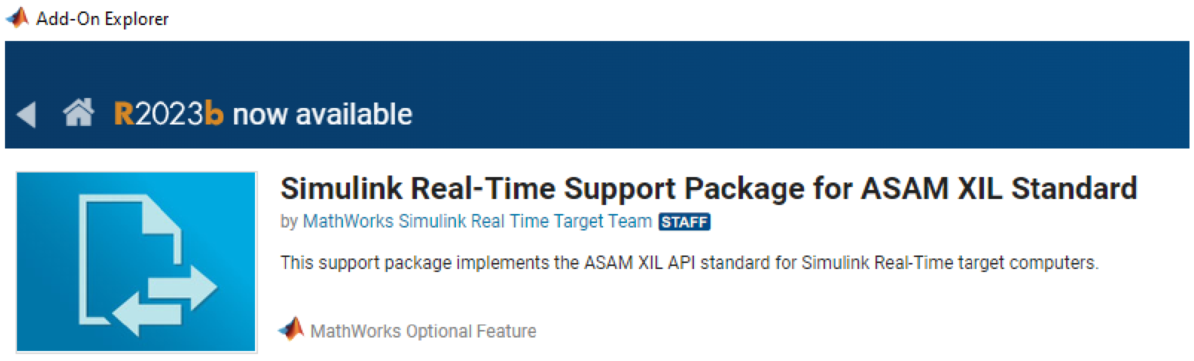
Test Automation with ecu.test using Simulink® and Speedgoat Test Systems

CONTENTS

1 Prerequisites for Using ecu.test

1.1 In MATLAB®, select Home > Add-Ons > Get Add-Ons and install the Simulink® Real-Time™ XIL API support package.


Add-On Explorer

R2023b now available

Simulink Real-Time Support Package for ASAM XIL Standard

by MathWorks Simulink Real Time Target Team STAFF

This support package implements the ASAM XIL API standard for Simulink Real-Time target computers.

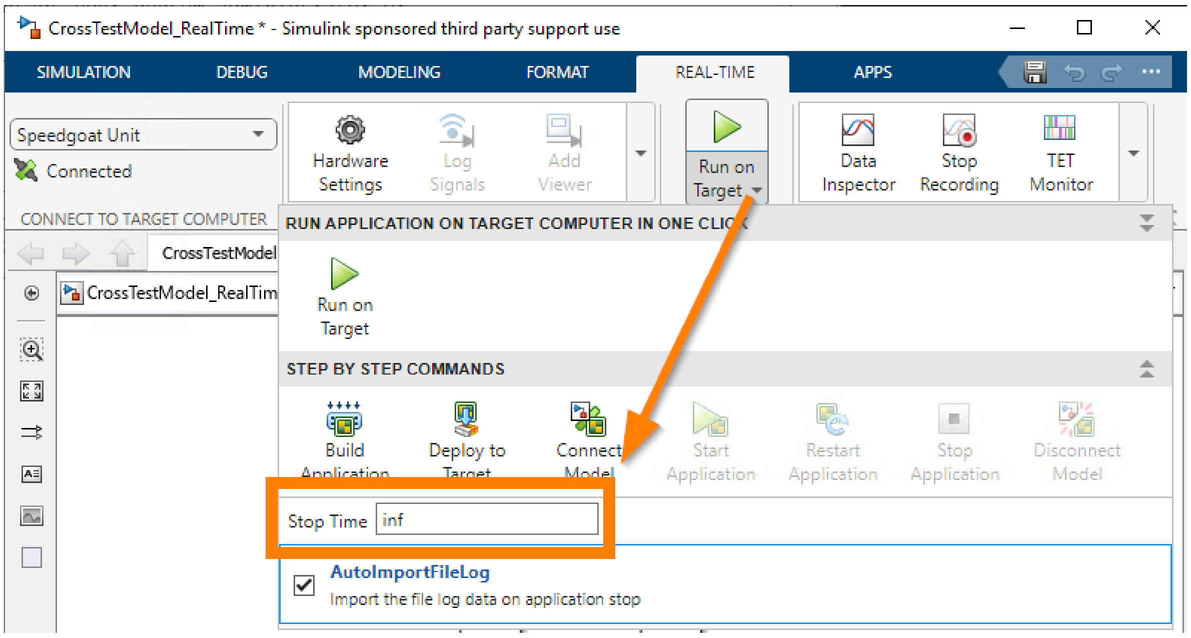
MathWorks Optional Feature

1.2 After support package installation, verify that the manifest file `MathWorksXILServer.imf` is located under `C:\ProgramData\ASAM\XIL\Implementation` and provides the correct Assembly path.
For R2024a, that would be:
```
C:\ProgramData\MATLAB\SupportPackages\R2024a\toolbox\slrealtime\xil\src\bin
\win64
```

1.3 Register MATLAB as the automation server and share the MATLAB session by typing in the MATLAB Command Window:
```
comserver('register','User','current');
enableservice('AutomationServer', true);
```
If you do not, ecu.test opens a new MATLAB session when configuring the test bench and test.

1.4 Connect your Speedgoat target machine to your development computer.

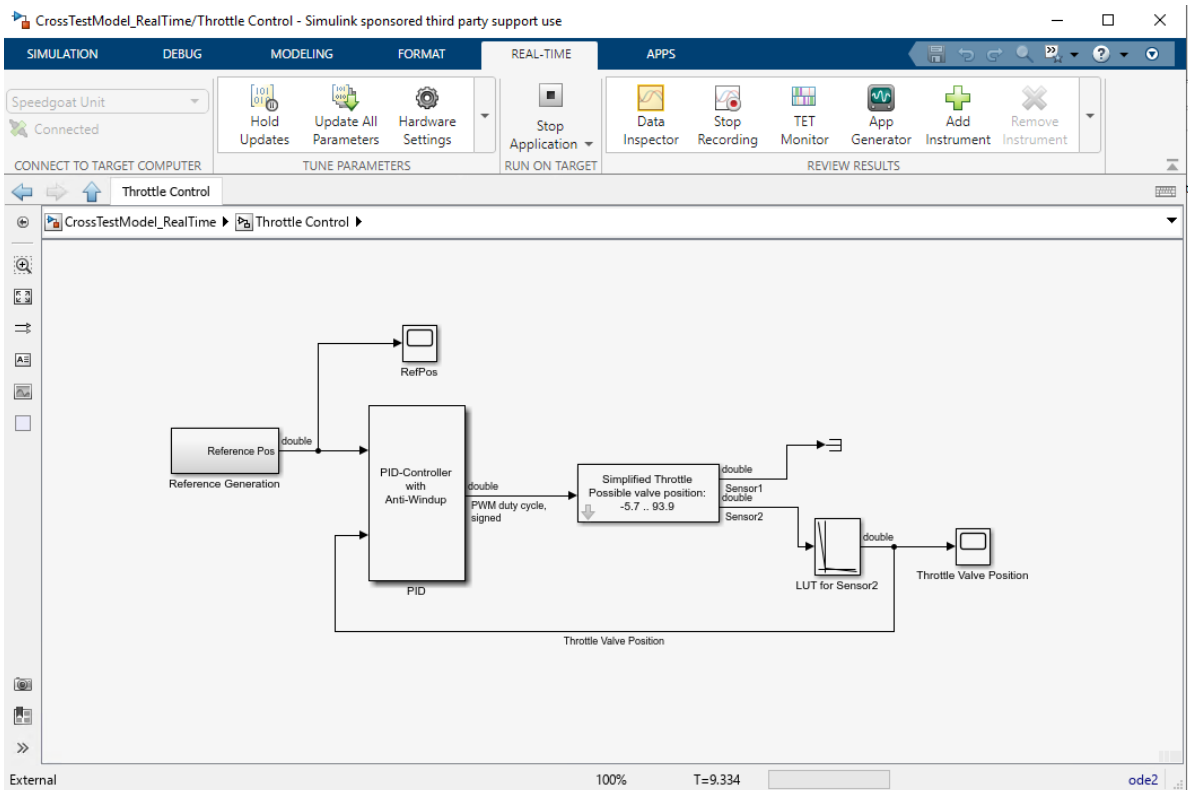
1.5 When building the real-time application (*.mldatx), ensure no shorter stop time is set than the test case requires. Otherwise, the real-time application would be terminated during test execution on the real-time system, even before the test case has been completed.
For this reason, it is recommended to set the stop time to “inf” when building the real-time application. ecu.test automatically takes care of terminating and reloading the real-time application.



