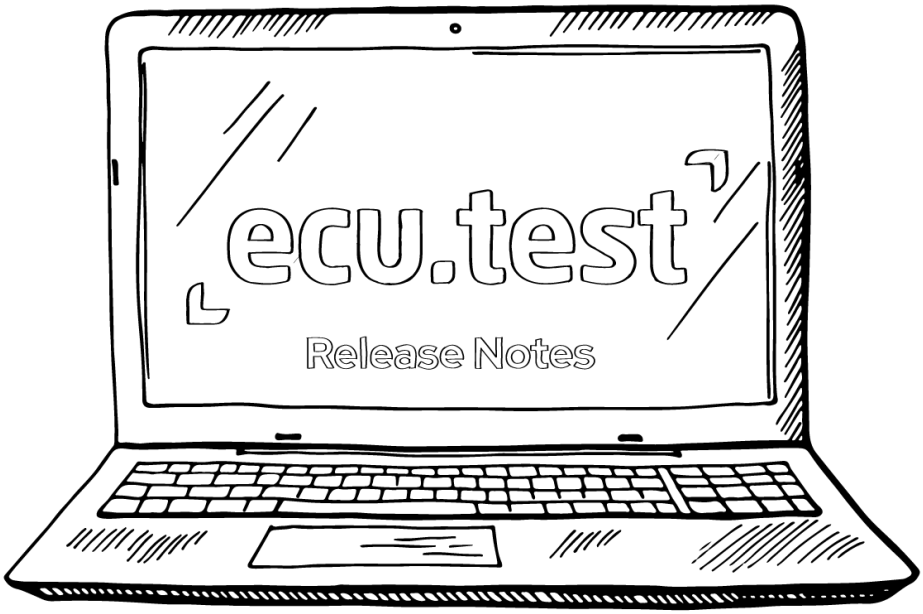
tracetronic



Release Notes

ecu.test 2025.2

trace.check 2025.2

Date: 05/27/2025

tracetronic GmbH
Stuttgarter Str. 3
01189 Dresden
www.tracetronic.com

tracetronic

Content

Overview

With **ecu.test** 2025.2, we are launching a new product extension: **ecu.test** *lab*. This add-on can be used platform-independently to control and visualize test benches via a web interface with no need for in-depth expertise or complex test environments. It is designed for use in the early development phases and exploratory testing in SiL environments, which can be tracked in real time.
For more information, please take a look at the user documentation.

Further highlights of this release include:

- **Online user documentation**
  The product documentations are now also available online
- **Scenario and test case workflows**
  Closer integration with CarMaker environments for comprehensive scenario and test case development and analysis
- **OBDonUDS**
  Full support of the OBDonUDS standard
- **Open with ecu.test diff**
  Browser extension to compare changes from GitHub or GitLab directly in **ecu.test** with **ecu.test** Diff
- **Library workspaces**
  Support for TBC and TCF configuration files
- **Test case coverage**
  New function for creating coverages
- **SOME/IP extension**
  Selection of specific consumers in service communication

It's definitely worth reading on!
In addition to our highlights, this release contains a wide range of other features and improvements covering all key areas: from new **diagnostic** and **communication** options to **API** enhancements and smart innovations in **multimedia**, **SiL** and **HiL** setups, as well as in **trace analysis**.
With numerous small but effective usability optimizations, we ensure that you can test even more efficiently and that ecu.test is as useful as possible in your everyday work.

And last but not least: Look forward to our sneak preview!

Note: Icons are used to indicate for which product a topic is relevant:
ecu.test **trace.check.**

1 Highlights in ecu.test 2025.2

ecu.test *lab* increases the accessibility of complex test benches

ecu.test *lab* is a web interface provided by **ecu.test** to visualize and control the behavior of a test bench, as well as the system under test.

Whether Windows or Linux, local or remote, CAN, XCP, model or multimedia - **ecu.test** *lab* runs where you need it and communicates via all interfaces available in **ecu.test** directly based on existing mappings or packages.

Try it now!

Every **ecu.test** license includes access to the full range of functions offered by ecu.test lab until the end of 2025.

ecu.test *lab* can be configured and started from within **ecu.test**.
It can be accessed from the web browser of your choice on the same machine or remotely via the network.

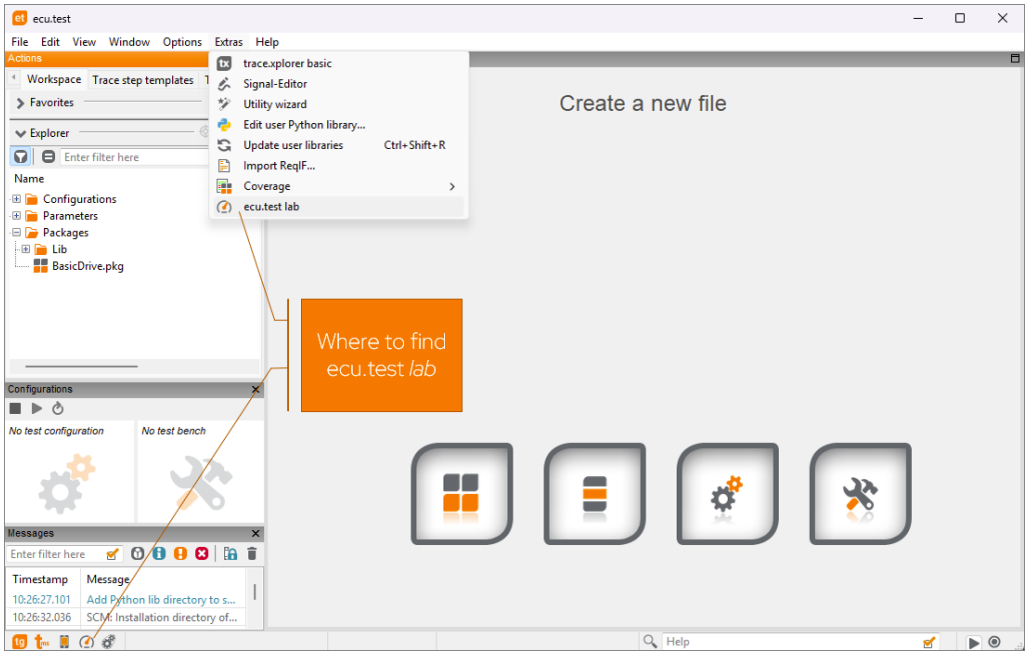


Figure 1: Starting ecu.test lab in the GUI

The user interface can be configured depending on personal needs of the user or user group. Therefore, it provides different widgets to display information or to allow manipulation of values and states of the test bench.

The widgets are always linked to **ecu.test** - mappings - packages or - expressions. For value access, **ecu.test** performs either direct tool access, package execution or an expression evaluation.

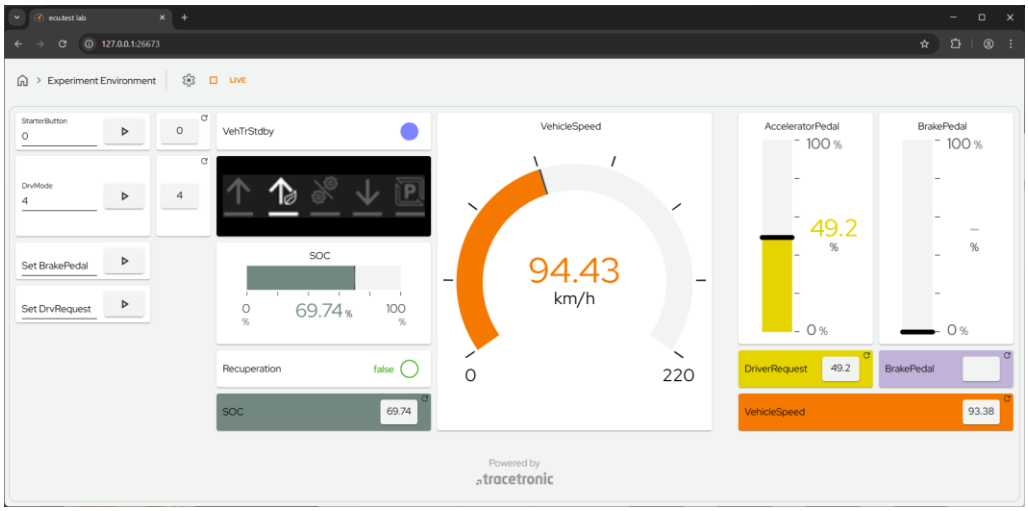


Figure 2: User-defined view of ecu.test lab

Note: More details about **ecu.test** *lab* are in the **user documentation**.

Online version of the user documentations

The **ecu.test** and **trace.check** user documentations are now also available online. Starting with version 2025.2, all documentation components from the last four releases will be available.

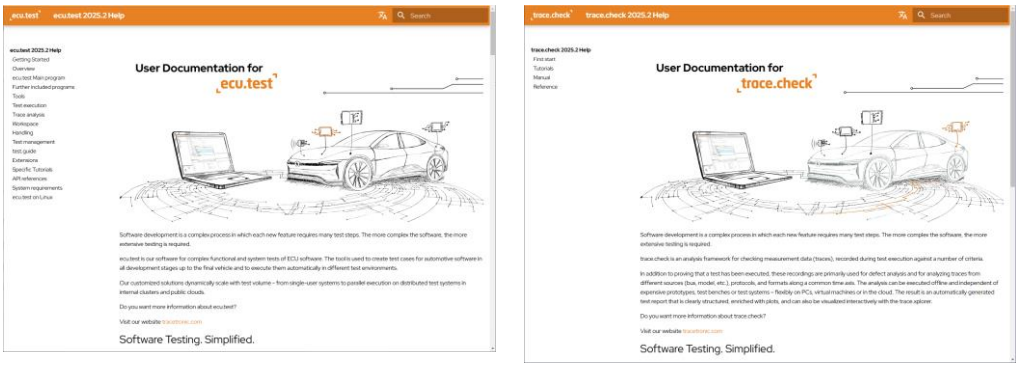


Figure 3: Online user documentation of ecu.test and trace.check

In the future, the online documentations will also be linked directly to **ecu.test/trace.check** so that the latest version is always used.
You can now also take a quick look at the documentations without having to install it first.

In addition, the file structure of the offline documentations has been modernized. Among other things, file names have been standardized (now English only) and minor structural adjustments have been made.

