

Release Notes

ecu.test 2025.3

trace.check 2025.3

Date: 09/30/2025
© 2025 tracetronic GmbH

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

Content

| | |
|---|----|
| Overview | 1 |
| 1 Highlights in ecu.test 2025.3 | 2 |
| 2 Usability | 7 |
| 3 Test aspects..... | 14 |
| 3.1 ADAS/AD..... | 14 |
| 3.2 Multimedia | 15 |
| 3.3 SiL | 17 |
| 3.4 HiL..... | 19 |
| 3.5 Test management | 20 |
| 3.6 ecu.test <i>calibration</i> | 21 |
| 3.7 ecu.test <i>diagnostics</i> | 22 |
| 3.8 ecu.test <i>lab</i> | 25 |
| 3.9 Communication..... | 27 |
| 3.10 Trace analysis | 29 |
| 3.11 trace.xplorer | 30 |
| 4 Tools and Interfaces | 36 |
| 4.1 New tool versions..... | 36 |
| 4.2 APIs..... | 36 |
| 4.2.1 Internal API..... | 36 |
| 4.2.2 REST API | 37 |
| 4.2.3 UserTool..... | 37 |
| 5 Discontinuations..... | 38 |
| 5.1 Discontinued features and incompatibilities in this version | 38 |
| 5.2 Discontinued features in future versions..... | 40 |
| 5.2.1 Removed API methods in future versions | 42 |

Overview

With **ecu.test** 2025.3, we are launching **two new products**:

ecu.test *agent*

... is an AI-powered assistant fully integrated into **ecu.test** that automates the test case generation.

ecu.test *code*

... is **ecu.test** in a Python environment and thus tailored directly to programmers' needs.

Further highlights of this release include:

- **Model time**
The model time can be received from any model port configured as time source and then automatically forwarded to the bus and diagnostic components.
- **User-defined synchronization of recordings**
Custom synchronization types can be implemented as UserPyModules.

ecu.test *lab* joined our product family in 2025.2, and it already offers numerous new features that will inspire as well as support you and encourage you to try it out. You can test **ecu.test** *lab* free of charge until the end of 2025. Get in touch with our [support](#).

Is your area of expertise **ecu.test** communication? How convenient that there are six new features for it. Keywords are: Some/IP, MACSec, CAN-FD, and XL-API. Read more about them in Chapter [3.9](#).

This release includes two key enhancements related to **test.guide**. First, coverage analysis has been integrated into **test.guide**. This means that coverage files can now be evaluated centrally instead of only locally. Secondly, the test case key indicators exported to **trace.xplorer** files now contain a link for each test execution that takes you directly to the corresponding test run details in **test.guide**.

In addition to the above-described features, there are numerous innovations for **ADAS/AD**, **multimedia**, **SiL** and **HiL** setups, **diagnostics**, **trace analysis**, and **usability**. It's definitely worth reading on!

Note: Icons are used to indicate for which product a topic is relevant:

 **ecu.test**  **trace.check**.

1 Highlights in ecu.test 2025.3

ecu.test agent



The **ecu.test agent** is an AI-powered assistant that significantly accelerates the creation of test cases in **ecu.test**. Fully integrated into **ecu.test** as a service, it automatically generates suitable, executable test steps from test case specifications with a single click. It accesses all relevant data directly from the workspace to accomplish this.

You can freely select the underlying AI model. 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, ensuring that sensitive test data never leaves your company network. We are happy to help you set up a suitable configuration with a robust database. Please contact our support team.

Anyone with an active **ecu.test** license can now receive 50 free test steps (50 credits) per month. Contact our sales team to receive your token and get started. Of course, you can also purchase larger credit packages.

Visit the [ecu.test agent](https://www.ecu.test/agent) website to learn more about our new tool.

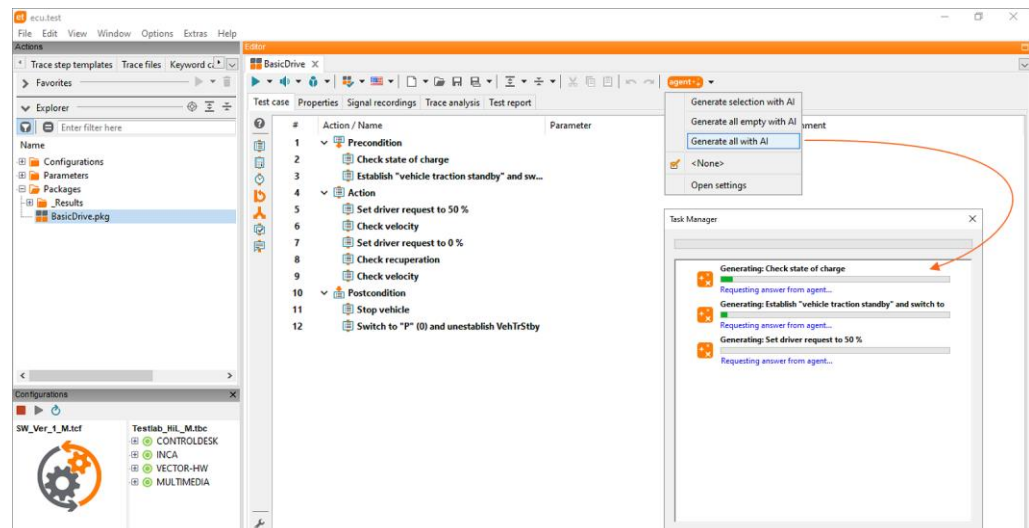


Figure 1: Test case generation with the **ecu.test agent**

ecu.test code

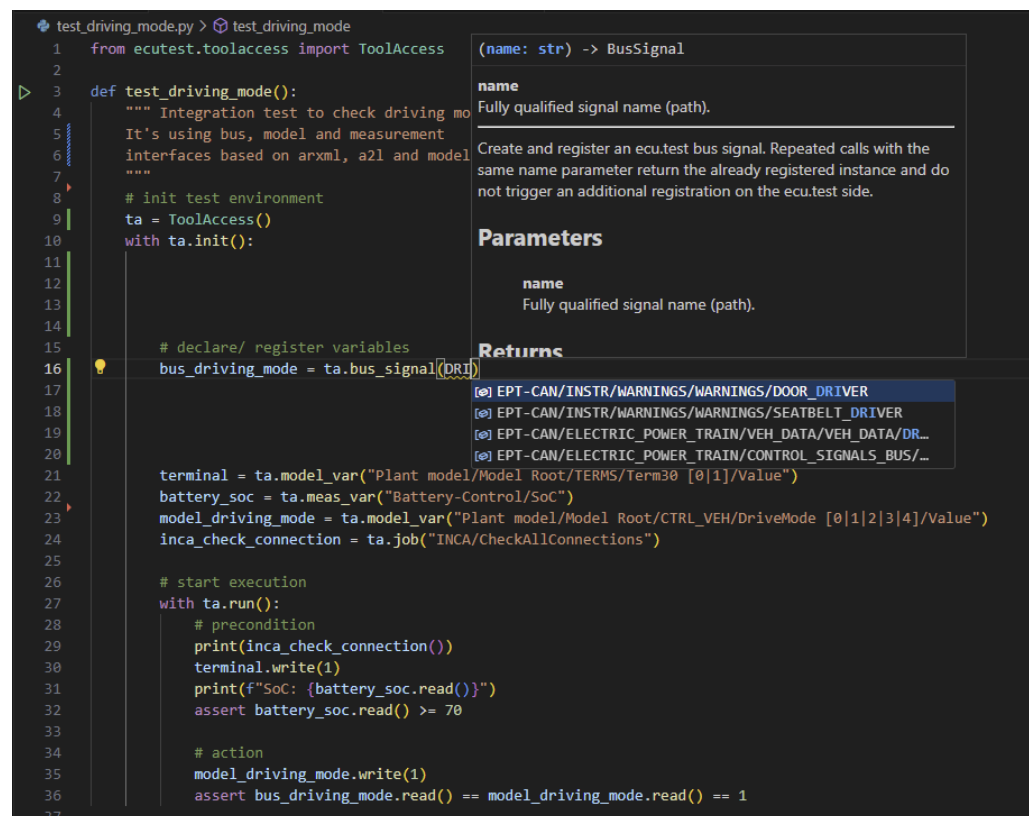


The test development workflows and test language in **ecu.test** are designed to meet the needs of test engineers. Reusability and an intuitive user interface for creating test cases were high priorities in the development of **ecu.test**.

As software developers become more important and responsible in the development process, there is a growing need for lightweight test solutions that can be implemented in code.

ecu.test code provides the capabilities of **ecu.test** in a Python environment to meet the needs of both worlds. Tailored to programmers' needs, it can be used without learning the extensive concepts of **ecu.test**.

It is not a replacement for **ecu.test**, but rather an alternative way to benefit from the functionality of **ecu.test**. It promotes collaboration between engineers and developers while pursuing the shift-left goal.



```

1 from ecutest.toolaccess import ToolAccess
2
3 def test_driving_mode():
4     """ Integration test to check driving mo
5     It's using bus, model and measurement
6     interfaces based on arxml, a2l and model
7     """
8     # init test environment
9     ta = ToolAccess()
10    with ta.init():
11
12
13
14    # declare/ register variables
15
16    bus_driving_mode = ta.bus_signal(DRI
17
18
19
20
21    terminal = ta.model_var("Plant model/Model Root/TERMS/Term30 [0|1]/Value")
22    battery_soc = ta.meas_var("Battery-Control/Soc")
23    model_driving_mode = ta.model_var("Plant model/Model Root/CTRL_VEH/DriveMode [0|1|2|3|4]/Value")
24    inca_check_connection = ta.job("INCA/CheckAllConnections")
25
26    # start execution
27    with ta.run():
28        # precondition
29        print(inca_check_connection())
30        terminal.write(1)
31        print(f"SoC: {battery_soc.read()}")
32        assert battery_soc.read() >= 70
33
34        # action
35        model_driving_mode.write(1)
36        assert bus_driving_mode.read() == model_driving_mode.read() == 1
37

```

(name: str) -> BusSignal

name
Fully qualified signal name (path).

Create and register an ecu.test bus signal. Repeated calls with the same name parameter return the already registered instance and do not trigger an additional registration on the ecu.test side.

Parameters

name
Fully qualified signal name (path).

Returns

- [e] EPT-CAN/INSTR/WARNINGS/WARNINGS/DOOR_DRIVER
- [e] EPT-CAN/INSTR/WARNINGS/WARNINGS/SEATBELT_DRIVER
- [e] EPT-CAN/ELECTRIC_POWER_TRAIN/VEH_DATA/VEH_DATA/DR...
- [e] EPT-CAN/ELECTRIC_POWER_TRAIN/CONTROL_SIGNALS_BUS/...

Figure 2: Test case creation directly in the Python environment

Features

- Create and execute test cases from any Python environment
- Access all tools supported by **ecu.test**
- Quickly access test variables through autocompletion in Visual Studio Code
- Seamless **test.guide** integration for result playback

Structure

- The Python library is the main component of **ecu.test** code. It provides direct access to **ecu.test** functions in the Python environment, as well as access to tools, data, and automation primitives.
- The VS Code extension is an optional add-on. Building on the library, it offers extensive autocompletion, online help, and optimized authoring to design, parameterize, and execute test cases without leaving VS Code.
- The pytest plugin enables seamless integration of test reports into **test.guide**.

Getting Started

- We welcome your feedback and want to further develop **ecu.test** code based on your use cases. That's why we are initially making **ecu.test** code available on request.
Just send us a short email to support@tracetronic.com.
An additional license for **ecu.test** is not required at this time.
- To use Visual Studio Code, you can install the extension from the VS Code Marketplace.
- To get started quickly, we have summarized further information in the **ecu.test** user guide under Getting Started.

Model time automatically affects all bus-related tools and diagnostics



With **ecu.test**, we support testing throughout the entire development cycle – from the first software baseline in SiL, through model testing in MiL, integration on HiL, up to vehicle testing.

However, the test domains are not always clearly separated. For instance, some low-cut models are already tested with virtual bus hardware. These hybrid test setups introduce new challenges, particularly regarding timing behavior. For example, the model's provided time must also affect individual communication layers to ensure synchronization of recordings with the model or to apply timeouts correctly.

This hurdle is now automatically handled by **ecu.test**, as model time is automatically forwarded to the bus and diagnostic components.

