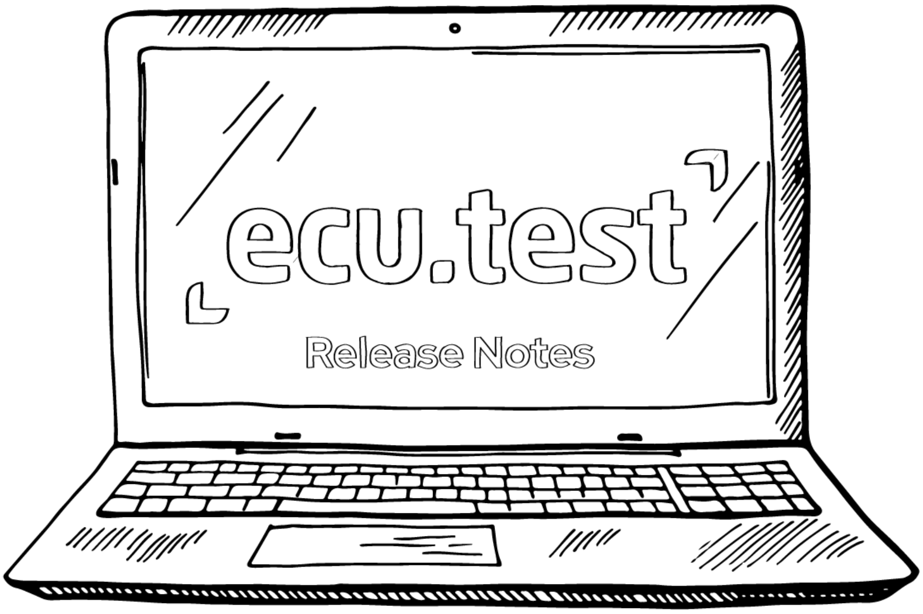


Release Notes

ecu.test 2026.1

trace.check 2026.1

Date: 03/03/2026

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

Content

Overview

ecu.test 2026.1 continues to drive forward modern testing processes for SDV, cloud, and AI scenarios: platform independence (Linux, ARM), significantly faster infrastructures, AI assistance for everyday use, and numerous improvements in the integration, analysis, and visualization of test data. The most important new features at a glance:

Environments & Infrastructure

- **ecu.test Linux GUI**
  Complete graphical user interface under Linux: testing without a Windows detour.
- **ecu.test Linux Runner & Runner for ARM (sneak preview)**
  Efficient test execution on Linux and ARM resources.

AI & Assistance

- **ecu.test** *agent*
  The agent can now be used to generate trace steps.
- **ecu.test** *agent* **chat (preview)**
  Planned co-pilot for workflows: pattern recognition, reviews, library packages, and collaboration with the **test.spec *agent***.
- **test.spec** *agent* **(Preview)**
  Generates test specifications from requirements.
- **TouchInput with AI prompt**
  HMI interactions via voice prompt instead of coordinates.

Analysis & Diagnostics

- **CheckResponse**
  This function facilitates time checks in trace analysis.
- **J1939 DM services (preview)**
  Symbolic, generic diagnostic test steps analogous to OBDonUDS, without additional database.

Everyday performance

- **Parallel tool start** shortens the start-up time of extensive TBCs.
- **CAN-FD Timing Helper** automatically determines missing timing parameters and simplifies CAN-FD configuration.
- **ecu.test** *code*
  - is now available via PyPI.
  - now enables direct package calls.

All other numerous new features and enhancements are described in detail below.

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

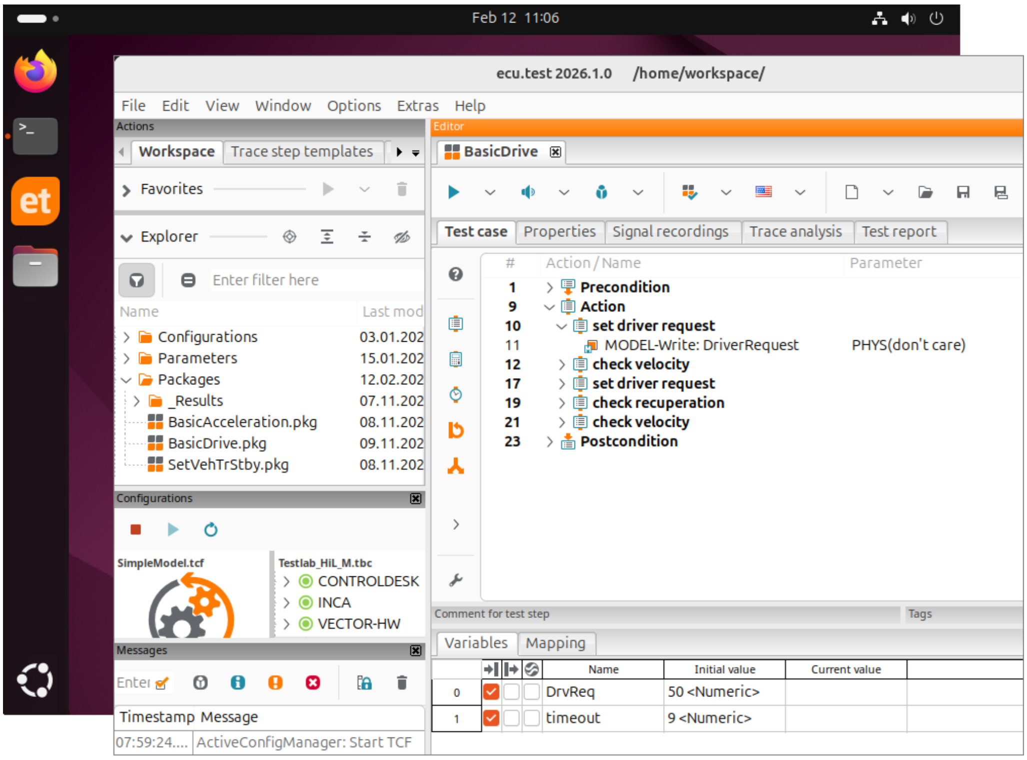
1 Highlights in ecu.test 2026.1

ecu.test Linux GUI

et

ecu.test is now available for Linux with a full graphical user interface, in addition to Windows. This allows users to create, execute, and evaluate tests directly in their Linux environment, without switching to Windows to make additional adjustments to tests and configurations.

The **ecu.test Linux GUI** is available for **Ubuntu 24.04** and, together with the **ecu.test Linux Runner**, simplifies development and validation while enabling more efficient use of existing Linux infrastructure.



Name	Last mod
Configurations	03.01.202
Parameters	15.01.202
Packages	12.02.202
_Results	07.11.202
BasicAcceleration.pkg	08.11.202
BasicDrive.pkg	09.11.202
SetVehTrStby.pkg	08.11.202

#	Action / Name	Parameter
1	Precondition	
9	Action	
10	set driver request	
11	MODEL-Write: DriverRequest	PHYS(don't care)
12	check velocity	
17	set driver request	
19	check recuperation	
21	check velocity	
23	Postcondition	

Testlab_HiL_M.tbc
- CONTROLDESK
- INCA
- VECTOR-HW

Timestamp	Message
07:59:24....	ActiveConfigManager: Start TCF

				Name	Initial value	Current value
0	☑	□	□	DrvReq	50 <Numeric>	
1	☑	□	□	timeout	9 <Numeric>	

Figure 1: ecu.test Linux GUI in Ubuntu 24.04

CAN-FD Timing Helper

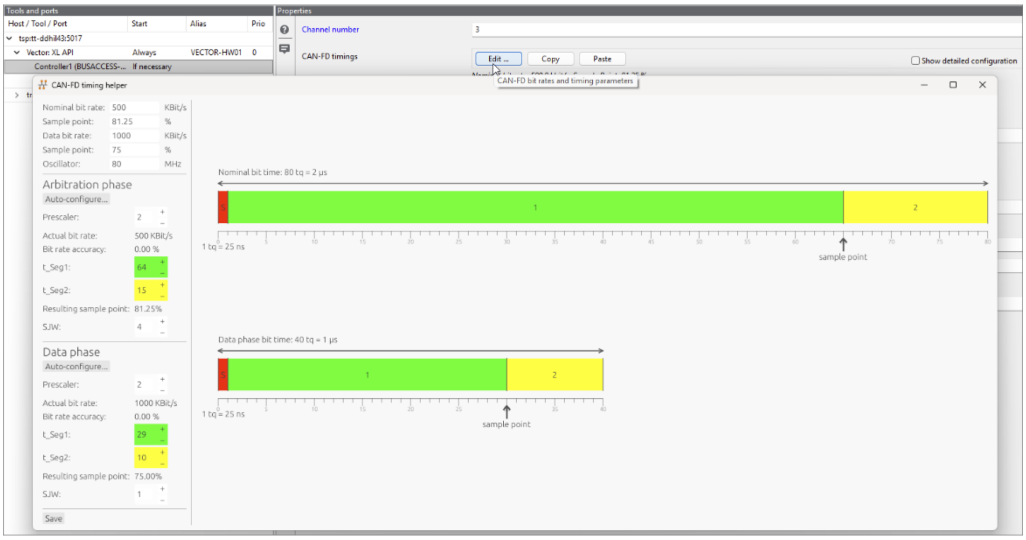
et

The new **CAN-FD Timing Helper** utility simplifies test bench configuration for CAN-FD-capable bus hardware.