Note: Links within the help have changed as a result.

Scenario and test case workflow with ecu.test and scenario.architect in CarMaker environments

et

One of the biggest challenges in scenario-based testing is the interaction between test cases and scenarios. With **ecu.test** 2025.2, we offer a comprehensive workflow for CarMaker that allows you to create and adapt both scenarios and test cases and execute them in your test environment.

The following graphic illustrates the interaction between **ecu.test** and the **scenario.architect**.

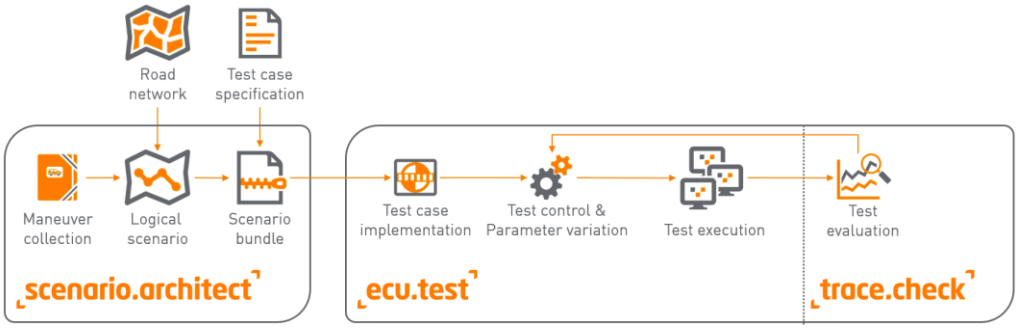


Figure 4: Schematic illustration of the interaction between ecu.test and scenario.architect

In **ecu.test**, new scenarios can be created and existing ones adapted. The scenario tree in the **environment simulation** tab provides new context menu entries for this purpose.

When creating or modifying scenarios, **scenario.architect** opens. In this tool, you can make adjustments or completely redesign scenarios. Once completed, **ecu.test** takes over artifact handling and ensures that the necessary scenarios are forwarded directly to the environment simulation. The corresponding scenarios are loaded and executed via the test cases.

The added value comes from the coupling of the toolings. Through this integration, scenarios, test cases, and analyses can be developed and executed very quickly, intuitively, and iteratively.

Figure 5: Interaction between ecu.test and scenario.architect

Further details can be found in the tutorial in the user documentation.

Note: The workflow is designed for use with OpenSCENARIO files. The functions are currently available for VTD and now also for CarMaker environment simulations.

Full support for OBDonUDS

ecu.test *diagnostics* now offers full support for the OBDonUDS standard. Highlights are the symbolic diagnostic service test steps, which you can simply drag and drop into the test case.

Simply activate the OBDonUDS option in the TCF **diagnostic** settings. After configuration is started, the test steps appear in the **Diagnostics** tab – natively and without the need for a diagnostic database.

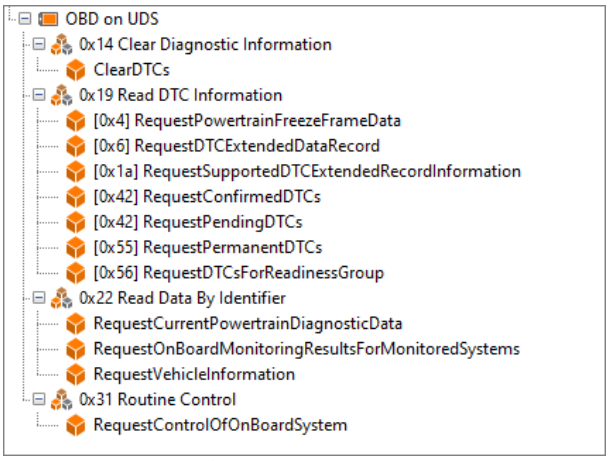


Figure 6: Full support for the OBDonUDS standard

The services are available for all diagnostic ports.

- IsoTp over CAN
- FrTp over FlexRay
- DoIP over Ethernet

Library workspaces: Support for configuration files (TBC/TCF)

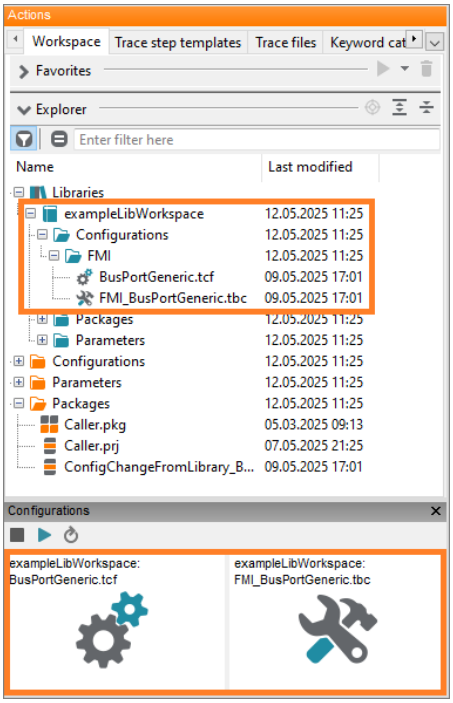


Figure 7: TBC and TCF files in the Workspace Explorer and the configuration window

Testbench configuration and test configuration files (TCF and TBC files) can now be managed centrally and are displayed in the **Workspace** Explorer within the configuration directory.

These files can be opened and edited by double-clicking them in the editor or by drag-and-dropping them into the configuration window.

In the **Change configuration** project step, the selection of TBC and TCF files has also been extended: In the dropdown menu of the respective file, library files can now be selected in addition to local files, which makes the configuration much more flexible. When loading the configuration, the library references are automatically resolved.

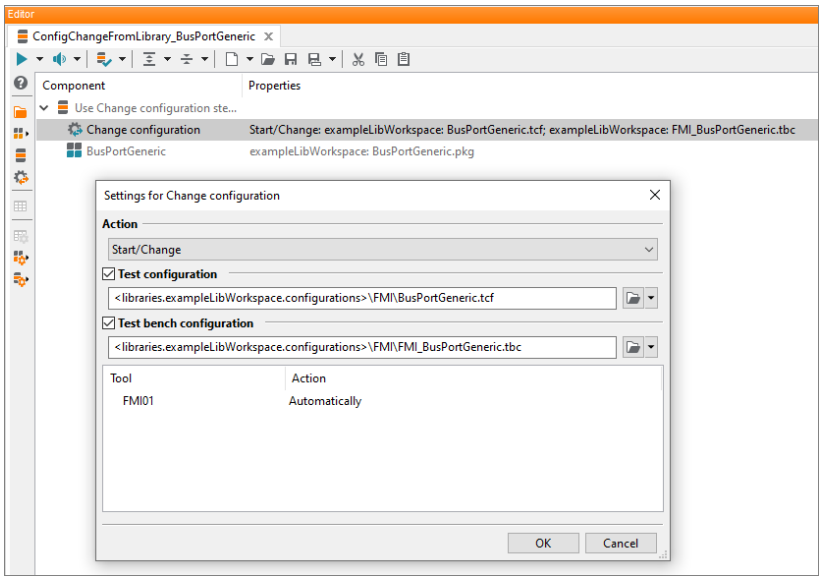


Figure 8: Selection of TBC and TCF files in the change configuration project step

To better distinguish between local and library configuration files, the icons have also been adapted according to the usual scheme and are now displayed in orange and teal, respectively.

Note: The Object API provides new methods for use in the configuration change project step. The existing tutorial has been extended accordingly.

Creation of test case coverage reports

et

Since **ecu.test** 2024.3, coverage metrics can be generated for test cases in order to systematically analyze the use and validation of packages as part of quality management. This function is particularly useful for identifying unused sections and safeguarding safety-critical libraries - similar to code coverage analyses in software development.

With the current release, this function has now been expanded: The coverage data of several test executions can be consolidated and displayed across the workspace in a clear report. This means there is no longer any need to tediously track coverage information through individual packages in the GUI.

The compact HTML and JSON reports provide a centralized view of test coverage, test gaps and improvement potential - as a basis for sound test planning, targeted maintenance and consistent quality assurance throughout the entire project.

The report is created via a new menu entry in the existing **Coverage** menu.

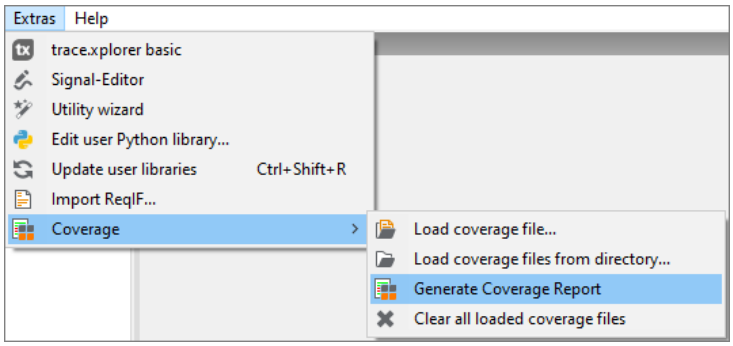


Figure 9: Coverage report generation

An HTML report is then generated and displayed directly in **ecu.test**. The links to packages are clickable and open the package directly in the tool itself.

The HTML file can also be displayed in the browser. The generated JSON file can be used for automated evaluation or data use.

