DE10000 USER MANUAL

EDITOR

TestPlanner

testplanner

DEICO

Document number: SBL-0038 Rev.1 2025

Contents

INTRODUCTION

Overview

TestPlanner Editor is the key software of the TestPlanner ecosystem. It enables users to create, modify, run and debug tests in a graphical user interface. With open source OpenTAP core, it is simple to create plugins for users' needs, in addition to provided packages.

TestPlanner supports debugging, report generation, variable and output binding and much more.

Target Audience

TestPlanner is designed for test engineers, developers, operators and anyone who wish to run their tests in a test automation environment.

Prerequisite Knowledge

This document is designed to serve as a guide for using TestPlanner Editor. To follow, an installation of TestPlanner Editor is required.

GETTING STARTED

TestPlanner installer is used to install TestPlanner in supported systems. It automatically installs TestPlanner, OpenTAP, required packages and DEICO License Manager.

Editor Components

TestPlanner Editor uses 2 essential packages in its core:

- ⇨ **TestPlanner:** This is the base API of the TestPlanner software. It contains the core functionality.
- ⇨ **TestPlanner.Wpf:** This is the GUI-specific API. It is used to create UI plugins for TestPlanner Editor.

Note "TestPlanner and TestPlanner.WPF User Guide" should be referenced for using these APIs.

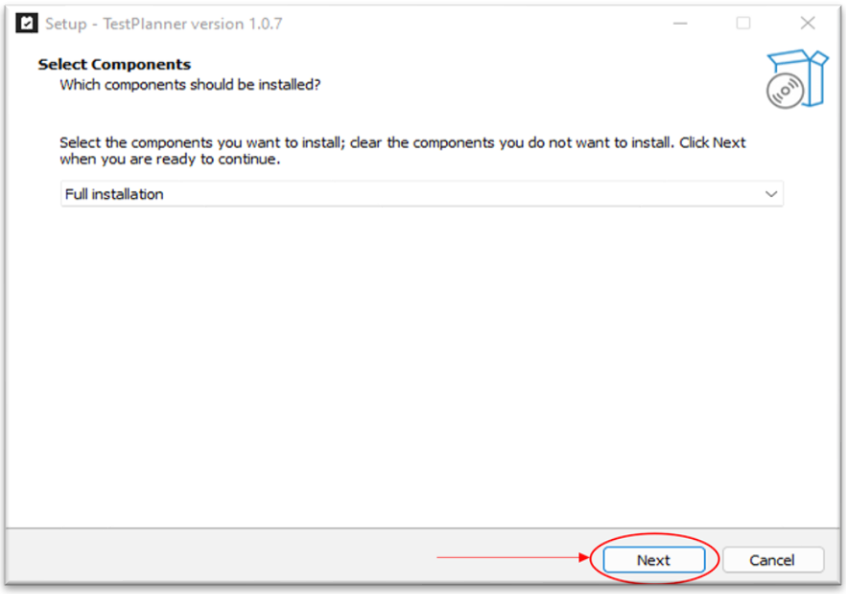
Editor comes with built-in plugins, such as variables and expression test step.

Installation and Setup Instructions

TestPlanner Editor works with Windows 10 and later versions, 64 bit only. To install, the instructions below should be followed.

TestPlanner Installation

1. Double click on the installer to start the installation process.
2. When asked for approval, click on **Yes**.
3. Click on **Next**.



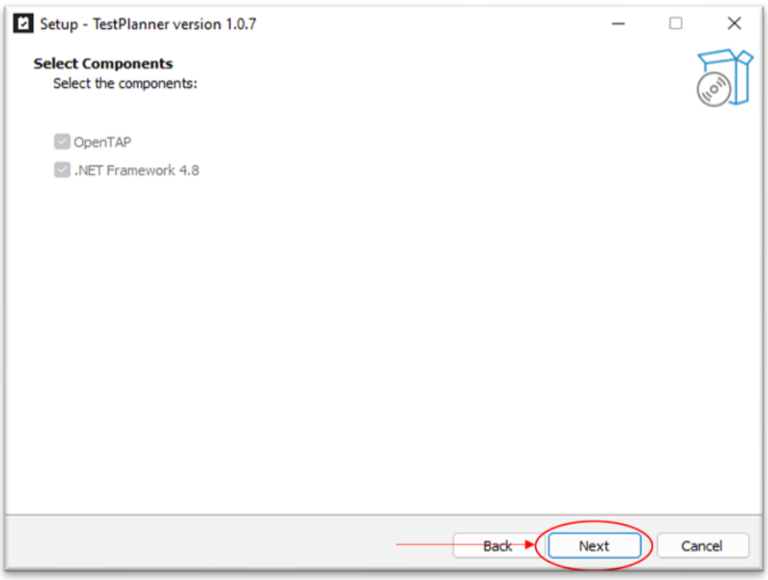
Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Full installation

Figure 1: TestPlanner Installation – Setup Screen

4. Click on **Next**.

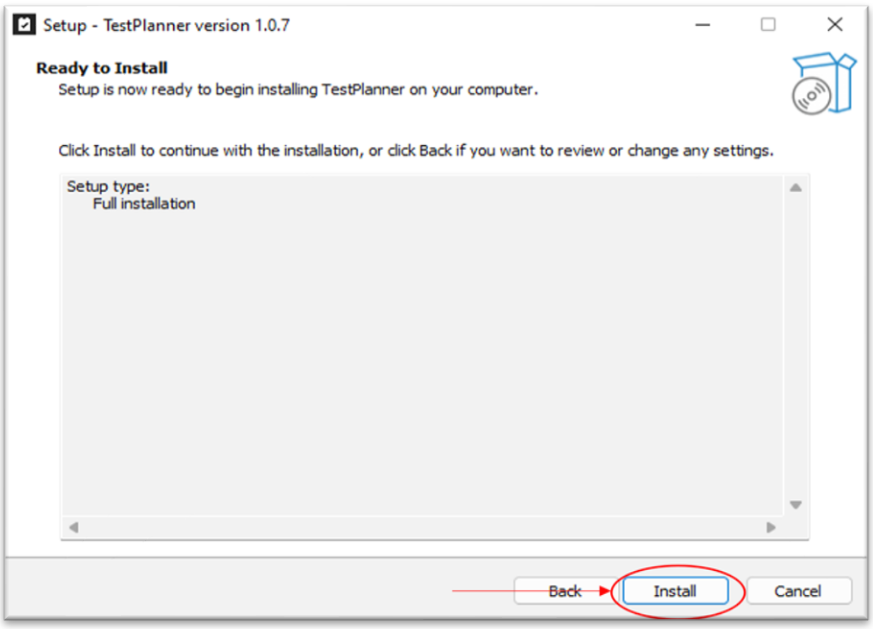


Select Components
Select the components:

☑ OpenTAP
☑ .NET Framework 4.8

Figure 2: TestPlanner Installation – Component Selection

5. Click on **Install**.


Setup - TestPlanner version 1.0.7

Ready to Install

Setup is now ready to begin installing TestPlanner on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Setup type:
    Full installation

Figure 3: TestPlanner Installation – Starting the Installation

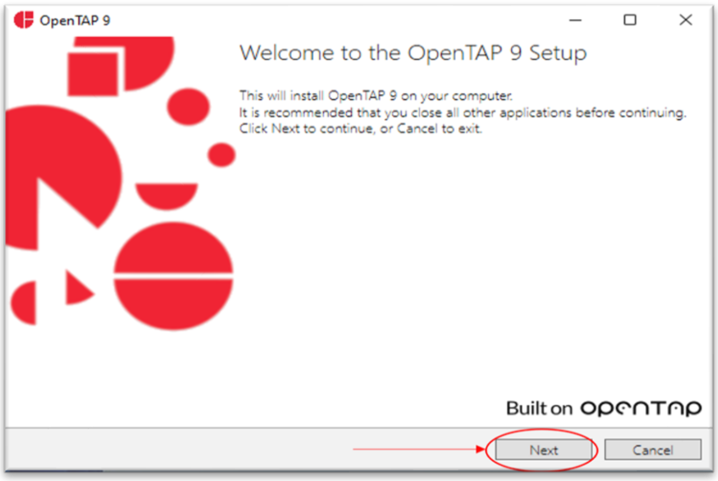
6. Wait until the progress bar is full.

Note If OpenTAP has not been installed previously, the OpenTAP installation window will be displayed.

OpenTAP Installation

Note If OpenTAP has already been installed, skip to License Manager Installation.

1. Click on **Next**.

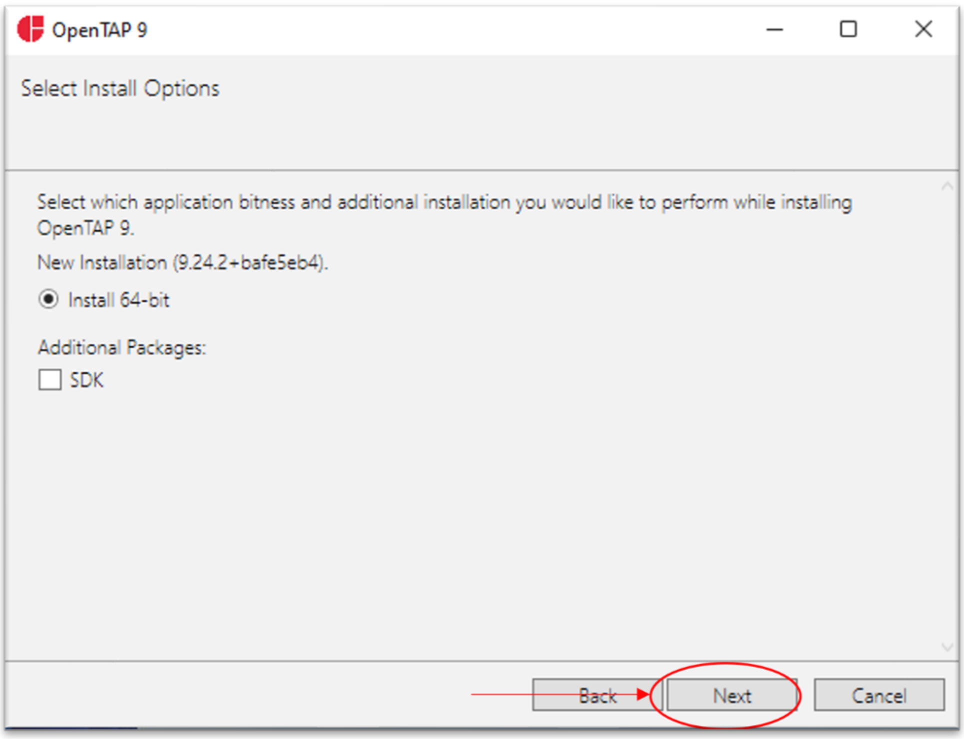

OpenTAP 9

Welcome to the OpenTAP 9 Setup

This will install OpenTAP 9 on your computer.
It is recommended that you close all other applications before continuing.
Click Next to continue, or Cancel to exit.

Figure 4: OpenTAP Installation – Setup Screen

2. Click on **Next**.



OpenTAP 9

Select Install Options

Select which application bitness and additional installation you would like to perform while installing OpenTAP 9.

New Installation (9.24.2+bafe5eb4).

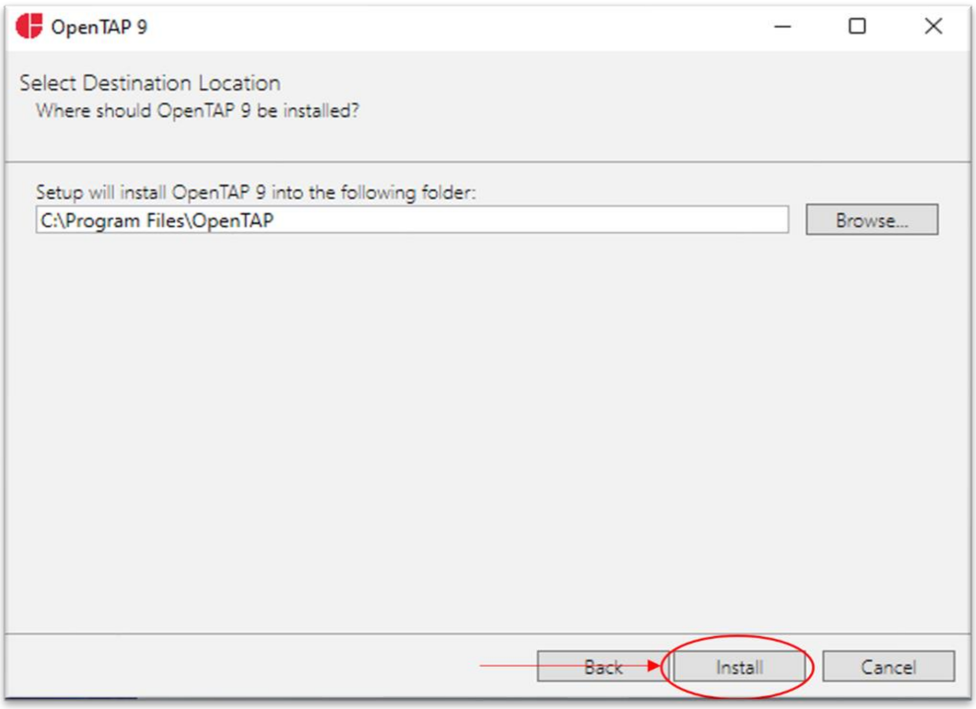
(●) Install 64-bit

Additional Packages:

☐ SDK

Figure 5: OpenTAP Installation – Application Bitness Selection

3. Click on **Install**.



OpenTAP 9

Select Destination Location

Where should OpenTAP 9 be installed?

Setup will install OpenTAP 9 into the following folder:

C:\Program Files\OpenTAP

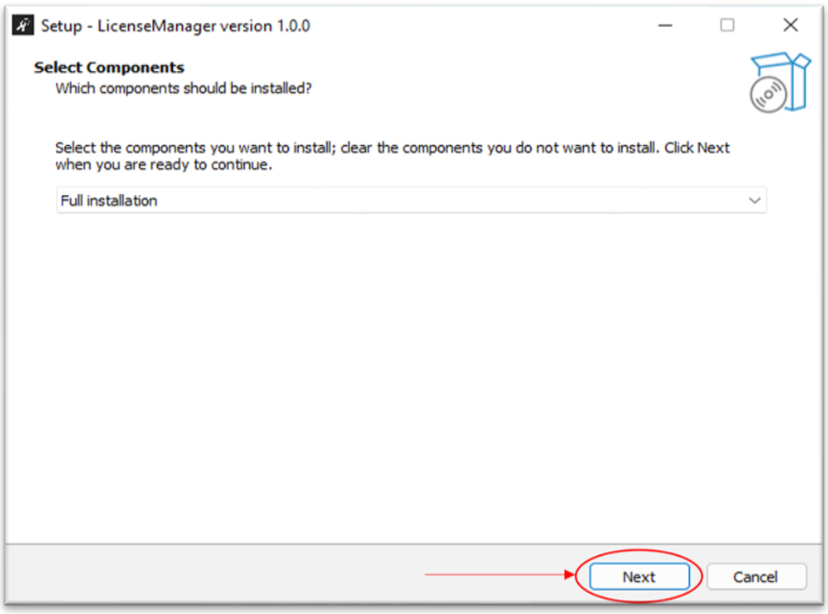
Figure 6: OpenTAP Installation – Starting the Installation

4. Click on **Finish** to end the installation.

License Manager Installation

Note If License Manager has already been installed, skip to Plugin Installations.