Based on the parameterization of bit rates, sample points and oscillator frequency, the missing parameters – **synchronization jump** and **time** segments – are automatically determined and visualized.

Although manual parameterization in the TBC with deviating values is still possible, it is no longer mandatory.

In addition, once values have been determined, they can be copied as a whole at the touch of a button and pasted onto other ports.



Tools and ports

Host / Tool / Port	Start	Alias	Prio
tsp:tt-ddhil43:5017			
Vector: XL API	Always	VECTOR-HW01	0
Controller1 (BUSACCESS-...	If necessary		

Properties

Channel number: 3

CAN-FD timings: Edit ... Copy Paste

□ Show detailed configuration

CAN-FD bit rates and timing parameters

CAN-FD timing helper

Nominal bit rate: 500 KBit/s
Sample point: 81.25 %
Data bit rate: 1000 KBit/s
Sample point: 75 %
Oscillator: 80 MHz

Arbitration phase

Auto-configure...

Prescaler: 2
Actual bit rate: 500 KBit/s
Bit rate accuracy: 0.00 %
t_Seg1: 64
t_Seg2: 15
Resulting sample point: 81.25%
SJW: 4

Data phase

Auto-configure...

Prescaler: 2
Actual bit rate: 1000 KBit/s
Bit rate accuracy: 0.00 %
t_Seg1: 29
t_Seg2: 10
Resulting sample point: 75.00%
SJW: 1

Save

Nominal bit time: 80 tq = 2 µs
S
1
2
1 tq = 25 ns
sample point
Data phase bit time: 40 tq = 1 µs
S
1
2
1 tq = 25 ns
sample point

Figure 2: CAN-FD Timing Helper

Parallel tool start

et

Operating the test infrastructure involves high costs. The time spent configuring the test bench and its tools must be subtracted from the valuable test time and, although the configuration start is necessary, does not contribute to the actual tests.

Especially with extensive test bench configurations involving many tools, the start of the configuration can be time-consuming. The start-up time of a tool depends heavily on the configuration steps required, the scope of the configuration, and the individual implementation of the tool and its API.

Previously, all tools in a TBC were started sequentially. With **ecu.test 2026.1**, the tools are now started in parallel, which results in substantial time savings. The individual start-up priority set by the user for each tool is, of course, still taken into account.

ecu.test *code* now available on pypi.org

et

ecu.test *code* is now available via the Python Package Index PyPI.
Provided PyPI is accessible, installation is now very easy via**:**

- `pip install ecutest_code`

Provision via email and Seafile link will be discontinued at the same time.

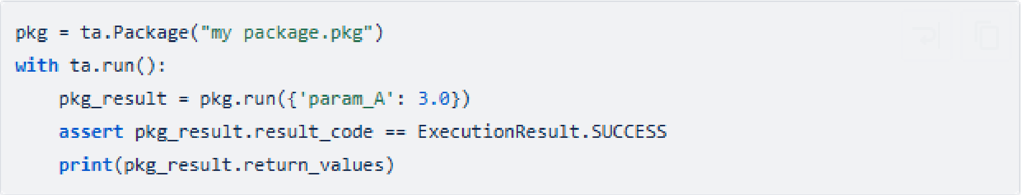
Package execution for ecu.test *code*

et

For certain tasks, helper packages might already exist, or expressing a subtask as a package might be more concise.

From now on, **ecu.test** *code* can call packages that respect certain limitations. Parameters can be passed in, return values can be passed out, and the evaluation of the package execution (SUCCESS, FAILED, etc.) can be queried.

Of course, package calls can also be used in conjunction with access to test signals.



```
pkg = ta.Package("my package.pkg")
with ta.run():
    pkg_result = pkg.run({'param_A': 3.0})
    assert pkg_result.result_code == ExecutionResult.SUCCESS
    print(pkg_result.return_values)
```

Figure 3: Package call in ecu.test *code*

CheckResponse simplifies time checks in trace analyses

et tc

Previously, it was difficult to model typical temporal relationships between two signals in trace analysis without using trace step templates.

The **CheckResponse** function is now available for calculation steps and trigger blocks. It significantly simplifies temporal checking by requiring only two signals and a *timeout* parameter.

If the value of the first signal is true at a given point in time, **CheckResponse** checks that the second signal also becomes true within the *timeout* period.
It is good practice to specify an edge function for the first signal, as the state change of a signal usually serves as the activation condition:

- **CheckResponse(RisingEdge(Sig1), Sig2 == 1, 0.5)**

The optional **useHoldValue** parameter can be used to address special cases where the second signal is rarely sampled and may have already been in the desired state before the activation time.