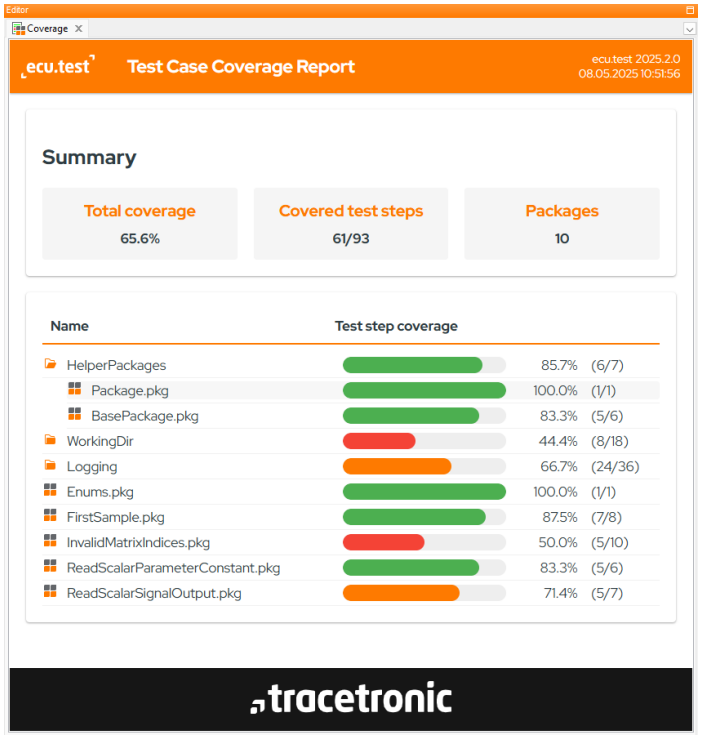


Figure 10: Created coverage overview

Browser extension for opening diffs in ecu.test

et tc

A new browser extension called **Open with ecu.test diff** is now available for Chrome, Edge, and Firefox.

As usual, it allows you to compare changes to packages and other artifacts from GitHub and GitLab directly with an installed **ecu.test**.

Note: Starting with this version, you can open the comparison without a license.

You can add the extension via your browser (Chrome+Edge, Firefox).

However, browser extensions are often managed by IT and require explicit approval. For easier verification, the extension's source code is open source on (GitHub).

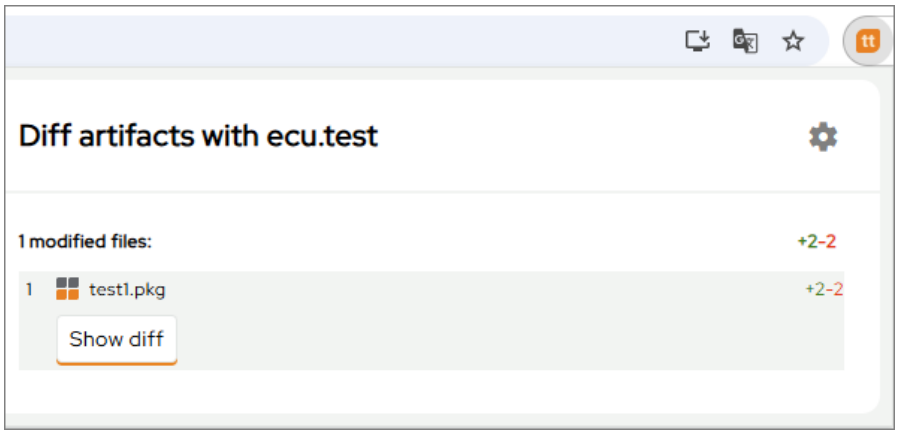


Figure 11: Using the extension in the browser

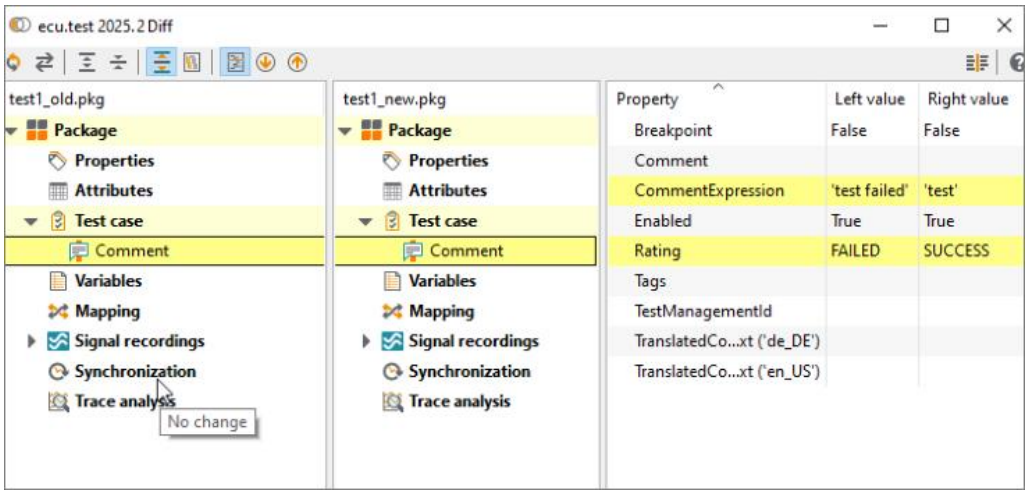


Figure 12: Compare the changes with ecu.test

Service-Communication via SOME/IP: Selection of the consumer

et

For use cases in which it matters which service consumer uses a service, it is now possible to select the consuming control unit in the mapping items of Ethernet test steps.

When a consumer is selected, the network adapter with the TCP/IP settings of the consuming control unit is automatically configured from the ARXML when the test case is started.

This feature removes the need to create and manually configure a port in the test-bench configuration for each consumer.

For existing test cases or cases where the consumer selection is left empty, the IP configuration in the TBC settings **Fallback IP Settings** continues to apply.

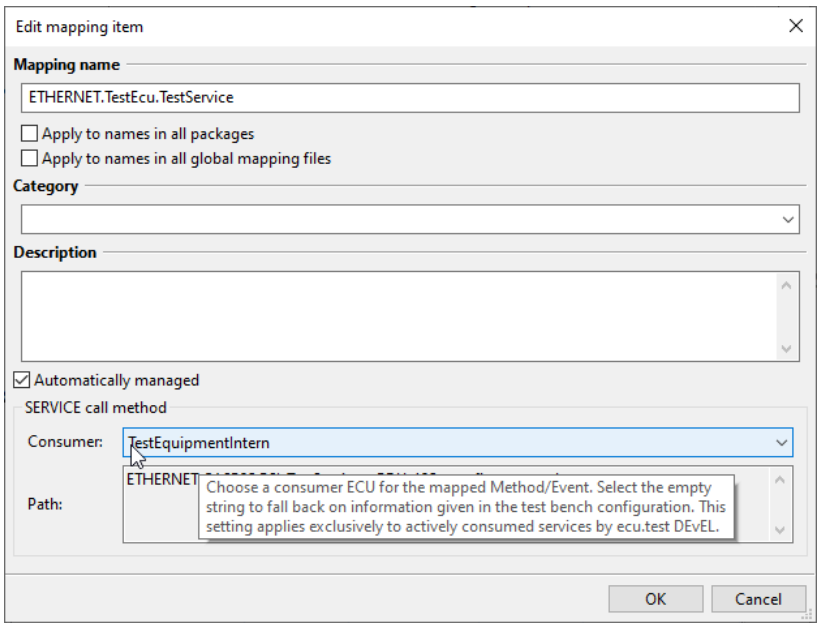


Figure 13: Selection of a consumer

The new option is only effective in conjunction with the active service port and the TCP/IP network stack integrated in **ecu.test**.

This extension also affects the connection of the ServiceManager port in dSPACE models. Here, the placeholder **consumerNode** can now also be used to define a mapping to model variables.

2 Usability

Introduction of a technical user for the vault

et tc

With the help of the vault, certificates or credentials can be stored securely in the workspace and then used in test cases. With **ecu.test** 2025.2, the vault has been expanded to include the option of a technical user – this user cannot read or edit the vault, but can only unlock it for test case execution.

The vault is created as usual with a password. Under the new **Show Tester Key** option, a generated key is listed which can be passed on to the technical user. The vault automatically recognizes whether it was opened with an admin password or a tester key and automatically restricts access.

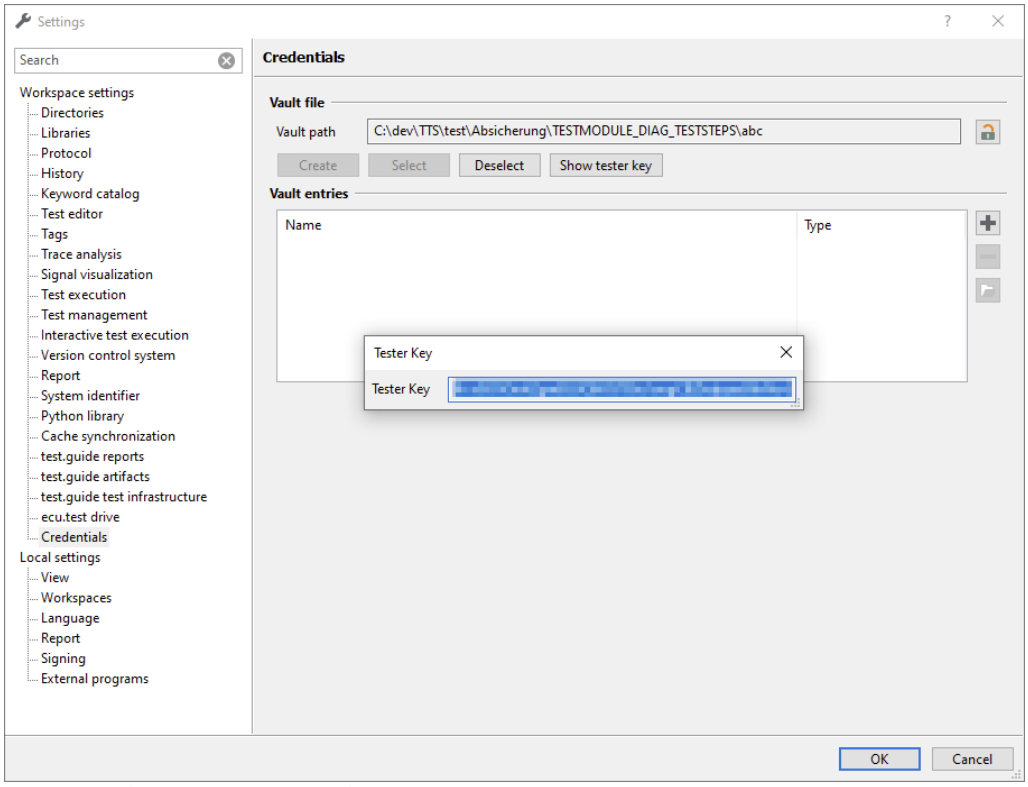


Figure 14: Defining a tester key vault for the technical user

New file cache technology

et tc

The file cache of **ecu.test** is used as a local cache for the files and metadata in the workspace. It is used in particular to avoid redundant file operations and thus accelerates the start and execution of test cases as well as the use of the GUI in general. It works automatically in the background.
Thanks to an update of the technology used and further optimization, the cache is now created up to four times faster than before, which has a significant effect, especially in large workspaces.

