

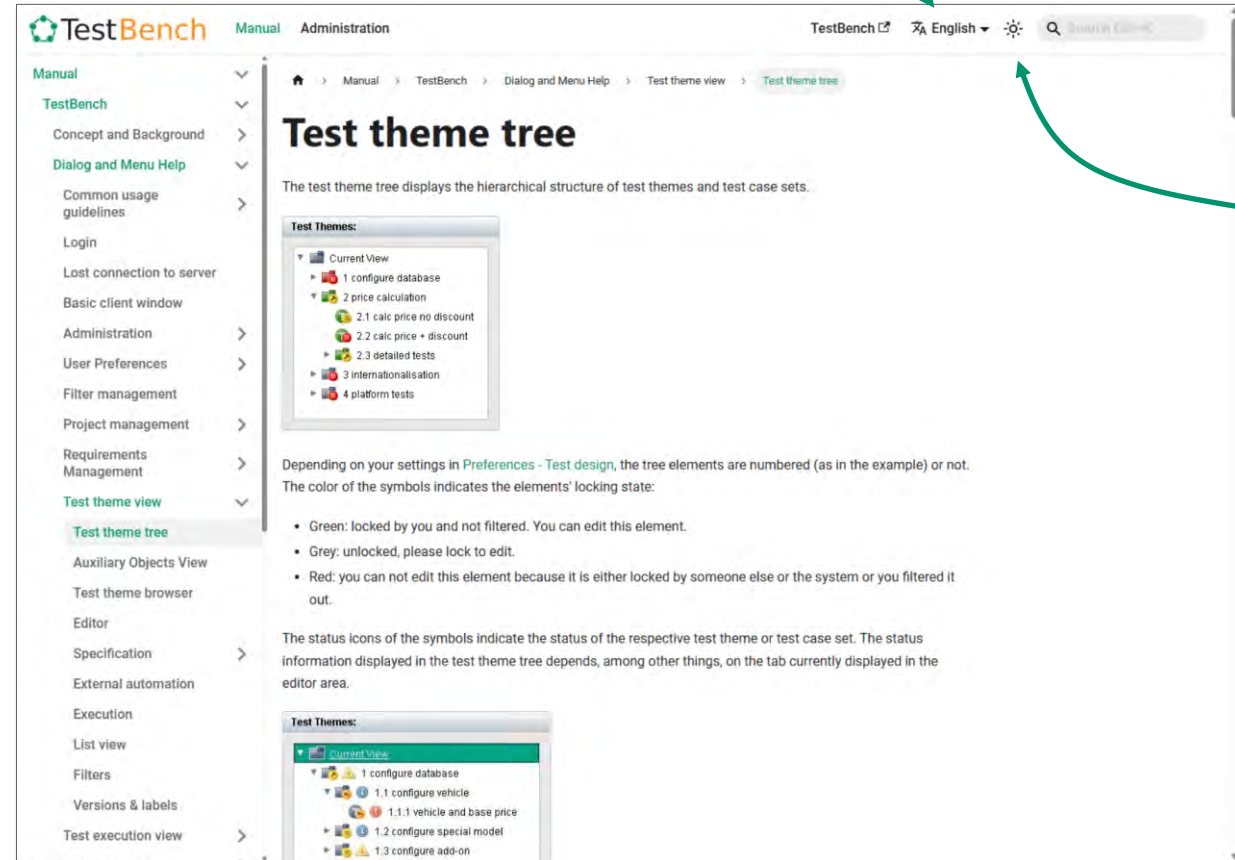
New Features in Version 4.0

Patchlevel 4.0.39 BETA – 10.02.2026

Online Help

- The online help is now opened in the browser and can be opened and read independently of the TestBench client.
- New features
 - Switching languages within the help section
 - Light and dark mode
 - Comprehensive full-text search
 - Glossary

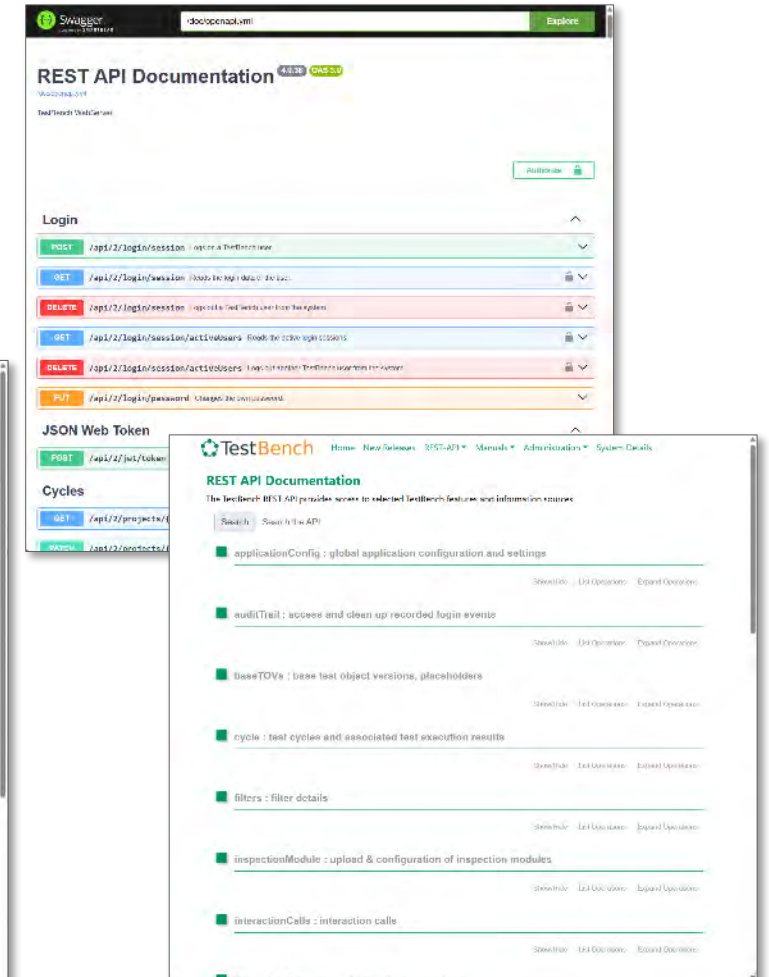
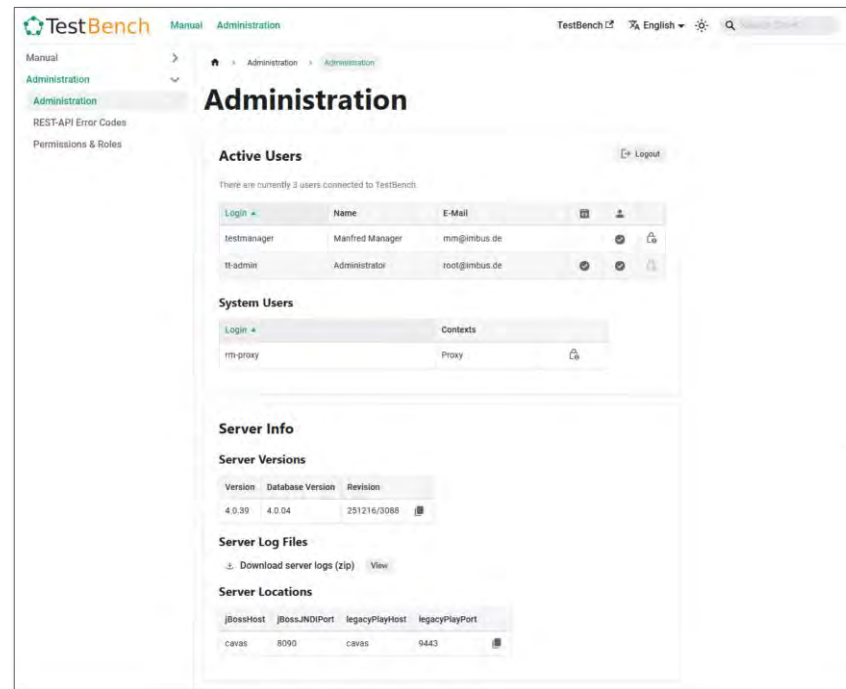
switch language



*dark mode
light mode*

Administration Pages

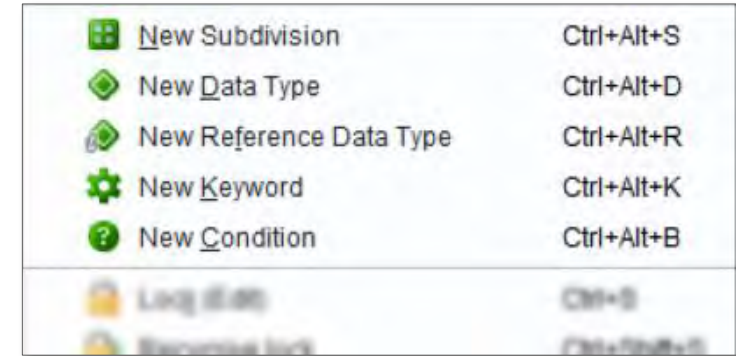
- The administration pages in the backend contain important information:
 - Logged-in users, system accounts
 - Information about the server
 - Server log files
 - Server version info
 - License info
- REST API documentation
- Release notes



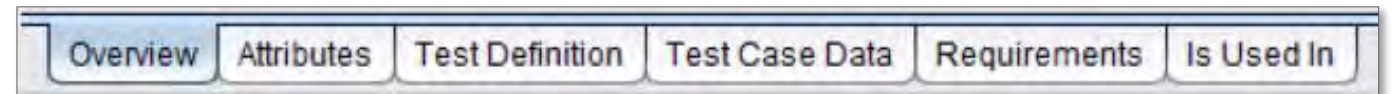
TestBench Terminology

Adaptation to the ISO 29119-5 standard Keyword Driven Testing

Previous term	New term
Interaction	Keyword
Keyword	Tag
Test Cases	Test Case Data
Definition	Test Case Definition