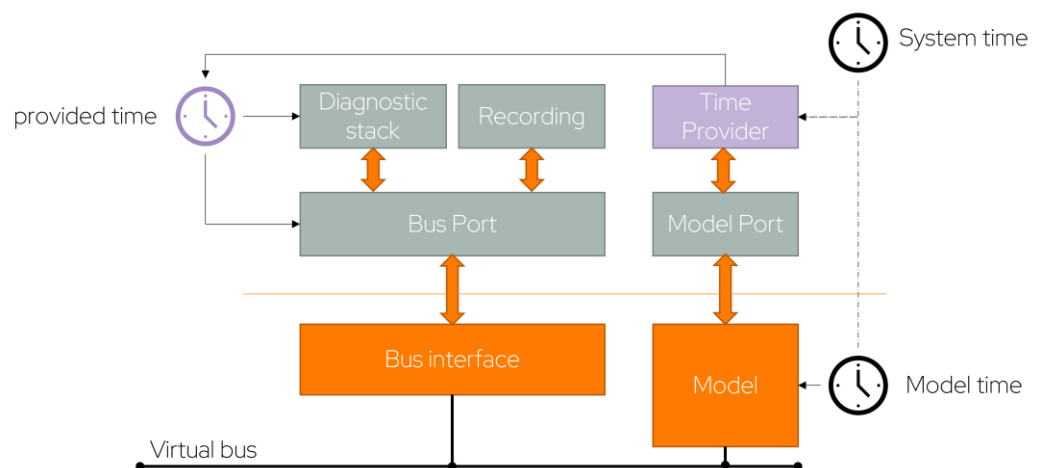


Figure 3: Schematic representation of model time transmission

To activate this feature, simply select a model port in the TCF. This ensures the model time is automatically applied to the communication components. This applies to bus and diagnostic ports as well as tracetronic: Ethernet ports.

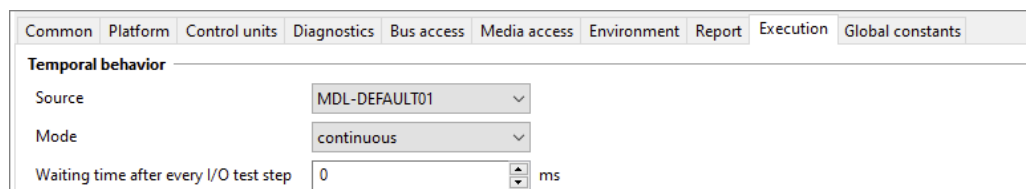


Figure 4: Activating the model port

Note: Currently, only the continuous mode is supported. However, plans for stepwise execution are already in place. Feel free to contact us about this topic!

User-defined synchronization of recordings



In addition to access to simple signals, access to protocols and multimedia data is required to ensure the security of complex vehicle systems. Trace analysis often requires subsequent synchronization of different recordings from multiple sources. This synchronization must be able to process complex data for each use case individually.

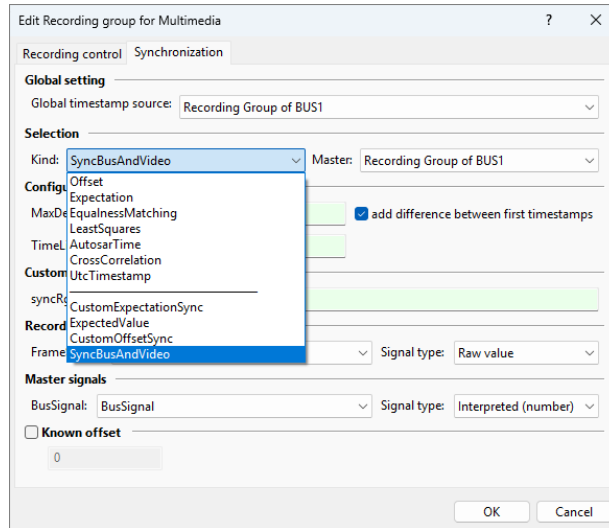


Figure 5: Selecting the capture mode for user-defined synchronization

The user-defined synchronization allows you to access signal values similar to trace step templates, apply your own algorithms or individual libraries, and determine a time offset.

Synchronization is provided in the form of a self-implemented user Python library.

2 Usability

Test case coverage: test.guide integration



Since **ecu.test** 2024.3, coverage metrics for test cases can be analyzed and, since **ecu.test** 2025.2, consolidated in a central report – but until now only on the basis of local coverage files. This was time-consuming, especially with distributed teams, many test runs, and large amounts of data.

With the new integration in **test.guide**, this process is now much more efficient: individually definable filters in your own **test.guide** instance allow you to select specific test runs. **ecu.test** loads the associated coverage data directly from **test.guide** and automatically generates a consolidated report without manual file storage.

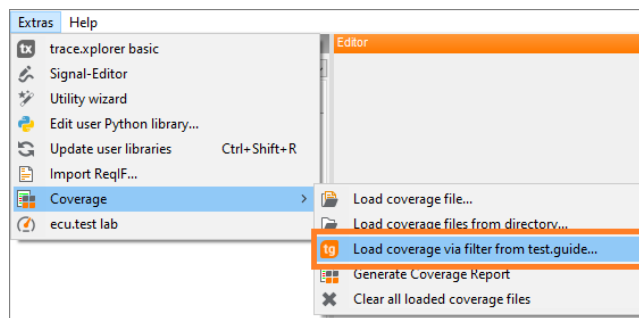


Figure 6: New entry in the Coverage menu under Extras

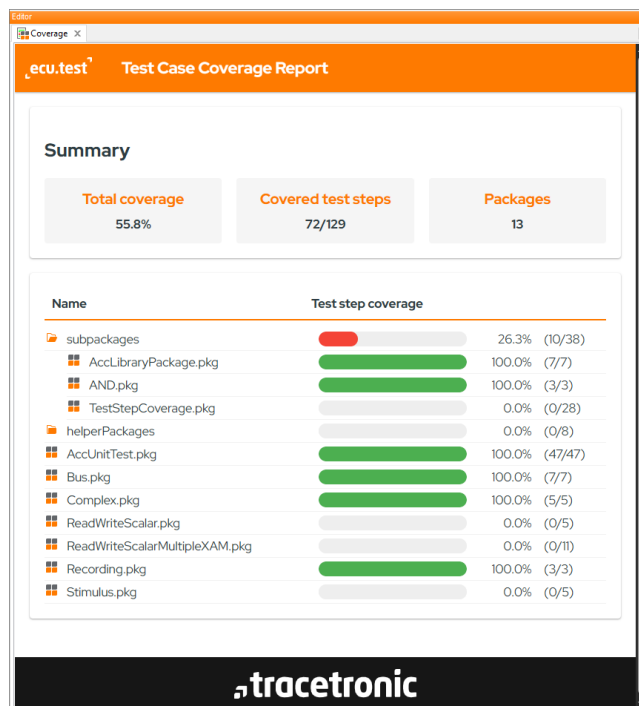


Figure 7: Consolidated Coverage Report

The combination with **test.guide** significantly expands the coverage analysis: It now enables centralized evaluation across many test runs and is ideal for systematically securing safety-critical libraries and identifying unused packages throughout the entire project.

With **ecu.test** 2025.3, coverage recordings are also automatically loaded into **test.guide** when report upload is active. This functionality is available since **test.guide** 1.199.0.

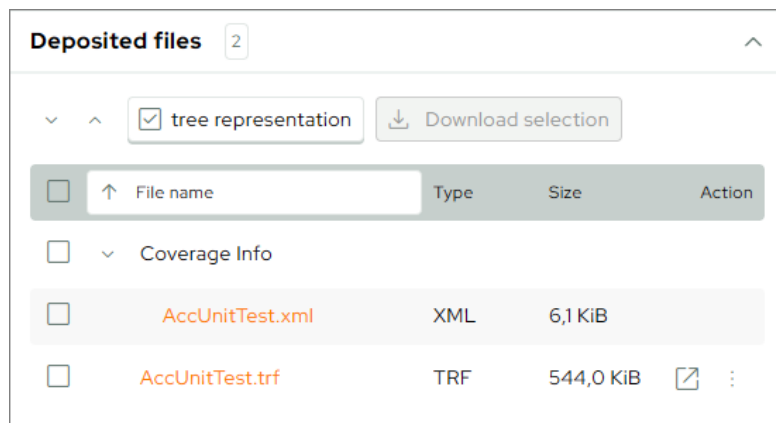


Figure 8: Storage of coverage records in test.guide

Test case coverage: Improvements in the HTML report



It is now possible to specify the maximum number of top-level elements in the HTML files of the consolidated coverage report. In addition, integrated **ecu.test** library workspaces are explicitly displayed.

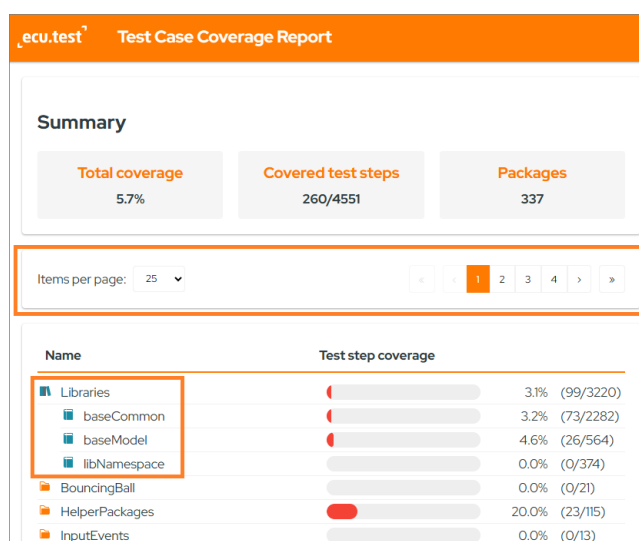


Figure 9: Consolidated Coverage Report

Direct integration of online help in ecu.test



The new online user documentaion was already introduced with **ecu.test 2025.2**. Starting with **ecu.test 2025.3**, it can now also be used when accessed via the Start menu ...

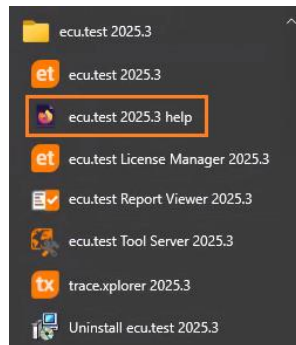


Figure 10: Help in the Start menu

... or directly from the tool.

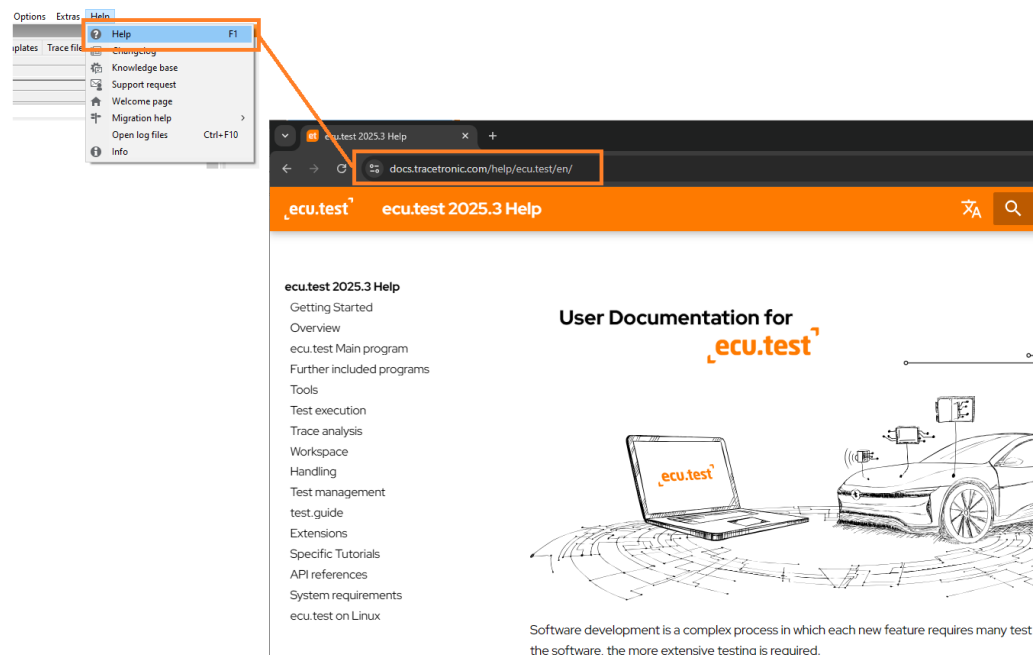


Figure 11: Clicking on Help now primarily opens the online help.

The language is recognized automatically, and if there is no Internet connection, a local fallback ensures that support is still available.

Thanks to the online connection, the content is always up to date, the search function has been significantly improved, and additional interactive functions will be provided in the future – for even faster and more targeted help in your daily work.