Copy and Paste in Testbench Configurations

et

It is now possible to copy tools (including their ports) and individual ports from one test bench configuration to another by transferring them using the copy and paste function (Ctrl-C & Ctrl-V or context menu).

The Object API has been adapted accordingly and now allows ports and tools to be reused:

- Tool.Clone
- Tool.InsertPort
- Port.Clone
- ToolHost.InsertTool

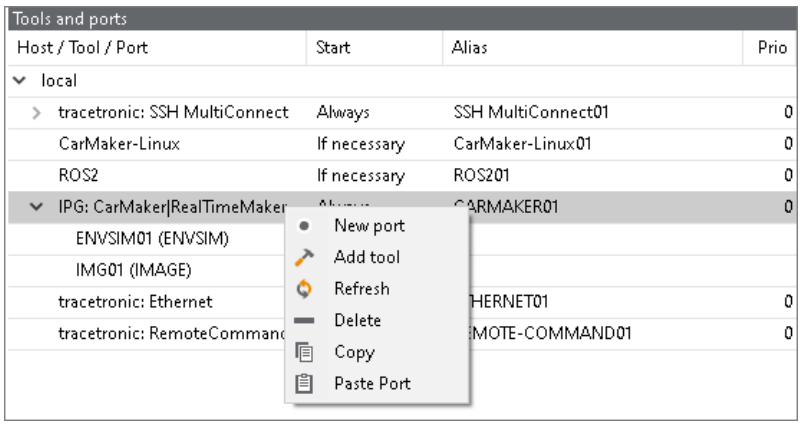


Figure 15: New copy and paste function in TBC

New attributes panel available in additional locations

In previous versions, the attribute panel for the package editor has been redesigned. The new panel now offers a clear tabular display, options for filtering attributes, and the ability to create your own lists.

With **ecu.test** 2025.2, other parts of the program have also changed. Editing project attributes and general attribute editing from the Workspace Explorer now offer the advantages of the new Attributes panel as well.

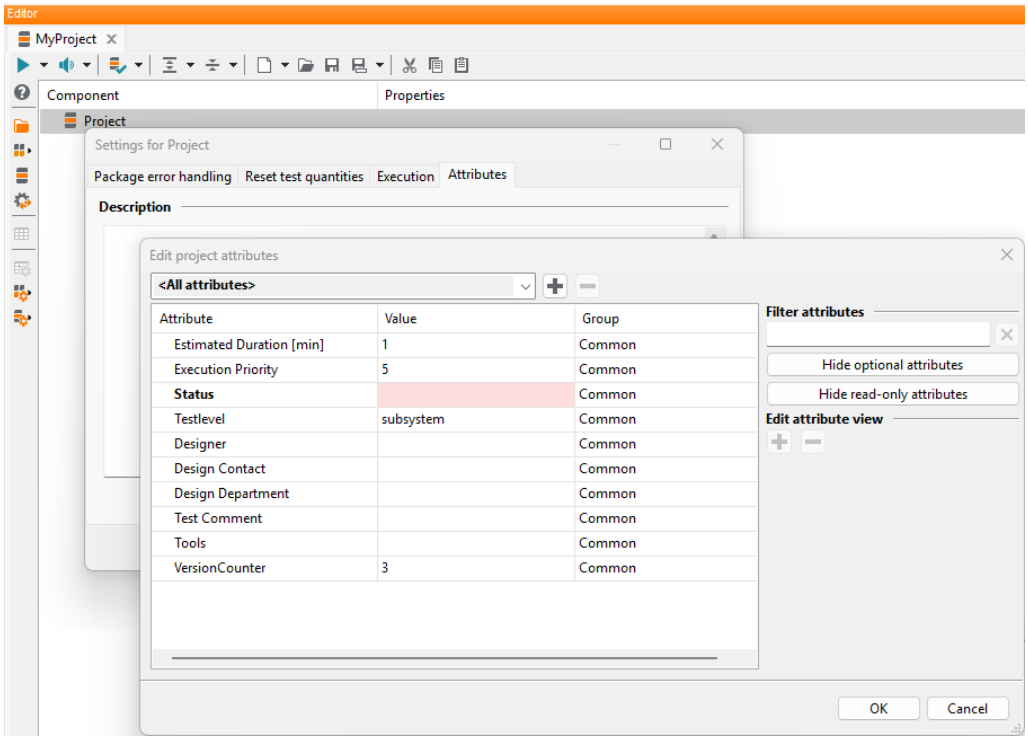


Figure 16: New attribute panel

Library workspaces: Support in favorites

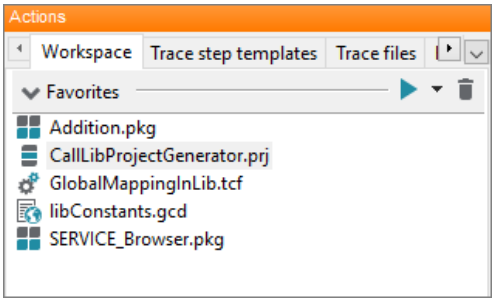


Figure 17: Favorites with library workspace support

To make it easier to distinguish between local and library artifacts, library items are now displayed in the favorites with matching petrol-colored icons.

Improvements to dependency management for Python libraries

et tc

The management of external Python libraries has been improved further. Dependencies from library workspaces are now considered and checked for conflicts with the libraries defined in **requirements.txt** before installation.

Overall, the installation process for Python libraries has improved, enabling the installation of source code libraries if the system meets their building requirements.

3 Test aspects

3.1 Multimedia

Göpel: LVDS channel is now configurable

et

Until now, ecu.test always assumed that channel 0 should be used. In case you use different LVDS channels of your Göpel framegrabbers, you can now select this in ecu.test.

Camera specific OpenCV properties can now be configured on the image port

et

You can now configure properties like focus and focal length on the **Image** port as OpenCV-Properties.

The properties are identical to those you might have already set using the job **SetOpenCVProperty**, but they are applied before the test start.

3.2 SiL

ASAM: XIL and derived tool connections: New job to pause the simulation

et

We now offer a job to pause the simulation. Jobs to start and stop were already available.

- **PauseSimulation**

Caution: If simulation time is used as test time (continuous or stepwise), pausing or stopping the simulation influences the progress of te test case and can block it completely.

3.3 HiL

INCA: Load experiment without data set

In particular for SiL applications, you may wish to avoid having to specify a data set for certain devices that also work without a data set.

Starting with this version of **ecu.test** 2025.2 in conjunction with INCA version 7.5.3 or higher, an empty data set is automatically generated and used when not specifying a hex file. It is not necessary to specify a hex file in the test configuration.

IPG: RealtimeMaker

The IPG RealtimeMaker is used to control the IPG XPack HiLs.
With **ecu.test**, it is therefore possible to automate interfaces such as bus communication or IO control. The range of functions of the RealtimeMaker connection corresponds to that of the CarMaker connection, with the exception of scenario control.

Note: IPG RealtimeMaker is supported via the CarMaker tool connection.

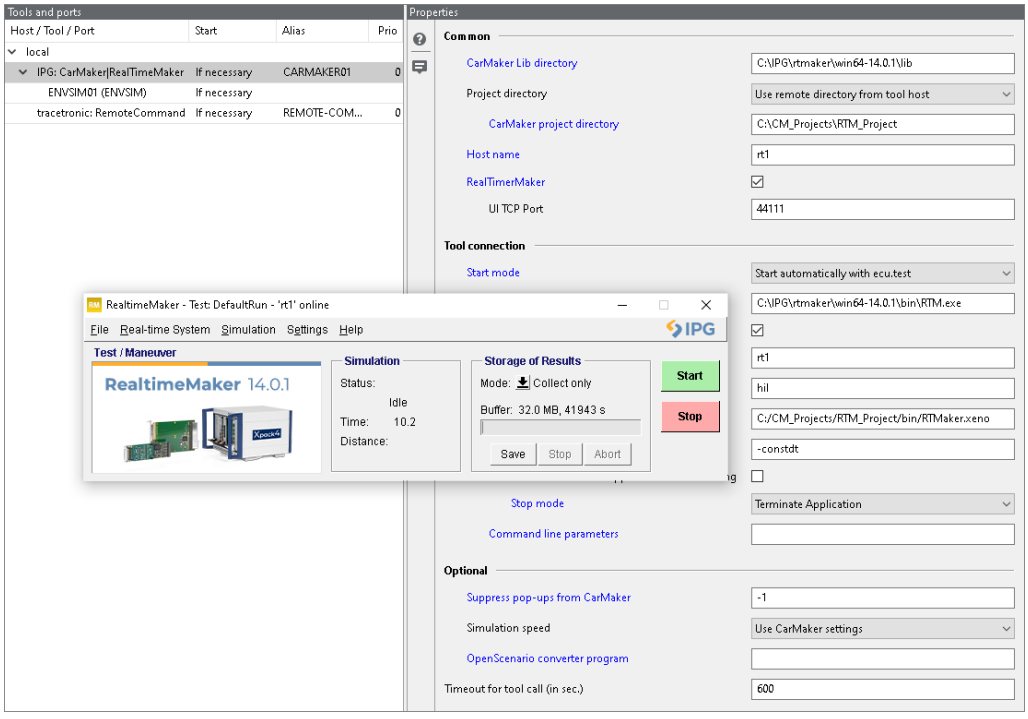


Figure 18: Support for IPG RealtimeMaker via the CarMaker tool connection

3.4 Test management

Jama connect: New possibility to connect jama to ecu.test

et

Until now, the connection to Jama connect was firmly integrated into **ecu.test** and could be adapted to specific workflows with the help of ALM hooks.

With the introduction of the new **UserTestmanagement** API, we now offer a Python-based sample workflow for Jama connect that offers significantly more customization options. This enables an even more flexible and customized connection from Jama to **ecu.test**.

3.5 ecu.test *calibration*

Recording many measurements

et

With the new two-staged recording mode, we enable the recording of a large amount of measured variables.