1. Click on **Next**.



Setup - LicenseManager version 1.0.0

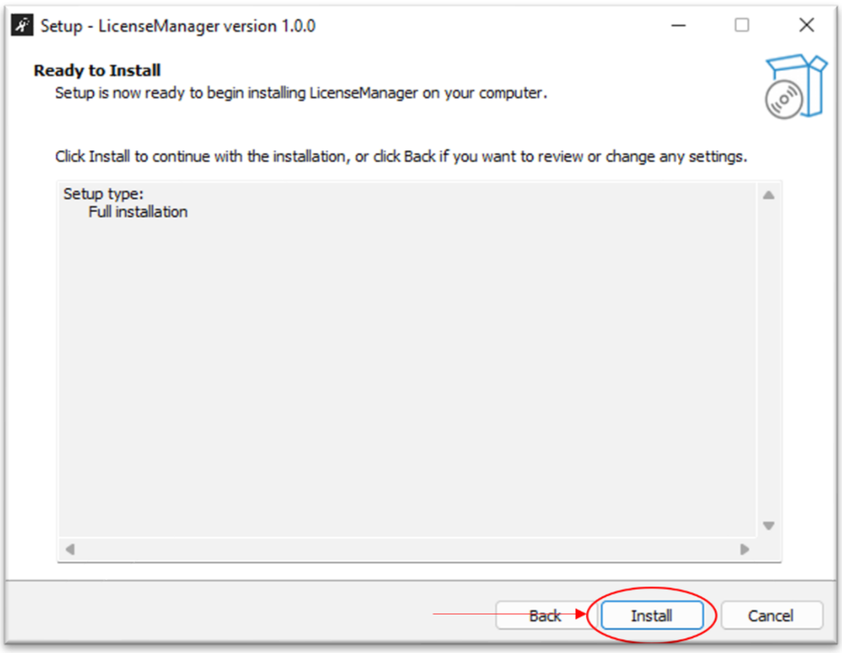
Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Full installation

Figure 7: License Manager Installation – Setup Screen

2. Click on **Install**.



Setup - LicenseManager version 1.0.0

Ready to Install
Setup is now ready to begin installing LicenseManager on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Setup type:
Full installation

Figure 8: License Manager Installation – Starting the Installation

3. Click on **Finish** to end the installation.

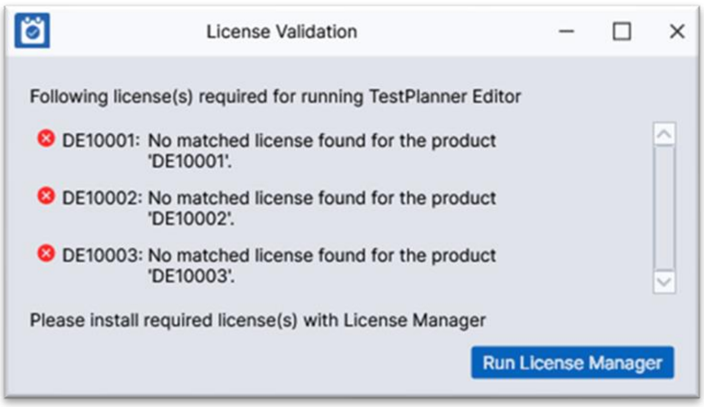
Plugin Installations

1. Wait until installations are completed.
2. Click on **Finish** to end the installation and reboot. **TestPlanner installation is completed**.

License Registration

Note When opening TestPlanner for the first time, if the licensing has not been completed, the License Validation screen may appear.

1. Click on **Run License Manager** to open the License Manager, or DEICO License Manager can be manually opened from your programs.



License Validation

Following license(s) required for running TestPlanner Editor

DE10001: No matched license found for the product 'DE10001'.

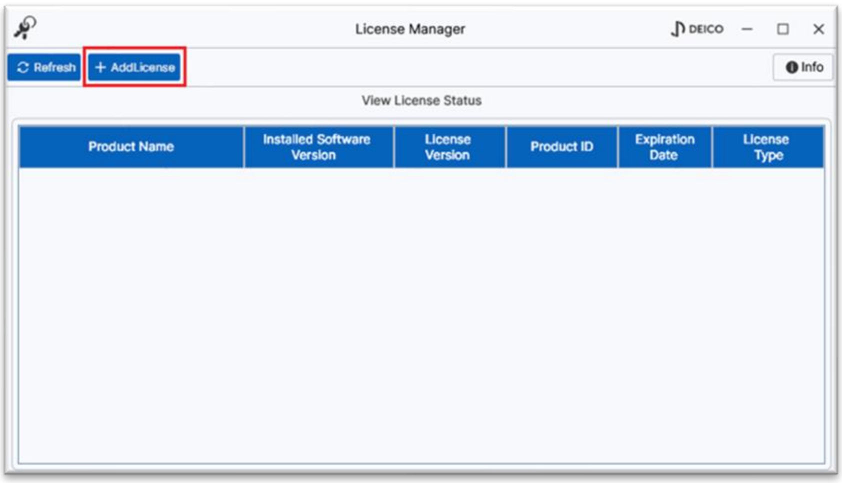
DE10002: No matched license found for the product 'DE10002'.

DE10003: No matched license found for the product 'DE10003'.

Please install required license(s) with License Manager

Figure 9: License Validation

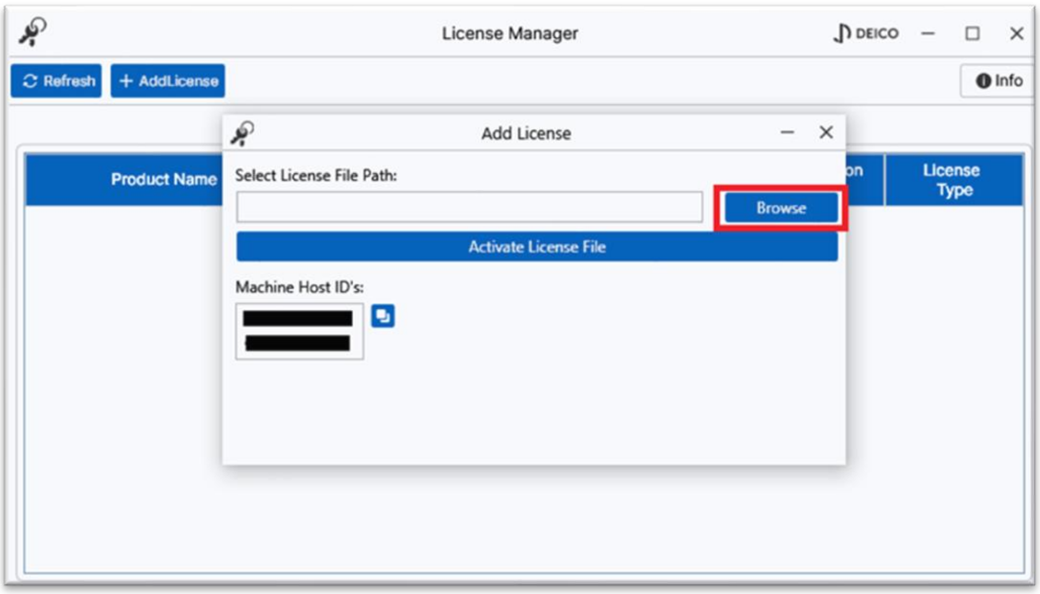
2. Once the License Manager is open, click on **Add License** in the upper-left corner.



Product Name	Installed Software Version	License Version	Product ID	Expiration Date	License Type

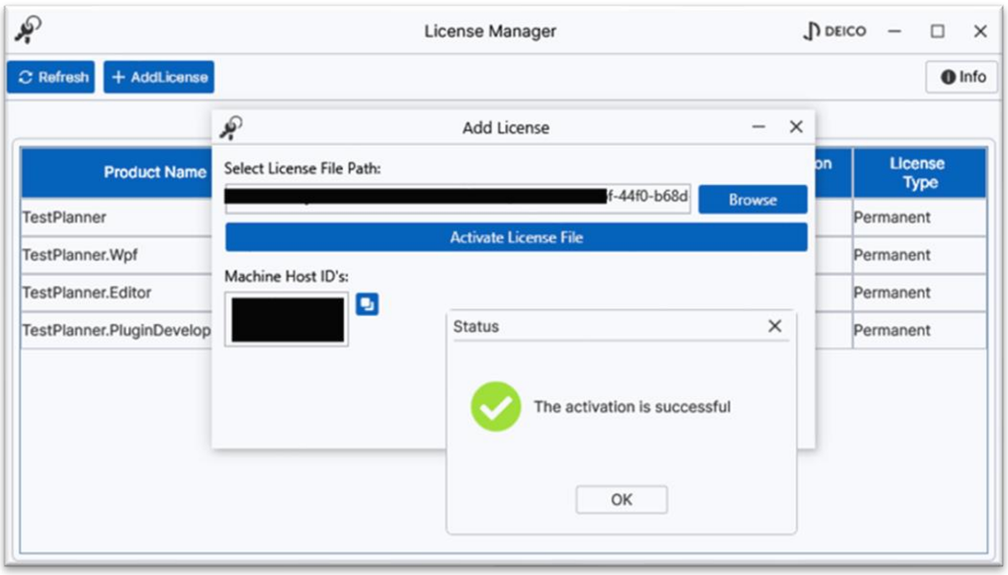
Figure 10: Adding License

3. In the next screen, use the **Browse** button to select the provided license file, then click on **Activate License File**. A message stating "activation is successful" will be displayed, and the main page of the License Manager will show the corresponding four licenses.



[redacted]

Figure 11: Browsing the License File



[redacted]f-44f0-b68d

[redacted]

Product Name	License Type
TestPlanner	Permanent
TestPlanner.Wpf	Permanent
TestPlanner.Editor	Permanent
TestPlanner.PluginDevelop	Permanent

The activation is successful

Figure 12: Activating the License

4. After this process, TestPlanner will be fully licensed and ready to use.

USER INTERFACE COMPONENTS

TestPlanner Editor comes with several panels, each serving a different purpose.

Menu Bar

Menu bar includes common functionality for TestPlanner, including save/load options, settings, debugging options, and views.

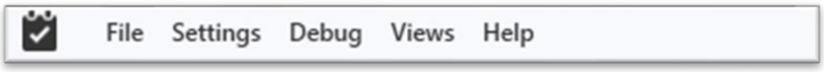


Figure 13: Menu Bar

File Menu

File Menu can be used for file related operations and user configuration.

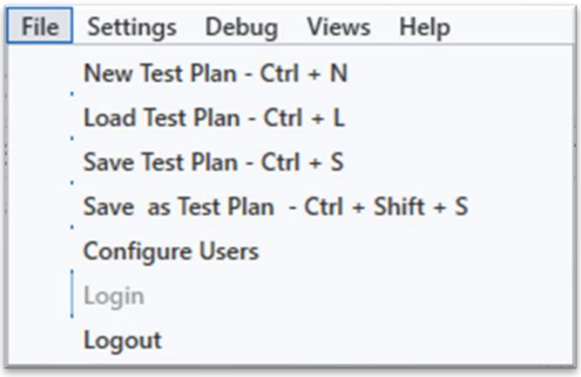

File
Settings
Debug
Views
Help
New Test Plan - Ctrl + N
Load Test Plan - Ctrl + L
Save Test Plan - Ctrl + S
Save as Test Plan - Ctrl + Shift + S
Configure Users
Login
Logout

Figure 14: File Menu

- ⇨ **New Test Plan:** Creates a new test plan.
- ⇨ **Load Test Plan:** Loads a test plan from file.
- ⇨ **Save Test Plan:** Saves the test plan to file.
- ⇨ **Save as Test Plan:** Saves the test plan as a new file.
- ⇨ **Configure Users:** Opens user configuration view.
- ⇨ **Login:** Logs in as a user.
- ⇨ **Logout:** Logs out from the current user.

Note User configuration options are discussed in User Configuration.

Settings Menu

Settings Menu can be used for interacting with settings.

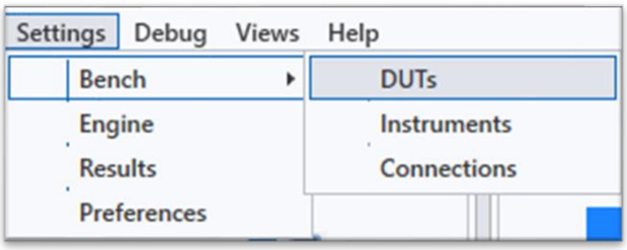


Figure 15: Settings Menu

- ➪ **Bench:** Change DUT (device under test), instrument and connection settings.
- ➪ **Engine:** Change engine settings such as log path and operator name.
- ➪ **Results:** Change result listener settings, such as report plugins.
- ➪ **Preferences:** Change user preferences, such as theme and font sizes.

Debug Menu

Debug Menu can be used for debugging options.

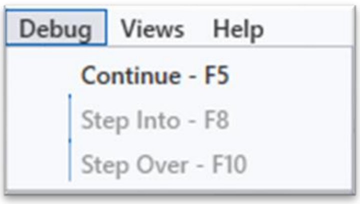


Figure 16: Debug Menu

- ➪ **Continue:** Continues test execution.
- ➪ **Step Into:** Moves to the next test step, including any child test steps.
- ➪ **Step Over:** Moves to the next step, without stopping at a child test step.

Views Menu

Views Menu can be used for interacting with panel layout.

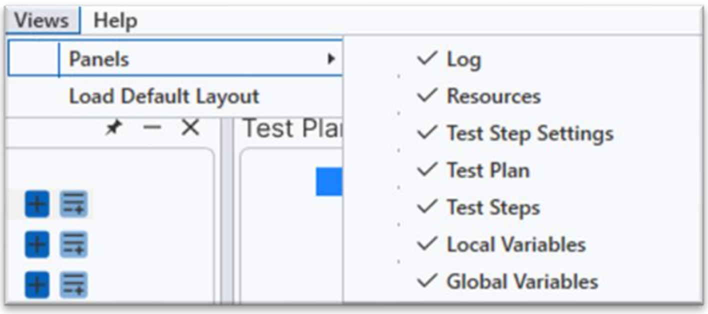


Figure 17: Views Menu

- ➪ **Panels:** Toggles visibility of each panel in TestPlanner.
- ➪ **Load Default Layout:** Resets the panel layout to default.

About Menu

About Menu can be used to view software information, such as the software version.

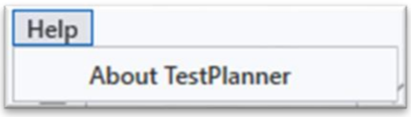


Figure 18: About Menu

Ribbon

Ribbon provides controls for the operations on the test plan. Below, each of their functionality, from left to right, are described.

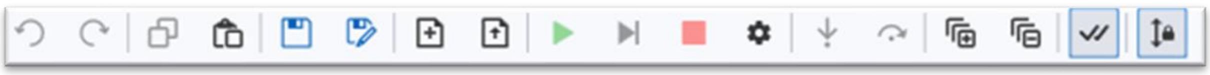


Figure 19: Ribbon

- ⇨ **Undo:** Undoes an action.
- ⇨ **Redo:** Redoes an action.
- ⇨ **Copy:** Copies selected test steps in Test Plan panel.
- ⇨ **Paste:** Pastes copied test steps in Test Plan panel.
- ⇨ **Save:** Saves the test plan.
- ⇨ **Save as:** Saves the test plan as a new file.
- ⇨ **New Test Plan:** Creates a new test plan.
- ⇨ **Load Test Plan:** Loads a test plan from file.
- ⇨ **Run:** Runs the test plan.
- ⇨ **Pause:** Pauses the test plan.
- ⇨ **Abort:** Aborts the test plan.
- ⇨ **Test Plan Settings:** Opens the test plan settings.
- ⇨ **Step Into:** Moves to the next text step, including any child test steps.
- ⇨ **Step Over:** Moves to the next text step, without stopping at a child test step.
- ⇨ **Expand All:** Makes all child test steps visible.
- ⇨ **Collapse All:** Makes all child test steps hidden.
- ⇨ **Enable All:** Makes all test steps enabled or disabled.
- ⇨ **Auto Scroll:** Follows the executed test step.

Test Steps Panel

Test Steps Panel displays the test steps that can be added to the test plan. In addition to built-in ones, the test steps from installed plugins also appear here.

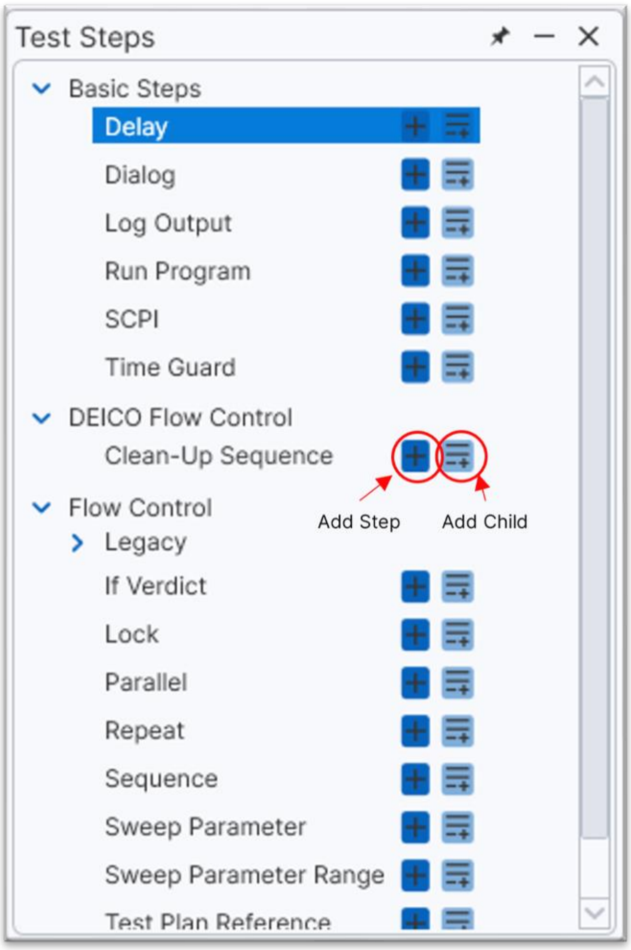


Figure 20: Test Steps Panel

- **To add a test step:** Click the **Add Step** button next to a test step.
- **To add a test step as a child for a "Sequence":** Click the **Add Child** button.

