

# Release Notes

Version 19u1



## Table of Contents

<b>Release Notes</b>	<b>4</b>
TPT 19u1	4
TPT 19	6
Highlights	6
General	7
codeBeamer	8
IBM ALM	8
Polarion	9
Requirements	9
Declarations	10
Test cases	10
Test platforms	11
Test execution	20
Autotester	21
Docker	21
Test assessment	21
APIs	25
TPT API	25
VM API	27
Test report	27
<b>Previous Release Notes</b>	<b>29</b>
Release Notes TPT 18	30
Release Notes TPT 17	39
Release Notes TPT 16	52
Release Notes TPT 15	60
Release Notes TPT 14	65

Release Notes TPT 13 ..... 69

Release Notes TPT 12 ..... 75

Release Notes TPT 11 ..... 82

Release Notes TPT 10 ..... 92

Release Notes TPT 9 ..... 100

Release Notes TPT 8 ..... 103

Release Notes TPT 7 ..... 107

# Release Notes

## TPT 19u1

### What's new

- If tests were executed multiple times by using the Test repetition field or a Parameter multi set in the Execution Configuration dialog, the results are exported cumulative to CSV or Excel but ignored when exporting to other tools like Polarion and IBM ALM.
- CANoe platform: More robust import interface, problematic signals are ignored.

### Bug fixes

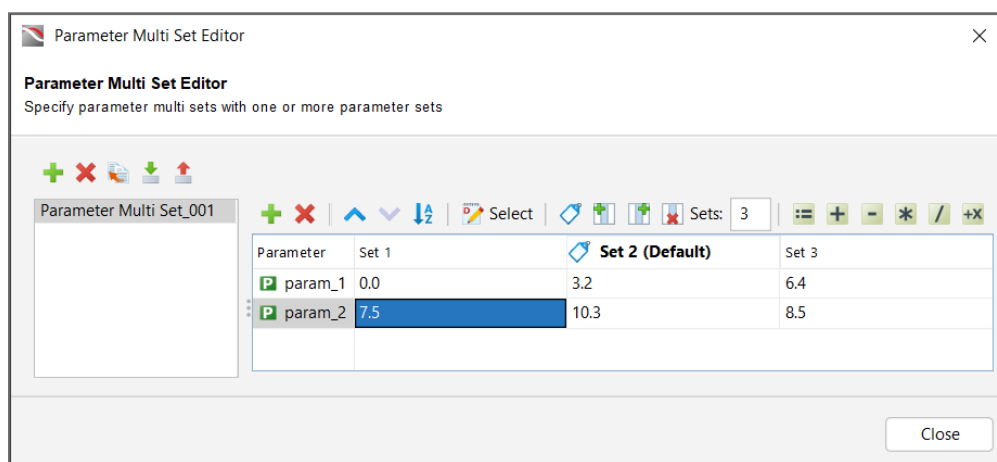
- Severe issue: Fixed bug that caused incorrect coverage results for MC/DC when using TPT Coverage (TASMO) with complex left-bound logical operations such as `if (<complex operation> || true)`. This bug caused some coverage targets to be marked as covered even though they were not.
- During the test set import from Excel files, IDs can now be read from standard and formatted fields.
- ASCET platform, ASCET@FUSION platform: Distributions are now prefixed by module name also in the value of 1D- and 2D-tables if needed.
- AUTOSAR platform: A call to `Rte_Write` will no longer trigger data received events in the generated test frame on the PR port with outer connectors it was called on. Data received events will now only be triggered on ports that actually receive data either from TPT or from another component.
- AUTOSAR platform: A call to `Rte_Write` will no longer set the updated-flag on the PR port it was called on. The updated-flag will now only be set on ports that actually receive data either from TPT or from another component.
- AUTOSAR platform: C typed per-instance memory (PIM) with type definitions that do not match implementation data types from ARXML files are now supported for multi-instance component types that occur more than once within a composition.
- C/C++ platform: When opening a TPT project saved with TPT 17 or TPT 18, the custom **Compiler options** in the Advanced source settings tab of the C/C++ platform are now loaded correctly.
- MATLAB/Simulink platform: Data store import with data store memory block outside of the SUT subsystem and no Simulink.Signal fixed.
- TASMO, MATLAB/Simulink platform: Fixed bug with lookup table blocks when analyzing the test frame model with TargetLink version 5.2 or newer.
- TASMO, MATLAB/Simulink platform: Fixed performance issue that occurred in rare cases with very large MATLAB/Simulink models in MATLAB R2020b and newer.

- TASMO: When measuring coverage with TASMO for TargetLink models with the option "Single simulation output" set in Simulink, TASMO was unable to evaluate the coverage. This bug is fixed.
- TASMO: When the **Enable direct feedthrough and add "Unit Delay" blocks for TPT inputs** checkbox in the MATLAB/Simulink platform was deselected, TASMO was unable to recognize coverage at the last sample of the test case. This bug is fixed.
- TASMO: Fixed an issue with using TASMO for Stateflow with MATLAB R2018b (or newer) which caused an error if TPT was installed below `C:\Program Files` or if a project was located on a different drive than the TPT installation.
- Vector CAN node: Fixed bug when using 2 CAN nodes with different Custom CRC dlls.
- C/C++ platform: Fixed bug that caused missing type definitions when creating the PiL target.
- Fixed bug that increasingly slowed down TPT while working with step lists.
- Fixed bug that occurred when channels or parameters of string data type were used as file name arguments in step lists to define service step types.
- When evaluating an environment variable from a test case attribute, the value is now only interpreted as potential markup text if it is from an attribute of type `Free text`. This solves e.g. issues accessing non-canonical file paths from test case attributes of type `File` as `$` variables.
- The TPT API function `PlatformConfiguration.setMapping()` now allows null arguments to be passed.
- New TPT API function `Back2BackSettings.updateRows()` has been added to update the rows automatically created with the function `Back2BackSettings.setAutoUpdate()` so that all changes are now applied.

# TPT 19

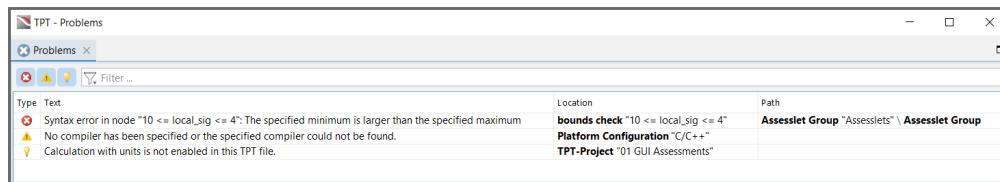
## Highlights

- **Test case autogeneration from formal requirements:** At the push of one button, test cases can be automatically generated based on formal requirements. These test cases are assessed after the test run without further steps.
- **Multi-execution of parameter sets:** By selecting a parameter multi set in the Execution Configuration dialog, you can execute test cases several times. Each time, a different set is used. Additionally, parameter sets are now easier to create with the new Parameter Multi Set Editor.



Parameter multi set with 3 parameter sets



- **Worst case execution time Indicator:** New indicator helps you to analyze performance problems at an early stage and helps you to reduce the selection of tests for measurements on target processors. This is done by means of the new checkboxes **Measure execution time of runnables** in the AUTOSAR platform and **Measure execution time of functions** in the C/C++ platform.
- **Stress testing:** Run test cases multiple times and increase the probability of detecting non-deterministic behavior of your system under test. Simply specify in the Execution Configuration dialog whether a test case should be executed several times with or without using different parameter sets.
- **All target compiler support:** Build your software for any target ECU, with any compiler for any processor architecture. This feature is only available for the C/C++ platform.
- **Simulink on Linux:** Running model-in-the-loop tests on Linux in the cloud, in Continuous Integration environments, or on local hosts is now supported with TPT in Docker.
- **Problems view:** During the setup of the test project, the new Problems view assists you by listing warnings and errors. With one click, you can jump directly to the warning or erroneous content for troubleshooting.



'Problems view' in TPT

## General

### What's new

- **Default configurations in the 'TPT Tool Preferences':** You can set your preferred C compiler configuration, Eclipse configuration, MATLAB/Simulink configuration, ASCET configuration, and XiL configuration in the TPT Tool Preferences as default. The default configuration is used when no specific configuration is selected.
- **Files created with TPT versions prior to TPT 8 are no longer supported:** To load such files anyway, open and save them beforehand, for example with TPT 18.
- **Markup Editor changed:** The  icon for displaying text formatted or plain has been replaced by . The logic has changed: when the icon is deselected, the formatted text is shown otherwise unformatted. Under TPT Tool Preferences|Workbench, you can specify how many clicks in a text field are necessary to open the Markup Editor.

### Bug fixes

- Files with invalid style attributes for example in the Documentation step of the step list can now be loaded in TPT.
- Fixed bug concerning typing and formatting of text in the Description view.
- Fixed bug concerning the deletion of entries by source in the Modifications view.
- Performance improved when opening and executing TPT project files where many Import Signal steps are used.
- Fixed bug that prevented TPT and TPT file from closing after deleting a Testlet step or a test case containing a Testlet step.
- If temp dir does not exist, TPT will create it instead of crashing during start up.
- Fixed bug that prevented PDF report generation and caused the test execution to fail when using multi-core.

# codeBeamer

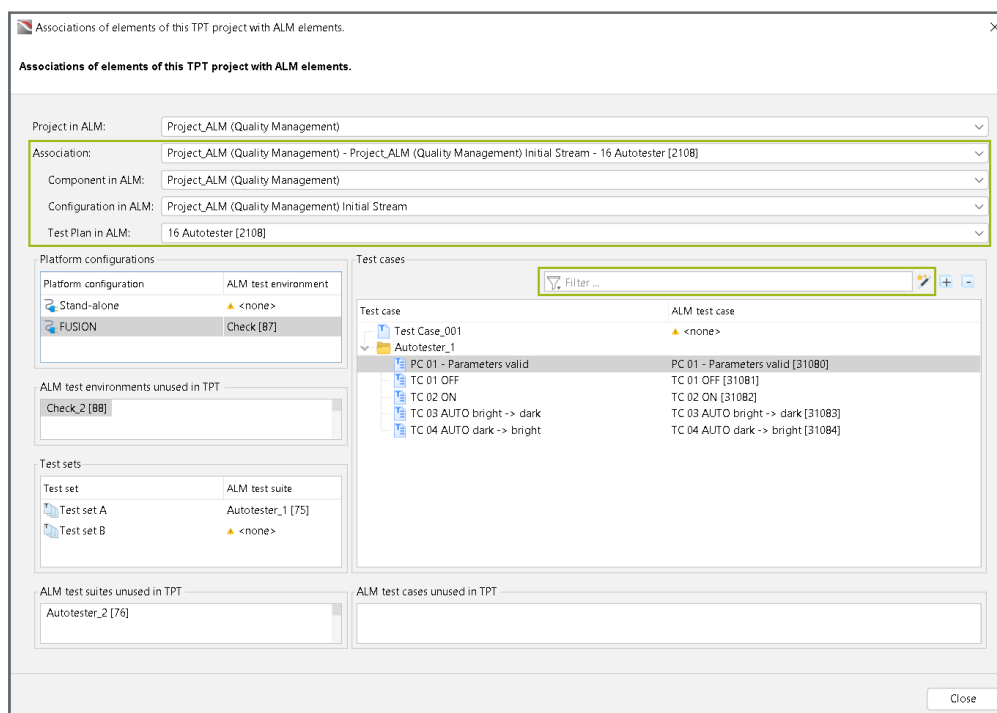
## What's new

- **Test results export to test run trackers improved:** During the test results export to codeBeamer, you can export test results to an existing test run tracker or create a new one. You can give the new test run tracker a name, a description, and add values to custom fields.
- **Export of item references supported:** It is now possible to export values of a TPT test case attribute to a custom choice field of the type "Work/ConfigItem" in codeBeamer.

# IBM ALM

## What's new

- **Import and export tables sortable and searchable:** There is a new filter field for free text search that runs either over a specific column or over all columns.
- **Multiple test plan associations, automatic test case associations, and test case filtering supported in the 'Show Associations dialog' supported.**



'Show Associations dialog'

## Bug fixes

- Fixed bug when exporting test results to IBM ALM while test results of a prior TPT version are present.



# Polarion

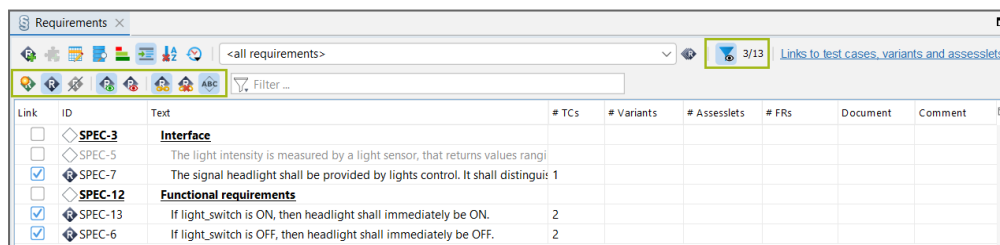
## What's new

- **Test results export extended:** You can upload test case reports (PDF or HTML), report archives (ZIP), and test data archives (ZIP) to the test records in Polarion during the test results export.
- **Test results export to Polarion optimized:** When creating a new test run, you can choose an existing test run template for the export, set a test run ID, and test run title. It is also possible to send the test results to an existing test run.

## Requirements

### What's new

- **New 'Requirements Settings section' in the 'TPT Preferences dialog':** In this project-specific section, you can specify which types of requirement links should be considered in your model (links to test cases, links to assesslet, links to test cases, variants and assesslets).
- **Retesting after a requirement's change simplified:** When a requirement has been changed and you have adapted the corresponding assessment, you can now execute only the test cases linked to the changed requirement by right-clicking on the requirement and selecting **Execute linked test cases**.
- **New filter options in the 'Requirements view':** Headings and information can now be filtered by 'new', 'normal', 'deleted', 'reviewed' and 'unreviewed'. In addition, the arrangement of the filter buttons has been changed.



The screenshot shows the 'Requirements' view in Polarion. At the top, there is a toolbar with various icons and a search bar containing '<all requirements>'. Below the toolbar, there is a row of filter buttons: 'New', 'Normal', 'Deleted', 'Reviewed', and 'Unreviewed'. A 'Filter ...' dropdown menu is also present. The main area displays a table of requirements with columns: Link, ID, Text, # TCs, # Variants, # Assesslets, # FRs, Document, and Comment. The table contains several rows, including 'SPEC-3 Interface', 'SPEC-5 The light intensity is measured by a light sensor, that returns values rangi', 'SPEC-7 The signal headlight shall be provided by lights control. It shall distinguis 1', 'SPEC-12 Functional requirements', 'SPEC-13 If light\_switch is ON, then headlight shall immediately be ON.', and 'SPEC-6 If light\_switch is OFF, then headlight shall immediately be OFF.'.

Link	ID	Text	# TCs	# Variants	# Assesslets	# FRs	Document	Comment
<input type="checkbox"/>	SPEC-3	Interface						
<input type="checkbox"/>	SPEC-5	The light intensity is measured by a light sensor, that returns values rangi						
<input checked="" type="checkbox"/>	SPEC-7	The signal headlight shall be provided by lights control. It shall distinguis 1	1					
<input type="checkbox"/>	SPEC-12	Functional requirements						
<input checked="" type="checkbox"/>	SPEC-13	If light_switch is ON, then headlight shall immediately be ON.	2					
<input checked="" type="checkbox"/>	SPEC-6	If light_switch is OFF, then headlight shall immediately be OFF.	2					

*New filter options in the 'Requirements view'*

## Bug fixes

- **Severe issue:** `REQUIREMENTS.checked("REQ-ID", signal)` used the signal value of constant boolean and integer signals instead of the signal result.

# Declarations

## What's new

- **Automatically remove signals from the 'Initial Values view':** New message box during interface import allows the user to automatically remove deleted signals from the Initial Values view.
- **The incubation feature "Physical unit support" requires a license.**

## Bug fixes

- **Severe issue:** If a constant `A` of the type array was declared in the Declaration Editor, accesses to individual values of the form `A[index]` were calculated incorrectly. The array element with index `0` was always used regardless of `index`. This error only occurred with declared constants (and **not** with declared channels or parameters) and has now been fixed.
- In very rare cases, the default value could be lost during the interface import when the old value was casted due to an update of the definition of a named data type.
- Fixed bug that MDF files with empty data blocks could not be opened.

# Test cases

## What's new

- **Import test cases dialog extended:** In the last step of the Import Test Cases dialog, you can add new test cases to a new or existing test set (**Target test set**). The test cases of an existing test set can be either removed from the test set (**Unlink**) or deleted from the project (**Remove**).
- **'If step' with 'Once' and 'Always' semantics:** Choose **Once** to check the condition only in the first cycle of the step list segment; choose **Always** to check the condition in every cycle of the step list segment.
- **Textual test specification generation extended:** You can generate the test specification to the Description view and/or Test Case Details view formatted or unformatted. Additionally, you can decide to generate only the names of the elements used in a time partition test case, or the specification and transition actions of a step list, or both. You can also exclude variants or testlets from the specification generation.

**Test Specification Generator**

Generate test specifications for the selected test cases or variants and save it in the selected destination.

Output Format: ☒ Markup formatted text ☐ Plain text

☒ Generate initial value specification into  
Preconditions

☒ Generate test specification into  
Test Specification

☒ Names and contents ☐ Names only ☐ Contents only

De/select variants Excluded variants: 8

☒ Omit repeated assignments of the same value to a channel/parameter

☒ Generate assessment specification into  
Pass Conditions

OK Cancel

*New test specification generation dialog*

## Bug fixes

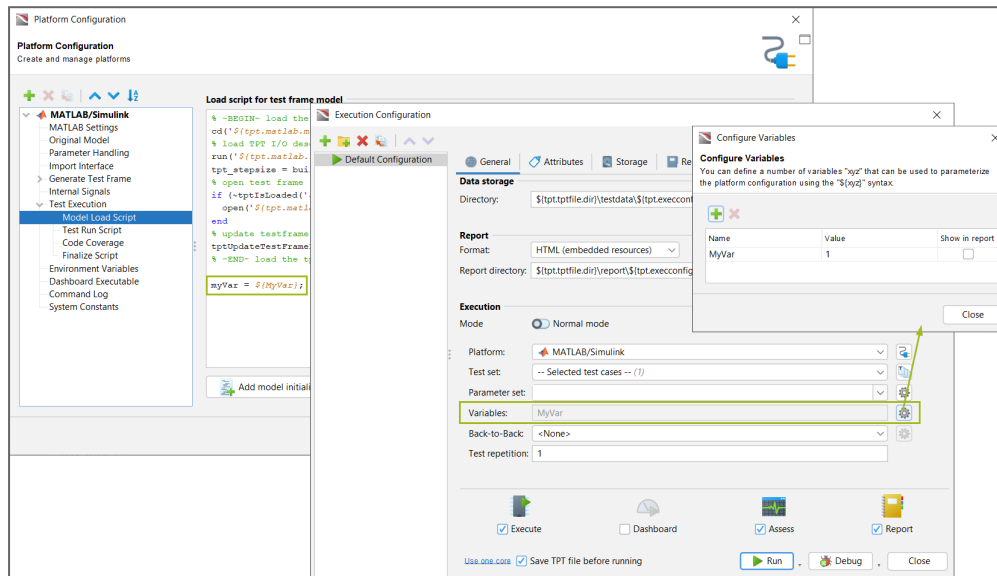
- Bug fix concerning the deletion of test case attributes in the Test Case Details view.
- Fixed bug that prevented copying disabled Testlet steps.
- Fixed bug concerning the export of test cases to ReqIF after deleting a test case attribute in TPT.

## Test platforms

### What's new

- **Usage of multiple DBC files for creating a CAN configuration supported:** When creating a new CAN configuration, you can select multiple DBC files via the CAN database field in the CAN Configuration dialog. The CAN configuration is created with the information from all selected DBC files.
- **Calculation of CAN signal values by custom CRC DLL improved:** You can set for all CAN signals with a length  $\leq 16$ -bit that a value is calculated for them by a custom CRC DLL in the CAN Configuration dialog. For a length  $\leq 8$ -bit, the method for 8-bit long CRCs from the DLL is used as in previous TPT versions. For signals  $>8$ -bit and  $\leq 16$ -bit, the method for 16-bit long CRCs is used.

- **Improved availability of custom execution configuration variables in the context of a platform configuration:** These variables are now available in the Model Load Script section, Code Coverage section and Finalize Script section of the MATLAB/Simulink platform, and in the Custom Script section of all platforms. In addition, these variables can be used for various other settings that allow `$variables` in the context of a platform configuration.



*Execution configuration variables specified in the 'Execution configuration dialog' and used in the 'Platform Configuration dialog'*

## Platform AUTOSAR

### What's new

- **Data received events supported:** Data received events are now imported and automatically triggered whenever data is received from either TPT or another connected component.
- **Asynchronous server call return events:** Asynchronous server call return events are now imported and automatically triggered. This is done at the end of the `Rte_Call` function as the server operation call is always done synchronously by TPT.
- **C typed per-instance memory (PIM) with type definitions that do not match implementation data types from ARXML files are now supported.** The data types are analyzed with a C parser.
- **Mapping based on record layout order:** The record layout settings of Com-Axis/Curve/Map/Cuboid Application data types are now matched to struct elements by order within the record layout and not by short label. Structs without COUNT element for one or more of the axes are supported.
- **AUTOSAR contract phase header options extended:** You can choose to generate compu method defines into the Application Types header file instead of into the Rte Types header file.
- **Port interface mappings:** TPT imports and processes port interface mappings during test frame generation for port connections.

- **Port-defined argument values on P-ports with C/S interface are supported.**
- **Struct data types with optional struct elements are supported.**
- **Values of variation point macros in parenthesis:** For Variation point proxy macros (`Rte_SysCon_...`) and Variation point macros (`Rte_VPCon_...`), the value is now always placed in parenthesis as it may contain a complex formula/calculation. This may affect the behavior in cases where such a macro is used in expressions like `#if !Rte_SysCon_MyConstant`.
- **The 'Use effective interface' checkbox of the 'Platform Configuration dialog' can be set via TPT API.**
- **The settings in the 'Code Coverage section' of the 'Platform Configuration dialog' can be set via TPT API.**
- **New button to filter sub-components:** Click the button **Filter sub-components** in the Platform Configuration dialog to exclude individual sub-components from a composition.
- **New checkbox 'Assume explicit access for all port elements':** Use this option in the Advanced Interface section to assume explicit access for all port elements for Sender/Receiver, Calprm, and Mode/Switch interfaces. This way, TPT always generates the corresponding Rte access functions, even if no access is declared in the ARXML files.
- **New checkbox 'Connect delegated data ports':** Use this option in the Advanced Interface section to import Sender-Receiver ports, Parameter interfaces and Mode-Switches even if they are delegated to other provided ports.
- **New checkbox 'Use data mapping from system':** Use this option in the Advanced Interface section to import channels for sender receiver elements with the name of the mapped system signals. In case more than one sender receiver element is mapped to the same system signal they will be imported as single channel.
- **Import parameters with curve/map application data types improved:** Select the new checkbox in the Advanced Interface section to import parameters with curve/map application data types of parameters as curve or map to TPT.
- **Setup for back-to-back tests between MATLAB and AUTOSAR platforms simplified:** New option in the Advanced Interface section to select a MATLAB/Simulink platform as interface names provider for an AUTOSAR platform. The rename mapping is set up automatically on interface import. This affects channels for which the AUTOSAR settings are present in the Simulink test frame model.
- **Visual Studio Code debugging:** New option in the Debug Settings section to choose whether to debug with Visual Studio Code or with another debugger, for example Eclipse CDT or Visual Studio. The Visual Studio Code debugger works independently of the compiler.