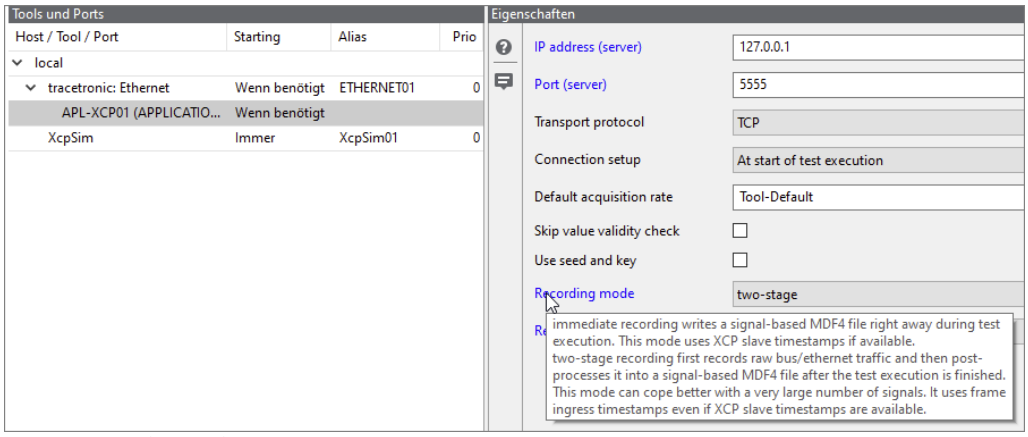
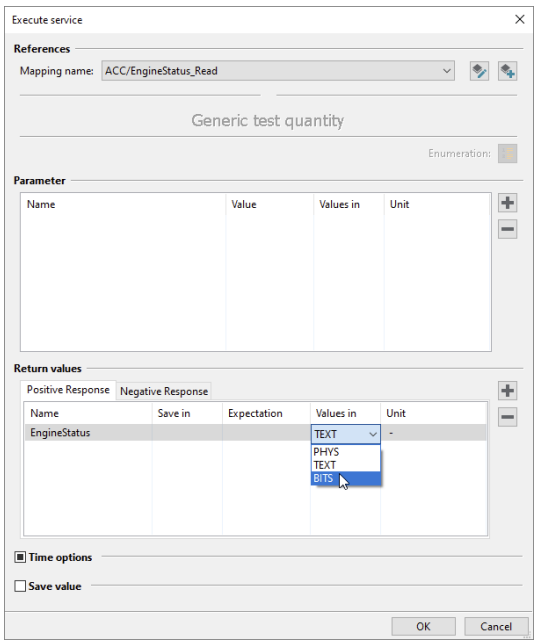


Figure 19: New feature for XCP: two-stage recording mode

3.6 ecu.test *diagnostics*

Bitcode representation in generic diagnostic test steps



The return values of the generic diagnostic steps have been extended to include bitcode (BITS) representation.

This corresponds to the bit stream of the data as it is sent at the lowest level of the transmission medium. Of course, the converted values (PHYS and TEXT) are still available.

You can easily switch between the representations using the selection menu. The expectation changes accordingly.

Figure 20: Switch between representations using the selection menu

The representation is also listed in a table in the report.

Figure 21: Display of representation in the report

Better support for functional addressing

ecu.test offers functional addressing for sending diagnostic requests to multiple control units or a functional group. With version 2025.2, this function has been improved in terms of robustness against timeouts and negative responses. These are now assigned correctly and no longer cause malfunctions.

DoSoAd for Vector: XL API

The DoSoAd transport protocol is now also available for the **Vector: XL API** tool. To use it, select the new DoSoAd port in the TBC. You will find it under the port type **DIAGNOSTICS**.

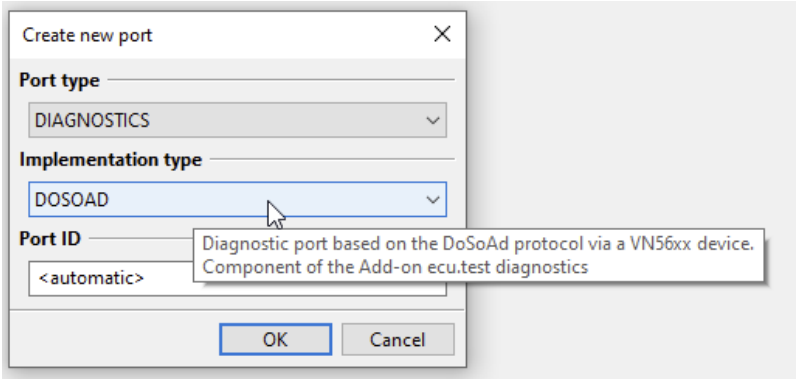


Figure 22: Configuring the DoSoAd port

The port offers you all the jobs you are already familiar with. Of course, you can also use all symbolic test steps from your diagnostic database with the port.

UDS via CAN in trace analysis

UDS messages from a loaded ODX or PDX can now be read in trace analysis. The parameters stored in the TCF for communication are taken into account.

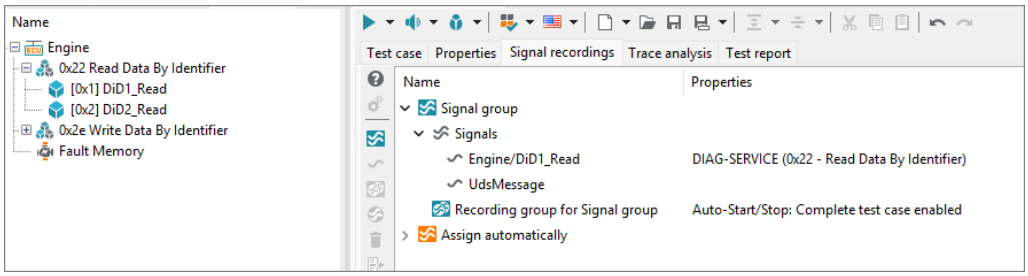


Figure 23: UDS- UDS messages in trace analysis

Accessing the protocol without a database is also possible using the pseudo signal **UdsMessage**. In this case, the parameters stored in the TCP are also required for communication.

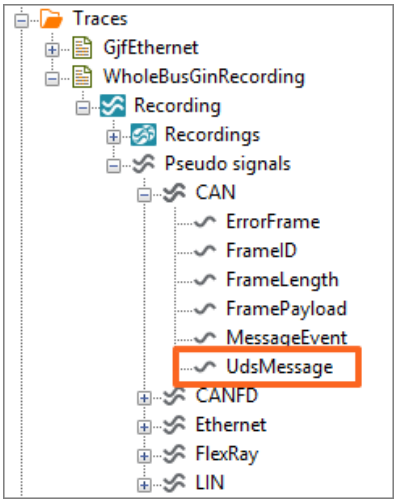


Figure 24: Pseudo signal *UdsMessage*

By default, the "system identifier" for the generated mapping target is empty if the signal is added to a signal group or mapping.

This ensures that all ECUs stored in the TCF are considered. However, if only the UDS communication of a single ECU is required, the system identifier can be set accordingly.

3.7 Communication

Logger connections: Reading and analyzing of current and voltage values

et

Logger protocols such as ASAM CMP, PLP, and TECMP support, besides access to bus communication, also measurement of current and voltage values at defined measuring points. These values can be transferred via Ethernet using the respective logger protocol.

ecu.test now offers a model port for reading the measured values in the test case and for recording them. Analyzing measured values from PCAP recordings in trace analysis is also supported.

To simplify configuration, multiple routes can be measured on a single port using a YAML-based configuration.

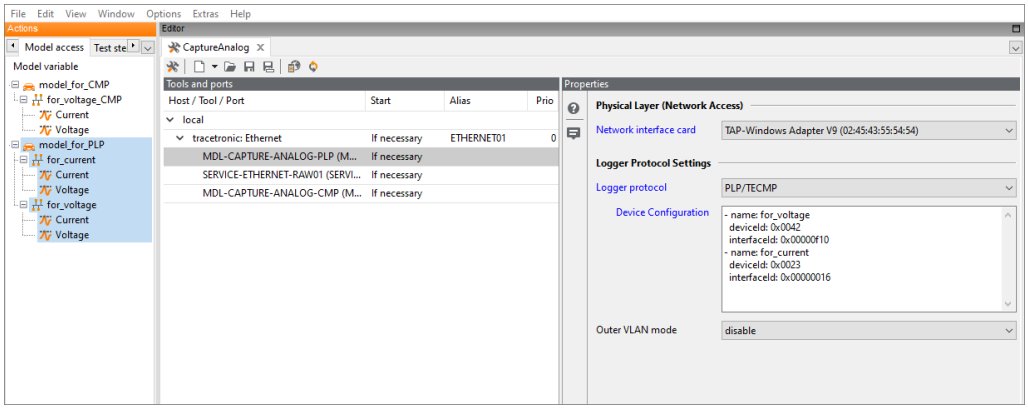


Figure 25: Configuring the logger protocol

SOME/IP Simulation: Leave Active

The option **Leave simulated services active** on SOME/IP service ports not only keeps the script-based simulation active, but also the cyclic sending that was activated using the **Send event** test step.

This ensures that a previously activated simulation remains active even between the execution of several test cases.