Note This is permitted only when a "Sequence" test step is selected in the user interface.

Note A test step can also be dragged and dropped into the Test Plan Panel for insertion.

Test Plan Panel

Test Plan Panel displays the test steps added to the test plan. For each test step, name, verdict, and the duration it took in the last text execution can be seen.

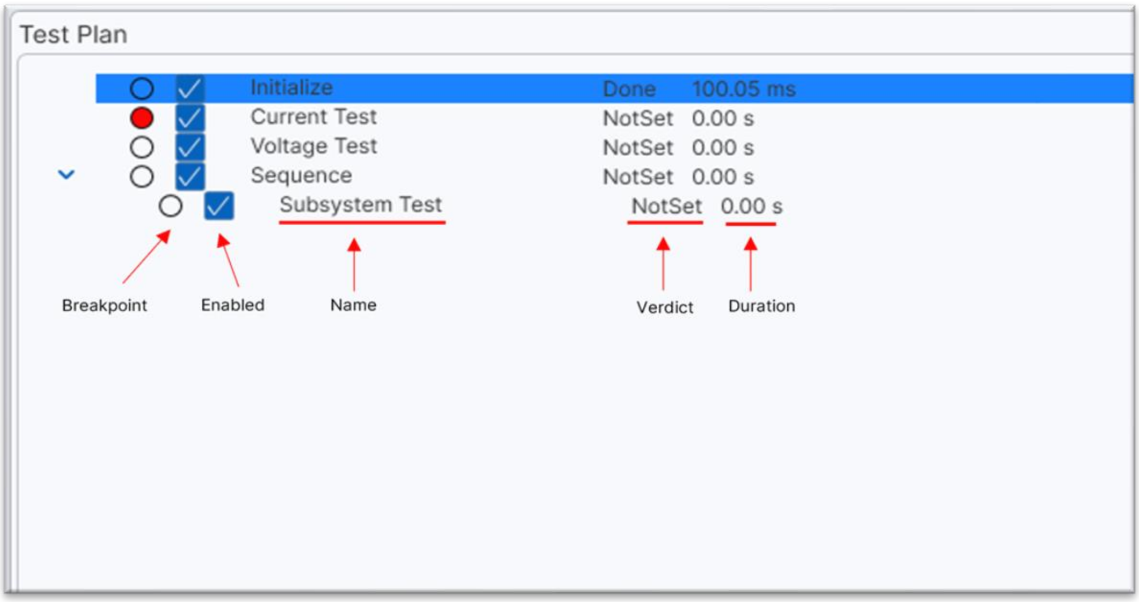

Test Plan
Initialize Done 100.05 ms
Current Test NotSet 0.00 s
Voltage Test NotSet 0.00 s
Sequence NotSet 0.00 s
Subsystem Test NotSet 0.00 s
Breakpoint
Enabled
Name
Verdict
Duration

Figure 21: Test Plan Panel

- ⇨ **To put a breakpoint to a test step:** Click on the curcle. The circle becomes red, denoting a break.
- ⇨ **To enable or disable a test step:** Click on the tick.

Note After adding test steps from the Test Steps Panel, their order can be changed by dragging them in this panel.

Note To make a test step a child test step, it should be dropped on another test step.

Test Step Context Menu

When right clicked on a test step, the following context menu is displayed.

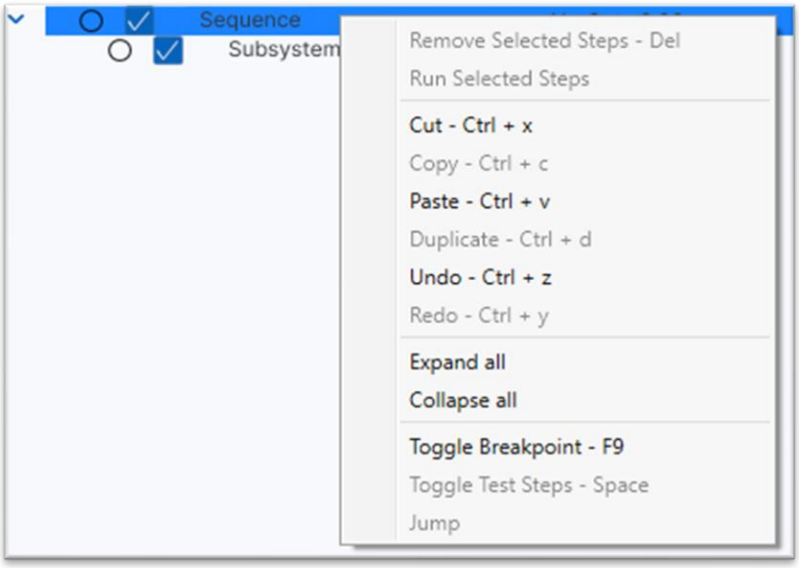


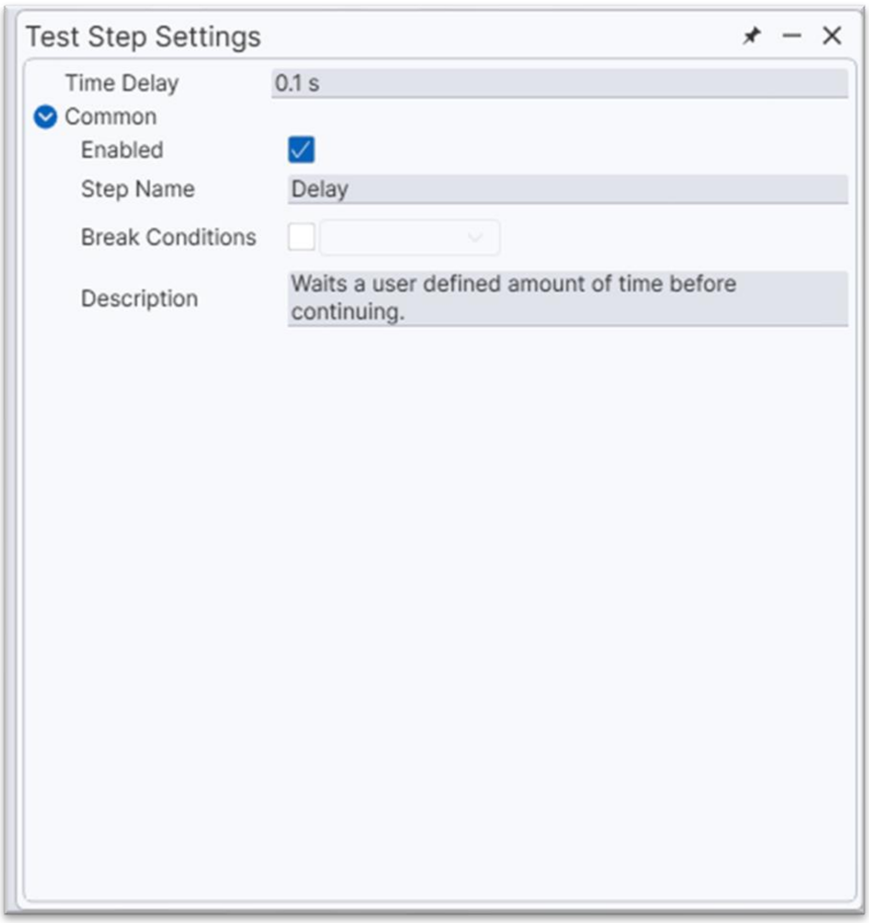
Figure 22: Test Step Context Menu

- **Remove Selected Steps:** Removes the selected test steps from the test plan.
- **Run Selected Steps:** Runs the selected test steps, disregarding the enabled steps.
- **Cut:** Cuts the selected steps.
- **Copy:** Copies the selected test steps.
- **Paste:** Adds the selected test steps.
- **Duplicate:** Creates a duplicate of the selected test step.
- **Undo:** Undoes the last operation.
- **Redo:** Redoes the last operation.
- **Expand All:** Makes all child test steps visible.
- **Collapse All:** Makes all child test steps hidden.
- **Toggle Breakpoint:** Sets or resets the breakpoint for the test step.
- **Toggle Test Steps:** Enables or disables the test step.
- **Jump:** Jumps to the test step from the paused step.

Test Step Settings Panel

The test step configuration can be changed from this panel.

Note This view is automatically updated according to the selected test steps in Test Plan Panel.

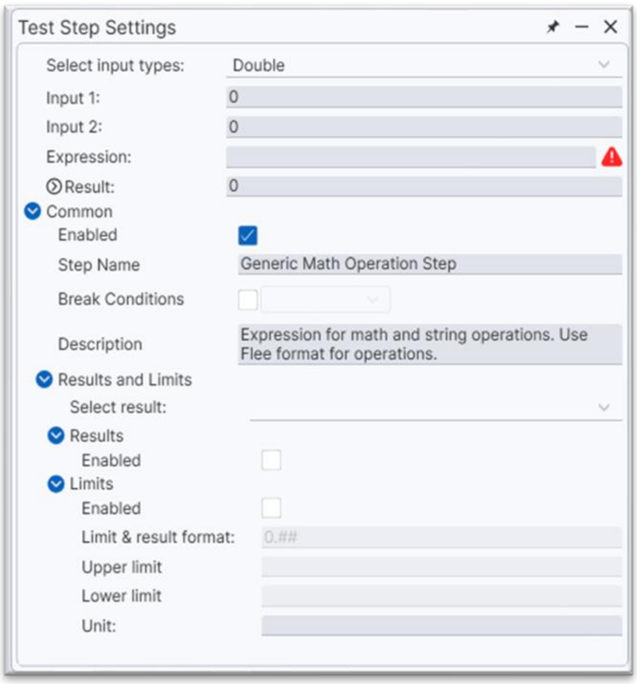


Test Step Settings

Time Delay	0.1 s
Common	
Enabled	✓
Step Name	Delay
Break Conditions	
Description	Waits a user defined amount of time before continuing.

Figure 23: Test Step Settings Panel

Note If there is a property that was not set correctly, a red alert icon will emerge next to it, preventing correct testing procedure. If the user hovers on the icon, a tooltip with an error explanation will be shown.



Select input types:	Double
Input 1:	0
Input 2:	0
Expression:	
Result:	0
Common	
Enabled	✓
Step Name	Generic Math Operation Step
Break Conditions	
Description	Expression for math and string operations. Use Flee format for operations.
Results and Limits	
Select result:	
Results	
Enabled	
Limits	
Enabled	
Limit & result format:	0.##
Upper limit	
Lower limit	
Unit:	

Figure 24: Error Explanation

Resources Panel

New resources can be added or modified in this panel. The connections, DUTs, instruments, and results can be worked on.

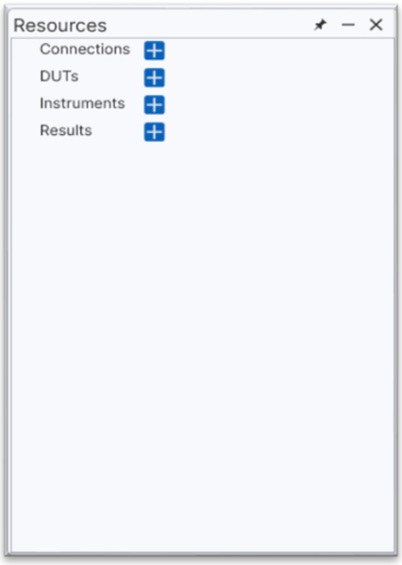
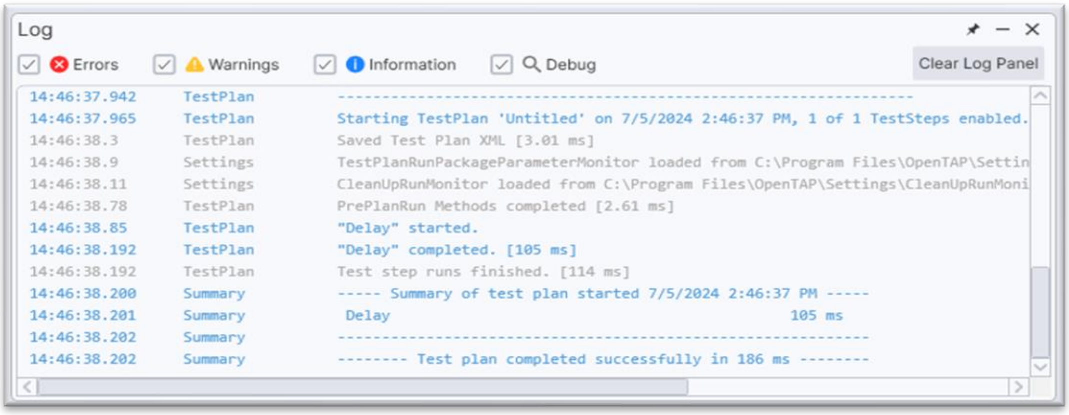


Figure 25: Resources Panel

Log Panel

Logs can be viewed in this panel.

Note By default, logs are saved in **C: Files** directory. The saving path can be changed by navigating Settings > Engine > General > Log Path in Menu Bar.



```
14:46:37.942    TestPlan     -----------------------------------------------------------------
14:46:37.965    TestPlan     Starting TestPlan 'Untitled' on 7/5/2024 2:46:37 PM, 1 of 1 TestSteps enabled.
14:46:38.3      TestPlan     Saved Test Plan XML [3.01 ms]
14:46:38.9      Settings     TestPlanRunPackageParameterMonitor loaded from C:\Program Files\OpenTAP\Settin
14:46:38.11     Settings     CleanUpRunMonitor loaded from C:\Program Files\OpenTAP\Settings\CleanUpRunMoni
14:46:38.78     TestPlan     PrePlanRun Methods completed [2.61 ms]
14:46:38.85     TestPlan     "Delay" started.
14:46:38.192    TestPlan     "Delay" completed. [105 ms]
14:46:38.192    TestPlan     Test step runs finished. [114 ms]
14:46:38.200    Summary      ----- Summary of test plan started 7/5/2024 2:46:37 PM -----
14:46:38.201    Summary       Delay                                              105 ms
14:46:38.202    Summary      ------------------------------------------------------------
14:46:38.202    Summary      -------- Test plan completed successfully in 186 ms --------
```

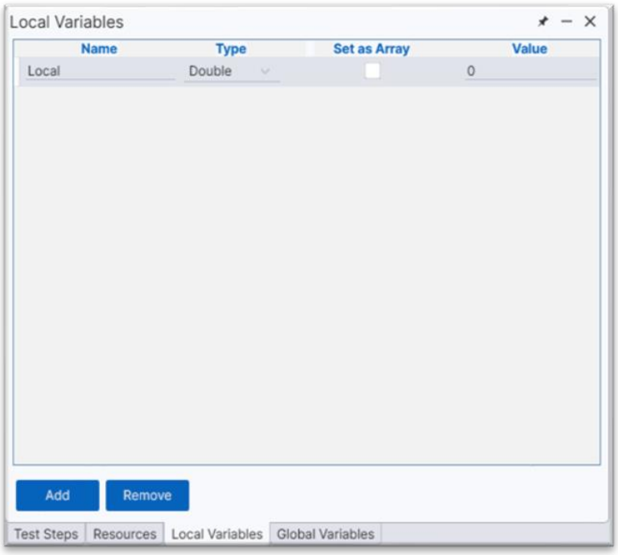
Figure 26: Log Panel

⇨ **To filter logs based on their severity:** Use the buttons to the left.
⇨ **To clear logs:** Click on the **Clear Log Panel** button to the right.

Local Variables Panel

The local variables can be viewed and changed in this panel.

Note The local variables are parameters that are tied to the test plan. TestPlanner saves these variables along with the test plan, allowing the users to use portable and reusable parameters.



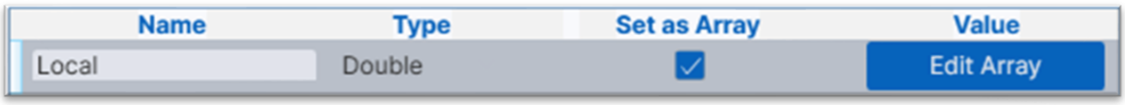
Name	Type	Set as Array	Value
Local	Double	☐	0

Figure 27: Local Variables Panel

- ⇨ **To add a new local variable:** Click on the **Add** button in the lower left corner.
- ⇨ **To remove a local variable:** Select the variable and click on the **Remove** button.
- ⇨ **To change the properties of a variable:** Click on the textbox on the left (to change the name of a variable).
- ⇨ **To change the type of the variable:** Use the selection menu under the **Type** header.
- ⇨ **To make the variable an array of specified type:** Use the **Set as Array** checkbox.
- ⇨ **To change the value of the variable:** Use the **Value** field.