Detail of the context menu for test elements



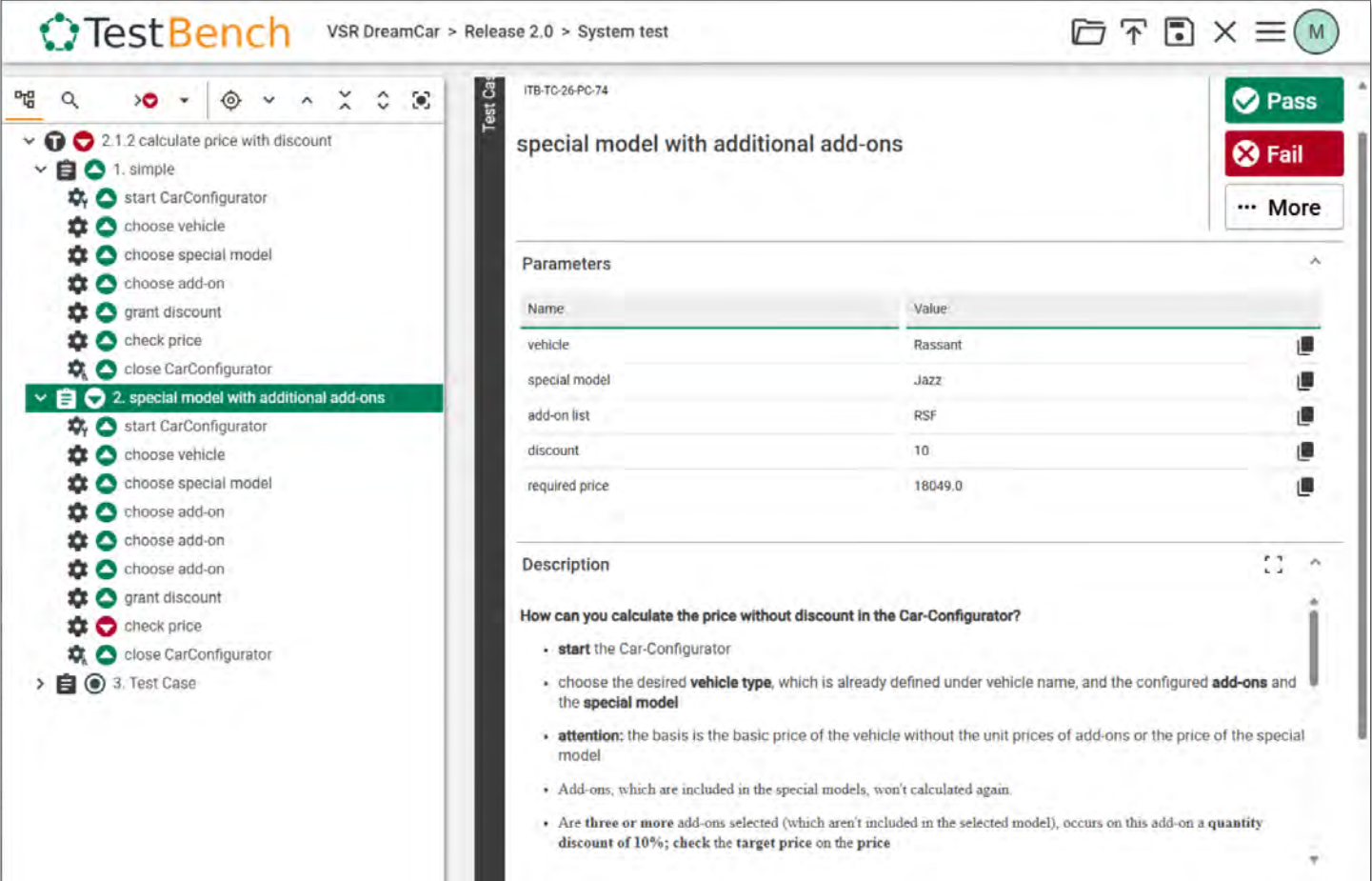
Tab cards of a test case set

No changes have been made to technical export formats, e.g. the XML full report or project export/import, so backward compatibility is maintained.

Java iTORX becomes Web iTORX

Existing iTORX becomes a Progressive Web App (PWA)

- Starts in the web browser
- Opens archives or is called from the client
- PWA = can be installed like an app via the browser, but still updates automatically



The screenshot displays the TestBench web interface. The top navigation bar shows the TestBench logo, the current test path "VSR DreamCar > Release 2.0 > System test", and standard browser window controls. The main content area is divided into three sections:

- Test Case List:** A tree view on the left shows test cases. The current test case, "2. special model with additional add-ons", is highlighted in green. It includes steps like "start CarConfigurator", "choose vehicle", "choose special model", "choose add-on", "grant discount", "check price", and "close CarConfigurator".
- Test Case Details:** The right side shows the details for the selected test case, "special model with additional add-ons". It includes a status indicator (Pass/Fail/More), a "Parameters" table, and a "Description" section.
- Parameters Table:**

Name	Value
vehicle	Rasant
special model	Jazz
add-on list	RSF
discount	10
required price	18049.0
- Description:** A text area containing instructions: "How can you calculate the price without discount in the Car-Configurator?" followed by a list of steps and attention points.

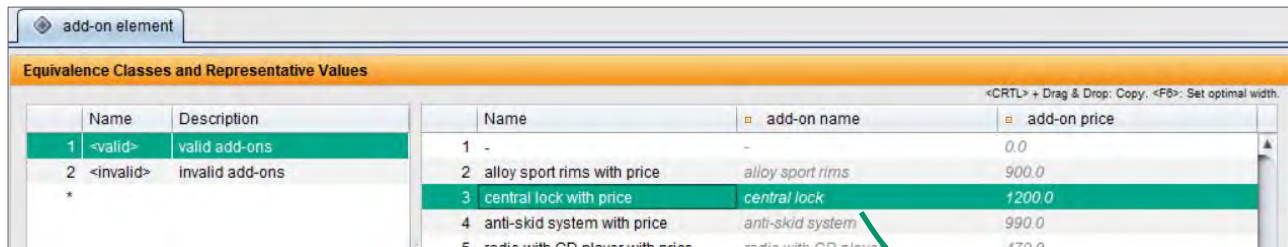
Compound data types in Web iTORX (1/2)

Values of compound representatives are displayed

Initial situation :

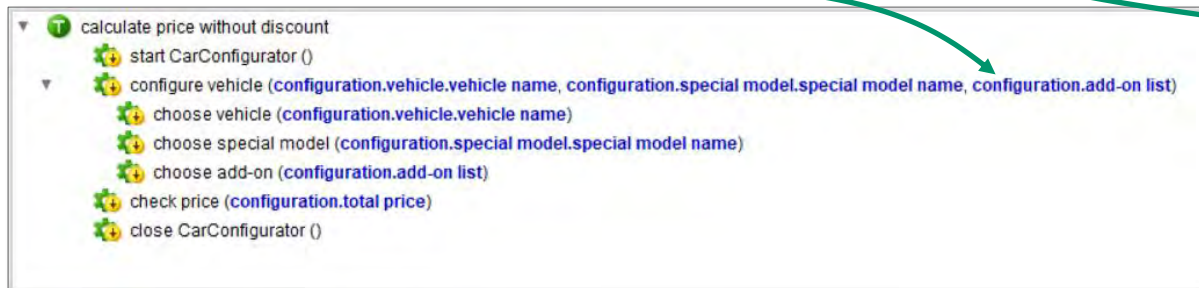
Names of representatives of compound data types are not displayed at the higher level; instead, the detailed values of the sub-data types are displayed.

Compound data type in TestBench

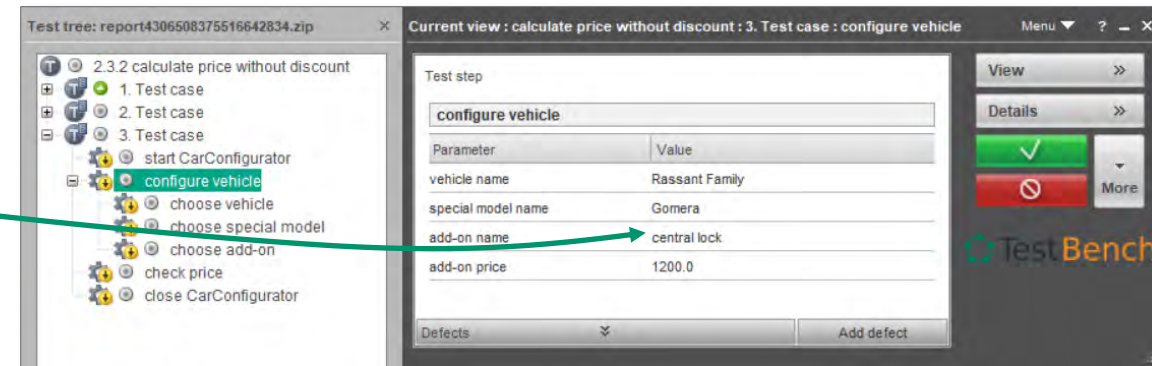


Name	Description	Name	add-on name	add-on price
1 <valid>	valid add-ons	1 -	-	0.0
2 <invalid>	invalid add-ons	2 alloy sport rims with price	alloy sport rims	900.0
*		3 central lock with price	central lock	1200.0
		4 anti-skid system with price	anti-skid system	990.0
		5 radio with CD player with price	radio with CD player	470.0

Usage in test sequence



- calculate price without discount
 - start CarConfigurator ()
 - configure vehicle (configuration.vehicle.vehicle name, configuration.special model.special model name, configuration.add-on list)
 - choose vehicle (configuration.vehicle.vehicle name)
 - choose special model (configuration.special model.special model name)
 - choose add-on (configuration.add-on list)
 - check price (configuration.total price)
 - close CarConfigurator ()



Test tree: report4306508375516642834.zip

Current view: calculate price without discount : 3. Test case : configure vehicle

Test step: configure vehicle

Parameter	Value
vehicle name	Rassant Family
special model name	Gomera
add-on name	central lock
add-on price	1200.0

Defects: [dropdown] Add defect

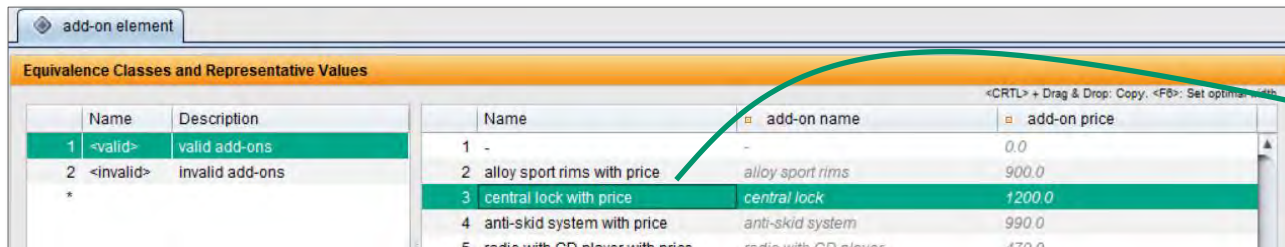
Detailed values in iTORX

Compound data types in Web iTORX (2/2)