Select Ethernet adapter names based on their names



Previously, the hardware name of the adapter, including the MAC address, was used to identify a network adapter in Windows. Since the MAC address differs for each test station, a separate test bench configuration had to be created for each one.

Starting with this version, the Windows name is used for identification. This allows test bench configurations to be reused on different test benches if the same Windows name is selected.

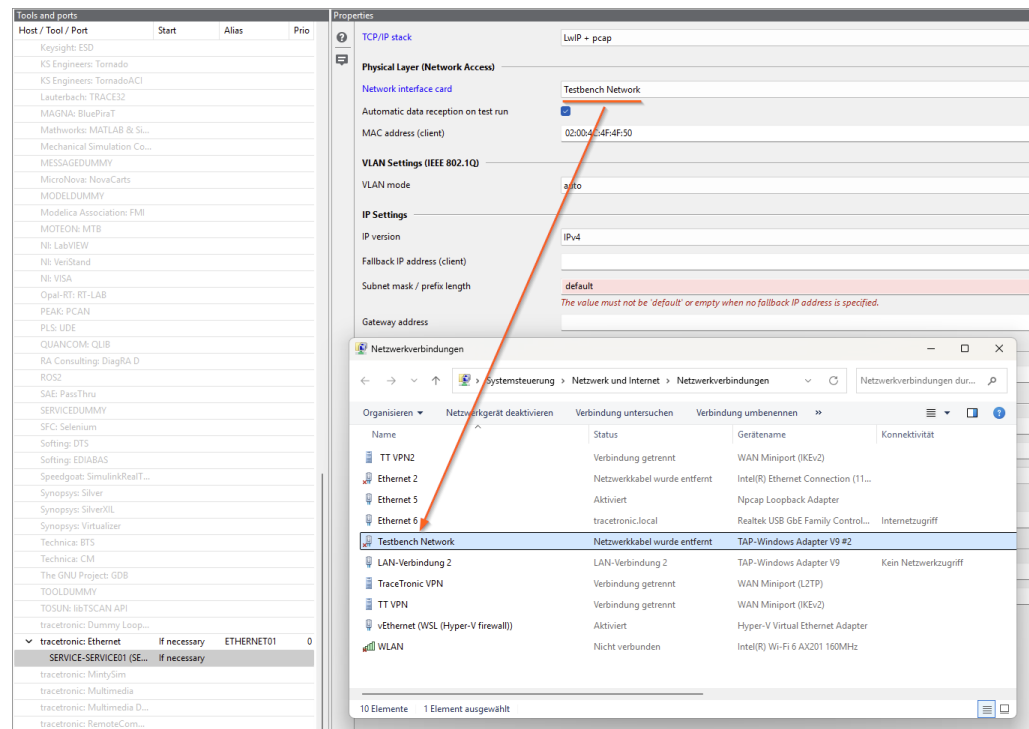


Figure 12: Using the Windows name for identification

Please note: Automatic migration to the new identifiers is not possible, meaning that all Ethernet TBCs must be updated and the respective network adapter must be re-selected.

Revised logging for significant performance improvement



Logging has been comprehensively optimized, which significantly speeds up execution, especially in test cases with very extensive logging. The new implementation is used by default – both in **ecu.test** and in the **Tool Server** – and thus ensures noticeably better performance.

If necessary, the behavior can still be adjusted in the settings under Protocol to use the previous logging (“Write output log files synchronously”).

Report Viewer: Switchable display of absolute time



Previously, absolute times in the report were always displayed in the user's local time. However, to facilitate cross-time zone collaboration and error tracking in external logs, it is helpful to be able to view times without converting them yourself.

The Report Viewer toolbar now allows you to choose from three display formats: Local time, the time of the test host that created the report, or UTC time without an offset.

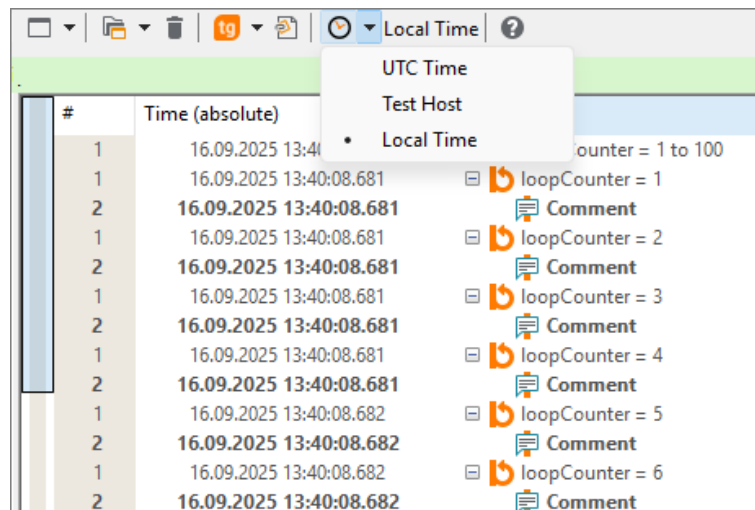


Figure 13: Switchable display of absolute time

Optimized Git connection



The Git connection has been optimized, which has a positive effect on the initial execution of larger projects from versioned workspaces.

Settings files are robustly written and loaded



In rare cases, local or workspace settings files may become corrupted. For example, using external script solutions or abruptly terminating **ecu.test** while writing settings files can result in an empty or incomplete file.

With **ecu.test 2025.3**, processing has become significantly more robust. Writing has been made atomic. Additionally, if an attempt is made to read an incorrect file, the default settings are restored.

Showing and hiding hidden files in Workspace Explorer



Previously, **ecu.test** always hid all files and folders whose names began with a dot. To make this configurable, especially for the development and use of library workspaces, a new button has been added to Workspace Explorer.

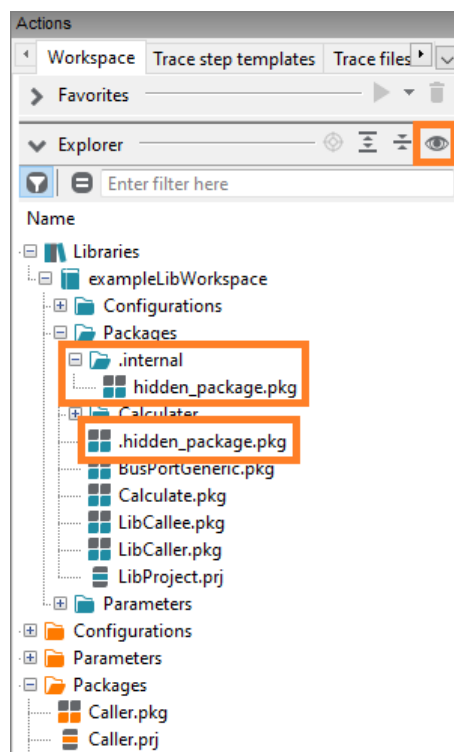


Figure 14: The new button, an eye, has been added to display hidden files when needed.

Re-execute parameterized test case execution with last parameterization



The selected values of a parameterized package execution are now saved and offered again when the package is executed again. If there are a large number of parameters and their values, changing individual values saves time when re-setting them.

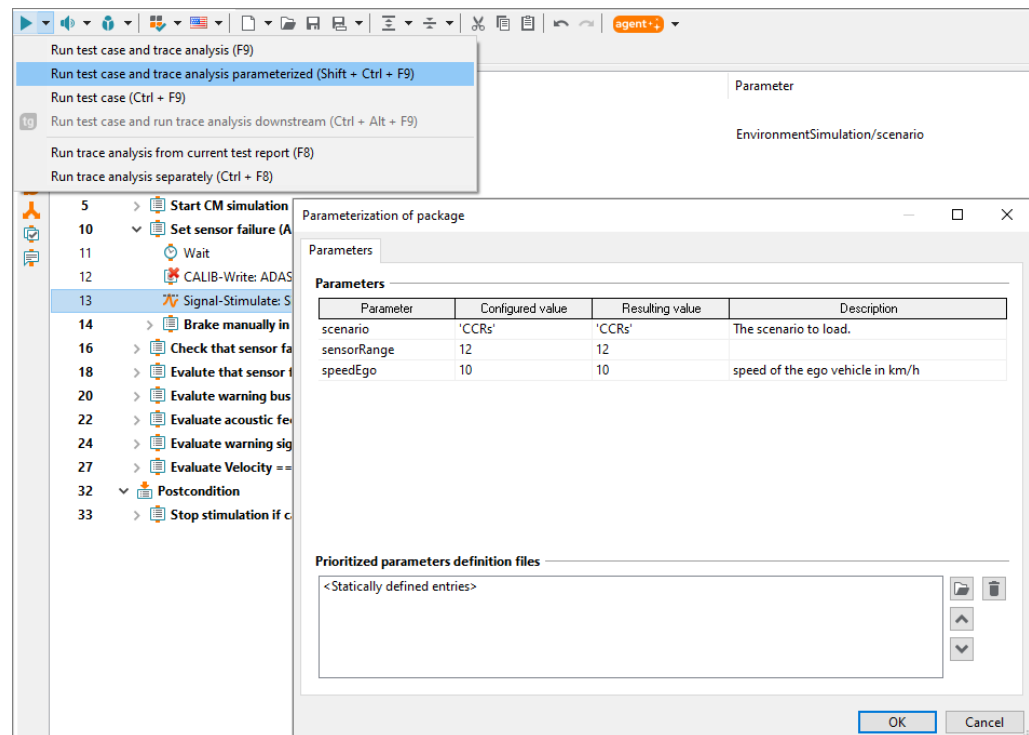


Figure 15: Execute test case and trace analysis with parameters

ecu.test considers the system's display scaling



For screens with higher resolution, scaling from 125% to 300% is used. **ecu.test** can now recognize this setting for each screen and adjust its display, making text particularly easy to read.

3 Test aspects

3.1 ADAS/AD

CarMaker: Real-time capability of stimulus steps



When using CarMaker on test benches such as xPack or Scalexio, it is often necessary to write signals. With the stimulus step, the user has the option to create and execute a signal curve consisting of constants and ramps. With previous versions of **ecu.test**, it was not possible to execute such signal curves in real time.

In **ecu.test 2025.3**, a technical feature has been made so that these signal profiles of the stimulus steps can be executed on the real-time platform. This ensures that the values and duration defined in the stimulus are exactly observed. No adjustment or migration is required for existing stimulus steps.

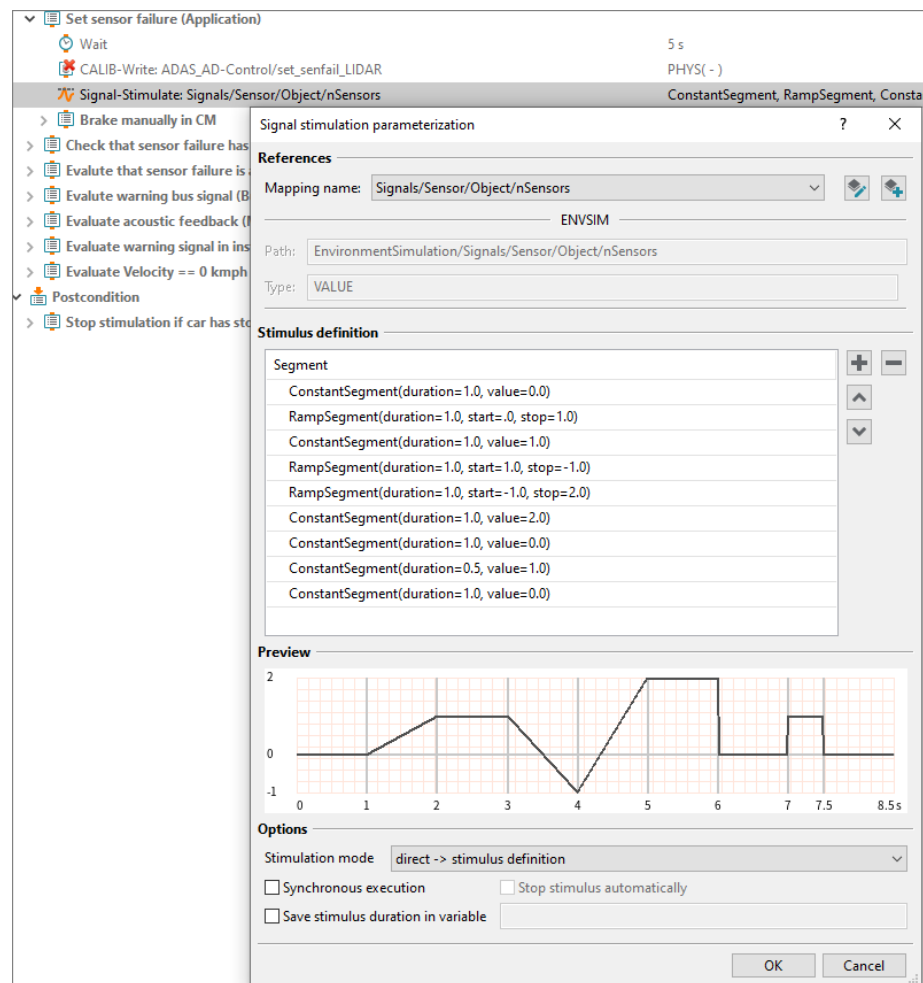


Figure 16: Real-time capability of stimulus steps

ModelDesk: Performance optimization when activating and downloading scenarios and roads



Previously, when executing the **Activate** and **Download** tool jobs, the entire experiment was saved. For larger experiments, this resulted in long execution times for these steps.