## Bug fixes

- AUTOSAR test frame generation via command line without explicitly setting the architecture (`x32`, `x64`) is possible again.
- Compiler initialization for Visual Studio with 64-bit mode by the AUTOSAR platform has been fixed to work again with Visual Studio 2013 and older.

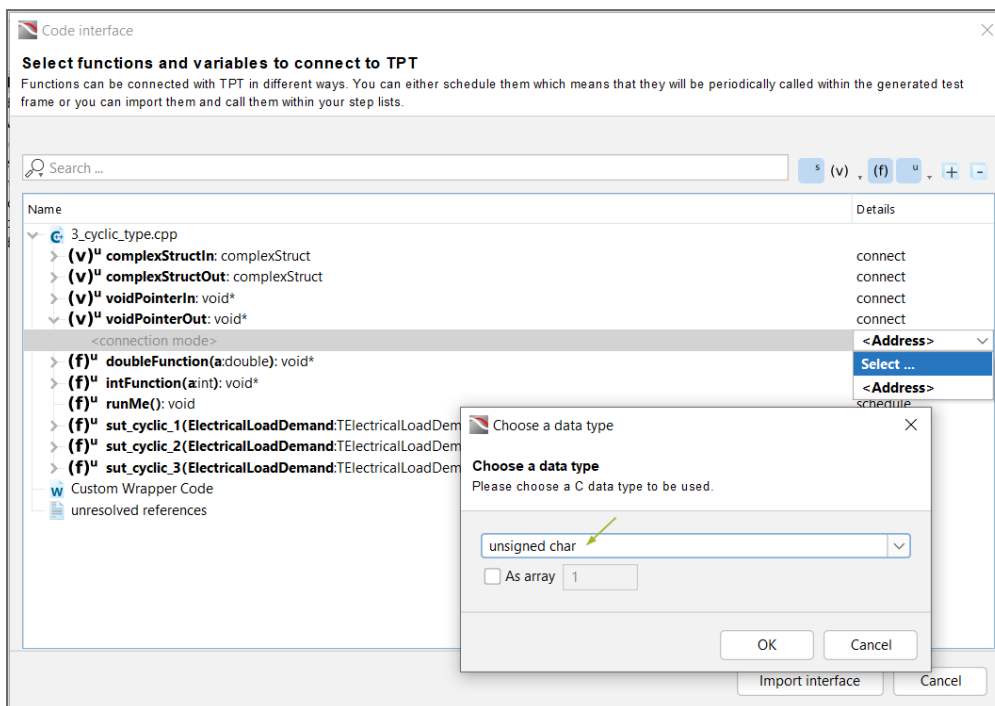
- An inner delegation and an outer assembly connector at the same PR-port on a AUTOSAR composition component type is now supported.
- When using the AUTOSAR platform, the `Rte_IsUpdated` function is now supported for PR-ports as well.
- Creating a Eclipse CDT project for debugging purposes via the AUTOSAR platform did not work with non-canonical project root folders ending with a parent directory (e.g. `${tpt.tptfile.dir}\..\`  ).

## Platform C/C++

### What's new

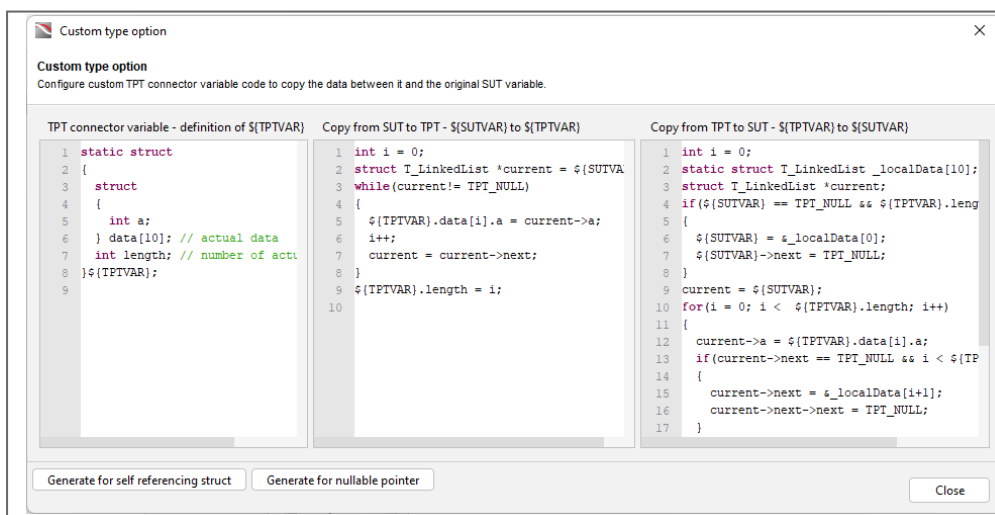
- **Preprocessor definitions are now imported as constants:** When importing C code, the preprocessor definitions which can be resolved to double or integer and whose identifiers are valid TPT names are now imported as constants. If the same definition appears in different files with different values, a warning is displayed and no import takes place. Default values of variables resulting from these definitions are now also resolved.
- **'TPT Coverage' now generates coverage reports that include code coverage in header files.**
- **During interface import via the 'C/C++ platform' from an A2L file, the imported A2L information (e.g. data type and scaling) is now mapped to struct and struct array elements.**
- **C++ reference data types are supported.**
- **The 'Use effective interface' checkbox of the 'Platform Configuration dialog' can be set via TPT API.**
- **The settings in the 'Code Coverage section' can be set via TPT API.**
- **New button to start TASMO:** Click the button **TASMO** in the Platform Configuration dialog to start the test case generation directly.
- **New TPT API functions for setting defines:** You can set defines in the Advanced Source Settings section via TPT API.
- **Defines with quotes (") supported:** If you want to use quotes (") in the Defines table in the Advanced Source Settings section, you need to escape the quotes (") with a backslash (\").
- **Visual Studio Code debugging:** New option in the Debug Settings section to choose whether to debug with Visual Studio Code or with another debugger, for example Eclipse CDT or Visual Studio. The Visual Studio Code debugger works independently of the compiler.
- **PiL test execution supported:** The C/C++ platform can now be used to run PiL tests via the TRACE32 Lauterbach debugger.
- **When using the Visual Studio debugger for debugging purposes, the paths to the libraries are automatically added to the VS projects.**

- 'Select ...' option allows to enter any C/C++ data type name when using the free text field.



'Select ...' option in the 'Code interface dialog'

- **New button to set up the platform faster:** Click the button **Import sources from CMake JSON file** in the Platform Configuration dialog to add the necessary information like source files, includes, and defines to the platform configuration by importing the information from a CMake JSON file.
- **Fully customizable connection to various data type structures for C variables:** Common cases can be done automatically. Set the connection mode to **<Custom>** to open this dialog.



New custom type option in the 'C/C++ platform'

- **Improved interface import:** The data types `char*` or `char array` can now be imported directly as strings.
- **New column 'Delay' in the scheduling table to configure functions to be scheduled only after a specified time:** It is now possible to configure a scheduled function to be executed only after a certain delay.
- **Struct element renaming for easier back-to-back testing:** The C/C++ platform test frame generation now allows external names for scalar channels that reference struct elements (and vice versa). This only applies to global C variables configured to be connected with TPT.
- Initializations of `structs` with member names are supported.

## Bug fixes

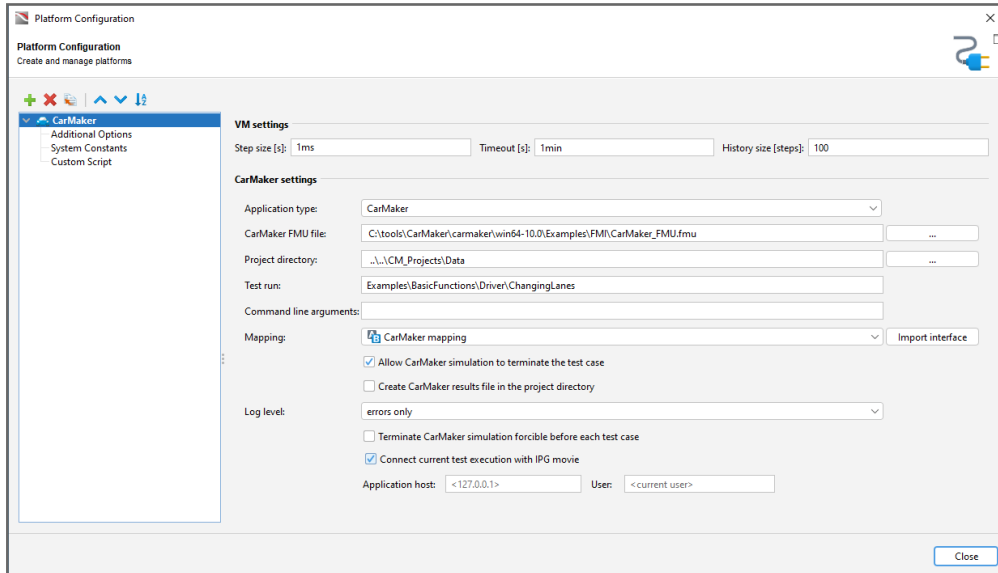
- Fixed bug when many include commands are used in a batch file. The include commands are now processed by using a response file.
- Compiler initialization for Visual Studio with 64-bit mode by the C platform has been fixed to work again with Visual Studio 2013 and older.
- Dependencies from the `stddef.h` file removed from the C/C++ platform. References to the `wchar_t` type removed from the `TPT_VM_API.h` file.
- It is now possible to synchronize the scheduling table in the C/C++ platform with the current source interface settings via TPT API.
- Fixed bug when using the `setProperties` TPT API method on a C/C++ platform configuration. The source file specific settings in the Code interface dialog are no longer lost.
- Fixed bug when importing the interface via the TPT API when using the C/C++ platform with a configured A2L interface file.
- Fixed bug when copying C/C++ platform configurations with option **Skip** or **Overwrite** in case of name conflicts in the Platform Configuration dialog.
- Creating a Eclipse CDT project for debugging purposes via the C/C++ platform did not work with non-canonical project root folders ending with a parent directory (e.g. `${tpt.tptfile.dir}\..\` ).



## Platform CarMaker

### What's new

- New 'CarMaker platform' lets you run tests on CarMaker, MotorcycleMaker, and TruckMaker.

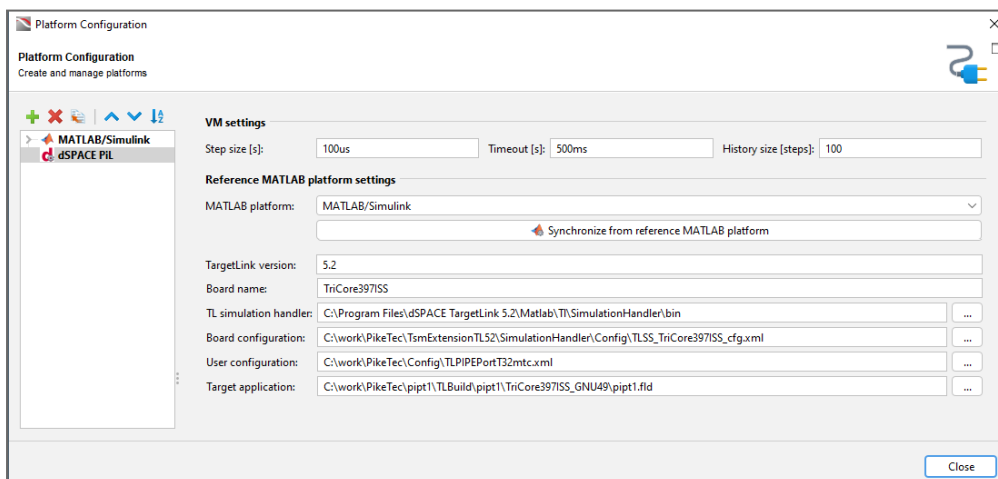


CarMaker platform configuration dialog

## Platform dSPACE PiL

### What's new

- New 'dSPACE PiL platform' lets you communicate with PiL boards or simulators via TargetLink without MATLAB/Simulink. The platform setup supports MATLAB R2015b and TargetLink 4.1 and higher.



dSPACE PiL platform configuration dialog

## Platform EXE

### What's new

- **Handling of external names for arrays improved:** In case of complex arrays and matrices, the handling of external names is more stable. This mostly concerns signals without explicitly defined external type and with an inner data type with enum constants.

## Platform FMI

### What's new

- **Mixed structs within an FMU are now supported:** If input channels, output channels, and parameters are mixed in one struct, the struct signal in TPT is imported as output signal with in/out semantics.

## Platform FUSION

### What's new

- **Co-simulation of 'FMI nodes' is supported.**
- **CRC calculation with custom DLLs is now supported for CAN messages longer than 8 bytes.**

### Bug fixes

- FUSION platform: The Parameter Exchange node now supports signal names longer than 255 characters.
- When `<inherit from platform>` was selected in the Arduino node, the mapping was ignored. This bug is fixed.

## Platform MATLAB

### What's new

- **Import of parameters with string data types in MATLAB (using " instead of ') is supported.**
- **MATLAB release name displayed:** In all text fields in TPT where the MATLAB version (for example, 9.1) can be selected, the MATLAB release name (for example, R2016b) is now additionally displayed.
- **Struct element names for MATLAB parameters may equal TPT keywords:** If a word reserved in TPT, for example, true or false, is used as a struct element name for a MATLAB parameter and the element is imported to TPT, the struct element is renamed and the original name is imported as external name.
- **Parameter exchange is now faster concerning large struct arrays.**

- **Changed behavior concerning parameter values in a step list:** Value changes of EXCHANGE or WRITEONLY parameters in a step list are still not forwarded to Simulink during the test execution but are now accessible in the assessment as well as displayed in the Signal Viewer. This could result in changes in the behavior of the test assessment.

### Bug fixes

- TPT input and output signals where the **Hidden** checkbox in the Rename mapping flavor is selected are now ignored when matching them to Simulink signals. This allows having multiple signals with the same external name at test execution as long as the **Hidden** checkbox is set for all but one.
- The proper constant is now stored within the TPT file when the default parameter exchange options are selected. This does as well affect the value for this option when accessing it via the property map from the TPT API.
- Fixed bug when writing parameters with string types in TPT that exist as string in MATLAB and with enum types that are not based on the `Simulink.IntEnumType`.
- 'If-action' control signals can now be connected if within bus signals.
- On interface import via the MATLAB/Simulink platform, importing channels via data store import settings can now properly handle dimensions explicitly defined as string expression instead of a numeric value.
- Import of offline loggings from Stateflow variables in TargetLink models now only imports variables that are configured for logging in TargetLink.

## Platform PLS UDE

### Bug fixes

- Bug fix concerning the "Update TPT signature" button.
- The number of UDE struct elements is now correctly compared with the TPT struct element count when writing and reading.
- A false positive test execution error when writing struct elements to PLS UDE succeeds is now removed.

## Platform Silver

### What's new

- **Arrays supported:** If array signals are used in TPT, TPT will exchange the array signals as flattened scalar signals with Silver. As example, an array `foo` in TPT will be scalars `foo[0]`, `foo[1]`, ... in Silver.
- **Better performance when using Silver with 64-bit simulation modules.**

## Platform VDT

### Bug fixes

- Better error handling when the connection to the Real Time Data Bus is broken.

## Test execution

### What's new

- **Signal comparison and back-to-back tests for MiL and SiL have been improved:** It is possible to load reference data from a measurement file stored in the reference directory to better compare MiL and SiL signals. Furthermore, the autocompletion feature now includes measurements (M).
- **Batch test execution via remote access:** Batch test execution can now be started from a running TPT instance via the command line `tpt.exe --remote --run batch <absolute path to .tptbatch>`. The output of the TPTBATCH file is redirected to the calling command line.
- **Relative paths in remote commands supported:** Remote commands like `tpt.exe --remote --run apiserver <file>` now allow relative paths to the current working directory of the calling command line. In previous versions this required either absolute paths or for relative paths the same working directory from which the remote TPT process was started.
- **Test repetitions using the 'Build Progress dialog' have been improved:** When executing tests again via the Build Progress dialog, the **Before execution startup script** of the Execution Configuration dialog is now executed again too.
- **Meta information available in 'testcase\_information.xml':** The information specified in the Report Meta Information assesslet or the information specified in the Script assesslets when `TPTReport.addMetaInformationRow()` is used, is now added to the `testcase_information.xml` file.
- **Execution of TPTBATCH files now always uses the '--batchmode' command line flag to suppress the loading of the UI:** Especially for batch files with many steps this means a significant improvement in runtime performance.

### Bug fixes

- The error recognition in Dashboard scripts is now restricted to syntax errors to prevent screen freeze while editing the script.
- TPT could not be launched on single core systems, for example on a VM emulator like VirtualBox.
- Batch execution via command line (`tpt.exe --run batch`) is possible again.

## Autotester

### Bug fixes

- Reading resume files works again without BLF entries. BLF entries are no longer required to read resume files.
- The prerequisites are now stored after a restart to avoid loss of information. The parameter values can be used in the assessment and report generation.
- Test cases with the same name are now executed again if they are located in folders or subfolders. And the candidates are now also created for them again.

## Docker

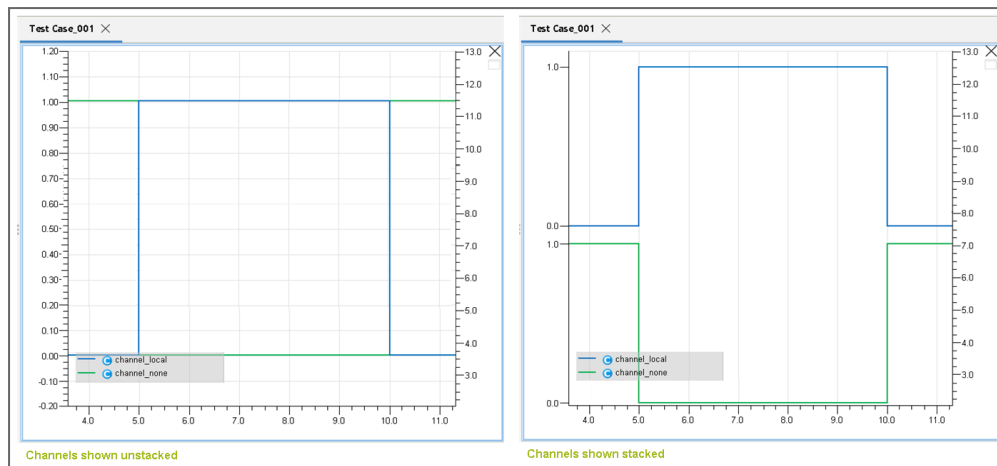
### What's new

- **TPT settings in Docker images improved:** The settings of the TPT Tool Preferences like paths to compilers are now easier accessible and editable when using a Windows-based Docker image.
- **PDF test reports in Linux-based TPT Docker images with Ubuntu 22.04 supported:** The example `Linux|0_Docker_Image|Dockerfile` supports the generation of PDF reports with Ubuntu 22.04. For UBUNTU versions up to 20.04, the commands are kept but commented.

## Test assessment

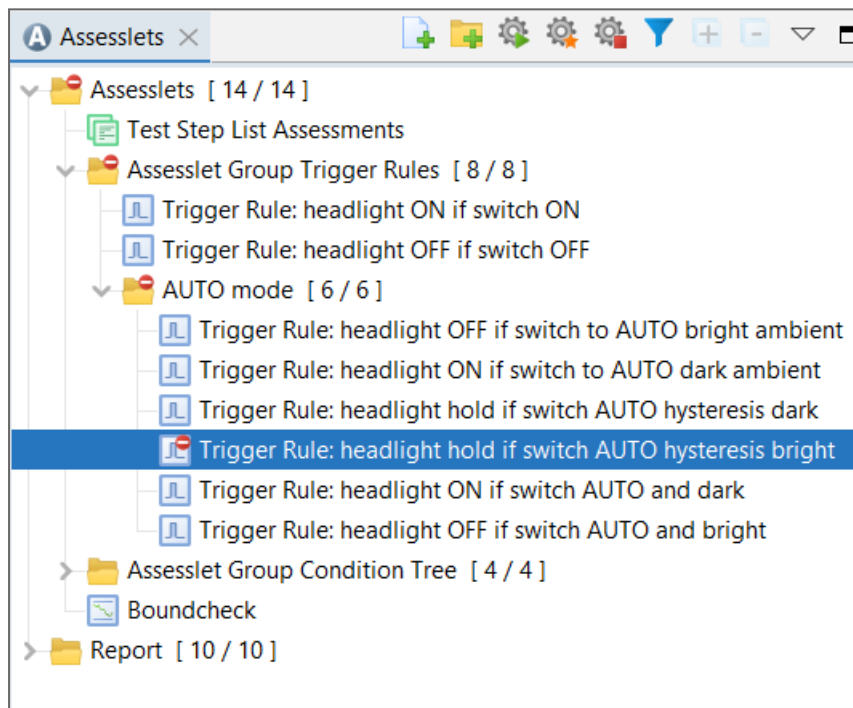
### What's new

- **'Separate signals' option in the 'Signal Viewer':** When you have multiple signals in a view, you can place each signal in a separate view by clicking the context menu entry **Separate signals**. This works also for elements of struct and array signals.
- **Equalize the heights of views in the 'Signal Viewer':** If several views have a different height, the views can be set to the same height by clicking the context menu entry **Equalize view heights**.
- **Display overlapping signals in the 'Signal Viewer' improved:** For better visibility, you can now stack signals in a view. Each signal will have its own y-axis.



Signals displayed unstacked (left) and stacked (right)

- **Error handling in 'Assesslets view' improved:** The folder containing an erroneous assesslet is now also marked as erroneous up to the top level.



The error flag is displayed up to the top level

- **Working directory of 'Script assesslets' changed:** The working directory of Script assesslets is now identical to the test data directory of the test cases. Previously, the installation path of TPT was used as working directory.
- **Behavior of 'Equivalence Classes assesslet' changed:** The Equivalence Classes assesslet returns an inconclusive result instead of a passed result if both the Mandatory column and Forbidden column are empty.
- **All assesslet results for requirements are integrated in the file 'test\_information.xml'.**

- **New 'Requirements Coverage assesslet' as global assesslet:** The Requirements Coverage assesslet replaces the Report Requirements assesslet. It contains evaluation criteria, integrity check settings, and report settings.

The integrity checks replace the RM Report options that could be set in the project-specific TPT Tool Preferences.