Values of compound representatives are displayed

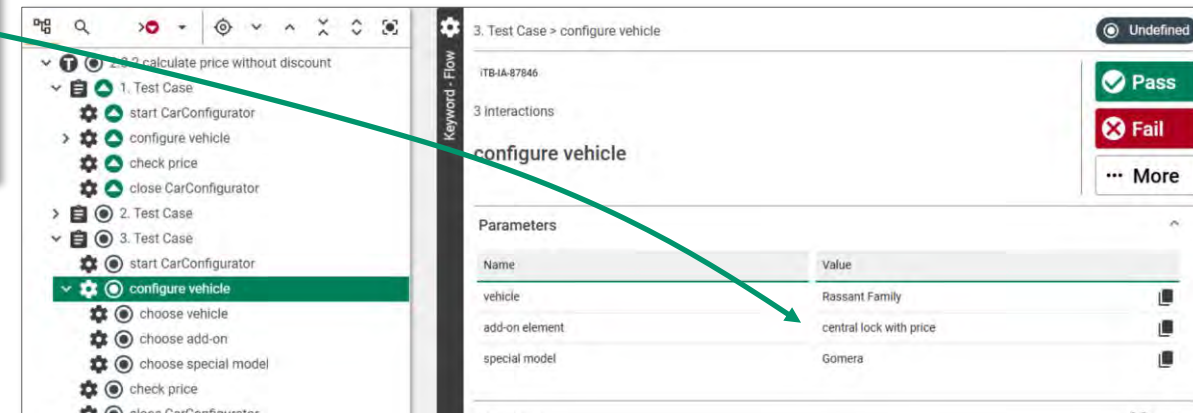
New:
Web iTORX displays the names of representatives of compound data types at the parent level.

Compound data type in TestBench



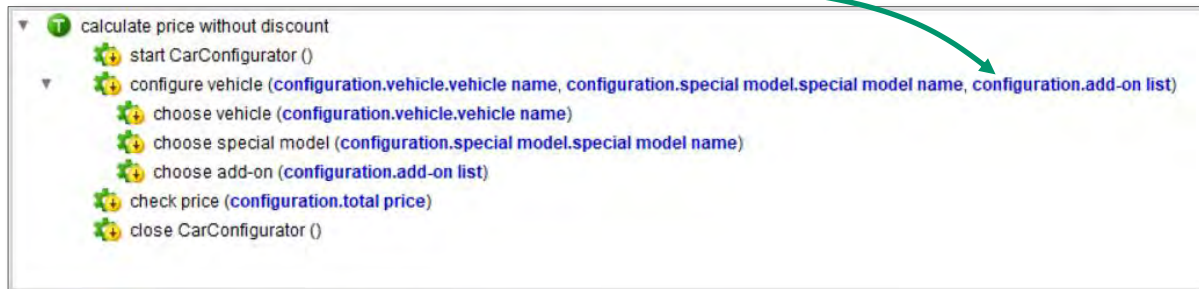
Name	Description
<valid>	valid add-ons
<invalid>	invalid add-ons

Name	add-on name	add-on price
1	-	0.0
2	alloy sport rims with price	900.0
3	central lock with price	1200.0
4	anti-skid system with price	990.0
5	radio with CD player with price	470.0



Name	Value
vehicle	Rassant Family
add-on element	central lock with price
special model	Gomera

Usage in test sequence



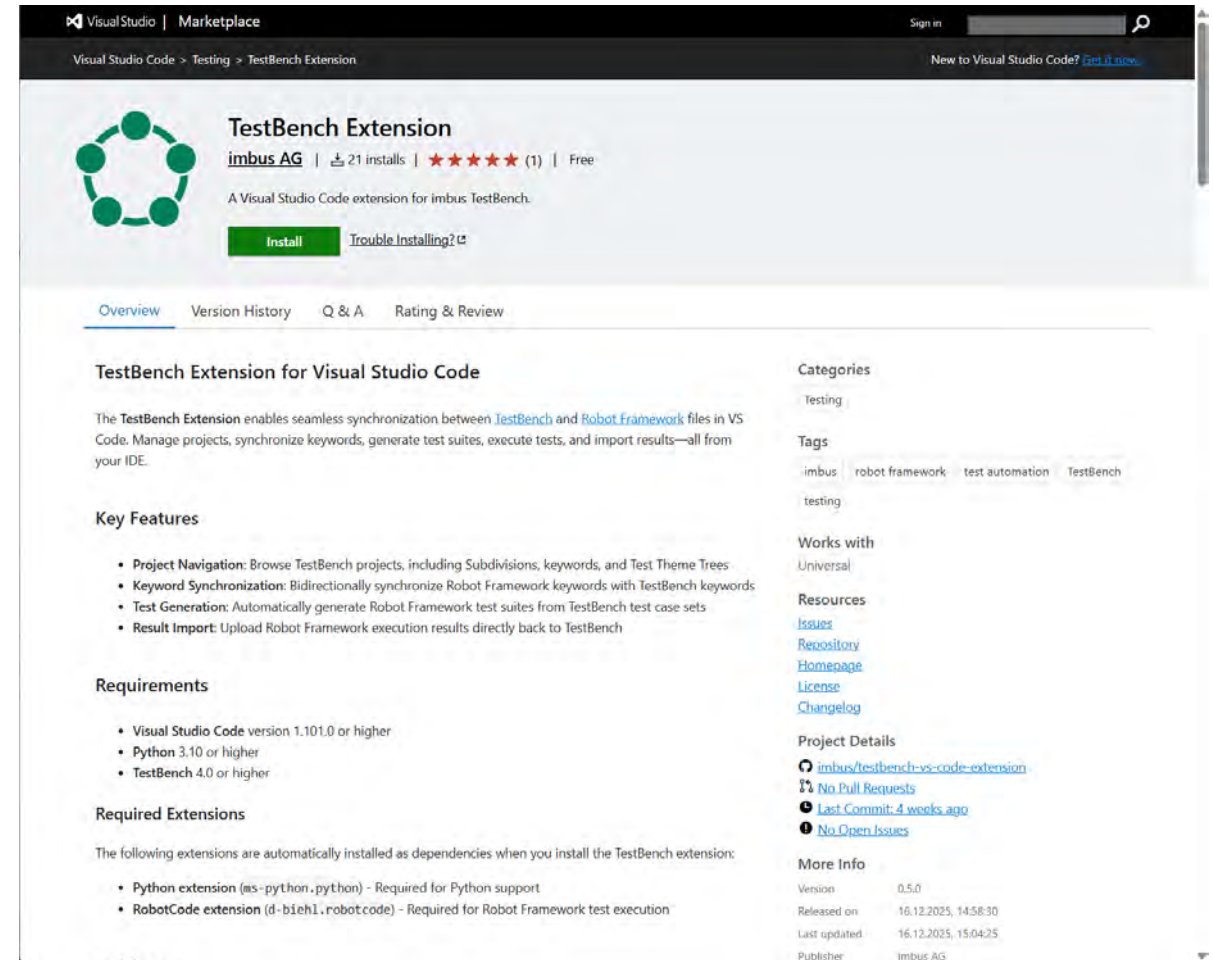
- calculate price without discount
 - start CarConfigurator ()
- configure vehicle (configuration.vehicle.vehicle name, configuration.special model.special model name, configuration.add-on list)
 - choose vehicle (configuration.vehicle.vehicle name)
 - choose special model (configuration.special model.special model name)
 - choose add-on (configuration.add-on list)
 - check price (configuration.total price)
 - close CarConfigurator ()

Name of representative

Visual Studio Code Extension

Specification = Automation

- Expert specifications for keyword-driven tests and their test data in TestBench
- Technical automation, coding and debugging of keywords in VS Code Extension
- Expert testers work exclusively in TestBench
- Test automation engineers work exclusively in VS Code.



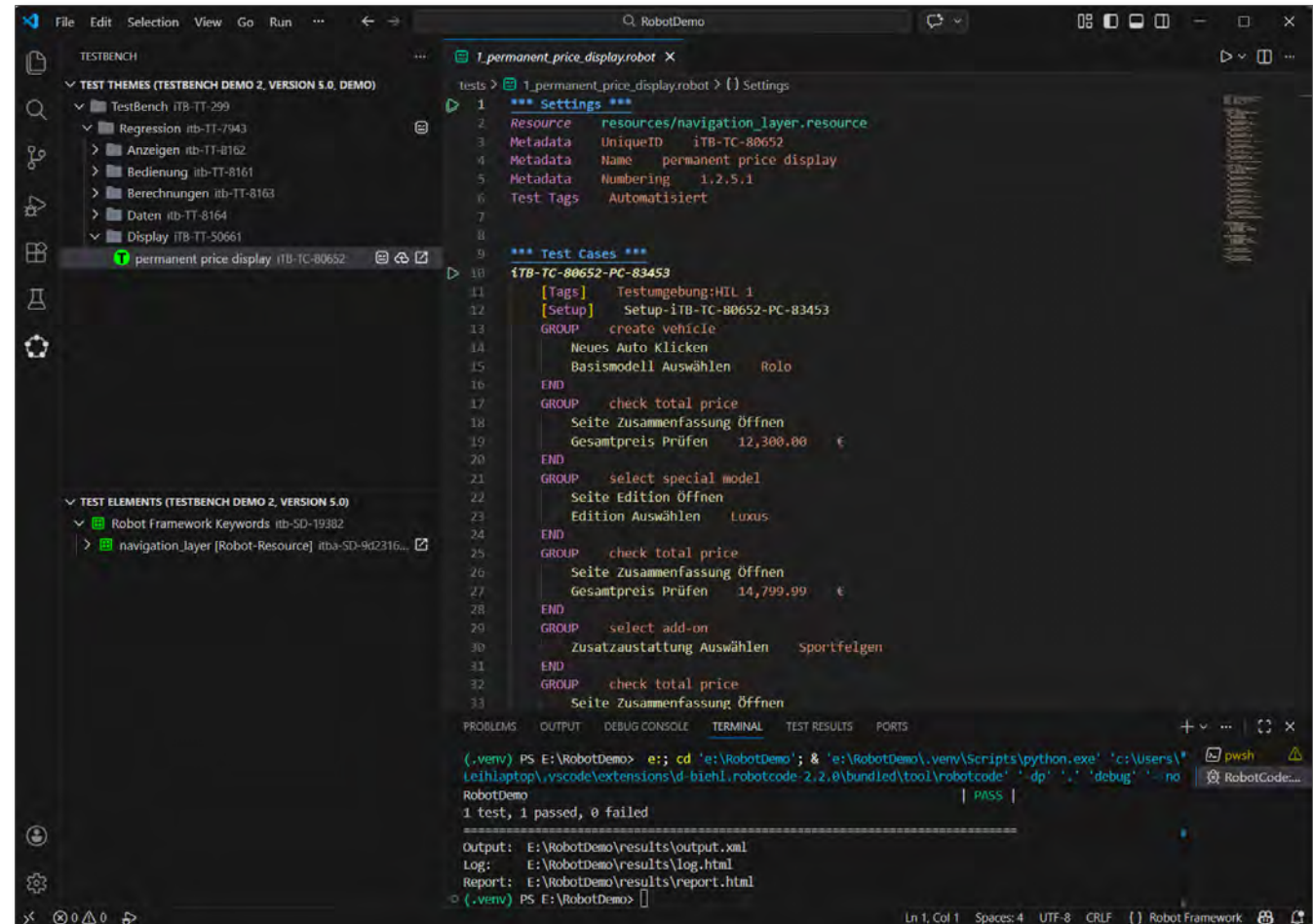
Visual Studio Marketplace:

<https://marketplace.visualstudio.com/items?itemName=imbus.testbench-extension>

Visual Studio Code Extension

Integration of Robot Framework based test automation

- TestBench tests are generated as Robot Framework tests.
- Keywords are implemented in VS Code
- New keywords are synchronised with TestBench (keywords can be created on both sides)
- Names and descriptions are synchronized with TestBench



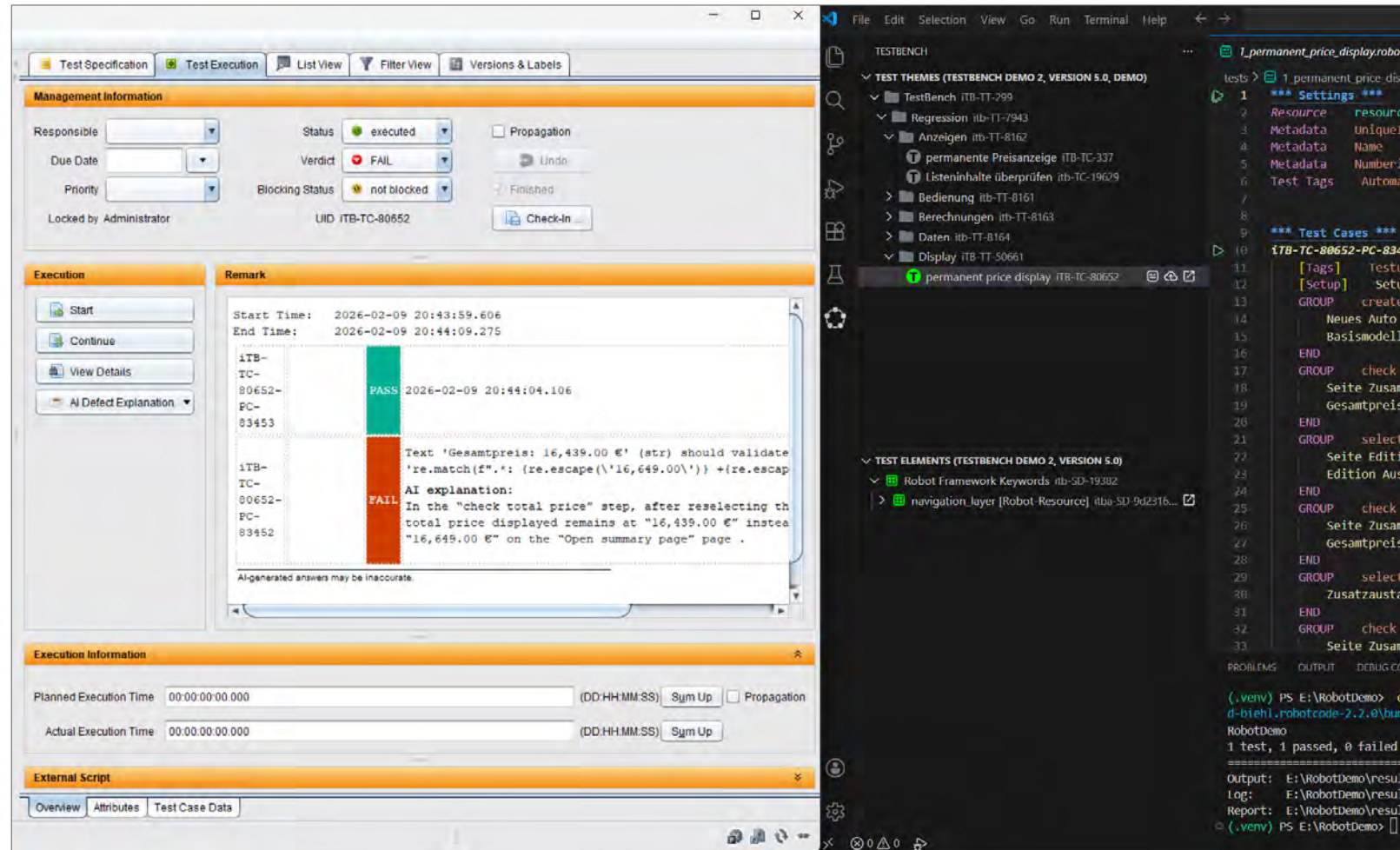
```
1 permanent_price_display.robot X
tests > 1_permanent_price_display.robot > {} Settings
1 *** Settings ***
2 Resource resources/navigation_layer.resource
3 Metadata UniqueID iTB-TC-80652
4 Metadata Name permanent price display
5 Metadata Numbering 1.2.5.1
6 Test Tags Automatisiert
7
8
9 *** Test Cases ***
10 iTB-TC-80652-PC-83453
11 [Tags] Testumgebung:HIL 1
12 [Setup] Setup-iTB-TC-80652-PC-83453
13 GROUP create vehicle
14     Neues Auto Klicken
15     Basismodell Auswählen Rolo
16 END
17 GROUP check total price
18     Seite Zusammenfassung Öffnen
19     Gesamtpreis Prüfen 12,300.00 €
20 END
21 GROUP select special model
22     Seite Edition Öffnen
23     Edition Auswählen Luxus
24 END
25 GROUP check total price
26     Seite Zusammenfassung Öffnen
27     Gesamtpreis Prüfen 14,799.99 €
28 END
29 GROUP select add-on
30     Zusatzausstattung Auswählen Sportfelgen
31 END
32 GROUP check total price
33     Seite Zusammenfassung Öffnen
```

```
(.venv) PS E:\RobotDemo> e:; cd 'e:\RobotDemo'; & 'e:\RobotDemo\.venv\Scripts\python.exe' 'c:\Users\Leihlaptop\vscode\extensions\d-bieh1.robotcode-2.2.0\bundled\tool\robotcode' -dp '' -debug -no
RobotDemo | PASS |
1 test, 1 passed, 0 failed
-----
Output: E:\RobotDemo\results\output.xml
Log: E:\RobotDemo\results\log.html
Report: E:\RobotDemo\results\report.html
(.venv) PS E:\RobotDemo>
```

VS Code Extension – Additional Functions

Save test results directly

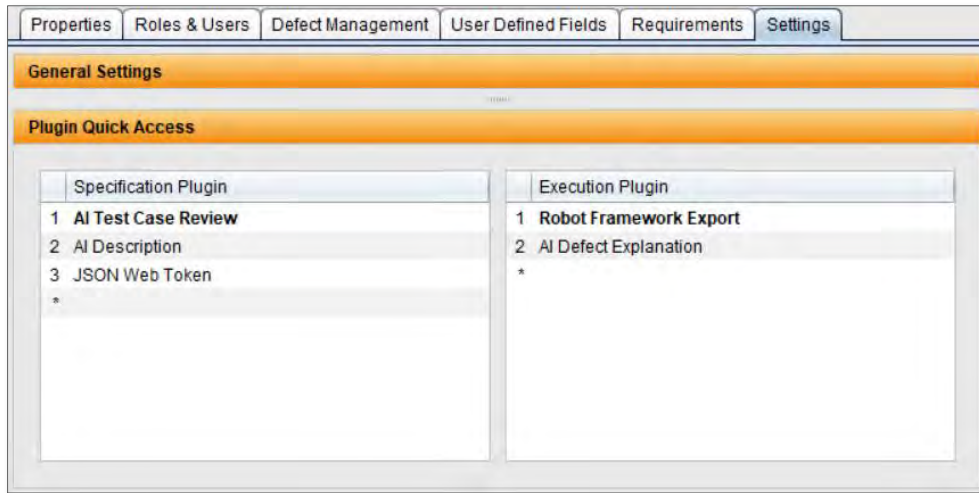
- Tests can be performed in VS Code(*)
- Test results can be saved directly back to TestBench



(*) with the help of free RobotCode Extension

Plugin Quick Access

- The buttons for calling up plugins introduced in the last release have been expanded so that several plugins can be configured behind each button:

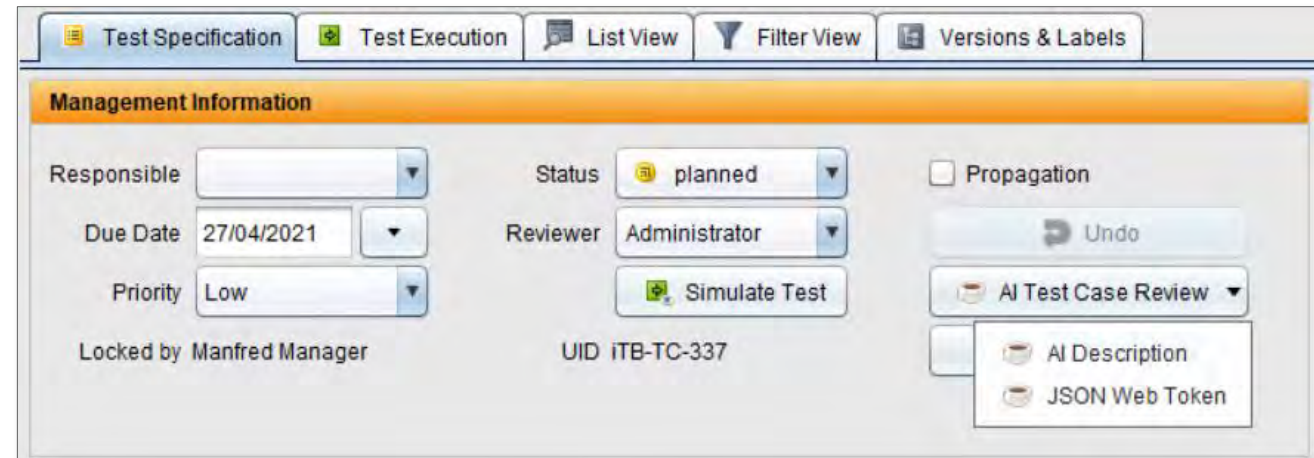


Definition of the buttons

Specification buttons



Execution buttons

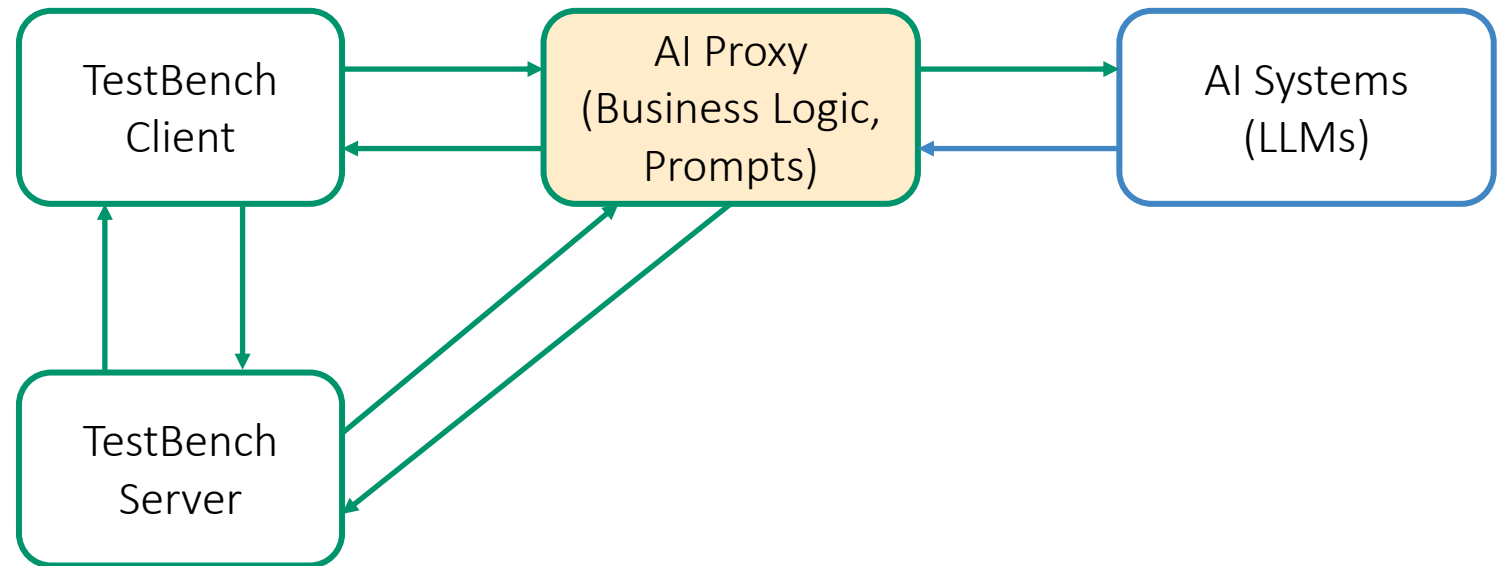


TestBench AI Assistants

Structure of the AI service(AI proxy)

Strong focus on flexibility and adaptability

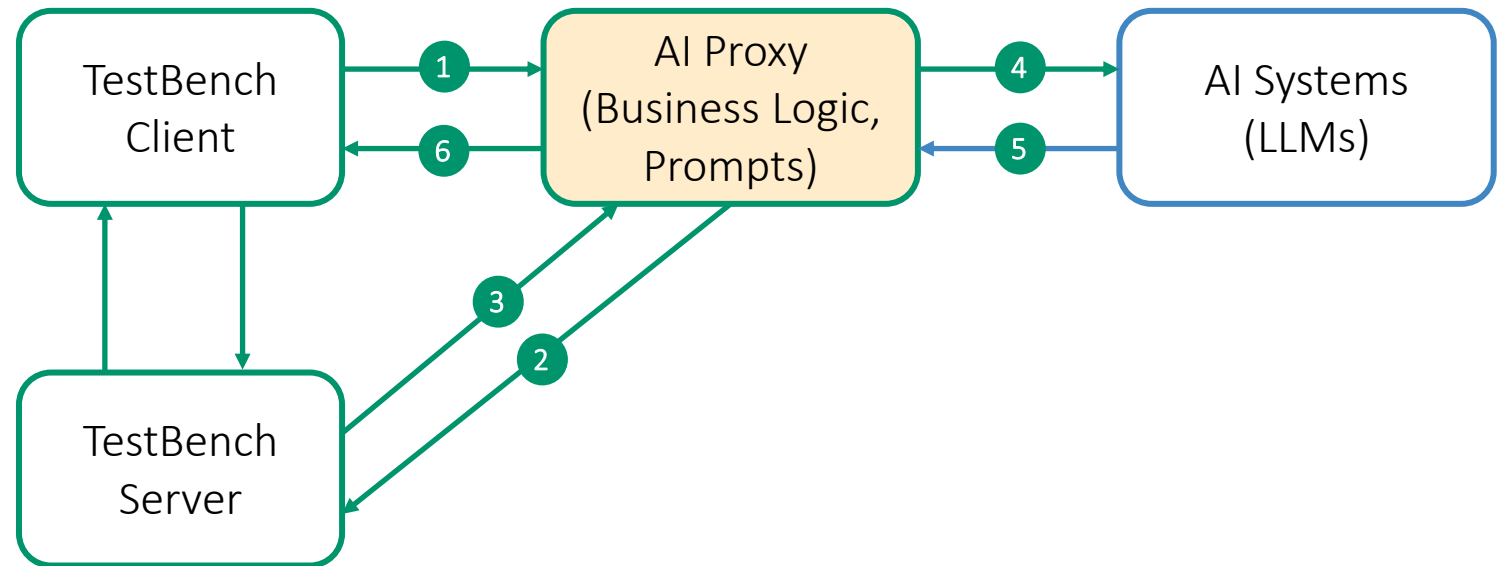
- Modular prompts can be customized and configured according to customer/project requirements
- open source
- Can be connected to various AI systems: local systems, private clouds, public clouds
- Standard:
Public OpenAI ChatGPT



TestBench AI Assistants

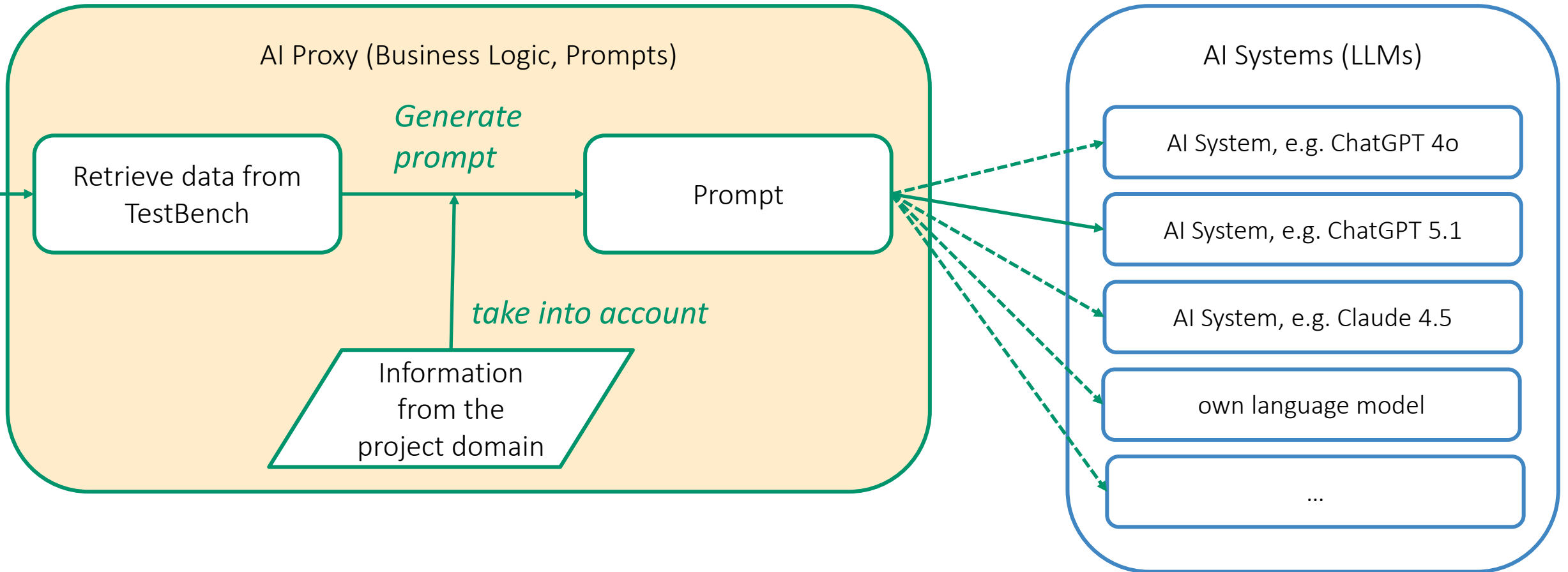
Internal flow

1. The client initiates the request for a specific use case to the AI bridge in the AI proxy and provides basic information, such as the ID of a test case set.
2. The AI Bridge retrieves the information required to generate prompts, etc., from the TestBench server.
3. The TestBench server delivers the requested data to the AI bridge.
4. The AI Bridge generates the request and prompt and sends them to the configured AI.
5. The AI bridge processes the AI's response.
6. The AI Bridge prepares the AI's response and sends the result to the client.