With **ecu.test 2025.3**, experiments are saved when changes are actually made, for example, by setting values. Without changes to the experiment, the **Activate** and **Download** tool jobs are now executed much faster.

3.2 Multimedia

Object SFE: Selenium: Direct interaction with WebDriver object



Access to the **WebDriver** object of the Selenium API is provided by the new tool job

- **GetWebDriver**

For use cases not covered by our existing jobs, this access provides direct access to the Selenium API and thus an unguided but broader range of functions.

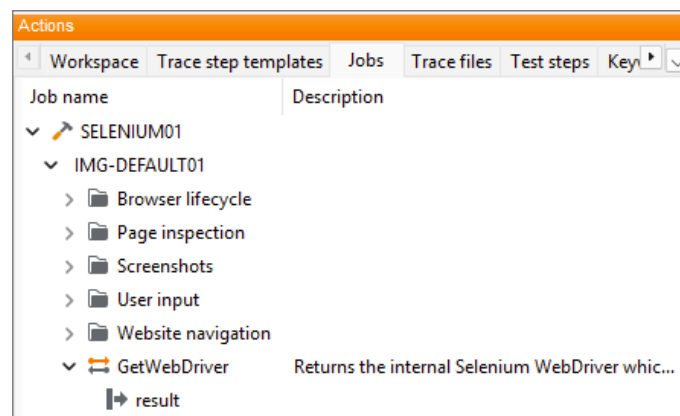


Abbildung 17: Job „GetWebDriver“ for interaction with the WebDriver object

EMVA: GenICam is available again



The incompatibilities have been resolved. GenICam is therefore available again with the same range of functions as in **ecu.test 2024.4**.

tracetronic: Multimedia: Audio port for RTP streams



Our internal tracetronic: Multimedia connection now offers a new port for reading and recording RTP audio streams.
Configuration is done via an SDP file. The stream can then be used in test cases in the same way as other audio sources.

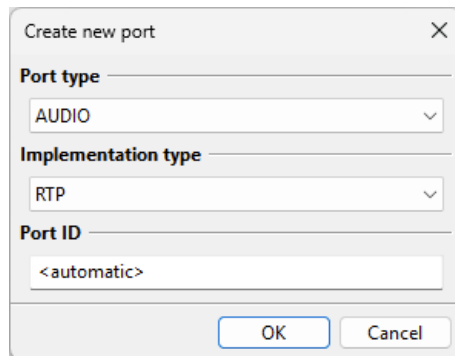


Abbildung 18: Audio port for RTP streams

New "FilterByText" method on TextMatchList



When using OCR algorithms other than the supplied **TesseractOCR**, there was previously no method for searching a list of texts found in the image for the occurrence of a specific text.

This option has now been created with **FilterByText** on the **TextMatchList**.

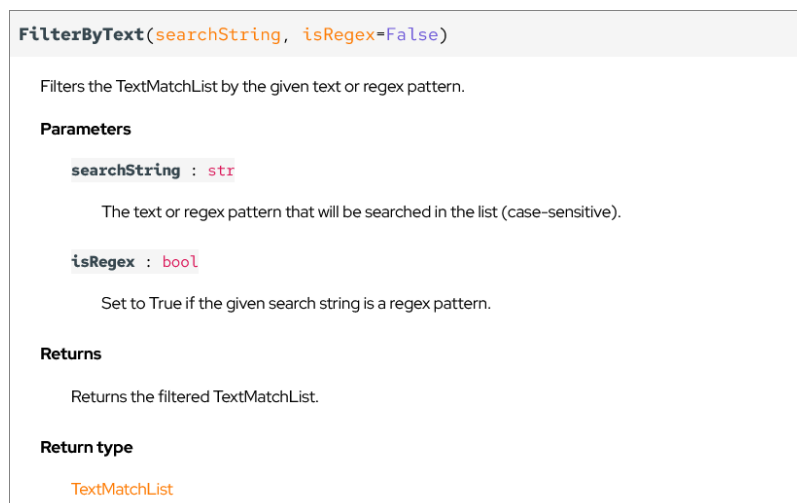


Figure 19: „FilterByText“ method

ADB: Screen-Recording



Tool connection to Google: ADB now also supports video recording via signal recording in the package.

This means that continuous tests can now also be performed as part of trace analysis.

A current Android version (version 11/API level 30) is required for video recording.

3.3 SiL

Support for the FMI Layered Standard for Network Communication (FMI-LS-Bus)



Starting with version 2025.3, **ecu.test** supports the FMI Layered Standard (LS) Bus. With the new port type **CAN**, communication with the FMU now takes place automatically on the basis of the LS (low-cut part).

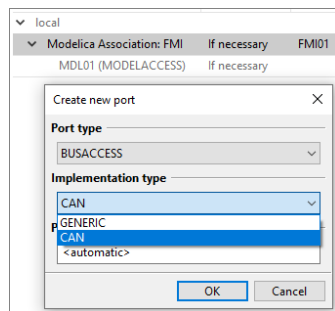


Figure 20: Creating a new CAN port for Tool FMI that uses the LS Bus

Users benefit from the comprehensive and tool-independent bus functionality that **ecu.test** has been offering for many years in the HIL and SiL areas, now also when using FMI. This closes the gap between model-oriented (unit) tests and complex integration tests.

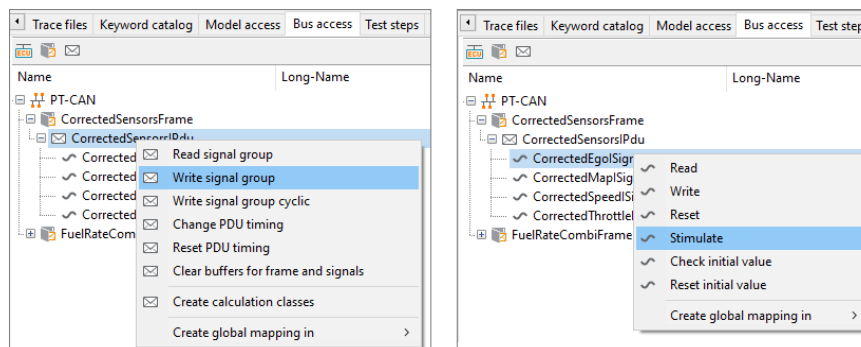


Figure 21: Bus access capabilities

Support for the FMI Layered Standard for XCP (FMI-LS-XCP)



In addition to the LS Bus, the Layered Standard for XCP can now also be used in **ecu.test 2025.3**. Together with the extra **ecu.test calibration**, an XCP server integrated in accordance with LS XCP can be used from within **ecu.test**.

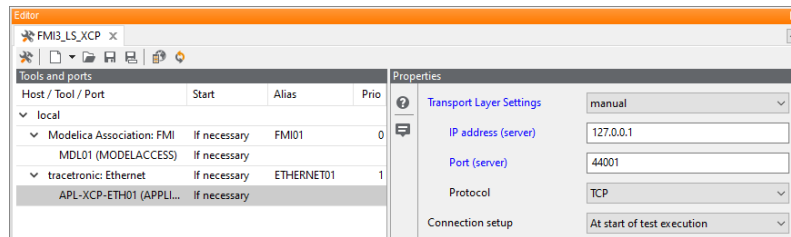


Figure 22: Setup of **ecu.test calibration** for use with FMI LS XCP

Setup and usage are similar to other measurement and calibration tools. Thanks to LS XCP and **ecu.test calibration**, ECU software can be used and tested in a practical way at an early stage of development without the need for additional tools.

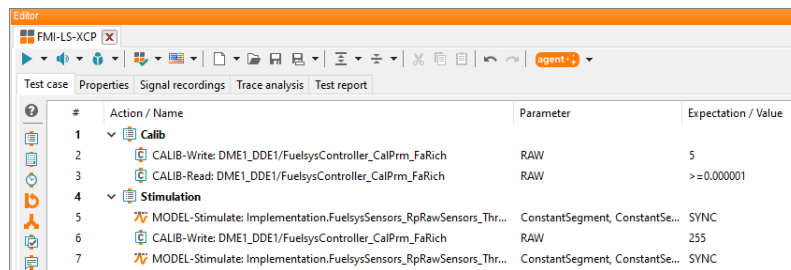


Figure 23: Typical test case that represents both model and calibration accesses.

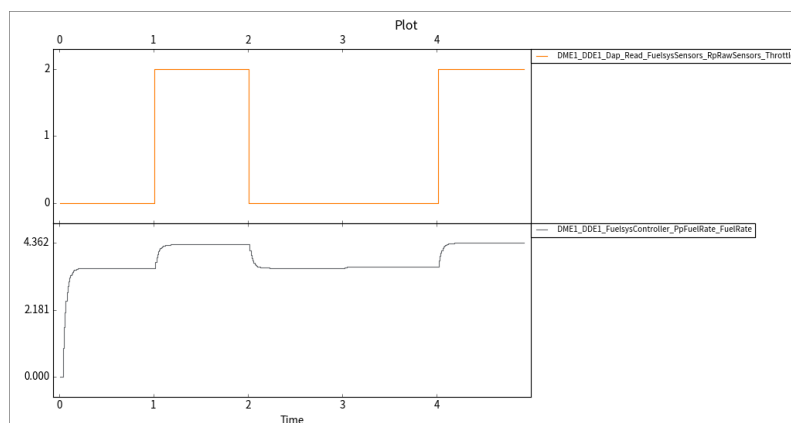


Figure 24: Example of a trace analysis

Setup and usage were described in detail in the user documentation.

Synopsys: Silver support for Linux



The Synopsys: Silver tool connection is now also available for Linux.

This should support the desire to shift testing efforts to Linux environments, although SilverSim is not yet supported. A full Silver license and execution of the graphical user interface are therefore still required.

Note: The Synopsys: SilverXIL connection is still only available for Windows.

3.4 HiL

CANape: Bus access for socket adapter



To enable access to Ethernet sizes via CANape, the bus port in **ecu.test** can now also be used to access the socket adapter protocol.

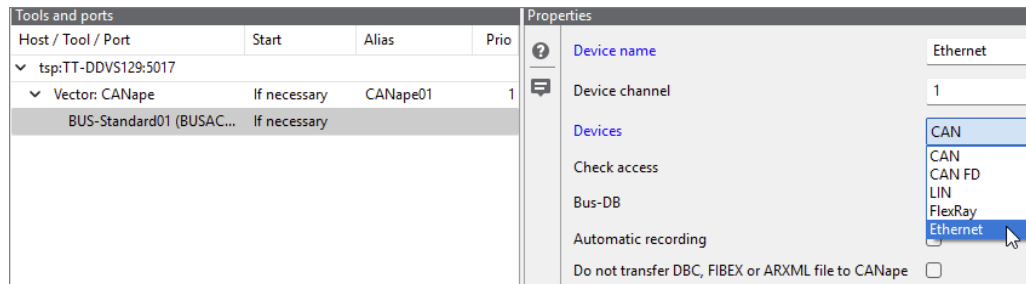


Figure 25: Access to Ethernet sizes via the bus port

dSPACE: ControlDesk: Service_Manager Port



The port now offers the option of recording scalar leaf nodes in signal acquisition, enabling downstream analysis of the signal curve in trace analysis. In addition, the port now supports values in the form of string arrays in UTF8 and UTF16. These can now be read and written as text.

dSPACE: ControlDesk: Automatic Model Reset



We now offer a new property on the model port to automatically reset the model when the test starts. This option is intended to contribute to the high-performance provision of a freshly initialized environment, particularly in SIL environments, so that tests can be performed independently of the context and without time-consuming resetting of simulations.

dSPACE: ControlDesk: Optional overwrite for ImportDCMFile



Previously, **ecu.test** threw an error when loading a DCM/CDFX file if one with the same file name was already loaded. This behavior can now be configured in the job. In addition to the existing behavior (default), **replace** and **(do) nothing** are now available as options.