Arrays

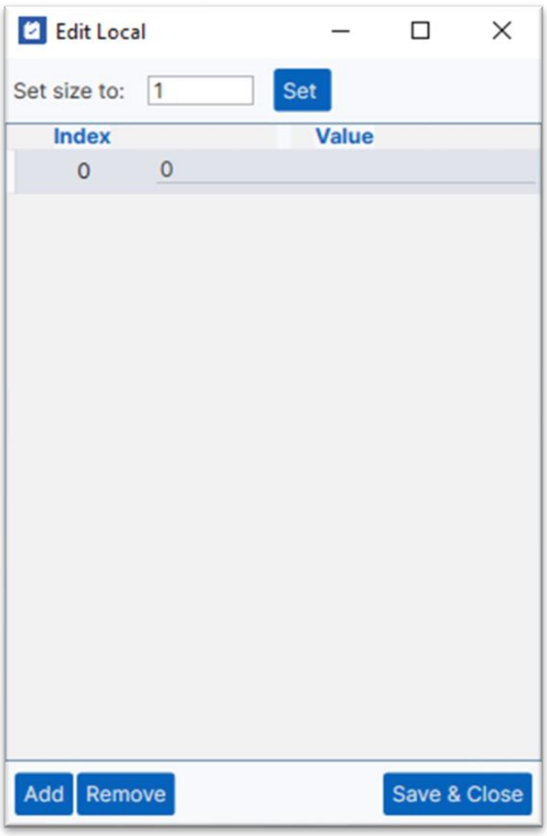
When a variable is set as an array, a button will replace the **Value** field.



Name	Type	Set as Array	Value
Local	Double	✓	Edit Array

Figure 28: Array Button

The elements of the array can be set by clicking on the **Edit Array** button.



Set size to: 1

Index	Value
0	0

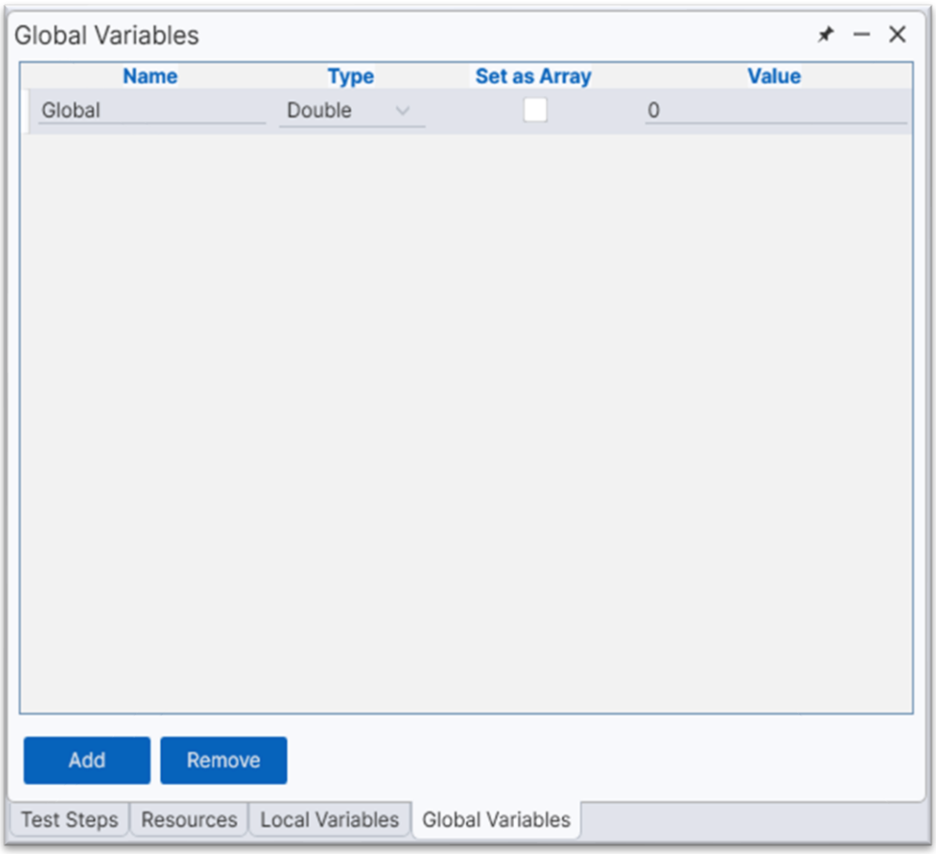
Figure 29: Array Editing

- ⇨ The length of the array can be changed from the upper textbox.
- ⇨ Any value in the array can be changed using the **Value** field.
- ⇨ The **Add** and **Remove** buttons can be used to increase or decrease the array length.
- ⇨ **Save & Close** button should be clicked on to complete the changes.

Global Variables Panel

The global variables can be viewed and changed in this panel.

Note The global variables are parameters that are tied to TestPlanner. They can be used for multiple test plans in the same computer. However, these variables will not be carried over if the test plan is loaded in another computer.



Name	Type	Set as Array	Value
Global	Double	☐	0

Figure 30: Global Variables Panel

Note The interaction with variables is similar to Local Variables Panel.

HOW TO USE TestPlanner EDITOR?

Creating a Test Plan

Note Some example .tapplans are generated through installer. These examples can be accessed in the directory as **C:\Program Files\OpenTAP\Sample Test Plans**.

To create a test plan in TestPlanner Editor:

1. Open TestPlanner Editor.
2. If the authentication window is displayed, click on **Login**.
3. From the menu, navigate **File > New Test Plan** in Menu Bar.

4. In the Test Steps Panel on the left, expand a group by clicking on the expander.

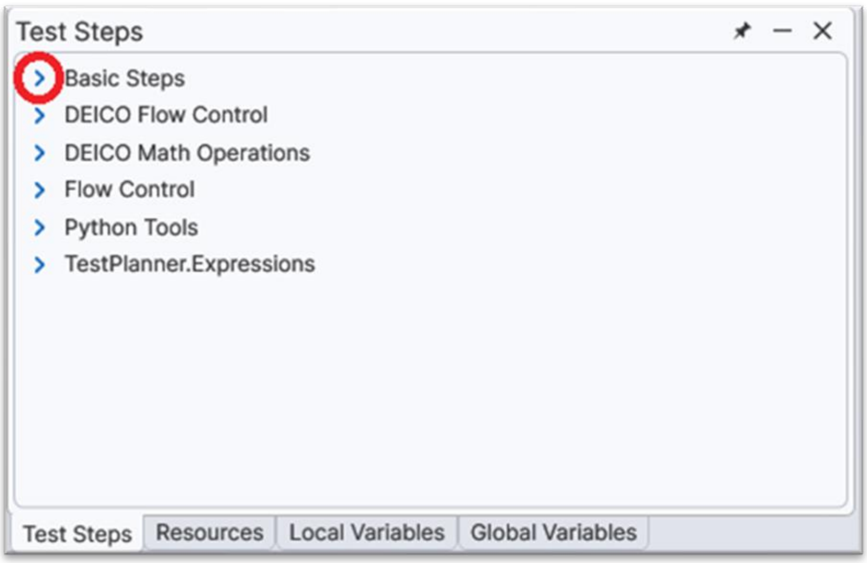


Figure 31: Test Step Expander

5. Click on the button with a plus icon to add the step to the test plan.
6. Click on the **Run** button in the ribbon.

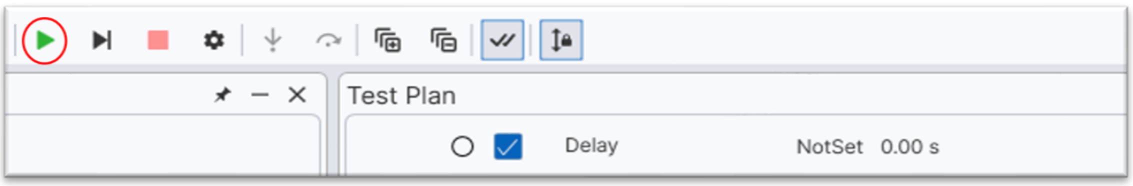


Figure 32: Run Button

Debugging a Test Plan

To debug a test plan in TestPlanner Editor:

1. Create a test plan with multiple test steps.
2. Put a breakpoint to a test step.

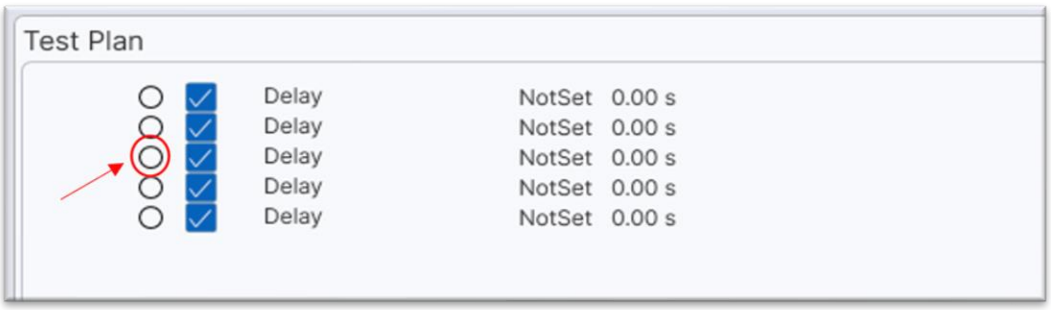
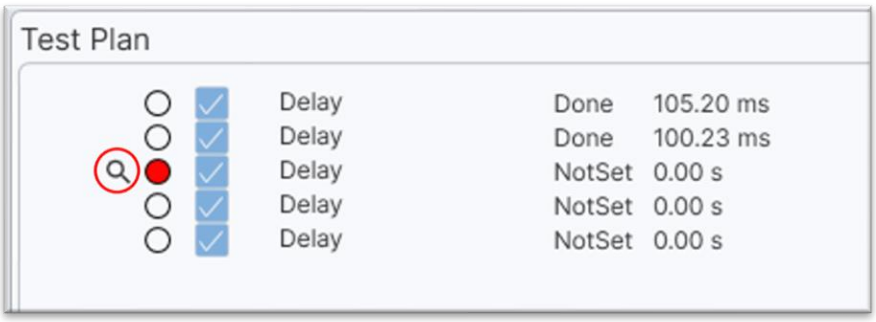


Figure 33: Breakpoint

3. Click on the **Run** button in the ribbon. Notice the execution will stop at the test step with the breakpoint, with a magnifying glass icon on the left.



Test Plan

○	☑	Delay	Done	105.20 ms
○	☑	Delay	Done	100.23 ms
●	☑	Delay	NotSet	0.00 s
○	☑	Delay	NotSet	0.00 s
○	☑	Delay	NotSet	0.00 s

Figure 34: Debug Icon

4. Right click on the next test step, select **Jump**. Notice the magnifying glass icon has moved to the next test step.
5. Click on the **Play/Pause** button in the ribbon. TestPlanner will execute the step, and stop before completing the next step.

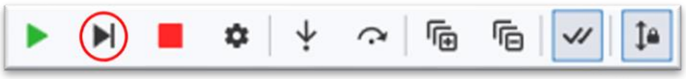


Figure 35: Play/Pause Button

6. Click on the **Abort** button in the ribbon. The last test step is not executed. The test is complete.

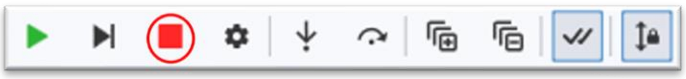


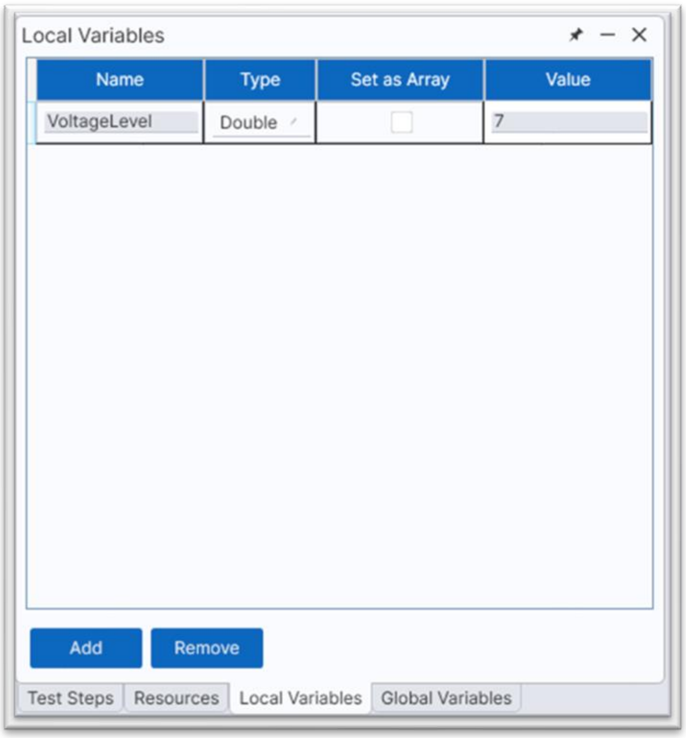
Figure 36: Abort Button

Using Variables

There are two types of variables in TestPlanner, which are configured similarly. These types are explained in Local Variables Panel and Global Variables Panel.

To use variables in TestPlanner Editor:

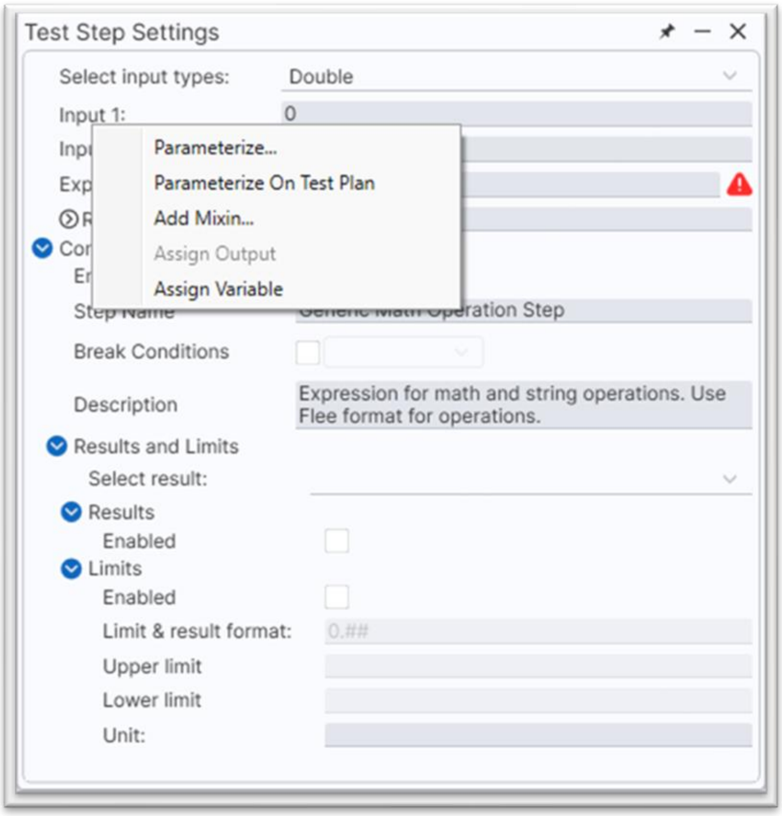
1. Start by creating a variable. Navigate to Local Variables Panel, click on **Add**. Configure the variable as following.



Name	Type	Set as Array	Value
VoltageLevel	Double	☐	7

Figure 37: Variable Configuration

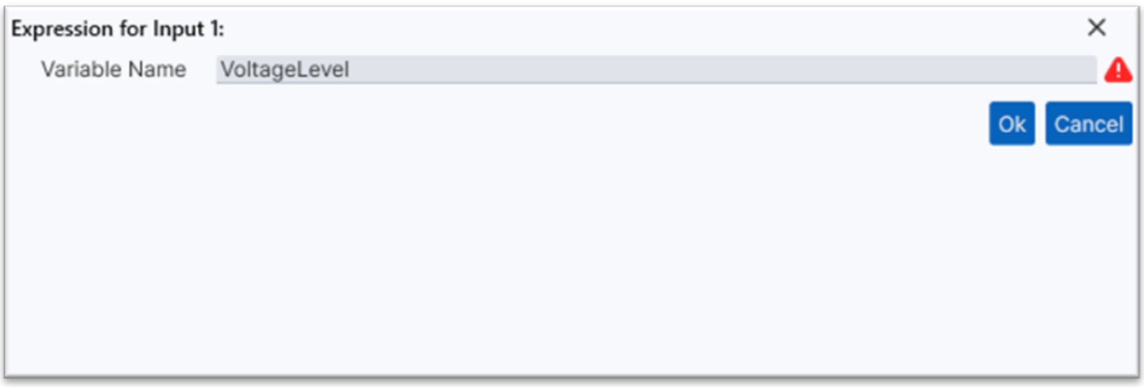
2. Navigate to **Test Steps > DEICO Math Operations > Generic Math Operation Test Step** in Test Steps Panel, click on the **Add** button.
3. Right click on “Input 1:”, a context menu will be displayed.