TestBench AI Assistants

Inner architecture

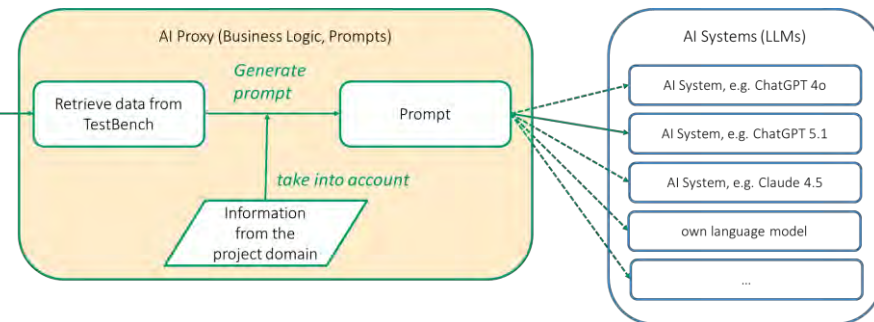


TestBench AI Assistants

AI functions (use cases) can be flexibly adapted and expanded

- Improving and more specialized language models are enabling more and more new AI solutions.
- Use cases each use the optimal(*) language model instead of all using the same one
- Integrating domain knowledge delivers better results
- Involvement of internal AI experts

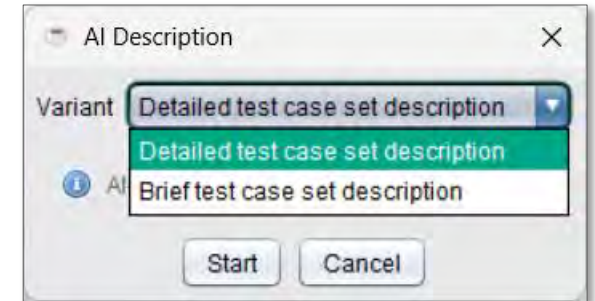
(*) *e.g. with regard to capabilities, licensing, location, data protection regulations*



TestBench AI Assistants

Generation of the test case set description

- Generation of general description from test sequence and test case data
- Variants can be stored for each assistant:
Here: Detailed and brief test case description



General Description

AI description - 2026-02-09 20:56:20

This test case set checks that the total price in CarConfig is continuously updated and displayed after each configuration step (create vehicle, select special model, select add-on).

Checks:

- After create vehicle, check total price is validated with total_price_after_vehicle.
- After select special model, check total price is validated with total_price_after_special_model.
- After the first select add-on, check total price is validated with total_price_after_add-on_1.
- After the second select add-on, check total price is validated with total_price_after_add-on_2.

Test cases:

- iTB-TC-80652-PC-83453: vehicle=Rolo, special model=Luxury, add-on - 1=Sports rims, add-on - 2=Leather steering wheel; Expected prices: total_price_after_vehicle=12,300.00, total_price_after_special_model=14,799.99, total_price_after_add-on_1=15,699.99, total_price_after_add-on_2=15,699.99.
- iTB-TC-80652-PC-83452: vehicle=mini golf, special model=Gomera, add-on - 1=floor mats, add-on - 2=heated exterior mirrors; expected prices: total_price_after_vehicle=15,000.00, total_price_after_special_model=16,413.00, total_price_after_add-on_1=16,439.00, total_price_after_add-on_2=16,649.00.

AI-generated answers may be inaccurate.

TestBench AI Assistants

Review of test sequences

- Review of test case sets to determine whether they correspond to the implementation and comply with a defined set of rules.
- Rules can be defined in the assistant's prompting.

Review Comments

AI Review - 2026-02-09 21:00:55

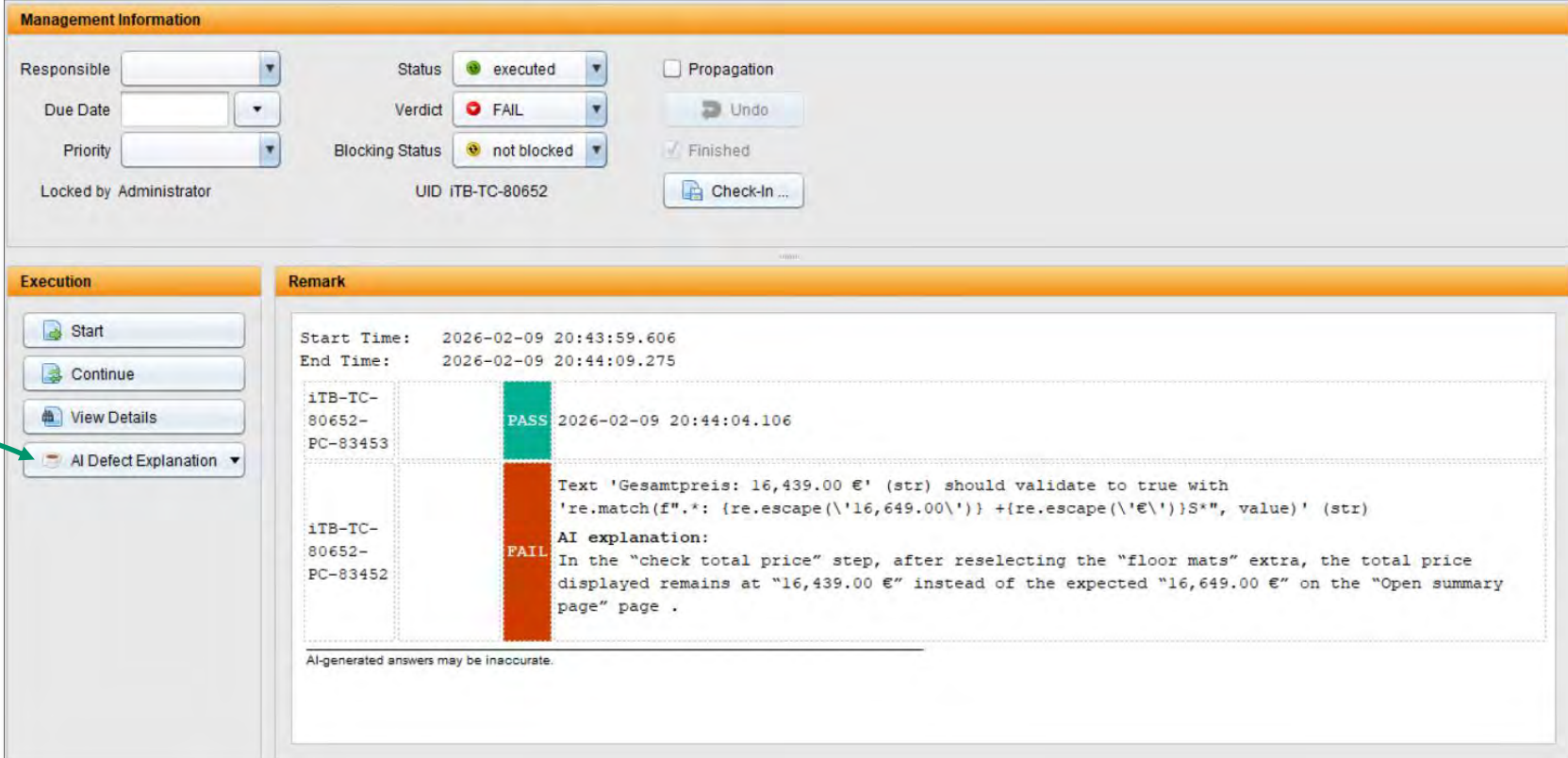
- Test case set does not contain any steps with `step_type:check`; all steps labelled 'check total price' are declared as flow and do not meet the required validation.
- Test case description explicitly refers to AI generation with 'AI description ...' or 'AI-generated answers may be inaccurate'.

AI-generated answers may be inaccurate.

TestBench AI Assistants

Explanation of error messages from automation

- Translation of the technical details of an error message into “domain specific” language
- The original error message remains unchanged.



The screenshot displays the TestBench interface. The top section, 'Management Information', includes fields for Responsible, Due Date, Priority, Status (executed), Verdict (FAIL), Blocking Status (not blocked), and UID (ITB-TC-80652). It also features buttons for Propagation, Undo, Finished, and Check-In. The bottom section, 'Execution', contains buttons for Start, Continue, View Details, and AI Defect Explanation. The 'Remark' section shows a table of test results:

Test Case ID	Status	Time	Message
ITB-TC-80652-PC-83453	PASS	2026-02-09 20:44:04.106	
ITB-TC-80652-PC-83452	FAIL		Text 'Gesamtpreis: 16,439.00 €' (str) should validate to true with 're.match(f".*: {re.escape('\16,649.00\')} +{re.escape('\€\')}S*", value)' (str) AI explanation: In the "check total price" step, after reselecting the "floor mats" extra, the total price displayed remains at "16,439.00 €" instead of the expected "16,649.00 €" on the "Open summary page" page .

AI-generated answers may be inaccurate.

Assistants are called up via plugins

Requirements Management Integration

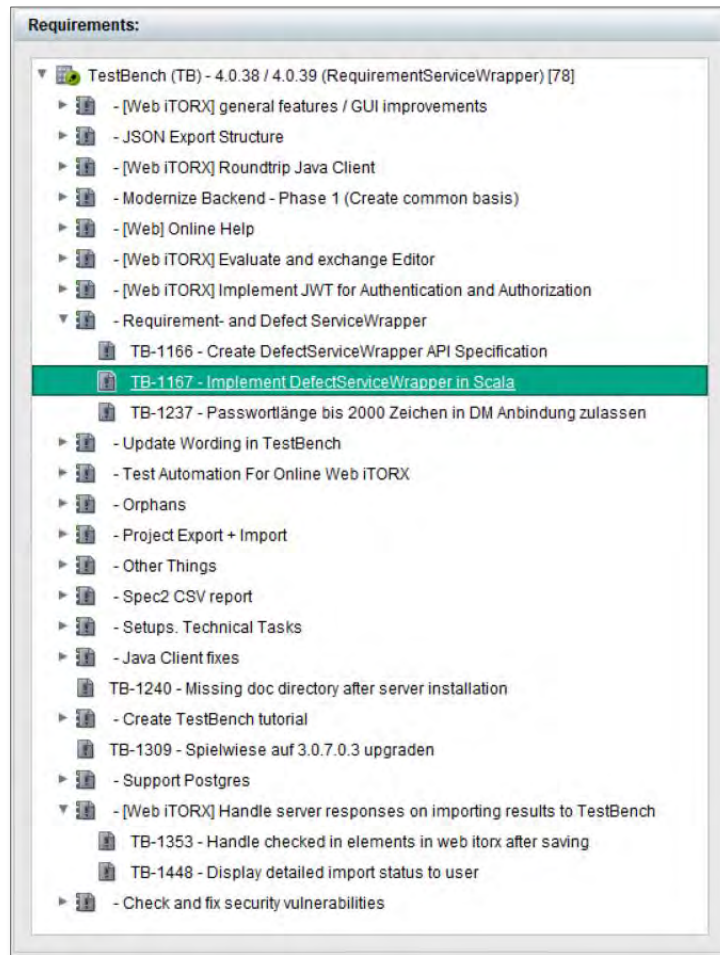
Modernization of the integration of requirements management systems

- New REST API for requirements management proxy
- Initial implementations: XLSX files, JSON line files, Atlassian Jira
- New features :
 - takeover of requirement trees, e.g. for epics and their user stories
 - change histories
- Open source implementation

- Can also be used with TestBench version 3.0.x.