INCA: Restoring the experiment after test case execution



To enable preconfigured experiments to be combined with **ecu.test** test cases, an INCA experiment is now restored after test case execution to the state it was in before the test case was executed. This includes restoring:

- the test variables
- the selected measurement rasters
- the display/recording status

3.5 Test management

Jama: Extension of the sample workflow



With **ecu.test 2025.2**, the new user test management API was introduced and an initial sample workflow was created for the ALM tool Jama.

This workflow has now been revised: Test execution can be planned in Jama directly via TestCycles or, more generally, via TestPlans. Since TestPlans represent the more generic approach, the **ecu.test** project import has been changed from a pure TestCycle import to a TestPlan import.

ecu.test user test management API: Sample workflow for Polarion



The new user test management API was introduced with **ecu.test 2025.2**. It enables **ecu.test** to be fully connected to any ALM system – exclusively in Python.

An initial sample workflow is now available for the ALM Polarion. This imports work items from Polarion to **ecu.test**. The workflow uses the Polarion REST API and accesses the data securely and without a password via a personal access token (PAT).

This new solution can be easily adapted to the individual configuration of the Polarion instance.

3.6 ecu.test calibration

Enable selection of transport layer from an A2L file



An A2L file can contain different configurations for different transport layers. To ensure that the correct parameters are used, the transport layer can now be selected.

The selection of a transport layer is required for the use of **ecu.test calibration** so that the correct connection settings can be determined. For other measurement and calibration tools, the selection has no effect and the default value <ALL> can be retained.

Selecting <ALL> provides all available measurement rasters.

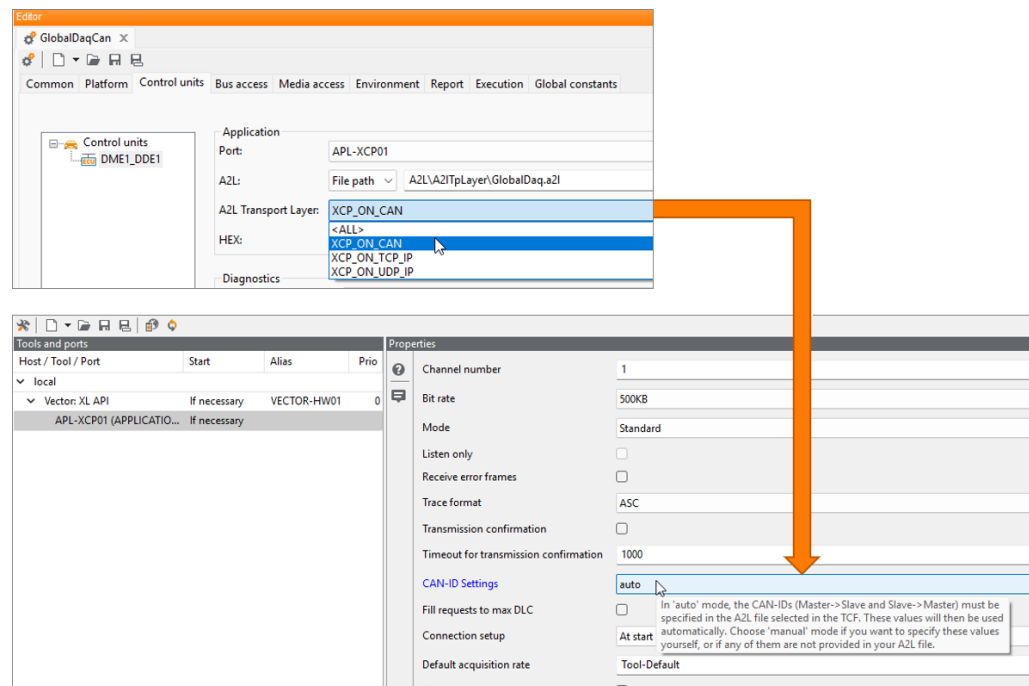


Abbildung 26: Selecting the transport layer from an A2L file

3.7 ecu.test diagnostics

Comprehensive improvements in the use of symbolic diagnostic test steps



Some diagnostic services are associated with complex parameters and return values. These can include deeply nested structures and arrays, making it challenging to fully capture them.

With version **2025.3**, we now list all leaf nodes in the parameter block that the user can fill in. If a default value is specified in the database, it is pre-selected. Additionally, there is a button for auto-completion. This feature is particularly beneficial for existing test steps.

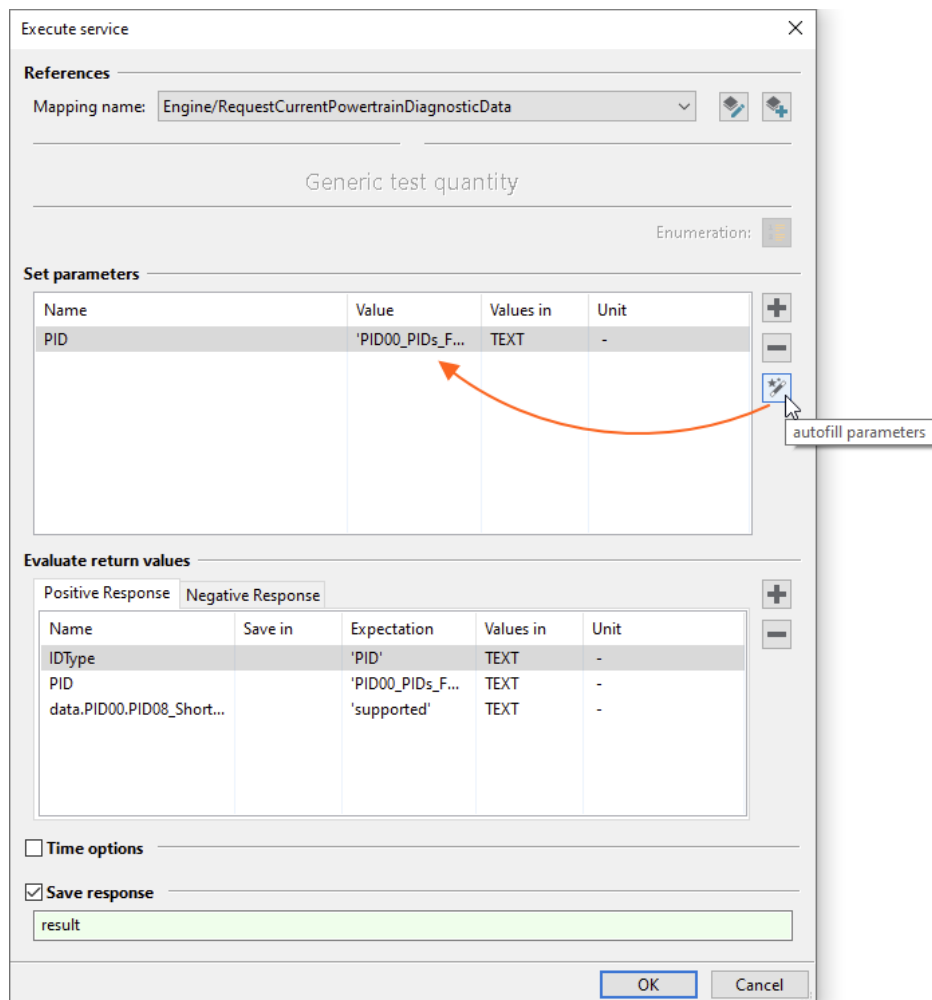


Figure 27: New auto-complete button

The reports have also been revised. A new section titled **Available return values** now lists all leaf nodes from the response. This provides a clearer overview, especially for complex return values.

| Evaluation | Test time [s] | Action/Name |
|---|---------------|---|
| SUCCESS | 0.004 | OBD-SERVICE (0x22 - Read Data By Identifier): Engine/RequestCurrentPowertrainDiagnosticData |
| DIAG-SERVICE Value Request 22:F4:00 Response 62:F4:00:BF:BF:A8:93 | | |
| Constants Value Representation ServiceID 0x22 PHYS IDType PID TEXT | | |
| Input parameters Value Expression Representation Unit PID 'PID00_PIDs_F401_to_F420_supportedStatus' 'PID00_PIDs_F401_to_F420_supportedStatus' TEXT - | | |
| Return value Value Expected value Expectation expression IDType 'PID' 'PID' 'PID' PID 'PID00_PIDs_F401_to_F420_supportedStatus' 'PID00_PIDs_F401_to_F420_supportedStatus' 'PID00_PIDs_F401_to_F420_supportedStatus' data.PID00.PID08_ShortTermFuelTrimBank2 'supported' 'supported' 'supported' | | |
| Response Expected Response Evaluation Positive Response Positive Response SUCCESS | | |
| Available return values ServiceID BitStream(0x62, 8) 98 ~No Representation~ IDType BitStream(0x7A, 7) ~No Representation~ 'PID' PID BitStream(0x0, 9) ~No Representation~ 'PID00_PIDs_F401_to_F420_supportedStatus' data.PID00.PID01_MonitorStatusSinceDTCsCleared BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID02_DTCThatCausedRequiredFreezeFrameDataStorage BitStream(0x0, 1) ~No Representation~ 'not supported' data.PID00.PID03_FuelSystemStatus BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID04_CalculatedLoadValue BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID05_EngineCoolantTemperature BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID06_ShortTermFuelTrimBank1 BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID07_LongTermFuelTrimBank1 BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID08_ShortTermFuelTrimBank2 BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID09_LongTermFuelTrimBank2 BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID0A_FuelRailPressureGauge BitStream(0x0, 1) ~No Representation~ 'not supported' data.PID00.PID0B_IntakeManifoldAbsolutePressure BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID0C_EngineRPM BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID0D_VehicleSpeedSensor BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID0E_IgnitionTimingSparkAdvanceForNo1Cylinder BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID0F_IntakeAirTemperature BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID10_AirFlowRateFromMassAirFlowSensor BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID11_AbsoluteThrottlePosition BitStream(0x1, 1) ~No Representation~ 'supported' data.PID00.PID12_CommandedSecondaryAirStatus BitStream(0x0, 1) ~No Representation~ 'not supported' data.PID00.PID13_LocationOfOxygenSensors1 BitStream(0x1, 1) ~No Representation~ 'supported' | | |

Figure 28: Revised report listing all leaf nodes from the response

J1939 from different recording formats



Options for checking diagnostic communication as part of trace analysis have been enhanced. Previously, J1939 could only be read from ASC recordings.

However, with **ecu.test 2025.3**, J1939 can now be read from BLF, GIN, MDF4, and TTL recordings as well.

Categorization of jobs for diagnostic ports



Over the past few years, a large number of jobs have been created for diagnostic ports, making the **job** list increasingly cluttered and less manageable over time. Starting with **version 2025.3**, the jobs will now be meaningfully categorized.

Additionally, the job descriptions have been shortened for clarity. Detailed descriptions remain accessible via tooltips or the help section.

| Workspace | Trace step templates | Trace files | Keyword catalog | Diagnostics | Test steps | Jobs |
|--|----------------------|-------------|-----------------|-------------|------------|------|
| Job name | | | Description | | | |
| ▼ ETHERNET01 | | | | | | |
| ▼ DIAG-01 | | | | | | |
| > 0x10 DiagnosticSessionControl | | | | | | |
| > 0x11 EcuReset | | | | | | |
| > 0x14 ClearDiagnosticInformation | | | | | | |
| > 0x19 ReadDTCInformation | | | | | | |
| > 0x22 ReadDataByIdentifier | | | | | | |
| > 0x23 ReadMemoryByAddress | | | | | | |
| > 0x27 SecurityAccess | | | | | | |
| > 0x29 Authentication | | | | | | |
| > 0x2E WriteDataByIdentifier | | | | | | |
| > 0x2F InputOutputControlByIdent... | | | | | | |
| > 0x31 RoutineControl | | | | | | |
| > 0x34 RequestDownload | | | | | | |
| > 0x36 TransferData | | | | | | |
| > 0x37 RequestTransferExit | | | | | | |
| > 0x3D WriteMemoryByAddress | | | | | | |
| > CallService Executes a diagnostic service request | | | | | | |
| > CallServiceFunctional Executes a diagnostic service request functionally | | | | | | |
| > CloseSession Closes a diagnostic session. | | | | | | |
| > OpenDiagSession Opens a UDS diagnostic session | | | | | | |
| > OpenTcfDiagSession Opens a diagnostic session configured in the TCF | | | | | | |
| > RecoverDiagnosticSessions Shutdown and reopens all open diagnostic sessions | | | | | | |
| > SetDiagTimeout Sets the timeout for the diagnostic session | | | | | | |
| > SetTargetAddress Changes the diagnostic target address of the given DoIP session | | | | | | |
| > SetTesterPresent Sets the parameters for cyclic TesterPresent messages | | | | | | |
| > ClearArpCache Clears all entries from the internal ARP/NDP cache | | | | | | |

Figure 29: Categorization of jobs for diagnostic ports

3.8 ecu.test lab

Improved Edit Workflow



Previously, the edit mode in lab had a two-step structure. It involved editing the size and arrangement of widgets, as well as editing individual widgets in a separate editor. These two actions are now combined into one.