Select input types: Double
Input 1: 0
Step Name: Generic Math Operation Step
Break Conditions
Description: Expression for math and string operations. Use Flee format for operations.
Results and Limits
Select result:
Results
Enabled ☐
Limits
Enabled ☐
Limit & result format: 0.##
Upper limit
Lower limit
Unit:

Figure 38: Input 1 Context Menu

4. Click on **Assign Variable**, a window will be displayed. Enter the name of the variable that have been created, e.g., "VoltageLevel". Click on **OK**.


Expression for Input 1:

Variable Name: VoltageLevel

Figure 39: Input 1 Name

5. Now, Input 1 is tied to the variable.

Note Any change on the variable will be reflected on the tied test steps.

Using Expression Step

To use expression step in TestPlanner Editor:

1. Navigate to **Test Steps > TestPlanner.Expressions > Expression Step** in Menu Bar, click on the **Add** button.

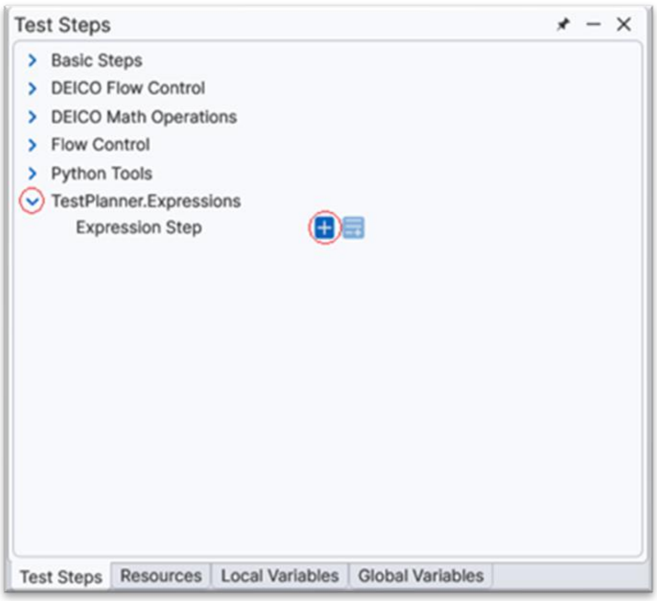
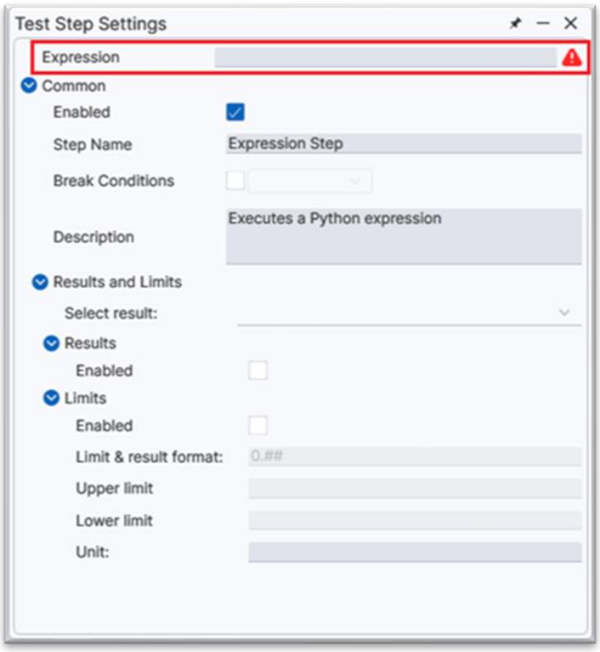


Figure 40: Expression Step

2. In test step properties, navigate the **Expression** field.

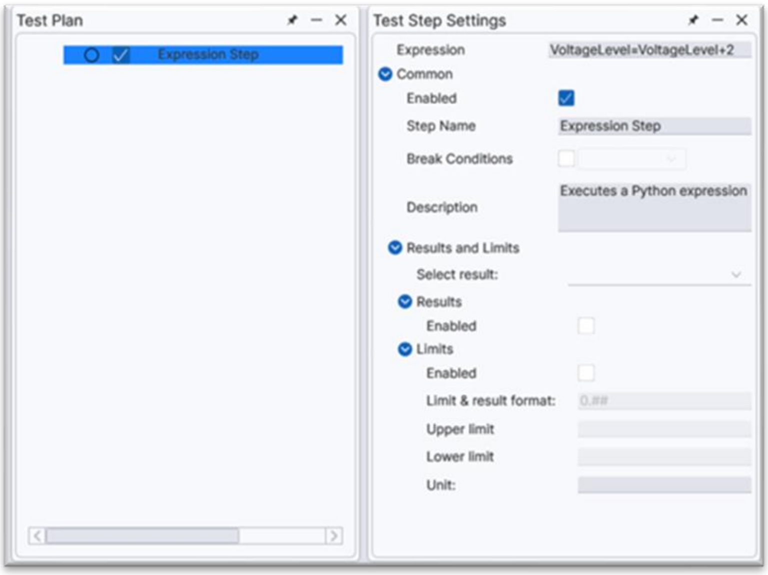


Test Step Settings

Expression	
Common	
Enabled	✓
Step Name	Expression Step
Break Conditions	
Description	Executes a Python expression
Results and Limits	
Select result:	
Results	
Enabled	
Limits	
Enabled	
Limit & result format:	0.##
Upper limit	
Lower limit	
Unit:	

Figure 41: Expression Step Configuration

Note This field allows a Python expression to be written and executed for this step. In the below example, the configured expression step will increment VoltageLevel local variable by 2.



Test Plan

Expression Step

Test Step Settings

Expression	VoltageLevel=VoltageLevel+2
Common	
Enabled	✓
Step Name	Expression Step
Break Conditions	
Description	Executes a Python expression
Results and Limits	
Select result:	
Results	
Enabled	
Limits	
Enabled	
Limit & result format:	0.##
Upper limit	
Lower limit	
Unit:	

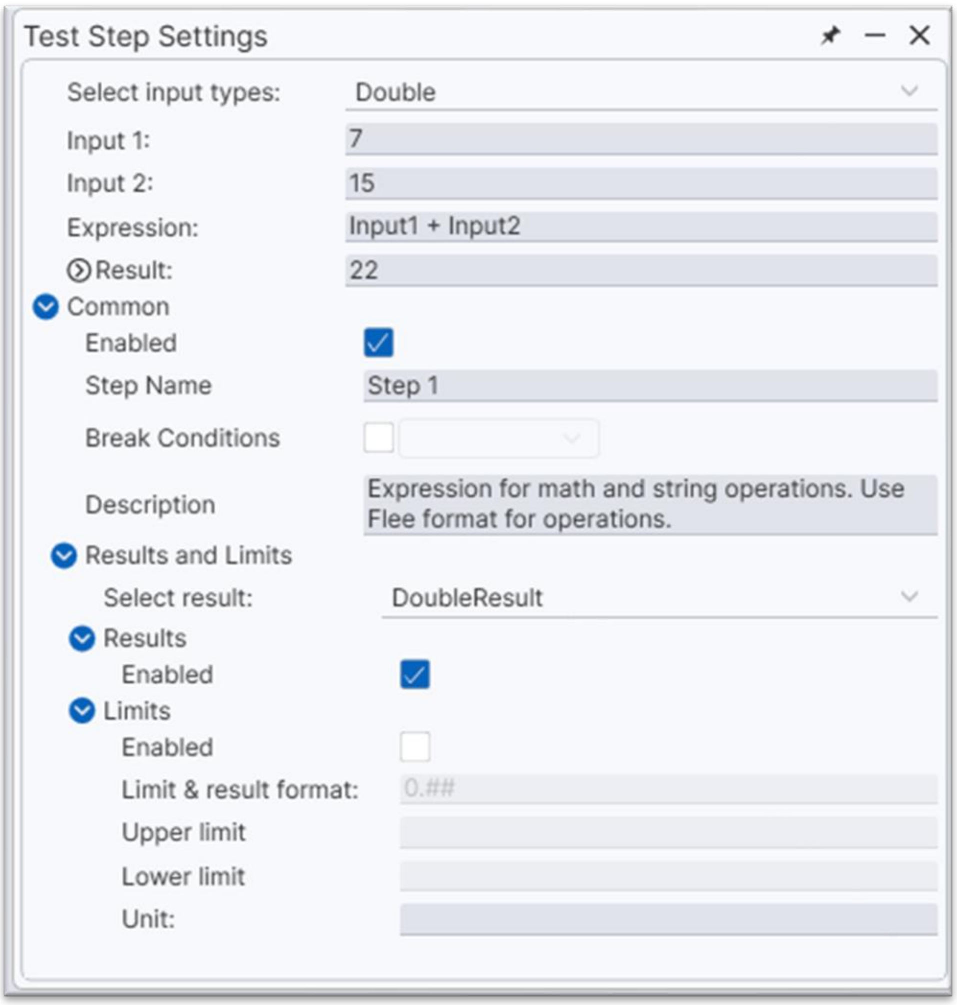
Figure 42: Expression Step Example

Using Outputs

The test steps can output values from their measurements or computations. These outputs can be assigned as input parameters to other test steps, allowing users to create interactions between the test steps.

To use outputs as inputs in TestPlanner Editor:

1. Navigate to **Test Steps > DEICO Math Operations > Generic Math Operation Test Step** in Menu Bar, click on the **Add** button.
2. Configure the test step properties as following.



Test Step Settings

Select input types:	Double
Input 1:	7
Input 2:	15
Expression:	Input1 + Input2
Result:	22
Common	
Enabled	✓
Step Name	Step 1
Break Conditions	
Description	Expression for math and string operations. Use Flee format for operations.
Results and Limits	
Select result:	DoubleResult
Results	
Enabled	✓
Limits	
Enabled	
Limit & result format:	0.##
Upper limit	
Lower limit	
Unit:	

Figure 43: Output Variable Configuration

3. Add another **Math Operation Test Step** to the test plan, change its name to “Step 2”.
4. Right click on **Input 1**, select **Assign Output**.
5. Select “Step 1” from the list. Output of the first test step is now assigned as the input of the second test step.

Using Parameters

Parameters can be generated as two types:

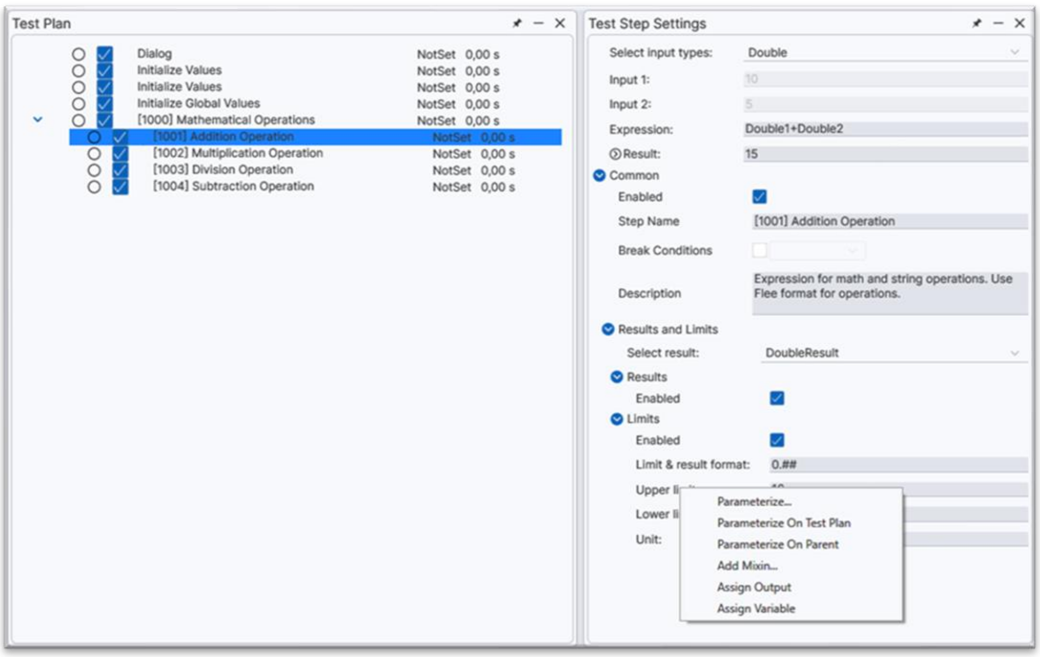
⇨ As parent

⇨ As test plan

Parameters in Parent

In the below example:

1. Navigate the Test Step Settings in **[1001] Addition Operation**.
2. Right click on **Upper Limit**.
3. Click on **Parameterize....**


Test Plan

Dialog	NotSet	0,00 s
Initialize Values	NotSet	0,00 s
Initialize Values	NotSet	0,00 s
Initialize Global Values	NotSet	0,00 s
[1000] Mathematical Operations	NotSet	0,00 s
[1001] Addition Operation	NotSet	0,00 s
[1002] Multiplication Operation	NotSet	0,00 s
[1003] Division Operation	NotSet	0,00 s
[1004] Subtraction Operation	NotSet	0,00 s

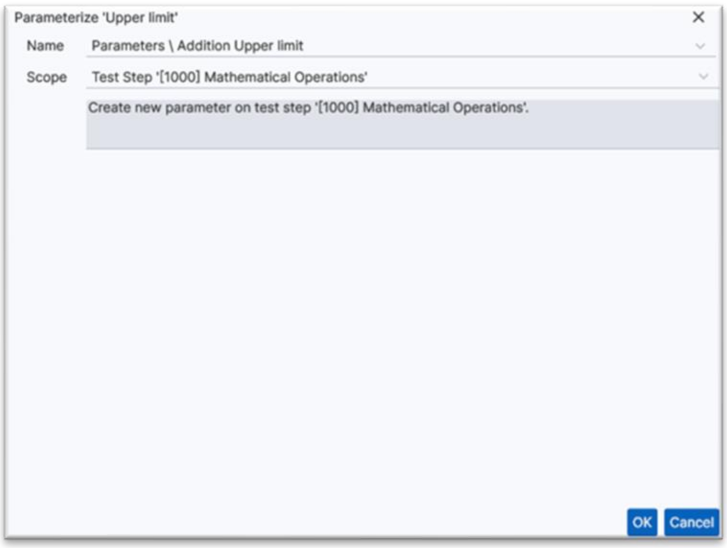
Test Step Settings

Select input types: Double
Input 1: 10
Input 2: 5
Expression: Double1+Double2
Result: 15
Common
Enabled: ✓
Step Name: [1001] Addition Operation
Break Conditions
Description: Expression for math and string operations. Use Flee format for operations.
Results and Limits
Select result: DoubleResult
Results
Enabled: ✓
Limits
Enabled: ✓
Limit & result format: 0.##
Upper li
Lower li
Unit:

Figure 44: Right Click for Parameterize

4. Select Parent Sequence as **[1000] Mathematical Operations** in **Scope**.
5. Change the name as “Parameters \ Addition Upper Limit [Desired Value]”.

Note “Parameters \” should not be deleted.

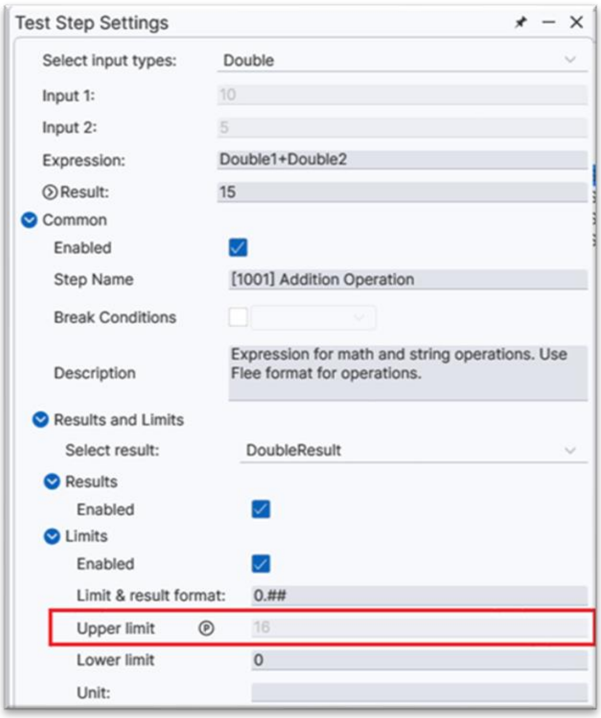

Parameterize 'Upper limit'

Name: Parameters \ Addition Upper limit
Scope: Test Step '[1000] Mathematical Operations'

Create new parameter on test step '[1000] Mathematical Operations'.