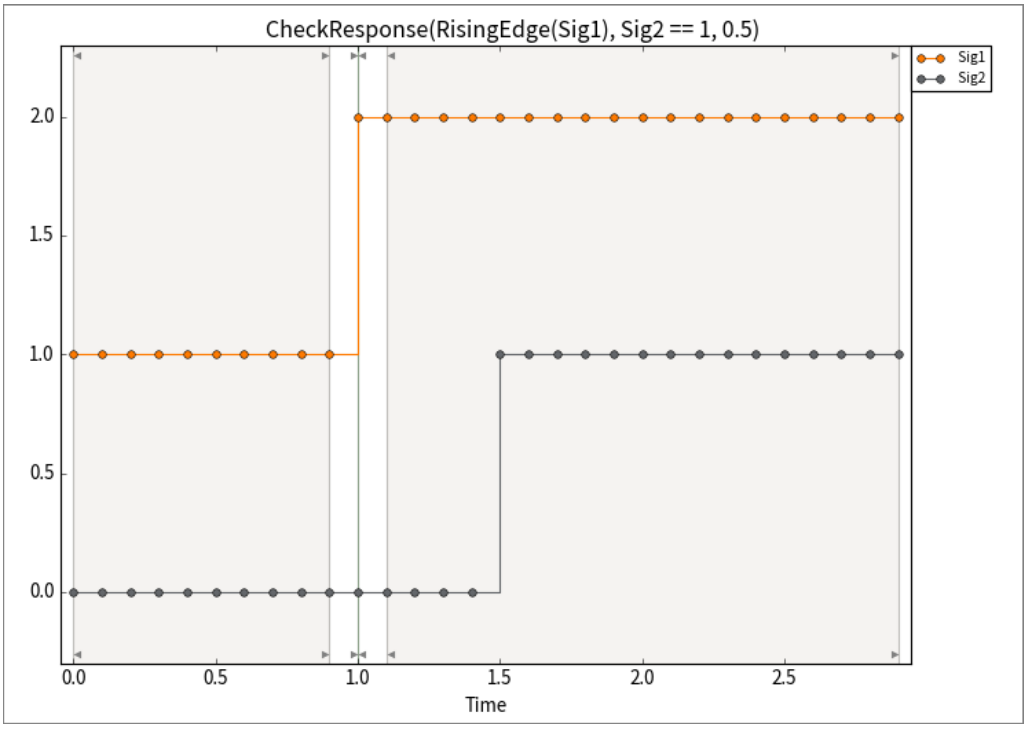


Figure 4: Check at rising edge of Sig1 that sig2 is 1 within 0.5s

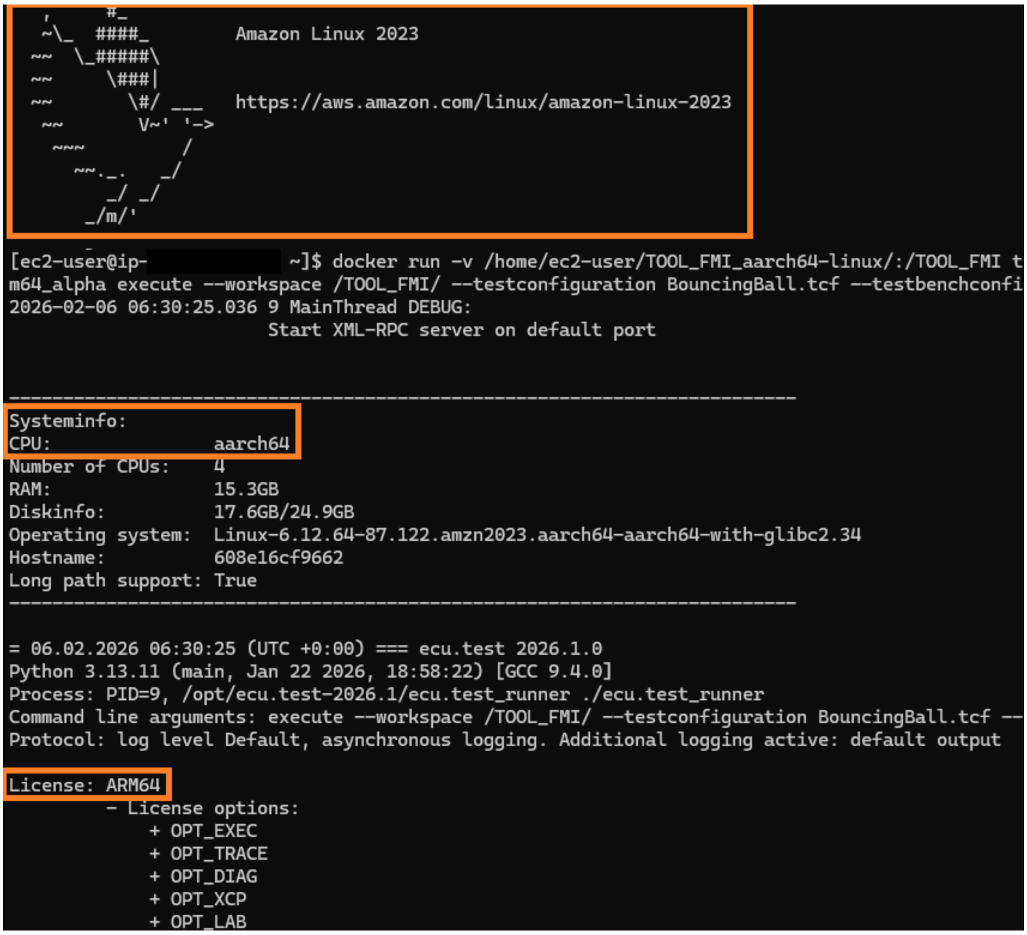
2 Sneak Preview

ecu.test Runner for ARM

ecu.test Linux Runner is now also available for ARM platforms. One of its key features is the efficient execution of FMU-based tests via FMI on ARM, enabling vECU Level 4 testing in an ARM-on-ARM scenario.

This allows users to test their virtual control units directly on ARM hardware – without the need for additional bridges or translation layers, making it particularly efficient.

Native use on ARM cores gives users better energy efficiency and lower infrastructure and operating costs – both in their own data centers and in the cloud. The **ecu.test Runner** can be run directly on ARM resources in AWS and other cloud environments, enabling a scalable, high-performance, and resource-efficient test environment.



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'

[ec2-user@ip-[redacted] ~]$ docker run -v /home/ec2-user/TOOL_FMI_aarch64-linux/:/TOOL_FMI t
m64_alpha execute --workspace /TOOL_FMI/ --testconfiguration BouncingBall.tcf --testbenchconfi
2026-02-06 06:30:25.036 9 MainThread DEBUG:
                        Start XML-RPC server on default port


-------------------------------------------------------------------------
Systeminfo:
CPU:                   aarch64
Number of CPUs:        4
RAM:                   15.3GB
Diskinfo:              17.6GB/24.9GB
Operating system:      Linux-6.12.64-87.122.amzn2023.aarch64-aarch64-with-glibc2.34
Hostname:              608e16cf9662
Long path support: True
-------------------------------------------------------------------------

= 06.02.2026 06:30:25 (UTC +0:00) === ecu.test 2026.1.0
Python 3.13.11 (main, Jan 22 2026, 18:58:22) [GCC 9.4.0]
Process: PID=9, /opt/ecu.test-2026.1/ecu.test_runner ./ecu.test_runner
Command line arguments: execute --workspace /TOOL_FMI/ --testconfiguration BouncingBall.tcf --
Protocol: log level Default, asynchronous logging. Additional logging active: default output

License: ARM64
         - License options:
             + OPT_EXEC
             + OPT_TRACE
             + OPT_DIAG
             + OPT_XCP
             + OPT_LAB
```

Figure 5: ecu.test Linux Runner on ARM

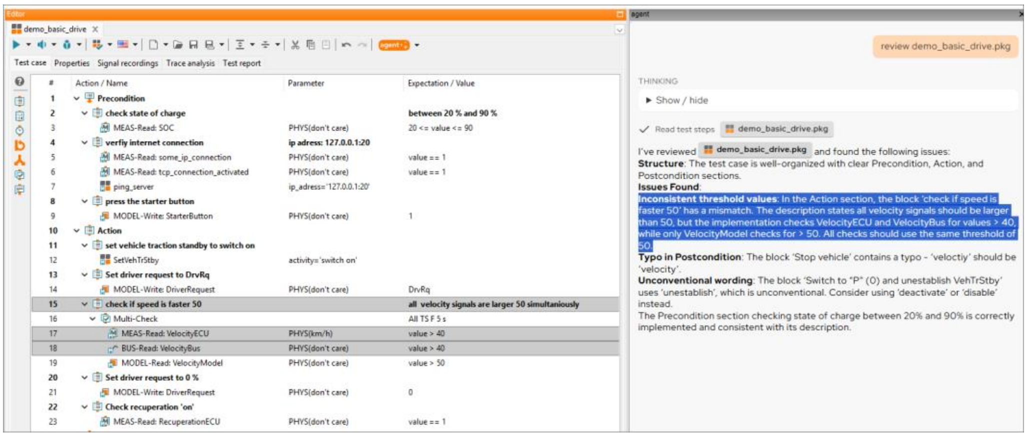
Additional use cases and application scenarios:

- Execution of package checks and generators
- Performing trace analyses and file conversions
- Operation on embedded hardware for dedicated test or analysis tasks or for test bench automation/monitoring (e.g., via **ecu.test** *lab*)

ecu.test *agent* chat – your ecu.test co-pilot

In the last two releases, we introduced the ecu.test *agent* and have been continuously improving it ever since. At the same time, we are working on the next big highlight – the **ecu.test** *agent* chat, a kind of co-pilot for **ecu.test**.

- Finding similar implementations in test cases and automated derivation of library packages
- Support for signal recording for optimized merging of test case and trace analysis generation
- Reviewing a test case: The agent points out shortcomings and differences between the specification and implementation
- Actions within the scope of an ecu.test project: e.g., adding parameter sets from attached files
- Communication with the test.spec agent so that new specifications can be derived from requirements



#	Action / Name	Parameter	Expectation / Value
1	**Precondition**		
2	**check state of charge**		**between 20 % and 90 %**
3	MEAS-Read: SOC	PHYS(don't care)	20 <= value <= 90
4	**verfiy internet connection**	**ip adress: 127.0.0.1:20**	
5	MEAS-Read: some_ip_connection	PHYS(don't care)	value == 1
6	MEAS-Read: tcp_connection_activated	PHYS(don't care)	value == 1
7	ping_server	ip_adress='127.0.0.1:20'	
8	**press the starter button**		
9	MODEL-Write: StarterButton	PHYS(don't care)	1
10	**Action**		