RM Report options (TPT 18)	Integrity checks (TPT 19)
Report unlinked requirements as issue	Each requirement should be linked to at least one test case, directly or indirectly via variants
Report not checked requirements as issue	Each requirement should be evaluated by an assesslet with Passed or Failed
Report an issue if requirements are not checked while executing linked test cases	Each requirement should be evaluated by an assesslet with Passed or Failed while executing linked test cases
Report an issue if requirements are checked while executing unlinked test cases	(removed)
Report an issue if requirements are not checked by assesslets they are linked with	Requirements should be evaluated by REQUIREMENTS.checked() in linked Script assesslets
Report an issue if requirements are checked by unlinked assesslets	No requirement should be evaluated by REQUIREMENTS.checked() in unlinked Script assesslets
--	Summarize the integrity check results as assesslet result

- **New and modified requirements assessment functions:**

```
REQUIREMENTS.getRequirementSet(boolean includeTextObjects)
```

lists all requirements contained in the used requirement set; can now contain headings and information objects

```
REQUIREMENTS.getAllRequirements(boolean includeTextObjects)
```

lists all requirements of the project; can now contain headings and information objects

- **Deprecated requirements assessment functions:**

```
REQUIREMENTS.getRequirementsLinkedToCurrentTestCase()
```

replaced by `REQUIREMENTS.getLinkedRequirements()`

```
REQUIREMENTS.getResult(String reqId)
```

replaced by `REQUIREMENTS.getRequirementResult(String reqId)`

- **New functions concerning the global requirements assessment:**

```
REQUIREMENTSGlobal.getRequirementResult(string reqId, global_testcase testCase)
```

returns the requirements result; can return the requirement result only in the context of a specific test case

```
REQUIREMENTSGlobal.getRequirementSuperSet(boolean includeTextObjects)
```

lists the requirements contained in at least one of the requirement sets used during the test execution; can contain headings and information objects

```
REQUIREMENTSGlobal.getAllRequirements(boolean includeTextObjects)
```

lists the requirements of the project; can now contain headings and information objects

- **New functions concerning the global test cases assessment:**

```
TPTGlobal.getTestCaseById(string testcase_id)
```

returns the first test case with ID 'testcase\_id' from the list of executed test cases

```
global_testcase.getRequirementSet()
```

lists all requirements contained in the currently used requirement set

```
global_testcase.getLinkedRequirements()
```

lists all requirements that are linked directly or indirectly to the test case

```
global_testcase.getRequirementResult(String reqId)
```

returns the result of the requirement with the passed ID



## Bug fixes

- Severe issue: The intervals discarded according to 'Abort'-conditions in Trigger Rules assesslets containing  $t$  might be discarded or retained under implausible but deterministic conditions. To fix this behavior for older TPT project files, open the TPT Tool Preferences and go to TPT Assessment Behavior and select the **Use of global "t" in Trigger Rule 'Abort' conditions** checkbox.
- When copying Equivalence classes assesslets, the settings of the two **Report unassigned equivalence classes** checkboxes are now copied as well.
- Fixed bug when the **Crop to fit** checkbox in the Signal duration fitting section of the Import Measurements assesslet was selected.
- Bug fix concerning accessing "array slices" such as  $x(t) := \text{myarray}[m:n]$ . If the addressed individual signals  $\text{myarray}[m] \dots \text{myarray}[n]$  from the array are not constant in the current context interval, such accesses have caused a runtime error. Now they are determined at each individual time  $t$ .

## APIs

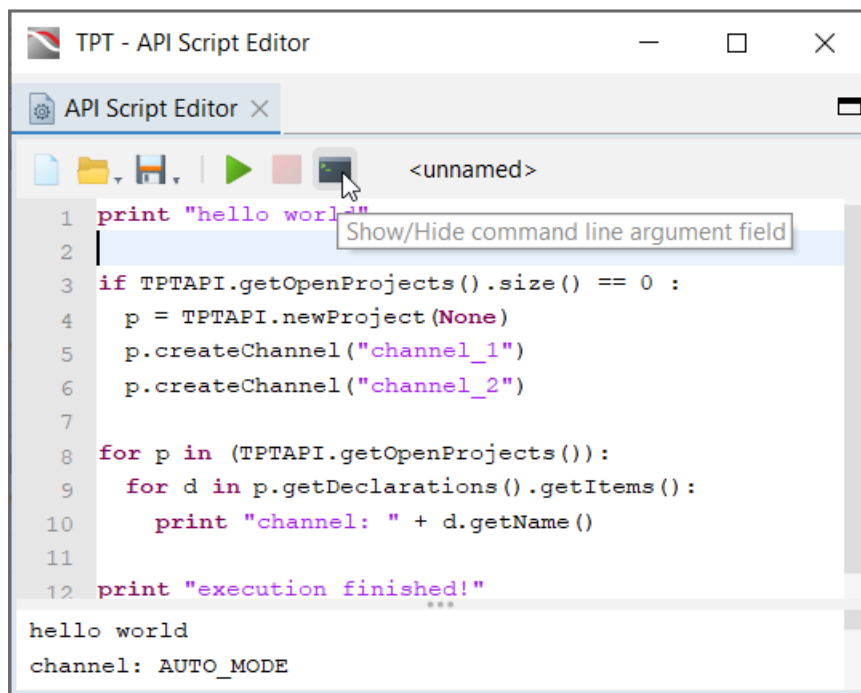
### TPT API

#### What's new

- **Slightly changed behavior when the 'tpt.exe --remote command' is executed and an error occurs:** If an already running TPT instance cannot be accessed with `tpt.exe --remote`, a new TPT instance is now started and the command to be executed is passed to it. The output of an remotely executed command will now be redirected to the calling command line. If this does not work, an error message appears on the command line.
- **Requirements can be imported or exported via TPT API.**
- **Working directory in 'API Script Editor view' changed:** The parent directory of a TPT API file opened in the API Script Editor view is now the working directory.
- **New assessment type 'RequirementsCoverage' replaces 'AdvancedReportSettings' methods:** The requirements related methods of the API class `AdvancedReportSettings` are deprecated and do not work anymore. Use instead the properties of the new assessment type `RequirementsCoverage`.
- **New TPT API method returns the settings of the last requirements import:** When using `Project.getLastImportRequirementsSettings(String documentName)`, you can now retrieve the settings of the last requirements import. An empty string as document name represents the `<default document>`.
- **New TPT API method to remove requirements attachments:** Use `Requirement.removeAttributeAttachment(String attributeName, Attachment attachment)` to remove a specific attachment of a requirement attribute. Use

`Requirement.removeAllAttributeAttachments(String attributeName)` to remove all attachments of a requirement.

- **New TPT API method to remove requirement attributes:** Use `Project.removeRequirementAttribute(String attributeName)` to remove a specific requirements attribute from the project.
- **MATLAB FUSION DLL generation via TPT API:** When using `buildModelForFusion(boolean prepareForBuildOnly)`, you can now generate a MATLAB FUSION DLL from the test frame specified in the MATLAB/Simulink platform. If the argument is set to `true`, the test frame FUSION model is prepared but the code generation will not be triggered.
- **Import of CMake JSON file to 'C/C++ platform' supported:** Use `importSourceFromCMakeJsonFile(File jsonFile)` to import the necessary information for the C/C++ platform like source files, includes, and defines automatically from a CMake JSON file.
- **The API method 'Project.getRequirementsDocuments()' delivers the names of all requirements documents contained in the project in alphabetical order.**
- **If the API scripts are not saved, the file names are marked with \*.** When you try to execute unsaved scripts, a dialog box opens asking if you want to save the script.
- **Stop button and command line argument field added to the 'API Script Editor view':** The command line argument field can be used to pass arguments to the script for testing, which can be accessed in the script via `sys.argv`.



'Show/hide command line argument field' button in the 'API Script Editor view'

## Bug fixes

- Fixed bug that made it impossible to set a mapping for a platform configuration via TPT API.
- Fixed TPT-API function `AutosarPlatformConfiguration.importIO` to properly consider the configured mapping and select the mapping automatically if a new one was created on import.

## VM API

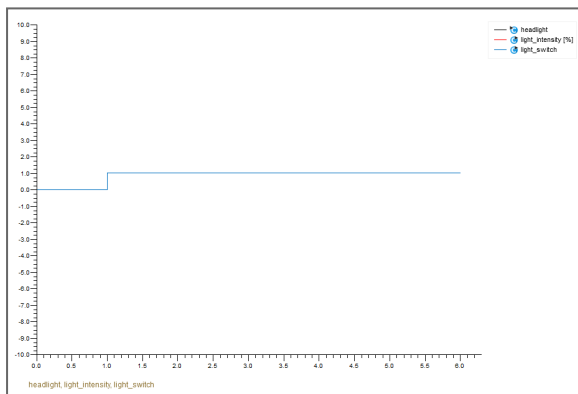
### Bug fixes

- Handling of the lower limit corrected which can be set via `tpt_vmapi_setLimits()`.

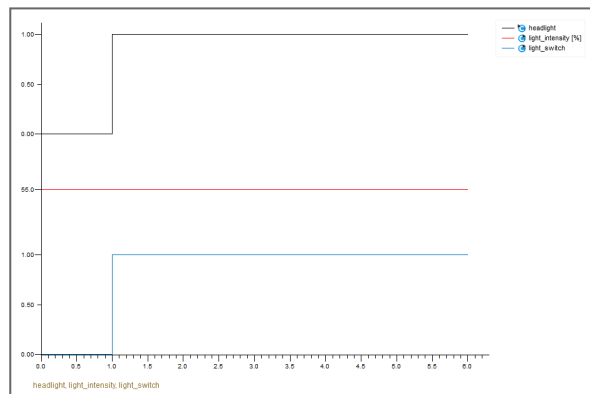
## Test report

### What's new

- **Show signals stacked:** To avoid overlapping signals in the graphic added to the test report, signals can be stacked. This setting is done in the Report Signal Graphic assesslet.



*Unstacked signal representation*



*Stacked signal representation*

- **When TPT Coverage (TASMO) is used, the relative path is now displayed instead of the folder hierarchy in the 'Coverage summary report page'.**
- **Requirements results table layout changed:** The column showing detailed information on how a requirement result was achieved has been renamed from Test Case Results to Detailed Results. The long explanations have been replaced by markers. To hide the Detailed Results column, deselect the **Show detailed results** checkbox in the Requirements Coverage assesslet.

Result	ID	Passed	Failed	Execution Error	Inconclusive	Detailed Results	Text	default value
Document: <Default document>								
	SPEC-1	-	-	-	-		Example Lights Control	
	SPEC-2	-	-	-	-		The headlights are operated depending on the light switch position and the light intensity of the surrounding area. The example "lights control" intends. If the light switch is turned on, the headlights will turn on. If the light switch is turned off, the headlights will turn off. If the light switch is run in automatic mode, there is a hysteresis applied on light intensity preventing the headlight from toggling in situations like undercrossing bridges and so on.	
	SPEC-3	-	-	-	-		Interface	
	SPEC-4	0	0	0	0		The signal light_switch shall be an input to lights control. It shall distinguish three modes: ON, OFF and AUTO.	
✓	SPEC-7	1	0	0	0	✓ OFF ON [ID=184] L.D	The signal headlight shall be provided by lights control. It shall distinguish two values: ON and OFF.	
✓	SPEC-13	1	0	0	0	✓ OFF ON [ID=184] L.D	If light_switch is ON, then headlight shall immediately be ON.	
✓	SPEC-6	1	0	0	0	✓ OFF ON [ID=184] L.D	If light_switch is OFF, then headlight shall immediately be OFF.	

Hints for detailed requirement results:  
L - Test case is linked to the requirement  
D - Requirement result is derived from test case result

'Detailed Results column' with markers

## Bug fixes

- Fixed bug concerning the generation of PDF test reports when Visual Studio 2015 redistributable is missing.

# Previous Release Notes

This section contains the previous release notes of TPT:

- [Release Notes TPT 18](#)
- [Release Notes TPT 17](#)
- [Release Notes TPT 16](#)
- [Release Notes TPT 15](#)
- [Release Notes TPT 14](#)
- [Release Notes TPT 13](#)
- [Release Notes TPT 12](#)
- [Release Notes TPT 11](#)
- [Release Notes TPT 10](#)
- [Release Notes TPT 9](#)
- [Release Notes TPT 8](#)
- [Release Notes TPT 7](#)

# Release Notes TPT 18

## Highlights

- **Formal requirements introduced:** Formal requirements facilitate the requirement-driven development. They are created and managed in the Requirements view. They belong to a requirement and can be assessed using the Check Formal Requirements assesslet.
- **'Rapid Accelerator Mode' and 'SiL Mode' used for the test execution with MATLAB/Simulink are now supported:** The TPT S-function now supports the use of the Rapid Accelerator Mode and SiL Mode on test frame level (with MATLAB R2017a and newer).
- **TASMO for MATLAB/Simulink and TASMO for C/C++ now supports MC/DC coverage.**
- **MC/DC coverage for C/C++ code can be measured and reported with TPT Coverage.**
- **Lightweight markup language available:** You can now add headings, lists, tables, and images in the Description view, Test Case Details view, and in the comments of your test steps in the Content view using the new Markup Editor.
- **'Batch Runner view' improved:** To create a batch file, use the new batch runner steps: Execute Tests step, Execute API Script step, Generate Testframe step, and Batch Overview Report step.

## General

- **'Visual Studio Code plug-in' replaces 'TPT File Viewer tool':** Use the Visual Studio Code plug-in "TPT Project Format Extension" to trace changes to TPT projects at file level and to resolve merge conflicts in versioned projects (<https://marketplace.visualstudio.com/items?itemName=piketec.tpt-project-extension>).
- The command line option `--split` works again. This option can be used to split large TPTBIN files into smaller files. For example, `tpt.exe --run dataconverter --split=<time> <input-tptbin> <output-tptbin>`.
- Undo and redo are now supported in the Batch Runner view.
- Fixed slow scrolling speed when using the touchpad.

## codeBeamer

- **Assignment of codeBeamer requirement types to TPT requirement types supported:** When importing requirements from codeBeamer, you can now determine how to treat requirements of a specific type in TPT and even exclude them from being imported at all.
- **Import of additional requirement attributes supported:** You can now select additional requirement attributes to be imported to TPT.

- **Import of requirements from codeBeamer changed:** Comments at requirements are no longer imported to TPT.
- **Import of test cases or requirements from a codeBeamer baseline supported:** You can now import test cases or requirements from a specific baseline of your codeBeamer project.
- **Import of links from codeBeamer is now optional:** When importing requirements or test cases from codeBeamer, importing the links is now optional.
- **Import of test cases from codeBeamer to TPT enhanced:** The import steps have been rearranged for better clarity.
- **Export of TPT test case IDs to codeBeamer supported:** To make it easier to identify test cases in TPT, TPT test case IDs can now be exported to codeBeamer.
- **Export a test case attribute value to codeBeamer choice list supported:** Assigning a single test case attribute value to a choice list in codeBeamer is supported as long it is not a multiple choice list. The item gets automatically selected in codeBeamer.
- **Export of TPT test reports supported:** You can now export the TPT test reports of the executed test cases to codeBeamer.


## Requirements

- **Creating requirements in TPT supported:** It is now possible to create requirements (🔗) and sub-requirements (🔗) in TPT and edit their content.
- **Create requirement attributes in TPT:** You can now create requirement attributes for requirements and sub-requirements created in TPT in the Requirements view or by using the API function `Requirement.setAttribute(string, string)`.
- **Requirements coverage summary in the 'test\_summary.xml' available:** The new `RequirementsSummary` tag lists the amount of requirements, how many of them reported an error, failed, passed, were inconclusive or uncovered. For example:  


```
<RequirementsSummary Errors="0" Failed="0" Inconclusive="0" Not-Covered="11" Passed="2" Requirements="13"/>
```
- **Export of requirements from TPT supported:** You can now export requirements from TPT to a CSV or Excel file by selecting Requirements|Export Requirements. You can either export all requirements or only those of a specific document.

## Declarations

- **Name column and signal icon column in the 'Declaration Editor' remain visible when scrolling:** The column for the signal icon and the column for the signal name are now fix in the Declaration Editor to make it easier for you to relate values in additional columns to a signal when scrolling horizontally in the editor.
- **Enumerations in DBC files can be imported:** When you have specified enumeration types in your DBC file, these enumeration types can now be imported as new data type to TPT.

- **Import/Export of mapping values of struct elements and functions from TPTAIF files supported.**
- **Refactoring feature for renaming equivalence classes is now available:** To enable this feature, either select the **Refactor on equivalence class rename** checkbox in TPT Tool Preferences|General settings , or click  **Refactor** in the toolbar of the Equivalence Class Set tab in the Declaration Editor, or click it in Tools|Equivalence Class Set Editor.
- **Interface import more user-friendly:** The following information is stored in the mapping and reloaded whenever you select this mapping as target mapping during the interface import:
  - synchronization method (name, external name)
  - filter setting: signal types (channel, parameter, constant, and more)
  - filter setting: import signal status (changed, unchanged, new, removed)
  - columns excluded from import
  - hidden/shown columns
  - mapping settings (hide new signals in other mappings, import rename to mapping, import values mapping)
- TPT now fetches the matrix/map values according to the selected **Deposit** value (**COL\_DIR** or **ROW\_DIR**). **Deposit** is an attribute of the A2L mapping flavor. This attribute is used to describe how the 2-dimensional matrix or map values are mapped to the 1-dimensional address space.
- Arrays and matrices where the value is specified in exponential notation or where `true` or `false` is set as Boolean value are now correctly imported from CSV files and Excel files.

## Test cases

- **'Reset parameter step' is replaced by 'Reset step':** The step has been generalized in the step list. In addition to parameters, the step can now be used to reset channels of the mode LOCAL, OUT, and NONE.
- **Textual test specification enhanced:** Testlet steps and Table steps can now be transferred to a textual test specification.
- **During test result export, reclassifications are considered:** This affects the test result export to Polarion and codeBeamer, and the test case export to CSV and Excel.
- **Accidental moving of elements can now be prevented:** The Content view toolbar now contains the  icon, which can be used to prevent automaton elements from being moved accidentally. The shortcut Ctrl+L can also be used.
- **New functions for 'Channel steps' and 'Parameter steps':**
  - `TPT.impulse()`
  - `TPT.ramp()`
  - `TPT.rampgradient()`



- `TPT.steps()`
- `TPT.sinwave()`
- `TPT.pulsewave()`
- `TPT.sawtooth()`

For more information about these functions, see [User Guide|Test cases|Modeling language reference|Built-in functions](#).

## Test platforms

- **Simulink Real-Time XiL platform introduced:** You can now run tests on Simulink Real-Time when using this platform.
- **Boost performance of ASAM XiL related platforms:** You can now minimize the number of online signals and/or exclude unused signals from being captured to boost the performance. The options are available in the Additional Options section of the CANoe XiL platform, dSPACE XiL platform, Simulink Real Time XiL platform, VeriStand XiL platform, and XiL platform.
- **Visual Studio debugging supported:** You can now create a Visual Studio project from within the AUTOSAR platform and C/C++ platform for debugging the generated test frame in Visual Studio via TPT.
- **Switch-case statements are supported by TASMO and TPT Coverage for measuring decision coverage.**

## Platform ASCET

- **Borland C++ compiler no longer supported.**

## Platform AUTOSAR

- **Automatic deletion of previously created test frames:** The previously generated test frames will now be deleted when starting the regeneration of a test frame in the AUTOSAR platform or AUTOSAR FUSION nodes.
- **Void pointer data types supported:** Void pointers are now supported for the communication between AUTOSAR software components. For communication between TPT and AUTOSAR software components, elements with `void*` as data type are also allowed but will be ignored by TPT.
- **Data type mapping enhanced:** For ports on composition component types, the data type mapping can now be derived from connected inner atomic components as well. Thus composition component types no longer need to reference a data type mapping to resolve application data types for the AUTOSAR platform.
- **To solve compilation issues for TPT generated C files, the stub header files are now included before the `Rte_<SWC>.h` and `Rte_type.h` entries.**

## Platform C/C++

- **C/C++ parser timeout value can now be set:** The C code is parsed to analyze the source code or to generate a test frame. If the process takes longer than the default timeout, parsing is aborted. The timeout value can now be set in the TPT Tool Preferences dialog.
- **Improved performance when analyzing and parsing C code or generating test frames by reducing memory consumption.**
- **Unresolved references can now be connected as client functions.**
- **TPT Coverage introduced:** In addition to CTC++ and GNU gcov, the coverage report can now be generated with TPT Coverage. TPT Coverage uses TASMO which generates a C/C++ code coverage report without using an external test coverage tool.
- **Automatic deletion of previously created test frames:** The previously generated test frames will now be deleted when starting the regeneration of a test frame in the C/C++ platform.
- In addition to function parameters, all variables connected to channels or parameters are now transferred to TPT when calling TPT server functions. This means that the current values of these variables are updated. In rare cases, this change may cause test cases to behave differently than before.

## Platform EXE

- **Parameter exchange arguments can now be set:** There is a new field in the new Advanced Execution Options section of the Platform Configuration dialog to add arguments for exchanging parameters during the test execution.
- **EXE test driver generation changed:** To ensure that C data types are used properly, `tpt_vmapi_getIntTypeEnum()` is now used in case of enum data types during the test frame generation.

## Platform FUSION

- **You can now run tests with 'FUSION XiL nodes' that use the same bit width (32-bit or 64-bit) but different ASAM XiL versions.**

## Platform MATLAB

- **'TPT Simulink Function Handler' block introduced:** To handle Simulink function stubbing, the new block is added automatically when generating the test frame with MATLAB R2018a or newer and the 'Stub Simulink functions called from within the SUT' checkbox is selected in the Platform Configuration dialog. This solves issues with Simulink if the input to a function caller block is driven by an output channel from TPT. Note that test frames generated with versions before TPT 18 must be regenerated or manually adjusted if this feature is used.

- **New option to run an M-script after the final test case:** You can now enter an M-script in the new `Finalize script` tab of the MATLAB/Simulink platform to run commands after the last test case has been executed.
- **Stateflow signals and state logging via internal signals from the 'Platform Configuration dialog' is now supported.**
- **Subsystems can be excluded from code coverage:** To check only specific subsystems concerning the code coverage by using `Simulink Coverage`, you can now exclude those subsystems that should not be checked.
- TPT now supports non-ANSI characters in file paths when using the MATLAB/Simulink platform. This does not apply if a LCC compiler is selected in MATLAB.
- Changed behavior when importing Simulink or TargetLink offline logging data:
  - If the `Enable direct feedthrough` option is disabled (default setting), the data for the last sample is no longer missing.
  - If the `Enable direct feedthrough` option is enabled, it may happen that the logging data now contains an excess sample.
- When running tests on multiple cores via the MATLAB/Simulink platform, the coverage data from these cores will now be merged in the report.

## Test execution

- **Run TPT tests on Windows-based Docker images:** Additionally to running tests on Linux-based Docker images, you can now also execute TPT tests on Windows-based Docker images. The corresponding Docker examples have been updated.
- **New dashboard widget available:** With the new `Matrix` widget, you can display and change the value of two-dimensional signals during the test execution.
- **Run custom scripts at the start or the end of the execution:** Set up Python scripts in the new `Custom Script` section of the Execution Configuration dialog. The scripts are executed at the start or the end of the test execution.

