not be partitioned.

```
}
```

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - text/plain

string

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

10.2 POST /ep/domain-checks-export

Export domain check ranges

Export domain check ranges of the given scopeUid.

REQUEST

REQUEST BODY - application/json

```
{
```

scopeUid*	string	The scopeUid for which to export/import domain check ranges.
filePath*	string	The file path used for exporting/importing domain check ranges. The directory path specified in the filePath must already exist.

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                  The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

10.3 GET /ep/scopes/{scope-uid}/domain-check-details

Get domain check details

Get domain check details for a scope. Some filters can be applied.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope, for which to retrieve the domain check details.

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
useCase	enum ALLOWED: B2B, RBT	The use case for which to retrieve the domain check details. Possible values: B2B, RBT. If not provided, details are shown for B2B.
signalNames	array of string	The names of the signals for which to retrieve the domain check details. If not provided, the details for all signals are shown.
signalKinds	array of string ALLOWED: INPUT, OUTPUT, LOCAL, PARAMETER	The kinds of the signals for which to retrieve the domain check details. Possible values: INPUT, OUTPUT, LOCAL, PARAMETER. If not provided, the details will be shown for all kinds.
goalStates	array of string ALLOWED: COVERED, UNREACHABLE, UNKNOWN, ERROR, NOT_DEFINED	The status filter for domain check details. Possible values: COVERED, UNKNOWN, UNREACHABLE, ERROR. If not provided, the details will be shown for all status.

NAME	TYPE	DESCRIPTION
goalTypes	array of string **ALLOWED:** VALID, INVALID	The goal type filter for domain check details. Possible values: VALID, INVALID. If not provided, the details will be shown for all types.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
    name            string      READ-ONLY
                                The name of the signal for which domain check goals where created.
    kind            enum        READ-ONLY
                                ALLOWED:INPUT, OUTPUT, LOCAL, PARAMETER
                                The kind of the signal for which domain check goals where created.
    pll             string      PLL string of the domain check goal.
    range           string      The range of the domain check goal.
    goalType        enum        ALLOWED:VALID, INVALID
                                The type of the domain check goal.
    goalStatus      enum        ALLOWED:COVERED, UNREACHABLE, UNKNOWN, ERROR, NOT_DEFINED
                                The status of the domain check goal.
    comment         string      The comment of the domain check goal.
    coveringVectors [string]    List of string vector names that cover the property.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

10.4 GET /ep/scopes/{scope-uid}/domain-checks-results

Get domain checks results

Get the domain checks results for a scope. Using the use case option will display the results for SVs and RBTest Cases if B2B is used or only for RBTest Cases if RBT option is used. Also, the results can be requested either only for the invalid goal types and for the valid ones, or for all goal types.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope for which to get the domain checks results.

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
useCase	enum **ALLOWED:** B2B, RBT	The use case for which to retrieve the domain check results. Possible values: B2B, RBT.Can be empty, in which case the results are shown for B2B use case.
goalTypes	array of string **ALLOWED:** VALID, INVALID	The goal type for which to retrieve the domain check results. Possible values: VALID, INVALID.Can be empty, in which case the results are shown for all goal types.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

{

totalCountValid	string	**READ-ONLY**
		The total number of valid range goals.
coveredCountValid	string	**READ-ONLY**
		The number of covered valid range goals.
unreachableCountValid	string	**READ-ONLY**
		The number of unreachable valid range goals.
errorCountValid	string	**READ-ONLY**
		The number of errorneous valid range goals.
handledCountValid	string	**READ-ONLY**
		The number of handled valid range goals.
unhandledCountValid	string	**READ-ONLY**
		The number of unhandled valid range goals.
coveredPercValid	string	**READ-ONLY**
		The covered percentage for valid range goals.
unreachablePercValid	string	**READ-ONLY**
		The unreachable percentage for valid range goals.
errorPercValid	string	**READ-ONLY**
		The error percentage for valid range goals.
handledPercValid	string	**READ-ONLY**
		The handled percentage for valid range goals.
unhandledPercValid	string	**READ-ONLY**
		The unhandled percentage for valid range goals.
totalCountInvalid	string	**READ-ONLY**
		The total number of invalid range goals.
coveredCountInvalid	string	**READ-ONLY**
		The number of covered invalid range goals.
unreachableCountInvalid	string	**READ-ONLY**
		The number of unreachable invalid range goals.
errorCountInvalid	string	**READ-ONLY**
		The number of errorneous invalid range goals.
handledCountInvalid	string	**READ-ONLY**
		The number of handled invalid range goals.
unhandledCountInvalid	string	**READ-ONLY**
		The number of unhandled invalid range goals.
coveredPercInvalid	string	**READ-ONLY**
		The covered percentage for invalid range goals.
unreachablePercInvalid	string	**READ-ONLY**
		The unreachable percentage for invalid range goals.
errorPercInvalid	string	**READ-ONLY**
		The error percentage for invalid range goals.
handledPercInvalid	string	**READ-ONLY**
		The handled percentage for invalid range goals.
unhandledPercInvalid	string	**READ-ONLY**
		The unhandled percentage for invalid range goals.

}

STATUS CODE - 400: Bad Request

   RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

   RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

   RESPONSE MODEL - text/plain

string

10.5 POST /ep/domain-checks

Import domain check ranges

Import domain check ranges on the given scopeUid.

REQUEST

REQUEST BODY - application/json

```
{
    scopeUid*  string   The scopeUid for which to export/import domain check ranges.
    filePath*  string   The file path used for exporting/importing domain check ranges. The directory path specified in the filePath must
                        already exist.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

   RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad request

   RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

   RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

   RESPONSE MODEL - text/plain

string

10.6 POST /ep/domain-check-comments

Set domain check comments

Set the comments for the domain check ranges specified by the PLL of their corresponding domain check goal. To remove a comment for a given domain check goal, provide an empty string as a comment.

REQUEST

REQUEST BODY - application/json

[{
Array of object:

    pll*   string   The PLL of the domain check goal for which to set the comment.

    comment* string   The comment to set on the domain check goal with the given PLL.

}]

RESPONSE

STATUS CODE - 200: OK

    RESPONSE MODEL - text/plain

string

STATUS CODE - 400: Bad Request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

11. EXECUTION CONFIGS

11.1 GET /ep/execution-configs

Get all execution configs

Get all execution configs available in the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
    execConfigNames  [string]  List of the available execution kinds
}
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

```
string
```

12. EXECUTION RECORDS

Handle execution records.

12.1 GET /ep/execution-records/{execution-record-uid}

Get an execution record

Get the requested execution record by UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*execution-record-uid	string	Execution record UID

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
executionConfig*	string	The execution config name
name*	string	The execution record name
status	enum	**ALLOWED:**OK, WARNING, ERROR
		The status of execution record.Possible options: OK, WARNING, or ERROR.
folderName*	string	The folder name on which this execution record can be found.
length*	integer	The length of execution record source
scopeName*	string	The scope name of execution record
sourceName*	string	The name of execution record source

```
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

```
string
```

12.2 DELETE /ep/execution-records/{execution-record-uid}

Delete an execution record

Deletes the specified execution record.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
*execution-record-uid	string	The UID of the execution record to be deleted.

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 406: Not acceptable

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

12.3 POST /ep/execution-records-export

Export execution records

LONG RUNNING TASK Export single or multiple execution record(s) by providing the list of the execution records which will be exported, the export directory, the export format and a list of additional options for export.

REQUEST

REQUEST BODY - application/json

```
{
    UIDs*              [string]   List with the UIDs of the elements which will be exported
    exportDirectory*   string     Directory where to export the elements
    exportFormat       enum       ALLOWED: MDF, EXCEL
                                  The format of the exported execution records. (Default value: "MDF").
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
  jobID  string  READ-ONLY
                  The ID of a job.
}
```

STATUS CODE - 400: Bad Request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

12.4 GET /ep/execution-records

Get all execution records

Get all execution records available in the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

    RESPONSE MODEL - application/json

```
[{
Array of object:
    uid                    string      READ-ONLY
                                       The unique identifier (UID) of this object.
    executionConfig*       string      The execution config name
    name*                  string      The execution record name
    status                 enum        ALLOWED:OK, WARNING, ERROR
                                       The status of execution record.Possible options: OK, WARNING, or ERROR.
    folderName*            string      The folder name on which this execution record can be found.
    length*                integer     The length of execution record source
    scopeName*             string      The scope name of execution record
    sourceName*            string      The name of execution record source
}]
```

STATUS CODE - 500: Internal server error.

    RESPONSE MODEL - text/plain

string

12.5 POST /ep/execution-records

Import execution records

Import multiple execution records by providing the path for each file.

REQUEST

REQUEST BODY - application/json

{

paths*	[string]	The path to all execution record files you'd like to import. Supported formats are MDF, MF4 and CSV.
kind*	string	The simulation kind that was used for creating the execution records from the given files. Possible values: TL MIL, SL MIL, SIL, PIL or any external simulation kind. If the user defined folder option is not specified, the simulation kind will define the default folder where the execution records will be imported.
folderName	string	User defined execution records folder name. <b color="red">If used, folderUID can not be used at the same time.
folderUID	string	Existing user defined folder uid to import records. <b color="red">If used, folderName can not be used at the same time.
referenceExternalFile	boolean	Relevant only for MF4 and CSV import format. Whether to only reference the external file rather than importing the execution record in the profile. This would be recommended for very big execution records.Default is 'false'.
csvDelimiter	enum	**ALLOWED:**SEMICOLON, COMMA, COLON, PIPE
		Relevant only for CSV import format. It can have one of the following values: "SEMICOLON", "COMMA", "COLON", "PIPE". Default value is "SEMICOLON".

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

{

jobID	string	READ-ONLY
		The ID of a job.

}

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

12.6 GET /ep/folders/{folder-uid}/execution-records

Get all execution records for a folder

Get all the execution records for a given folder UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The folder UID from which to retrieve all execution records.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
executionConfig*	string	The execution config name
name*	string	The execution record name
status	enum	**ALLOWED:**OK, WARNING, ERROR
		The status of execution record.Possible options: OK, WARNING, or ERROR.
folderName*	string	The folder name on which this execution record can be found.
length*	integer	The length of execution record source
scopeName*	string	The scope name of execution record
sourceName*	string	The name of execution record source

```
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

12.7 PUT /ep/folders/{folder-uid}/execution-records

Move a list of execution records to a folder

Moves the list of execution records to the requested user-defined execution record folder.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The UID of user-defined execution record folder.

REQUEST BODY - application/json

```
{
   UIDs* [string]  UIDs of execution records to be moved.
}
```

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
executionConfig*	string	The execution config name
name*	string	The execution record name

```
      status              enum       ALLOWED:OK, WARNING, ERROR
                                     The status of execution record.Possible options: OK, WARNING, or ERROR.
      folderName*         string     The folder name on which this execution record can be found.
      length*             integer    The length of execution record source
      scopeName*          string     The scope name of execution record
      sourceName*         string     The name of execution record source
  }]
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

12.8 GET /ep/scopes/{scope-uid}/execution-records

Get all execution records for a scope

Get all the execution records for the given scope UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The scope UID from which to retrieve all execution records.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
Array of object:

      uid                 string     READ-ONLY
                                     The unique identifier (UID) of this object.
      executionConfig*    string     The execution config name
      name*               string     The execution record name
      status              enum       ALLOWED:OK, WARNING, ERROR
                                     The status of execution record.Possible options: OK, WARNING, or ERROR.
      folderName*         string     The folder name on which this execution record can be found.
      length*             integer    The length of execution record source
      scopeName*          string     The scope name of execution record
      sourceName*         string     The name of execution record source
  }]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

   RESPONSE MODEL - text/plain

```
string
```

13. FOLDERS

Handle folders.

13.1 GET /ep/folders

Get a list of folders
Get a list of folders by name and/or kind.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
name	string	Enter the name of the folder you would like to search for. If null, then all folders will be returned.
kind	enum ALLOWED: RB_TEST_CASE, EXECUTION_RECORD, STIMULI_VECTOR	Enter the folder kind. If null, then all folder kinds will be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY** The unique identifier (UID) of this object.
name*	string	The name of the folder.
kind*	string	The folder kind.
isDefault*	boolean	Set to 'true' if it is a default folder.

```
}]
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

13.2 POST /ep/folders

Create a folder
Add a folder by providing a folder kind and optionally a folder name. Note that a new UID will be assigned.

REQUEST

REQUEST BODY - application/json

```
{
  folderKind*   string   The folder kind. Possible: "RB_TEST_CASE", "EXECUTION_RECORD", "STIMULI_VECTOR"
  folderName    string   The folder name. This parameter is optional
}
```

RESPONSE

STATUS CODE - 201: Created

RESPONSE MODEL - application/json

```
{
  uid          string    READ-ONLY
                         The unique identifier (UID) of this object.
  name*        string    The name of the folder.
  kind*        string    The folder kind.
  isDefault*   boolean   Set to 'true' if it is a default folder.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

13.3 DELETE /ep/folders/{folder-uid}

Delete a folder

Delete a folder by providing its UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	Enter the UID of the folder you would like to delete.