11	**set vehicle traction standby to switch on**		
12	SetVehTrStby	activity='switch on'	
13	**Set driver request to DrvRq**		
14	MODEL-Write: DriverRequest	PHYS(don't care)	DrvRq
15	**check if speed is faster 50**		**all velocity signals are larger 50 simultaniously**
16	Multi-Check		All TS F 5 s
17	MEAS-Read: VelocityECU	PHYS(km/h)	value > 40
18	BUS-Read: VelocityBus	PHYS(don't care)	value > 40
19	MODEL-Read: VelocityModel	PHYS(don't care)	value > 50
20	**Set driver request to 0 %**		
21	MODEL-Write: DriverRequest	PHYS(don't care)	0
22	**Check recuperation 'on'**		
23	MEAS-Read: RecuperationECU	PHYS(don't care)	value == 1

review demo_basic_drive.pkg

THINKING

▶ Show / hide

✓ Read test steps demo_basic_drive.pkg

I've reviewed demo_basic_drive.pkg and found the following issues:
Structure: The test case is well-organized with clear Precondition, Action, and Postcondition sections.
Issues Found:
Inconsistent threshold values: In the Action section, the block 'check if speed is faster 50' has a mismatch. The description states all velocity signals should be larger than 50, but the implementation checks VelocityECU and VelocityBus for values > 40, while only VelocityModel checks for > 50. All checks should use the same threshold of 50.
Typo in Postcondition: The block 'Stop vehicle' contains a typo - 'veloctiy' should be 'velocity'.
Unconventional wording: The block 'Switch to "P" (0) and unestablish VehTrStby' uses 'unestablish', which is unconventional. Consider using 'deactivate' or 'disable' instead.
The Precondition section checking state of charge between 20% and 90% is correctly implemented and consistent with its description.

Abbildung 6: ecu.test *agent* chat

These lists are examples. Let's talk about your workflows together. Where do you lose the most time?

In **ecu.test 2026.1**, the **ecu.test** *agent chat* is already available on request. Please feel free to contact us at support@tracetronic.com.

test.spec *agent*

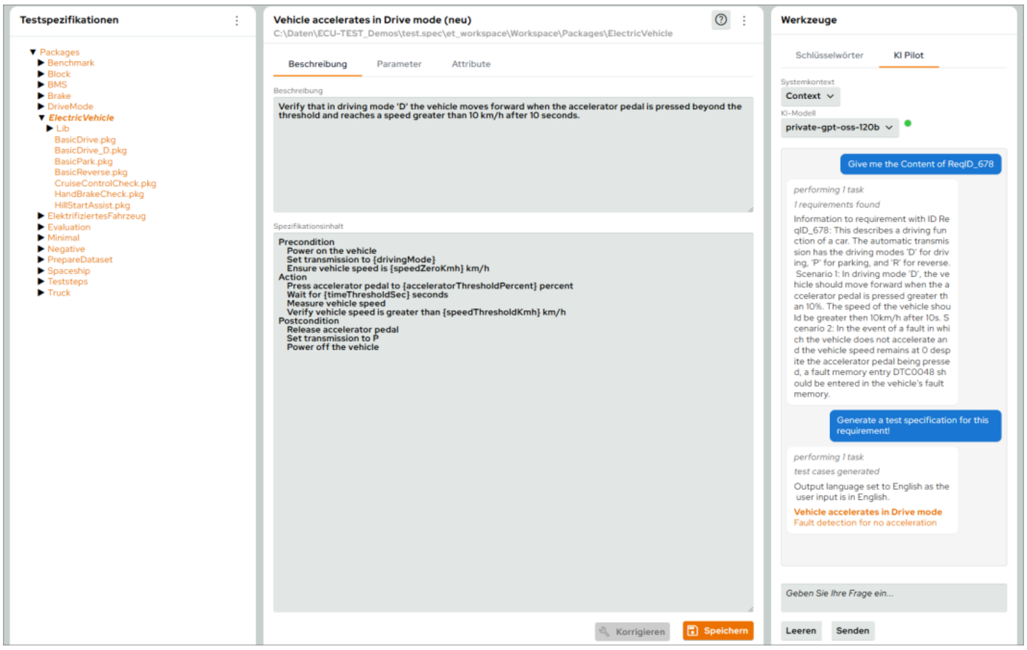
With the current release, the **test.spec** *agent* has undergone consistent further development. The focus was on improving the quality of the entire Agentic workflow. Now, specifications are generated more consistently. Test methodologies, such as positive/negative tests and equivalence classes, are automatically considered. Rules for generating specifications can be specifically defined and selectively applied.

Another new feature is the ability to connect to Codebeamer, which allows requirements from the ALM system to be integrated directly into the specification process.

Outlook:

- Planned release at the end of June
- Closer interaction between the **test.spec** *agent* and the **ecu.test** *agent* to generate consistent test cases
- Editor optimizations to efficiently maintain specifications
- Initial functions to support trace analysis and quality assessment of requirements and specifications

Take advantage of the preview phase to evaluate the **test.spec** *agent* in your tool landscape and experience how it reduces the time it takes to create specifications from hours to minutes.
We are happy to assist you with the first steps and look forward to your inquiry!



Testspezifikationen

- Packages
  - Benchmark
  - Block
  - BMS
  - Brake
  - DriveMode
  - *ElectricVehicle*
    - Lib
    - BasicDrive.pkg
    - BasicDrive_D.pkg
    - BasicPark.pkg
    - BasicReverse.pkg
    - CruiseControlCheck.pkg
    - HandBrakeCheck.pkg
    - HillStartAssist.pkg
  - ElektrifiziertesFahrzeug
  - Evaluation
  - Minimal
  - Negative
  - PrepareDataset
  - Spaceship
  - Teststeps
  - Truck

Vehicle accelerates in Drive mode (neu)
C:\Daten\ECU-TEST_Demos\test.spec\et_workspace\Workspace\Packages\ElectricVehicle

Beschreibung: Verify that in driving mode 'D' the vehicle moves forward when the accelerator pedal is pressed beyond the threshold and reaches a speed greater than 10 km/h after 10 seconds.

Spezifikationsinhalt:

```
Precondition
   Power on the vehicle
   Set transmission to {drivingMode}
   Ensure vehicle speed is {speedZeroKmh} km/h
Action
   Press accelerator pedal to {acceleratorThresholdPercent} percent
   Wait for {timeThresholdSec} seconds
   Measure vehicle speed
   Verify vehicle speed is greater than {speedThresholdKmh} km/h
Postcondition
   Release accelerator pedal
   Set transmission to P
   Power off the vehicle
```

Werkzeuge

Systemkontext: Context

KI-Modell: private-gpt-oss-120b

Give me the Content of ReqID_678

performing 1 task

1 requirements found

Information to requirement with ID ReqID_678: This describes a driving function of a car. The automatic transmission has the driving modes 'D' for driving, 'P' for parking, and 'R' for reverse. Scenario 1: In driving mode 'D', the vehicle should move forward when the accelerator pedal is pressed greater than 10%. The speed of the vehicle should be greater then 10km/h after 10s. Scenario 2: In the event of a fault in which the vehicle does not accelerate and the vehicle speed remains at 0 despite the accelerator pedal being pressed, a fault memory entry DTC0048 should be entered in the vehicle's fault memory.

Generate a test specification for this requirement!

performing 1 task

test cases generated

Output language set to English as the user input is in English.

Vehicle accelerates in Drive mode
Fault detection for no acceleration

Figure 7: test.spec agent

TouchInput with AI prompt

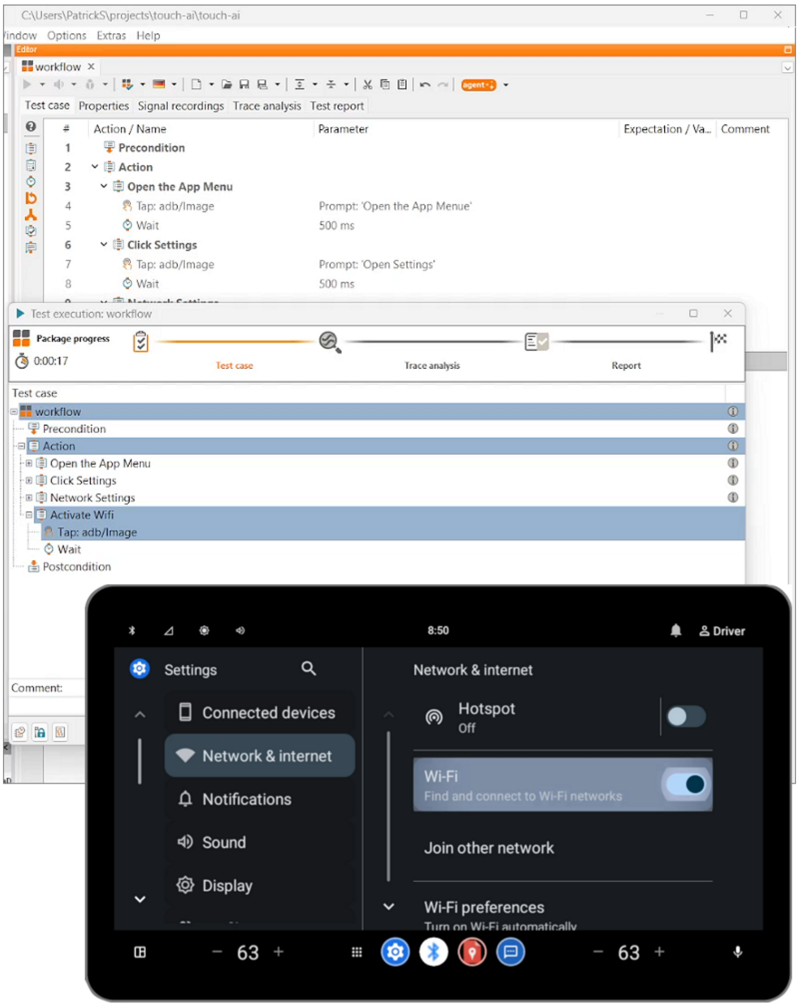
et

With the **TouchInput** test step, **ecu.test** already offers the option of performing various actions (clicking, swiping, rotating, etc.) to interact with an HMI or a website. Until now, however, fixed coordinates had to be specified.

This approach is not particularly robust, especially when testing HMI elements. If the positions and sizes of the elements change during development, manually created test cases must be laboriously adjusted.

The new AI function in the **TouchInput** test steps allows prompts to be used instead of coordinates. For testing an HMI, for example, the following prompts are conceivable:

- Press the "Radio" button.
- I want to make a phone call.
- I want to listen to music.



C:\Users\PatrickS\projects\touch-ai\touch-ai

#	Action / Name	Parameter	Expectation / Va...	Comment
1	Precondition			
2	Action			
3	Open the App Menu			
4	Tap: adb/Image	Prompt: 'Open the App Menue'		
5	Wait	500 ms		
6	Click Settings			
7	Tap: adb/Image	Prompt: 'Open Settings'		
8	Wait	500 ms		

Test execution: workflow

Package progress 0:00:17

Test case
- workflow
  - Precondition
  - Action
    - Open the App Menu
    - Click Settings
    - Network Settings
    - Activate Wifi
      - Tap: adb/Image
      - Wait
  - Postcondition

Figure 8: Touch input function with AI prompt

The model analyzes the image and returns the coordinates regardless of the language setting, element arrangement, or displayed content. This enables the creation of robust test steps.

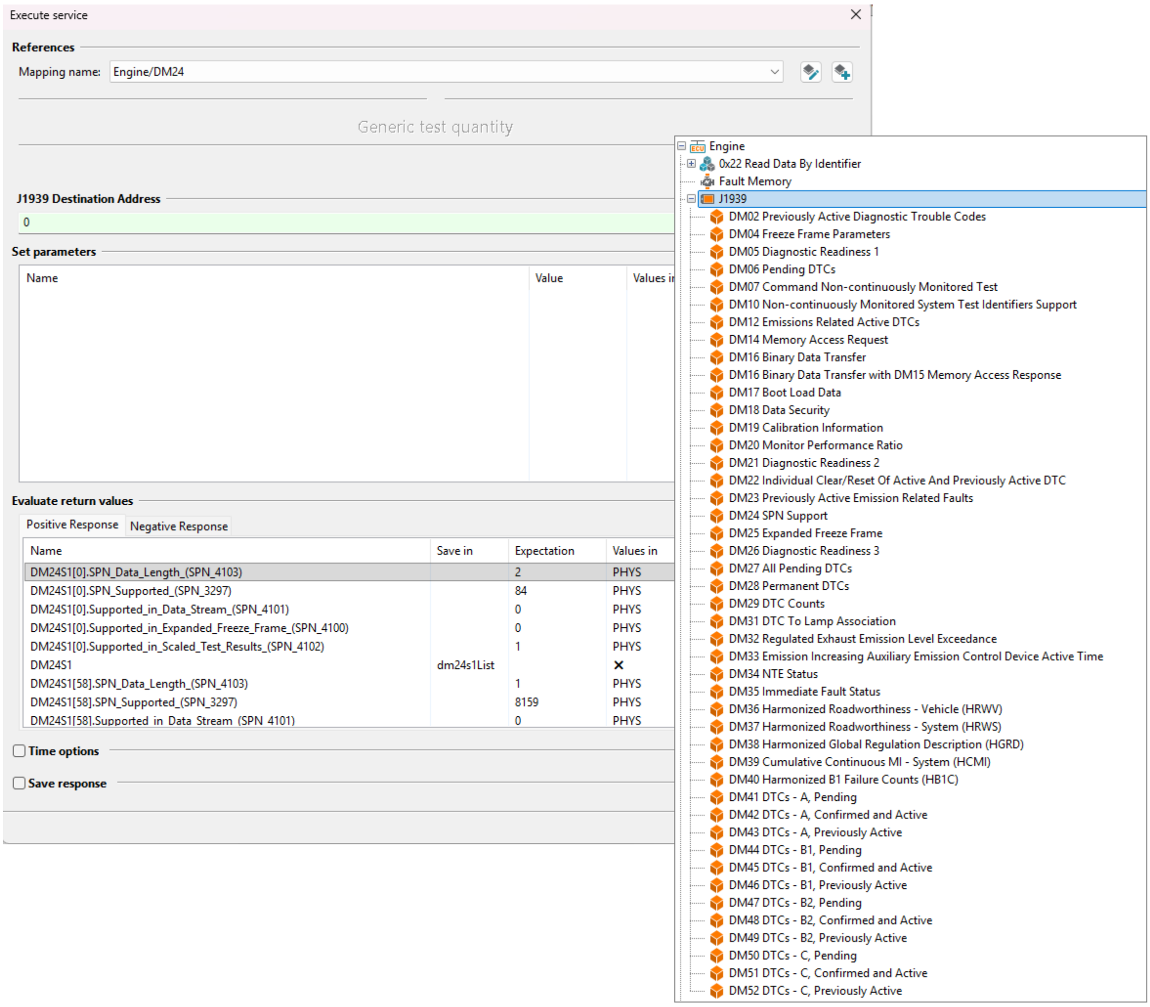
The model is connected via the **ecu.test** *agent* connector and can be customized to individual needs. We have had positive experiences with both Claude 4.5 and ChatGPT.
Would you like to try out the new feature now? Please contact your tracetronic representative or our support team
.

J1939 as symbolic test steps

In the context of J1939, the most important services are probably the DM services. These services are defined and equivalent to OBD-UDS in the context of UDS.

Due to their large scope and complexity, especially regarding the return parameters, we are experimenting with symbolic support for J1939 DM services. Similar to OBDonUDS, they appear as generic **diagnostic** test steps in the **Diagnostics** tab after activation. A database is not necessary.

Activation is available upon request and requires a license **ecu.test** *diagnostics*.



Execute service

References

Mapping name: Engine/DM24

Generic test quantity

J1939 Destination Address

0

Set parameters

Name	Value	Values i

Evaluate return values

Positive Response | Negative Response

Name	Save in	Expectation	Values in
DM24S1[0].SPN_Data_Length_(SPN_4103)		2	PHYS
DM24S1[0].SPN_Supported_(SPN_3297)		84	PHYS
DM24S1[0].Supported_in_Data_Stream_(SPN_4101)		0	PHYS
DM24S1[0].Supported_in_Expanded_Freeze_Frame_(SPN_4100)		0	PHYS
DM24S1[0].Supported_in_Scaled_Test_Results_(SPN_4102)		1	PHYS
DM24S1	dm24s1List		✕
DM24S1[58].SPN_Data_Length_(SPN_4103)		1	PHYS
DM24S1[58].SPN_Supported_(SPN_3297)		8159	PHYS
DM24S1[58].Supported_in_Data_Stream_(SPN_4101)		0	PHYS

☐ **Time options**

☐ **Save response**

- Engine
  - 0x22 Read Data By Identifier
  - Fault Memory
  - J1939
    - DM02 Previously Active Diagnostic Trouble Codes
    - DM04 Freeze Frame Parameters
    - DM05 Diagnostic Readiness 1
    - DM06 Pending DTCs
    - DM07 Command Non-continuously Monitored Test