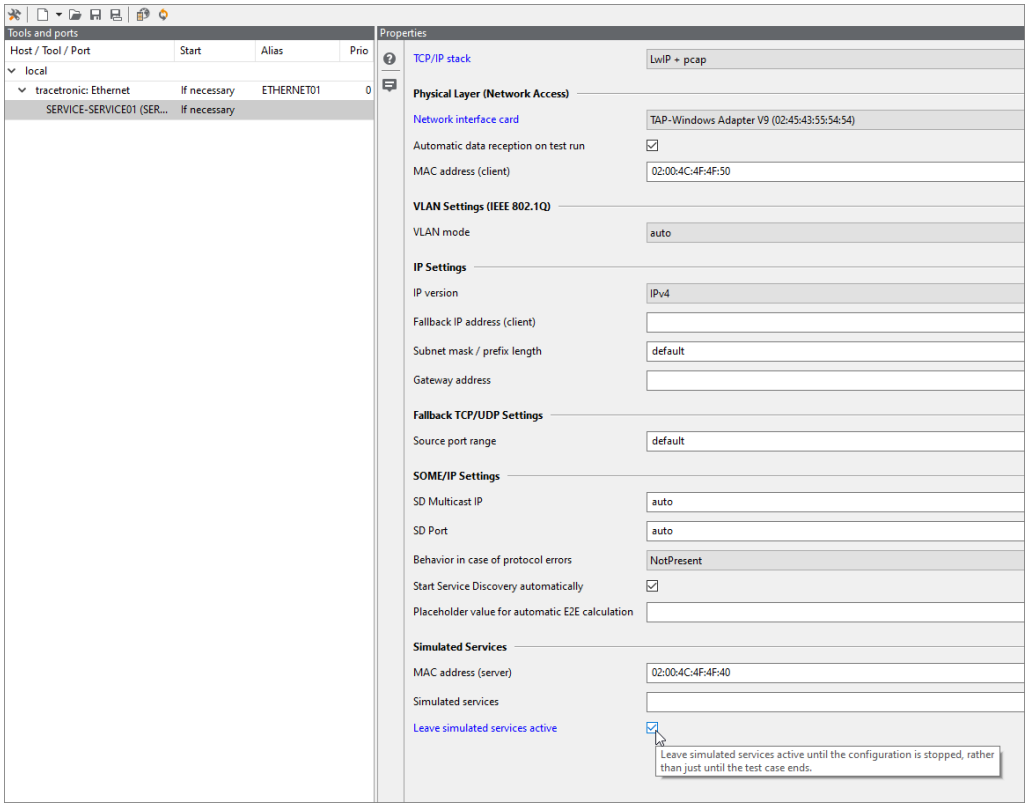


Figure 26: New option on SOME/IP service ports: Leave simulated services active

J1939: Bus reading and trace analysis for long messages

et

It is now possible to read bus signals or signal groups from messages that are longer than 8 bytes and that are sent via the J1939 transport protocol.

To do this, simply activate **J1939 mode** for **extended CAN IDs** in the test configuration of the hardware-related bus ports.
The use of the trace analysis of ASC files is also supported.

Note: The previously existing option **Ignore upper 3 bits (J1939 priority)** is automatically migrated to the new mode.

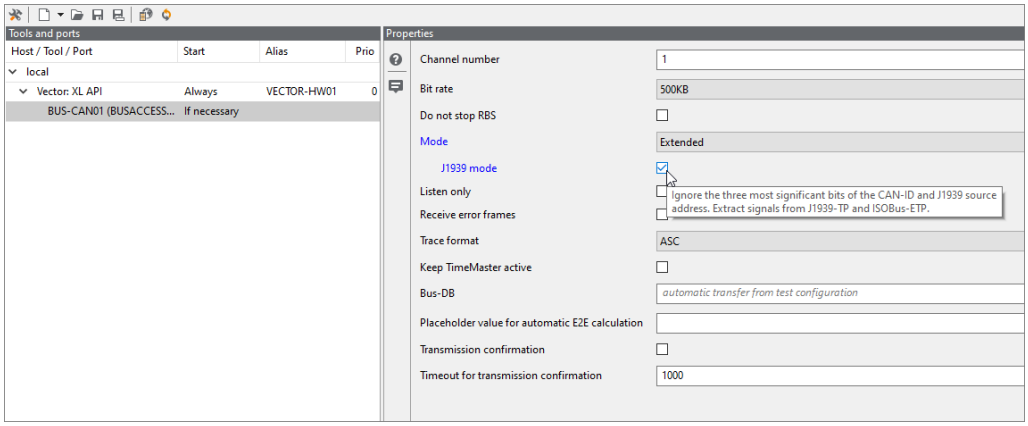


Figure 27: Activation of J1939 mode in the test configuration of the hardware-related bus ports

3.8 Trace analysis

Extended support for GIN logger records

et tc

The GIN recording directory is now provided as a single recording. In addition, GJF files from Ethernet loggers are now supported. This means that, as with other message-based formats, a wide range of protocols for interpreted signals can now be analyzed (when the database is loaded).

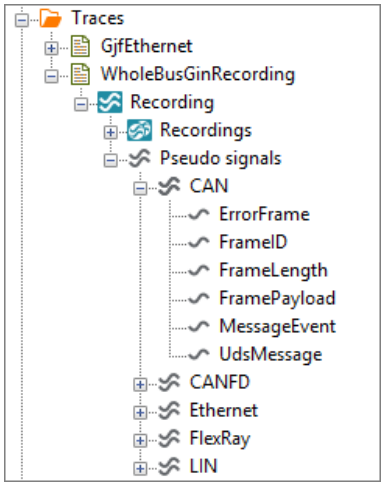


Figure 28: Display of GIN recordings with available pseudo signals for access at the protocol level

Performance optimization for signal-based trace merging

et tc

Signal processing from bus recordings, such as those from ASC, has been optimized, which significantly speeds up trace merging.

ROS2: Support for structure information in rosbag recordings

et tc

ROS2 trace analysis has been extended to support the new rosbag recording format, which stores structure information (IDLs) in the SQLite database.

With this change, additional MSG files, which previously stored the structure information, are no longer required if the structure information already exists in the SQLite database.

4 Tools and Interfaces

4.1 New tool versions et

	Provider	Website	System	Product name	Version
1	**National Instruments**	Release link	Software for embedded software testing for hardware-in-the-loop applications	**VeriStand**	**2025**
2	**Synopsys**	Link	Software-in-the-Loop solution for Virtual ECUs	**Silver**	**W-2025.03**
3	**Vector**	Release link	Programming interface for Vector hardware products	**XL Driver Library**	**25.20.14.0**

Support for Npcap 1.81 for Ethernet tests on Windows

et

Caution: When using **Npcap 1.70 and newer** in conjunction with **802.1q tagged VLAN**, there may be restrictions when recording self-sent packets (depending on the used network driver).

4.2 APIs

4.2.1 General

Package Check: New parameter „recursive"

et tc

The APIs for running checks have been extended with a **recursive** parameter to control the check of referenced artifacts. This extension is available in the REST and COM APIs, enabling more flexible and efficient testing of workspaces.

4.2.2 Command Line Interface (CLI)

Specifying the report folder in the execute command

et tc

The execute command now supports the **`--reportdirectory`** option, which can be used to specify the target directory for the generated report.

- **Absolute path specification**: `--reportdirectory /abs/Path`
  - stores the report in the specified absolute directory
- **Relative path specification**: `--reportdirectory rel/Path`
  - stores the report relative to the current working directory (analogous to the behavior of `--workspace`)

This extension allows flexible further processing of the reports according to individual requirements in your own CI/CT environment.

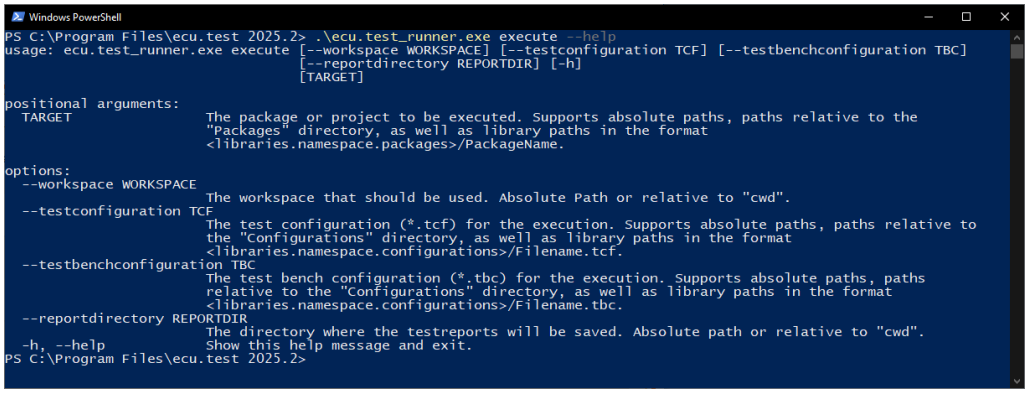


Figure 29: Specify the report folder in the execute command

Specifying library workspace references

et tc

The command line interface (CLI) now supports the so-called angle bracket notation accessing library workspace artifacts:

- `<libraries.libname.packages>/package.pkg`

This allows packages, projects, and configurations to be referenced from included library workspaces.

Specifically, this means that library workspace references are also supported for

- TARGET
- `--testconfiguration`
- `--testbenchconfiguration`

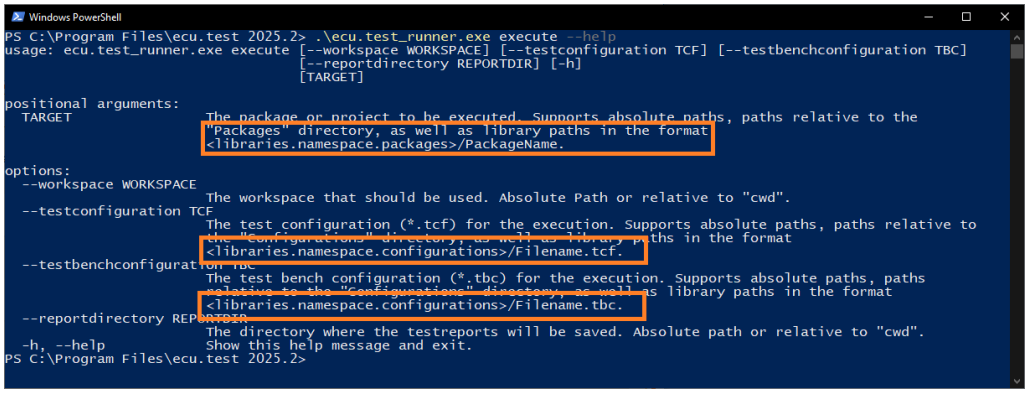


Figure 30: Specifying library workspace references in the execute command

4.2.3 REST API

Using Library workspace references

et tc

The REST API now supports the so-called angle bracket notation for accessing library workspace artifacts:

- `<libraries.libname.packages>/package.pkg`

This allows packages, projects, and configurations to be referenced from integrated libraries. Specifically, this affects the endpoints for

- `execution (put)`
- `configuration (put).`

Corresponding path specifications are of course also possible in Playbooks in **test.guide**.