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

14. FORMAL SPECIFICATION REPORTS

Formal Specification Reports

14.1 GET /ep/formal-specification-reports

Get all reports

Get all formal specification reports from the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
reportName	string	Name of the report.
reportType	string	Type of the report.

```
}]
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

```
string
```

14.2 POST /ep/formal-specification-reports

Create a formal specification report

Create a formal specification report on a list of formal specification UID's.

REQUEST

REQUEST BODY - application/json

```
{
   UIDs*  [string]   List with unique identifiers of the objects.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
   jobID  string   READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

15. FORMAL SPECIFICATIONS

Handle Formal Specifications.

15.1 GET /ep/environmental-assumptions

Get all environmental assumptions

Get all environmental assumptions present on active profile, or from the specified scope-uid.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
scope-uid	string	The UID of scope from which to get the environmental assumptions.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
name*	string	The name of the environmental assumption.
description*	string	The description of the environmental assumption.
scopeUID*	string	The unique identifier (UID) of the scope this object belongs to.
draft	string	States whether or not the FormalRequirement is in Draft-Mode.
errors	[string]	List of errors

```
}]
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

15.2 GET /ep/formal-requirements

Get all formal requirements

Get all formal requirements from the profile, or from the specified scope-uid.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
scope-uid	string	The UID of scope from which to get the formal requirements.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
name*	string	The name of the formal requirement.
description*	string	The description of the formal requirement.
scopeUID*	string	The unique identifier (UID) of the scope this object belongs to.
draft	string	States whether or not the FormalRequirement is in Draft-Mode.
errors	[string]	List of errors

```
}]
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

15.3 GET /ep/formal-requirements/{formal-requirement-uid}/environmental-assumptions

Get all environmental assumptions from a formal requirement

Use this command to retrieve the environmental assumptions from a formal requirement.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*formal-requirement-uid	string	The uid of the formal requirement for which all environmental assumptions should be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
name*	string	The name of the environmental assumption.
description*	string	The description of the environmental assumption.
scopeUID*	string	The unique identifier (UID) of the scope this object belongs to.
draft	string	States whether or not the FormalRequirement is in Draft-Mode.
errors	[string]	List of errors

```
}]
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

15.4 POST /ep/specifications-export

Export a SPEC file

Long running task Exports the given formal specifications belonging to the same scope to the specified SPEC file.

REQUEST

REQUEST BODY - application/json

```
{
    specFile*                      string        The file name of the target SPEC file. Should have .spec extension.
    archUid                        string        The architecture UID to define the interface names. If specified, the SPEC file will use
                                                 the interface and expression names based on the interfaces and signals defined in the
                                                 given architecture.If not specified, the first imported architecture is used for the name
                                                 space.
    formalSpecificationUids* [string]    A list of uids of formal requirements or environmental assumptions to export. Note:
                                                 The formal requirements / environmental assumptions must reside on the same scope!
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

15.5 POST /ep/specifications-import

Import a SPEC file

Imports formal specifications from the specified SPEC file.
Long running task Specify artifact existing policy, one of EXTEND_NAME, OVERWRITE, or SKIP. By default it will be used 'EXTEND_NAME'.

REQUEST

REQUEST BODY - application/json

```
{
    specPath*      string    The path of the given SPEC file. Should have .spec extension.
    scopeId        string    The scopeId to use, when scope definition in SPEC file is invalid. This can happed for two reasons, the first is
                             that in the SPEC file, the component(scope) name is invalid and can not be found in the opened profile, or when
                             the given component(scope) name in the SPEC file is not unique within current profile.e.g. a TL architecture
                             scope name is unique to the one in the generated CCode.
    isDraft        boolean   The draft status setting for the formal requirements that will be imported. By default its value is false.
    optionParam    enum      ALLOWED:EXTEND_NAME, OVERWRITE, SKIP
                             The options of importing a SPEC file, when the artifacts already exists. If no value is provided, 'EXTEND_NAME'
                             is used.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

16. FORMAL TEST EXECUTION

Execute Formal Test.

16.1 POST /ep/execute-formal-test

Execute formal tests

Long running task Execute formal test for all not yet existing or outdated results.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

17. FORMAL TEST REPORTS

Create and get formal test reports.

17.1 POST /ep/formal-requirements/{formal-requirement-uid}/formal-test-reports

Create a report on a formal requirement

Create a formal test report on a formal requirement for the given execution configurations.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*formal-requirement-uid	string	The formal requirement UID for which the formal test report is created.

REQUEST BODY - application/json

```
{
    executionConfigNames*  [string]   The comma separated list of execution configurations for which the Formal Test report shall
                                      be created. It can include values like "TL MIL", "SIL" etc. Please keep the specified format
                                      when you want to introduce a value.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                   The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

17.2 POST /ep/formal-requirements/formal-test-reports

Create a report on a list of formal requirements

Create a formal test report on on a list of formal requirements for the given execution configurations.

REQUEST

REQUEST BODY - application/json

```
{
    UIDs*                        [string]          List with unique identifiers of the objects.
    executionConfigInfo {
        executionConfigNames*  [string]  The comma separated list of execution configurations for which the Formal Test report
                                         shall be created. It can include values like "TL MIL", "SIL" etc. Please keep the specified
                                         format when you want to introduce a value.
    }
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

17.3 POST /ep/requirements-sources/{requirements-source-uid}/formal-test-reports

Create a report on a requirements source

Create a formal test report on a requirements source for the given execution configurations.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*requirements-source-uid	string	The requirements source UID for which the formal test report is created.

REQUEST BODY - application/json

```
{
    executionConfigNames*  [string]   The comma separated list of execution configurations for which the Formal Test report shall
                                       be created. It can include values like "TL MIL", "SIL" etc. Please keep the specified format
                                       when you want to introduce a value.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                        The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

17.4 POST /ep/requirements-sources/formal-test-reports

Create a report on a list of requirements sources

Create a formal test report on on a list of requirements sources for the given execution configurations.

REQUEST

REQUEST BODY - application/json
```
{
    UIDs*                        [string]                    List with unique identifiers of the objects.
    executionConfigInfo {
        executionConfigNames*  [string]   The comma separated list of execution configurations for which the Formal Test report
                                          shall be created. It can include values like "TL MIL", "SIL" etc. Please keep the specified
                                          format when you want to introduce a value.

    }
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                        The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

17.5 POST /ep/scopes/{scope-uid}/formal-test-reports

Create a report on a scope

Create a formal test report on a scope for the given execution configurations.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The scope UID for which the formal test report is created.

REQUEST BODY - application/json

```
{
    executionConfigNames*  [string]   The comma separated list of execution configurations for which the Formal Test report shall
                                        be created. It can include values like "TL MIL", "SIL" etc. Please keep the specified format
                                        when you want to introduce a value.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                        The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

17.6 POST /ep/scopes/formal-test-reports

Create a report for a list of scopes

Create a formal test report on a list of scopes for the given execution configurations.

REQUEST

REQUEST BODY - application/json

```
{
    UIDs*                        [string]              List with unique identifiers of the objects.
    executionConfigInfo {
        executionConfigNames*  [string]   The comma separated list of execution configurations for which the Formal Test report
                                          shall be created. It can include values like "TL MIL", "SIL" etc. Please keep the specified
                                          format when you want to introduce a value.
    }
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

17.7 GET /ep/formal-test-reports

Get all reports

Retrieve all the Formal Test reports from the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
reportName	string	Name of the report.
reportType	string	Type of the report.

```
}]
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

18. FORMAL VERIFICATION REPORTS

Create Formal Verification reports.

18.1 POST /ep/scopes/{scope-uid}/formal-verification-reports

Create a formal verification report on given scopeUid.

Create a formal verification report on given scope-uid which is found on this active profile. If multiple architectures for report creation are preset, the architecture-uid query parameter must be specified.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope.

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
architecture-uid	string	The UID of the architecture on which the report to be created. If there are multiple valid architectures, but no architecture is specified, the C-Code architecture is used as a default.

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

18.2 GET /ep/formal-verification-reports

Get all reports

Get all formal verification reports from the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
reportName	string	Name of the report.
reportType	string	Type of the report.

```
}]
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

```
string
```

19. INPUT RESTRICTIONS

19.1 POST `/ep/input-restrictions-export`

Export input restrictions

Export input restrictions to a file

REQUEST

REQUEST BODY - application/json
```
{
    filePath*  string   The file containing input restrictions.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json
```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

19.2 POST `/ep/input-restrictions-import`

Import input restrictions

Import input restrictions from a file

REQUEST

REQUEST BODY - application/json
```
{
    filePath*  string   The file containing input restrictions.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the <a

href='#get-/ep/progress'>Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                        The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

20. INTERFACE REPORTS

Handle interface reports.

20.1 POST /ep/scopes/{scope-uid}/interface-reports

Create a report on a scope

Create an interface report on given scope. The interface report will use the interface of the architecture of the provided scope.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The scope on which the interface report should be created. The interface report will use the interface of the architecture of the provide scope.

RESPONSE

STATUS CODE - 201: Created

RESPONSE MODEL - application/json

```
{
   uid              string   READ-ONLY
                             The unique identifier (UID) of this object.
   reportName   string   Name of the report.
   reportType   string   Type of the report.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

20.2 GET /ep/interface-reports

Get all reports

Retrieve all interface reports from the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
reportName	string	Name of the report.
reportType	string	Type of the report.

```
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

```
[{
```
Array of object:

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
reportName	string	Name of the report.

21. MATLAB SCRIPT EXECUTION

Execute a MATLAB script.

21.1 POST /ep/execute-long-matlab-script

Execute long-running MATLAB script

Execute a long-running MATLAB script using the given parameters. Should be used when the time it takes for the script to end is longer than the request timeout of your REST client. Otherwise, for ease of use, the Execute a short-running MATLAB script method should be utilized.

REQUEST

REQUEST BODY - application/json
{

scriptName*   string   MATLAB script name.

outArgs*   integer   Number of output arguments to return. Exception will be thrown
                                                         if the given m-script returns less arguments.

inArgs* [{

Array of object: Parameters of the MATLAB script. The order in the list can be important, dependent of the executed m-script. The parameters will
be received in the MATLAB script as follows: primitive types will be converted into their MATLAB equivalents, JSON arrays will be converted into
cell arrays, JSON objects will be converted into MATLAB structures.

}]

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

{

jobID   string   **READ-ONLY**

The ID of a job.

}

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

21.2 POST /ep/execute-short-matlab-script

Execute short-running MATLAB script

Execute a short-running MATLAB script using the given parameters. If the time it takes for the script to end is longer than the request timeout of your REST client, this request may fail. In this scenario the Execute a long-running MATLAB script method should be used.

REQUEST

REQUEST BODY - application/json
{

```
scriptName*                    string              MATLAB script name.
outArgs*                       integer             Number of output arguments to return. Exception will be thrown
                                                   if the given m-script returns less arguments.

inArgs* [{
Array of object:  Parameters of the MATLAB script. The order in the list can be important, dependent of the executed m-script. The parameters will
be received in the MATLAB script as follows: primitive types will be converted into their MATLAB equivalents, JSON arrays will be converted into
cell arrays, JSON objects will be converted into MATLAB structures.

}]
}
```

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
   outArgs [{
Array of object:  Output objects of the MATLAB script. The objects returned from the script must be primitive types, cell arrays or structures.

   }]
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

22. MESSAGES

Handle messages and message markers.

22.1 POST /ep/message-markers

Create a message marker

Use this command to create a new message marker at the current time. The response will contain the created message marker time stamp as java Timestamp

REQUEST

No request parameters

RESPONSE

STATUS CODE - 201: Created

RESPONSE MODEL - application/json

```
{
    date   string   READ-ONLY
                     The date when the marker was set.
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

22.2 GET /ep/message-markers/{marker-date}/messages

Get a list of messages from a message marker

Search for messages created after a given message marker.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*marker-date	string	The message marker after which the messages should be queried from the Database.

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
search-string	string	Enter the beginning of the messages you would like to search for. If null, then all messages will be returned.
severity	enum ALLOWED: INFO, WARNING, ERROR, CRITICAL	Choose any severity you would like to search for. If null, then all severities will be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
date	string	READ-ONLY
		The creation-date of the message.
message*	string	The message itself.
hint	string	An additional hint.
severity*	enum	ALLOWED:INFO, WARNING, ERROR, CRITICAL
		The severity of the message.