Stop Time: inf

☑ **AutoImportFileLog**
Import the file log data on application stop

1.6 Build the model and click on “Run on Target”. Make sure this runs without any errors. The real-time application MLDATX file is required to set up the test bench and test in ecu.test.

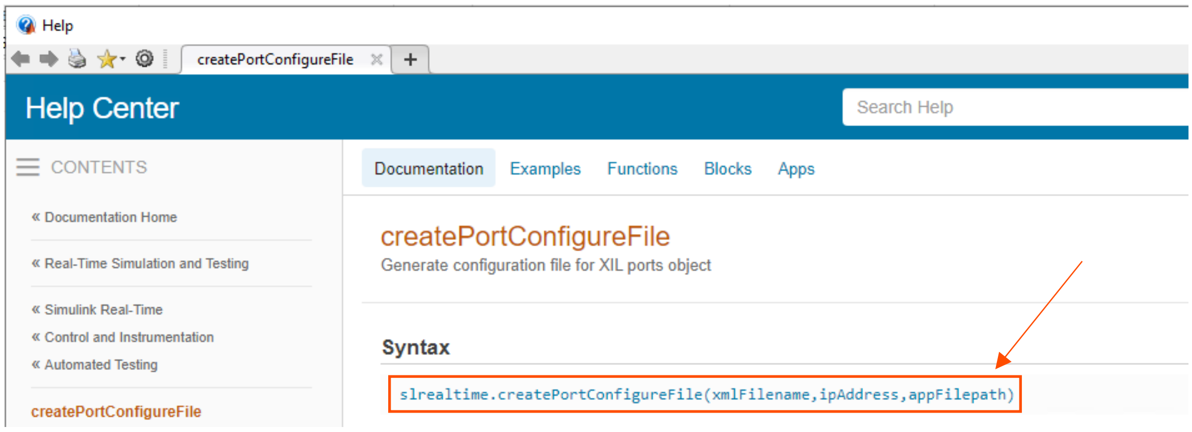


1.7 Generate XIL configuration with MATLAB command createPortConfigureFile. The command: `slrealtime.createPortConfigureFile(xmlFilename,ipAddress,appFilepath)`

In the example from the figure, such a command could look like this:

```
slrealtime.createPortConfigureFile('CrossTestModel_RealTime.xml','192.168.1
42.29',‘CrossTestModel_RealTime')
```

generates an XML file configuring a XIL ports object for ecu.test.



createPortConfigureFile

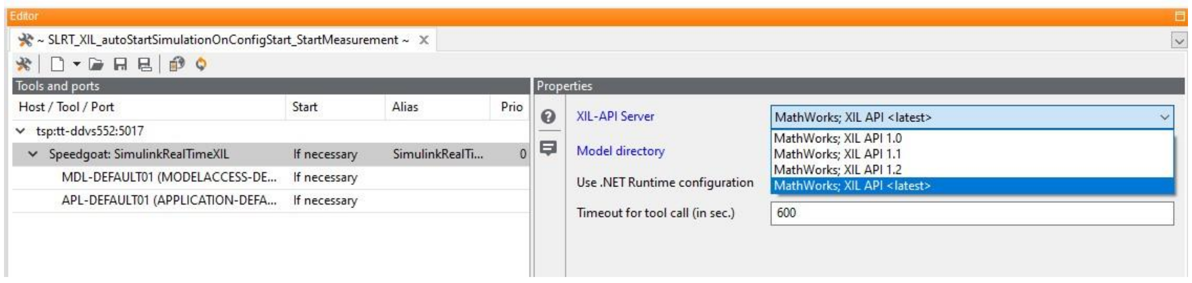
Generate configuration file for XIL ports object

Syntax

```
slrealtime.createPortConfigureFile(xmlFilename,ipAddress,appFilepath)
```

2 Create a New Test Bench

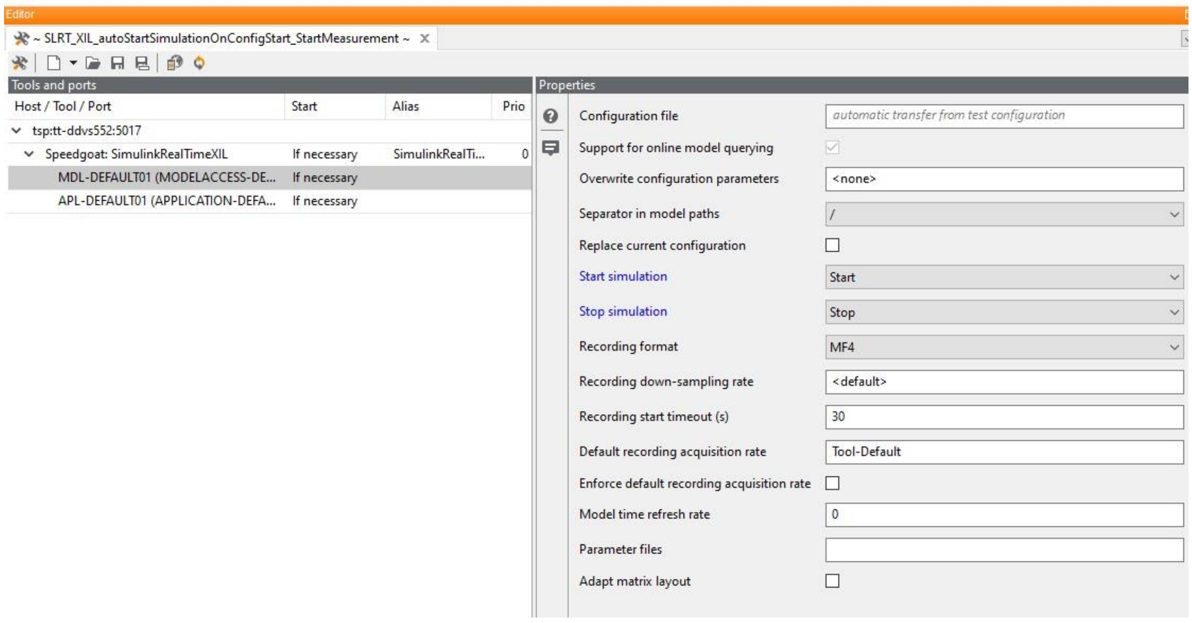
2.1 Open ecu.test. Select File > New > Test Bench Configuration. Add the tool "Speedgoat: SimulinkRealTimeXIL" and select the installed XIL-API Server from MathWorks® as shown below:


Editor
~ SLRT_XIL_autoStartSimulationOnConfigStart_StartMeasurement ~
Tools and ports
Properties

Host / Tool / Port	Start	Alias	Prio
tsp:tt-ddvs552:5017			
Speedgoat: SimulinkRealTimeXIL	If necessary	SimulinkRealTi...	0
MDL-DEFAULT01 (MODELACCESS-DE...	If necessary		
APL-DEFAULT01 (APPLICATION-DEFA...	If necessary		

XIL-API Server	MathWorks; XIL API <latest>
Model directory	
Use .NET Runtime configuration	
Timeout for tool call (in sec.)	600