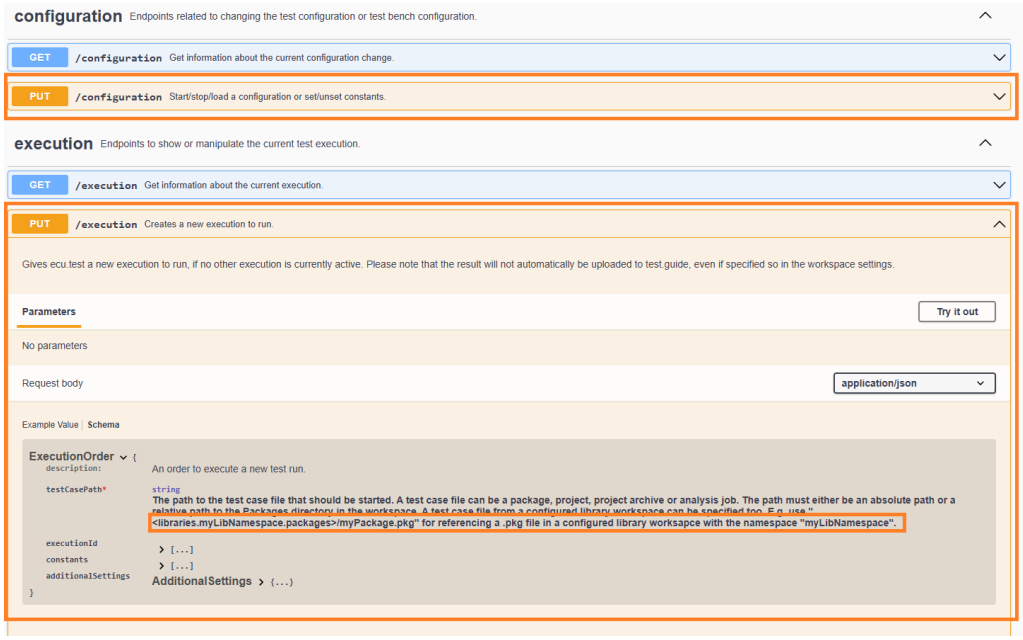


Figure 31: Specifying library workspace references in the REST API

4.2.4 UserTool

More flexible job parameters and model selection in the testconfiguration

- Job parameters with domains now also support boolean and integer values
- It is now possible to offer a selection of models and to select the one to start in the test configuration

```
class UserModelPort ¶
```

Bases: `UserPort`

This class describes the interface of a user model port. Classes that fulfill this interface can be used as model port of a user tool.

```
classmethod GetModels()
```

Optional (does not have to be implemented)

This method is called to get all available models for test configuration. If it's implemented, a new property "Model" will be added to your port properties which contains a selected model from this list.

Returns

list of available models.

Return type

list[str]

Figure 32: Documentation UserModelPort

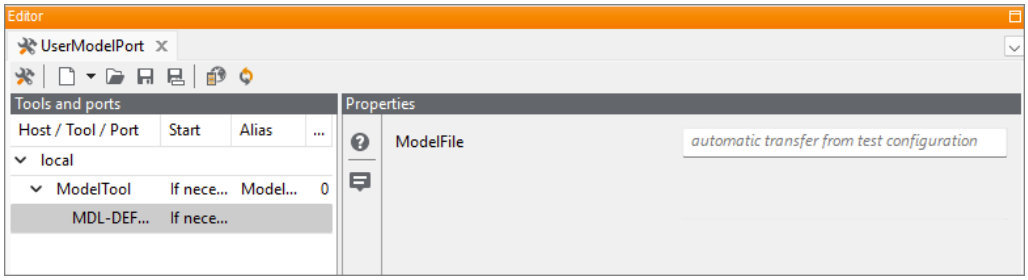


Figure 33: TBC settings of the model port

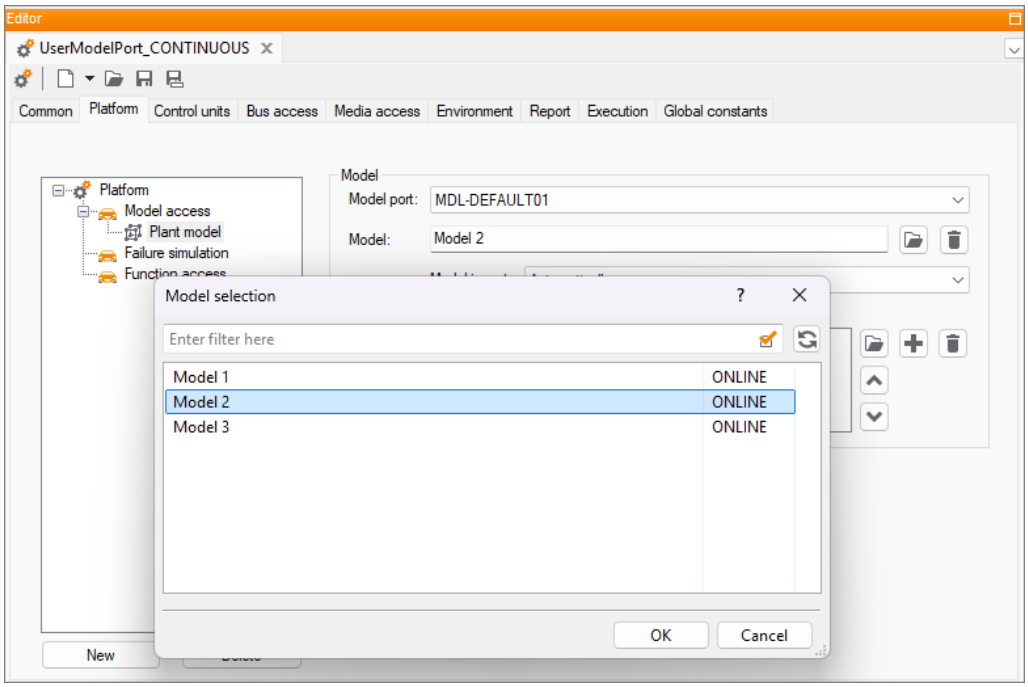


Figure 34: TCF of the configured model

4.2.5 UserTestmanagement

New API for custom connections to test management systems

et

With the **UserTestmanagement API**, it is now possible to implement a connection to a test management system in Python.

4.3 Sneak Preview

AI-based test case creation with ecu.test *agent*

The **ecu.test** *agent* is an AI-powered assistant that significantly accelerates the creation of test cases in **ecu.test**. It is fully integrated into ecu.test, retrieves the data for the current context directly from the workspace, and generates suitable, executable test steps for defined instructions in the test case at the touch of a button.

The underlying AI model is freely selectable. We support both proprietary models, such as ChatGPT, Gemini, and Claude, and open-source models, such as Llama, Mistral, and Qwen.

The **ecu.test** *agent* can also be operated entirely on-premise so that sensitive test data never leaves your company network.

We are happy to help you set up a suitable configuration with a solid database. Contact our support team.

Note: More information about the ecu.test agent is also available on our homepage.

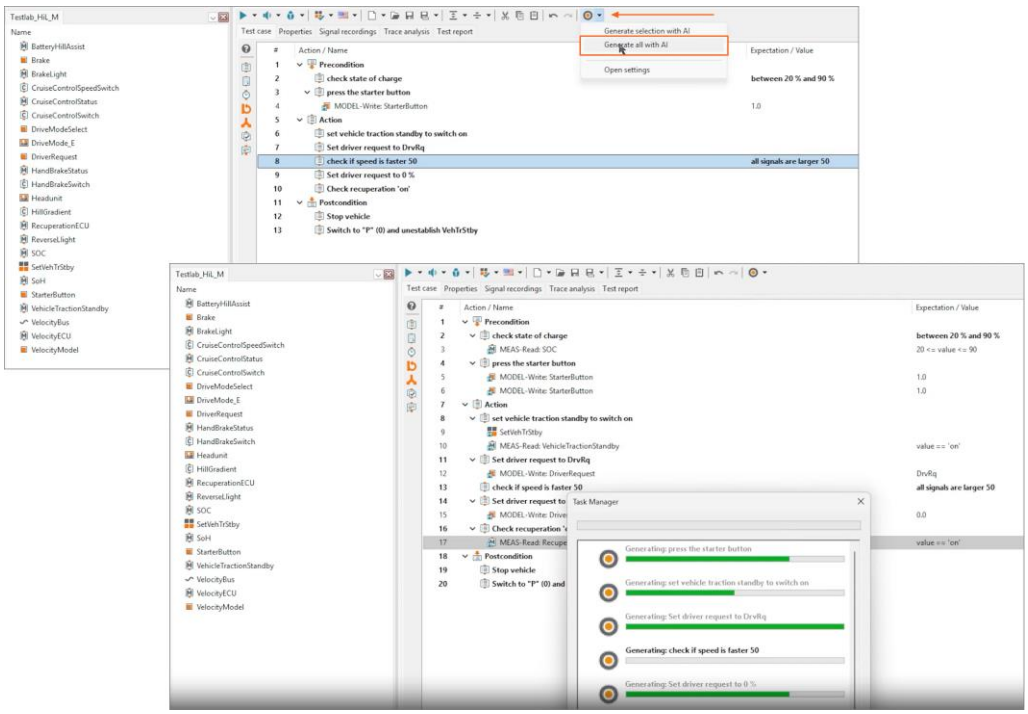


Figure 35: AI-based test case generation with the ecu.test *agent*

Python-based test case creation ecu.test *code*

With the **ecu.test** *code*, developers can test their software in their familiar programming environment without having to switch tools.

The Python-based framework seamlessly integrates into Visual Studio Code, offering lightweight access to **ecu.test** and more than 80 tools and automotive interfaces via code completion.

Developers can write ECU code, define appropriate tests, and execute them in a SiL environment. They receive immediately usable results directly in the programming environment or from the CI system. This allows defects in the code to be detected early and corrected directly.