```
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

22.3 GET /ep/messages

Get a list of messages

Search for past messages up until a certain amount you can set yourself.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
search-string	string	Enter the beginning of the messages you would like to search for. If null, then all messages will be returned. Furthermore, you may use the following wildcards: '*' for any string, '?' for any single character.
severity	enum ALLOWED: INFO, WARNING, ERROR, CRITICAL	Choose any severity you would like to search for. If null, then all severities will be returned.
max-messages	int32	The maximum number of messages returned. Cannot be > 1000. If > 1000, negative, or null, at most 1000 messages will be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
date	string	READ-ONLY
		The creation-date of the message.
message*	string	The message itself.
hint	string	An additional hint.
severity*	enum	ALLOWED:INFO, WARNING, ERROR, CRITICAL

```
  }]
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

22.4 POST /ep/messages

Create a message

Add a message by providing a Message. Note that a new UID will be assigned.

REQUEST

REQUEST BODY - application/json
```
{
    uid         string    READ-ONLY
                          The unique identifier (UID) of this object.
    date        string    READ-ONLY
                          The creation-date of the message.
    message*    string    The message itself.
    hint        string    An additional hint.
    severity*   enum      ALLOWED:INFO, WARNING, ERROR, CRITICAL
                          The severity of the message.
}
```

RESPONSE

STATUS CODE - 201: Created

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

22.5 DELETE /ep/messages

Delete a list of messages

Search for past messages up until a certain amount you can set yourself and delete them.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
search-string	string	Enter the beginning of the messages you would like to search for. If null, then all messages will be deleted.

NAME	TYPE	DESCRIPTION
severity	enum **ALLOWED:** INFO, WARNING, ERROR, CRITICAL	Choose any severity you would like to search for. If null, then all severities will be deleted.
max-messages	int32	The max number of messages deleted. If negative or null, all messages will be deleted.

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

22.6 GET /ep/messages/{message-uid}

Get a message

Get the message with the provided UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*message-uid	string	The UID of the message to be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
    uid         string  READ-ONLY
                        The unique identifier (UID) of this object.
    date        string  READ-ONLY
                        The creation-date of the message.
    message*    string  The message itself.
    hint        string  An additional hint.
    severity*   enum    ALLOWED: INFO, WARNING, ERROR, CRITICAL
                        The severity of the message.
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

22.7 DELETE /ep/messages/{message-uid}

Delete a message

Delete a message by providing its UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*message-uid	string	Enter the UID of the message you would like to delete.

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

22.8 POST /ep/messages/message-report

Export messages

Export all messages to the specified report file (in HTML format).
If a Message Marker is provided, all messages starting from the marker will be exported.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*file-name	string	The path and file name of the message report file to create. Note: An existing file will be overwritten!
marker-date	string	If specified, only messages that were posted after this Message Marker was created will be exported.

RESPONSE

STATUS CODE - 201: Exported Succesfully. Returns exported report location.

STATUS CODE - 400: Bad Request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

23. MODEL COVERAGE REPORTS

Creates RBT or B2B model coverage reports.

23.1 POST /ep/folders/{folder-uid}/model-coverage-reports

Create a report for a folder

Long running task Create RBT or B2B model coverage report for given folder.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The folder UID for which to create the model coverage report.

REQUEST BODY - application/json

```
{
    type*                 enum      ALLOWED: RBT, B2B
                                    Specifies the testing use-case for which the model coverage report should be created.<br>Must
                                    be 'B2B' or 'RBT' for Back-to-back testing or Requirement based test, respectively.
    simulationKind*       string    Specifies the simulation mode: 'SL MIL' or 'TL MIL'.
    useShortCircuitLogic  boolean   Specifies if short circuit logic should be used for Simulink blocks. <br>The paramter is optional.
                                    If not provided, the current setting from EP is used.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                   The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

23.2 POST /ep/folders/model-coverage-reports

Create a report for a list of folders

Long running task Create RBT or B2B model coverage report for a list of folders.
In the RBT use-case, RBTestCases from the folders will be used to generate the report.
In the B2B use-case, RBTestCases and Stimuli-Vectors from the folders will be used to generate the report.

REQUEST

REQUEST BODY - application/json

```
{
    type*                 enum        ALLOWED:RBT, B2B
                                      Specifies the testing use-case for which the model coverage report should be
                                      created.<br>Must be 'B2B' or 'RBT' for Back-to-back testing or Requirement based test,
                                      respectively.
    simulationKind*       string      Specifies the simulation mode: 'SL MIL' or 'TL MIL'
    folderUIDs*           [string]    The comma separated list of folder UIDs for which the model coverage report shall be
                                      created.
    useShortCircuitLogic  boolean     Specifies if short circuit logic should be used for Simulink blocks. <br>The paramter is
                                      optional. If not provided, the current setting from EP is used.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

23.3 POST /ep/scopes/{scope-uid}/model-coverage-reports

Create a report for a scope

Long running task Create RBT or B2B model coverage report on given scope.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The scope UID for which to create the model coverage report.

REQUEST BODY - application/json

```
{
    type*                    enum    ALLOWED: RBT, B2B
                                     Specifies the testing use-case for which the model coverage report should be created.<br>Must
                                     be 'B2B' or 'RBT' for Back-to-back testing or Requirement based test, respectively.
    simulationKind*          string  Specifies the simulation mode: 'SL MIL' or 'TL MIL'
    useShortCircuitLogic     boolean Specifies if short circuit logic should be used for Simulink blocks. <br>The paramter is optional.
                                     If not provided, the current setting from EP is used.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

23.4 GET /ep/model-coverage-reports

Get a list of reports

Retrieve all model coverage reports of the specified testing use-case from the profile.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
coverage-type	enum ALLOWED: RBT, B2B	The model coverage testing use-case.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
reportName	string	Name of the report.
reportType	string	Type of the report.

```
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

24. PREFERENCES

Set and retrieve preferences.

24.1 GET /ep/preferences/{preference-name}

Get a preference

Get the preference with a given name. If the retrieved value is empty, the preference might have its default value.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*preference-name	string	Enter the name of the preference that you want retrieved.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
   preferenceName*    string   The name of the preference.
   preferenceValue*   string   The value of the preference.
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

24.2 PUT /ep/preferences

Set a list of preferences

Set new values for a list of given preferences. The preference name and new value must be provided for each of them.

REQUEST

REQUEST BODY - application/json

```
[{
Array of object:

   preferenceName*    string   The name of the preference.
   preferenceValue*   string   The value of the preference.
}]
```

RESPONSE

STATUS CODE - 200: Preferences set

RESPONSE MODEL - application/json

```
{
    messages [string]
}
```

STATUS CODE - 400: Bad request

   RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

   RESPONSE MODEL - text/plain

string

```
{
    messages [string]
}
```

STATUS CODE - 400: Bad request

   RESPONSE MODEL - text/plain

string

25. PROFILES

Create and handle EP Profiles.

25.1 GET /ep/profiles

Get the active profile
Use this command to get the currently active profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
   uid      string      READ-ONLY
                           The unique identifier (UID) of this object.
   metadata* {
   The metadata containing all relevant info about the profile.
   }
   profilePath {
      path  string   The location where the profile is stored.
   }
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

25.2 PUT /ep/profiles

Save the profile
Use this command to save your active profile at a given location. Keep in mind to use only legal profile paths.

REQUEST

REQUEST BODY - application/json

```
{
   path  string   The location where the profile is stored.
}
```

RESPONSE

STATUS CODE - 201: Created

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

25.3 POST /ep/profiles

Create a profile

Use this command to create a new empty profile. It won't contain a path, since it's not stored anywhere yet.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
discardCurrentProfile	boolean	If 'true' the current profile is discarded and a new profile will be created. Otherwise a new profile will only be created when the current profile is not in a dirty state.Default is 'false'

RESPONSE

STATUS CODE - 201: Created

RESPONSE MODEL - application/json

```
{
   uid    string    READ-ONLY
                     The unique identifier (UID) of this object.
   metadata* {
   The metadata containing all relevant info about the profile.
   }
   profilePath {
      path  string  The location where the profile is stored.
   }
}
```

STATUS CODE - 428: Precondition Required

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

25.4 DELETE /ep/profiles

Discard a profile

Use this command to discard a profile, even if it is dirty. Changes will not be saved!

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

25.5 GET /ep/profiles/{profile-path}

Open a profile

Use this command to open an existing profile. Specifiy the path to the profile with a name of your choice. Keep in mind to only use legal profile paths of already existing profiles.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*profile-path	string	The path to the existing profile.

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
discardCurrentProfile	boolean	If 'true' the current profile is discarded and a new profile will be created. Otherwise a new profile will only be created when the current profile is not in a dirty state.Default is 'false'

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
    uid      string      READ-ONLY
                         The unique identifier (UID) of this object.
    metadata* {
    The metadata containing all relevant info about the profile.
    }
    profilePath {
       path   string   The location where the profile is stored.
    }
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 428: Precondition Required

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

26. PROGRESS

Get the current status of long-running operations.

26.1 GET /ep/progress

Get the status of a long running operation

Get the status of a long running operation. If the operation is on-going, the current progress will be returned. If the operation is complete, the resulted object will be returned if there is one. An error will be returned if it occured during the long-running operation.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
progress-id	string	The progress id. Can be retrieved by starting a long-running operation. If not specified then last long running operation result will be returned.

RESPONSE

STATUS CODE - 200: Operation is complete. No resulting object is returned.

RESPONSE MODEL - application/json

```
{
   message   string
   progress  integer
   result {
   }
}
```

STATUS CODE - 201: Operation is complete. Resulting object is returned as JSON.

RESPONSE MODEL - application/json

```
{
   message   string
   progress  integer
   result {
   }
}
```

STATUS CODE - 202: Operation is currently in progress

RESPONSE MODEL - application/json

```
{
   message   string
   progress  integer
   result {
   }
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

27. PROJECT REPORTS

Create and retrieve project reports.

27.1 POST /ep/scopes/{scope-uid}/project-report

Create a project report

Create a project report.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope for which the report shall be created.

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
template-name	string	The name of the template to use for report creation. This can be either the full path and filename of a template file to use, or the name of a template that was provided to EP by setting the template folder via the preference REPORT_TEMPLATE_FOLDER If no template is given, all available data will be included.

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

> **RESPONSE MODEL - application/json**
>
> {
>   jobID   string   READ-ONLY
>   The ID of a job.
> }

STATUS CODE - 400: Bad request

> **RESPONSE MODEL - text/plain**

string

STATUS CODE - 404: Not found

> **RESPONSE MODEL - text/plain**

string

STATUS CODE - 500: Internal server error

> **RESPONSE MODEL - text/plain**

string

27.2 GET /ep/project-reports

Get all project reports

Retrieve all project reports from the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
reportName	string	Name of the report.
reportType	string	Type of the report.

```
}]
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

28. PROOFS

Create, verify and execute proofs.

28.1 GET /ep/proofs/{proofUID}

Get a proof based on given proofUID.

Get a proof based on given proofUID found on the active profile.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*proofUID	string	The UID of the proof.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
  uid                                    string          READ-ONLY
                                                         The unique identifier (UID) of this object.

  name                                   string          Name of the proof.
  description                            string          Description of the proof.
  formalRequirementName                  string          Name of the formal requirement used by this
                                                         proof.

  proofSettings {
  The proof settings object. If not provided, default settings will be used.

      timeoutSeconds          integer    The timeout in seconds, i.e. the limit on how long this proof execution may run.
                                         Non-positive numbers (<= 0) are interpreted as infinity (i.e. no timeout has been
                                         specified).

      memoryLimit             integer    The memory limit in MB. Valid values are <= 0 (to denote infinity) or any positive
                                         integer >= 500 to set a positive memory limit.

      maxNumberOfThreads      integer    The maximum number of parallel threads for the proof execution. Valid values
                                         are between 1 and available number of cores. <br>It is possible to set this
                                         parameter to a value of -1, which will compute the number of threads
                                         automatically as half of the available number of cores.

      maximumSearchDepth      integer    The maximum search depth, which is the number of unwindings of the main
                                         function. Non-positive numbers (<= 0) are interpreted as infinity.

      maximumLoopUnroll       integer    The maximum number of unwindings for internal loops. Non-positive numbers
                                         (<= 0) are interpreted as infinity.

      resultExpectation       enum       ALLOWED: FULFILLED, VIOLATED, UNKNOWN
                                         The expected result for the proof's execution.

      engines [{
      Array of object:  The engines of the proof execution.

          name*      enum      ALLOWED: SMIBMC, VIS, AUTOFXP, CBMC, ISAT
                               The name of the engine.
          enabled    boolean   Whether or not this engine should be considered. Default is true
      }]
      isAllowAssumption           boolean    Set to 'true', if assumption check is allowed.
      useLocalInputRestriction    boolean    Whether or not the proof uses local input restrictions.
  }
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

28.2 PUT /ep/proofs/{proofUID}

Apply proof settings.

Apply proof settings on given proofUID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*proofUID	string	The ID of the proof.

REQUEST BODY - application/json
{
The proof settings object. If not provided, default settings will be used.

timeoutSeconds	integer	The timeout in seconds, i.e. the limit on how long this proof execution may run. Non-positive numbers (<= 0) are interpreted as infinity (i.e. no timeout has been specified).
memoryLimit	integer	The memory limit in MB. Valid values are <= 0 (to denote infinity) or any positive integer >= 500 to set a positive memory limit.
maxNumberOfThreads	integer	The maximum number of parallel threads for the proof execution. Valid values are between 1 and available number of cores. It is possible to set this parameter to a value of -1, which will compute the number of threads automatically as half of the available number of cores.
maximumSearchDepth	integer	The maximum search depth, which is the number of unwindings of the main function. Non-positive numbers (<= 0) are interpreted as infinity.
maximumLoopUnroll	integer	The maximum number of unwindings for internal loops. Non-positive numbers (<= 0) are interpreted as infinity.
resultExpectation	enum	**ALLOWED:** FULFILLED, VIOLATED, UNKNOWN The expected result for the proof's execution.

engines [{
Array of object: The engines of the proof execution.

name*	enum	**ALLOWED:** SMIBMC, VIS, AUTOFXP, CBMC, ISAT The name of the engine.
enabled	boolean	Whether or not this engine should be considered. Default is true

}]

isAllowAssumption	boolean	Set to 'true', if assumption check is allowed.
useLocalInputRestriction	boolean	Whether or not the proof uses local input restrictions.

}

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

> **RESPONSE MODEL - text/plain**

string

28.3 POST /ep/proofs/{frUid}

Create a proof.

Create a proof based on Formal Requirement Uid.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*frUid	string	The UID of the formal requirement.

RESPONSE

STATUS CODE - 201: Created

> **RESPONSE MODEL - application/json**