The edit mode is still accessed via the gear icon in the title bar. Here, widgets can be inserted, moved, or scaled using drag and drop. Once a widget is selected, its configuration opens on the right side of the window. Now, changes to the widget's configuration and the "arrangement" of the widgets can be made in parallel.

If the widget's configuration is valid, selecting another widget opens its configuration. Clicking on an empty area closes the editor. If the configuration is invalid, the configuration remains open.

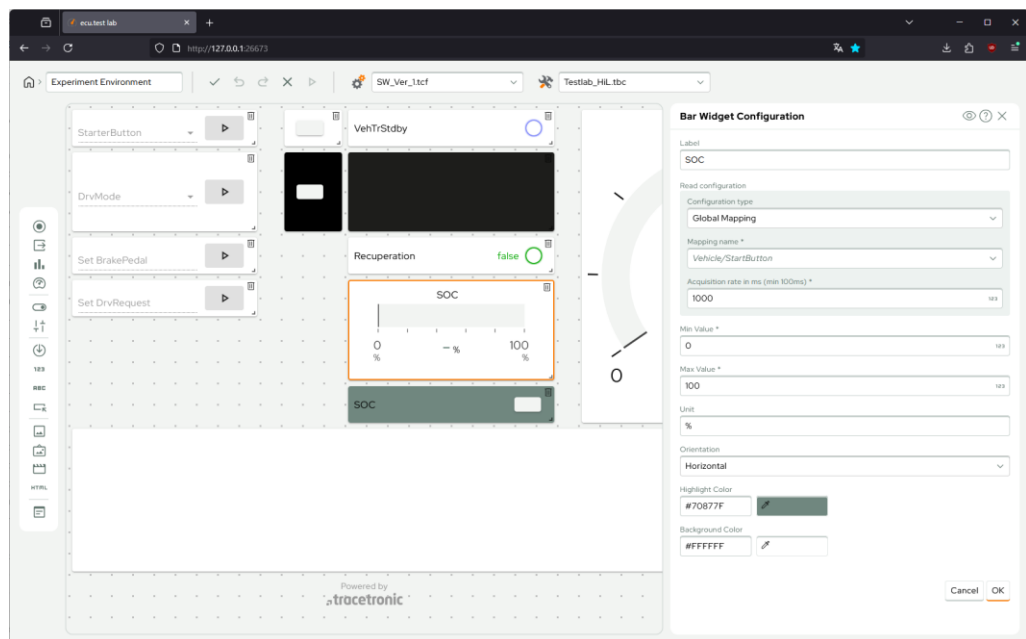


Figure 30: Configured dashboard in ecu.test lab

Undo/Redo



Changes to a View can now be undone and redone using the Undo/Redo buttons in the title bar.

Starting the Monitoring Mode during test execution



Until now, the Monitoring Mode in **ecu.test lab** could not be activated if a test execution was already running.

This is now possible, provided the View in lab was already open before the test execution started, or the View is set for auto-start in the workspace settings of lab.

New "Selection" widget



The **Selection** widget now allows defining text values for selection, which are displayed in the form of a dropdown.

If the tool interface only allows manipulation using numerical representations of the values and the described mapping via an Enum/Vtab/Texttable is available, an automatic conversion to the numerical value will occur.

New "Call Job" widget



The **Call** widget allows selecting a Job Mapping and executing the associated job. Unlike previous widgets, this widget's user interface is aligned with the job's interface.

In the widget's configuration, parameters can be selected that are to be filled in during execution via the widget GUI. The remaining parameters are statically parameterized in the configuration and automatically populated during execution.

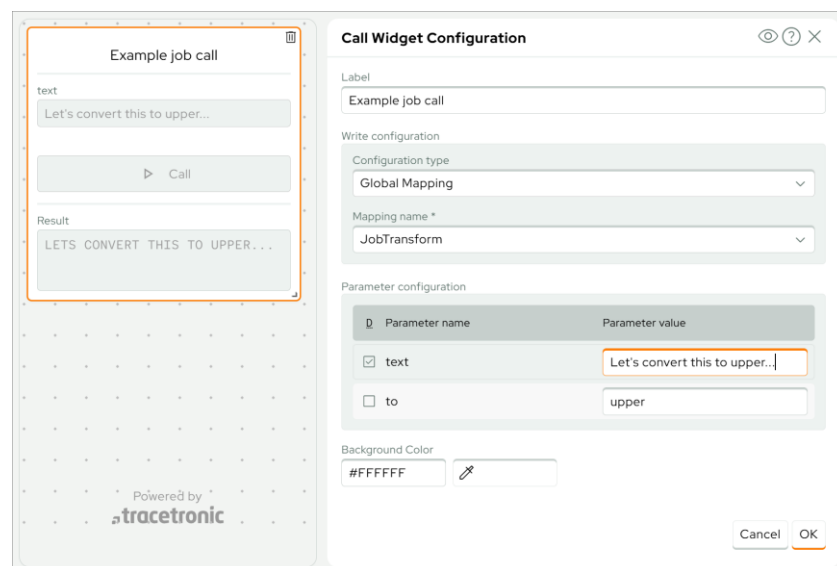


Figure 31: "Call Job" widget

3.9 Communication

New simulation feature for SOME/IP – Providing methods in test cases



With the newly created mechanism for offering methods, service simulation can now be activated quickly and easily.

You also benefit from the mapping options, which allow you to adjust paths when system descriptions change. The function can be used with the SERVICE-PCAP port on the tracetronic: Ethernet tool or the SERVICE port on the Vector: XL-API tool.

Currently, only fixed values can be defined for the method call. Dynamic calculation of return values will be available in one of the next versions.

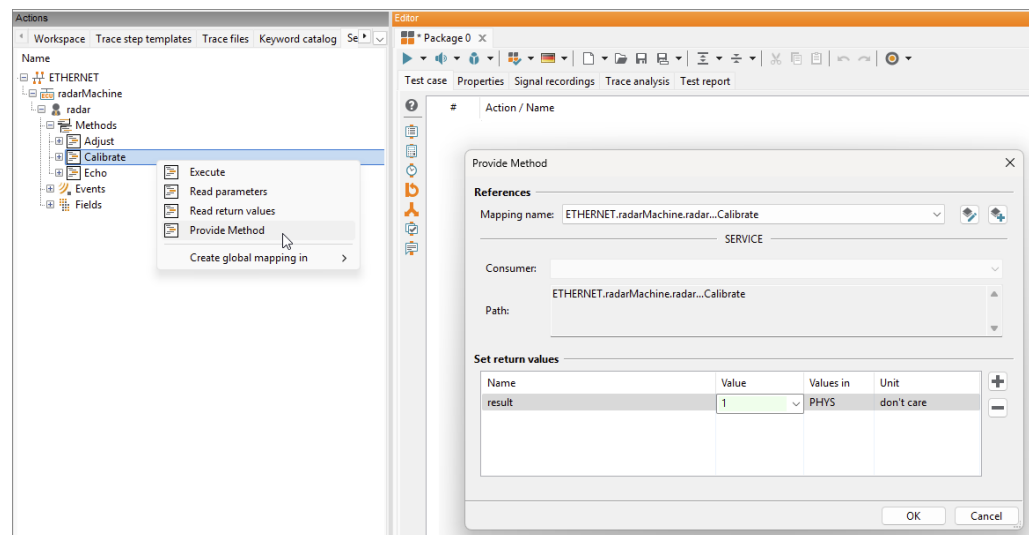


Figure 32: Provide methods in the test case

Hardware-related bus connections for CAN(-FD): New job for determining bus idle time



With the new port job **GetLastReceptionTimestamp**, available on all hardware-related CAN and CAN-FD ports, the timestamp of the last message received can be requested.

This can be used, for example, to determine whether and since when a bus has been inactive.

MACsec: Simpler configuration with fewer ports



The MACsec protocol can be used for cryptographic security of Ethernet vehicle network communication.

Previously, a port had to be created in the test bench configuration for each IP connection to be secured.

What is new is that a negotiated security association now affects all TCP/IP stacks with the same device and the same MAC address. The packet number counter is shared. This reduces the number of ports required for multiple IP connections.

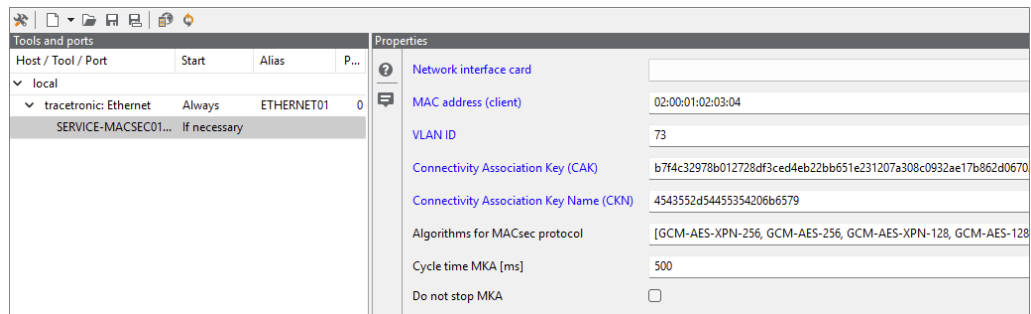


Figure 33: Simpler configuration of MACsec thanks to fewer ports

XL-API: New option for CAN-FD to disable buffering



If a CAN-FD bus becomes inactive, messages may remain in the send buffer of the Vector hardware. These messages are automatically sent by the hardware when the bus is reactivated. This behavior may be undesirable, as the data may be out of date.

With the new TBC option **Preempt send buffer**, which is only provided by the XL-API for CAN-FD, the use of the send buffer can be prevented

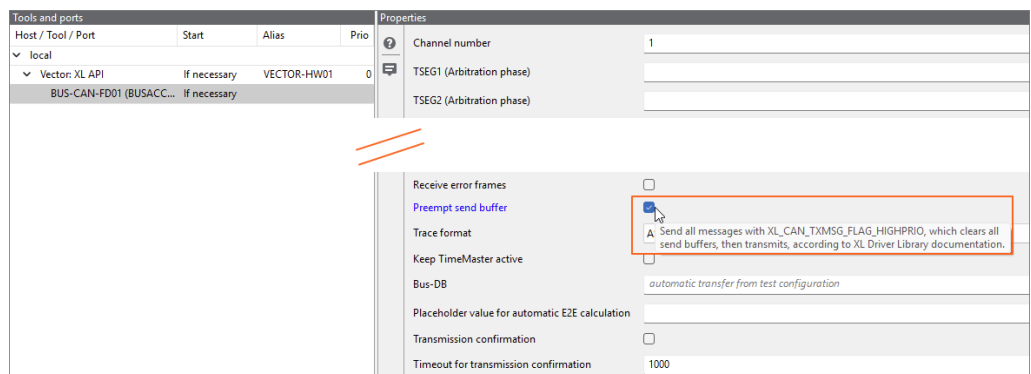


Figure 34: New option to disable CAN-FD buffering

XL-API: New job *GetLinkState*



The new port job on the Ethernet RAW port allows you to check the status of the XL network adapter's media connection.

3.10 Trace analysis

MDF4: Handling transposed matrices in XIL records



When interpreting multidimensional variables, **ecu.test** may interpret the order of the dimensions in reverse order to the tool connected via XIL. To ensure that the desired elements can still be accessed uniformly via index, a TBC option for transposing matrices was added in previous versions. Until now, however, this option only worked for test steps.

With **ecu.test 2025.3**, the TBC option for transposing matrices is now also taken into account for recorded matrices. It is important that the recording was made "live" during test execution via the tool adapter. Otherwise, this option is not applied when reading in signals for trace analysis.

MDF4: Improved parsing performance



The MDF4 parser has been optimized specifically for recordings with compression, allowing signals to be read up to 70% faster. Writing has also been accelerated by this optimization. Many tools write compressed MDF4 recordings as soon as MDF 4.1 or higher is selected.