## Test assessment

- **New assesslets for global assessment available:** With the three new global assesslets - `Global Variable assesslet`, `Global Script assesslet`, and `Global Equivalence Classes assesslet` - all test cases of a test execution can be evaluated, the test results of all test cases can be summarized, and new calculations can be made from the available results of the individual test cases. The global assesslets are the successors of the `Global Assessment` section that was part of the Execution Configuration dialog.
- **New options in 'Equivalence Classes assesslet':** With the new option `Report unassigned equivalence classes` you can display equivalence classes in the report that have not been defined in the `Equivalence Classes assesslet` but for which values exist. You can also display equivalence classes in the report that have been defined in the `Equivalence classes assesslet` as not-forbidden but for which no value was found.

- The number of columns is reduced. The table only contains the columns **Covered Equivalence Classes**, **Non-Covered Equivalence Classes**, and **Values Found Out Of Equivalence Class Ranges**.
- **'Check Log Entries assesslet' with new option:** Select the new option **Assesslet is FAILED if pattern does not match** to mark the Check Log Entries assesslet as FAILED if the pattern does not match and as INCONCLUSIVE if the pattern matches.
- **Timed expression checks in 'Script assesslet' possible:** With the new assessment functions `TPT.getConstant(expr, starttime, endtime)` and `TPT.isConstant(expr, starttime, endtime)`, you can now check if an expression is constant between the specified `starttime` and `endtime` and what the constant value is.
- Assigning values to integer signals that are out of range of the particular data type is now forbidden in the assessment. Note that this can lead to incompatibilities to previous versions.

```
myUInt8(t) := -1 # execution error
myInt8(t) := 129 # execution error
```

Assigning values without (t) that are out of range of the particular data type to integer signals was already forbidden in previous TPT versions:

```
myUInt8 := -1 # execution error
myInt8 := 129 # execution error
```

- Missing data in external files now leads to an execution error when comparing signals during back-to-back testing.
- Concerning the results of the Signal Comparison assesslet, the test report now always displays a value in the **Max deviation from reference signal** column. For example, 0 means no deviation from the reference signal. The values are displayed in green if the result is PASSED and in red if the result is FAILED.

## Test management

- **Generate test specification feature extended:** The specification of initial values and assesslets belonging to a specific variant/test case can now also be generated into a test case attribute or variant/test case description.
- **'Status view' extended:** The Status view now shows the total amount of requirements and active assesslets linked to a test case.

## TPT API

- **There are now new interfaces in the TPT API for the global assesslets:**
  - `GlobalScript`
  - `GlobalVariable`
  - `GlobalEquivalenceClasses`

These interfaces have been inserted into the package `com.piketec.tpt.api.constants.assessments`. The `Global` interface has been removed from this package.

- **Properties of the 'Global Equivalence Classes assesslet' and the 'Equivalence Classes assesslet' can now be set or read.**
- **New method for synchronized test execution:** Use the new method `ExecutionStatus.join()` to wait for the execution to finish. You can optionally specify a timeout to set a maximum waiting time, for example, `ExecutionStatus.join(100)` would wait a maximum of 100 s.
- **New methods for displaying global assesslet results and global assessment results:**
  - `ExecutionStatus.getGlobalAssessmentStatus()`  
returns the global assessment result of the overall test execution
  - `GlobalAssessmentStatus.getOverallResult()`  
returns the worst result of all executed Global assesslets as string (`ResultError`, `ResultFailed`, `ResultSuccess`, or `ResultUnknown`)
  - `GlobalAssessmentStatus.getAssessletResults()`  
returns the result for each executed Global assesslet
  - `GlobalAssessletStatus.getConstraintResults()`  
returns the constraints table results of a single Global assesslet
  - `GlobalAssessletStatus.getScriptResult()`  
returns the script results of a single Global assesslet
- **String arguments supported:**

The following methods can now handle both `java.util.regex` patterns and string arguments:

  - `Project.getAssessmentOrGroupByNamePattern()`
  - `Project.getExecutionConfigurationOrGroupByNamePattern()`
  - `Project.getMappingByNamePattern()`
  - `Project.getPlatformConfigurationByNamePattern()`
  - `Project.getScenarioOrGroupByNamePattern()`

**Example with `java.util.regex` pattern argument:**

```
from java.util.regex import Pattern
scenarios = prj.getScenarioOrGroupByNamePattern(Pattern.compile(
    ("^Test.*")))
```

Example with string argument:

```
scenarios = prj.getScenarioOrGroupByNamePattern("^Test.*")
```

- Global assessment status is now also taken into account for the total test result.
- The test result is now determined according to the following priority list from high to low: ResultError, ResultFailed, ResultSuccess, ResultUnknown.

## TPT VM API

- **Non-ANSI characters in path names can now be used:**

In the TPT VM API, there are now new methods for using `wchar_t` instead of `char`:

- `tpt_vmap_i_runTestcase_v2()`
- `tpt_vmap_i_exchangeParamsFromSuT_v2()`
- `tpt_vmap_i_exchangeParamsToSuT_v2()`

## Test report

- **Step List assesslet tables order changed:** For each executed state variant, a Step List assesslet table is added to the report. The tables are now arranged as follows:
  - the earlier the start time, the higher in the list
  - in case of the same start time: the longer the duration, the higher in the list
  - in case of the same start time and duration: the larger the table, the higher in the list
- **Linked requirements displayed:** The requirements linked to test cases are now displayed in the Test Case Summary table in the report when you have selected the **Show linked requirements** column option in the Execution Configuration dialog.

# Release Notes TPT 17

## Highlights

- **Docker now supported:** You can now run TPT in a Docker container to execute and assess tests. You can split tests using TPT API scripts and run them in several containers of the same Docker image.
- **Run tests on several XiL systems simultaneously:** You can now execute tests on one FUSION Platform that contains several XiL nodes. This way, you can run tests at the same time for example on CANoe (via a CANoe Node), on VeriStand (via a VeriStand Node), on dSPACE (via a dSPACE HiL Node), and on a system supporting the ASAM XIL API standard (via a XiL Node).
- **New Service steps for easier function handling:** All platform dependent functions can now be set using the new Service steps. This applies to CAN, CANape, CANoe, DiagRA D, dSPACE XiL, INCA, LABCAR, Sound, VeriStand, VTD, and XiL. Click [Add service step](#) in the Content view to select a step.
- **New version control feature:** TPT projects can now be versioned using Visual Studio Code that supports versioning with GIT and SVN via plug-ins. With the free additional TPT plug-in for Visual Studio Code, you can open files with conflicts like invalid XML, duplicate or missing UUIDs and solve these conflicts. UUIDs can be shown in clear text.
- **TPT can now be connected to IPG CarMaker:** Via the new CarMaker Platform and the new CarMaker FMU Fusion Node, a connection to CarMaker is now possible. Note that this feature is an incubation feature and must be enabled before it can be used.
- **RT-LAB supported:** To run TPT tests in real-time on an RT-LAB model use the new RT-LAB Node in the FUSION Platform.
- **DiagRA D supported:** You can now connect to the diagnostic software DiagRA D during test case execution and perform diagnostic requests by using the new DiagRA D Node in the FUSION Platform.
- **Access TPT from MATLAB/Simulink via TPT Toolbox:** The TPT Toolbox can be used to open a TPT project from MATLAB/Simulink, to set up a TPT project for a subsystem, and to configure internal signals for offline logging. It can be used with MATLAB R2017b and newer.
- **Computing with arrays is now much more flexible:** It is now possible to form array slices whose boundaries are dynamically determined at runtime based on arbitrary expressions. This allows flexible array assignments, copying columns or rows of a matrix into an array, copying partial arrays, ramping array slices, and much more. Supported slices are:
  - from fixed start to fixed end: `[start:end]`
  - from first array index to fixed end: `[ :end]`
  - from fixed start to last array index: `[start:]`

- full array range: `[ : ]`
- negative index addresses from end of array, for example `[-3 : -1]`
- usage of variables in range expressions, for example `[0 : myVar]`
- arbitrary slices of multidimensional arrays, for example `myCube[0 : 1][ : ][-1]` with `myCube` being a three-dimensional array
- slices on the left and right side, for example `array_a[1 : 3] := array_b[4 : 6]`

## General

- **TPT has a new look and feel:** Text and icons are displayed crisp and clear. Readability and usability are improved, and HiDPI is supported. Above that, a dark mode is now available.
- **Maximum Java memory heap size adjustable:** You can now specify a maximum Java memory heap size in the General Settings of the TPT Tool Preferences. By default, 3072 MB is set; the maximum is 64000 MB.
- **MDF import improved:** Half precision float data types (16-bit) can now be imported from MDF files of version 4.20. During import, these data types are converted to float data types (32-bit).
- **The performance has been significantly improved when importing large structured signals or large test data for the assessment.**
- **The Debug Expressions view now supports autocompletion.**
- Managing TPT licenses is now supported when FLEXlm is running on a Linux server.

## Requirements

- **Requirement coverage statistics with extended filter functions and export option:** A new filter option in the coverage statistics window allows you to display the coverage for all requirements, requirements of a specific document, or the requirements that are currently displayed in the Requirements view. Additionally, you can select a test set to check which test cases of the set are linked to the selected or displayed requirements. You are warned about unlinked test cases. The document coverage can be exported to a CSV file.
- **Semantics of check boxes in the Requirements view has been simplified:** A checkbox shows if none ☐, some ☒, or all ☒ of the selected test cases, variants or assesslets are linked to the requirement.
- **Requirements selection history available:** The 10 recently selected requirements are listed in the Requirements view; click on an entry to select the requirement in the requirements table.
- **It is now possible to create and delete links to requirements for multiple selection via the checkboxes in the Requirements view.**
- **codeBeamer:** codeBeamer Version 20.11-LTS Carmen or higher is required due to changes regarding the REST API and the associated model.



- **Requirement import from Polarion changed:** Instead of the field IDs, the field name is now used.
- **New assessment function to access all requirements:** To access all requirements of your TPT project, use the new function `REQUIREMENTS.getAllRequirements()` in the Script assesslet.
- **New assessment function to access requirements linked to a test case:** To access only requirements that are directly or indirectly linked to the currently running test case, use the new function `REQUIREMENTS.getRequirementsLinkedToCurrentTestCase()` in the Script assesslet.
- Fields that are hidden in codeBeamer are no longer taken into account when importing and exporting requirements.
- When using the simple query for selecting the requirements to be imported from codeBeamer, the items are now automatically ordered by the outline numbers in codeBeamer instead of ordered by their IDs.

## Declarations

- **New tool for renaming multiple signals at once:** Use the new Rename Selected Signals tool to change signal names in the Declaration Editor. Multiple selection is possible. The Java syntax for regular expressions is used.
- **Namespaces and struct element paths can be copied:** You can now easily copy the complete path of a struct element like `my_struct.elem_001` from the Declaration Editor with Ctrl+Alt+C and paste it for example into a step list or assesslet. Ctrl+Alt+C will also copy the namespaces a signal uses, for example `namespace_1::namespace_2::my_signal`.
- **Array or struct element values can now be set in the Initial Values view:** It is now possible to set a value for a single element of an array or struct in the Initial Values view.
- **Initial Values view lists only signals for which an initial value has been set:** The table in this view is now initially empty. Channels and parameters can be added or deleted line by line. The **Defined** checkbox is no longer needed.
- **Testlet library mapping import and export supported:** Instead of adding the mapping manually, you can now import the mapping from a CSV file. You can also export the mapping to a CSV file for later use.
- **TPTAIF format changed:** If you are parsing TPTAIF files in an external tool, you might need to adjust your external tool or script because of the following changes:
  - the section `[ENUMTYPES]` is deprecated
  - the section `[CONSTANTS]` no longer contains enum constants
  - the declaration attribute `+ENUMTYPE` is deprecated
- **Interface export to a TPTAIF file extended:** The interface export from TPT to a TPTAIF file contains now a section called `[TYPES]`. This section lists all reusable data types including their definitions as declared in the TPT Type Editor.

## Test Cases

- **'Always' option in Call function step added:** If selected, cyclic function calls are now possible at each cycle.
- **Excel sheet selection in Import signal step possible:** When you select an Excel file, the Import signal step has now an additional field named to select an Excel sheet.
- **Behavior of step list expression  $x$  and  $x(t)$  harmonized:** Expressions in step lists no longer return an error if the value of  $x(t)$  is not yet defined at the current time. Just as with the time-independent value  $x$ , either the last known value from the previous calculation step, or the value specified in the Declaration Editor is used.
- **\$ variables can be used in step lists and automatons:** A \$ variable can be a scalar constant by data type or a string. You can insert them into expressions used in step lists, transition specifications, or in the Initial Values view.
- **Enumeration groups in Test Case Details view supported:** For better clarity, you can now group test case attributes of the type **Enumeration (select one)** by putting a slash after the group name in the Edit Test Case Attributes dialog. This shortens the display in the drop-down menu when selecting an attribute of this type in the Test Case Details view.
- **Test case import from Polarion changed:** Instead of the field IDs, the field name is now used. Importing and exporting the test case description from and to Polarion is now optional.
- **Log info when exporting test results to codeBeamer:** Instead of the complete log as it appears in the Build Progress dialog after executing a test case, only the logging data of the assessment is now transferred to codeBeamer when you export test results.
- When you delete history entries of an assesslet or test case in the Status view, the revision number and the change column content of the remaining history entries are kept.
- Fields that are hidden in codeBeamer are no longer taken into account when importing or exporting test cases.
- When using the simple query for selecting the test cases to be imported from codeBeamer, the items are now automatically ordered by the outline numbers in codeBeamer instead of ordered by their IDs.
- Using the **from scaling** checkbox in the Compare step is now only allowed, if the left hand side is a declared signal: Due to these changes, it could happen that TPT projects which were executable with TPT 16, now report a compiler error.

## Test Generation

- **Create step lists from textual specifications:** Use the new **Generate Test Implementation** option to transfer textual test specifications into step lists.
- **Extended random value generation settings for equivalence classes and output signals in step lists:** The value generation via `random()` can be specified in the TPT Tool Preferences for a TPT project. The settings are now also available for the creation of random numbers for equivalence classes. There are three random value generation methods:

- new random values with each test run
- random values for each test case that are kept no matter how often you run the test
- random values based on a specific seed value

## TASMO

- **Switch Case block types are now supported by TASMO:** These blocks are now considered for decision coverage.
- **Action subsystems are now supported by TASMO:** The content of Action subsystem blocks is now considered for coverage.
- **TASMO overview report is now available in HTML format:** The report can be exported from the TASMO Test Data Generation dialog. Or it can be displayed directly in the Coverage Summary section of the test report when selecting the **Enable measurement during execution** checkbox in the Platform Configuration dialog.
- **Improved test generation:** TASMO can now stimulate signals of array-of-structs data types in the test case generation. To support this data type, the range and initial value input has changed. Only scalar values can now be entered in the minimum and maximum fields of the range settings and in the initial value field. Specifying a minimum/maximum/initial value for different elements of an array is no longer supported. A scalar value is applied to all array elements.
- **TASMO settings are made easier:** The wizard has been replaced by a dialog that is easier to use. The input specification is automatically derived from default values - and in the case of a Simulink model, also directly from the model. You only need to make changes to the input specification and the coverage goals if you disagree with the automatically set values.
- **Input specification for TASMO more user-friendly:** The input specification is now done in a user-friendly tree table, so all input specifications are visible at the same time. For all values, multi-editing is enabled. This makes the specification much easier and faster. And in case of any inconsistency, a warning or error is displayed directly at the affected cell.

## Test Sets

- **Lock test sets to prevent unwanted changes:** To prevent unwanted changes, test sets can be locked. New test cases are by default deselected in a locked test set.

## TPT API

- **Set mapping attribute values for struct elements:** Use the new `setMappingFlavorColumnValue(declaration, <struct-element>, <mapping attribute>, <value>)` method to set a value to a mapping attribute for a struct element. Use `getMappingFlavorColumnValue()` to read the value of a specific struct element.
- **Passing API script arguments via command line:** Arguments ensure that scripts can behave differently. Arguments can now be passed when calling API scripts from the command line,

for example, `tpt.exe --run apiserver myScript.tptapi firstarg "second argument" argument3`.

- **New method to generate textual test specifications:** A test specification from a step list can now be generated into text by using the new TPT API method `String StepListScenario.exportTestSpecification()`. To import and apply a textual specification via the TPT API, use `void StepListScenario.importTestSpecification(String)`.
- **New methods to return the elements selected in the Project view and the Assesslets view:** With the following new TPT API methods, you can now determine which elements are selected in the Project view and the Assesslets view. This allows you to build more flexible scripts that refer to a currently selected element.

`getSelectedProject()`  
returns the selected project

`getSelectedTestlet()`  
returns the selected testlet

`getSelectedScenarios(boolean traverseSelectedGroups)`  
true = returns the selected test cases or variants, and test cases or variants contained in the selected groups; false = returns the selected test cases or variants

`getSelectedScenarioGroups()`  
returns the selected test cases folders or variant folders

`getSelectedAssessments(boolean traverseSelectedGroups)`  
true = returns the selected assesslets and assesslets contained in the selected groups;  
false = returns the selected assesslets

`getSelectedAssessmentGroups()`  
returns the selected assesslets folder

- **New TPT VM API function supports 32-bit coefficients:** The function `vmapi_setIndexCoeffs_v2()` specifies the coefficients as `uint32` instead of `uint16`. This function replaces `vmapi_setIndexCoeffs()` which is now marked as deprecated.
- You can now call the method `RemoteCollection.retainAll()` in the API Script Editor to remove all collection elements except those of the given collection, and the method `RemoteCollection.deleteAll()` to remove all elements of the given collection.

## Platforms

- **HTML sources can be excluded from the CTC code coverage report:** New **Disable HTML sources in CTC report** option in the following test environments: ASCET@FUSION Platform, AUTOSAR Platform, C/C++ Platform, and EXE Platform.

- **Scaling results can be rounded in the AUTOSAR Platform and the EXE Platform:** With the new option **Round scaling results**, you can now automatically add the function call `tpt_vmapi_setRoundScalingResults()` to the generated test frame. This will cause scaled values to be rounded to the nearest integer value before written to an integer-based data type.
- **New names for XiL platforms:** XiL platform names have been changed as follows:
  - from CANoe@FUSION Platform to CANoe XiL Platform
  - from VeriStand@FUSION Platform to VeriStand XiL Platform
  - from dSPACE HiL@FUSION Platform to dSPACE XiL Platform
  - from XiL@FUSION Platform to XiL Platform
- **Initial values for output channels are read from the SUT if the 'Read initial values for output channels from SUT' checkbox is selected:** If this checkbox is selected in the C/C++ Platform or the EXE Platform, the initial values for the output channels are read from the SUT before the test execution. This has the effect that the initial values from the SUT are prioritized over the default value set in TPT but not over the value defined in the Initial Values view. For the AUTOSAR Platform, this behavior is always enabled for static memory signals in the user interface.
- **Optional performance boost for several test environments available:** Select the new **Use effective interface** checkbox in the Platform Configuration dialog of the AUTOSAR Platform, C/C++ Platform, EXE Platform, or CANoe Platform to include only those signals of a test case in the runtime interface that are read or written at least once in the test model, the assessment, or in the dashboard (if dashboard execution is enabled). The limitation to used signals increases the performance because only the signals that are "important" for the specific test case are exchanged and recorded


#### Platform ASCET@FUSION

- **The ASCET-DEVELOPER tool as of version 7.8 is now supported.**
- **Automatic TPT plug-in installation in Eclipse CDT:** When using the ASCET-DEVELOPER version, an **Install TPT plug-in** button is now available in the Platform Configuration dialog to install the TPT plug-in in Eclipse CDT automatically.
- **When using the ASCET-DEVELOPER tool:**
  - Additional C sources and libraries included in the ASCET-DEVELOPER project are now automatically used for test frame generation.
  - Any data type is supported for the interface import or code generation, in particular, arrays of structs and structs of arrays.

#### Platform AUTOSAR

- **More consistent postfix generation:** When AUTOSAR application data types are imported into TPT and their names already exist in TPT but differ in their definition, TPT now automatically appends an "\_xxx" postfix starting at "\_001". The postfix "\_x" known from

previous TPT versions is no longer used.

- **AUTOSAR signals with the data type uint64 can be tested:** To make it possible to run tests with AUTOSAR signals of the `uint64` type, these signals are now treated by TPT as `int64` signals. The test driver C code is generated by TPT with the correct data type `uint64` but the signals will be treated as `int64` during the test execution.
- **TPT API implementation for AUTOSAR Platform:** The AUTOSAR Platform can now be set up via the TPT API. The interface import and the test frame generation can also be started via the TPT API.
- **Relative references in ARXML files are now supported.**
- **Import of system constants from ARXML file supported:** System constants can now be imported from an ARXML file. To overwrite the values from the configured system constant value sets of the ARXML files, select the checkbox **Allow editing and override values from system constant value sets** in the System Constants section of the AUTOSAR Platform.
- **Additional interface files supported:** To use variables in the test execution that are not defined in an ARXML file, you can now add them via additional interface files (TPTAIF, XLS, XLSX).
- **AUTOSAR imported data type names changed:** In case of primitive data types that are associated with a compu method, TPT will now always use the short name from the AUTOSAR data type that originally defines the compu method to be used. This may be the name of the application or implementation data type. Due to renamed imported data types, compatibility issues in case enumeration constants are used in the test modeling may occur.
- In the AUTOSAR Platform Configuration dialog, the paths of the **Include folders** in the SWC Code section are now specified relative to the configured AUTOSAR project root folder. This is now properly reflected in the include paths table, e.g. when adding a new folder via the  **Add** button.

## Platform C/C++

- **Separated compiler and linker options:** The compiler and linker options have been separated. In case you load older TPT files in TPT 17, you must review these settings. By default, the commands `-static-libgcc -static-libstdc++` are included to link the standard libraries of C and C++ statically into the testframe.
- **Unions supported:** Variables with `union` data types are now supported. For each variable (or struct-element of a variable with a struct data type), the `union` data type element that shall be used, can be selected from a drop-down list in the C/C++ Platform. Union data types are also supported for function arguments and function return values.
- **Improved exchange of pointer function arguments:** For not implemented (unresolved) functions, the option **arguments as channels** can now be selected in the Code interface dialog of the Platform Configuration dialog. The functions are stubbed by TPT to connect function arguments with pointer data type as output or input channels and function arguments with non-pointer data types input channels.

- **Option renamed:** The option `stubbed by TPT` for stubbed functions has been renamed to `server-function`.
- **Unresolved variables with constant- pointer and pointer to constant data types are now supported.**
- **Non-ANSI characters in Windows user names and path names supported:** The C/C++ Platform is now capable of parsing source code files if the Windows user name or the path name contains non-ANSI characters (like Chinese or Japanese characters).
- **Struct element names in C may equal TPT keywords:** If a word reserved in TPT, for example, `true` or `false`, is used as a struct element name in a C file and the element is imported to TPT, the struct element is renamed and the original name is imported as external name.
- **Struct data types import despite unsupported elements:** You can now use struct data types even if they contain some elements that are not supported by TPT. The unsupported elements will be ignored and the struct data types are imported without these elements.
- **Function-internal variables of class functions can now be connected to TPT:** To record the values of function-internal variables of class functions, you can connect them to TPT. If there is more than one instantiation of the class function, the last value of the function-internal variable will be recorded; it is not recorded which instance called the function.
- **Separated compiler options for the custom wrapper code:** The C/C++ Platform has now a separated field to specify compiler options for the custom wrapper code. Due to this separation, you can use compiler options for the source code files for example written in C and different options for the wrapper code written in C++.
- **Dashboard Player Export supports executable from C/C++ Platform:** The C/C++ Platform can now be selected in the dialog of the Dashboard Player Export to export the executable for the test execution in the Dashboard Player.
- **Enumeration data type support extended:** The C/C++ Platform now supports enumeration data types that use a data type as base which is defined via a `typedef` statement.
- **Address arithmetic supported:** For pointer and array data types there is now an option to connect the address instead of the data to make it accessible in TPT. This option can be selected within the sub-tree for C variables or function arguments.
- **Data type selection for pointer of data type void:** For accessing void pointer data types, a data type that corresponds to the data type of the pointer listed in the C file can be selected within the sub-tree for C variables or function arguments.
- **Connecting elements with pointer data types as arrays is now supported:** For variables, function arguments, function return values, or struct elements with pointer data types, there is now an option to connect them as an array and specify its size. The same option is available in case of the data type `void*`, where the user additionally can select the underlying data type.