```
{
    uid                                     string              READ-ONLY
                                                                The unique identifier (UID) of this object.

    name                                    string              Name of the proof.
    description                             string              Description of the proof.
    formalRequirementName                   string              Name of the formal requirement used by this
                                                                proof.

    proofSettings {
    The proof settings object. If not provided, default settings will be used.

        timeoutSeconds          integer     The timeout in seconds, i.e. the limit on how long this proof execution may run.
                                            Non-positive numbers (<= 0) are interpreted as infinity (i.e. no timeout has been
                                            specified).

        memoryLimit             integer     The memory limit in MB. Valid values are <= 0 (to denote infinity) or any positive
                                            integer >= 500 to set a positive memory limit.

        maxNumberOfThreads      integer     The maximum number of parallel threads for the proof execution. Valid values
                                            are between 1 and available number of cores. <br>It is possible to set this
                                            parameter to a value of -1, which will compute the number of threads
                                            automatically as half of the available number of cores.

        maximumSearchDepth      integer     The maximum search depth, which is the number of unwindings of the main
                                            function. Non-positive numbers (<= 0) are interpreted as infinity.

        maximumLoopUnroll       integer     The maximum number of unwindings for internal loops. Non-positive numbers
                                            (<= 0) are interpreted as infinity.

        resultExpectation       enum        ALLOWED:FULFILLED, VIOLATED, UNKNOWN
                                            The expected result for the proof's execution.

        engines [{
        Array of object: The engines of the proof execution.

            name*       enum        ALLOWED:SMIBMC, VIS, AUTOFXP, CBMC, ISAT
                                    The name of the engine.
            enabled     boolean     Whether or not this engine should be considered. Default is true
        }]
        isAllowAssumption           boolean     Set to 'true', if assumption check is allowed.
        useLocalInputRestriction    boolean     Whether or not the proof uses local input restrictions.
    }
}
```

STATUS CODE - 404: Not found

> **RESPONSE MODEL - text/plain**

```
string
```

STATUS CODE - 500: Internal server error

> **RESPONSE MODEL - text/plain**

```
string
```

28.4 POST /ep/proofs/{proofUID}/execute

Execute a proof.

Execute a proof based on the given proofUID. If multiple executable architecture are preset, the architecture-uid query parameter must be specified.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*proofUID	string	The UID of the proof.

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
architecture-uid	string	The UID of the architecture on which this proof to be executed. If there are multiple valid architectures, but no architecture is specified, the C-Code architecture is used as a default.

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

> **RESPONSE MODEL - application/json**

```
{
   jobID  string  READ-ONLY
                   The ID of a job.
}
```

STATUS CODE - 400: Bad request

> **RESPONSE MODEL - text/plain**

```
string
```

STATUS CODE - 403: Forbidden

> **RESPONSE MODEL - text/plain**

```
string
```

STATUS CODE - 404: Not found

> **RESPONSE MODEL - text/plain**

```
string
```

STATUS CODE - 500: Internal server error

> **RESPONSE MODEL - text/plain**

string

28.5 GET /ep/proofs

Get all proofs.

Get all proofs found on the active profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

[{
Array of object:

uid	string	**READ-ONLY** The unique identifier (UID) of this object.
name	string	Name of the proof.
description	string	Description of the proof.
formalRequirementName	string	Name of the formal requirement used by this proof.

proofSettings {
The proof settings object. If not provided, default settings will be used.

timeoutSeconds	integer	The timeout in seconds, i.e. the limit on how long this proof execution may run. Non-positive numbers (<= 0) are interpreted as infinity (i.e. no timeout has been specified).
memoryLimit	integer	The memory limit in MB. Valid values are <= 0 (to denote infinity) or any positive integer >= 500 to set a positive memory limit.
maxNumberOfThreads	integer	The maximum number of parallel threads for the proof execution. Valid values are between 1 and available number of cores. It is possible to set this parameter to a value of -1, which will compute the number of threads automatically as half of the available number of cores.
maximumSearchDepth	integer	The maximum search depth, which is the number of unwindings of the main function. Non-positive numbers (<= 0) are interpreted as infinity.
maximumLoopUnroll	integer	The maximum number of unwindings for internal loops. Non-positive numbers (<= 0) are interpreted as infinity.
resultExpectation	enum	**ALLOWED:** FULFILLED, VIOLATED, UNKNOWN The expected result for the proof's execution.

engines [{
Array of object: The engines of the proof execution.

name*	enum	**ALLOWED:** SMIBMC, VIS, AUTOFXP, CBMC, ISAT The name of the engine.
enabled	boolean	Whether or not this engine should be considered. Default is true

}]

isAllowAssumption	boolean	Set to 'true', if assumption check is allowed.
useLocalInputRestriction	boolean	Whether or not the proof uses local input restrictions.

}
}]

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

28.6 GET /ep/proofs/{proofUID}/detailed-proof-results

Get detailed proof results of a proof.

Get detailed proof results of a proof by providing a proofUID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*proofUID	string	The UID of the proof.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

[{
Array of object:

proofId	string	The proof UID for which this result was generated.
steps	integer	The amount of steps after which the execution was terminated. -1 represents a step number which has no useful meaning.
architectureName	string	The architecture name on which the result was executed.
proofResult	string	The proof result after the execution.
terminationReason	string	Termination reason of the proof execution.
timeoutSeconds	integer	The timeout in seconds, i.e. the limit on how long this proof execution may run. Non-positive numbers (<= 0) are interpreted as infinity (i.e. no timeout has been specified).
memoryLimit	integer	The memory limit in MB. Valid values are -1 (to denote infinity) or any positive integer >= 500 to set a positive memory limit.
maxNumberOfThreads	integer	The maximum number of parallel proof execution threads. Valid values are between 1 and MAX_INT.
maximumSearchDepth	integer	The maximum search depth, which is the number of unwindings of the main function. Non-positive numbers (<= 0) are interpreted as infinity.
maximumLoopUnroll	integer	The maximum number of unwindings for internal loops. Non-positive numbers (<= 0) are interpreted as infinity.
usedEngines	[enum]	**ALLOWED:** SMIBMC, VIS, AUTOFXP, CBMC, ISAT
		The engines used in the proof execution.
isAllowAssumptionCheck	boolean	Returns true if assumption check was allowed.

inputRestrictions {
A list with the used input restriction for this execution.

}

expectedResult	enum	**ALLOWED:** FULFILLED, VIOLATED, UNKNOWN
		The expected result for the proof's execution.

}]

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

29. REAL-TIME TESTING

Execute operations with Real-Time Testing observers such as export

29.1 POST `/ep/rtt-observers-export`

Export RTT Observers

Long running task Export RTT Observers at specified folder location, and select which formal requirements to contain the export.

REQUEST

REQUEST BODY - application/json
```
{
    folderPath*             string      The path of the folder to which the RTT Observers should be exported.
    formalRequirementUIDs*  [string]    Formal requirements UIDs list.
    resetOnFailure          boolean     Set reset on failure flag. Default is false.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json
```
{
    jobID  string  READ-ONLY
                   The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

30. REGRESSION TEST REPORTS

Creates a Regression Test Report for a given Regression Test. Retrieves all existing Regression Test reports from the profile.

30.1 POST /ep/regression-tests/{regression-test-uid}/regression-test-reports

Create a report for a regression test

Creates a Regression Test Report on a regression test.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*regression-test-uid	string	The Regression test UID for which the Regression Report is created.

RESPONSE

STATUS CODE - 201: Created

RESPONSE MODEL - application/json

```
{
    uid          string    READ-ONLY
                           The unique identifier (UID) of this object.
    reportName   string    Name of the report.
    reportType   string    Type of the report.
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

30.2 GET /ep/regression-test-reports

Get all reports

Retrieve all the Regression Test reports from the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
Array of object:
```

```
  uid           string    READ-ONLY
                          The unique identifier (UID) of this object.
  reportName    string    Name of the report.
  reportType    string    Type of the report.
}]
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

31. REGRESSION TESTS

Creates regression tests.

31.1 GET /ep/regression-tests/{regression-test-uid}

Get a test

Get the Regression Test with the provided UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*regression-test-uid	string	The UID of the Regression Test to be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
  uid                  string      READ-ONLY
                                   The unique identifier (UID) of this object.
  referenceMode        string      The reference execution config type.
  comparisonMode       string      The comparison execution config type.
  referenceFolderUIDs  [string]    Reference folder UIDs
  comparisonFolderUID  string      Comparison folder UID
  executionDate        string      Execution Date
  verdictStatus        enum        ALLOWED:PASSED, FAILED, ERROR, FAILED_ACCEPTED
                                   The verdict status
  verdictState         enum        ALLOWED:VALID, OUTDATED_TOLERANCE_UPDATE,
                                   OUTDATED_MISSING_EXECUTIONS
                                   Verdict State
  failed               integer     Number of failed comparisons.
  failedAccepted       integer     Number of failed accepted comparisons.
  passed               integer     Number of passed comparisons.
  error                integer     Number of comparisons with error.
  total                integer     Total number of comparisons.
  comparisons [{
  Array of object: All comparisons.

     uid                             string    READ-ONLY
                                               The unique identifier (UID) of this object.
     name                            string    The name of Test Case / Stimuli Vector used in Comparison.
     verdictStatus                   enum      ALLOWED:PASSED, FAILED, ERROR, FAILED_ACCEPTED
                                               The verdict status
     referenceExecutionRecordUID     string    UID of reference execution record.
     comparisonExecutionRecordUID    string    UID of compared to execution record.
     comment                         string    Added comment for Comparison.
  }]
  name                 string      The name of the RegresionTest Test.
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

    RESPONSE MODEL - text/plain

string

31.2 PATCH /ep/regression-tests/{regression-test-uid}

Update verdict status for a comparison

Changes verdict status for a Comparison.If accept is true, the comparison verdict status is changed from 'failed' to 'failed (accepted)'. If accept is false, the comparison verdict status is changed from 'failed (accepted)'to 'failed'

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*regression-test-uid	string	The Regression Test UID for which to change the Comparison verdict.

REQUEST BODY - application/json

```
{
    comparisonUID*   string    UID of the Comparison
    accept*          boolean   If accept is true, the comparison verdict status is changed from 'failed' to 'failed (accepted)'. If accept is
                               false, the comparison verdict status is changed from 'failed (accepted)'to 'failed'
}
```

RESPONSE

STATUS CODE - 200: OK

    RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

31.3 POST /ep/folders/{folder-uid}/regression-tests

Generate a test on a folder

Long running task Generates a Regression Test on a given folder UID for the specified comparison execution kind. Optionally, the folder where to save the simulated execution records can be provided. If this property is not provided, the execution records are not saved.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The folder UID for which the Regression Test is generated.