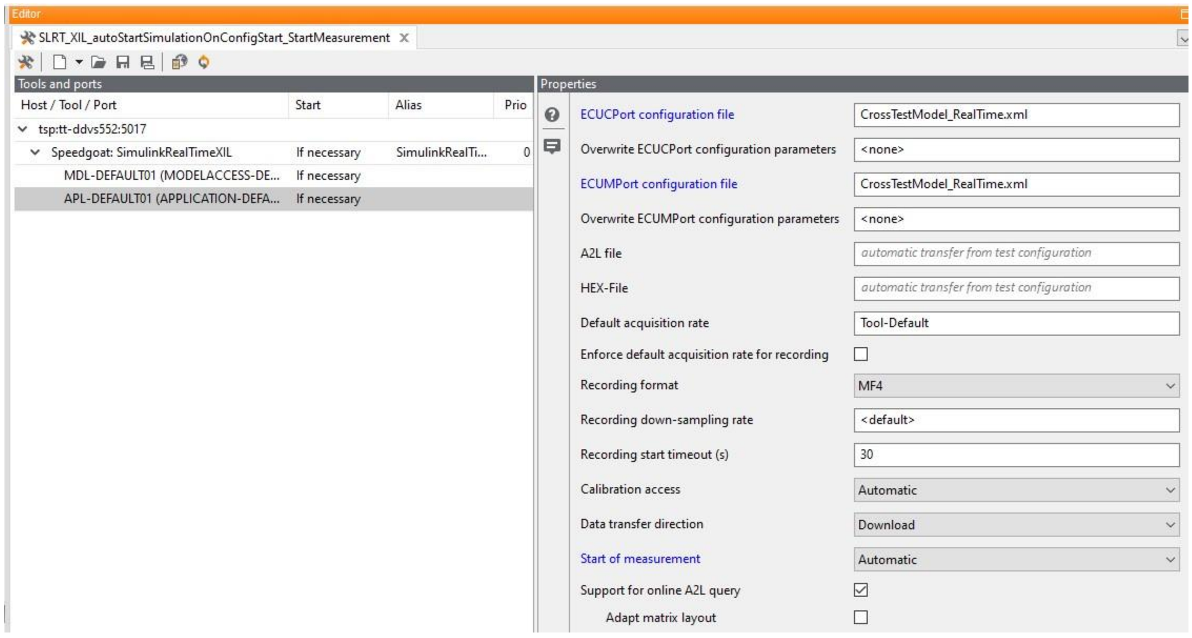
2.2 Create a Model Access Port for the test bench. Right-click Speedgoat: SimulinkRealTimeXIL and New Port -> Create Port -> Model Access Port. Edit the 'Properties' as shown below:


Editor
~ SLRT_XIL_autoStartSimulationOnConfigStart_StartMeasurement ~
Tools and ports
Properties

Host / Tool / Port	Start	Alias	Prio
tsp:tt-ddvs552:5017			
Speedgoat: SimulinkRealTimeXIL	If necessary	SimulinkRealTi...	0
MDL-DEFAULT01 (MODELACCESS-DE...	If necessary		
APL-DEFAULT01 (APPLICATION-DEFA...	If necessary		

Configuration file	*automatic transfer from test configuration*
Support for online model querying	✓
Overwrite configuration parameters	<none>
Separator in model paths	/
Replace current configuration	☐
Start simulation	Start
Stop simulation	Stop
Recording format	MF4
Recording down-sampling rate	<default>
Recording start timeout (s)	30
Default recording acquisition rate	Tool-Default
Enforce default recording acquisition rate	☐
Model time refresh rate	0
Parameter files	
Adapt matrix layout	☐

2.3 Create an Application Port for the test bench. Right-click Speedgoat: SimulinkRealTimeXIL and New Port -> Create Port -> Application Port. Edit the 'Properties' as shown below:



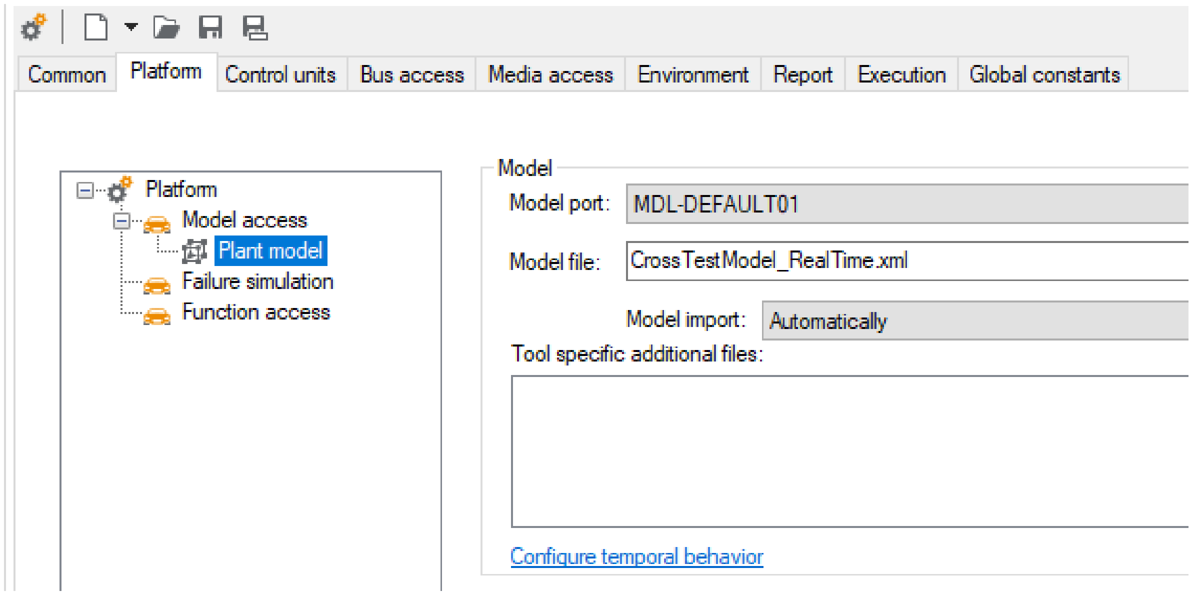
Host / Tool / Port	Start	Alias	Prio
tsp:tt-ddvs552:5017			
Speedgoat: SimulinkRealTimeXIL	If necessary	SimulinkRealTi...	0
MDL-DEFAULT01 (MODELACCESS-DE...	If necessary		
APL-DEFAULT01 (APPLICATION-DEFA...	If necessary		

ECUCPort configuration file	CrossTestModel_RealTime.xml
Overwrite ECUCPort configuration parameters	<none>
ECUMPort configuration file	CrossTestModel_RealTime.xml
Overwrite ECUMPort configuration parameters	<none>
A2L file	automatic transfer from test configuration
HEX-File	automatic transfer from test configuration
Default acquisition rate	Tool-Default
Enforce default acquisition rate for recording	☐
Recording format	MF4
Recording down-sampling rate	<default>
Recording start timeout (s)	30
Calibration access	Automatic
Data transfer direction	Download
Start of measurement	Automatic
Support for online A2L query	☑
Adapt matrix layout	☐

Note: The configuration files must point to the created config file from createPortConfigureFile.
The .xml files under ECUCPort configuration file and ECUMPort configuration file are the port configuration file created using createPortConfigureFile.