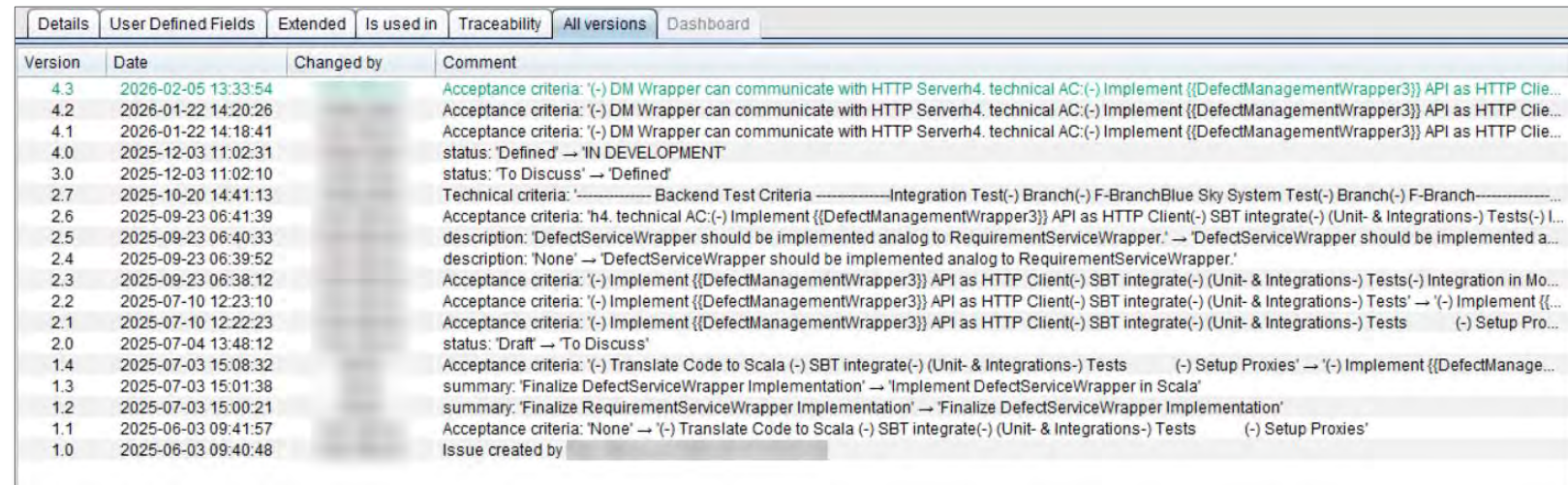
Requirements Management Integration

Example: Atlassian Jira



Requirements:

- TestBench (TB) - 4.0.38 / 4.0.39 (RequirementServiceWrapper) [78]
 - [Web iTORX] general features / GUI improvements
 - JSON Export Structure
 - [Web iTORX] Roundtrip Java Client
 - Modernize Backend - Phase 1 (Create common basis)
 - [Web] Online Help
 - [Web iTORX] Evaluate and exchange Editor
 - [Web iTORX] Implement JWT for Authentication and Authorization
 - Requirement- and Defect ServiceWrapper
 - TB-1166 - Create DefectServiceWrapper API Specification
 - TB-1167 - Implement DefectServiceWrapper in Scala**
 - TB-1237 - Passwortlänge bis 2000 Zeichen in DM Anbindung zulassen
 - Update Wording in TestBench
 - Test Automation For Online Web iTORX
 - Orphans
 - Project Export + Import
 - Other Things
 - Spec2 CSV report
 - Setups, Technical Tasks
 - Java Client fixes
 - TB-1240 - Missing doc directory after server installation
 - Create TestBench tutorial
 - TB-1309 - Spielweise auf 3.0.7.0.3 upgraden
 - Support Postgres
 - [Web iTORX] Handle server responses on importing results to TestBench
 - TB-1353 - Handle checked in elements in web itorx after saving
 - TB-1448 - Display detailed import status to user
 - Check and fix security vulnerabilities



Version	Date	Changed by	Comment
4.3	2026-02-05 13:33:54		Acceptance criteria: '(-) DM Wrapper can communicate with HTTP Serverh4. technical AC:(-) Implement {{DefectManagementWrapper3}} API as HTTP Clie...
4.2	2026-01-22 14:20:26		Acceptance criteria: '(-) DM Wrapper can communicate with HTTP Serverh4. technical AC:(-) Implement {{DefectManagementWrapper3}} API as HTTP Clie...
4.1	2026-01-22 14:18:41		Acceptance criteria: '(-) DM Wrapper can communicate with HTTP Serverh4. technical AC:(-) Implement {{DefectManagementWrapper3}} API as HTTP Clie...
4.0	2025-12-03 11:02:31		status: 'Defined' → 'IN DEVELOPMENT'
3.0	2025-12-03 11:02:10		status: 'To Discuss' → 'Defined'
2.7	2025-10-20 14:41:13		Technical criteria: '----- Backend Test Criteria -----Integration Test(-) Branch(-) F-BranchBlue Sky System Test(-) Branch(-) F-Branch-----...
2.6	2025-09-23 06:41:39		Acceptance criteria: 'h4. technical AC:(-) Implement {{DefectManagementWrapper3}} API as HTTP Client(-) SBT integrate(-) (Unit- & Integrations-) Tests(-) I...
2.5	2025-09-23 06:40:33		description: 'DefectServiceWrapper should be implemented analog to RequirementServiceWrapper.' → 'DefectServiceWrapper should be implemented a...
2.4	2025-09-23 06:39:52		description: 'None' → 'DefectServiceWrapper should be implemented analog to RequirementServiceWrapper.'
2.3	2025-09-23 06:38:12		Acceptance criteria: '(-) Implement {{DefectManagementWrapper3}} API as HTTP Client(-) SBT integrate(-) (Unit- & Integrations-) Tests(-) Integration in Mo...
2.2	2025-07-10 12:23:10		Acceptance criteria: '(-) Implement {{DefectManagementWrapper3}} API as HTTP Client(-) SBT integrate(-) (Unit- & Integrations-) Tests' → '(-) Implement {{...
2.1	2025-07-10 12:22:23		Acceptance criteria: '(-) Implement {{DefectManagementWrapper3}} API as HTTP Client(-) SBT integrate(-) (Unit- & Integrations-) Tests (-) Setup Pro...
2.0	2025-07-04 13:48:12		status: 'Draft' → 'To Discuss'
1.4	2025-07-03 15:08:32		Acceptance criteria: '(-) Translate Code to Scala (-) SBT integrate(-) (Unit- & Integrations-) Tests (-) Setup Proxies' → '(-) Implement {{DefectManage...
1.3	2025-07-03 15:01:38		summary: 'Finalize DefectServiceWrapper Implementation' → 'Implement DefectServiceWrapper in Scala'
1.2	2025-07-03 15:00:21		summary: 'Finalize RequirementServiceWrapper Implementation' → 'Finalize DefectServiceWrapper Implementation'
1.1	2025-06-03 09:41:57		Acceptance criteria: 'None' → '(-) Translate Code to Scala (-) SBT integrate(-) (Unit- & Integrations-) Tests (-) Setup Proxies'
1.0	2025-06-03 09:40:48		Issue created by

Change history of a user story in Jira

Requirements tree with epics, their user stories, tasks and bugs

Defect Management Integration

Modernization of the integration of defect management systems

- New REST API for defect management proxy
- Initial implementations: XLSX files, JSON line files, Atlassian Jira
- New features :
 - Creating bugs from TestBench (also for Jira)
 - Synchronization of all custom fields (= user defined fields in TestBench)
- Open source implementation

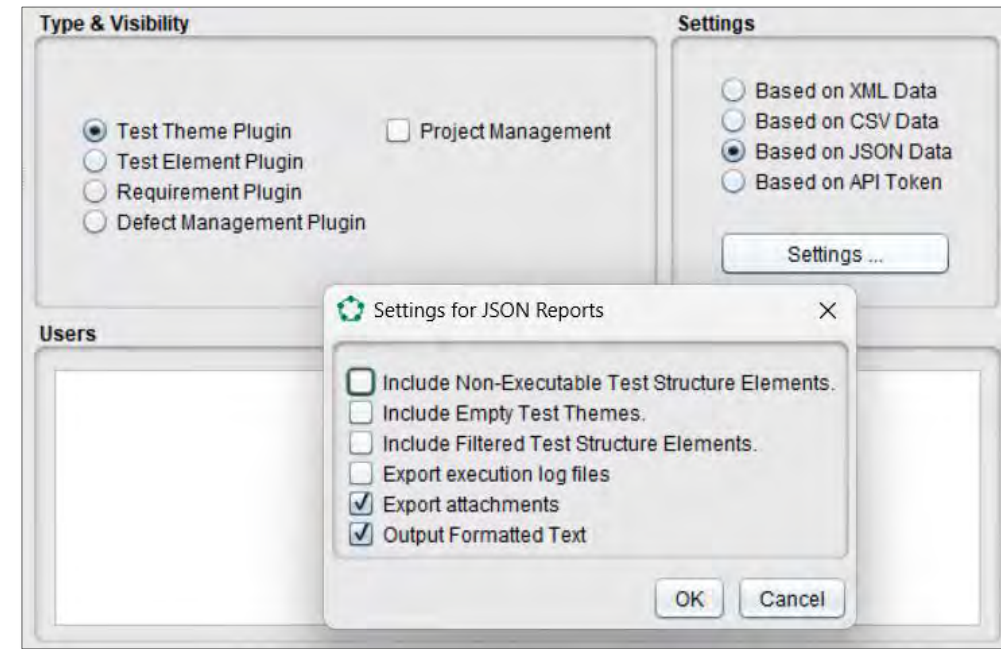
- Can also be used with TestBench version 3.0.x.

JSON as a New Export/Import Format

- JSON is another export and import format for TestBench.
- Web iTORX and other new integrations use this format.
- No log files are generated during the test run; instead, the test result data is added to the report.
- TestBench therefore provides a new JSON-based plugin type based on a JSON full report.