    - DM10 Non-continuously Monitored System Test Identifiers Support
    - DM12 Emissions Related Active DTCs
    - DM14 Memory Access Request
    - DM16 Binary Data Transfer
    - DM16 Binary Data Transfer with DM15 Memory Access Response
    - DM17 Boot Load Data
    - DM18 Data Security
    - DM19 Calibration Information
    - DM20 Monitor Performance Ratio
    - DM21 Diagnostic Readiness 2
    - DM22 Individual Clear/Reset Of Active And Previously Active DTC
    - DM23 Previously Active Emission Related Faults
    - DM24 SPN Support
    - DM25 Expanded Freeze Frame
    - DM26 Diagnostic Readiness 3
    - DM27 All Pending DTCs
    - DM28 Permanent DTCs
    - DM29 DTC Counts
    - DM31 DTC To Lamp Association
    - DM32 Regulated Exhaust Emission Level Exceedance
    - DM33 Emission Increasing Auxiliary Emission Control Device Active Time
    - DM34 NTE Status
    - DM35 Immediate Fault Status
    - DM36 Harmonized Roadworthiness - Vehicle (HRWV)
    - DM37 Harmonized Roadworthiness - System (HRWS)
    - DM38 Harmonized Global Regulation Description (HGRD)
    - DM39 Cumulative Continuous MI - System (HCMI)
    - DM40 Harmonized B1 Failure Counts (HB1C)
    - DM41 DTCs - A, Pending
    - DM42 DTCs - A, Confirmed and Active
    - DM43 DTCs - A, Previously Active
    - DM44 DTCs - B1, Pending
    - DM45 DTCs - B1, Confirmed and Active
    - DM46 DTCs - B1, Previously Active
    - DM47 DTCs - B2, Pending
    - DM48 DTCs - B2, Confirmed and Active
    - DM49 DTCs - B2, Previously Active
    - DM50 DTCs - C, Pending
    - DM51 DTCs - C, Confirmed and Active
    - DM52 DTCs - C, Previously Active

Figure 9: DM-Services for J1939

3 ecu.test extras

3.1 ecu.test *agent*

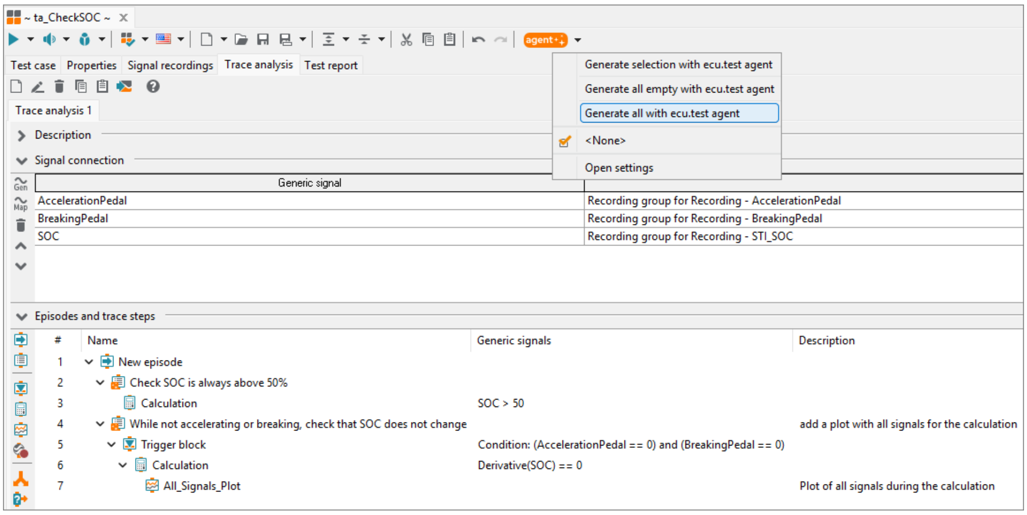
Support for trace analysis

et tc

In the last release, the generation of trace steps was available as a preview, but this feature is now officially available.

The **ecu.test** *agent* generates content for blocks on request. Currently, trigger blocks, calculation steps, and plots are currently supported.

Similar to test cases, blocks with generated content are marked with a small symbol: 📋.



Generic signal	
AccelerationPedal	Recording group for Recording - AccelerationPedal
BreakingPedal	Recording group for Recording - BreakingPedal
SOC	Recording group for Recording - STI_SOC

#	Name	Generic signals	Description
1	New episode		
2	Check SOC is always above 50%		
3	Calculation	SOC > 50	
4	While not accelerating or breaking, check that SOC does not change		add a plot with all signals for the calculation
5	Trigger block	Condition: (AccelerationPedal == 0) and (BreakingPedal == 0)	
6	Calculation	Derivative(SOC) == 0	
7	All_Signals_Plot		Plot of all signals during the calculation

Figure 10: Generation of trace steps via ecu.test *agent*

Optimizations and support for further test steps

et

The current **ecu.test** release, 2026.1, includes numerous optimizations for generating test cases. These include, among others:

- Values from enumerations are taken into account during generation
- Test steps from unsupported reference implementations are generated as **ToDo** steps
- The time options "Wait until true," "Wait for at least," and "Polling cycle" are included in the generation
- Optimization in the generation of tool and port jobs

ecu.test also supports the following test steps:

- Exit
- Comment

3.2 ecu.test *calibration*

Simultaneous recording of a bus and a calibration port for CAN(-FD)

et

Records for bus and XCP communication often have to be made using the same controller as the measurement hardware. This was previously possible and resulted in an error message.
As of this release, this restriction no longer applies.

3.3 ecu.test *drive*

Shortcuts to pause and cancel test cases

et

You can now pause a test execution by pressing Space and cancel a test case by pressing **Esc**.

3.4 ecu.test *lab*

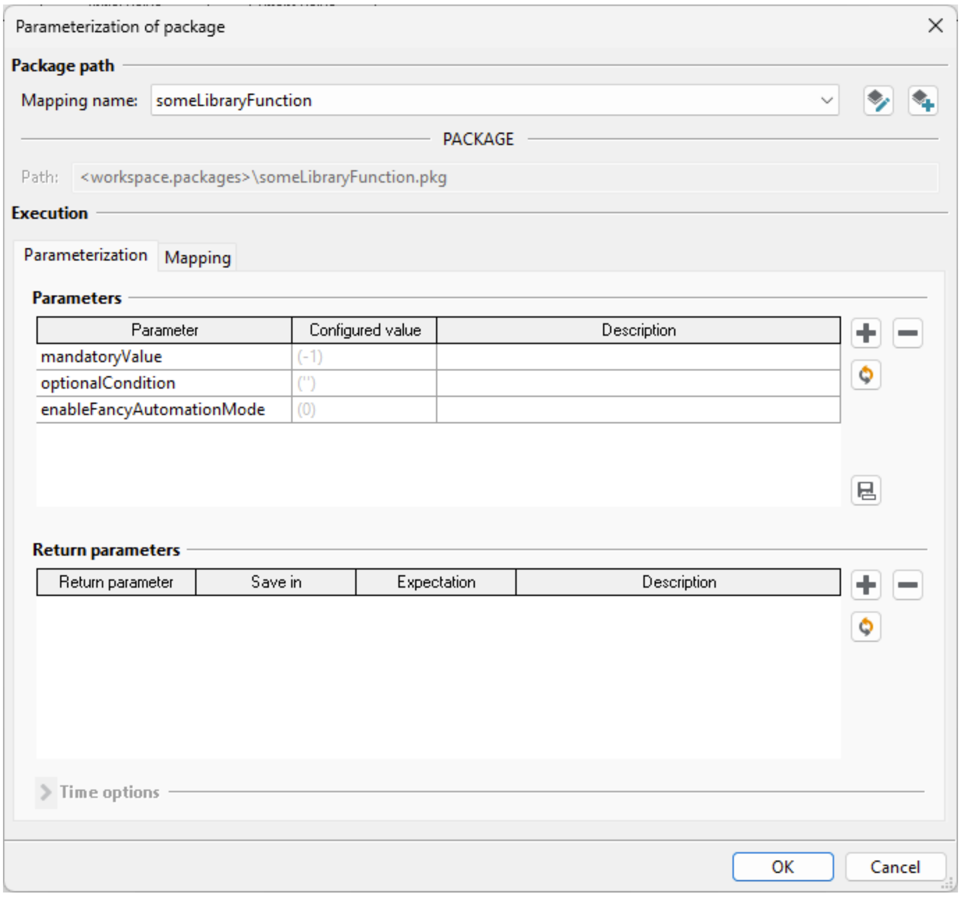
Parameterizing all parameters of a package

et

Previously, only a single parameter could be assigned in write widgets for a stored package.

With this version, all parameters can now be taken care of. In most cases, this means that one parameter is assigned dynamically by operating the widget, and the remaining parameters are either assigned static values or deliberately executed with the initial value stored in the package.

For reading widgets, packages previously only offered the option of selecting the return value. In this case, the parameters can now also be configured statically.



Parameterization of package

Package path

Mapping name: someLibraryFunction

PACKAGE

Path: <workspace.packages>\someLibraryFunction.pkg

Execution

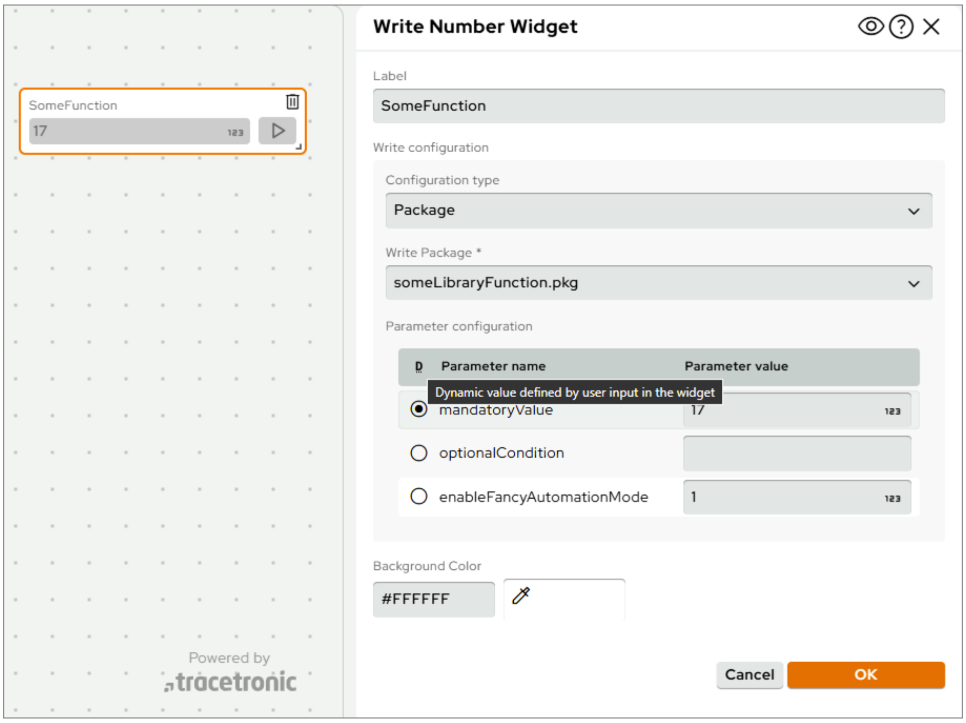
Parameters

Parameter	Configured value	Description
mandatoryValue	(-1)	
optionalCondition	('')	
enableFancyAutomationMode	(0)	

Return parameters

Return parameter	Save in	Expectation	Description