3.11 trace.xplorer

Link to test case results from test.guide key indicator export



Since 2023, **test.guide** provides the option to export test case indicators to an ASTRACE file in order to visualize KPI data from numerous test runs in **trace.xplorer** in a high-performance manner and analyze it in detail.

However, if you wanted to access the details of the corresponding test run for an interesting data point, you had to identify it manually by comparing timestamps in **test.guide**. This process was cumbersome and prone to errors.

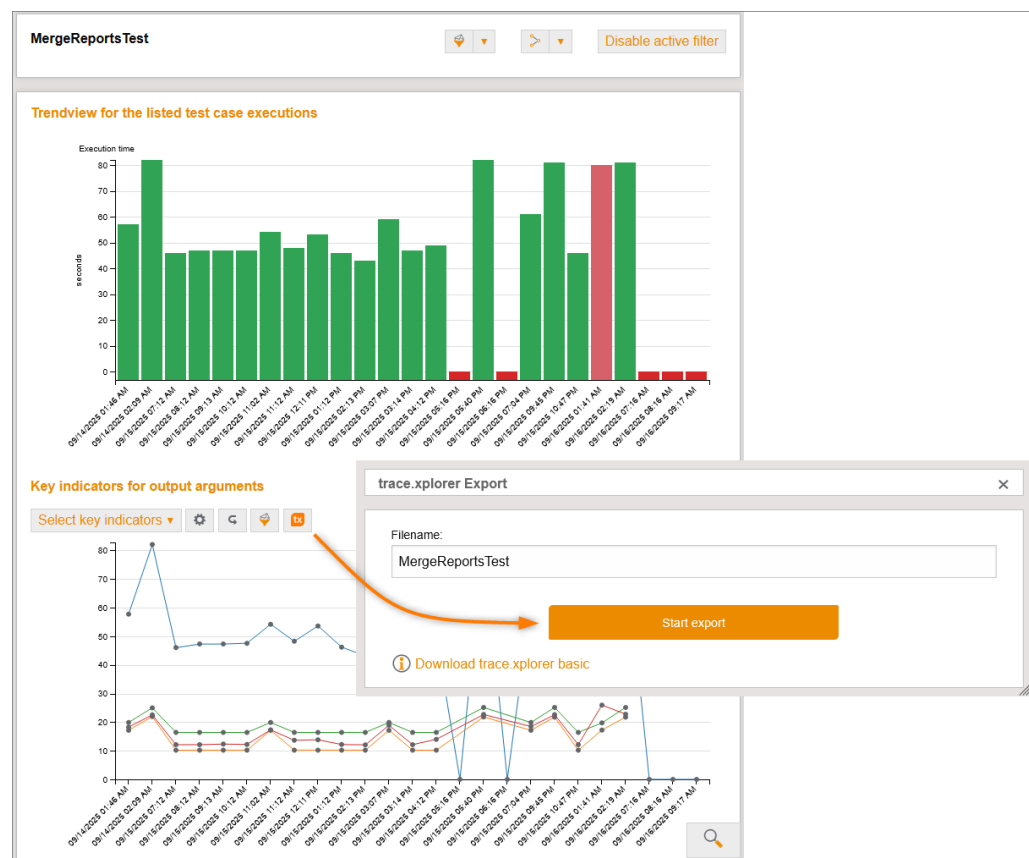


Figure 35: Exporting test case indicators from test.guide into a trace.xplorer file

From now on, the key indicator exports for each test case contain a new URL signal **Link to test case execution**, which stores a link to the results page in **test.guide** for each test run.

To access this page, **trace.xplorer** offers the new menu item **Open URL in Browser** in the context menu of the table view for URL signals. This takes you directly to the desired details of the test run without a long search.

For this to work, the **test.guide** instance used to create the ASTRACE document must be accessible. The display name of the stored links specifies the ID of the test run (TCE ... Test Case Execution) and can be used for additional comparison of information between **trace.xplorer** and **test.guide**.

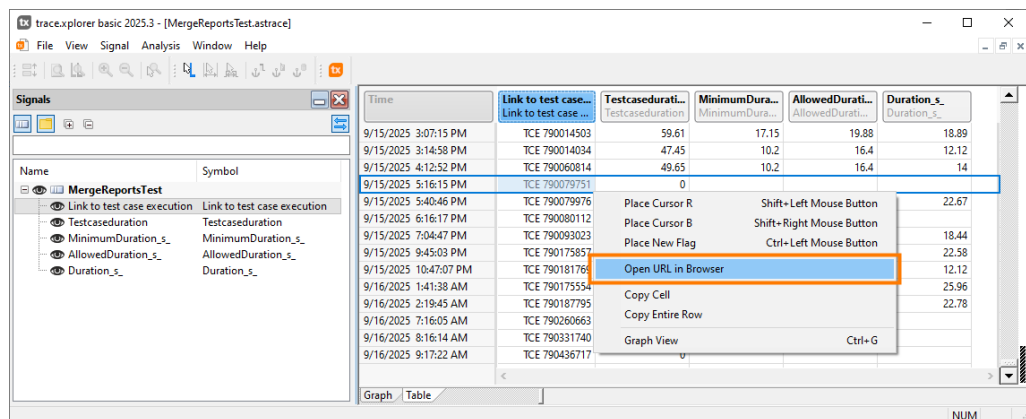


Figure 36: Opening the test.guide link in the table view of trace.xplorer

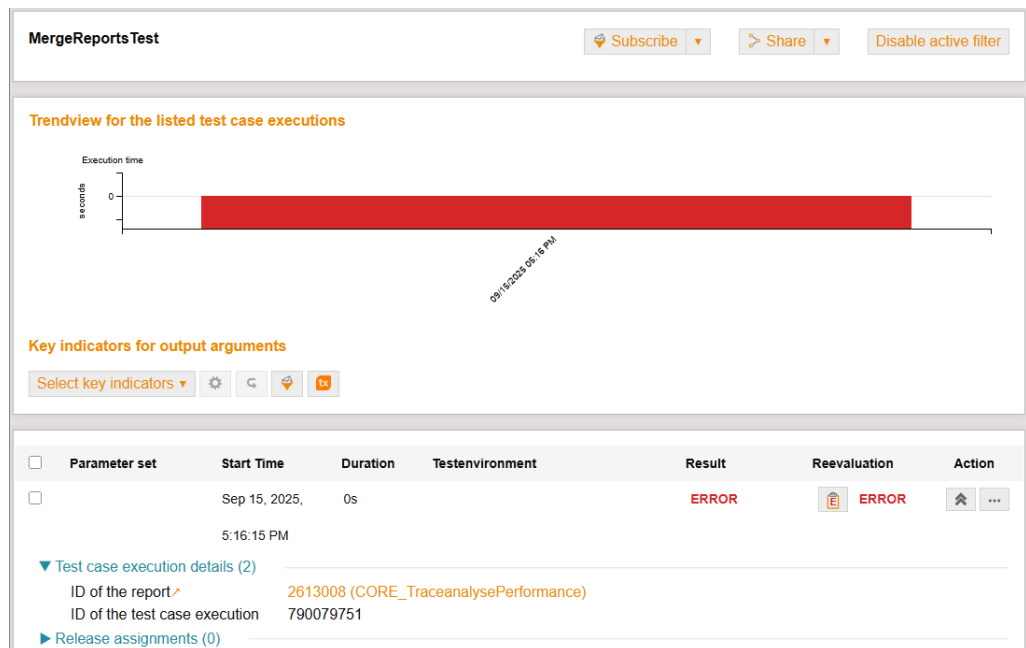


Figure 37: Jump directly from trace.xplorer to the test run result details in test.guide

Import Socket Adaptor signal data from PCAP files



Recordings of Socket Adaptor communication are stored in PCAP files. Previously, the **trace.xplorer** import filter for PCAP files could only import SOME/IP signal data and queried only the **Service** namespace. However, Socket Adaptor communication transmits CAN or FlexRay data and uses the **Bus access** namespace for this purpose. The import filter now also supports this namespace, allowing Socket Adapter signal data to be imported into ASTRACE documents.

Note: A **trace.xplorer** license is required to use this feature.

Alternative number representations for integer data types



Until now, all integer data types could be visualized either decimally or hexadecimally. However, if an integer signal actually represents individual flags, it would have been very helpful to see the values in binary representation, for example. This is now possible.

The formatting of the text output of integer signal values can be configured individually for each signal in the **Properties** toolbox. You can select the number system (**binary/decimal/hexadecimal/octal**) and whether the corresponding prefix should be displayed. Digits can be grouped for better readability. The formatting affects the table view, the **Measurements** toolbox, and the CSV export.

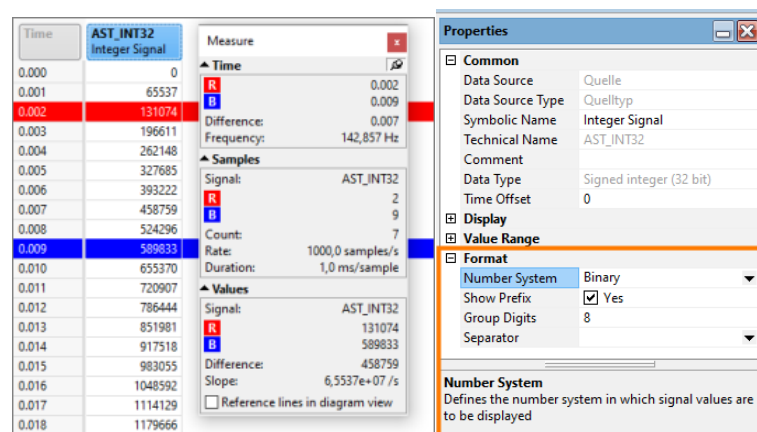


Abbildung 38: Setting the individual formatting of the text output of integer signals

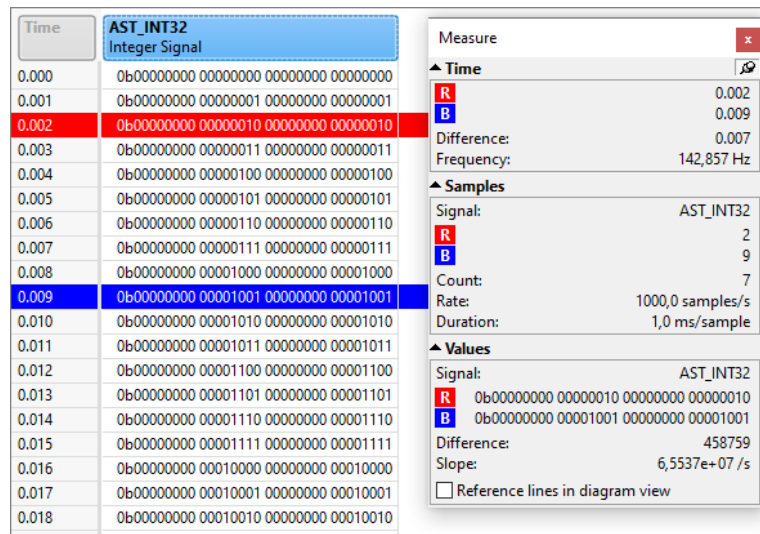


Figure 39: Individual formatting of the text output of integer signals

Minor usability improvements



New option in workspace settings

The preferred signal viewer is now selected per workstation, not per workspace, as is the licensing (of the **trace.xplorer**).

Therefore, we have moved the relevant settings:

- The previous category **Signal visualization** no longer exists.
- Choosing the default signal viewer, **trace.xplorer** or **trace.xplorer basic**, is now a local setting in the new section **Included programs**.

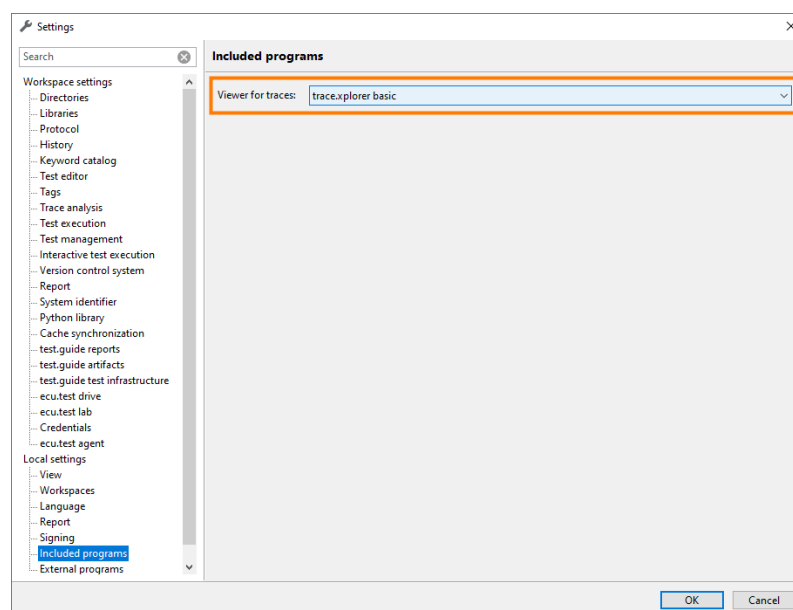


Figure 40: Setting for the default signal viewer

- All other options of the previous **Signal visualization** category have been moved to the **Trace analysis** section.