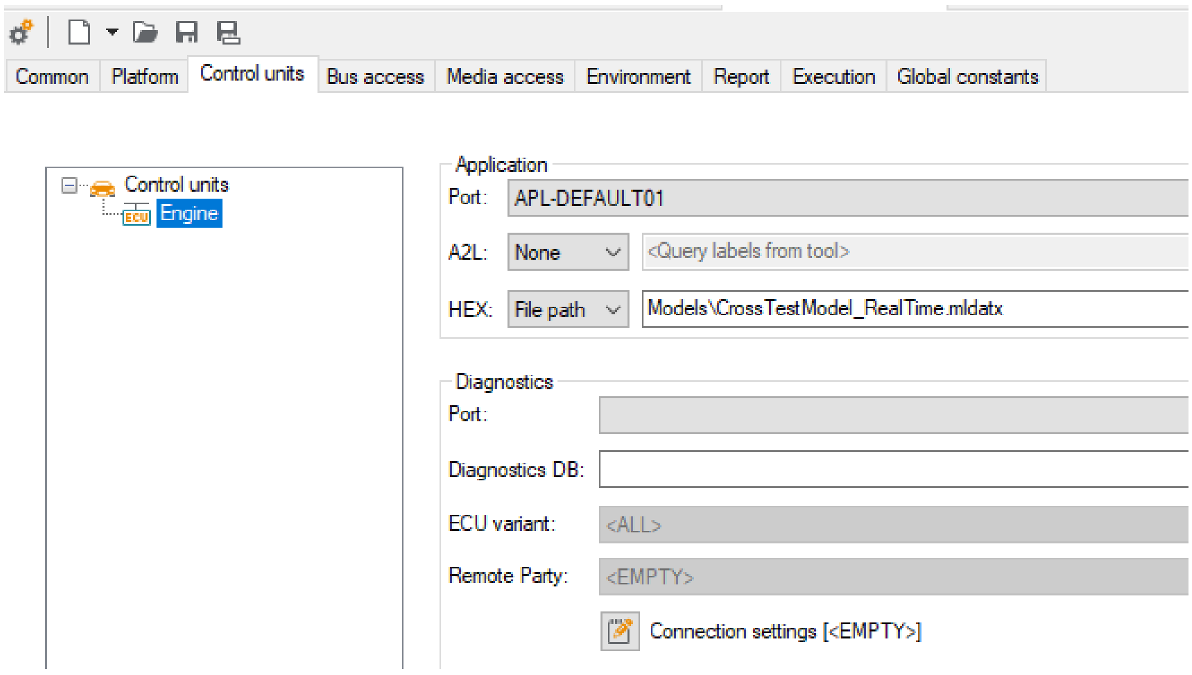
3 Create a Test Configuration

3.1 Under the 'Platform' tab, select 'Model access' and add a new model named 'Plant model'. The 'Model file' for the 'Model port' must be set to the created .xml file from createPortConfigureFile.



Model port: MDL-DEFAULT01
Model file: CrossTestModel_RealTime.xml
Model import: Automatically

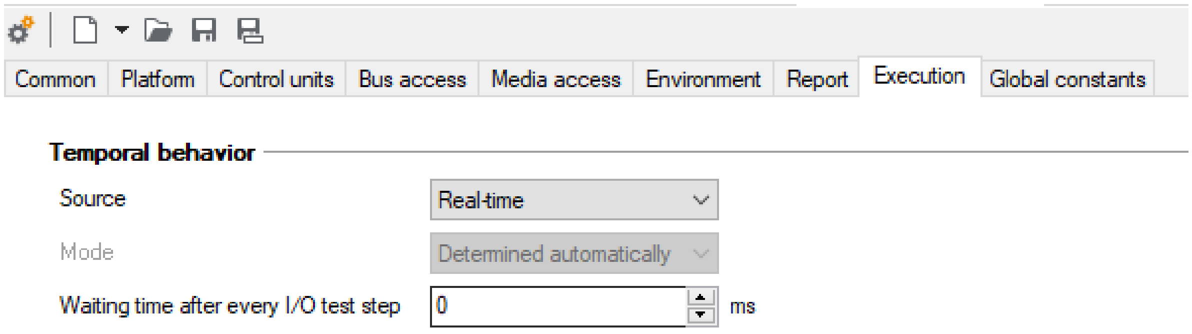
3.2 Under the 'Control units' tab, add a new control unit named 'Engine'. The HEX file must be set to the MLDATX file from the real-time application.



Port: APL-DEFAULT01
A2L: None <Query labels from tool>
HEX: File path Models\CrossTestModel_RealTime.mldatx

Port:
Diagnostics DB:
ECU variant: <ALL>
Remote Party: <EMPTY>

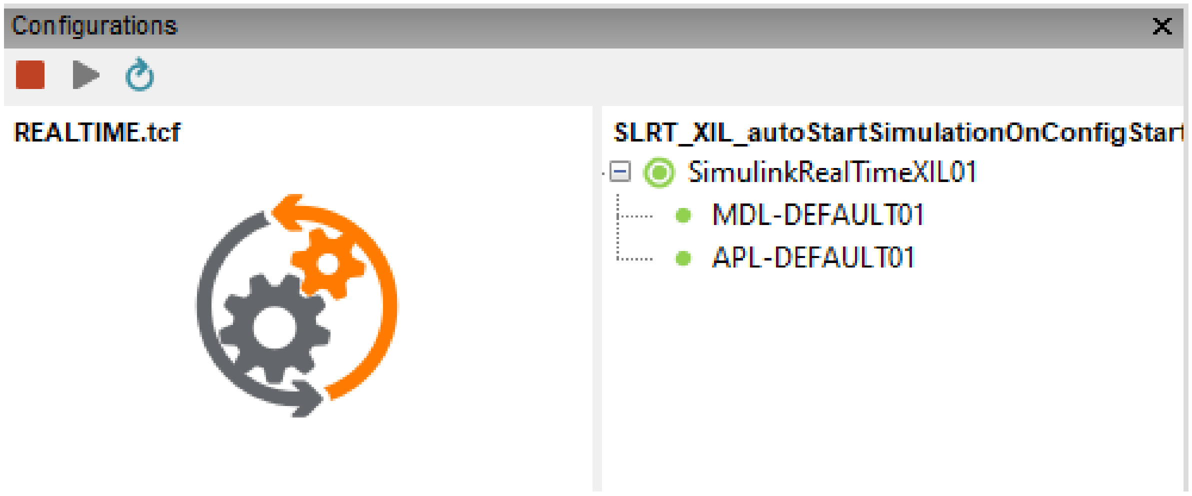
3.3 Under the ‘Execution’ tab, the temporal behavior must be set to ‘Real-time’.



Temporal behavior

Source: Real-time
Mode: Determined automatically
Waiting time after every I/O test step: 0 ms