REQUEST BODY - application/json

```
{
    compMode*        string   Comparison execution mode (e.g. 'SL MIL', 'TL MIL', 'TL MIL (EV)', 'SIL', 'PIL', 'SL SIL')
    compFolderUID    string   (Optional) The folder where to save the simulated ExecutionRecords. If this property is not provided, the
                              ExecutionRecords are not saved.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

31.4 POST /ep/folders/regression-tests

Generate a test on a list of folders

Long running task Generates a Regression Test on a given list of folder UIDs for the specified comparison execution kind. Optionally, the folder where to save the simulated execution records can be provided. If this property is not provided, the execution records are not saved.

REQUEST

REQUEST BODY - application/json

```
{
    compMode*        string   Comparison execution mode (e.g. 'SL MIL', 'TL MIL', 'TL MIL (EV)', 'SIL', 'PIL', 'SL SIL')
    compFolderUID    string   (Optional) The folder where to save the simulated ExecutionRecords. If this property is not provided, the
                              ExecutionRecords are not saved.
    UIDs*            [string]  Folder UID list
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
   jobID  string  READ-ONLY
                      The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

31.5 GET /ep/regression-tests

Get all tests

Get all Regression Tests from active profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
referenceMode	string	The reference execution config type.
comparisonMode	string	The comparison execution config type.
referenceFolderUIDs	[string]	Reference folder UIDs
comparisonFolderUID	string	Comparison folder UID
executionDate	string	Execution Date
verdictStatus	enum	ALLOWED:PASSED, FAILED, ERROR, FAILED_ACCEPTED
		The verdict status
verdictState	enum	ALLOWED:VALID, OUTDATED_TOLERANCE_UPDATE, OUTDATED_MISSING_EXECUTIONS
		Verdict State
failed	integer	Number of failed comparisons.
failedAccepted	integer	Number of failed accepted comparisons.
passed	integer	Number of passed comparisons.
error	integer	Number of comparisons with error.

total	integer	Total number of comparisons.

comparisons [{
Array of object: All comparisons.

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
name	string	The name of Test Case / Stimuli Vector used in Comparison.
verdictStatus	enum	**ALLOWED:**PASSED, FAILED, ERROR, FAILED_ACCEPTED
		The verdict status
referenceExecutionRecordUID	string	UID of reference execution record.
comparisonExecutionRecordUID	string	UID of compared to execution record.
comment	string	Added comment for Comparison.

}]

name	string	The name of the RegresionTest Test.

}]

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
name	string	The name of Test Case / Stimuli Vector used in Comparison.
verdictStatus	enum	**ALLOWED:**PASSED, FAILED, ERROR, FAILED_ACCEPTED
		The verdict status

32. REPORTS

Retrieve and export Reports.

32.1 GET /ep/reports/{report-uid}

Get a report

Get the report with the provided UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*report-uid	string	The UID of the report to be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
   uid           string    READ-ONLY
                           The unique identifier (UID) of this object.
   reportName    string    Name of the report.
   reportType    string    Type of the report.
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

32.2 POST /ep/reports/{report-uid}

Export a report

Use this command export a report to a given location. The exported report corresponds to the given UID inside the command. New name can be set for file. If file exists, will be overwritten. Keep in mind to use only legal profile paths.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*report-uid	string	The UID of the report to be returned.

REQUEST BODY - application/json

```
{
   exportPath*   string    Path to export report
```

```
    newName        string    (Optional) New report name.
}
```

RESPONSE

STATUS CODE - 201: Exported Succesfully. Returns exported report location.

STATUS CODE - 400: Bad Request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

    RESPONSE MODEL - text/plain

string

33. REQUIREMENT-BASED TEST CASES

Handle requirement-based Test Cases.

33.1 GET /ep/test-cases-rbt/{testcase-uid}

Get a test case

Get a requirement-based Test Case by providing its UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*testcase-uid	string	The UID of the requirement-based Test Case to be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
    uid                 string      READ-ONLY
                                    The unique identifier (UID) of this object.
    name                string      The name of the RBTestCase.
    description         string      An optional description of the RBTestCase.
    kind                string      The datatype or kind of the RBTestCase. Usually "tc" or "csv".
    length              integer     The length of the vector.
    draft               boolean     States whether or not the RBTestCase is in Draft-Mode.
    lastModifiedDate    string      The date of the last modification to the RBTestCase
    folderUID           string      The unique identifier of the folder the RBTestCase belongs to.
    scopeUID            string      The unique identifier of the scope the RBTestCase belongs to.
    requirementUIDs     [string]    The unique identifiers of the requirements belonging to the RBTestCase.
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

33.2 DELETE /ep/test-cases-rbt/{testcase-uid}

Delete a test case

Delete a requirement-based Test Case by providing its UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
*testcase-uid	string	The UID of the requirement-based Test Case to be deleted.

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

33.3 POST /ep/test-cases-rbt-export

Export test cases

Long running task Export single or multiple requirement-based Test Case(s) by providing the list of the test cases which will be exported, the export directory, the export format and a list of additional options for export.

REQUEST

REQUEST BODY - application/json

```
{
   UIDs*                [string]   List with the UIDs of the elements which will be exported
   exportDirectory*     string     Directory where to export the elements
   exportFormat         enum       ALLOWED:TC, EXCEL, CSV, JSON
                                   The format of the exported test cases. Default value is EXCEL.

   additionalOptions {
     csvDelimiter       enum       ALLOWED:SEMICOLON, COMMA, COLON, PIPE
                                   Relevant only for CSV export format. It can have one of the following values: "SEMICOLON",
                                   "COMMA", "COLON", "PIPE". Default value is "SEMICOLON".

     singleFile         boolean    Relevant only for CSV export format: false - each vector will be exported in it's own file; true - all
                                   vectors will be exported in same file. Default is 'false'

     architectureUid    string     Relevant only for Excel export format. It specifies the UID of the architecture on which the
                                   interfaces of the vectors will be exported. Default is the master architecture
   }
   overwritePolicy      enum       ALLOWED:EXTEND_NAME, OVERWRITE, SKIP
                                   Overwrite policy: allowed values (not case-sensitive) are: EXTEND_NAME, in which case if the
                                   exported file exists on disk, its name will be extended and the original file on disk will be kept,
                                   OVERWRITE, in which case the original file on disk is overwritten, if it exists. Default value is
                                   EXTEND_NAME

}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
   jobID  string   READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad Request

> **RESPONSE MODEL - text/plain**

string

STATUS CODE - 403: Forbidden

> **RESPONSE MODEL - text/plain**

string

STATUS CODE - 500: Internal server error

> **RESPONSE MODEL - text/plain**

string

33.4 GET /ep/test-cases-rbt

Get all test cases

Get all requirement-based Test Cases.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

> **RESPONSE MODEL - application/json**

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
name	string	The name of the RBTestCase.
description	string	An optional description of the RBTestCase
kind	string	The datatype or kind of the RBTestCase. Usually "tc" or "csv".
length	integer	The length of the vector.
draft	boolean	States whether or not the RBTestCase is in Draft-Mode.
lastModifiedDate	string	The date of the last modification to the RBTestCase
folderUID	string	The unique identifier of the folder the RBTestCase belongs to.
scopeUID	string	The unique identifier of the scope the RBTestCase belongs to.
requirementUIDs	[string]	The unique identifiers of the requirements belonging to the RBTestCase.

```
}]
```

STATUS CODE - 500: Internal server error

> **RESPONSE MODEL - text/plain**

string

33.5 PUT /ep/test-cases-rbt

Import test cases

LONG RUNNING TASK Import multiple requirement-based Test Cases from external files by providing their Path and an Overwrite Policy.

REQUEST

> **REQUEST BODY - application/json**

```
{
    paths*          [string]    The paths to all test cases you would like to import.
    folderUID       string      The UID of the folder you want to import into. If not specified Test Cases will be imported in the
                                default Test Cases folder.
    overwritePolicy enum        ALLOWED:EXTEND_NAME, OVERWRITE, SKIP
                                Decides what happens in case of duplicate names. Can be "EXTEND_NAME", "OVERWRITE" or
                                "SKIP". Default is "SKIP".
    draft           boolean     Sets the Draft-Mode of the test cases. By default its value is false.
    csvDelimiter    enum        ALLOWED:SEMICOLON, COMMA, COLON, PIPE
                                Relevant only for CSV export format. It can have one of the following values: "SEMICOLON",
                                "COMMA", "COLON", "PIPE". Default value is "SEMICOLON".
    importKind      enum        ALLOWED:TC, EXCEL
                                The kind a file should be imported as. Possible values are: "TC" or "EXCEL". "TC" by default
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

33.6 GET /ep/folders/{folder-uid}/test-cases-rbt

Get a list of test cases from a folder

Get all requirement-based Test Cases from a certain Folder by providing its UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The UID of the Folder containing the requirement based Test Cases.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
name	string	The name of the RBTestCase.
description	string	An optional description of the RBTestCase
kind	string	The datatype or kind of the RBTestCase. Usually "tc" or "csv".
length	integer	The length of the vector.
draft	boolean	States whether or not the RBTestCase is in Draft-Mode.
lastModifiedDate	string	The date of the last modification to the RBTestCase
folderUID	string	The unique identifier of the folder the RBTestCase belongs to.
scopeUID	string	The unique identifier of the scope the RBTestCase belongs to.
requirementUIDs	[string]	The unique identifiers of the requirements belonging to the RBTestCase.

```
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

33.7 GET /ep/scopes/{scope-uid}/test-cases-rbt

Get a list of test cases from a scope

Get all requirement-based Test Cases from a certain Scope by providing its UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the Scope containing the requirement based Test Cases.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
name	string	The name of the RBTestCase.
description	string	An optional description of the RBTestCase
kind	string	The datatype or kind of the RBTestCase. Usually "tc" or "csv".
length	integer	The length of the vector.
draft	boolean	States whether or not the RBTestCase is in Draft-Mode.
lastModifiedDate	string	The date of the last modification to the RBTestCase
folderUID	string	The unique identifier of the folder the RBTestCase belongs to.
scopeUID	string	The unique identifier of the scope the RBTestCase belongs to.
requirementUIDs	[string]	The unique identifiers of the requirements belonging to the RBTestCase.

```
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

33.8 GET /ep/requirements/{requirement-uid}/test-cases-rbt

Get a list of test cases linked to a requirement

Use this command to retrieve the test cases linked to a given requirement uid.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*requirement-uid	string	The uid of the requirement for which all linked test cases should be returned.

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
only-indirect-testcases	boolean	The flag used to specify whether to retrieve only test cases which are indirectly linked to a given requirement. This flag has an effect only on non-leaves requirements. If no value is specified, this flag will be set to false.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
name	string	The name of the RBTestCase.
description	string	An optional description of the RBTestCase
kind	string	The datatype or kind of the RBTestCase. Usually "tc" or "csv".
length	integer	The length of the vector.
draft	boolean	States whether or not the RBTestCase is in Draft-Mode.
lastModifiedDate	string	The date of the last modification to the RBTestCase
folderUID	string	The unique identifier of the folder the RBTestCase belongs to.
scopeUID	string	The unique identifier of the scope the RBTestCase belongs to.
requirementUIDs	[string]	The unique identifiers of the requirements belonging to the RBTestCase.

```
}]
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error.

    RESPONSE MODEL - text/plain

```
string
```

33.9 PUT /ep/requirements/test-cases-rbt

Unlink requirements from test cases

Use this command to unlink requirements from test cases. Only their uids must be specified.

REQUEST

REQUEST BODY - application/json

```
{
    requirementUIDs*    [string]   The uids of the requirements that need to be linked to test cases.
    testCaseUIDs*       [string]   The uids of the test cases that need to be linked to the requirements.
}
```

RESPONSE

    STATUS CODE - 200: Requirements unlinked from test cases.

    STATUS CODE - 400: Bad request.

        RESPONSE MODEL - text/plain

```
string
```

    STATUS CODE - 500: Internal server error.

        RESPONSE MODEL - text/plain

```
string
```

33.10 POST /ep/requirements/test-cases-rbt

Link requirements to test cases

Use this command to link requirements to test cases. Only their uids must be specified.

REQUEST

REQUEST BODY - application/json

```
{
    requirementUIDs*    [string]   The uids of the requirements that need to be linked to test cases.
    testCaseUIDs*       [string]   The uids of the test cases that need to be linked to the requirements.
}
```

RESPONSE

    STATUS CODE - 200: Requirements linked to test cases.

    STATUS CODE - 400: Bad request.

        RESPONSE MODEL - text/plain