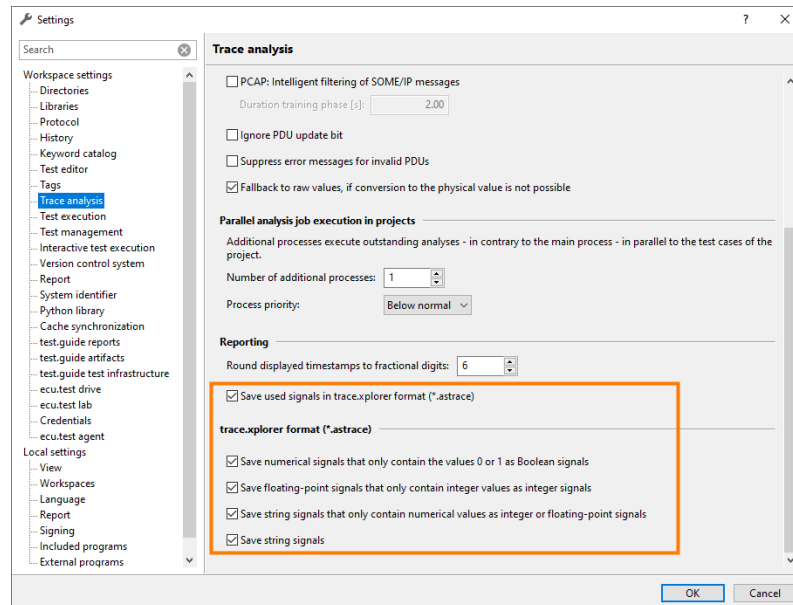


Figure 41: Settings for the trace.xplorer file format

Keep trace.xplorer in foreground

When working with many windows open at the same time, the **trace.xplorer** quickly gets pushed into the background.

If you want to keep it in sight, there is now an additional command in the system menu of the application window that allows you to permanently show the window in the foreground. The menu can be accessed by right-clicking on the title bar.

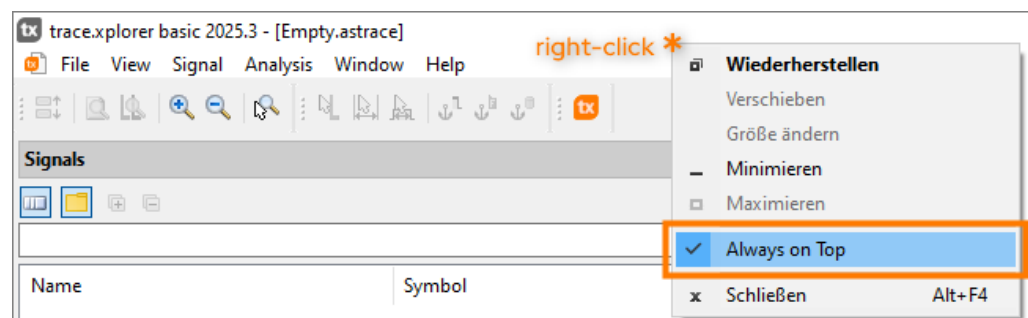


Figure 42: Show trace.xplorer application window always topmost

Expand/collapse data sources and signal groups

ASTRACE documents with many data sources or deeply nested signal groups were previously difficult to handle because there was no option in the **Signals** toolbox to expand or collapse entire branches at once. Instead, users were forced to make many individual clicks.

Such documents are created, for example, when exporting key indicators from **test.guide** or when importing Ethernet signals.

Now, entries for data sources and signal groups can be easily expanded and collapsed using their context menu or the new functions in the toolbar.

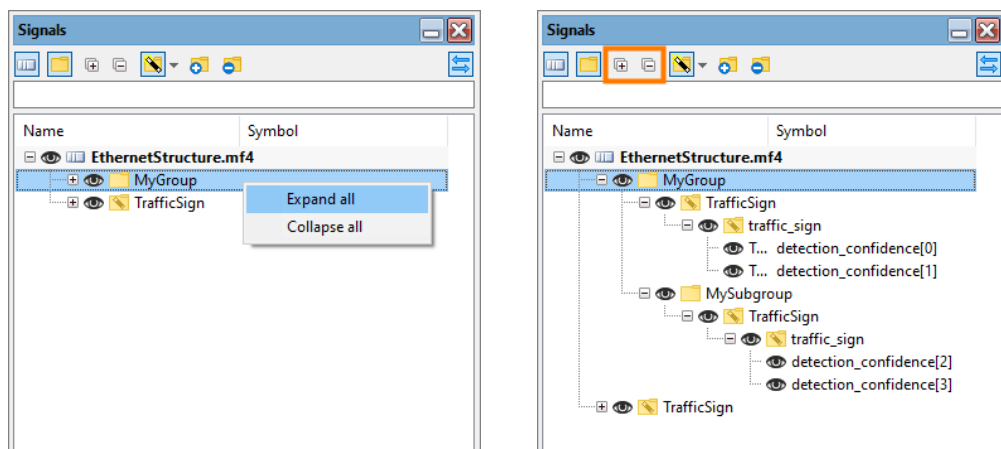


Figure 43: Fully expand and collapse data sources and signal groups using the context menu (left) or toolbar (right).

4 Tools and Interfaces

4.1 New tool versions

| | Provider | Website | System | Product name | Version |
|---|-----------------|------------------------------|--|-------------------------|------------------|
| 1 | dSPACE | Release link | Software for integrated control unit development | ControlDesk | 2025-A |
| 2 | MOTEON | Link | Testing and measurement of embedded systems | Motor Test Bench | |
| 3 | Synopsys | Link | Software-in-the-Loop solution for virtual ECUs | Silver | W-2025.09 |

4.2 APIs

4.2.1 Internal API

ToolAccess API for Executing Jobs



Until now, it was necessary to create a mapping in the package to call a job. A generic call to a job on any tool/port was not possible, as all mappings had to be created before the test case started.

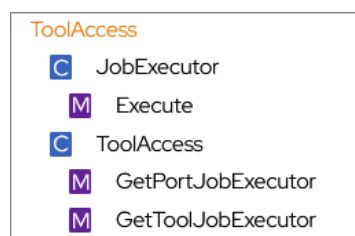


Abbildung 44: ToolAccess API

With the ToolAccess API, it is now possible to call all jobs of a tool or port directly via the API.

Creating a mapping is no longer required.

Note: The ToolAccess API is part of the Internal API.

4.2.2 REST API

DELETE endpoint no longer cancels postcondition



If the REST API endpoint DELETE is called during the execution of a postcondition, the running actions of the postcondition are not canceled. When a play-book is canceled manually or automatically in **test.guide**, this ensures that the test bench or environment is reset to a clean state via the postconditions.

4.2.3 UserTool

Differentiation between internal property name and display name



Previously, the same identifier was used for a property in the implementation as well as for display in the test bench configuration. This significantly limited the design freedom for user guidance in the editor.

This distinction is now possible by defining an explicit **DisplayName**"

5 Discontinuations

5.1 Discontinued features and incompatibilities in this version

Revised terminology



To make the user experience more intuitive, a number of established terms have been changed. The new terms provide greater clarity, ensure consistent language, and make it easier to navigate the entire tool landscape.

The changes affect GUI elements in **ecu.test** itself, as well as file names, user help, and API names.

| | Old term | New term |
|-------------------|---|---|
| Tool | diff License-Manager Signal editor Signal viewer TRF viewer/Report viewer Tool server ecu.test-Runner | ecu.test Diff Viewer 2025.3 ecu.test License Manager 2025.3 ecu.test Signal Editor 2025.3 ecu.test Signal Viewer 2025.3 ecu.test Report Viewer 2025.3 ecu.test Tool Server 2025.3 ecu.test Runner 2025.3 |
| Datei-name | Tool-Server.exe tool-server tool-server_runner Tool-Server_out.log LicenseManager.exe TRF-Viewer.exe | tool_server.exe (W) tool_server (L) tool_server_runner (L) tool_server_out.log (W+L) license_manager.exe (W) report_viewer.exe (W) |
| API-Name | Project parameter generator API Project package generator API Project generator API Model based bus port API Advanced properties of bus related objects Advanced properties of diagnostics related objects Advanced properties of media related objects Advanced properties of DLT logging related objects | Parameter Set Generator API Package Generator API Project Generator API Model-based Bus Port API Advanced properties of bus-related objects Advanced properties of diagnostics-related objects Advanced properties of media-related objects Advanced properties of DLT logging-related objects |

*W=Windows, *L= Linux

Note: The version information depends on the release. Parts not mentioned keep their names (e.g. "**ecu.test.exe**" or other APIs).

Alternative call representation" of Package properties



The **Alternative call representation** in the general properties of a package has been removed in **ecu.test 2025.3**.

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.

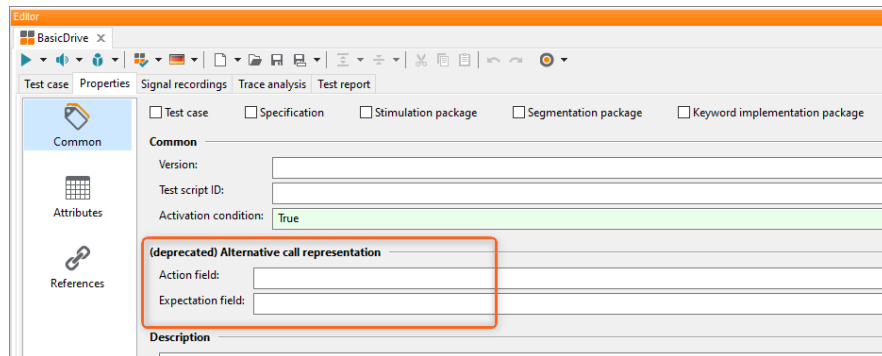


Figure 45: Package features: Alternative call representation

ODX Migration Helper



Generic diagnostic test steps were introduced in version 2023.4 for the use of ODX/PDX databases. These provide all ODX/PDX services.

To simplify the transition, a migration helper was offered to convert existing packages to the new test steps. This was removed with version 2025.3.

DiagBrowser: GetUdsName is permanently replaced by GetServiceName



The method for querying the service name in DiagBrowser **GetUdsName** is been permanently removed.

The following method is available as an alternative:

- **GetServiceName**

5.2 Discontinued features in future versions

KS: Tornado only via ASAM ACI



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



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



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



The permanently integrated connection to Jama in **ecu.test** will be removed with **ecu.test 2026.2**. Please use the new Python-based connection instead.

Discontinuation of the ReqIf Import



With **ecu.test 2025.4**, the functionality for importing ReqIf data as a package and project attribute will be removed.

Alternative report directory for separate subproject execution



Currently, it is possible to specify the report folder for separate subproject execution. This feature is marked as deprecated in version **2025.3** and will be removed in **ecu.test 2026.1**.

Discontinuation of the FEP2 connection



The FEP2 connection will be removed in **ecu.test 2026.1**.

Playbook-Export from ecu.test to test.guide



The playbook export from **ecu.test** to **test.guide** will be removed in **ecu.test 2026.1**.

Port BUSACCESS – GENERIC_MAPPINGFILE



The newly introduced port type, "BUSACCESS – MODEL BASED," is much more flexible, maintainable, and intuitive to configure. To clarify the available options and guide users to the best solution, we are removing "BUSACCESS – GENERIC_MAPPINGFILE" in **ecu.test 2026.1**.

Support for Ubuntu 20.04 LTS and 22.04 LTS



With **ecu.test 2026.1**, support for Ubuntu 20.04 LTS and 22.04 LTS will be discontinued.

Supported versions of MATLAB/Simulink in Linux



As part of the discontinuation of support for older versions of Ubuntu with **ecu.test 2026.1**, support for MATLAB/Simulink up to and including R2023b will be removed under Linux. These versions do not offer support for Ubuntu 20.04. Support for Windows remains unchanged for versions from R2015b onwards.

The ICMP-RAW port is being discontinued for the Ethernet, XL-API, and SIL-Kit tools.



The port will be removed with **ecu.test 2026.1**.

5.2.1 Removed API methods in future versions



| 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(). |