## Platform CANoe

- **Network names as prefix for CANoe bus signal names are now supported:** Besides the database name, the network name can also be used as a prefix.

## Platform FUSION

- **32-bit and 64-bit Custom Node .NET nodes in the same FUSION Platform:** Multiple Custom Node .NET nodes can now be added to the same FUSION Platform, regardless of whether they run in a 32-bit or 64-bit environment.
- **.NET framework support:** The .NET framework 3.5 is no longer supported. Nodes built with .NET framework 3.5 must be rebuilt with .NET framework 4.0.
- **32-bit and 64-bit FusionAPI DLLs renamed and moved to the same folder:** The new file names are now `PikeTec.FusionApi32.dll` and `PikeTec.FusionApi64.dll`. Both files are located in the `<tpt-install-dir>\public\lib\>` folder. FUSION nodes built in previous TPT versions must be rebuilt using the new path and files.
- **GNU Debugger Node, PLS UDE Node, and Lauterbach Node with improved read-write error handling:** There are three new options in the Advanced Options of the debugger nodes for a better read-write-error handling:
  - **Test execution error on read failure**  
Finishes a test execution with execution error whenever a channel or parameter cannot be read from the debugger target.
  - **Test execution error on write failure**  
Finishes a test execution with execution error whenever a channel or parameter cannot be written to the debugger target.
  - **Create channels that count failed reads and writes.**  
This button creates the channel `debugger_tpt_read_status` that counts the failed reads, and the channel `debugger_tpt_write_status` that counts the failed writes.
- **Performance improvement concerning debugger nodes:** The communication log messages of the GDB Node, PLS UDE Node, and Lauterbach Node during the test execution are now written into a log file instead of into Build Progress dialog to increase the performance.
- **PLS UDE controllers for read/write actions can now be addressed:** The controller can now be specified in the Declaration Editor by using a suffix or in the Breakpoints section of the PLS UDE Node dialog by entering a **Controller Index**. If you use multiple cores in a UDE controller, you can address in the **Core Index** column.
- **PLS UDE run argument list:** You can now specify in the Program section of the PLS UDE Node dialog as argument of the `run()` command a list of breakpoints instead of a single breakpoint. The first breakpoint in this list is the target breakpoint which defines the next step of the program.



- **Synchronization via INCA signal during runtime:** The signal can be used to synchronize INCA and TPT, even if the SUT is not executed in real time.
- **MATLAB/Simulink offline loggings in generated FUSION node enhanced:** When creating FUSION nodes from a MATLAB/Simulink model, offline loggings connected as measurements when running a test with the generated FUSION node.
- **CAN messages now also selectable via multiple selection:** In the CAN Configuration dialog, it is now possible to set multiple messages to IN/OUT/NONE via multiple selection.
- Leave the table in the Program tab of the GDB Node, PLS UDE Node, and Lauterbach Node empty if the sequence of breakpoint processing is not important for the program execution. In this case, the debugger is started by TPT and a new FUSION cycle begins at each reached breakpoint.

## Platform MATLAB/Simulink

- **Simulink bus element names are automatically converted to TPT-compliant names during interface import:** Simulink bus elements with names not allowed in TPT will now automatically be renamed on interface import and the original name will be imported as external name for the struct-element in TPT.
- **Action Port signals supported:** Action signals are imported as channels with boolean data type to TPT. In case of an output channel, the subsystem will be called when the boolean value is `true`. In case of an input signal, each call results in a high-low-change.
- **Subsystems with "Bus Element In" and "Bus Element Out" blocks for input/output ports are now supported.**
- **Parameter values automatically casted during parameter write:** If the data type of a parameter in TPT differs from the data type in the MATLAB model, the value is automatically casted to the data type of the parameter used in the MATLAB model.
- `uint64` parameters are imported as `int64` parameters. If the value exceeds the `int64` value range, the parameter is not imported.
- The data of previous TargetLink simulations is now automatically cleared before the test execution. This is done by the `tpt_clear_logs` function which is automatically added to Test Run Script when adding a signal via the Internal Signals section of the MATLAB/Simulink Platform.

## Test Execution

- **Storage of test cases with non-ASCII characters:** Many external tools have problems with non-ASCII characters in path names, for example when writing in Chinese. The folder names in the execution storage directory are therefore escaped with "\_" for each non-ASCII character. This is also true for the folder containing the execution data like the test report. To nevertheless recognize the test case, a TXT file is automatically added that displays the original test case name with the non-ASCII characters you have used in the TPT project.

- **New layout directory option for data storage:** Use the new **Flat layout with indexes and unique IDs (based on test case tree)** directory layout to save the execution data using indices for the folder structure and unique IDs for the test cases. The pattern is as follows:  
`testdata|<platform name>|<folder index>_<sub-folder index>_..._  
Testcase_<test case ID>.`
- **Test case IDs displayed in the Build Progress dialog:** When the test case IDs are shown in the Project view, they are now also displayed in Build Progress dialog.
- **Batch Runner view more powerful:** You can now add a TPTAPI or Python script in the new **Pre-execution .tptapi/.py script file** field. This file is executed before the batch test execution; argument values can be set in the **Command line arguments for your pre-execution script**. To run a script after the batch test execution, use the new **Post-execution .tptapi/.py script file**.
- **TPT can handle empty test sets:** If a test set with no activated test cases is assigned to an execution configuration, you can still start the test run although the execution configuration will have no test results. The **Run** button in the Execution Configuration dialog is no longer disabled. This way you can execute several execution configurations where test sets are generated dynamically, for example when the test set is based on requirements, and could be empty. The Build Progress dialog will show a message that the test set is empty.

## Test Assessment

- **New assessment methods in Script assesslets to access test case attributes:** The test case attributes as specified in the Test Case Details view can now be accessed with the new method `.getAttribute("<attribute name>", "<value>")` and the test case attribute value can be set or changed with the new method `.setAttribute("<attribute name>", "<value>")`. Both methods are bound to the test case instance `TPT.getTestCase()`.
- **New assessment functions in Script assesslet to trace server function calls:** To see how often a server function has been called, use the new assessment method `<server_function_name>.getCallCount()` in the Script assesslet. To see when the server function was called, use `<server_function_name>.getCallTimes()`.
- **Maps supported in Script assesslets:** Similar to creating curves using the function `TPT.createTable2D()`, you can now create maps in Script assesslets with the function `TPT.createTable3D()`.
- **NumPy-like calculations possible:** With the new object `TPTNumpy` you can run the following methods you might know from NumPy in the Script assesslet:

```
TPTNumpy.add  
TPTNumpy.array  
TPTNumpy.identity  
TPTNumpy.matmul  
TPTNumpy.matrix  
TPTNumpy.subtract  
TPTNumpy.zeros  
TPTNumpy.linalg.det  
TPTNumpy.linalg.eig  
TPTNumpy.linalg.inv
```

- It is now allowed to exit a function directly with "return" from within "during" statements. In previous TPT versions this was not officially supported and could lead to erroneous context intervals in the further course of the assessment.
- When the **Show sub elements checkbox** in the Report Signal Table assesslet was deselected in TPT 16, the overall result for an array with boolean elements was not displayed in the test report. This is now fixed.

## Signal Viewer

- **Improved CSV-export:** When exporting signal values to a CSV file, ";" instead of a tab is now used as a delimiter. If no value is assigned at a specific point in time, the field in the CSV file is left empty.
- **Signal description available in the Signal Viewer:** The signal description from the Declaration Editor is now displayed in a tooltip of the Signal Viewer and the Debug Signal Viewer.
- **Signal Viewer handling of special floating point values:** The Signal Viewer distinguishes intervals where the signal is not defined (n/a) from values like NaN and +-Infinity. Undefined (n/a) as well as NaN and +-Infinity values will be displayed in the view as empty interval. The reason for this empty interval is shown in the Sample Point Table of the signal as well as in the signal list to the left.

## Test Report

- **Link to test case in test report:** Click **Show in TPT** in the test report to open the test in TPT. The test will be selected in the TPT project.
- **Gcovr supported:** When you have instrumented and compiled your C code with the GNU compiler, you can now generate the code coverage report of the TPT test report with gcovr.

# Release Notes TPT 16

## Highlights



- **Generate test specifications automatically from step list:** You can generate test specifications from implemented test cases, simply at the click of a mouse.
- **Generate test implementation from test specification:** You can generate test implementations from test specifications in simple text and paste them into TPT via keyboard shortcuts Ctrl+C and Ctrl+V.
- **Edit and copy test specifications:** You can also edit your test specifications and copy them into TPT via keyboard shortcuts Ctrl+C and Ctrl+V.
- **Build your own Arduino HiL:** Build your own mini-HiL from an Arduino. Measure and control with analogue and digital I/O and real hardware.
- **Extended MATLAB Simulink support:** Extended Simulink support including structured signals, message signals, and function stubbing.
- **C++ and automatic test harness generation:** Testing C++ code is made easier including classes and automatic test harness generation.
- **Namespaces used for AUTOSAR, C++ and HiL:** Use namespaces for logical grouping of signals and better readability of TPT projects.
- **More AUTOSAR support:** With AUTOSAR, you can now also specify static and constant memory and a central repository of ARXML.
- **Modify and send CAN messages:** You can modify and send CAN messages directly for BUS protocol testing.
- **Support of ALM via codeBeamer and Polarion:** TPT also comes with direct ALM, requirements and test management support by interfacing to codeBeamer ALM and Polarion.
- **Simplified concurrent editing:** And for joint working on TPT projects we simplified the file format and made a specialized viewer to solve merge conflicts.
- **Show dependencies of artefacts in TPT projects:** With the new "Usage view" it is very simple to show where artefacts such as assessments, test sets, testlets and so on are being used.

## General

- **Files opening in running TPT instances:** The TPT file types \*.tpt, \*.tptz, \*. tptprj, \*.tptbin, \*. tptbatch, and \*.tptviewerpref that have been linked to TPT during the installation process, will now open in the currently running TPT instance when they are double-clicked.

- **New automatism for finding compile errors via script:** You can now check your TPT file for errors automatically by calling `tpt.exe --run compiler <path to your TPT project file>`.
- **New view provides better overview:** By right-clicking on an element, for example, a testlet, and selecting **Show Usage**, all elements using the selected item are listed in the new Usage view.
- **Optimized interpolation of maps and curves:** TPT now calculates with a more precise arithmetic when interpolating maps and curves.
- **Autocompletion has been rebuilt:** In general, autocompletion can be accessed by Ctrl + Space and works roughly as follows: When you enter a string into a text field or editor, it is first split into a sequence of substrings. Substrings are split at capital letters or underlines "\_". The autocompletion then tries to match all of your sequences. For example, when entering the string `LT_i`, you can find the signals `light_Intensity_Max`, `light_intensity_min` and `Light_switch`.
- **Automatic setting of quotation marks is no longer supported:** When creating custom menu items, arguments with spaces, for example, inserted by `$variables` specified in the `externalcommands.xml` file must now be enclosed in quotes manually.
- **New tool for opening files that contain conflicts:** The new external TPT File Viewer tool that comes along with the TPT installation helps you to better understand changes and resolves merge conflicts within a TPT project. It also allows you to open files with conflicts like invalid XML, duplicate or missing UUIDs and to solve these conflicts. UUIDs can be shown in clear text.

## Search/Filter

- **Introduced test result filter:** For a better overview, you can now display only the test cases with a specific test result of the last test execution by clicking  in the Project view.
- **Introduced test set filter:** If you have test sets, you can now display only the test cases of one or more selected test sets by clicking  in the Project view.
- **Introduced name filter:** In addition, a filter feature is now available in the Project view. Testlets, variants or test cases can now be filtered by their names.
- **Introduced test case attributes filter:** You can now also filter test cases so that only the test case(s) with the entered attribute value are displayed.

## Requirements

- **Requirement gets result from assesslet:** If a requirement is linked to an assesslet and in this assesslet it is specified that it should fail when the precondition is never met or when the time context is not matched, the linked requirement will now also be marked as being failed.
- **Supported codeBeamer ALM:** Requirements can now be imported from codeBeamer to TPT. Test cases and test results can now be exported from TPT to codeBeamer.

- **Document version bound to individual requirements:** The document version is now linked to the individual requirement. As a result, requirements of the same document can have different document versions.
- **Manually set document version is applied to all selected requirements to be imported:** When entering the document version manually, all new, changed, and unchanged requirements of your project will receive the new document version. When you decide to not update a changed requirement in the last step of the requirements import, it will keep its old document version.
- **Document version can now be displayed in test report:** There is a new check box in the Execution Configuration dialog to display the requirements document version in an additional column.
- **Three new methods available to ask for the requirement type:** The following three methods are now available `REQUIREMENT.isRequirement()`, `REQUIREMENT.isHeading()`, `REQUIREMENT.isInformation()`

## Declaration

- **Larger arrays supported:** Arrays can now have more than  $2^{16}$  elements.
- **New Type Definition Editor:** With the new editor, creating data types is more convenient than ever.
- **Export old and new values during the interface import:** When importing an interface, you can now export the items marked as "to be modified" to an Excel file that displays the old and new values.
- **New option to guess renames based on name similarities:** When at least one "New" and one "Removed" signal exists in the interface file to be imported, you can now let TPT guess possible renaming. The matches are displayed in an extra dialog and can be accepted or discarded individually.
- **Namespaces supported:** Namespaces can now be used in declarations to keep signals identifiable if they share the same signal name. Just enter the namespace in the **Using namespaces** text field in the Content view, Assesslet Content view, and Functions view to refer to the signals by their signal name only. Note that the fully qualified signal name can still be used in the different views.

## Test Case Modeling

- **Change variant and test case types with one click:** You can now change the variant type or test case type from Step List to Time Partition and vice versa. While the content of the variants or test cases will be deleted, the descriptions, the links to requirements and assesslets, as well as the test case details are kept.
- **Generate test specifications from variants or test cases:** You can generate the test specification from test cases into an attribute of the test case description.

- **Edit and copy test specifications:** You can copy the test case description text into a step list. If this text is pasted, TPT attempts to create appropriate steps. Embedded signal, Import signal, Table and Testlet steps are currently not supported.

## Test Sets

- **"Automatic Include feature" replaced by new selection semantics:** To automatically include all current and future sub-elements of a folder to a test set, just select ☒ the folder in the Test Set Definition dialog. The semantics of the selection icons at a folder have slightly changed.

## Test Case Import/Export

- **New wizards for the test case import and export:** The test case import and export has now a step-by-step approach which makes the import and export easier and faster.
- **Polarion integration is now a regular feature.**
- **Test case and test result export to Polarion:** Test cases as well as test results can now be exported to Polarion.

## Test Management

- **Improved file version control (e.g. in Git or SVN):** Instead of names, objects are now referenced in XML via UUIDs. As a result, renaming the object only leads to local changes and the references remain unchanged. The sorting in the TPT file does not change by renaming objects or even by moving the objects in the automaton.
- **Working in a team on the same TPT project is more convenient:** Assesslet and test case IDs now get automatically a prefix when the TPT project is saved as TPTPRJ. This makes it more convenient to work in a team on the same project. The assesslet and test case IDs are now strings.

## RMI API

- **Autocompletion and tooltips in the API Script Editor:** The API Script Editor in TPT has now autocompletion. The documentation of the functions can be shown in a tooltip.
- **Changed file extension for TPT RMI scripts:** \*.tptapi is the new file extension for TPT RMI scripts, so you can easily recognize the TPT RMI scripts.
- A double-click on the API script and the command lines `tpt.exe --remote <API script>` and `tpt.exe --remote --run apiserver <API script>` will open it in the currently running TPT.
- **RMI API more comfortable:** `RemoteCollection` and `RemoteList` now have an `asIterable()` method that returns an iterable view of the collection.
- **TASMO Test data generation via RMI API:** With the function `PlatformConfiguration.initTasmoTestDataGeneration(Mapping mapping)`,

the test data generation with TASMO can be initialized. This function returns a controller object that can be used to monitor and control TASMO.

- **New binding functions for supporting array sizes of 65536 or greater:** `tpt_vmap_i_bindSignal_v2`, `tpt_vmap_i_bindSignalDynamic_v2`, `tpt_vmap_i_bindSignalGetSet_v2`. The following functions have thus become obsolete: `tpt_vmap_i_bindSignal`, `tpt_vmap_i_bindSignalDynamic`, and `tpt_vmap_i_bindSignalGetSet`.
- **Extended API for access to the C/C++ platform available**

## Platforms

### Platform AUTOSAR

- **Cross-project files supported:** In addition to the project root folder, you can now specify a repository folder to use files across different projects.
- **CMake support:** When you generate the test frame, TPT automatically generates a `CMakeLists.txt` file that can be used with CMake from version 3.0.2.
- **Static Memory and Constant Memory definitions can now be imported to and integrated into TPT:** The Static Memory definitions are integrated into TPT as output channels with read/write semantics. The Constant Memory definitions are integrated into TPT as parameters.
- Multiple P-Ports writing data for non-queued port elements to the same R-Port is now supported.

### Platform C/C++

- **TPT now supports C++ natively.**
- **Channel names for arguments in scheduled functions can be set manually:** It is now possible to manually set the channel names for each argument in a function listed in the Source Interface dialog. Functions using the same argument name with the same data type will be connected to TPT with a shared channel.
- **Improved source file copying mechanism:** Only files that need to be changed, for example, due to instrumentation for TASMO, are copied during test driver generation. This leads to better performance.
- **Source files via #include statement:** If no TASMO instrumentation is selected, the original source files will be included into the TPT test frame via #include statement instead of a copy.
- **C++ class instances can now be connected to TPT:** The public fields and public functions of the classes can be connected like variables and functions outside of classes.
- **Select multiple C/C++ files in one step:** It is possible to select multiple C/C++ files in one step by using the new "Select C Source Files" dialog.



- **Automatic saving and restoring of the expanded files and classes in the C code interface UI:** The currently expanded files in the "Current settings for the generated C code" dialog are now automatically saved and restored when you reopen the dialog.
- **Modifier const or volatile are now supported:** Variables with those modifiers can now be connected to TPT.

## Platform FUSION

- **FUSION:** TPT can run test cases on external hardware via the Arduino board by using either the FUSION Arduino Node or the Custom Node DLL. TPT stimulates the pins of the Arduino board to control the SUT. Signals can be read from the Arduino board.

## Platform MATLAB

- **Multiple function call signals triggering different subsystems with dependencies are now supported:** The TPT-S-function will act as a single function call source and generate function calls based on a trigger-semantic if the option **Natively use bus/enumeration/fixed-point data types specified by SUT** is enabled.
- **New option to automatically fix activation issues for internal signals:** Select the **Fix automatically** check box in the Matlab test platform to fix issues concerning the activation of internal signals in the test frame for Simulink and TargetLink during test execution.
- **SUT output message signals supported:** SUT output message signals are now imported as TPT input signal of the data type struct with the elements "data" and "status". "Status" informs you if a message has been received at that sample, "data" shows the most recent value that has been received. Simulink uses message signals in the context of Adaptive AUTOSAR.
- **SUT input message signals supported:** SUT input message signals are now imported as TPT output signal of the data type struct with the elements "data" and "enable". "Enable" must be active (set to 1) to sent one message at that sample. "Data" holds the current value to be send (if "enable" is set to 1).
- **Stubbing of Simulink functions:** Functions called by function caller blocks that are not implemented will now be imported as server functions to TPT. The Simulink model will call those server function via the test frame.
- **New option available for native use of data types specified by SUT:** If activated, the TPT S-function will natively use enum/Bus/fixed-point types specified by the SUT. Bus signals will be represented as structs in TPT (instead of multiple scalar signals).
- **Bus array signals are now supported:** If the new option to natively use SUT data types is enabled, bus array signals are now supported by TPT.
- **New option available to work with or without direct feedthrough for input ports of the S-function:** If enabled, unit delay blocks are added for the inputs. If disabled the required the delay will be handled within the S-function.

- **Subsystems without output channels are now supported:** Now only one input/output channel or - if function stubbing is enabled - one server function is required.
- MATLAB versions older than MATLAB 7.10 (R2010a) are no longer supported by TPT.

## Test Execution

- **New command line option for TPTBATCH file execution:** TPTBATCH files can now be executed using the `tpt.exe --run batch <tptbatchfile>` command line.
- **New Batch Runner view:** The view replaces the TPT Batch Test tool that was started from the Windows start menu. Using this view, it is possible to run multiple execution configurations in multiple TPT project files.

## Autotester

- **Improved interruption handling:** In case of test drive interruptions, the previous state can be restored so that the tests can be continued and do not have to be repeated.
- **Relative widget icon path:** The widget icon path can now be relative to the TPT project file.

## Debugger Nodes

- **Supported measurements in GNU Debugger node, PLS UDE node and Lauterbach-Trace32 node:** In addition to channels and parameters, measurements can be used in these debugger nodes.

## Test Assessment

- **New method `TPT.GetMapping()` available:** The new method can be used in the Script assesslet to find out which mapping has been used during the test execution. You can also restrict the execution of an assesslet to the usage of a specific mapping by typing `TPT.GetMapping()=="MappingName"` in the **Precondition** text field of the assesslet.
- **"Automatic Include feature" in the execution context settings "Variants" and "Test Cases" replaced by new selection semantics:** To automatically include all current and future sub-elements of a test cases or variants folder to an assesslet, just select ☒ the folder in the corresponding execution context setting in the Assesslets Content View. The semantics of the selection icons at a folder have slightly changed.

## Signal Viewer

- **Condition tree saved with Signal Viewer preferences:** If the Signal Viewer preferences are saved, the current condition tree is now also saved.

## Report

- **Report archive settings support wildcards:** You can now exclude or include files from the report archive using the wildcards "\*" for no or several characters, and "?" for one

character. For example, "setting\*.xml" added to the **Exclude filter** field, excludes all XML files which names start with "setting", like "setting\_one.xml", "setting\_one\_extended" and so forth.

# Release Notes TPT 15

## Special features

- **Automation of in-vehicle-tests:** Autotester is a new software testing tool which allows "drive-by" test case executions. While you are driving a test vehicle, Autotester automatically recognizes the actions you perform and gives you instructions when needed and executes the corresponding test cases of your TPT project.
- **Calculations with physical units supported:** Units can be used in the test design and assessments. The consistency of units can be checked.
- **Global assessments introduced:** To check min/max constraints, mean values, amount and sum of a variable for all test cases of the test execution use the new global assessment. The global assessment can be set up for each execution configuration.
- **Polarion supported:** Siemens provides an ALM software called Polarion. TPT supports the exchange of requirements and test case exchange with Polarion.
- **Vires VTD supported:** TPT supports testing with Vires Virtual Test Drive (VTD). ADAS testing in a virtual environment is possible. For the communication with VTD scenarios can be run. SCP commands can be used, and signal data can easily be exchanged.