Figure 45: Parameterize Upper Limit

6. The visual change on this parameter can be seen.

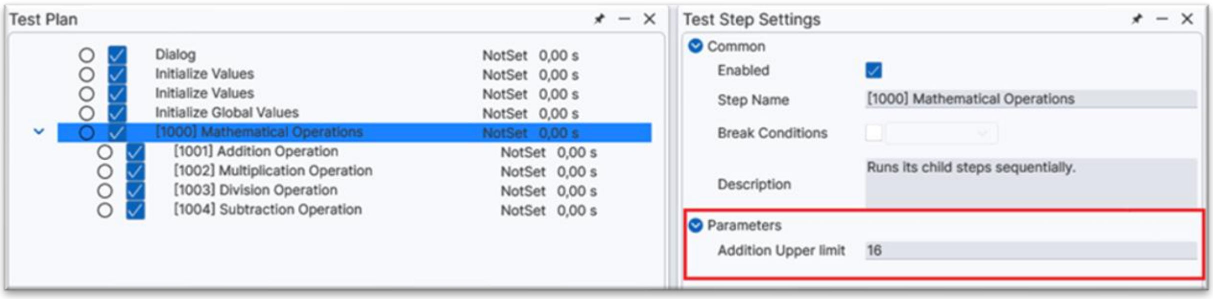


Test Step Settings

Select input types: Double
Input 1: 10
Input 2: 5
Expression: Double1+Double2
Result: 15
Common
Enabled ✓
Step Name [1001] Addition Operation
Break Conditions
Description: Expression for math and string operations. Use Flee format for operations.
Results and Limits
Select result: DoubleResult
Results
Enabled ✓
Limits
Enabled ✓
Limit & result format: 0.##
Upper limit Ⓟ 16
Lower limit 0
Unit:

Figure 46: Parameterized Upper Limit

7. Navigate the Parent Sequence as **[1000] Mathematical Operations**.
8. In Test Step Settings, navigate **Parameters** and **Addition Upper Limit** value.



Test Plan

```
Dialog                              NotSet  0,00 s
Initialize Values                   NotSet  0,00 s
Initialize Values                   NotSet  0,00 s
Initialize Global Values            NotSet  0,00 s
[1000] Mathematical Operations      NotSet  0,00 s
  [1001] Addition Operation         NotSet  0,00 s
  [1002] Multiplication Operation   NotSet  0,00 s
  [1003] Division Operation         NotSet  0,00 s
  [1004] Subtraction Operation      NotSet  0,00 s
```

Test Step Settings

Common
Enabled ✓
Step Name [1000] Mathematical Operations
Break Conditions
Description: Runs its child steps sequentially.
Parameters
Addition Upper limit 16

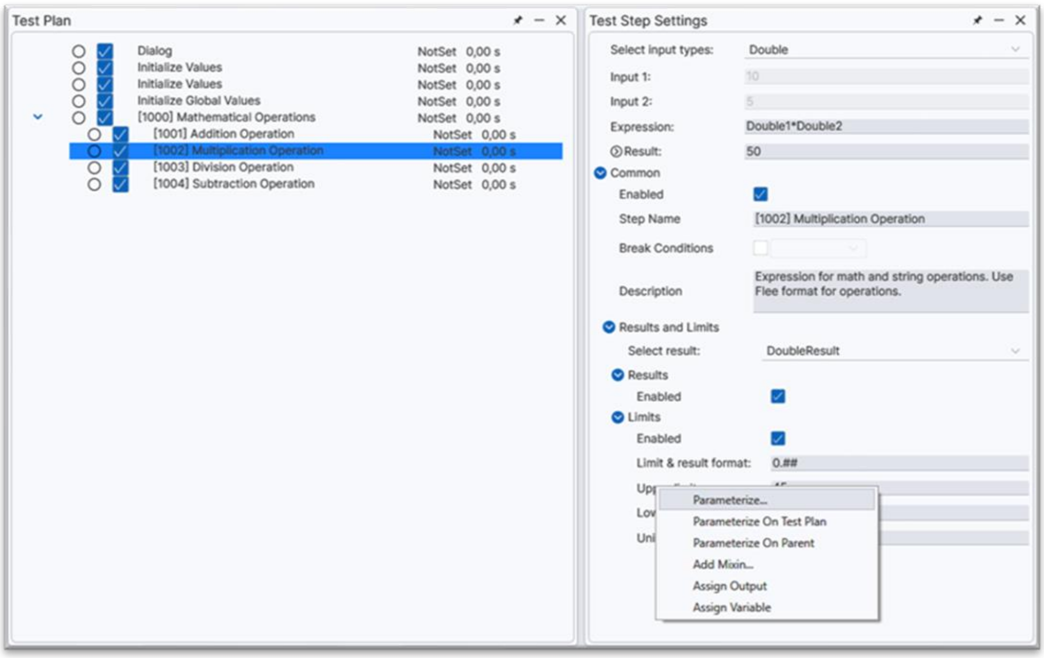
Figure 47: Addition Upper Limit

9. This value can be edited and used in any test step as **Parameters**.

Parameters in Test Plan

In the below example:

1. Navigate the Test Step Settings in **[1002] Multiplication Operation**.
2. Right click on **Upper Limit**.
3. Click on **Parameterize...**.



Test Plan

```
Dialog                                  NotSet  0,00 s
Initialize Values                       NotSet  0,00 s
Initialize Values                       NotSet  0,00 s
Initialize Global Values                NotSet  0,00 s
[1000] Mathematical Operations          NotSet  0,00 s
    [1001] Addition Operation               NotSet  0,00 s
    [1002] Multiplication Operation         NotSet  0,00 s
    [1003] Division Operation               NotSet  0,00 s
    [1004] Subtraction Operation            NotSet  0,00 s
```

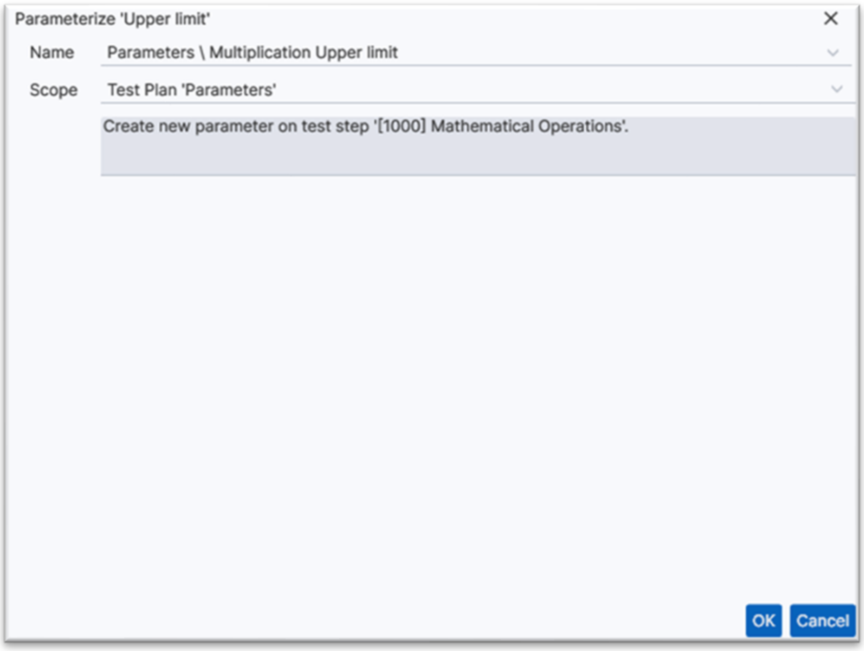
Test Step Settings

Select input types:	Double
Input 1:	10
Input 2:	5
Expression:	Double1*Double2
Result:	50
Common	
Enabled	✓
Step Name	[1002] Multiplication Operation
Break Conditions	
Description	Expression for math and string operations. Use Flee format for operations.
Results and Limits	
Select result:	DoubleResult
Results	
Enabled	✓
Limits	
Enabled	✓
Limit & result format:	0.##

Figure 48: Right Click for Parameterize

4. Select Parameters as **Test Plan** in **Scope**.
5. Change the name as "Parameters \ Multiplication Upper Limit [Desired Value]".

Note "Parameters \" should not be deleted.



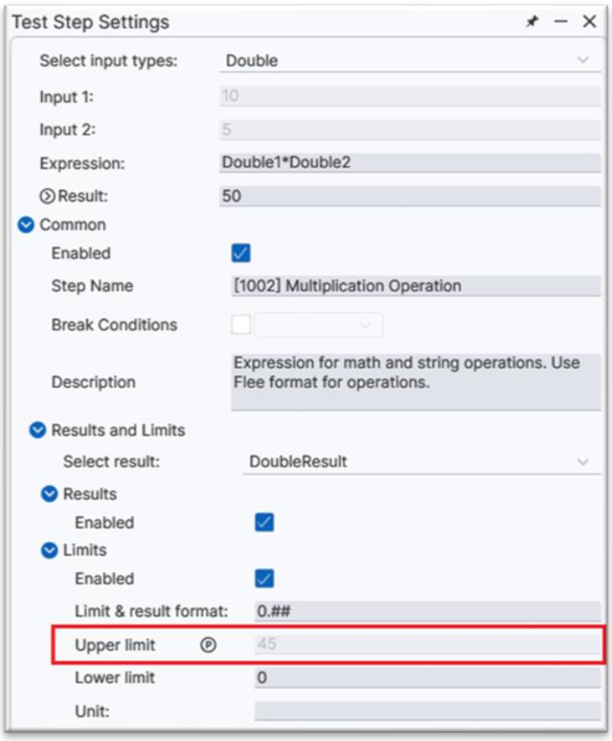
Parameterize 'Upper limit'

Name	Parameters \ Multiplication Upper limit
Scope	Test Plan 'Parameters'

Create new parameter on test step '[1000] Mathematical Operations'.

Figure 49: Parameterize Upper Limit

6. The visual change on this parameter can be seen.


Test Step Settings

Select input types: Double
Input 1: 10
Input 2: 5
Expression: Double1*Double2
Result: 50
Common
Enabled ✓
Step Name [1002] Multiplication Operation
Break Conditions
Description Expression for math and string operations. Use Flee format for operations.
Results and Limits
Select result: DoubleResult
Results
Enabled ✓
Limits
Enabled ✓
Limit & result format: 0.##
Upper limit 45
Lower limit 0
Unit:

Figure 50: Parameterized Upper Limit

7. Navigate the **Settings** in **TestPlanner Ribbon**.

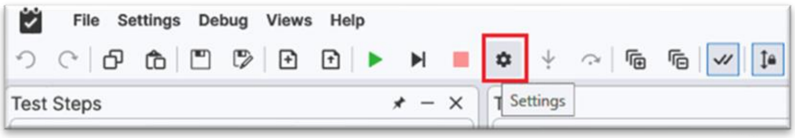
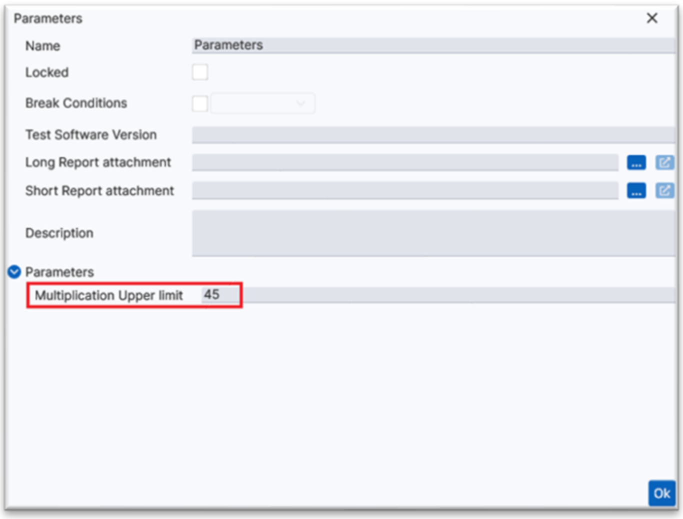


Figure 51: Settings

8. In settings, navigate **Parameters** and **Multiplication Upper Limit** value.


Parameters

Name Parameters
Locked
Break Conditions
Test Software Version
Long Report attachment
Short Report attachment
Description
Parameters
Multiplication Upper limit 45

Ok

Figure 52: Multiplication Upper Limit

9. This value can be edited and used in any test step as **Parameters**.

Using Mixins

TestPlanner includes numerous mixins, some of which are particularly significant and can be analyzed in detail within this document.

Pre-Expression Mixin

Pre-expression is a versatile mixin in TestPlanner that enables users to run custom Python scripts before executing a test step. By integrating this mixin, users can define pre-test logic and operations to prepare or modify conditions dynamically, ensuring enhanced flexibility and control over test execution workflows.

Pre-expression mixin can be added as follows.

1. Navigate the test step where the pre-expression is to be added.
2. In Test Step Settings, right click any properties and select **Add Mixin…**.

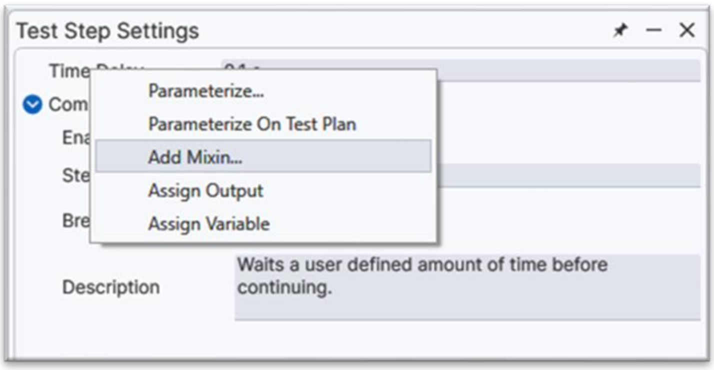


Figure 53: Adding Mixin

3. In the page, select **Pre-Expression Mixin**.

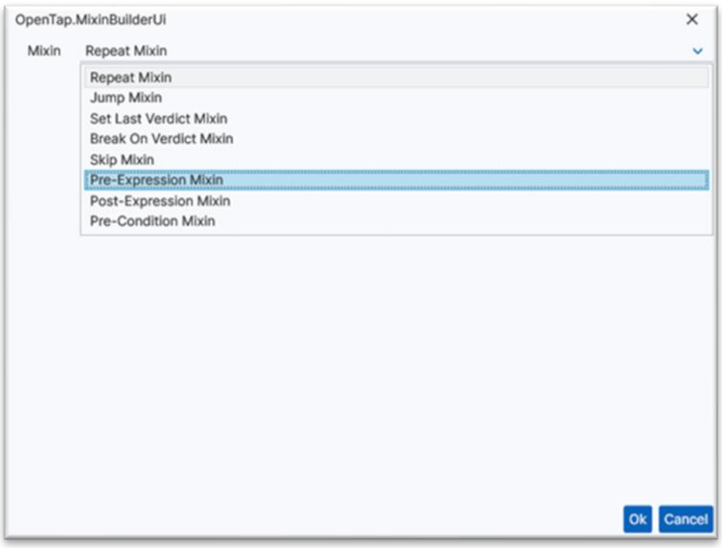
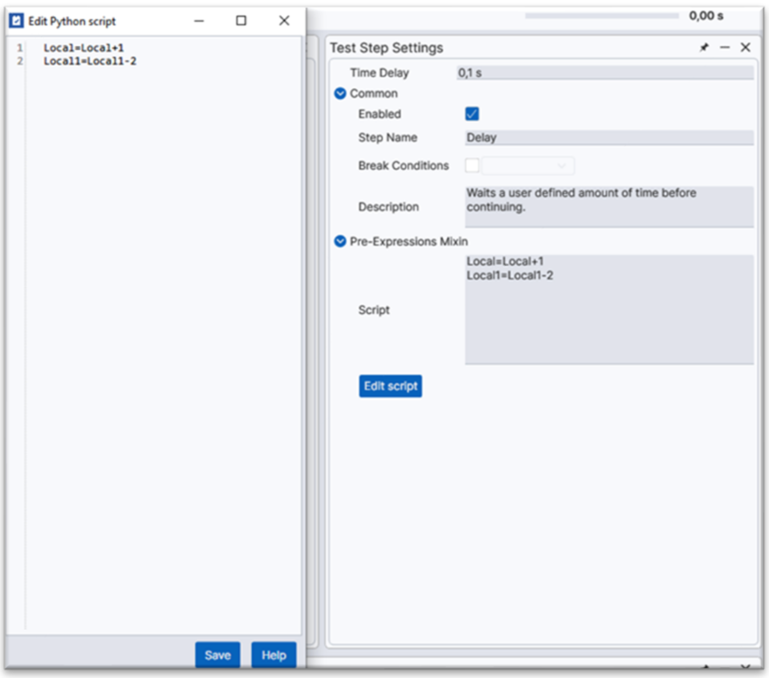


Figure 54: Pre-Expression Mixin Selection

4. In Test Step Settings, **Pre-Expression Mixin** tag can be seen as activated. Python script can be modified by clicking on **Edit Script** or writing the expression through the area defined.



```
Local=Local+1
Local1=Local1-2
```

Time Delay: 0,1 s
Common
Enabled: ✓
Step Name: Delay
Break Conditions:
Description: Waits a user defined amount of time before continuing.
Pre-Expressions Mixin
Script: Local=Local+1 Local1=Local1-2

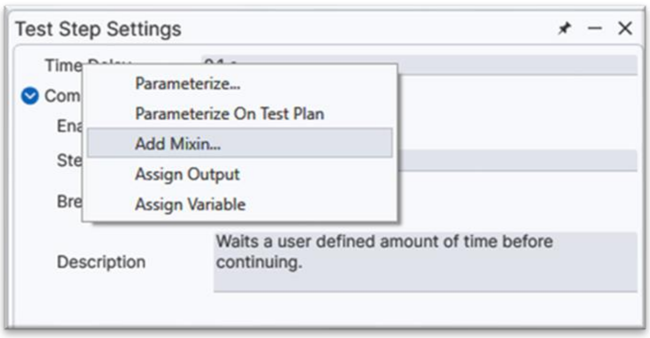
Figure 55: Writing the Python Script

Post-Expression Mixin

Post-expression is a powerful mixin in TestPlanner designed to execute custom Python scripts after a test step has completed. By adding this mixin, users can define post-test logic and actions, such as result processing, logging, or state adjustments, providing greater flexibility and control over post-execution workflows.