```
string
```

    STATUS CODE - 500: Internal server error.

string

33.11 POST /ep/formal-requirements/test-cases-rbt

Generate test cases based on formal requirements

Long running task Generate test cases based on formal requirements using a specific configuration.

REQUEST

REQUEST BODY - application/json

{

formalRequirements*	[string]	The formal requirements for which the coverage will be generated.
behaviour	enum	ALLOWED:IMPLEMENTATION, FORMAL_REQUIREMENTS
		Specifies if the vector generation shall be based on the behaviour of the SUT or the behaviour specified via formal requirements. 'IMPLEMENTATION' (default, if profile contains SUT): Uses behaviour from SUT 'FORMAL_REQUIREMENTS': Uses behaviour described in formal requirements (see SurroundingFormalRequirements).
architecture	string	Architecture uid for which generation is performed. (Only valid for behaviour: FORMAL_REQUIREMENTS).
surroundingFormalRequirements	[string]	Specifies the formal requirements that describe the behaviour as array. (only valid for corresponding 'Behaviour' selection) By default all formal requirements of the scope (see first parameter) are taken into account.
restrictOutputBehaviour	boolean	Set whether or not values for outputs and locals should keep their values if these are not written by formal requirement (only valid for Behaviour: Formal Requirements). Default is 'true'
coverageMetric	enum	ALLOWED:ONCE, LINEAR_CONDITION_COVERAGE, MULTIPLE_CONDITION_COVERAGE
		Specifies the type of coverage metric. Default is 'ONCE'
coverageTarget	enum	ALLOWED:NONE, TRIGGER_EVENT_COVERAGE, TRIGGER_CONDITION_COVERAGE, TRIGGER_EVENT_CONDITION_COVERAGE
		Specifies the type of coverage target. Allowed values depend on the selected 'coverageMetric': 'NONE' (default) for 'ONCE' coverageMetric, 'TRIGGER_EVENT_COVERAGE' (default), 'TRIGGER_CONDITION_COVERAGE', or 'TRIGGER_EVENT_CONDITION_COVERAGE' (Trigger event/condition coverage) for'LINEAR_CONDITION_COVERAGE' or 'MULTIPLE_CONDITION_COVERAGE' coverageMetric.
onlyFulfillingTestCases	boolean	Specifies if only fulfilling test cases shall be considered. Default is 'true'
memoryLimit	string	Defines the memory limit (MB), default: 'inf'.
globalTimeout	string	Defines the global timeout (s), default 'inf'.
timeoutPerGoal	string	Defines the timeout per goal (s), default 'inf'.
searchDepth	string	Defines the search depth (steps), default 'inf'.
loopUnroll	string	Defines the loop unroll, default 32.
numberOfThreads	integer	Defines the maximum number of parallel proof execution threads. Valid values are between 1 and available number of cores. It is possible to set this parameter to a value of -1, which will compute the number of threads automatically as half of the available number of cores. Default is '1'
initializationVectorUID	string	The UID of the RequirementBasedTestCase which shall be used to initialize the engine
checkBoundaryCoverageGoals	boolean	Specifies if boundary coverage goals shall be considered. Default is taken from the preferences.
relativeEpsilon	string	Relative epsilon used to derive boundary coverage goals. Default is taken from the preferences.
minimumEpsilon	string	Minimum epsilon used to derive boundary coverage goals. Default is taken from the preferences.

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

34. REQUIREMENT-BASED TEST EXECUTION

Executes requirement-based Testing.

34.1 POST /ep/folders/{folder-uid}/test-execution-rbt

Execute all test cases from a folder

Long running task Executes a requirement-based Test on all Test Cases from a given folder for the specified execution kinds. Optionally, also the model coverage report can be generated or the test execution can be forced to not re-use any previous results.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The folder UID for which the requirement-based Test is executed.

REQUEST BODY - application/json

```
{
    execConfigNames*            [string]    List of execution kinds (example: SIL, TL MIL, SL MIL, etc.). Not case-sensitive.
    generateModelCoverageReport boolean     Specify (optional) if the model coverage report should be generated. Default
                                            value is false.
    forceExecute                boolean     Specify (optional) if the test execution should be forced (all previous results will
                                            be discarded). Default value is false.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

34.2 POST /ep/folders/test-execution-rbt

Execute all test cases from a list of folders

Long running task Executes a requirement-based Test on all Test Cases from a given list of folders for the specified execution kinds. Optionally, also the model coverage report can be generated or the test execution can be forced to not re-use any previous results.

REQUEST

REQUEST BODY - application/json

```
{
  UIDs*                        [string]              List with unique identifiers of the objects.
  data {
    execConfigNames*           [string]    List of execution kinds (example: SIL, TL MIL, SL MIL, etc.). Not case-
                                           sensitive.
    generateModelCoverageReport boolean    Specify (optional) if the model coverage report should be generated.
                                           Default value is false.
    forceExecute               boolean     Specify (optional) if the test execution should be forced (all previous
                                           results will be discarded). Default value is false.
  }
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
  jobID  string  READ-ONLY
                 The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

34.3 POST /ep/test-cases-rbt/{testcase-uid}/test-execution-rbt

Execute a test case

Long running task Executes a requirement-based Test on a requirement-based Test Case for the specified execution kinds. Optionally, also the test execution can be forced to not re-use any previous results.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*testcase-uid	string	The requirement-based Test Case UID for which the RBT is executed.

REQUEST BODY - application/json
```
{
    execConfigNames*   [string]   List of execution kinds (example: SIL, TL MIL, SL MIL, etc.).
    forceExecute       boolean    Specify (optional) if the test execution should be forced (all previous results will be discarded).
                                  Default value is false.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

34.4 POST /ep/test-cases-rbt/test-execution-rbt

Execute a list of test cases
Long running task Executes a requirement-based Test on a list of requirement-based Test Cases for the specified execution kinds. Optionally, also the test execution can be forced to not re-use any previous results.

REQUEST

REQUEST BODY - application/json

```
{
    UIDs*                  [string]                List with unique identifiers of the objects.
    data {
        execConfigNames*   [string]   List of execution kinds (example: SIL, TL MIL, SL MIL, etc.).
        forceExecute       boolean    Specify (optional) if the test execution should be forced (all previous results will be
                                      discarded). Default value is false.
    }
```

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                      The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

34.5 POST /ep/requirements-sources/{requirements-source-uid}/test-execution-rbt

Execute all test cases linked to a requirements source

Long running task Executes a requirement-based Test on all Test Cases linked to the requirements of the given requirements source for the specified execution kinds. Optionally, also the model coverage report can be generated or the test execution can be forced to not re-use any previous results.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*requirements-source-uid	string	The requirements source UID for which the RBT is executed.

REQUEST BODY - application/json

```
{
    execConfigNames*            [string]   List of execution kinds (example: SIL, TL MIL, SL MIL, etc.). Not case-sensitive.
    generateModelCoverageReport boolean    Specify (optional) if the model coverage report should be generated. Default
                                           value is false.
    forceExecute                boolean    Specify (optional) if the test execution should be forced (all previous results will
                                           be discarded). Default value is false.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID    string    READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

34.6 POST /ep/requirements-sources/test-execution-rbt

Execute all test cases linked to a list of requirements sources

Long running task Executes a requirement-based Test on all Test Cases linked to the requirements of the given requirements sources for the specified execution kinds. Optionally, also the model coverage report can be generated or the test execution can be forced to not re-use any previous results.

REQUEST

REQUEST BODY - application/json

```
{
    UIDs*                        [string]                      List with unique identifiers of the objects.
    data {
        execConfigNames*         [string]    List of execution kinds (example: SIL, TL MIL, SL MIL, etc.). Not case-
                                             sensitive.
        generateModelCoverageReport  boolean  Specify (optional) if the model coverage report should be generated.
                                             Default value is false.
        forceExecute             boolean    Specify (optional) if the test execution should be forced (all previous
                                             results will be discarded). Default value is false.
    }
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID    string    READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

34.7 POST /ep/scopes/{scope-uid}/test-execution-rbt

Execute all test cases from a scope

Long running task Executes a requirement-based Test on all Test Cases from a given scope for the specified execution kinds. Optionally, also the model coverage report can be generated or the test execution can be forced to not re-use any previous results.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The scope UID for which the requirement-based Test is executed.

REQUEST BODY - application/json

```
{
    execConfigNames*              [string]   List of execution kinds (example: SIL, TL MIL, SL MIL, etc.). Not case-sensitive.
    generateModelCoverageReport   boolean    Specify (optional) if the model coverage report should be generated. Default
                                             value is false.
    forceExecute                  boolean    Specify (optional) if the test execution should be forced (all previous results will
                                             be discarded). Default value is false.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

34.8 POST /ep/scopes/test-execution-rbt

Execute all test cases from a list of scopes

Long running task Executes a requirement-based Test on all Test Cases from a given list of scopes for the specified execution kinds. Optionally, also the model coverage report can be generated or the test execution can be forced to not re-use any previous results.

REQUEST

REQUEST BODY - application/json

```
{
  UIDs*                       [string]              List with unique identifiers of the objects.
  data {
    execConfigNames*          [string]    List of execution kinds (example: SIL, TL MIL, SL MIL, etc.). Not case-
                                          sensitive.
    generateModelCoverageReport boolean   Specify (optional) if the model coverage report should be generated.
                                          Default value is false.
    forceExecute              boolean     Specify (optional) if the test execution should be forced (all previous
                                          results will be discarded). Default value is false.
  }
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

    RESPONSE MODEL - application/json

```
{
  jobID  string  READ-ONLY
                  The ID of a job.
}
```

STATUS CODE - 400: Bad Request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

35. REQUIREMENT-BASED TEST EXECUTION REPORTS

Creates requirement-based Test Execution Reports.

35.1 POST /ep/folders/{folder-uid}/test-execution-reports-rbt

Create a report on a folder

Create a requirement-based Test Execution Report on a given folder.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The folder UID for which the Test Execution Report is created.

REQUEST BODY - application/json
```
{
    execConfigName*   string   Execution kind (example: SIL, TL MIL, SL MIL, etc.). Not case-sensitive.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json
```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain
```
string
```

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain
```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain
```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain
```
string
```

35.2 POST /ep/folders/test-execution-reports-rbt

Create a report on a list of folders

Create a requirement-based Test Execution Report on a given list of folders.

REQUEST

REQUEST BODY - application/json
```
{
    UIDs*              [string]    List with unique identifiers of the objects.
    execConfigName*    string      Execution kind (example: SIL, TL MIL, SL MIL, etc.).
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json
```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

`string`

35.3 POST /ep/requirements-sources/{requirements-source-uid}/test-execution-reports-rbt

Create a report on a requirements source

Create a requirement-based Test Execution Report on a given requirements source.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*requirements-source-uid	string	The requirements source UID for which the Test Execution Report is created.

REQUEST BODY - application/json
```
{
```

```
        execConfigName*    string    Execution kind (example: SIL, TL MIL, SL MIL, etc.). Not case-sensitive.
    }
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
    {
        jobID   string   READ-ONLY
                         The ID of a job.
    }
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

35.4 POST /ep/requirements-sources/test-execution-reports-rbt

Create a report on a list of requirements sources

Create a requirement-based Test Execution Report on a given list of requirements sources.

REQUEST

REQUEST BODY - application/json
```
    {
        UIDs*              [string]   List with unique identifiers of the objects.
        execConfigName*    string     Execution kind (example: SIL, TL MIL, SL MIL, etc.).
    }
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
    {
        jobID   string   READ-ONLY
                         The ID of a job.
    }
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

35.5 POST /ep/scopes/{scope-uid}/test-execution-reports-rbt

Create a report on a scope

Create a requirement-based Test Execution Report on a given scope.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The Scope UID for which the Test Execution Report is created.

REQUEST BODY - application/json

```
{
    execConfigName*  string  Execution kind (example: SIL, TL MIL, SL MIL, etc.). Not case-sensitive.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                   The ID of a job.
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

35.6 POST /ep/scopes/test-execution-reports-rbt

Create a report on a list of scopes

Create a requirement-based Test Execution Report on a given list of scopes.

REQUEST

REQUEST BODY - application/json
```
{
    UIDs*              [string]    List with unique identifiers of the objects.
    execConfigName*    string      Execution kind (example: SIL, TL MIL, SL MIL, etc.).
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

    RESPONSE MODEL - application/json
```
{
    jobID  string  READ-ONLY
                   The ID of a job.
}
```

STATUS CODE - 400: Bad request

    RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

    RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

35.7 GET /ep/test-execution-reports-rbt

Get all reports

Retrieve all the requirement-based Test Execution reports from the profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
reportName	string	Name of the report.
reportType	string	Type of the report.

```
}]
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

36. REQUIREMENTS

Retrieve Requirements, Linked Requirements, or Requirement Sources.

36.1 GET /ep/requirements-sources

Get all requirement sources
Get all requirement sources found on the opened profile.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

kind*	string	The kind of this requirement source. The kind identifies this source to be from a specific requirement management tool.