## Declarations

- **New Scaling attributes:** The test execution platform potentially uses scaling for signals so that values in TPT internally have a different data type and scaling than in the SUT. The two new Scaling mapping attributes DT\_Scale\_Min and DT\_Scale\_Max represent the minimum or maximum value of the signal in TPT that results in the minimum or maximum possible value of the data type in the SUT.


## TASMO

- **Explicit values instead of range setting possible:** For TASMO the search ranges for variables is either specified as minimum-maximum value and a quantization between. Alternatively, a list of explicit values can now be provided instead of a range.
- **Coverage goals changed:** TASMO coverage goals for the Simulink blocks Saturation, Relay and Dead Zone are now the same as the MATLAB V&V/Coverage Toolbox analyzes.

## Test Case Import

- **Hyperlinks import possible:** Test case details can be imported from Excel. When importing test cases from an Excel file, hyperlinks can now be imported as clickable links to the Test Case Details of the test cases.

## Test Case Design

- **Documentation step specification extended:** The default font size as well as other style options for the appearance of the Documentation step in the step list and in the report can be specified in the preferences.
- **Import Signal step performance improved:** The performance of the data import from MDF-4 files into the Import Signal step has been improved.
- **Compare step with tolerance:** The "Compare" assessment step supports the comparison with relative and absolute tolerances.
- **Copy/paste in Initial Values view:** Initial values can now be copied and pasted to a test case or test case folder.
- **Copy/paste of variants:** Variants of testlets or transitions can be copied to other testlets or transitions in automats by using the  button. This works only for testlets/transitions and reference testlets/transitions.

## Test Set

- **Overview test cases in test sets:** Use the new dialog **Test set overview matrix** to find out to which test sets a test case belongs to.
- **Group test sets:** Test sets can now be arranged in folders (Test Set groups).
- **Test sets based on variants:** You can now configure test sets based on testlet variants by using `TESTCASE.getAllSubvariantsComments()` in the condition field of the Test Set Definition dialog.
- **Syntax highlighting:** The condition field of the Test Set dialog has now syntax highlighting. Autocompletion is also available. Press Alt+Enter for a line break.

## Test Environments

- **New structure:** Library files, header and tptapi.jar can be found in folder `<tpt-installation-folder>\public`.

## AUTOSAR

- **Pointer Data Types supported:** AUTOSAR Pointer Data Types (Data Reference Types) are now supported. When reading/writing values TPT will always access the current address.
- **New option to configure custom compiler options:** In the AUTOSAR Platform Configuration under SWC Code there is a new option to configure compiler options applied to C files.
- **External Trigger Events and Trigger interfaces supported:** External Trigger Events together with Trigger Interfaces are now supported. `Rte_Trigger` functions are generated according to the AUTOSAR standard.

- **CompuMethod data behavior changed:** CompuMethod data (scaling) for Application Data Types of category COM\_AXIS, CURVE, MAP, CUBOID, CUBE\_4, CUBE\_5 (and their axes) is now getting applied during test frame generation when the Record Layout definition matches with the used Implementation data types.
- **AUTOSAR platform connectable with Eclipse:** Debugging via Eclipse CDT is now possible with the AUTOSAR platform.

## C platform

- **Improved interface import**
- **Improved, easier, more flexible possibilities to schedule C functions.**
- **Stubbing of external functions and variables.**
- **Support of Bit-fields.**
- **C-data type size\_t is supported and automatically declared in TPT as unit32.**

## dSPACE HiL@FUSION

- **Startup/Shutdown Script support environment variables:** Startup/shutdown Scripts now supports environment variables. Additionally, dSPACE HiL@FUSION specific variables have been added.

## INCA

- **ETAS INCA interface extended:** TPT interfaces to INCA for calibration and measurement. Reprogramming and flashing of controllers is also possible.

## MATLAB/Simulink

- **MinGW compiler supported to generate DLL from Simulink model:** A FUSION DLL from Simulink models can now also be generated using the MinGW compiler.

## Test Execution

- **Group execution configurations:** Execution configurations can be grouped to form a tree of execution configurations.

## Test Assessment

- **Matrix assessment variables supported in Script assesslet:** Like other variables, matrix assessment variables can be declared in the Script assesslet, for example `f00= TPT.Int16MatrixX(3,4).`
- **New Trigger Rule assesslet option:** It can be specified that a Trigger Rule assesslet should get the result "failed" when the check box "Report as Failed if the condition always matches" was selected and no else-rule was checked.

- **Script assesslet editor with syntax highlighting:** The script editor highlights the code blocks of the current time interval.
- **Variable Definitions assesslet improved:** The graphical interface of the Variable Definitions assesslet has been improved. The variable definitions can be shown or hidden by clicking the [Show Operator Panel](#).
- **New assessment function in Script assesslet:** The function `TPT.getSignals()` returns all variables such as all channels, parameters, functions, assessment variables, and measurement variables.
- **TPT.hose() supports time-dependent time values and value tolerances:** In previous versions of TPT just the first tolerance value has been chosen and was considered constant for all subsequent samples afterwards. Now, the assessment function `TPT.hose()` supports time-dependent time values and value tolerance values. Note that this change might lead to behavioral changes of `TPT.hose()` if a time-dependent tolerance expression has been specified.
- **Signal Comparison assesslet behavior changed:** If a defined segment is compared with an undefined segment of another signal, the "difference to reference" signal no longer receives the value 0 but is undefined.

## Signal Viewer

- **Condition tree export improved:** In case several TPT projects are loaded and you export a condition tree from the Signal Viewer, you are now asked to which loaded TPT project you want to export the condition tree to.
- **Loading condition tree content optimized:** A drop-down list in the Signal Viewer lists all Condition Tree assesslets that are enabled for the selected test case. Select one of it to load the content to the Signal Viewer.

## Requirements

- **Access requirement attributes with Script assesslet:** With the new method `REQUIREMENTS.get("<id>")`, you can access requirement attributes and fields with the Script assesslet.
- **Requirements set with syntax highlighting and autocompletion:** The condition field of the Requirements set dialog has now syntax highlighting. Autocompletion is also available. Press `Alt+Enter` for a line break.
- **Requirement links can be imported:** When importing test cases from CSV or Excel files, requirement links can be imported from a column containing the requirement IDs separated by a comma. For this purpose, the entry "Requirement links" must be assigned to the corresponding column.
- **Hyperlinks can be imported:** When importing requirements from an Excel file, you can now import hyperlinks as clickable links.

- **New requirements import wizard:** The import wizard guides you now step by step through the requirements import.

## API

- **New function available:** The new function `tpt_vmapi_getIntTypeEnum` in the TPT-VM API is available to determine the corresponding type-enum value based on the byte size and the signed-flag.
- **Extended interface import options:** You can now import interfaces including server functions from TPT files, TPTPRJ, and TPTZ files.

## Report

- **New test report format:** The test report format "HTML (embedded resources)" is now available. Select this report format to reduce the number of created folders and files, and to add images, JS and CSS data used in the test report directly to the HTML files.
- **More report archive options:** You can specify which files should be included in the report archive.
- **Test case attributes in Report:** You can now add individual test case attributes to the [Test Case Summary](#) table of the report.
- **Synchronize button removed from Report view:** The synchronize button has been removed. Instead, you will always see the report of the selected test case in the Report view. When selecting a variant, the report of the previously selected test case is shown.

## Test Management

- **Test Case Status view renamed:** The Test Case Status view has been renamed to Status view.
- **Status view for test cases and assesslets:** TPT has a status model for the user-defined test design process. This status can be seen and changed in the Status view. The status can be checked for new revisions and changed for test cases and assesslets.
- **Synchronize button removed from Test Case Details view:** The synchronize button has been removed. Instead, the Test Case Details view always shows the content of the selected test case. When selecting a variant, the details of the previously selected test case are shown.

## General

- **Searching IDs:** Artefacts such as Testlets, Variants, Assesslets, Test Cases etc. have IDs in TPT. These IDs can now be found using local and global search of TPT.
- **TPT15 is based on Java version 11.**
- **TPT for 64-bit OS only:** The installation of TPT is for 64-bit Windows operating system only available. 32-bit installer is no longer available for TPT versions 15 and later.



# Release Notes TPT 14

## General

- Attributes and attribute values of an execution configuration can now be copied and pasted to another execution configuration.
- A new environment variable `${tpt.version}` has been introduced. It can be used to add the current TPT version to file paths, reports, data directories and so forth.
- Recently opened files are now listed in the file chooser when opening, saving, or importing files and can also be favored.
- In transitions of automaton or in step lists, expressions referring to an array variable with an explicit array index and an explicit point in time, for example `myarray[0] (t)` or `myarray[0] (t-@)`, do not produce a compile error anymore.
- Three new context menu items are available in the Project view:
  - **Execute Selected Test Cases**- Executes the selected test cases and test case groups with the last chosen execution configuration.
  - **Used Assesslets** - Displays all assesslets used by the selected variant or test case. of testlet variants and test cases.
  - **Used Report Assesslets** - Displays all report assesslets used by the selected variant or test case.

## Declarations

- The new Unit Editor lets you create and manage units.
- To use units in calculations, the **Physical Unit Support** incubation feature must be enabled.

## Test Set

- Test sets can now be restricted to test cases that are linked to at least one requirement of a selected requirement set.
- The new **Test Set** drop-down list in the Dashboard Properties view lets you execute individual test cases with the Dashboard while executing the other test cases without the Dashboard.

## Import / Export

- The **Name** column is now by default deactivated when you apply **External name** as synchronization method in the Import Interface dialog.

- When exporting a CSV file, signal names with whitespaces are automatically put in quotes, for example "my signal with whitespaces".

## Execution Environments

### ASCET

- The import interface option works now properly when called from the ASCET platform configuration in TPT.

### AUTOSAR

- The AUTOSAR configuration wizard has been replaced by a new platform configuration.
- The AUTOSAR platform considers "variation-point-proxy" objects for atomic software components and "variation-point" objects for ports, runnables, exclusive areas, PIMs, data-access definitions (implicit and explicit) for access to S/R port elements, Calprm data elements, and IRVs.
- New option to automatically generate the required AUTOSAR RTE contract phase headers.
- Arrays which array size is defined by a system constant can be resolved.
- An array-of-structs that contains arrays is now supported.

### C Platform

- The new option **Initialize interface variables with pointer types in C** in the **Test driver generation** section lets you generate code to allocate the required memory for uninitialized pointer variables that are connected to TPT.

### FUSION

- Based on ASAM XiL API, three new nodes are available for the co-simulation platform FUSION: CANoe node, VeriStand node, and XiL node.
- The Sound Emitter node has been enhanced:
  - The client functions for playing sound files are now automatically created.
  - MP3 audio files are now supported.
  - A new speak function reads aloud the string content added to it.
- Data exchange via XCP on CAN has been added; ECU-internal parameters can be read and written; ECU-internal channels can be read. To use XCP on CAN, enable the **XCP Support** incubation feature.
- TPT searches now automatically for a 64-bit version of the DLL of a Functional Mockup Unit if no 32-bit version can be found.
- FUSION platform: In case of a parallel test execution where a parameter exchange should only take place for the first test case, all other test cases now wait until the first test case has completed the parameter exchange.

- The "Prepare model for FUSION" option that is used to generate a custom FUSION node DLL from a MATLAB/Simulink model supports now the following use cases:
  - the TPT-S Function is placed inside a library
  - the TPT-S Function is placed inside a variant of a variant subsystems

## MATLAB

- Simulink models with internal signals (measurements) can now be tested with the FUSION platform.
- TPT supports the logging of TargetLink in referenced models and multiple referenced models. The logging is also supported in case of nested references inside a model-in-the-loop environment.
- Better performance when writing parameters with SLDD.

## XiL

- New test environment named XiL@FUSION has been introduced. It supports the MAPort and the EESPort. This test environment makes it possible to run XiL tests in co-simulation.
- Client functions are now automatically created when a XiL platform (dSPACE, CANoe, VeriStand, XiL) is set up. The client functions are used in a "Call function" step:
  - to start, pause and stop a simulation
  - to load parameter files
  - to play stimulation files (STI, STZ)

## Assessment

- New keyword `otherwise` has been introduced. It is used in the Script assesslet to run through all intervals that are not covered by the `during` block.
- A new assessment result has been added: `EXECUTION_ERROR`. This result is exclusively for assesslets that are stopped by a runtime error.
- An assesslet that is stopped by a runtime error does no longer stop the assessment process of the following assesslets.
- If a signal `S` is assigned an interval `x` that is not defined for all points in time in Script assesslets (using the syntax `S(t) := x`), the signal `S` is now undefined at all points in time where `x` is undefined. In other words, undefined ranges of `x` remain undefined in `S`. In previous versions of TPT the signal `S` was always defined after this assignment.
- Text is no longer cut off on the left side in the Script assesslet GUI.
- Trigger Rule assesslet: If one of the THEN/ELSE conditions of a trigger rule caused an execution error after another THEN/ELSE condition has been computed successfully (with result `PASSED`) then the execution error will be considered by the overall test case result

again.

- In the **Precondition** text field in the **Assesslet Content** view, line breaks are allowed again.

## Debug

- The debugging feature has been completely overhauled.
  - Additionally to breakpoints at testlets and in Script assesslets, breakpoints can now also be added in test step lists.
  - The new Debug Expressions view lets you add a list of expressions that are checked at every breakpoint.
  - To prevent unintentional debugging, the debug check box in the Execution Configuration has been replaced by the Debug button.
- The Debug view has been replaced by the Debug Breakpoints view. The view is used to manage breakpoints at test steps, testlets, and in Script assesslets, and to specify breakpoint conditions.

## API

- The new API Script Editor lets you write API scripts inside TPT.
- Changing the value of an array-of-struct with an expression containing channels or parameters that are changed during test execution is not allowed anymore.
- `Project.closeProject()` closes the TPT project without prompting any GUI dialog. Files referenced to the project must be closed manually.
- Fixed a bug that eventually could have caused an issue when using `Project.ImportIO()`.

## Report

- The file size of an HTML and MHTML report has been optimized.
- In the `test_summary.xml`, the ID instead of the UUID is given for assesslets.

# Release Notes TPT 13

## General

- The binding strength of exponent operator (\*\*) is now the same as in the Python language. Thus,  $a * b ** c$  is interpreted as  $a * (b ** c)$  instead of  $(a * b) ** c$ .
- Loading TPT files where a unit has been added to the parent of a signal of the type "struct", this unit will be discarded unless the parent unit is set in curly braces, and the children of the "struct" have no units on their own. If the signal is of the data type "curve" or "map" and the children have no units on their own, the parent unit is added to the "values" element.
- Higher performance of the Modifications view in showing changes and items affected by the changes even when a large number of item is altered simultaneously.
- Modifications View: you can specify to display only the modified object, the modified object and the objects directly linked to it, or the modified object plus the directly linked objects and indirectly linked objects.
- Test Case Details view got more functionality.
  - Instead of only "string" and "URI", you can specify an attribute as free text, check box, file, or enumeration.
  - URIs are automatically highlighted as hyperlink according to your settings in the TPT preferences.
  - The view can be locked to a test case, so the content stays visible when you select another test case in the project browser.
  - An unrestricted number of files can be added to a test case.
- The new Report view in TPT can display HTML reports inside TPT. The view can be locked to a specific test case.
- Test Set Definition view: you can now optionally create a test set based on the results of selected assesslet.
- When generating test cases or variants from value ranges, you can now either use the representatives based on the quantization, manually add representatives, or create representatives based on seven predefined calculations.

## Import interface

- By default, the "external name" is used for synchronization. You can change the synchronization method to "name" using the respective radio buttons.
- The action "Rename" replaces the "Rename (Name Only)" and "Rename And Copy Attributes" actions.

- Structs, enumeration types, Client and Server functions can be expanded.
- The target mapping can be selected from a drop-down list in the import interface table.
- When importing signals to TPT, you can now change the signal type of channels, parameters, constants, assessments, and measurements.
- Next to channels, parameters, constants, system constants, measurements, assessments, and custom data types, you can now show or hide functions.
- You can now specify in the project properties to append the signal source path (si\_tx\_path of cn\_si\_source) to the signal name when importing signals from MDF 4.
- Large MDF files up to a customizable threshold are read faster when importing an interface or when importing signals in a 64bit TPT version.
- In case a "Logging" mapping flavor exists, the logging will be disabled automatically in non-target mappings when a new signal is imported and the "Hide new signals in other mappings" option is selected
- If the imported signal is of the data type curve, map, or struct, the logging will now be disabled for all elements in case a "Logging" mapping flavor exists.

## API

- You can now configure and run back-to-back tests via the RMI API.
- Managing requirements via the RMI API is now possible. That is, you can among other things add descriptions, comments, and attachments, link requirements to assesslets and test cases using the RMI API functions.
- TPT VM API: New VM-API function `tpt_vmapi_bindSignalDynamic` for binding with dynamic access to the variable address.

## Test step list

- The test steps "Reset parameter" and "Reset all parameters" can now also reset the value of parameters that have the data type array, curve, or map.
- If the data type of the channel or parameter is an **int8-array** or an **uint8-array**, you can now enter a string value to the Channel step and to the Parameter step.
- Measurement signals can now be used in the Check column of a Table step.
- You can now join several "Wait for value" steps together so they are combined to a single complex condition that must be true to end a test step.

## Requirements

- When importing requirements, you can now filter the requirements by their status (new, changed, unchanged, deleted) and sort them by column.
- It is now possible to assign the result of a signal check to a requirement by using the Script assesslet. For example:

```
#check if mysig is always greater than 4 and
#assign the result to the requirement "SPEC-22"
```

```
REQUIREMENTS.checked("SPEC-22", TPT.assertAlways(mysig > 4));
```

## Report

- Concerning the PDF report, you can now select a font type from your operating system. As a result, also Asian characters can be displayed in the PDF report.
- Double-clicking on a Signal Comparison assesslet result graphic in the report, will open the graphic information split in two views in the Signal Viewer. One view shows the test signals and reference signals, the other shows the difference signal.
- Initial values can now be added to the report by using the Report Signal Table.

## PLATFORMS

- Each execution platform configuration in TPT has now a specific tab to specify definitions of system constants.
- TPT plugins for Eclipse CDT can now automatically be installed from within the FUSION platform, ASCET@FUSION platform, the C platform, and within the configuration of the FUSION GNU Debugger node.
- Silver platform: you can now use signal names with over 40 characters.

## ASCET

- As with the ASCET platform, you can now trigger the init-tasks in a ASCET@FUSION node at sample point 0s.

## ASSESSMENT

- To assess test case specific measurement files, it is now necessary to select a test case details attribute of the type "file" in the Assessment platform configuration.
- You can now link to a "Shared measurement file" by using environment variables, global variables, and test case specific variables.

## AUTOSAR

- AUTOSAR is now a regular TPT feature. To use the AUTOSAR platform, you need the autosar-plugin and the corresponding license.
- New AUTOSAR FUSION node to test AUTOSAR code in co-simulation environments.
- Different data type mappings for different software components are supported.
- You can set up "Write-Counter" signals that is extra channels to count SUT write operations.
- Connecting runnables as client functions is supported.
- Rte\_IsUpdated() in TPT output channels possible.
- New VM-API function "tpt\_vmapi\_isUpdated" is used.
- Activation reasons are supported.

## **CAN**

- The workflow for the configuration of a CAN connection has been redesigned to be more user-friendly.
- The Vector CAN node supports the use of CAN FD.

## **CANape**

- The import interface of the CANape FUSION node now offers a global device. Also, variables can be imported as TPT parameters. These are then added to the measurement and can be adjusted in the test case.

## **C code**

- The C-platform is now a standard TPT feature.
- Fields of pointer types are now supported if they get initialized after startup or if the address is changed during runtime
- Pointer fields within structs are now supported.
- You can now use GCov as code coverage tool.
- The C-platform has now an "I/O consistency check" option.
- When using CTC++ as code coverage tool, you can now parse the MON.sym file and add the listed entries either to the source file list or to the exclude file list.

## **dSPACEHIL@FUSION**

- The failure simulation now also supports RS232 and CAN as driver types.
- The binary file for the failure stimulation can be also an x86 file.
- Tests with duplicate external names are canceled.

## **EXE**

- You can now enable the system under test to read and write to channels specified as output channels in TPT by selecting the "EnableRead/Write for output" channels check box in the EXE platform configuration before you generate and compile the test driver.

## **FMI**

- For each FMI FUSION node, a node-specific mapping with rename mapping flavor can be selected



## FUSION

- The MCD3 Client node no longer exists. Use the INCA Client node or CANape Client node instead.
- New function "tpt\_fusion\_isUpdated()" has been added to "tpt\_fusion.h". It reports if any other node has written to a specific channel since the last call of "tpt\_fusion\_node\_callfcn" on the current node.
- For each test executed using the FUSION platform it is now checked if all of the channels/parameters declared in TPT are connected to at least one other FUSION node at runtime.

## MATLAB

- You can now enable the test frame generation option "Set data types explicitly for TPT outputs" so the data types are directly written to the Data Type Conversion (DTC) blocks.
- There is a new option to sort input and output channels for the TPT S-Function. New channels are automatically added at the end of the list.
- When importing the interface where a channel or parameter is named like a keyword in TPT, the name of the channel/parameter is extended by "\_rename". The original name is added to a rename mapping.
- New `${tpt.matlab.modelname.original}` variable to set a name for the original model.
- TPT can now read and write multidimensional arrays in Simulink parameter objects.

## PLS UDE, Lauterbach Trace32, GDB

- All exceptions can now be caught by an external debugger. This translates into better support for the debugging nodes in case of exceptions.
- You can now activate and deactivate breakpoints in the GDB node, Lauterbach node, and PLS UDE node.
- For a fast execution of the SUT via Lauterbachs Trace32, it is now possible to access values of variables/registers/memory locations in every TPT test step cycle without the need to define a breakpoint.
- Lauterbach node supports enumerations.
- Calculation of values index during write corrected.

## VeriStand

- Custom Devices are now delivered by TPT as LabView project. This way, you can compile and use the project with your VeriStand/LabView version.

## Signal Viewer

- The condition tree that is used to interactively design new assesslets based on the evaluation of test results is now a standard TPT feature.
- For a better overview, the results of an assesslet that has been run a check in several intervals are now displayed in a single Assesslet Result channel.
- When you open test results from within the report, the Signal Viewer shows now the path to the test result data in the title bar.
- Test results of several test cases can now be displayed either in a single tab or in different tabs.

## Assesslets

- The filtering has been improved. That is, the names of assesslet groups are only displayed if at least one assesslet in this group is enabled for the selected test case.

## Assessment

- New assessment function `usec()` to convert time values into microseconds without any loss.
- The function `TPT.getBit(value, bitindex)` that is used to check if a single bit of an integer value is set, is now also available in the TPT modeling language. It can be used, for example, in step lists.
- New assessment function `interval.getTimeAtSample(int)` to return the time of the sample with index of the signal.
- You can now specify a tolerance directly to the equals-with-tolerance operator (`=~=`).