Figure 11: Parameterization of a package in ecu.test



Write Number Widget

Label: SomeFunction

Write configuration

Configuration type: Package

Write Package *: someLibraryFunction.pkg

Parameter configuration

D	Parameter name	Parameter value
●	mandatoryValue	17
○	optionalCondition	
○	enableFancyAutomationMode	1

Dynamic value defined by user input in the widget

Background Color: #FFFFFF

Figure 12: Parameterization of all packages in ecu.test *lab*

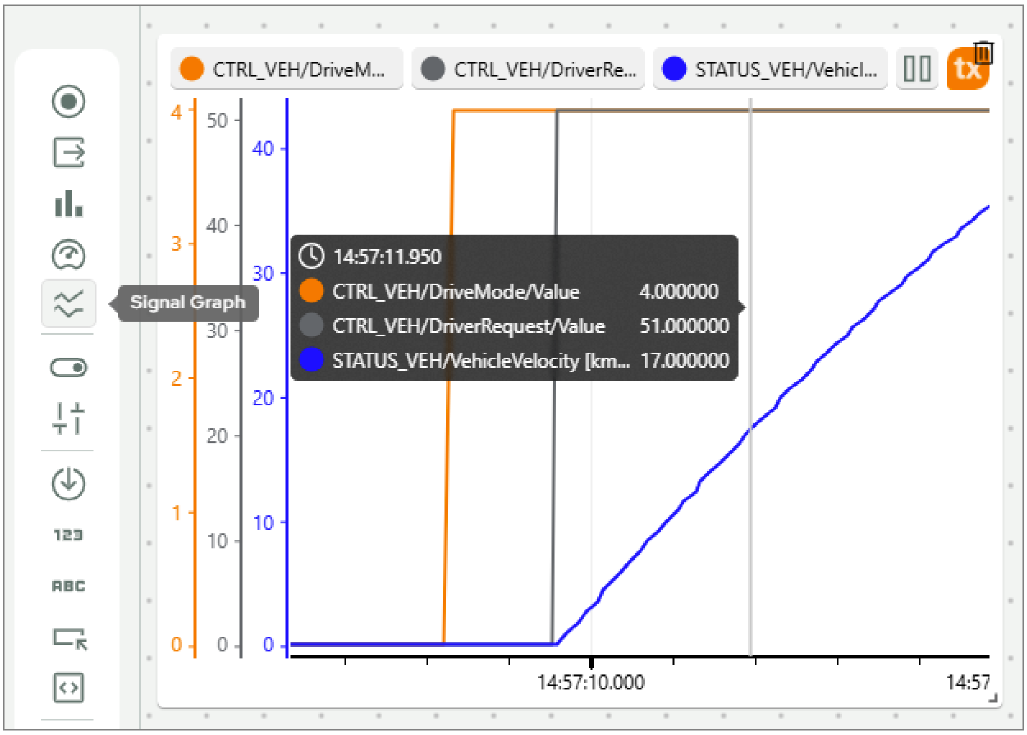
New widget: Signal Graph

et

Ever since **ecu.test** *lab* was launched, there has been a demand for signal curves to be displayed. With **ecu.test 2026.1** and our new **Signal Graph** widget, we are now able to meet this request.

The widget is based on the signal visualization of the **trace.xplorer** and can show up to four signals in one graph. You can configure whether these signals share a common Y-axis or each have their own axis. Optionally, markers for signal values and interpolated curve plotting can be activated.

During execution, you can display the signal values at the cursor position by clicking on the graph. Signals can be temporarily hidden for a better overview. In addition, the drawing of the waveform can be paused if necessary and resumed later (without loss of values).



14:57:11.950

● CTRL_VEH/DriveMode/Value	4.000000
● CTRL_VEH/DriverRequest/Value	51.000000
● STATUS_VEH/VehicleVelocity [km...	17.000000

Figure 13: New widget for displaying signal graphs

Multiple pages for each view

et

To increase the available workspace and improve its structure, we have added tabs (or pages) to the views. This allows multiple pages to be filled with widgets and initialized and used simultaneously.

This eliminates the need to stop monitoring mode and to change the view.

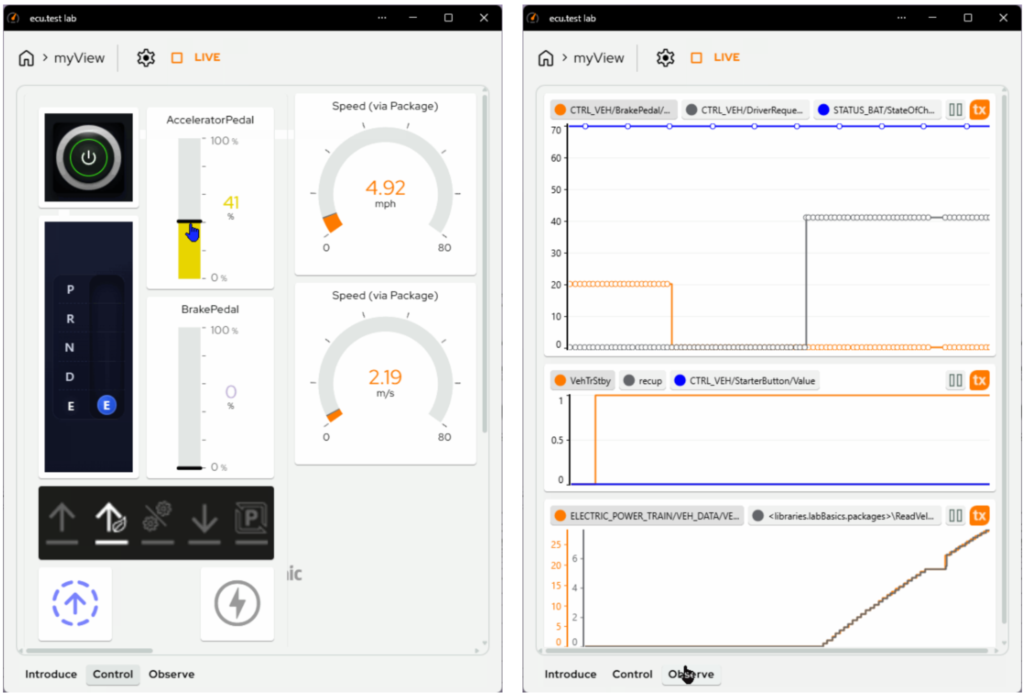


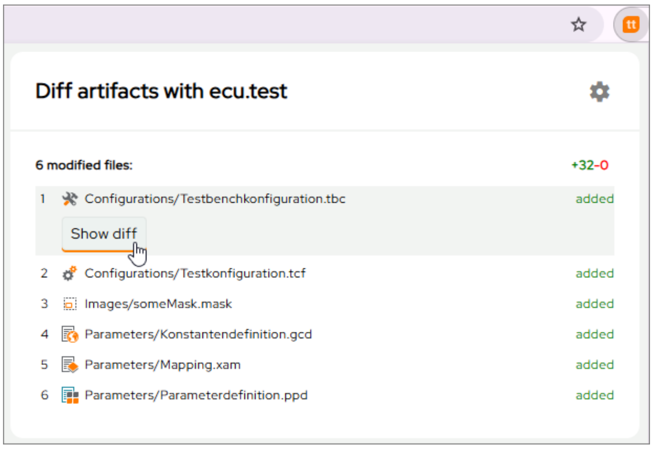
Figure 14: Implementation of tabs for navigating through multiple pages

4 Usability

Browser extension "Open with ecu.test diff"

Open with ecu.test diff allows you to open changes to **ecu.test** artifacts in the **ecu.test Diff Viewer** directly from your browser.

The newly released version 1.1.0 now supports **BitBucket Cloud** in addition to GitHub and GitLab. Furthermore, the setup and some error messages have been improved.
The extension is available from the Firefox and Chrome store.



Diff artifacts with ecu.test

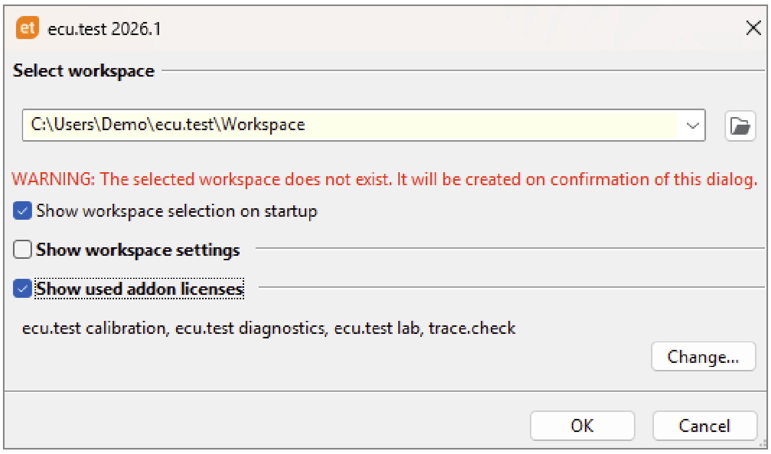
6 modified files: +32-0

1	Configurations/Testbenchkonfiguration.tbc	added
2	Configurations/Testkonfiguration.tcf	added
3	Images/someMask.mask	added
4	Parameters/Konstantendefinition.gcd	added
5	Parameters/Mapping.xam	added
6	Parameters/Parameterdefinition.ppd	added

Figure 15: Browser- "Open with ecu.test diff"

Workspace selection dialog: Licenses for ecu.test extras

In the workspace selection dialog, before **ecu.test** is fully started, the licenses used for the **ecu.test** extras can be displayed and, if necessary, configured directly in the License Manager.



ecu.test 2026.1

Select workspace

C:\Users\Demo\ecu.test\Workspace

WARNING: The selected workspace does not exist. It will be created on confirmation of this dialog.

☑ Show workspace selection on startup

☐ Show workspace settings

☑ Show used addon licenses

ecu.test calibration, ecu.test diagnostics, ecu.test lab, trace.check

Figure 16: Display and configure licenses for ecu.test *extras*

5 Test aspects

5.1 HiL

Support for Intrepid CAN(-FD) bus hardware connection

et

All features of hardware-related bus connections, including ecu.test *diagnostics* and **ecu.test** *calibration*, can now be used with CAN and CAN-FD-capable measurement hardware that can be controlled in conjunction with the Intrepid neoVI API.

ETAS INCA: Revised handling of data sets

et

The INCA .tbc-option **Handling of existing project** for application ports determines which of the configurations for project (A2L file) and data set (HEX file) in **ecu.test** and INCA are used when opening the INCA experiment.

Since there can be a variety of requirements depending on whether a data set exists and which data set should be prioritized, the selection options have been fundamentally revised and documented. Details can be found in the user documentation.

5.2 Test management

Sample workflow for PTC Codebeamer

et

Based on the new User Test Management API, there is an initial sample workflow for importing packages and projects from PTC Codebeamer.

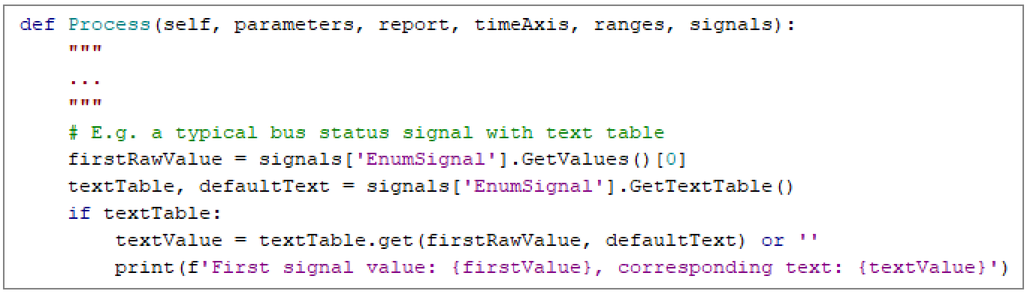
Together with the tractronic Codebeamer client, you get full access to the Codebeamer REST API. The purely Python-based solution can be easily adapted to your individual workflows.

5.3 Trace analysis

Access to enum values in trace step templates

et tc

It is now possible to access the enum values of a signal via the API in event-based Python and array-based NumPy trace step templates.