Post-expression mixin can be added as follows.

1. Navigate the test step where the post-expression is to be added.
2. In Test Step Settings, right click any properties and select **Add Mixin…**.



Description: Waits a user defined amount of time before continuing.

Figure 56: Adding Mixin

3. In the page, select **Post-Expression Mixin**.

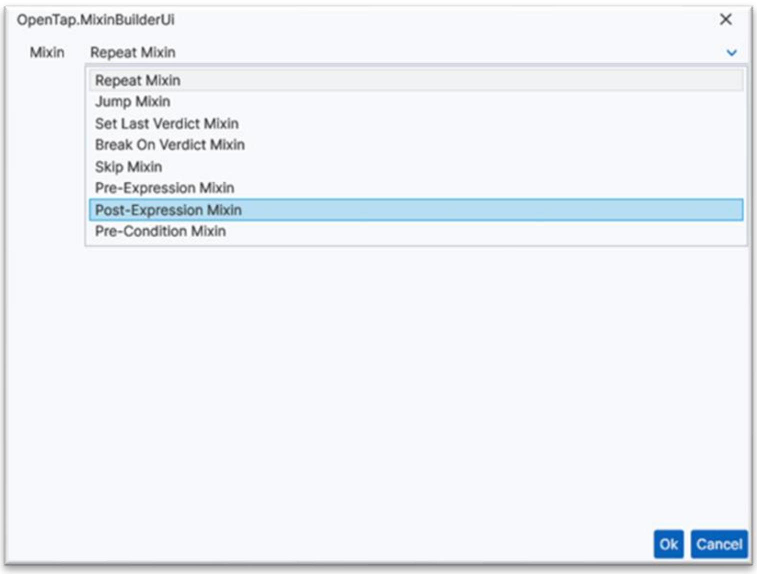
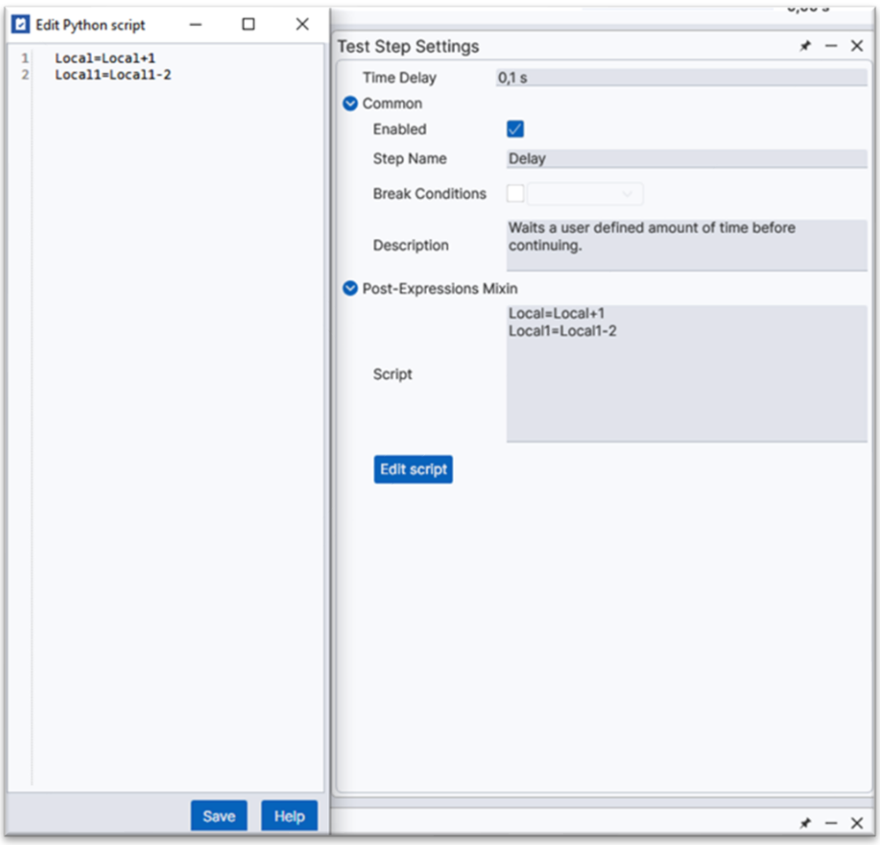


Figure 57: Post-Expression Mixin Selection

4. In Test Step Settings, **Post-Expression Mixin** tag can be seen as activated. Python script can be modified by clicking on **Edit Script** or writing the expression through the area defined.



```
Local=Local+1
Local1=Local1-2
```

Time Delay: 0,1 s
Enabled: ✓
Step Name: Delay
Break Conditions:
Description: Waits a user defined amount of time before continuing.
Script:
Local=Local+1
Local1=Local1-2

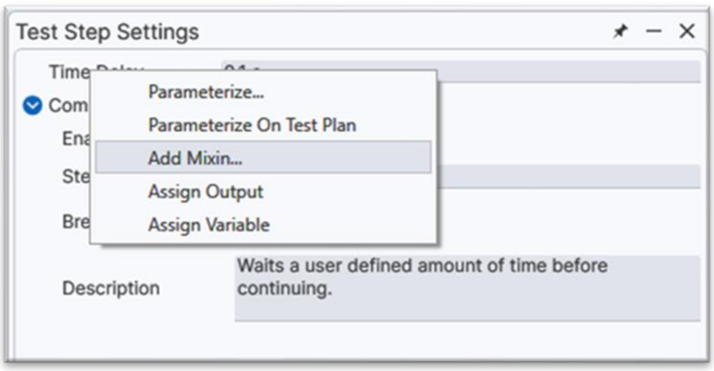
Figure 58: Writing the Python Script

Pre-Condition Mixin

Pre-condition is a conditional mixin in TestPlanner that allows users to control whether a test step will be executed based on a custom Python script. Within the script, setting the **PRECONDITION** value to **TRUE** enables the test step to run, while assigning **FALSE** or leaving it unset prevents its execution. This mixin empowers users to define dynamic execution logic, improving test flow efficiency and adaptability.

Pre-condition mixin can be added as follows.

1. Navigate the test step where the pre-condition is to be added.
2. In Test Step Settings, right click any properties and select **Add Mixin…**.



Waits a user defined amount of time before continuing.

Figure 59: Adding Mixin

3. In the page, select **Pre-Condition Mixin**.

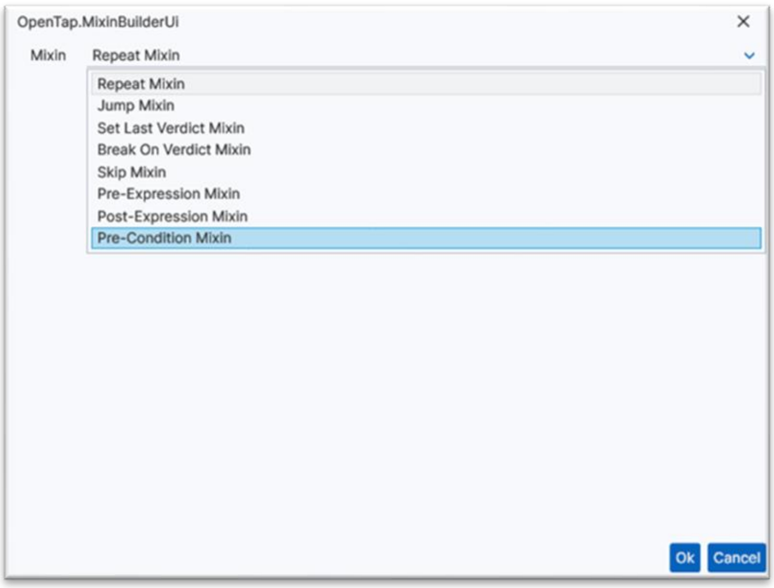
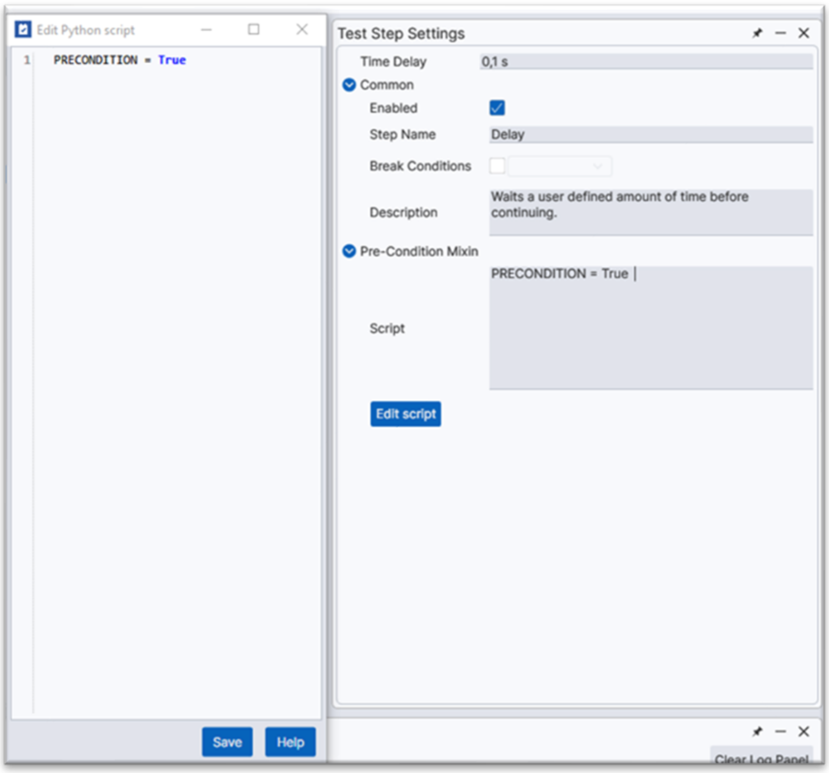


Figure 60: Pre-Condition Mixin Selection

4. In Test Step Settings, **Pre-Condition Mixin** tag can be seen as activated. Python script can be modified by clicking on **Edit Script** or writing the expression through the area defined.

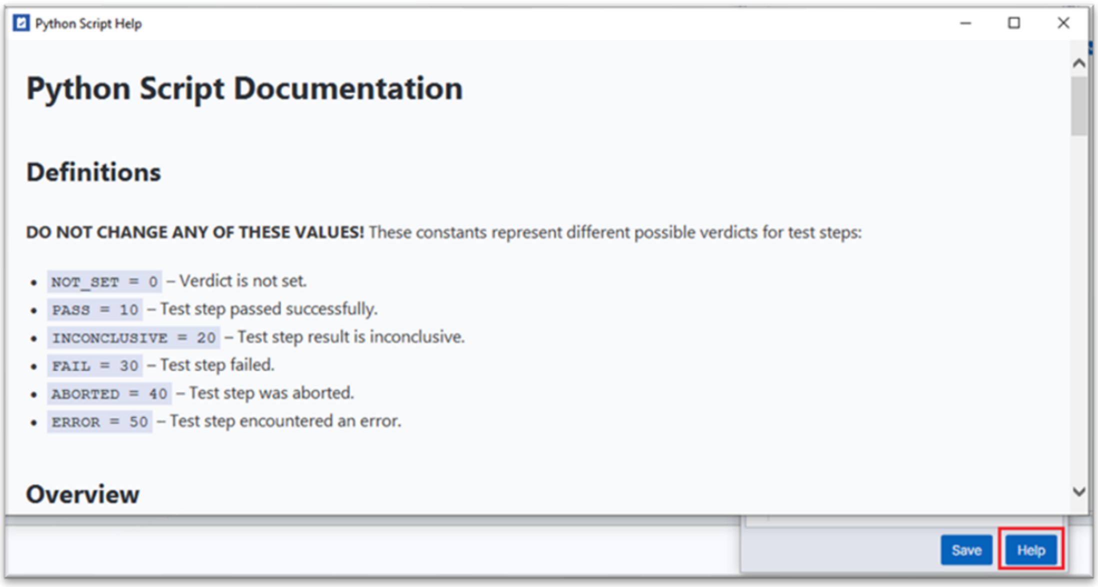


```
PRECONDITION = True
```

Time Delay	0,1 s
Common	
Enabled	✓
Step Name	Delay
Break Conditions	
Description	Waits a user defined amount of time before continuing.
Pre-Condition Mixin	
Script	PRECONDITION = True

Figure 61: Writing the Python Script

Note Selection can be made by **Help** button in **Edit Python Script** page. There are lots of examples and explanation how Python script and TestPlanner communicates.



Python Script Documentation

Definitions

DO NOT CHANGE ANY OF THESE VALUES! These constants represent different possible verdicts for test steps:

- `NOT_SET = 0` – Verdict is not set.
- `PASS = 10` – Test step passed successfully.
- `INCONCLUSIVE = 20` – Test step result is inconclusive.
- `FAIL = 30` – Test step failed.
- `ABORTED = 40` – Test step was aborted.
- `ERROR = 50` – Test step encountered an error.

Overview

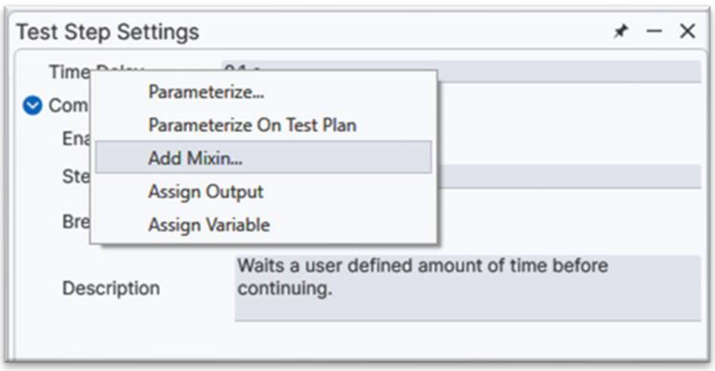
Figure 62: Python Script Documentation

Jump Mixin

Jump mixin is a control flow mixin in TestPlanner that enables conditional branching between test steps based on the outcome of an *expected verdict*. When the test result matches the expected verdict, the mixin allows jumping directly to a specified test step, streamlining test sequences. Additionally, users can configure a maximum jump count to limit how many times a jump action can be performed, ensuring better control over test execution logic.

Jump mixin can be added as follows.

1. Navigate the test step where the jump mixin is to be added.
2. In Test Step Settings, right click any properties and select **Add Mixin…**.



Description: Waits a user defined amount of time before continuing.