3.4 Select and load the configurations.



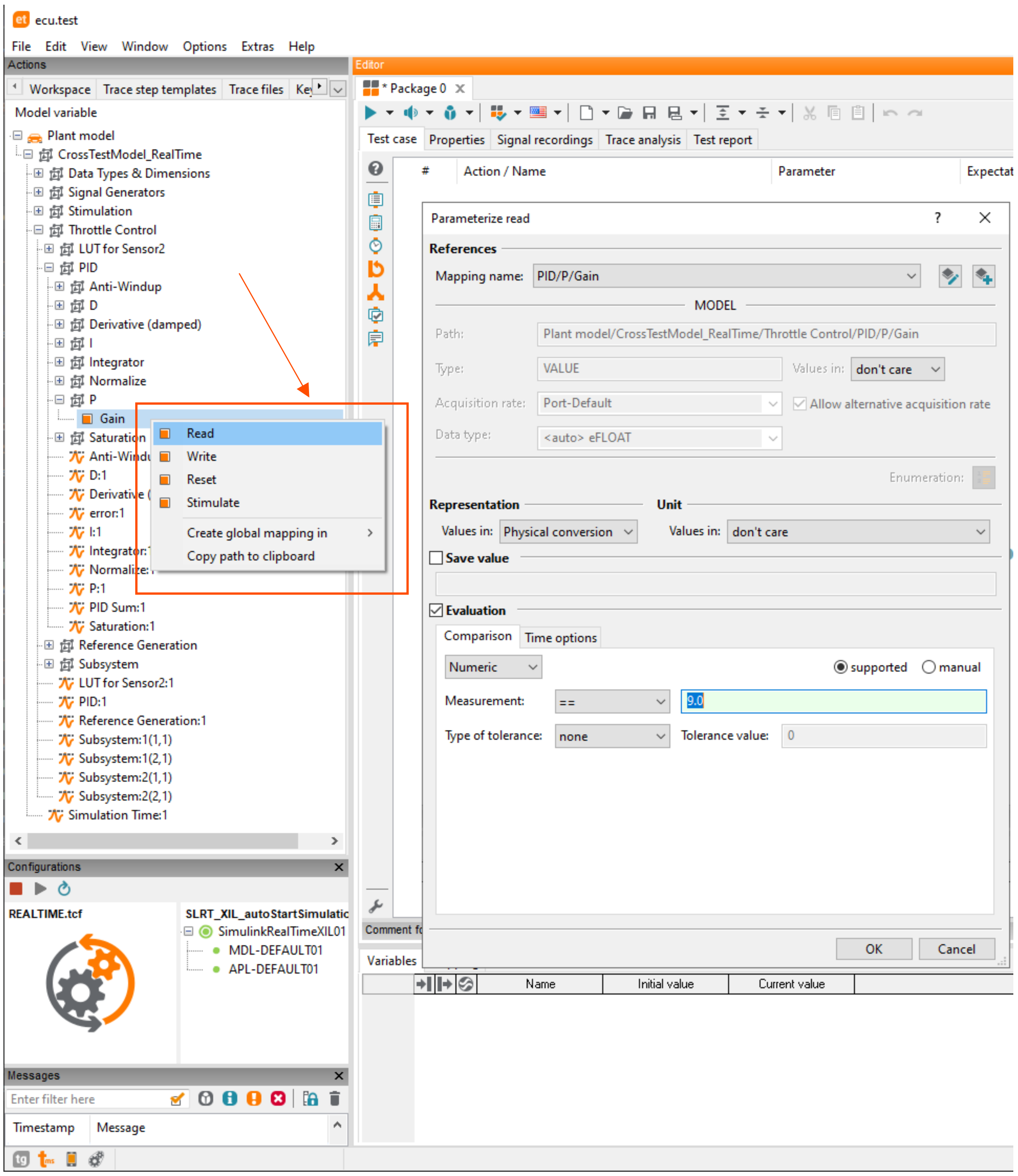
REALTIME.tcf

SLRT_XIL_autoStartSimulationOnConfigStart
SimulinkRealTimeXIL01
MDL-DEFAULT01
APL-DEFAULT01

4 Create and Run Packages

In ecu.test, packages refer to the collection of test cases, test configurations, and related resources organized for efficient management and execution of test activities.

4.1 Create a new package and add the ‘Read’, ‘Write’, and ‘Stimulate’ steps from the model access tab to interact with the model.



References

Mapping name: PID/P/Gain

MODEL

Path: Plant model/CrossTestModel_RealTime/Throttle Control/PID/P/Gain

Type: VALUE   Values in: don't care

Acquisition rate: Port-Default   ☑ Allow alternative acquisition rate

Data type: <auto> eFLOAT

Enumeration:

Representation

Values in: Physical conversion

Unit

Values in: don't care

☐ **Save value**

☑ **Evaluation**

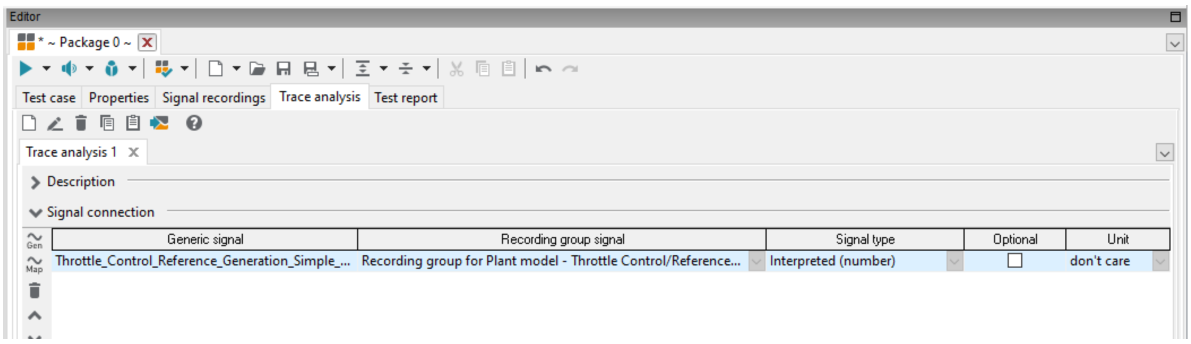
Comparison | Time options

Numeric   [✓ supported] manual

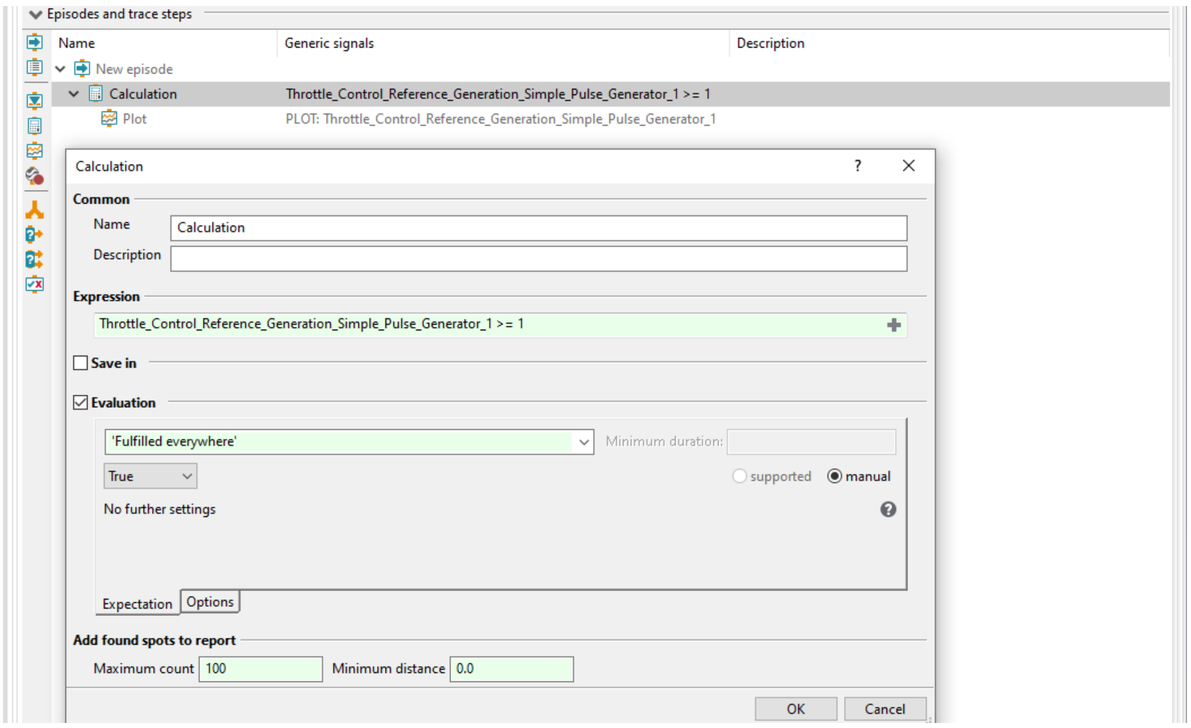
Measurement: ==   9.0

Type of tolerance: none   Tolerance value: 0

4.2 Add trace analyses and plots to check the behavior of signals over a specific period or the whole time.



Generic signal	Recording group signal	Signal type	Optional	Unit
Throttle_Control_Reference_Generation_Simple_...	Recording group for Plant model - Throttle Control/Reference...	Interpreted (number)	☐	don't care



Name	Generic signals	Description
New episode		
Calculation	Throttle_Control_Reference_Generation_Simple_Pulse_Generator_1 >= 1	
Plot	PLOT: Throttle_Control_Reference_Generation_Simple_Pulse_Generator_1	

Common
Name: Calculation
Description:

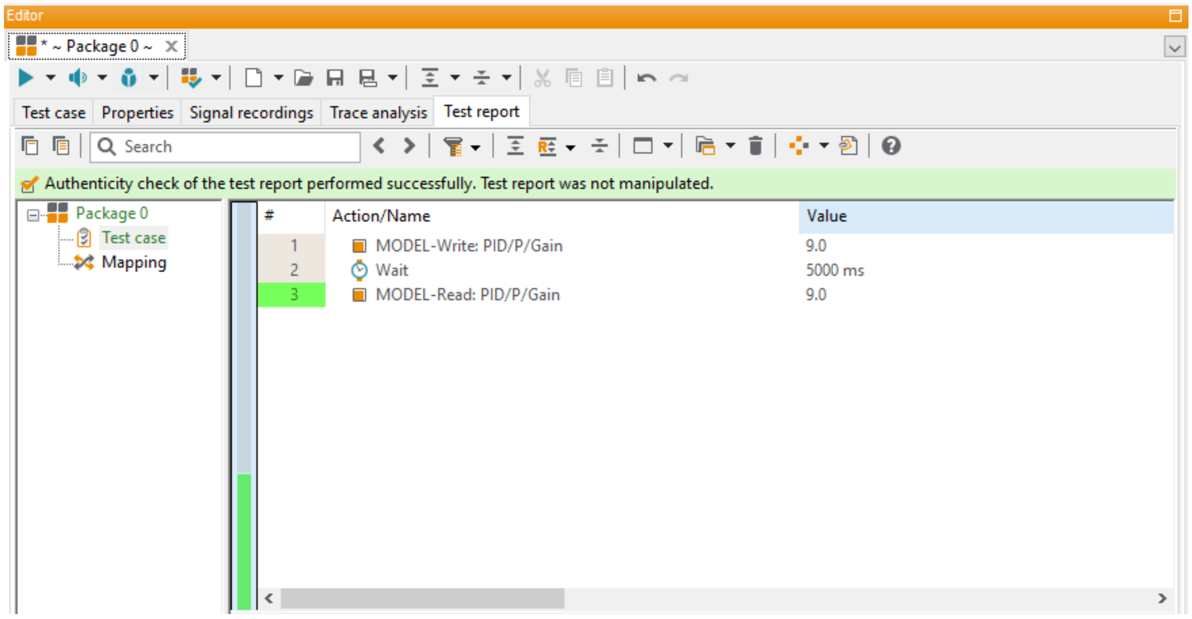
Expression
Throttle_Control_Reference_Generation_Simple_Pulse_Generator_1 >= 1

☐ **Save in**

☑ **Evaluation**
'Fulfilled everywhere' Minimum duration:
True supported [✓ manual]
No further settings
Expectation | Options

Add found spots to report
Maximum count: 100 Minimum distance: 0.0

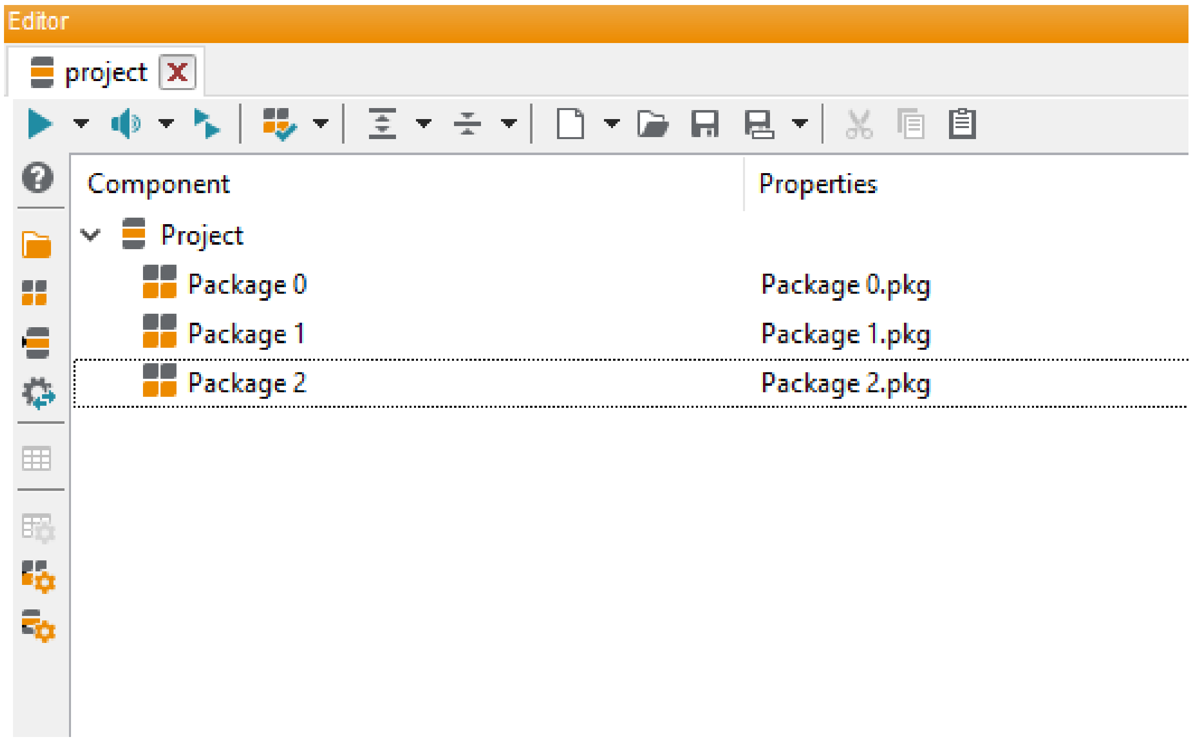
4.3 Run the package and check the report.



Authenticity check of the test report performed successfully. Test report was not manipulated.

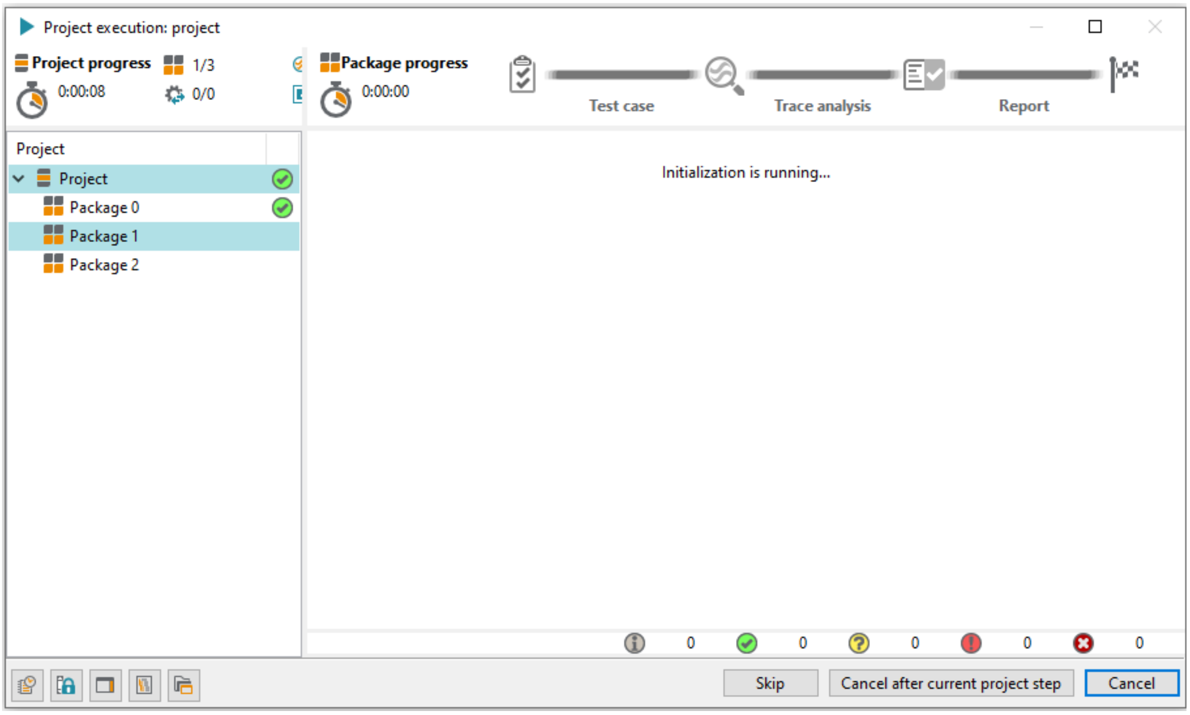
#	Action/Name	Value
1	MODEL-Write: PID/P/Gain	9.0
2	Wait	5000 ms
3	MODEL-Read: PID/P/Gain	9.0