```
def Process(self, parameters, report, timeAxis, ranges, signals):
    """
    ...
    """
    # E.g. a typical bus status signal with text table
    firstRawValue = signals['EnumSignal'].GetValues()[0]
    textTable, defaultText = signals['EnumSignal'].GetTextTable()
    if textTable:
        textValue = textTable.get(firstRawValue, defaultText) or ''
        print(f'First signal value: {firstValue}, corresponding text: {textValue}')
```

Figure 17: Access to enum values via API

New standard trace step template: CalculateInterpolatedSignal

et tc

Rarely sampled signals from originally continuous (analog) measured quantities often make robust testing difficult. By default, signal values are mapped to a common time axis with other signals using sample-and-hold. However, the use of the **Hose** function, for example, requires individual interpolation.

The new standard trace step template **CalculateInterpolatedSignal** allows you to linearly interpolate the values of a signal onto the time axis of a reference signal. This makes it easy to implement many use cases.

5.4 Further test aspects

Improved support for measurement lists

et

The current version builds on the option introduced in **ecu.test** 2025.4 to use XAM files as measurement lists for signal groups and offers further practical improvements.

- **Integrated library workspaces:** XAM files can now be imported directly from library workspaces.
- **Reporting:** For better traceability, measurement lists used for recording are documented.
- **Easy configuration:** Measurement lists can now be easily configured using drag-and-drop.

6 Tools and Interfaces

6.1 New tools and versions

et

	Provider	Website	System	Product name	Version
1	**Accurate Technologies Inc.**	Release link	Calibration and data acquisition tool	**ATI VISION**	**7.0**
2	**Lauterbach**	Release link	Tool for analyzing, optimizing, and certifying embedded systems	**TRACE32**	**R.2025.2**
3	**PLS**	Release link	Tool for debugging, tracing, and testing embedded software	**UDE**	**2025.2**

7 Discontinuations

7.1 Discontinued features and incompatibilities in this version

Python 3.13

et tc

ecu.test 2026.1 is based on Python 3.13.

According to current estimates, the switch should have little impact on existing **ecu.test** workspaces.

However, you should check your own Python code in packages, user libraries, and external Python libraries for compatibility with Python 3.13 and update it if necessary.

- **https://pyreadiness.org/3.13/**

You can check if your code locations are affected by removals using the following link:

- **https://docs.python.org/3/whatsnew/3.13.html#removed**

Helpful information about switching can be found in the

- **Knowledge Base**

Port BUSACCESS – GENERIC_MAPPINGFILE

et

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, the BUSACCESS - GENERIC_MAPPINGFILE has been removed from **ecu.test 2026.1**.

Supported versions of MATLAB/Simulink in Linux

et

Following the discontinuation of support for older versions of Ubuntu with **ecu.test 2026.1**, support for MATLAB/Simulink up to and including R2023b has been removed under Linux, as these versions do not offer support for Ubuntu 20.04.
Support for Windows remains unchanged for versions from R2015b onwards.

FEP2 connection

et

The FEP2 connection has been removed from **ecu.test 2026.1**.

Ubuntu 20.04 LTS and 22.04 LTS

et

Support for Ubuntu 20.04 LTS and 22.04 LTS has been removed from **ecu.test 2026.1.**

ICMP-RAW port

et

For the Ethernet, XL-API and SIL-Kit tools, the ICMP-RAW port has been removed without replacement, as announced.

Alternative report directory for separate subproject execution

et tc

When executing separate subprojects, it was possible to specify the report folder. This feature was removed in **ecu.test 2026.1**.

Playbook-Export from ecu.test to test.guide

et tc

With **ecu.test 2026.1**, the playbook export from **ecu.test** to **test.guide** has been removed.

7.2 Discontinued features in future versions

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.

KS: Tornado only via ASAM ACI

et

The tool connection will be removed with **ecu.test 2026.2**. It will be replaced by the new connection based on ASAM ACI, which is available since **ecu.test 2023.3**.

Jama Connect

et

The integrated test management adapter for **Jama Connect** will be removed with **ecu.test 2026.2**.

As a replacement the user defined test management adapters can be used. These adapters offer significantly more flexibility and enable optimal adaptation to the respective workflow.

A sample workflow is provided to assist with your own implementation.

TRF files can no longer be generated for project reports

et

With the introduction of the PRF format for project reports, the TRF format has become obsolete. Although the TRF format could be activated via the workspace settings during the migration phase, this fallback setting is expected to be removed in **ecu.test 2026.3**. Until then, based on feedback received, work is underway to improve some workflows with and without **test.guide**.

This discontinuation only affects project reports. Package executions will continue to be saved in the TRF format.

Interface TmUserAdapter

et

The following interface methods will be removed with **ecu.test 2026.2**:

- GetPackageFromTms
- GetProjectsFromTms

These newly introduced methods serve as replacements:

- **FetchChildrenForPackageImport**
- **FetchChildrenForProjectImport**

Despite the change, the familiar tree structure of the test management system is still present in the import dialogs.

Import in connection with CustomChecks

et

Custom tests may use a module that has been moved. Therefore, check the imports and adjust them if necessary.

- **old: tts.core.common.check.Constants**
- **new: tts.interface.customCheck.Constants**

Real-time package references

et

dSPACE has changed its interface for controlling the real-time environment with ControlDesk 2024-B. Since this dSPACE release, it is no longer possible to execute real-time package references with **ecu.test**.

Due to the lack of user feedback on this issue, we assume that this feature is rarely used and will remove it with **ecu.test 2026.3**.

Attention: The discontinuation only applies to real-time package references. Other features related to real-time testing (RTT) are not affected.

Bus monitoring test steps

et

The bus monitoring test steps used to check **timings** and **DLC** can be implemented much more effectively with trace analysis and the provided analysis modules.

Starting with **ecu.test 2026.3**, support for inserting new test steps will be discontinued. Support for execution will be removed in **ecu.test 2027.1** at the earliest.

Basler: Pylon

et

Basler cameras are compatible with the GenICam standard, which eliminates the need for a separate tool connection. For this reason, the **Basler: Pylon** tool will be removed from **ecu.test 2026.3**.

When using a Basler camera, the GenICam tool connection can be used as an alternative.

Siemens Polarion

et

The integrated test management adapter for **Siemens Polarion** will be removed with **ecu.test 2027.1**.

As a replacement the user defined test management adapters can be used.
These adapters offer significantly more flexibility and enable optimal adaptation to the respective workflow.
A sample workflow is provided to assist with your own implementation.

IBM Rational Quality Manager (RQM)

et

The integrated test management adapter **for IBM Rational Quality Manager (RQM)** will be removed with **ecu.test 2027.1**.

As a replacement the user defined test management adapters can be used.
These adapters offer significantly more flexibility and enable optimal adaptation to the respective workflow.
A sample workflow is provided to assist with your own implementation.

HP Application Lifecycle Management (ALM)

et

The built-in **HP Application Lifecycle Management (ALM)** test management adapter will be removed in **ecu.test 2027.1**.

For continued interaction with HP Application Lifecycle Management (ALM) after that release, a user-defined test management adapter needs to be implemented using the ecu.test Test Management API.

PTC Integrity

et

The built-in **PTC Integrity** test management adapter will be removed in **ecu.test 2027.1**.

For continued interaction with PTC Integrity after that release, a user-defined test management adapter needs to be implemented using the ecu.test Test Management API.

Interactive test execution in ecu.test

et

The use cases for interactive test execution within **ecu.test** and the web application **ecu.test** *drive* largely overlap.

ecu.test *drive* is the significantly more flexible solution.

- More intuitive user guidance
- Better adaptability to the areas of application in the vehicle
- Usability with runner and remote through decoupling from **ecu.test** GUI

We will therefore focus on this solution in the future and will remove interactive test execution with **ecu.test 2027.1**.