```
settings* [{
```
Array of object: A list with requirement settings.For EXCEL requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date, excel_file_path, projectName_attr (or excel_sheet_name), excel_id_attr, excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date, projectName_attr, doors_module_qualifier, doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.

key*	string	For EXCEL requirement kind, the following keys are mandatory: excel_file_path, projectName_attr, excel_id_attr. The optional setting keys are: excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys are mandatory: projectName_attr, doors_module_qualifier. The optional setting keys are: doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.
value*	string	Setting value

```
}]
```

definedAdditionalAttributes	[string]	A list with additional attributes.
externalUUID*	string	The unique ID identifying the external requirement source.
name*	string	The requirement source name.
uid*	string	The universally unique identifier of this requirement source.

```
}]
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

36.2 GET /ep/requirements/{requirement-source-id}

Get all requirements of a requirement source
Get the requirements of the given requirement source id.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*requirement-source-id	string	The UID of the requirement source.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid*	string	The universally unique identifier.
identifier*	string	The requirement identifier (e.g. a chapter number within DOORS).
isRemoved	boolean	Value meaning whether this requirement has been removed within the requirement management tool.

```
additionalAttributes {
```
Map containing all additional attributes.

```
}
```

scopeId*	string	The scope id this requirement is directly linked to.
description*	string	The requirement description.
name*	string	The requirement name.
dateOfLastUpdate*	string	The requirement last update date.

```
requirementSource* {
```

kind*	string	The kind of this requirement source. The kind identifies this source to be from a specific requirement management tool.

```
settings* [{
```
Array of object: A list with requirement settings.For EXCEL requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date, excel_file_path, projectName_attr (or excel_sheet_name), excel_id_attr, excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date, projectName_attr, doors_module_qualifier, doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.

key*	string	For EXCEL requirement kind, the following keys are mandatory: excel_file_path, projectName_attr, excel_id_attr. The optional setting keys are: excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys are mandatory: projectName_attr, doors_module_qualifier. The optional setting keys are: doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.
value*	string	Setting value

```
}]
```

definedAdditionalAttributes	[string]	A list with additional attributes.
externalUUID*	string	The unique ID identifying the external requirement source.
name*	string	The requirement source name.
uid*	string	The universally unique identifier of this requirement source.

```
  }
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

36.3 GET /ep/scopes/{scope-id}/linked-requirements

Get all requirements for a scope

Get the linked requirements of this given scope id.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-id	string	The UID of the scope id.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid*	string	The universally unique identifier.
identifier*	string	The requirement identifier (e.g. a chapter number within DOORS).
isRemoved	boolean	Value meaning whether this requirement has been removed within the requirement management tool.

```
additionalAttributes {
```
Map containing all additional attributes.
```
}
```

scopeId*	string	The scope id this requirement is directly linked to.
description*	string	The requirement description.
name*	string	The requirement name.
dateOfLastUpdate*	string	The requirement last update date.

```
requirementSource* {
```

kind*	string	The kind of this requirement source. The kind identifies this source to be from a specific requirement management tool.

```
settings* [{
```
Array of object: A list with requirement settings.For EXCEL requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date, excel_file_path, projectName_attr (or excel_sheet_name), excel_id_attr, excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date, projectName_attr, doors_module_qualifier, doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.

key*	string	For EXCEL requirement kind, the following keys are mandatory: excel_file_path, projectName_attr, excel_id_attr. The optional setting keys are: excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys are mandatory: projectName_attr, doors_module_qualifier. The optional setting keys are: doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.
value*	string	Setting value

```
}]
```

definedAdditionalAttributes	[string]	A list with additional attributes.
externalUUID*	string	The unique ID identifying the external requirement source.
name*	string	The requirement source name.
uid*	string	The universally unique identifier of this requirement source.

```
  }
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

37. REQUIREMENTS IMPORT

Import Requirements.

37.1 POST /ep/requirements-import

Import requirements

Import requirements by specifying the kind of requirements.

REQUEST

REQUEST BODY - application/json
{

kind*	enum	**ALLOWED:**EXCEL, DOORS
		The kind of imported requirements. Allowed types: EXCEL or DOORS.
nameAttribute	string	The name of imported requirements. Required for DOORS and EXCEL import.
descriptionAttribute	string	The description of imported requirements. Required for DOORS and EXCEL import.
additionalAttributes	[string]	A list with additional attributes.

settings* [{

Array of object: A list with requirement settings.For EXCEL requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date, excel_file_path, projectName_attr (or excel_sheet_name), excel_id_attr, excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date, projectName_attr, doors_module_qualifier, doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.

    key*   string   For EXCEL requirement kind, the following keys are mandatory: excel_file_path, projectName_attr, excel_id_attr. The optional setting keys are: excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys are mandatory: projectName_attr, doors_module_qualifier. The optional setting keys are: doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.

    value* string   Setting value

  }]

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

{

  jobID string **READ-ONLY**

                The ID of a job.

}

STATUS CODE - 400: Bad request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

37.2 GET /ep/requirements-import/doors

Get the DOORS™ configuration template

Get the configuration with default settings for importing different kinds of requirements.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
    kind*                          enum          ALLOWED:EXCEL, DOORS
                                                 The kind of imported requirements. Allowed
                                                 types: EXCEL or DOORS.
    nameAttribute                  string        The name of imported requirements. Required
                                                 for DOORS and EXCEL import.
    descriptionAttribute           string        The description of imported requirements.
                                                 Required for DOORS and EXCEL import.
    additionalAttributes           [string]      A list with additional attributes.
    settings* [{
```
Array of object: A list with requirement settings.For EXCEL requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date, excel_file_path, projectName_attr (or excel_sheet_name), excel_id_attr, excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date, projectName_attr, doors_module_qualifier, doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.

```
        key*    string   For EXCEL requirement kind, the following keys are mandatory: excel_file_path, projectName_attr, excel_id_attr.
                          The optional setting keys are: excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys
                          are mandatory: projectName_attr, doors_module_qualifier. The optional setting keys are:
                          doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.
        value*  string   Setting value
    }]
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

37.3 GET /ep/requirements-import/excel

Get the Excel™ configuration template

Get the configuration with default settings for importing different kinds of requirements.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
    kind*                                    enum            ALLOWED:EXCEL, DOORS
                                                             The kind of imported requirements. Allowed
                                                             types: EXCEL or DOORS.
    nameAttribute                            string          The name of imported requirements. Required
                                                             for DOORS and EXCEL import.
    descriptionAttribute                     string          The description of imported requirements.
                                                             Required for DOORS and EXCEL import.
    additionalAttributes                     [string]        A list with additional attributes.
    settings* [{
    Array of object:  A list with requirement settings.For EXCEL requirement kind, the following keys can be: name_attr_value, desc_attr_value,
    additional_attr, modification_date, excel_file_path, projectName_attr (or excel_sheet_name), excel_id_attr, excel_start_row,
    excel_parent_id.For DOORS requirement kind, the following keys can be: name_attr_value, desc_attr_value, additional_attr, modification_date,
    projectName_attr, doors_module_qualifier, doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.

        key*       string   For EXCEL requirement kind, the following keys are mandatory: excel_file_path, projectName_attr, excel_id_attr.
                            The optional setting keys are: excel_start_row, excel_parent_id.For DOORS requirement kind, the following keys
                            are mandatory: projectName_attr, doors_module_qualifier. The optional setting keys are:
                            doors_baseline_major, doors_baseline_minor, doors_baseline_suffix.
        value*     string   Setting value
    }]
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

38. SCOPES

Handle Scopes.

38.1 GET /ep/scopes/{scope-uid}

Get a scope

Get a specific scope by providing its UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope to be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
   uid            string      The unique identifier (UID) of this object.
   name           string      The scope name.
   topLevel       boolean     True if scope is a toplevel scope.
   kind           enum        ALLOWED:SUT, DUMMY, ENVIRONMENT, HIDDEN_INTERNAL, VIRTUAL
                              Scope kind.
   path           string      Scope path.
   architecture   string      The corresponding architecture of the scope.
   sampleTime {
   The sample time of the scope.

      uid         string      The unique identifier (UID) of this object.
      seconds     string      The sample time as a value given in seconds.
   }
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

```
string
```

38.2 GET /ep/scopes

Get a list of scopes

Get a list of scopes by a query which filters by path and top-level.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
path	string	Enter the path of the scope you would like to search for. If null, then all scopes will be returned.
top-level	boolean	Specifies, if only top level scopes shall be returned. If set to 'true', only top level scopes will be returned. If set to 'false', all scopes will be returned, including the top level. If not specified, then the default value is 'false'.
architecture-uid	string	Specifies from which architecture the scopes are being collected. If null, then the first imported architecture will be used.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	The unique identifier (UID) of this object.
name	string	The scope name.
topLevel	boolean	True if scope is a toplevel scope.
kind	enum	**ALLOWED:**SUT, DUMMY, ENVIRONMENT, HIDDEN_INTERNAL, VIRTUAL Scope kind.
path	string	Scope path.
architecture	string	The corresponding architecture of the scope.

```
    sampleTime {
```
The sample time of the scope.

uid	string	The unique identifier (UID) of this object.
seconds	string	The sample time as a value given in seconds.

```
    }
}]
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

39. SIGNALS

Handle signals' operations.

39.1 GET /ep/scopes/{scope-uid}/signals

Get all signals from a scope

Get all signals from the given scope

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The scope for which to get the signals.

RESPONSE

STATUS CODE - 200: OK

   RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY
		The unique identifier (UID) of this object.
identifier	string	READ-ONLY
		The signal identifier.

```
}]
```

STATUS CODE - 404: Not found

   RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

   RESPONSE MODEL - text/plain

string

40. STIMULI VECTORS

Handle stimuli vectors.

40.1 GET /ep/stimuli-vectors/{stimuli-vector-uid}

Get a stimuli vector
Get a specific stimuli vector by UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*stimuli-vector-uid	string	The UID of the stimuli vector to be returned.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
    uid          string    READ-ONLY
                           The unique identifier (UID) of this object.
    name         string    The name of the StimuliVector.
    length       integer   The length of the vector.
    folderUID    string    The unique identifier of the folder the StimuliVector belongs to.
    scopeUID     string    The unique identifier of the scope the StimuliVector belongs to.
}
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

40.2 DELETE /ep/stimuli-vectors/{stimuli-vector-uid}

Delete a stimuli vector
Delete a stimuli vector by its UID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*stimuli-vector-uid	string	The UID of the stimuli vector to be deleted.

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 404: Not found

   RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

   RESPONSE MODEL - text/plain

string

40.3 POST /ep/stimuli-vectors-export

Export Stimuli Vectors

Long running task Export single or multiple Stimuli Vector(s) by providing the list of the stimuli vectors which will be exported, the export directory, the export format and a list of additional options for export.

REQUEST

REQUEST BODY - application/json

```
{
   UIDs*              [string]    List with the UIDs of the elements which will be exported
   exportDirectory*   string      Directory where to export the elements
   exportFormat       enum        ALLOWED:EXCEL, CSV
                                  The format of the exported stimuli vectors. It can be: "EXCEL" or "CSV". Default value is EXCEL.
   additionalOptions {
      csvDelimiter    enum        ALLOWED:SEMICOLON, COMMA, COLON, PIPE
                                  Relevant only for CSV export format. It can have one of the following values: "SEMICOLON",
                                  "COMMA", "COLON", "PIPE". Default value is "SEMICOLON".
      singleFile      boolean     Relevant only for CSV export format: false - each vector will be exported in it's own file; true - all
                                  vectors will be exported in same file. Default is 'false'
      architectureUid string      Relevant only for Excel export format. It specifies the UID of the architecture on which the
                                  interfaces of the vectors will be exported. Default is the master architecture
   }
   overwritePolicy    enum        ALLOWED:EXTEND_NAME, OVERWRITE, SKIP
                                  Overwrite policy: allowed values (not case-sensitive) are: EXTEND_NAME, in which case if the
                                  exported file exists on disk, its name will be extended and the original file on disk will be kept,
                                  OVERWRITE, in which case the original file on disk is overwritten, if it exists. Default value is
                                  EXTEND_NAME
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

   RESPONSE MODEL - application/json

```
{
   jobID   string   READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad Request

   RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

40.4 GET /ep/stimuli-vectors

Get all stimuli vectors

Get all stimuli vectors.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY	
		The unique identifier (UID) of this object.	
name	string	The name of the StimuliVector.	
length	integer	The length of the vector.	
folderUID	string	The unique identifier of the folder the StimuliVector belongs to.	
scopeUID	string	The unique identifier of the scope the StimuliVector belongs to.	

```
}]
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

40.5 PUT /ep/stimuli-vectors

Import stimuli vectors

Long running task Import multiple stimuli vectors from external files into a Folder by providing their path and the Folders UID. Can either overwrite or skip stimuli vectors that are already imported.