# Release Notes TPT 12

## General

- Step list: "No Button" option in the Message box test step to close the message box only when the termination condition becomes true.
- Build progress: It's possible to reclassify test results of several test cases at once.
- Build Progress: Test results and execution results can be exported to an MDF4 file directly from the Build Progress.
- Modifications view: New **Hide Affected Items** filter option in the modifications view to see a simple list of modified items.
- Test Set: New button to activate or deactivate the automatic include of new test cases to Test Sets.
- Copy / Paste is now available for Test Case Attributes in the Test Case Details View.
- Project view: New option to show test case count and variant count for each folder in the project view.
- Test Cases Import, Raw Data Assignment: new option to decide whether to create test cases as time partition or as step list.
- The TPT menu item **Tools | Copy Outputs to Local Proxies** now also works for step list test cases.
- TPT RMI API: `equals()` and `hashCode()` supported for all API objects.
- Batch script: New command `--testSet "test_set_name"` to use a specific test set via a batch script.
- The semantics of `TPT.length()` in "Wait" steps in conjunction with "Import signal" steps has changed, but only when the following conditions are given:
  - An "Import signal" step exists with at least two imported signals which have different end times.
  - A subsequent "Wait" step exists that contains the expression `TPT.length(X)`. Consider "X" to be a channel used in the "Import signal" step, which assigned signal has not the shortest end time (is not the shortest signal found in the "Import signal" step).

The semantic change is as follows:

- Until TPT11: `TPT.length(X)` had been resolved to the shortest overall end time (shortest signal) found in the "Import signal" step.
- With TPT12: `TPT.length(X)` is now resolved as the end time of the signal assigned to the channel "X".

This semantic change has been introduced to ensure a consistent and intuitive behavior of `TPT.length()` in all cases.

- Units cannot be set for structs, curves and maps. When loading old files that had units defined in structs, curves or maps, these units will be passed on just to those elements that are one level down, provided they are not structs, curves or maps and do not have their own unit (if any existing, it will be kept). If the type of a signal is set to struct, curve or map, an existing unit is deleted.
- Fixed axes of maps and curves are supported when importing the interface from MDF files.

## Declaration Editor

- The types of struct elements are now displayed in the Declaration Editor and can be changed for custom types.
- Interface import from FMI model description files (\*.xml)
- Import/ Export: Importing interface from an XLSX file: you can decide how to order the columns to be imported. Columns with unreadable names are ignored.
- Type Editor: "Select unused" button to highlight those custom data types that are not used in any channel, parameter, constraint, assessment, measurement, or function, and that are not referred to by other types.
- You can now edit integer values bit by bit using the value editor.
- The currently selected mapping is used during the interface import. In case no mapping is selected in the Declaration Editor, the last mapping used is automatically selected.
- New shortcut to open the Import Interface wizard (Ctrl + I). The Export Interface wizard can be opened from within the Declaration Editor by using Ctrl + O.
- For int8, int16, int32, and int64 data types, negative values can be represented in complementary notation in HEX- or BINARY-format in a step list.

## Assesslets

- A comment can be added to every definition interval of a signal using the function `signal.setComment()`.
- Signal comparison assesslet and back-to-back settings:
  - New option to "Ignore undefined time phases" in both reference and observed signals. This way, signals are not compared in those contexts where signals are not defined.
  - Float precision check box: select this check box to compare the observed and reference signals with float precision instead of double precision.
- Behavior of `TPT.checkAlways` changed: For `TPT.checkAlways (EXPR, "text1", "text2")`, if the `EXPR` is "undefined" for all samples in the current context interval, this is now considered to be an error and the result signal is accordingly set to "false" in the entire context interval.

## Assessment

- For signals that do not start at  $t == 0s$ , the function `TPT.length()` now always returns the duration from 0s to the signal end.
- New assessment functions for Script assesslet:
  - **TPT.assertAlways**: checks if the time dependent expression `expr` of type `boolean` is `true` for all points in time of the current context interval and generates a report entry.
  - **TPT.assertExists**: checks if the time dependent expression `expr` of type `boolean` is `true` for at least one point in time of the current context interval and generates a report entry.
  - **TPT.assertTrue**: checks if the condition `cond` is true and generates a report entry.
  - **TPT.assertFalse**: checks if the condition `cond` is false and generates a report entry.
- New "Save" button to synchronize the condition tree inside the Signal Viewer with the already created condition tree assesslet.

## Requirements

- Requirements marked in TPT as removed are displayed in the report with struck through ID and struck through object text.
- Auto review check box in Raw Data Assignment. Changes in requirement attributes during a requirements import can be automatically reviewed during import to prevent:
  - Your requirements from being marked as changed.
  - All linked elements from being marked as to be reviewed in TPT.
- Comments added to requirements in TPT can be displayed in the report.
- Requirement sets can now be created also based on the content of the TPT internal comment field.
- Requirement sets can now be dynamically generated based on conditions given in requirements attributes, using the new `REQUIREMENT.evalAttribute(String attributeName)` function.
- Requirements Import, Raw Data Assignment: In case no module information is found, you can manually assign a module name using the **Module** field.
- You can import OLE objects from DOORS as attachments (images) to TPT.

## TASMO

- Instead of entering the range information for signals manually, you can now import the range data from an equivalence class, a min/max and scaling flavor, and in case of TASMO for Simulink also from a Simulink model.
- Automatic import of min/max values from TargetLink Data Dictionary as default values.
- Support for matrix-signals as model inputs.
- Model reference blocks are now supported by TASMO.

- Regarding the coverage analysis by TASMO for C, all data types used inside a condition are supported.
- In some cases TASMO did generate step lists that did not match the reported coverage. This issue has been fixed.

## Jenkins

- Improved Jenkins plug-in:
  - You can now select test sets for the execution.
  - The TPT HTML report can be opened in Jenkins.
  - A TPT test results trend graph is shown in Jenkins.

## Dashboard

- New Dashboard String Display widget to display the value of string channels.

## Signal Viewer

- Signal Viewer preferences can be loaded together with test data by using the command line.
- Constants and System Constants are visible in the Signal Viewer.

## PLATFORMS

### FUSION Platform

- New CANape connection with new features:
  - Measurement
  - Calibration
  - Diagnostic request, read / write requests
  - Restart measurement on error
  - Reset calibrations after test run
  - Minimize number of online signals
  - Exclude unused online signals
  - Write SNA values for signals that are not available at the specified device
- New FUSION node to connect TPT with Lauterbach Trace32 software.
  - Support of Lauterbach commands. You can also configure a number of breakpoints, without being listed in the program for each program step, in the advanced options tab.
  - Support of PiL testing via Lauterbach.
- GNU Debugger node is not an incubation feature anymore and it is now available for all users.
- PLS UDE node:
  - New dockable GUI.
  - Specifying breakpoints using long description of location is now possible.
  - Search field now available.
  - Breakpoints can be ordered in the breakpoints table.
  - Changes made through the GUI take effect immediately (OK and Cancel buttons were removed).
  - New option to display a message during the test execution whenever UDE stops at a breakpoint that is not specified in TPT.
- FMUs for co-simulation are supported as FUSION nodes.
- "New Test Execution - Co-Simulation with FUSION" example available in TPT.
- New coverage tab in the INCA Client node to define the INCA variable and its raster.

## C Platform

- Enums are supported.
- You can now start a debug session in Eclipse CDT from within TPT during the test execution. TPT can automatically create an Eclipse project for the C platform.
- Apart from the data type Boolean, TPT can now also handle variables of the data type `_Bool`.
- The extracted interface is automatically bound to the SUT by TPT.
- Function Kind: You can select if a function should be executed directly after starting the executable (Startup), at the beginning of the test execution (Initial), or repeatedly during the test execution (Periodic).
- "Enable Read / Write for output channels" option enables read / write functionality for TPT output channels when you generate and compile the test driver.
- C-Platform supports CTC++ code coverage. Coverage can be enabled, and metrics can be selected.
- Scalar pointer variables like `int *p` are now properly connected.

## EXE Platform

- New CTC++ options, metrics can be selected.

## MATLAB Platform

- Support for writing string array parameters.
- Import Code coverage with CTC++ and Targetlink Version 4.3
- Code coverage with MATLAB 2017b supported through "Simulink Coverage" MATLAB toolbox.
- The "Prepare model for fusion" button now makes MATLAB to use a matching ert target if the original model also uses an ert target (for example: `ert.tlc`).
- Generation of a custom FUSION dll node now supports MATLAB from version 2016b onwards.

## VeriStand Platform

- Parameters can be imported from a SDF file and manipulated in TPT.

## dSPACE HIL Platform

- Import interfaces via the XiL-API.
- Running tests in real-time via the XiL-API.



### **dSPACE HiL@FUSION Platform**

- The dSPACE HiL@FUSION platform can now run test cases with both dSPACE 32 and 64 bit installations.

### **Silver Platform**

- New example for the Silver platform to run the lights control model.

### **ASCET@FUSION Platform**

- You can now start a debug session in Eclipse CDT from within TPT during the test execution.

### **AUTOSAR Platform**

- Support of Rte\_feedback functions for sender ports with acknowledgement request.
- New option to add extra "include" files outside an AUTOSAR project using the AUTOSAR code generator wizard.
- In case of S/R ports: channels are only created if any access to the corresponding AUTOSAR element is declared.
- New CTC++ options, metrics can be selected.
- Support of per instance parameters.
- Multiple instantiation of Atomics are now supported.
- Ports that are sender and receiver at the same time are now supported.
- Fault injection feature to inject faults (manipulate data) between two ports.
- Fault injection also with S/R ports using queued communication.
- TPT supports now explicit reading via return value (Rte\_DRead). Previously, only explicit reading by argument (Rte\_Read) was supported.
- If the generation of the CDS component has been suppressed during the generation of the C code, TPT no longer uses Rte\_DE and Rte\_DES types.
- You can now choose a Visual Studio compiler from a list of available compilers when specifying the settings for generated SWC C code wrappers.

# Release Notes TPT 11

## General

- A transition can now refer to a another transition.
- A new view has been added, called Modifications: this view is used to show changes and the items affected by these changes in the project and also the differences between two TPT files.
- The default or custom perspective can now be selected from the toolbar. The last two used perspectives are displayed as buttons beside the default perspective icon.
- All open TPT projects can now be saved by clicking a "Save all" button.
- Auto-cast feature. TPT will now automatically cast the expression of type X to type Y if type Y is a superset of the value range of X and is defined for primitive types only.
- New functions to convert a int64 value to the corresponding IEEE754 64bit floating point equivalent (ieee754double), and an int32 value to the corresponding IEEE754 32bit floating point equivalent (ieee754float).
- The plus sign/operator (+) now concatenates two strings.
- New cast operator (string)num to convert numbers into strings.
- Information about the transition usage has been added to the Specification of Transition to see in how many test cases a transition variant is at least used once.
- The parameter consistency check inside a TPT model is now more strictly. Detected inconsistencies are immediately shown in the TPT project browser as well as in the test step list.
- Windows XP is no longer supported.
- The Step List testlet and the Time Partition testlet have been merged into the Local Content testlet. As a result
  - A single state can be implemented by Step List variants AND by Time Partition variants.
  - A test case can be set up as a Time Partition diagram / automaton or as variant-independent test step list.
- Alternative paths are no longer used in Library testlets when storing a TPT file to prevent unnecessary changes when working with a versioning tool.
- FLEXlm License Borrow option has been improved.
- TPT prevents the creation of circular references when reference testlets are copied and pasted.
- Initial values that were set in the Parameter view are no longer overwritten by the default value when the "Select all" button in the Parameter view is clicked.
- Byte order mixture is allowed except when a HEX file is loaded together with an A2L file.
- Bug fix concerning multicore use with the Jenkins plugin.

## Declaration Editor

- New refactoring feature in the Declaration Editor to rename signals in the TPT model. Any occurrences of the signal name in comments or / and strings, remain unchanged.
- Measurement variables can be now also be of the data type "struct".
- The channel mode "local" and also the information if a constant is set to "fixed" or "system constant" is now exported and imported as expected.

## Import and Export

- New wizard to import and export test cases and requirements:
  - Automatic file format recognition (CSV, Excel)
  - More robust, intuitive interface
  - New explicit import type assignment for imported values to determine what is a test case or what is a group (instead of filtering with regular expression). This applies for the test case import. For the requirement import will be determined what is a requirement, heading or information.
  - Test cases can be exported based on an existing test set.
- The modification dialog can be accessed from the Requirements view via the toolbar or by pressing the key F3.
- Settings concerning the requirements import and the test case import / export are now saved within the TPT project file.
- The two pie charts of the requirements section in the report have been merged into a single pie chart. The single pie chart shows now if requirements have been "Passed (with issues n)", "Failed (with issues n)", or were "Not Covered (with issues n)". The issues are displayed radius-proportionally in a darker color shading.
- When importing signal data from an A2L file and the interface from a DCM file, the maximal dimension of array based parameters can now be taken from the A2L file instead of the DCM file.
- New option to import only undefined parameters in the Parameter view.
- When importing test cases with URIs from Excel files, the URIs must be written in plain text. TPT does not read Excel links or hyperlink formulas anymore.
- Settings concerning the test case import and export are now saved within the TPT project.

## Test Step List

- Import Signal step, signals of the data type "struct " can be imported.
- With the Ramp step, signals of the data type "struct" can be ramped.
- The syntax `a ** b` can be used inside step lists additionally to the syntax `pow(a,b)`.
- New option in the preferences to allow the auto-cast of floating point values to integral data types for channels and parameters in Embedded Signal steps and Import Signal steps.
- In the check columns of the Table step, Python's math functions can be used.

- MATLAB functions can be called by the Call Function step.
- C-code functions that are triggered by a Call Function step can read values that have just been written to the TPT-VM and also write values to the TPT-VM that can be instantaneously read by TPT.
- Alternative paths are no longer used in Import Signal steps when storing a TPT file. This prevents unnecessary changes when working with versioning tools.
- Bug fix concerning the undo (Ctrl+Z) and redo (Ctrl+Y) in the Table step.
- Bug fix concerning multiple selection in the step list.

## Automatic Test Data Generation

- New option to generate individual variants (single probes) for each permutation while generating test cases from value ranges.
- The enumeration constants of a signal are also used when automatically generating test cases from value ranges.
- Test cases from value ranges can be inserted as Channel step or as Embedded step.

## Platforms

- New execution platform named C platform that can:
  - Test C-code.
  - Import a C-code interface.
  - Automatically generate the test driver.
  - Generate test data for testing C-code with TASMO for C.
- New option in the Platform Configuration (Code Coverage) of the ASCET and FUSION platform to exclude files when using CTC++ as code coverage tool.

## AUTOSAR platform

- Changed execution order of the runnables is saved with the TPT file.
- Queued receiver communication is supported. The return value of the functions RTE\_receive() and the RTE\_send() can be defined in TPT.
- Problems with the queued communication between two AUTOSAR software components have been fixed.

## dSPACE HiL platform

- Configuration of failures in the Call Function step of TPT's test step list via a graphical interface. The failures are triggered in the time context of the test step list when used with the dSPACE@FUSION execution platform.

## **C Platform**

- Option to select individual source files for the interface analysis and instrumentation.
- The compile commands are written to a batch file that can be manually adapted.
- Each function can be individually set to run only once at the start of a test case or periodically.

## **EXE platform**

- Option in the Platform Configuration to include the I/O consistency check into test driver.
- The new function `tpt_vmap_i_bindSignalFinalize()`, directly placed after the `tpt_vmap_i_bindSignal()` function, checks if there are `tpt_vmap_i_bindSignal()` calls for each input and output channel in the TPT test model.
- The generation of the `tpt_vmap_i_bindSignal()` function is suppressed if a selected mapping has a hidden flag for input or output signals/parameters.

## **FUSION platform**

- New option in the INCA Client Node to define whether TPT should try to re-connect to hardware or not after a failed initial connection.

## **ASCET@FUSION platform**

- New option to compile the DLL with debug information for the GNU debugger.
- ASCET processes can now be called directly from TPT test cases, that is independently from the scheduler.
- The highest common factor of all periodical processes is used automatically to determine the step size.

## **MATLAB platform**

- MATLAB functions can be called by the Call Function step in TPT's test step list.
- Simulink bus signals are now supported by TPT internal signals and can be imported as measurements.
- New option in the Platform Configuration to use the actual names that are available to the bus selector instead of the names provided by the Simulink.Bus object during the test frame generation.
- The TPT S-function supports the usage of symbolic dimensions.
- The test frame can be manipulated after its generation by using a custom script that is set up in the Platform Configuration.
- Concerning the test frame generation from Simulink libraries, the solver is now per default set to "fixed discrete".
- When using a version before MATLAB 2009b, parameters from Targetlink models are now imported correctly.

## **CANoe Platform**

- CANoe platform is not an incubation feature anymore but a full licensed feature.
  - COM-interface importer improvements.
  - Stability and performance improvements

## **PLS UDE node**

- Core index of the ALWAYS breakpoint is editable.
- A source code directory can be chosen.
- New option to write only those variables (parameters/channels) that are used within the current test and assessments via Fusion to UDE.
- In breakless mode, the debugger will be started once after the target is reset.
- The path from the TPT HTML report to the UDE coverage report is now up-to-date.

## **TASMO for C**

- TASMO for C (acronym for "Testing via Automated Search for Models for C code") generates test data for the testing of C code. It uses the C platform.
- You can provide C-code input details to generate valid and meaningful test cases. Signal specification can be set via signal files or manually.
- You can set the coverage criteria (decision/condition coverage) and select the coverage goals. It's possible to filter coverage goals by selecting specific functions, statement types (if- statement, ?-operator, switch, for-loop, while-loop) and coverage criteria.

## TASMO

- A set of signals can be assigned to certain TPT channels or parameters. This assignment helps TASMO to build test cases for each coverage model state that might require a complex combination of signal values. At step list generation, TASMO will create an Import Signal step.
- TASMO supports the data port order "Specify indices" for multiport-switch blocks.
- The block paths in TASMO can be filtered with or without using regular expressions.
- Multiple selection of coverage goals is supported.
- Coverage goals for individual subsystems as well as single expressions of a subsystem can be manually canceled before running the test data generation.
- Better performance regarding the selection and deselection of coverage goals.
- Several signal files can be imported during the test data specification.
- Export of TASMO input and/or the coverage overview to CSV.
- Bug fix at model import process when the "Comment Through" option was used in a Simulink block.
- The Stateflow debug mode is now deactivated during instrumentation when one uses any MATLAB version older than MATLAB 2011b to prevent the process from being paused.

## Requirements

- New requirement set definitions to define requirement subsets. These subsets can be used in the test execution to restrict the coverage to the subset. The requirement set for the report is chosen in the Advanced Report Options of the Execution Configuration.
- Requirements can be linked with test case groups and assesslet groups by a drag-and-drop operation.
- Besides the linked object's name, also its path and/or ID can be displayed.
- The status of unlinked requirements can be manually set to "New".
- Directly linked requirements of a test case can be shown in the Test Case Details view.
- The Coverage Statistics can now be opened from the toolbar.
- Delete or create requirement links to selected test cases.
- TPT internal comments can be added to individual requirements.
- When using requirements in Script assesslets with the REQUIREMENTS.checked() function, autocompletion is supported. The autocompletion shows besides the ID of the requirement also its object text as tooltip.
- The order of requirement attributes can be manually changed.
- The requirements coverage statistics shows also the assesslet coverage.
- When importing requirements with URIs from Excel files, the URIs must be written in plain text. TPT does not read Excel links or HYPERLINK functions anymore.
- An optional table called "Requirements Assesslet Results" in the report shows the result of the requirements checked by assesslets for each execution platform.

- In the Preferences for the local file, you can specify to show additional issue tables in the report if requirements are:
  - Not linked to executed test cases.
  - Not checked by an assesslet during test execution.
  - Unchecked by an assesslet while executing test cases they are linked to.
  - Checked by an assesslet while executing test cases they are not linked to.
  - Not checked by linked assesslets.
  - Checked by unlinked assesslets.
- Text objects (headings and information objects) can be hidden in the report.
- The modification dialog can be accessed from the Requirements view via the toolbar (shortcut key F3).
- Modifications concerning requirements are now shown in the Modifications view.
- The "Needs to be reviewed" flag is replaced by the "modified" flag. The flag is controlled in the Modifications view.
- Settings concerning the requirements import are now saved within the TPT project file.
- The two pie charts of the requirements section in the report have been merged into a single pie chart. The single pie chart shows now if requirements have been failed, inconclusive, not covered or passed.

## Assesslets

- Python math module is automatically loaded for assessments so its functions can be used to assess tests.
- It is now allowed to call the function `TPT.detectTimeShift()` in time contexts. A cache is used to avoid redundant calculation for each time step.
- Parameters can now also be selected from the a list; constants can no longer be added to the table of variables to be compared in the Min/Max Comparison assesslet and in the Signal Comparison assesslet.
- Complete arrays can be compared in assesslets element by element.
- The method `TPT.readMeasurementRecord()` has again only two arguments. A third argument is taken automatically from the TPT preferences (TPT Model Behavior | Assessment) and specifies whether `name_[d]`-signals should be joined as an array or not.
- More information is added to the report when signals are compared for example in the Signal Comparison assesslet. The table lists the start and end time as well as the duration and maximum deviation from the tolerance band.

## Min/Max Comparison assesslet

- Result channels (`range_check`, `min_value`, `max_value`) can be exported individually.
- Constant variables are no longer shown in the autocompletion in the Min/Max Comparison assesslet.



### **Report Signal Graphic assesslet**

- The Report Signal Graphic assesslet can also handle arrays and structs.

### **Report Signal Table assesslet**

- When selecting the check box "Show as trace table" in the Report Signal Table assesslet, the generated table shows 32 entries also when the amount of entries exceeds 32

### **Script assesslet**

- Script assesslets can be used as libraries to insert functions, classes, and objects defined in this assesslet via autocompletion in the following / next assesslets.
- Bug fix concerning the assignment operator (":=") in the Script assesslet. Logical operations concerning the variable on the left work now properly.

### **Signal Comparison assesslet**

- Signal comparison assesslet: new "Find in reference file" option to specify a regular expression pattern to match TPT channels to signals in the reference file.
- When using the Signal Comparison assesslet, also the relative tolerance is now graphically shown in the report.
- Constant variables are no longer shown in the autocompletion in the Signal Comparison assesslet.

### **Trigger Rule assesslet**

- Multiple THEN - ELSE conditions can be specified in the Trigger Rule assesslet. All conditions have to be fulfilled to pass the assesslet.

## **Execution Configuration**

- The test execution can now be terminated after a given number of failed tests.
- You can switch from simple to advanced mode to specify the execution items. You can do the same with both modes. But only in the advanced mode you can set up several execution configuration items for a single execution configuration and activate / deactivate the assessment for single execution configuration items.

## **Back-to-Back Test**

- Back-to-back tests can be configured in the Execution Configuration.
- Parameter sets can be selected for Back-to-back testing.
- The compare mode in the Back-to-back settings lets you specify if the signals must be of

the same length (strict) or if differences in signal length are tolerable (normal).

- When copying an execution configuration, the Back-to-Back configuration is copied too.