4.4 To execute several packages, create a new project and drag and drop the packages.



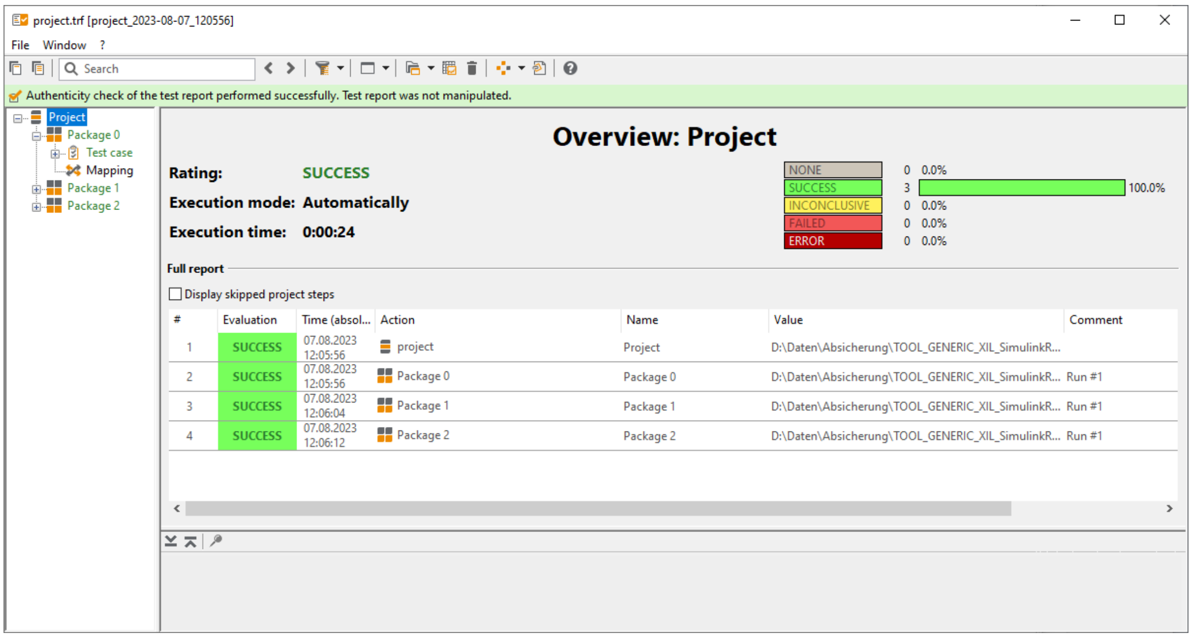
Component	Properties
Project	
Package 0	Package 0.pkg
Package 1	Package 1.pkg
Package 2	Package 2.pkg

4.5 Execute the project.



Initialization is running...

4.6 The report contains the runs of all three packages.



Authenticity check of the test report performed successfully. Test report was not manipulated.

Overview: Project

Rating: SUCCESS

Execution mode: Automatically

Execution time: 0:00:24

NONE	0	0.0%
SUCCESS	3	100.0%
INCONCLUSIVE	0	0.0%
FAILED	0	0.0%
ERROR	0	0.0%

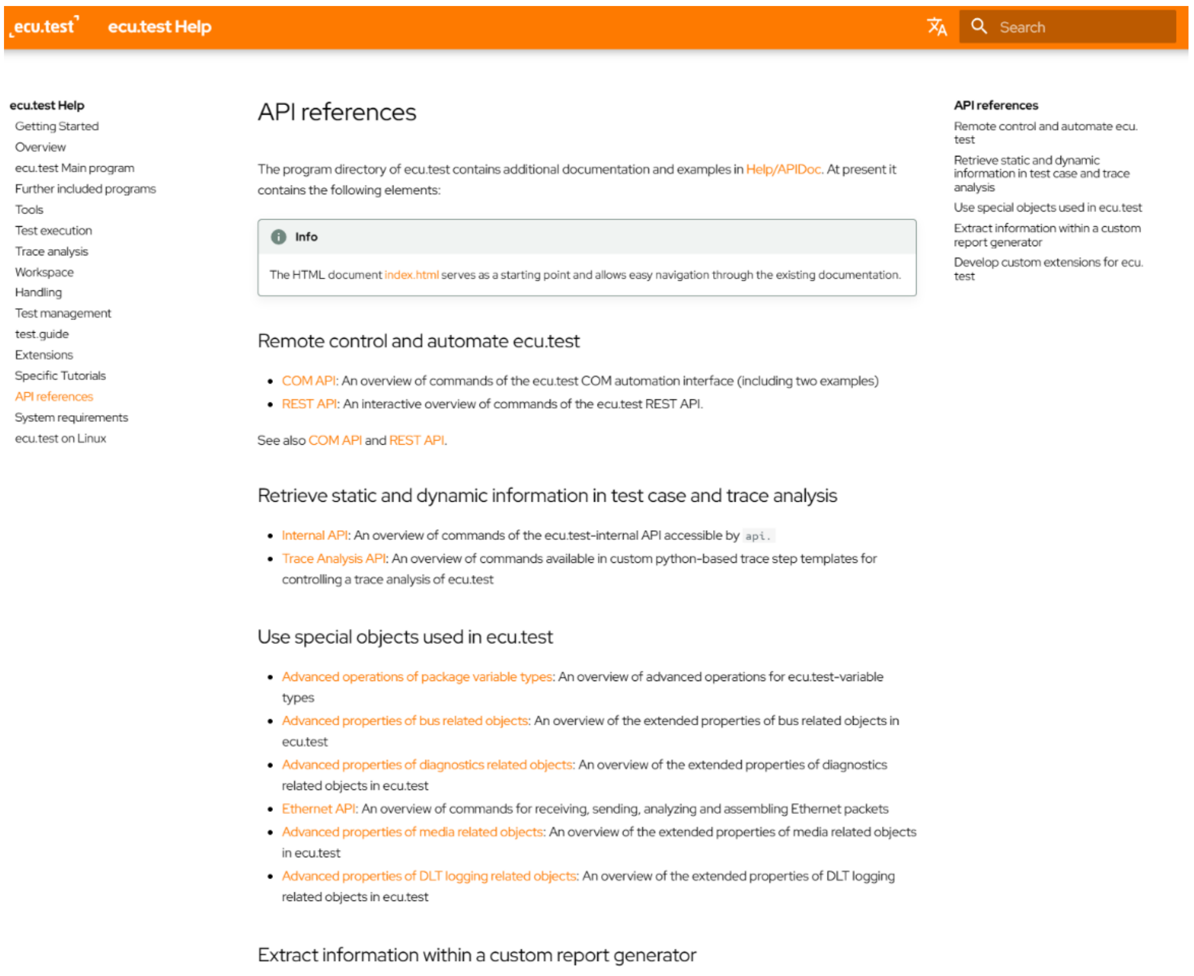
Full report

☐ Display skipped project steps

#	Evaluation	Time (absol...	Action	Name	Value	Comment
1	SUCCESS	07.08.2023 12:05:56	project	Project	D:\Daten\Absicherung\TOOL_GENERIC_XIL_SimulinkR...	
2	SUCCESS	07.08.2023 12:05:56	Package 0	Package 0	D:\Daten\Absicherung\TOOL_GENERIC_XIL_SimulinkR...	Run #1
3	SUCCESS	07.08.2023 12:06:04	Package 1	Package 1	D:\Daten\Absicherung\TOOL_GENERIC_XIL_SimulinkR...	Run #1
4	SUCCESS	07.08.2023 12:06:12	Package 2	Package 2	D:\Daten\Absicherung\TOOL_GENERIC_XIL_SimulinkR...	Run #1

5 Test Automation APIs

5.1 Check out API references for the APIs required for test automation, such as REST and COM. All information can be found in the help.



ecu.test ecu.test Help Search

ecu.test Help
Getting Started
Overview
ecu.test Main program
Further included programs
Tools
Test execution
Trace analysis
Workspace
Handling
Test management
test.guide
Extensions
Specific Tutorials
API references
System requirements
ecu.test on Linux

API references

The program directory of ecu.test contains additional documentation and examples in Help/APIDoc. At present it contains the following elements:

Info

The HTML document index.html serves as a starting point and allows easy navigation through the existing documentation.