REQUEST

REQUEST BODY - application/json
```
{
```

paths*	[string]	The paths to all stimuli vectors you'd like to import.	
vectorKind	enum	ALLOWED: TC, EXCEL	
		The stimuli vector type. Default value is "TC"	
folderUID	string	The UID of the folder you want to import into.	
delimiter	enum	ALLOWED: SEMICOLON, COMMA, COLON, PIPE	
		The CSV file delimiter, can be: "SEMICOLON", "COMMA", "COLON", "PIPE". Default value is "SEMICOLON"	

overwritePolicy	enum	ALLOWED:EXTEND_NAME, OVERWRITE, SKIP
		Decides what happens in case of duplicate names. Can be "EXTEND_NAME", "OVERWRITE" or "SKIP". Default is "SKIP".

}

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

{

   jobID   string   READ-ONLY

                  The ID of a job.

}

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

40.6 GET /ep/folders/{folder-uid}/stimuli-vectors

Get all stimuli vectors from a folder

Get all stimuli vectors from the specified folder.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The UID of the folder from which to get stimuli vectors.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

[{

Array of object:

   uid   string   READ-ONLY

                        The unique identifier (UID) of this object.

   name   string   The name of the StimuliVector.

   length   integer   The length of the vector.

   folderUID string   The unique identifier of the folder the StimuliVector belongs to.

   scopeUID   string   The unique identifier of the scope the StimuliVector belongs to.

}]

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

40.7 GET /ep/scopes/{scope-uid}/stimuli-vectors

Get all stimuli vectors from a scope

Get all stimuli vectors from the specified scope.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The UID of the scope from which to get stimuli vectors.

RESPONSE

STATUS CODE - 200: OK

    RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
name	string	The name of the StimuliVector.
length	integer	The length of the vector.
folderUID	string	The unique identifier of the folder the StimuliVector belongs to.
scopeUID	string	The unique identifier of the scope the StimuliVector belongs to.

```
}]
```

STATUS CODE - 404: Not found

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

    RESPONSE MODEL - text/plain

string

41. TEST

41.1 GET /ep/test

Test the connection to the REST server

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Request successfully sent.

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

    RESPONSE MODEL - text/plain

string

42. TEST CASE/STIMULI VECTOR SIMULATION

Simulate TestCases.

42.1 POST /ep/folders/{folder-uid}/testcase-simulation

Simulates all Test Cases/StimuliVectors from a folder

Long running task Simulates all Test Cases/StimuliVectors from a given folder for the specified execution kinds. Optionally, simulation can be forced to not re-use any previous results.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*folder-uid	string	The Folder UID for which the Test Cases/Stimuli Vectors are simulated.

REQUEST BODY - application/json

```
{
    execConfigNames*   [string]   List of execution kinds (example: SIL, TL MIL, SL MIL, etc.) for which to simulate. Not case-
                                  sensitive.
    forceExecute       boolean    Specify (optional) if the simulation should be forced (all previous results will be discarded).
                                  Default is 'false'.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

42.2 POST /ep/folders/testcase-simulation

Simulates all Test Cases/StimuliVectors from a list of folders

Long running task Simulates all Test Cases/StimuliVectors from a given list of folders for the specified execution kinds. Optionally, simulation can be forced to not re-use any previous results.

REQUEST

REQUEST BODY - application/json

```
{
    execConfigNames*    [string]    List of execution kinds (example: SIL, TL MIL, SL MIL, etc.) for which to simulate. Not case-
                                    sensitive.
    forceExecute        boolean     Specify (optional) if the simulation should be forced (all previous results will be discarded).
                                    Default is 'false'.
    UIDs*               [string]    List with unique identifiers of the objects.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID    string    READ-ONLY
                       The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

42.3 POST /ep/scopes/{scope-uid}/testcase-simulation

Simulates all Test Cases/StimuliVectors from a scope

Long running task Simulates all Test Cases/StimuliVectors from a given scope for the specified execution kinds. Optionally, simulation can be forced to not re-use any previous results.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*scope-uid	string	The Scope UID for which the Test Cases/Stimuli Vectors are simulated.

REQUEST BODY - application/json

```
{
```

```
    execConfigNames*   [string]   List of execution kinds (example: SIL, TL MIL, SL MIL, etc.) for which to simulate. Not case-
                                   sensitive.
    forceExecute       boolean    Specify (optional) if the simulation should be forced (all previous results will be discarded).
                                   Default is 'false'.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

```
string
```

42.4 POST /ep/scopes/testcase-simulation

Simulates all Test Cases/StimuliVectors from a list of scopes

Long running task Simulates all Test Cases/StimuliVectors from a given list of scopes for the specified execution kinds. Optionally, simulation can be forced to not re-use any previous results.

REQUEST

REQUEST BODY - application/json

```
{
    execConfigNames*   [string]   List of execution kinds (example: SIL, TL MIL, SL MIL, etc.) for which to simulate. Not case-
                                   sensitive.
    forceExecute       boolean    Specify (optional) if the simulation should be forced (all previous results will be discarded).
                                   Default is 'false'.
    UIDs*              [string]   List with unique identifiers of the objects.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

42.5 POST /ep/test-cases/{testcase-uid}/testcase-simulation

Simulates a Test Case/Stimuli Vector

Long running task Simulates a Test Case/Stimuli Vector for the specified execution kinds. Optionally, also the test execution can be forced to not re-use any previous results.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*testcase-uid	string	The Test Case/Stimuli VectorUID to simulate.

REQUEST BODY - application/json

```
{
    execConfigNames*  [string]   List of execution kinds (example: SIL, TL MIL, SL MIL, etc.) for which to simulate. Not case-
                                 sensitive.
    forceExecute      boolean    Specify (optional) if the simulation should be forced (all previous results will be discarded).
                                 Default is 'false'.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID  string  READ-ONLY
                    The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

42.6 POST /ep/test-cases/testcase-simulation

Simulates all Test Cases/StimuliVectors

Long running task Simulates all Test Cases/StimuliVectors on the specified execution kinds. Optionally, simulation can be forced to not re-use any previous results.

REQUEST

REQUEST BODY - application/json

```
{
    execConfigNames*   [string]    List of execution kinds (example: SIL, TL MIL, SL MIL, etc.) for which to simulate. Not case-
                                   sensitive.
    forceExecute       boolean     Specify (optional) if the simulation should be forced (all previous results will be discarded).
                                   Default is 'false'.
    UIDs*              [string]    List with unique identifiers of the objects.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json

```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

string

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

string

43. TOLERANCES

Import, reset and retrieve global and local tolerances.

43.1 POST /ep/profiles/export-global-tolerances

Export global tolerances

Use this command to export the global tolerances per usecase.

REQUEST

REQUEST BODY - application/json
```
{
    path*                string    The path to the xml file to use for import or export of tolerances.
    toleranceUseCase*    enum      ALLOWED:B2B, RBT
                                   The use case of the tolerances for which tolerances should be applied. Allowed values are RBT and
                                   B2B.
}
```

RESPONSE

STATUS CODE - 202: Long running operation started. The status of the operation can be reviewed by using the Progress i.e. GET '/ep/progress?progress-id={progress-id}' using the id received by this command.

RESPONSE MODEL - application/json
```
{
    jobID   string   READ-ONLY
                     The ID of a job.
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 403: Forbidden

RESPONSE MODEL - text/plain

`string`

STATUS CODE - 500: Internal server error

RESPONSE MODEL - text/plain

`string`

43.2 GET /ep/scopes/{scope-id}/global-tolerances

Get the global tolerances

Use this command to retrieve the global tolerances. A valid scope uid and use case must be specified. The lead-lag-unit can be either Seconds or Steps.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
*scope-id	string	The scope from which to retrieve the global tolerances.

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*kind	enum **ALLOWED:** B2B, RBT	The use case kind. Can be either RBT or B2B.
lead-lag-unit	enum **ALLOWED:** SECONDS, STEPS	For existing tolerances, lead and lag values should be displayed as either SECONDS or STEPS. Default is STEPS.

RESPONSE

STATUS CODE - 200: Global tolerances retrieved.

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	**READ-ONLY**
		The unique identifier (UID) of this object.
name	string	
lead	string	
lag	string	
absTol	string	
relTol	string	
kind	string	

```
}]
```

STATUS CODE - 400: Bad request.

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Not found.

RESPONSE MODEL - text/plain

string

STATUS CODE - 406: Not acceptable.

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

43.3 PUT /ep/profiles/global-tolerances

Import the global tolerances

Use this command to import the global tolerances.

REQUEST

REQUEST BODY - application/json
```
{
    path*                string    The path to the xml file to use for import or export of tolerances.
    toleranceUseCase*    enum      ALLOWED:B2B, RBT
                                   The use case of the tolerances for which tolerances should be applied. Allowed values are RBT and
                                   B2B.
}
```

RESPONSE

STATUS CODE - 200: Global tolerances imported.

STATUS CODE - 400: Bad request.

    RESPONSE MODEL - text/plain

string

STATUS CODE - 406: Not acceptable.

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

    RESPONSE MODEL - text/plain

string

43.4 DELETE /ep/profiles/global-tolerances

Reset the global tolerances

Use this command to reset the global tolerances for the profile.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*kind	enum ALLOWED: B2B, RBT	The use case kind. Can be either RBT or B2B.

RESPONSE

STATUS CODE - 200: Global tolerances reset.

STATUS CODE - 400: Bad request.

    RESPONSE MODEL - text/plain

string

STATUS CODE - 406: Not acceptable.

    RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

    RESPONSE MODEL - text/plain

string

43.5 GET /ep/test-cases/{test-case-id}/local-tolerances

Get the local tolerances

Use this command to retrieve the local tolerances. A valid test case uid must be specified. The lead-lag-unit can be either Seconds or Steps. Setting the create-if-missing to true will create local tolerances from the global tolerances.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*test-case-id	string	The UID of the test case for which local tolerances will be retrieved.

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
lead-lag-unit	enum ALLOWED: SECONDS, STEPS	For existing tolerances, lead and lag values should be displayed as either SECONDS or STEPS. Default is STEPS.
create-if-missing	boolean	If true, the local tolerances will be created if they do not exist yet. Default is: false.

RESPONSE

STATUS CODE - 200: Local tolerances retrieved.

RESPONSE MODEL - application/json

```
[{
```
Array of object:

uid	string	READ-ONLY The unique identifier (UID) of this object.
name	string	
lead	string	
lag	string	
absTol	string	
relTol	string	
kind	string	

```
}]
```

STATUS CODE - 400: Bad request.

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Test case not found.

RESPONSE MODEL - text/plain

string

STATUS CODE - 406: Not acceptable.

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

43.6 PUT /ep/test-cases/{test-case-id}/local-tolerances

Import the local tolerances

Import local tolerances for a given test case.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*test-case-id	string	The UID of the test case for which local tolerances will be imported.

REQUEST BODY - application/json

```
{
    path*    string    The path to the xml file to use for import or export of local tolerances.
}
```

RESPONSE

STATUS CODE - 200: Local tolerances imported.

STATUS CODE - 400: Bad request.

RESPONSE MODEL - text/plain

string

STATUS CODE - 404: Test case not found.

RESPONSE MODEL - text/plain

string

STATUS CODE - 406: Not acceptable.

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: Internal server error.

RESPONSE MODEL - text/plain

string

43.7 POST /ep/test-cases/{test-case-id}/local-tolerances

Export local tolerances

Export local tolerances for a given test case.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*test-case-id	string	The UID of the requirement-based Test Case to export the local tolerances for.

REQUEST BODY - application/json

```
{
```

`path*` `string`   The path to the xml file to use for import or export of local tolerances.
  `}`

RESPONSE

STATUS CODE - 200: Local tolerances exported.

STATUS CODE - 400: Bad request.

   RESPONSE MODEL - text/plain

`string`

STATUS CODE - 404: Test case not found.

   RESPONSE MODEL - text/plain

`string`

STATUS CODE - 500: Internal server error.

   RESPONSE MODEL - text/plain

`string`

43.8 DELETE /ep/test-cases/{test-case-id}/local-tolerances

Reset the local tolerances
Remove the local tolerances for a test case.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*`test-case-id`	`string`	The UID of the test case for which local tolerances will be removed.

RESPONSE

STATUS CODE - 200: Local tolerances removed.

STATUS CODE - 400: Bad request.

   RESPONSE MODEL - text/plain

`string`

STATUS CODE - 404: Test case not found.

   RESPONSE MODEL - text/plain

`string`

STATUS CODE - 406: Not acceptable.

   RESPONSE MODEL - text/plain

`string`

STATUS CODE - 500: Internal server error.

   RESPONSE MODEL - text/plain

`string`