JSON full report in the report menu



Plugin definition based on JSON data

New Authentication via JSON Web Token

- A JSON Web Token (JWT) is an access token standardised according to [RFC 7519](#)
- General structure: `xxxxx.yyyyy.zzzzz`
 - Header xxxxx: **Metadata**, such as signature algorithm and token type
 - Payload yyyyy: **Operational data** (known as claims), such as resource, user, target system, expiry date, etc.
 - Signature zzzzz: **Signature** over header and payload, generated with secret or private key
- Signature prevents manipulation

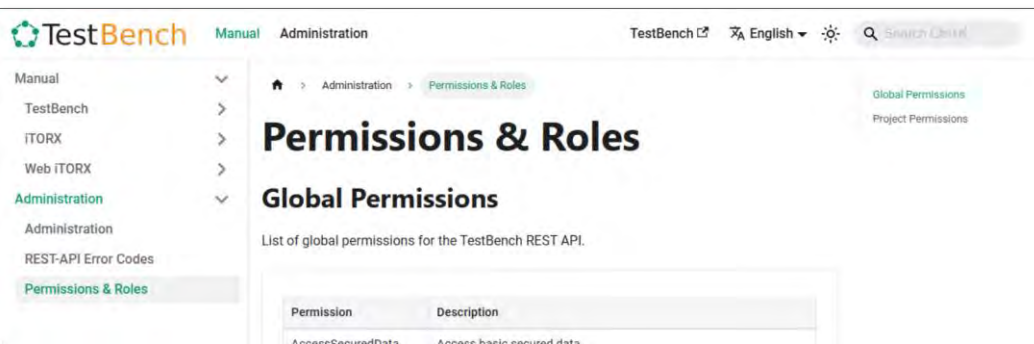
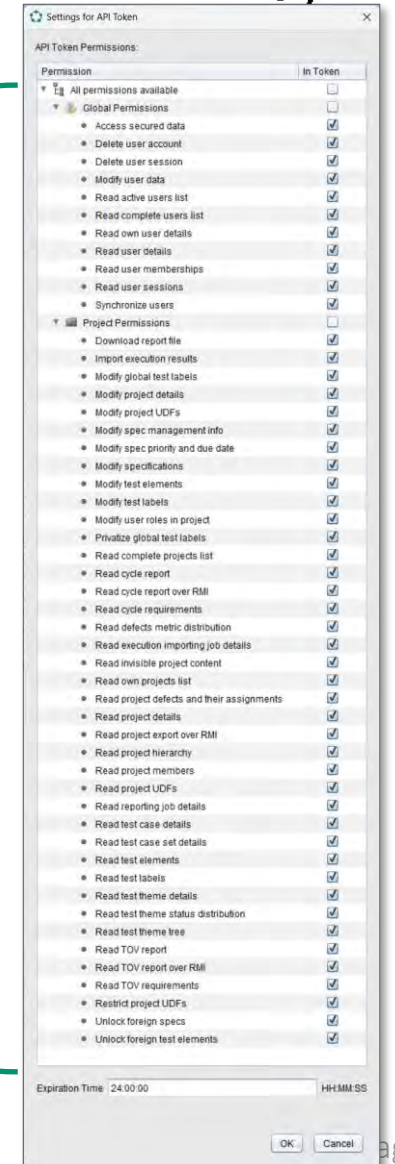
A JWT is a signed but unencrypted data packet consisting of a header, payload and signature that securely transmits identity claims.

Benefits of JSON Web Token

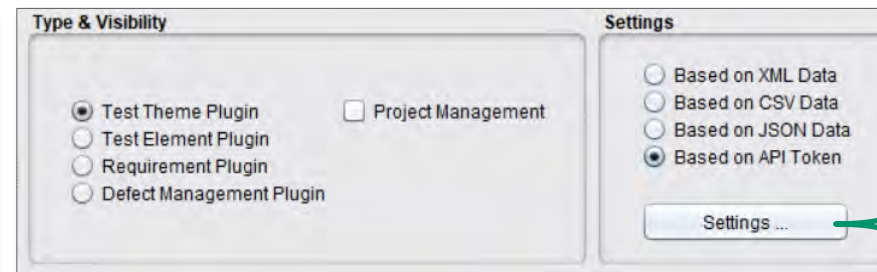
- Enables secure data exchange between TestBench and other systems.
 - Contains **all important information** for accessing data on the TestBench server.
 - Determines who can access which data and for how long.
 - No exchange of authentication data, such as login and password, is necessary.
-
- Web iTORX is connected to TestBench via JWT.
 - Web iTORX can be accessed via a URL that incorporates the JWT and loads the data stored in the token from the TestBench server.
This allows integration into systems such as Jira or Jenkins via a link with the corresponding URL.

Plugin for JSON Web Token

- To enable the use of JSON Web Tokens, TestBench provides a new plugin type. : **Based on API-Token**
- The plugin definition specifies which rights the token passes on.
- Connection to external systems can be established without sharing highly privileged access data.



Description of possible permissions and roles



Plugin definition based on an API token with a list of possible permissions

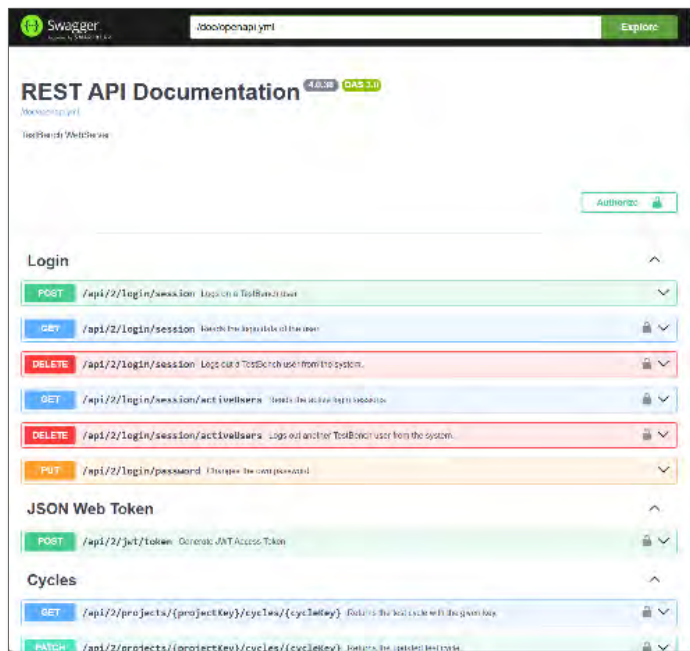
Plugins Know the Call Context

- The interface of all plugins has been expanded to include information describing the context of the call., e.g.:
 - Name and type of plugin
 - Key of test object version and test cycle
 - ID of the node from which the call is made

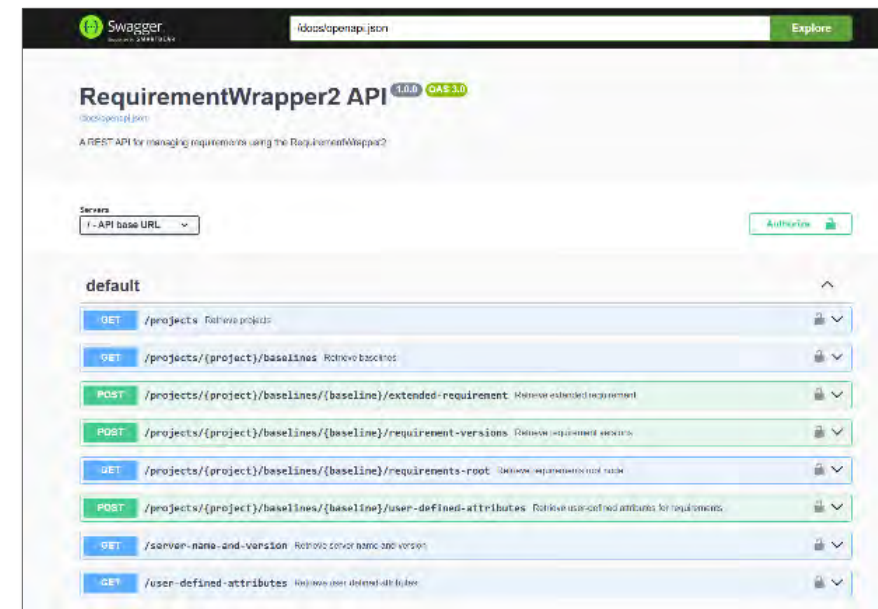
Visible context	Execution context
Plugin name: <input type="text" value="JSON Web Token"/>	Project key: <input type="text" value="3"/>
User: <input type="text" value="tt-admin"/>	TOV key: <input type="text" value="13"/>
Encoding: <input type="text" value="—"/>	Cycle key: <input type="text" value="43"/>
Tree type: <input type="text" value="TESTTHEMES"/>	Tree root ID: <input type="text" value="ITB-TC-337"/>
	Tree root key: <input type="text" value="4611686020000020668"/>
	Element type: <input type="text" value="TESTCASESET"/>
	Tree type: <input type="text" value="TESTTHEMES"/>
	Data type: <input type="text" value="API"/>
	Filtering: <input type="text" value="—"/>

Extended REST-API

- The application server layer now consists of three servers. The third, new server provides a greatly expanded REST API.
- The new services for connecting requirements and defect management systems and the AI proxy also provide new REST APIs.



REST API of the new server



REST API of the requirements management proxy





Show Password in the Login Window







In the login screen, the password can be temporarily displayed by clicking on the eye icon in the password field:



Keyboard Shortcuts for creating elements

Test structure elements(*) can now also be created using keyboard shortcuts:

 New <u>T</u> est Theme	Ctrl+Alt+T
 New Test <u>C</u> ase Set	Ctrl+Alt+S
 Unlock (Finish editing)	Ctrl+D
 Recursive lock	Ctrl+Shift+S

 <u>N</u> ew Subdivision	Ctrl+Alt+S
 New <u>D</u> ata Type	Ctrl+Alt+D
 New Reference Data Type	Ctrl+Alt+R
 New <u>K</u> eyword	Ctrl+Alt+K
 New <u>C</u> ondition	Ctrl+Alt+B
 Unlock (Finish editing)	Ctrl+D

(*) Test structure elements are test subjects, test case sets, subdivisions, keywords, data types and conditions.



Time for

Questions!

CONTACT

imbus AG

Kleinseebacher Str. 9
91096 Möhrendorf
GERMANY



info@testbench.com



www.testbench.com



+49 9131 7518-0