Remote control and automate ecu.test

- COM API: An overview of commands of the ecu.test COM automation interface (including two examples)
- REST API: An interactive overview of commands of the ecu.test REST API.

See also COM API and REST API.

Retrieve static and dynamic information in test case and trace analysis

- Internal API: An overview of commands of the ecu.test-internal API accessible by `api.`
- Trace Analysis API: An overview of commands available in custom python-based trace step templates for controlling a trace analysis of ecu.test

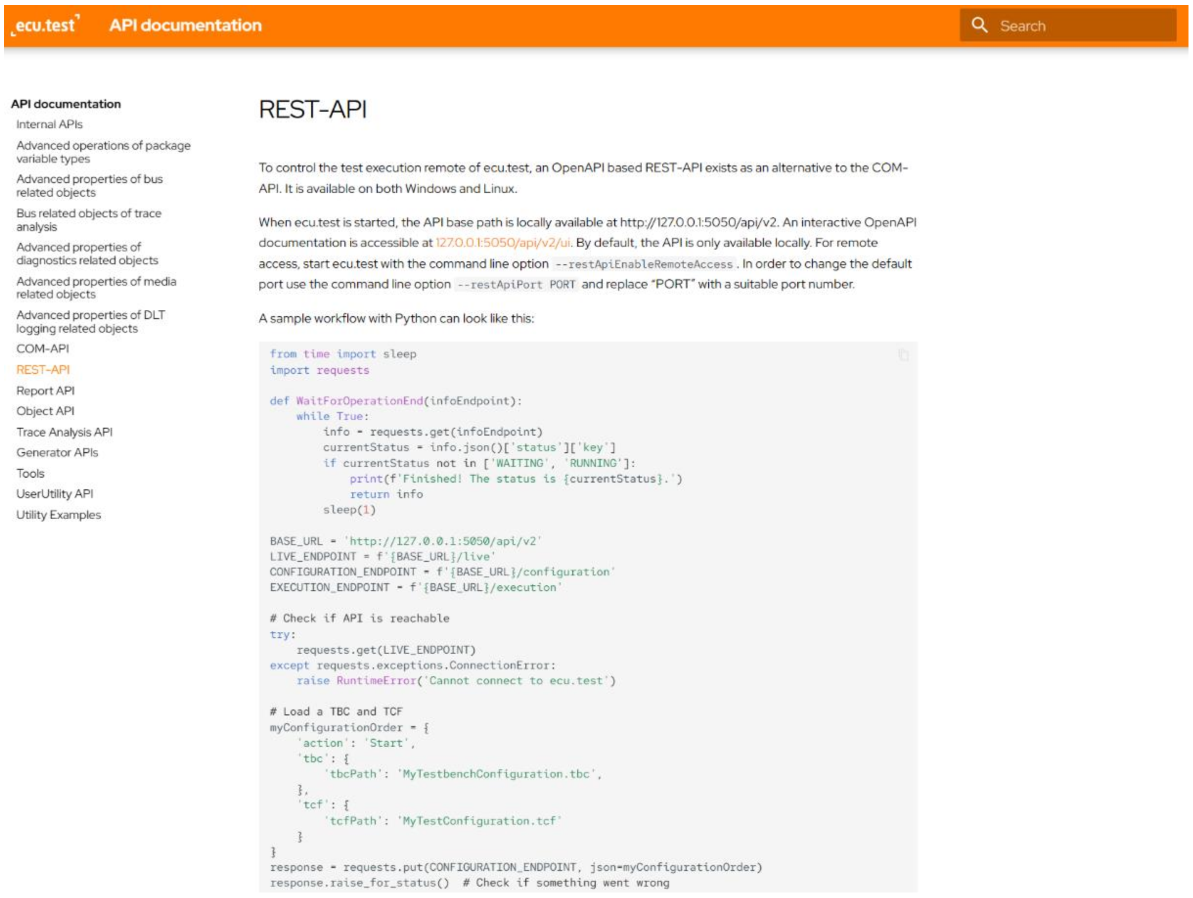
Use special objects used in ecu.test

- Advanced operations of package variable types: An overview of advanced operations for ecu.test-variable types
- Advanced properties of bus related objects: An overview of the extended properties of bus related objects in ecu.test
- Advanced properties of diagnostics related objects: An overview of the extended properties of diagnostics related objects in ecu.test
- Ethernet API: An overview of commands for receiving, sending, analyzing and assembling Ethernet packets
- Advanced properties of media related objects: An overview of the extended properties of media related objects in ecu.test
- Advanced properties of DLT logging related objects: An overview of the extended properties of DLT logging related objects in ecu.test

Extract information within a custom report generator

API references
Remote control and automate ecu.test
Retrieve static and dynamic information in test case and trace analysis
Use special objects used in ecu.test
Extract information within a custom report generator
Develop custom extensions for ecu.test

5.2 Check out this example of using the REST API.



API documentation
- Internal APIs
- Advanced operations of package variable types
- Advanced properties of bus related objects
- Bus related objects of trace analysis
- Advanced properties of diagnostics related objects
- Advanced properties of media related objects
- Advanced properties of DLT logging related objects
- COM-API
- REST-API
- Report API
- Object API
- Trace Analysis API
- Generator APIs
- Tools
- UserUtility API
- Utility Examples

REST-API

To control the test execution remote of ecu.test, an OpenAPI based REST-API exists as an alternative to the COM-API. It is available on both Windows and Linux.

When ecu.test is started, the API base path is locally available at http://127.0.0.1:5050/api/v2. An interactive OpenAPI documentation is accessible at 127.0.0.1:5050/api/v2/ui. By default, the API is only available locally. For remote access, start ecu.test with the command line option `--restApiEnableRemoteAccess`. In order to change the default port use the command line option `--restApiPort PORT` and replace "PORT" with a suitable port number.

A sample workflow with Python can look like this:

```python
from time import sleep
import requests

def WaitForOperationEnd(infoEndpoint):
    while True:
        info = requests.get(infoEndpoint)
        currentStatus = info.json()['status']['key']
        if currentStatus not in ['WAITING', 'RUNNING']:
            print(f'Finished! The status is {currentStatus}.')
            return info
        sleep(1)

BASE_URL = 'http://127.0.0.1:5050/api/v2'
LIVE_ENDPOINT = f'{BASE_URL}/live'
CONFIGURATION_ENDPOINT = f'{BASE_URL}/configuration'
EXECUTION_ENDPOINT = f'{BASE_URL}/execution'

# Check if API is reachable
try:
    requests.get(LIVE_ENDPOINT)
except requests.exceptions.ConnectionError:
    raise RuntimeError('Cannot connect to ecu.test')

# Load a TBC and TCF
myConfigurationOrder = {
    'action': 'Start',
    'tbc': {
        'tbcPath': 'MyTestbenchConfiguration.tbc',
    },
    'tcf': {
        'tcfPath': 'MyTestConfiguration.tcf'
    }
}
response = requests.put(CONFIGURATION_ENDPOINT, json=myConfigurationOrder)
response.raise_for_status()  # Check if something went wrong
```

5.3 Use the test.guide to handle the rising number of reports. test.guide also offers many automation features, such as executing projects with an intelligent distribution of available resources out of the box. See here: https://www.tracetronic.com/products/test-guide/