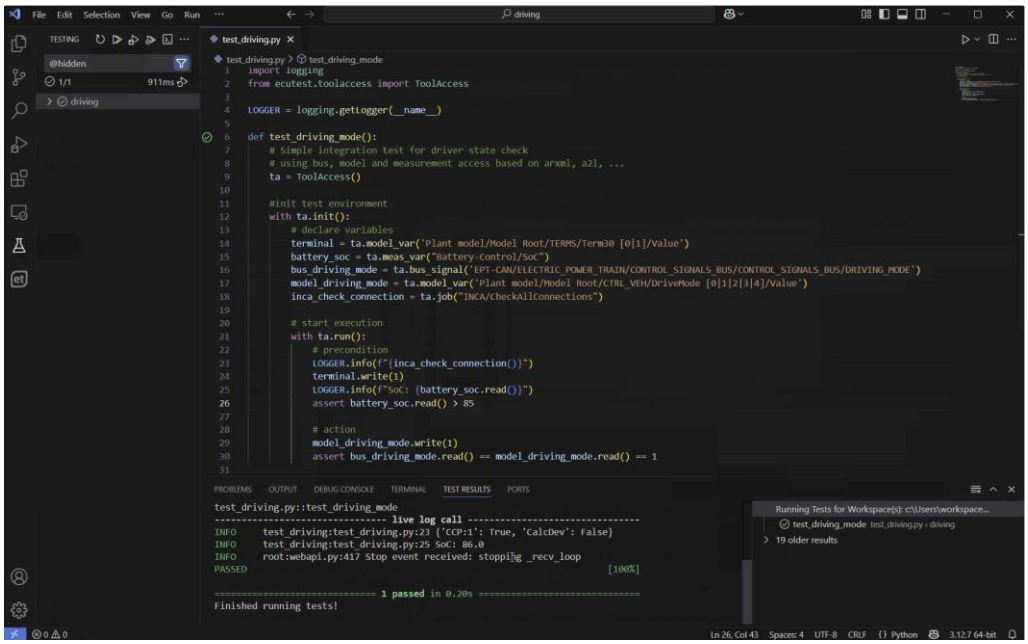
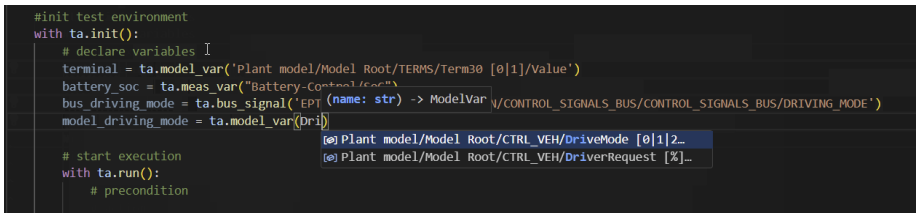


Abbildung 36: Erstellung von Testfällen mit ecu.test code

Advantages

- Access to all tools and automotive interfaces supported by **ecu.test** via code completion



- Lightweight and lean API that can be used without prior knowledge of **ecu.test**
- Full integration into the tracetronic tool chain, including **test.guide**
- Flexible integration into CI environments and Python test frameworks such as unittests, pytest, ...
- Compatible with Linux and Windows

Note: Have we caught your interest? If so, get in touch with us!
We would be happy to give you an early look at **ecu.test** *code* before its official release.

ecu.test Linux GUI for Ubuntu 24.04 LTS

ecu.test 2023.1 was the first version to provide a native Linux GUI.
This made it possible to develop, execute, and, if necessary, debug tests under Linux. Available tools, such as various ADAS/AD tools, MATLAB/Simulink, or various network interfaces, could be used without workarounds such as remote tool servers or virtualization.

In the current version, the Linux GUI has been updated to **Ubuntu 24.04 LTS** and has been extensively tested and stabilized in collaboration with pilot users.

The **ecu.test** Linux version is worth a look, especially for those who already work with Linux-based toolchains or want to make their test environment more platform-independent.

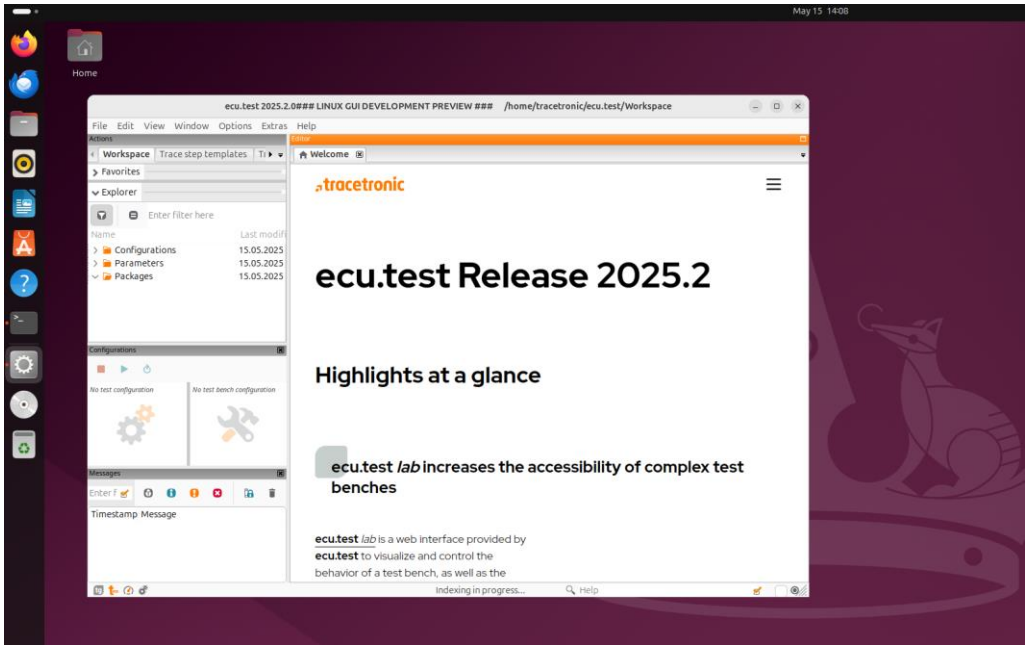


Figure 37: ecu.test Linux GUI for Ubuntu 24.04 LTS

5 Discontinuations

5.1 Discontinued features and incompatibilities in this version

CARLA connection

et

The CARLA connection is being removed with **ecu.test 2025.2**.

SILVER XIL

et

Support for versions 2024.9 and older was removed.
Due to incompatible changes in the XIL interface of Silver, the added support for version 2025.03 made it necessary to remove support for older versions.

BusBrowser API

et tc

The **GetPhysResolution()** method has been removed.
Please use **GetLinearCoeffs()** instead.

5.2 Discontinued features in future versions

"Alternative call representation" of Package properties

et

The **Alternative call representation** in the general properties of a package has been deprecated since version **2024.3** and will **be removed permanently with ecu.test 2025.2.**

The feature was implemented as part of keyword-based testing in the mapping of reference package calls and will be used there from now on.

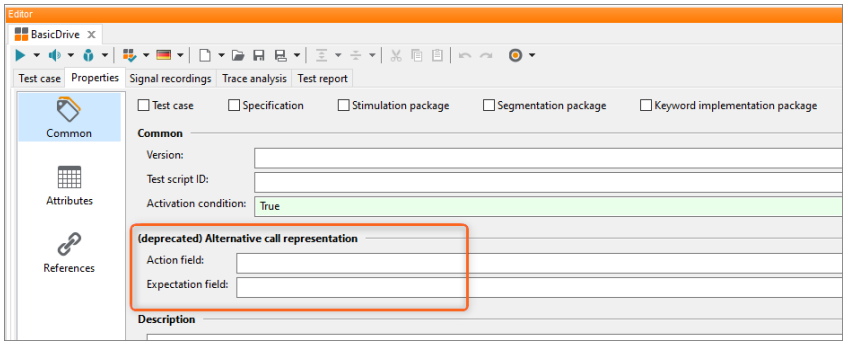


Figure 38: Package properties: Alternative call representation

Odx Migration Helper

et

The **Odx Migration Helper** will be removed with **ecu.test** 2025.3.

DiagBrowser: GetUdsName

et

The **GetUdsName** method will be removed with **ecu.test** 2025.3.

Discontinuation of the ReqIf import

et

The functionality for importing **ReqIf** data as a package and project attribute will be removed with **ecu.test** 2025.4.

KS: Tornado only via ASAM ACI

et

The tool connection will be removed with **ecu.test 2026.1**.

It will be replaced by the new connection based on ASAM ACI, which is available since **ecu.test 2023.3**.

Fibex Support

et tc

Fibex support for Bus is being discontinued and will be removed with **ecu.test 2025.4**.
Fibex support for DLT will continue to be provided.

Jobs RequestSeed und SendKey

et

The **RequestSeed** and **SendKey** jobs are replaced.

- RequestSeed → SecurityAccessRequestSeed
- SendKey → SecurityAccessSendKey

The new jobs also support the Seed & Key DLLs.

Discontinuation of the integrated test management connection for Jama

et

The connection to Jama, which was firmly integrated into **ecu.test**, is being replaced by the new Python-based approach.

5.2.1 Removed API methods in future versions

et tc

Old Command	New Command	Remarks
Report API		
ReportItem. GetActivity	ReportItem.GetLabel()	Obsolete since version 6.4: Name and activity must be replaced by label.
ReportItem. SetActivity	ReportItem.SetLabel()	Obsolete since version 6.4: Name and activity must be replaced by label.
ReportItem.GetName	ReportItem.GetLabel()	Obsolete since version 6.4: Name and activity must be replaced by label.
ReportItem.SetName	ReportItem.SetLabel()	Obsolete since version 6.4: Name and activity must be replaced by label.
ReportItems.ReportItem. GetActivity	ReportItem.GetLabel()	Obsolete since version 6.4: Name and activity must be replaced by label.
ReportItems.ReportItem. SetActivity	ReportItem.SetLabel()	Obsolete since version 6.4: Name and activity must be replaced by label.
ReportItems. ReportItem.GetName	ReportItem.GetLabel()	Obsolete since version 6.4: Name and activity must be replaced by label.
ReportItems. ReportItem.SetName	ReportItem.SetLabel()	Obsolete since version 6.4: Name and activity must be replaced by label.

Old Command	New Command	Remarks
Object API		
API for Reports		
ReportAnalysisEpisode.GetActivity	ReportAnalysisEpisode.GetLabel	Deprecated since version 2020.2, please use: GetLabel().
ReportAnalysisEpisode.GetName	ReportAnalysisEpisode.GetLabel	Deprecated since version 2020.2, please use: GetLabel().
ReportAnalysisStep.GetActivity	ReportAnalysisStep.GetLabel	Deprecated since version 2020.2, please use: GetLabel().
ReportAnalysisStep.GetName	ReportAnalysisStep.GetLabel	Deprecated since version 2020.2, please use: GetLabel().