## Build Progress

- The toolbar of the Build Progress is undockable.
- Bug fix concerning the behavior of the pause button in the Build Progress Dialog.

## Signal Viewer

- The Signal Viewer is now as a standalone application available with its own documentation.
- Up to five quick preferences can be saved for a single project and applied by shortcut or button to the currently presented signal data.
- Test case data or assesslet data can be placed in the Signal Viewer by a drag-and-drop operation.
- Besides filtering of signals in the Signal Viewer, it is now possible to search for signals in plain text or by using regular expressions.
- The graphical representation of a signal in the Signal Viewer is highlighted when the signal is selected in the list of signals. New shortcut keys for this.
- Double-click on a tab name to open the respective test case in the TPT Project Browser.

## Report

- It is now possible to generate a separate report that contains an overview of all assessment variables/assesslets and their usage in test cases.
- Test Case Attributes are no longer chosen in the Advanced Report Settings of the Execution Configuration but in the Meta Info Report Table assesslet.

## Dashboard

- New methods to change panels by condition and track panel changes by using the Dashboard script.
- Deadlock in Bits Widget in Dashboard script fixed.

## Remote API

- New classes and methods have been added to the Remote-API, for example, to create assessment variables and to add mappings and mapping flavors.
- Bug fix concerning issue with relative paths.

## Search and Replace

- Parameters whose values have been defined in the Parameter view are now found by the local and global search.
- Local search is now also supported in the Debug view and the Functions view.
- Text fields in the Advanced Report Settings are now searchable.
- The global search now finds also entries in custom scripts that are set up in the Platform Configuration.
- Bug fix concerning string replacement.
- Bug fix concerning the search and filtering in the Declaration Editor.

## Documentation

- In the HTML5 help, a client feedback button has been added so you can easily submit a feedback to the documentation team.
- Documentation updated describing all TPT-VM-API functions for the EXE platform.
- New Equivalence Classes example and documentation.
- New "Test Modeling - Automaton and Step Lists" example and documentation.

# Release Notes TPT 10

## Declaration Editor

- The Declaration Editor has been completely redesigned.
- Direct display and edit of mapping flavors (scaling, rename, ...).
- Direct manage and edit of equivalence classes.
- Set interface roles directly (IN/OUT/LOCAL).
- Multi-editing supported.
- You can switch between the decimal, binary, and hexadecimal representation of an integer.
- Global undo (CTRL+Z) and redo (CTRL+Y) affects also actions made in the Declaration Editor.
- You can show and hide columns in the Declaration Editor.
- New shortcuts to create channels (CTRL+1), parameters (CTRL+2), constants (CTRL+3), measurements (CTRL+4), and assessment variables (CTRL+5).
- The single elements of a struct are now displayed as an expandable list in the Declaration Editor.
- Display of a struct in the Declaration Editor
- The values of string channels are now displayed.
- Min and max values for a struct or an array can also be scalar values that apply to all elements of the struct or array.
- Initial values can be set for channels by using an Init Values mapping flavor and individual test cases by using the Initial Values view.

## Equivalence Classes

- New Equivalence Classes Editor.
- Direct editing of all equivalence class sets in the Declaration Editor.
- Equivalence classes can be accessed in assessment scripts.
- Equivalence classes set in an assesslet with auto-completion.
- You can automatically generate test cases from subsets of Equivalence Classes Sets.
- Equivalence classes can be defined as mandatory or forbidden in the Equivalence Classes assesslets.

## Mapping Editor

- The Mapping Editor has been redesigned.

## Import Interface

- Instead of remove, you can hide a signal during the interface import if this signal is declared in TPT but cannot be found in the external interface.

## Import and Export

- TPT can import array elements from MDF files as array. The import can be configured in the Preferences dialog, TPT Model Behavior section.
- Comment information from MDF3 is imported as string.
- MDF4 event blocks are imported as string variables. These string variables are also displayed in the Test Data Viewer.

## Test step list

- New Table step.
- Each step in the step list has its own documentation field.
- The Ramp channel step can now ramp parameters.
- Inside the step list, the set If, Else, While, and the Parallel steps can now be expanded and collapsed.
- Test cases can be generated from value ranges.
- You can use mapping information of test platforms by typing the channel name and flavor and set a ->. For example: light\_intensity > light\_intensity->Min.

## Platforms

- You can now add custom Python scripts to the platform configuration. The scripts run either before or after the execution of the platform / test cases.

## ASCET

- You can now extract the SUT interface, generate the test environment, and configure ASCET via the RMI API.

## **ASCET@FUSION**

- New option "Limit channels and parameters" in the platform configuration lets you limit the physical values of channels and parameters to the bounds of a Min/Max flavor.
- Parameter are read back from the SUT. That is, in case parameter values are limited or changed in the SUT, TPT reads the limited value.
- Support of quantized physical experiment.
- Support of different implementations.
- You can set individual task-counter names.
- The order of tasks/processes is no longer limited.
- You can specify "round" or "cut" for calculations of internal values (scaling).
- TPT 64bit can now communicate with ASCET.

## **VERISTAND**

- TPT checks whether the signals from a TPT project file are present on the HiL. The test cases are executed only when all signals given in the TPT project file can be found on the HiL.

## **FUSION**

- New section in the FUSION platform configuration, named "Advanced Options", to specify the runtime and the debug options.
- Parameters in FUSION have now read-write semantic, so the values of parameters can be read back from the SUT via FUSION and become a part of the test data.
- 64bit DLLs can be used.
- Rename mapping flavor supported.
- The name of the MS.NET Assembly Node has been changed to Custom Node.NET DLL.

## **MCD3 client node**

- You can now specify how many times TPT should try to connect to the controller if the initial connection breaks down, and how many seconds take between each attempt to reconnect.

## **INCA node**

- The initialization process for the INCA Node has been optimized.
- You can suppress the initialization of INCA if INCA has already been started and configured and a test has to be executed several times.
- The INCA node supports array signals and measurements.
- INCA array measurements are now also array channels at TPT.

## **PLS UDE node**

- You can now read from bit fields or write to bit fields.
- You can specify to open a given Executable and Linking Format file (ELF) automatically when the UDE is opened.
- Breakpoint timeout in the node configuration.

## **LABCAR**

- A new platform has been implemented, called "LABCAR Platform". It runs TPT test cases via the FUSION and exchanges the signal data with the LABCAR OPERATOR (LCO). The measurement and calibration is done via INCA.

## **dSPACE HiL@FUSION**

- A new platform has been implemented, called "dSPACE HiL@FUSION". It runs TPT test cases via the FUSION in non-realtime on a dSPACE HiL environment. Communication is performed using the ASAM XiL-API.

## **MATLAB**

- Better support of TargetLink Data Dictionaries.
- TargetLink subsystems built from a library are supported by TPT for automatic test frame generation.
- TPT environment variables are now written to the MATLAB workspace.
- In conjunction with the MATLAB platform, the MinGW 64 bit is supported in TPT.
- TPT can handle the Simulink.LookupTable that holds explicit values and the Simulink.Breakpoint that represents an axis of a Simulink.LookupTable.
- TPT can connect to resettable subsystems in MATLAB to analyze the subsystem's interface and to generate the test frame.
- Instead of memory blocks in the test frame, unit delay blocks ( $1/z$ ) are generated.
- Parameters in referenced Simulink models are taken into account by TPT during the import.
- Coverage with V&V is always cumulative.
- Coverage measurement is possible for referenced models.

## **TASMO**

- TASMO is more robust and faster, and supports more Simulink blocks for coverage goals.
- TASMO supports decision coverage for the following blocks: Sign (Simulink, TargetLink), Relay (Simulink, TargetLink), and Dead Zone (Simulink).
- TASMO supports decision and condition coverage for Stateflow Truth Tables.

- You can filter subsystems by block type and coverage criteria in the "Select Coverage Criteria / Goals" dialog section.
- New signal characteristic named "Constant" has been added.
- TASMO performs a static analysis of the test model before the test data generation to detect non-reachable coverage goals.
- TASMO can analyze existing test cases and achieved coverage during the normal test execution to avoid redundant test data and to increase the coverage goals.
- The coverage information of the test cases generated by TASMO can be exported from TASMO into a CSV file.
- TASMO supports parameter changes in the test data generation.
- TASMO can generate test data as linear step lists or as parallel step lists.

## Requirements

- TPT10 supports DOORS 9.6.
- You can deactivate in the Advanced Report option in TPT, that a requirement which is not linked to any assesslet, derives its result from a test case. When this option is deactivated, the requirement is set to "not covered" instead of "inconclusive".
- TPT can handle the URI Object attribute from DOORS. URI Objects are links.
- For better visibility, the position of the number located in variant and test case icons that shows how many requirements are linked to that specific test case or variant has moved upward.
- The requirements result of several execution platforms are displayed in a single report table; the pie charts show the results of the different execution platforms combined.
- The "Report Linked Requirement" displays more information, for example about the kind of link (direct, inherited).
- Report shows if the link to a requirement is direct or inherited.

## Assessment

- The Jython version has been migrated to Jython 2.7, thus TPT can now load DLLs at assessment time.
- The Assessment Library is now a regular feature and has been moved to the Preferences dialog.
- The Assessment Library has file extension \*.tptpy.
- Specified functions can be used in assessments.
- New dialog to select variables in assesslets (Min/Max, Signal Comparison, Equivalence Classes).
- You can increase the context interval by positive arguments or reduce them by negative arguments with the new assessment function TPT.extendContextRel (before, after).
- In STRICT mode, undefined inputs lead to undefined results.



- The new option in assesslets tree "Copy the Structure of the Selected Test Cases" creates assesslet groups that are structured with regard to the test case groups. The corresponding linked assesslets are sorted accordingly.
- New operator "=~=" that compares two signals with respect to a tolerance.
- The semantics of the getTolerance() function also considers that the ScalingMode can be OFF.

### **Condition Tree assesslet**

- You can disable and enable single checks by selecting the "Enabled" check box.

### **Trigger Rule assesslet**

- The trigger condition can trigger an "Else check" for the intervals in which "Then check" is not true.

### **Min/Max Comparison assesslet**

- You can now select a mapping with Min/Max flavor in the section "Mapping for bound information".

### **Script assesslet**

- Locally declared functions and variables can be entered via auto-completion.
- Already declared elements are highlighted when you hover with the mouse over the script and at the same time press the CTRL key. When you click on a highlighted element, the Declaration Editor opens and directly shows the selected element, so you can immediately edit it.

### **Import Measurements assesslet**

- The check box "Use mapping from platform" has been removed. Go to the "Import" section, click and select "Use mapping from platform" from the mapping list.

### **Report Signal graphic assesslet**

- You can now filter signals that differ from default value to include them in the Report Signal graphic.

### **Build Progress dialog**

- The Build Progress dialog has been redesigned.
- The execution details tree shows also which assesslet in what time context has been erroneous and at which Compare step something went wrong.
- You can directly jump to the assesslet in the Assesslet view or in the Assesslet Content view by clicking on it in the Build Progress.

- New toolbar icons added to cancel the current execution, to create and to open the overview report, to automatically generate an overview report after every test case is run.

## Test Data Viewer

- You can load several test or measurement data into an independent Test Data Viewer tab or window.

## Report

- The section "Test Case Status Summary" in the report table shows now the test case status that you set.
- Report shows information, which ASCET@FUSION project has been used for the generation of the DLL.
- PDF reports are now generated in landscape format for better table display.
- New variable `#{tpt.scenario.inheritedcomment}` to add the descriptions that were set to test case groups or test cases to the report.
- You can specify for which files the coverage report should be generated using CTC++.
- New Section "Code Coverage" in the report.

## Dashboard

- You can now choose to set the mode of the button widget and check box widget either to push button or to toggle button.

## Dashboard Player

- Simplified generation of the executable Dashboard Player file in a new format called DBPLAY.
- The DBPLAY files are automatically associated with the Dashboard Player.
- New command line options to start the a DBPLAY file automatically (`--autostart`) and in fullscreen.
- New button to select a TVM file in case several TVM files exist.
- Choose a TVM file in the Dashboard Player

## Miscellaneous

- New command line option `--noconsole` to suppress any output in the command prompt.
- The variables that should be used in all projects of a TPT installation can be now defined in the "General" section of the Preferences dialog. Variables that should only apply to a specific TPT project are defined in a project-specific section in the Preferences dialog.
- Select "Enable auto line wrap" menu option in the Description view to automatically break lines.
- To open the most recently closed TPT file, select the new menu item "Open Most Recent".

- Multiple selection is now available in the project tree and the test cases tree.
- The names of project tree elements (testlet, variant, test case) that are created, copied or duplicated and have the same name as another project tree element, are automatically extended by an underscore and a number to ensure the uniqueness of the name.
- New search dialog: The local and global search are now in the same search dialog (CTRL+F).
- Text in the Description view and Documentation steps and fields can be formatted (font size, color, style).
- You can assign initial channel values for every single test case, variant, and group.
- New documentation available and searchable in TPT. New help view.
- Improved context sensitive help in TPT (F1).
- New Preferences dialog.
- The image resolution settings are directly available in the "Save image as" dialog.

# Release Notes TPT 9

## Test Step List

- New Parallel step.
- New options "first" and "last" for the Compare step.
- Nested If-steps.
- Improved Signal Import Wizard in Step List.

## Assessment

- Assessment variables can be declared as arrays.
- Comparison of array, matrix and structured signals supported in the Signal Comparison assesslet and Min/Max Comparison assesslet.
- New assessment function resampleOnChange().
- New report table of the Signal Comparison assesslet.

## Incubation Features

- New Condition Tree Assesslet.
- Linked Declarations: share declarations between different files .

## TASMO

- TASMO is now a regular feature and no longer incubation feature.
- Support of Stateflow models with specific coverage goals.
- Support of a static analysis of model input dependencies for more efficient test data search.
- Support of more blocks like "Compare to Zero " and "Compare to Constant".
- Matrix and vector signals are supported.
- TASMO will generate test data even if the model contains elements that are not specifically supported.

## Platforms

- New Silver platform.
- Multi-core test execution possible for EXE and FUSION platforms.
- Support of internal signals in MATLAB/Simulink improved.
- Interface import and parameter exchange from Simulink Data Dictionaries are supported.
- Dataset format is now supported for Simulink signal logging.

- Signal logging in referenced models is supported.
- Support of AUTOSAR signals.
- Test of inner TargetLink subsystems possible .
- Support of Simulink Fast Restart for MATLAB 2015b or newer.

## **PLS UDE**

- The PLS UDE FUSION node can handle all data types available in , including structs and user defined custom data types.
- Several instances of TPT and UDE can be executed in parallel.
- Progress logging of the PLS UDE FUSION node can be turned off to increase the performance.
- The PLS UDE FUSION node can read and write axis of maps and curves.

## **FUSION**

- Use of two .NET dlls is now possible.
- The initial signal values may optionally be taken from the custom nodes dll instead of the default values from the Declaration Editor.

## **Equivalence classes**

- Sets of equivalence classes can be created and managed in the new Equivalence Classes Editor.
- Assign equivalence classes to channels in a Test Step List.

## **Dashboard**

- Dashboard script API extended.
- New Dashboard Player Guide.
- Undo / Redo for text fields in the Dashboard Configuration is now supported.
- Signals in widgets can be set via drag and drop operation from the Declaration Editor.
- Creation of new widgets by dragging signals onto an empty space in the dashboard.
- Dashboard file (\* .dashboard) can be opened by a drag and drop operation into TPT.

## **Import/Export**

- TPT can read signal data from .xls and .xlsx files .
- Import interface :
  - Check boxes to ignore differences of specific properties.
  - It is possible to import the rename information and the default values into a mapping instead of changing the values in the Declaration Editor.

## Requirements

- In case a requirement is linked to test cases, the result of the test cases will influence the result of the requirement. This way inconclusive results are prevented if requirements are not directly linked to assesslets but to test cases.
- Re-import of test cases keeps the folder structure of tests.
- Requirements coverage information in the report extended.
- Improved display of the changes in the modifications table of the requirements.
- The modifications table in the Requirements view is accessible via a shortcut.

## Miscellaneous

- Revised Type Editor.
- Revised FUSION documentation.
- New Test Modeling Quick Reference.
- New Jenkins plug-in documentation.
- Transitions can be prioritized (primary, secondary).
- Coverage information table is also shown in the Reclassification overview report.
- Test sets can be defined using a conditional expression.
- Integration with Test Rail (Gurock) for the exchange of test cases and test results.
- Local Search mechanism extended to more views and dialogs.
- Auto-completion in the Parameter tab.
- New examples
- Mil/Sil example for Simulink
- MATLAB platform for Stateflow
- Simulink Datastores

# Release Notes TPT 8

## Test Step List

- Improvement of the performance when loading measurement data using the Import signal step.
- Import several signal from the same file using only one Import signal step.

## Execution Configuration

- Assessments can be deactivated for parts of the Execution Configuration. This way you can choose to run the assessment only on specific platforms and therefore, increase the performance.

## Assessment

- New completely revised browser based Assessment API for script assessment function. This API is accessible through the context sensitive help system (Ctrl+Spacebar while writing a script).
- The header area for assesslet in the "Assesslet Content" view has been redesigned and can be folded/unfolded on demand.
- New PT1 filter function. This new function filters a time dependent float expression using the specified PT1 with coefficients K (gain) and T (time constant) and returns the filtered signal.
- The report option "Report always (not only on error)" is by default deactivated for each assesslet. Now this option can be activated by default in the General Settings | Assesslet Settings section in the TPT ToolPreferences dialog box.
- The functions TPT.always, TPT.exists, and TPT.never return FALSE if the expression is undefined for all points in time in the current context interval.
- The State Sequence Viewer known from the Test Data Viewer is now also available in the report.
- Previous assessment results were imported within signals when using import measurements, signal comparison or a script assesslets. Now, no result information is imported within signals, so imported signals cannot influence the actual test result.

## Parameter

- The Parameter tab has one new column showing whether values have been defined in variants or subgroups.

- MATLAB: the parameter exchange of curves, maps and structs parameters is now supported. You have to use a custom script/function. See User Guide for detailed information.

### **TASMO: Structural test case generation for Simulink models**

This new tool analyzes a MATLAB/Simulink or TargetLink model and generates test data with the aim of satisfying structural coverage criteria.

With the help of such automatically generated test data, TASMO supports you to improve your MATLAB/Simulink model test coverage in an early state of the development process.

TASMO finds the minimum number of test cases to achieve a maximum of the structural coverage.

You can choose among single criteria, decision coverage, condition coverage, and all coverage types together.

The generated test data can also be used for back-to-back tests to check whether the model equals the compiled code.

### **Dashboard**

- Automatic build of the executable for the Dashboard Player from the EXE platform or the MATLAB platform.
- Dashboard projects can be exported to a separate folder containing all necessary data to be run in the Dashboard Player.
- You can now record user actions in the Dashboard during the test execution. Afterwards, you can use this recording to generate a Test Step List.
- Multi State widget:
  - Click Area Editor
- Selector widget:
  - Automatic recognition of enums
  - New button style
- Bits widget:
  - Horizontal position
  - New button style
  - Bit list generator, offers the possibility to automatically generate a list of bits to be used in the widget
- Discretization via interval and quantization for
  - Gauge widget
  - Digital Display widget
  - Slider widget
- Rotate or mirror images in



- Image widget
- Multi State widget
- Custom Button widget

## Testing Platforms

- New Assessment platform available for the assessment of measurement data without any test execution.
- New ASCET@FUSION platform.
- The step size for the FUSION can be set independently from the step size of the virtual machine.
- ASCET platforms can send and receive messages.

## Test Case Status

The Test Case Status view gives you an overview over the current status of your test cases, as well as the revisions of a test case. The history of modifications is tracked in a list (date of execution, status of the test case, author, comment, tags, number of revisions). You can also create your own custom test case status.

## Import/Export

- MDF 4 supported for export and import.
- For CSV imports arrays, maps and curves are supported.

## Automatic Test Case Generation from Equivalence Classes.

Automatic test case and variant generation from equivalence classes. Use your already defined equivalence classes mappings to automatically generate test cases and variants. In the example below we set the following equivalence classes:

- Light intensity: low = [0,60[, medium = [60,70], high = ]70,100]
- Headlight: OFF = [0,0], ON = [1,1]
- Light switch: OFF = [0,0], ON = [1,1], AUTO = [2,2]

Before generating the test cases/variants, you can choose from different combinatorics, like for example, merging all generated data into one single variant.

## Miscellaneous

- The new First Use Wizard can build a new fully configured MATLAB/Simulink or ASCET project for you in just a few steps.
- New Test Case Details view with editable test case attributes.
- Signal preview can be switched to different platforms with different mappings in the step list view.

- Installer allows combined installation of a 32 bit and 64 bit process.
- Installation process possible without GUI setup wizard: silent mode.
- Improved display of structured signal names in the test data viewer.
- New context menu for test step list
  - Create Step
  - Activate/Deactivate
- New local search dialog box available at certain views.
- Scaling removed from the Declaration Editor. Scaling only applies to the scaling flavor.
- Unit declaration will be shown in any signal graphic.
- Execution information in the project browser: test report, test data, and assessment data is now directly available in a dropdown menu for each test case.

# Release Notes TPT 7

## Step list

- Direct definition step is now integrated in the channel step. You can choose to assign a value to the channel once or to assign a direct definition to the channel always.
- New while loop step.
- Compare steps are now shown in the report.
- Embedded signal step can now handle elements from structs, arrays and matrices.
- Ramp channel supports matrices and arrays.

## Assessments

- Automatic testing for equivalence classes using an equivalence classes mapping.
- New `tpt.hose` options

## Requirements

- Improved multiselection for requirements.
- Remove all "Needs to be reviewed" marks at once.

## New incubation features

- Integration with CANoe.
- Remote API to run the main features.
- Version control using Subversion.

## Dashboard

- Use a dashboard player to run dashboard files without a TPT license.
- Create a dashboard in a more detailed way using a script environment instead of a GUI.
- New widgets: gauge, multi state and selector.
- Graph widget can display several signals.
- Image widget can change rotation, brightness and transparency according to a signal.
- You can use internal signals to influence widgets without having to declare them in the declaration editor.
- Widgets can be now grouped.
- Quickly switch between panels.
- Change the visibility of widgets or groups.

- All widgets have notes / usage instructions that can be displayed during runtime.
- Dashboard browser.

## Documentation

- New examples description tab. From there you can directly open the examples files.

## Platforms

- dSPACE HiL supports MLIB/MTRACE API.
- Array-/Matrix signals can be now exported to the MATLAB workspace using the M-Script assesslet.
- You can configure different ASCET versions (or edit/rename/delete them) in the ASCET platform preferences dialog without changing any configuration.
- Integration with Concurrent HiL and VeriStand (National Instruments) platforms.

## Report

- New advanced report options through the execution configuration.
- Pie charts in all reports.
- Scatter plots with `scatter_plot = TPTReport.ScatterPlot` function.

## Miscellaneous

- New functions and controls for the test data viewer.
- Attachments can be added in the description window.
- 32bits/64bits versions in just one installer.
- System constants support.
- New command line option "`--prefvar foo=bar`" to set environment variables.
- New copy/paste features.
- Maps and curves can be more easily edited with a new wizard.
- Declaration Editor: It is possible to enter the value in the value field directly in a binary format (only with integer data types).
- SMF 4.0 support.