Figure 63: Adding Mixin

3. In the page, select **Jump Mixin**.

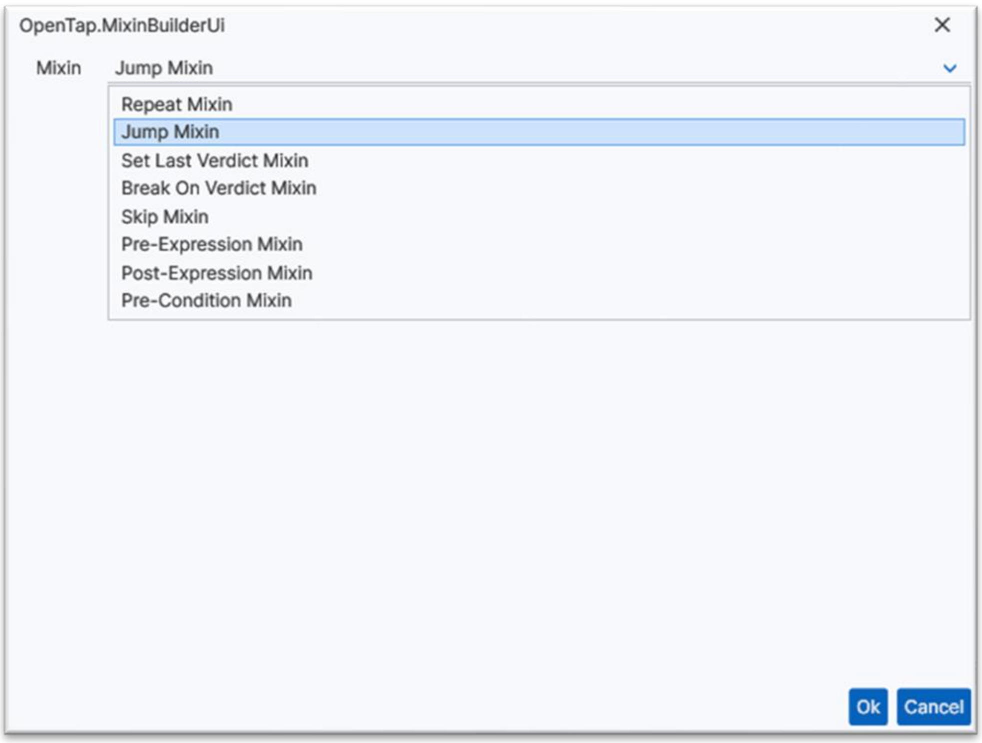


Figure 64: Jump Mixin Selection

4. In Test Step Settings, **Jump Mixin** tag can be seen as activated.

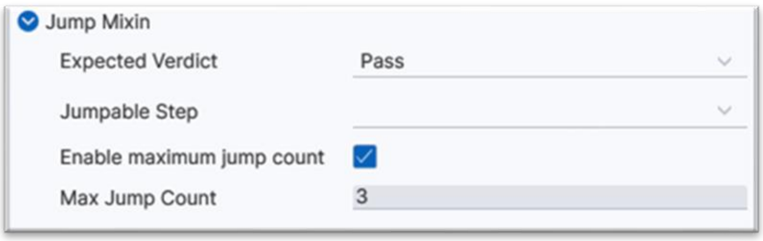

Jump Mixin
Expected Verdict: Pass
Jumpable Step:
Enable maximum jump count: ✓
Max Jump Count: 3

Figure 65: Jump Mixin

Note In jump mixin area, there are 3 inputs defined as:

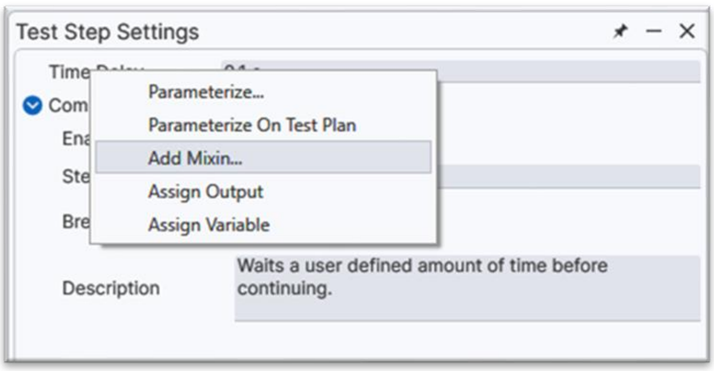
- ⇨ **Expected Verdict:** This input defines the specific verdict that activates the jump logic. If the result of the test step matches this value, the jump action is triggered.
- ⇨ **Jumpable Step:** This input specifies the target step to jump to when the jump logic is activated. This allows dynamic redirection of the test sequence based on conditions.
- ⇨ **Enable Maximum Jump Count:** This input limits the number of times the jump action can be executed within a test sequence. This input prevents infinite loops and ensures controlled test flow.

Repeat Mixin

Repeat mixin is a loop control mixin in TestPlanner that enables the repeated execution of a test step until a specified repeat condition is satisfied. The condition can be based on a dynamic logic or a fixed number of iterations. Additionally, users can count the number of **Pass/Fail** outcomes and determine the final result of the test step based on these counters, providing precise control over repetitive test scenarios.

Repeat mixin can be added as follows.

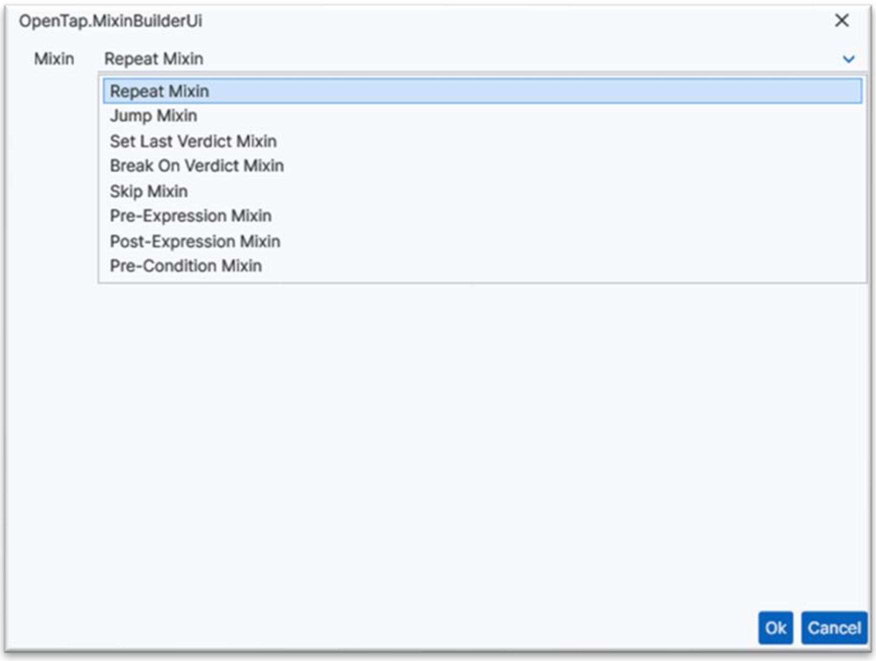
1. Navigate the test step where the repeat mixin is to be added.
2. In Test Step Settings, right click any properties and select **Add Mixin…**.



Description: Waits a user defined amount of time before continuing.

Figure 66: Adding Mixin

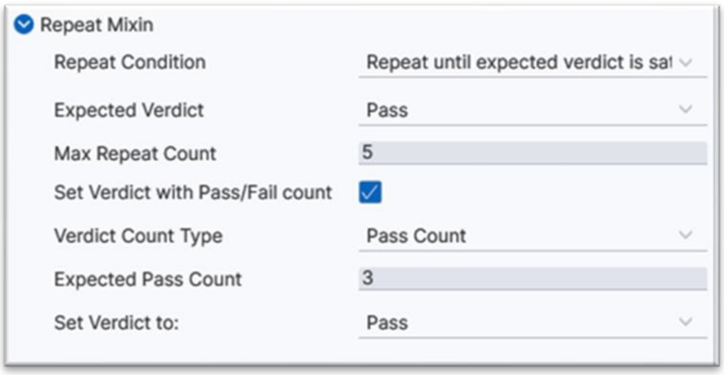
3. In the page, select **Repeat Mixin**.



Repeat Mixin
Jump Mixin
Set Last Verdict Mixin
Break On Verdict Mixin
Skip Mixin
Pre-Expression Mixin
Post-Expression Mixin
Pre-Condition Mixin

Figure 67: Repeat Mixin Selection

4. In Test Step Settings, **Repeat Mixin** tag can be seen as activated.


Repeat Mixin

Repeat Condition	Repeat until expected verdict is sat
Expected Verdict	Pass
Max Repeat Count	5
Set Verdict with Pass/Fail count	✓
Verdict Count Type	Pass Count
Expected Pass Count	3
Set Verdict to:	Pass

Figure 68: Repeat Mixin

Note In repeat mixin area, there are 7 inputs defined as:

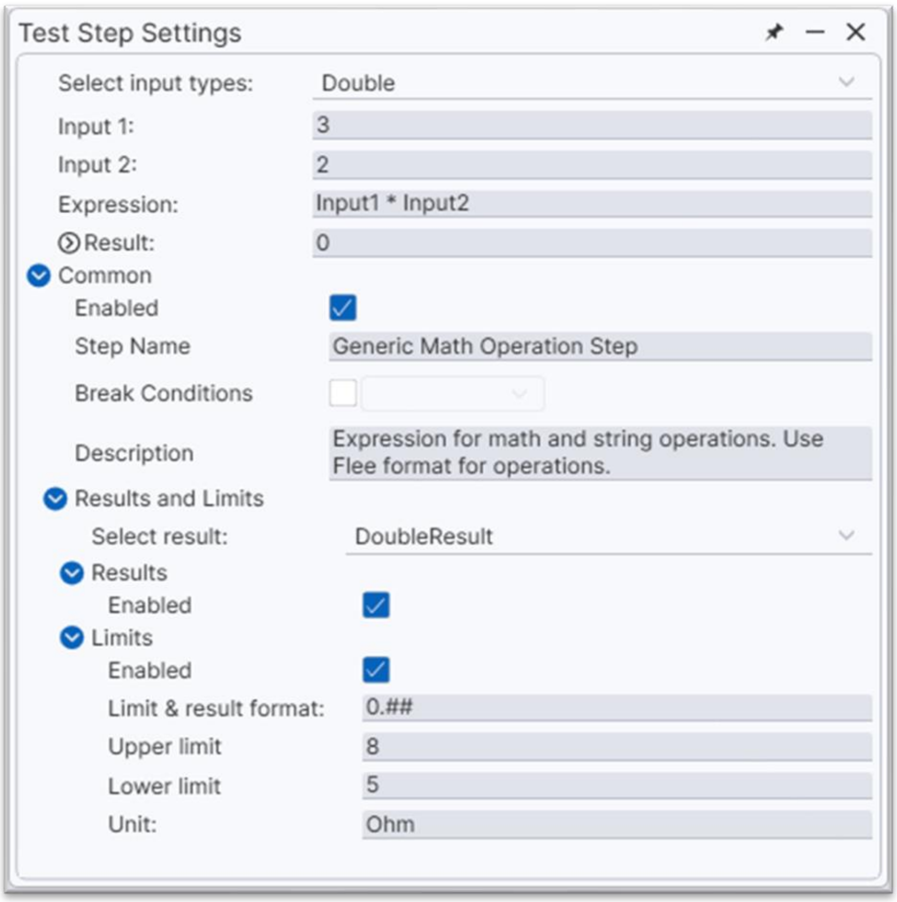
- ⇨ **Repeat Condition:** This input determines whether the test step should repeat based on a fixed number of iterations or until a specified condition is met. This input controls the core logic for repetition.
- ⇨ **Expected Verdict:** This input defines the condition value that triggers repeated execution if the condition-based repeat mode is selected. The step continues to run until this verdict is satisfied.
- ⇨ **Max Repeat Count:** This input specifies the maximum number of times the test step can be repeated. It prevents infinite loops by capping repetitions.
- ⇨ **Set Verdict with Pass/Fail Count:** This input enables the option to count Pass or Fail outcomes and use the count to decide the final verdict of the repeated step. This option must be selected to activate verdict counting.

- ⇨ **Verdict Count Type:** This input determines which verdict (Pass or Fail) will be counted when verdict counting is enabled. This allows targeted tracking of specific outcomes.
- ⇨ **Expected Count:** This input sets the number of times the specified verdict must be observed to influence the final verdict of the step. It defines the success or failure threshold.
- ⇨ **Set Verdict to:** This input specifies the final verdict to assign to the test step once the desired verdict count is reached. This input determines how the step's outcome is concluded after repetition.

Generating a Report

To generate a report in TestPlanner Editor:

1. Navigate to **Test Steps > DEICO Math Operations > Generic Math Operation Test Step** in Menu Bar, click on the **Add** button.
2. Configure the test step properties as following.



Test Step Settings

Select input types:	Double
Input 1:	3
Input 2:	2
Expression:	Input1 * Input2
Result:	0
Common	
Enabled	✓
Step Name	Generic Math Operation Step
Break Conditions	☐
Description	Expression for math and string operations. Use Flee format for operations.
Results and Limits	
Select result:	DoubleResult
Results	
Enabled	✓
Limits	
Enabled	✓
Limit & result format:	0.##
Upper limit	8
Lower limit	5
Unit:	Ohm

Figure 69: Test Step Settings

3. Navigate to **Settings > Results** in Menu Bar. Click on the expander next to **Text**, and click on the **Add** button next to **Text Log**. Then, click on the settings button in the right-hand side.

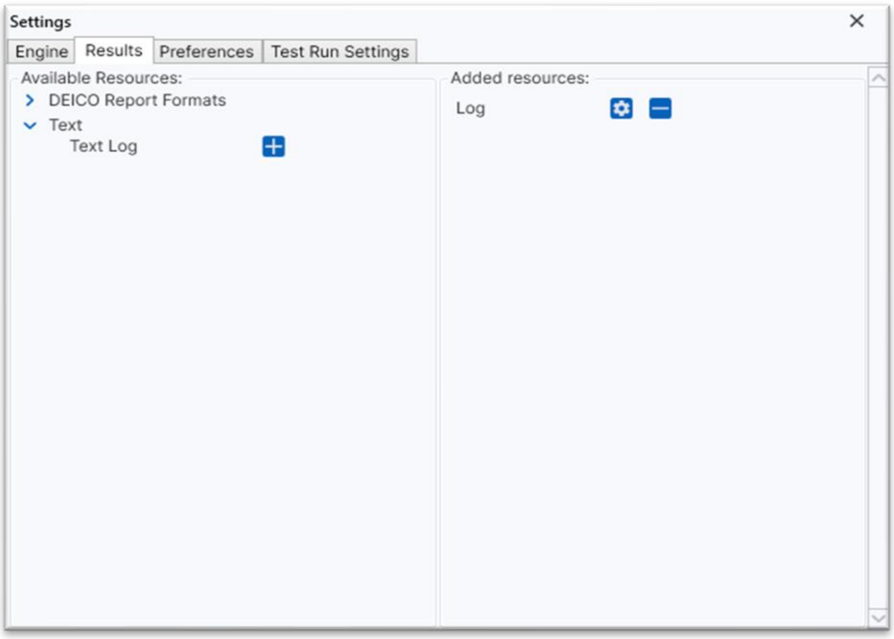


Figure 70: Results View

4. The path of the report file and filtering for logs can be changed.
5. Run the test plan. **The report file will be saved in the specified directory**.

Note Note that the **Text** plugin only writes the logs to a file. Other plugins can be used to generate detailed reports in various formats such as CSV and PDF.

Generating PDF Reports with Sequences

Note In the previous section, the process of saving test results in text format using the result listener structure of OpenTAP was explained. This section will provide an overview of the general usage of the PDF Report Plugin included with TestPlanner, along with a description of an example usage scenario.

To generate a PDF report in TestPlanner Editor:

1. Navigate **Resources** page and click on **Add** button in **Results** tag (Error! Reference source not found.), or navigate **Settings > Results > Available Resources > TestPlanner Report Formats > PDF Formats** and click on **Add** for TestPlanner PDF reports (Error! Reference source not found.).

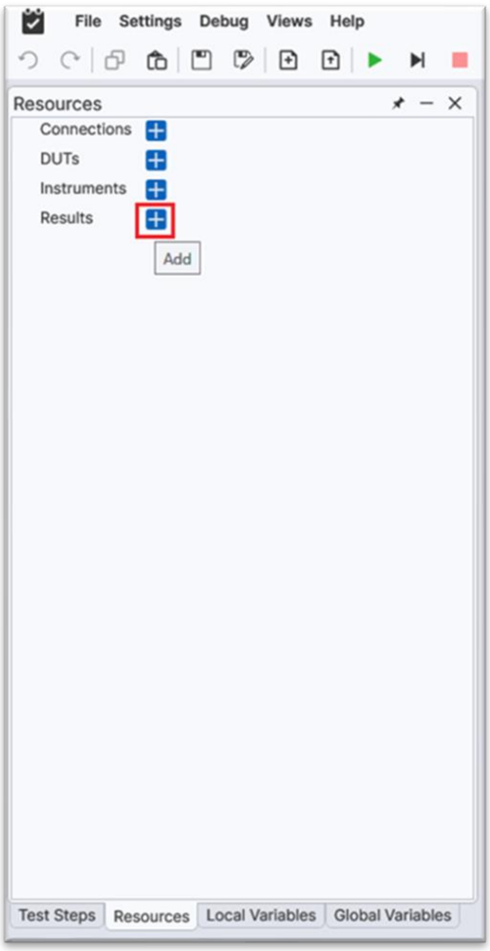


Figure 71: Adding Result Resource

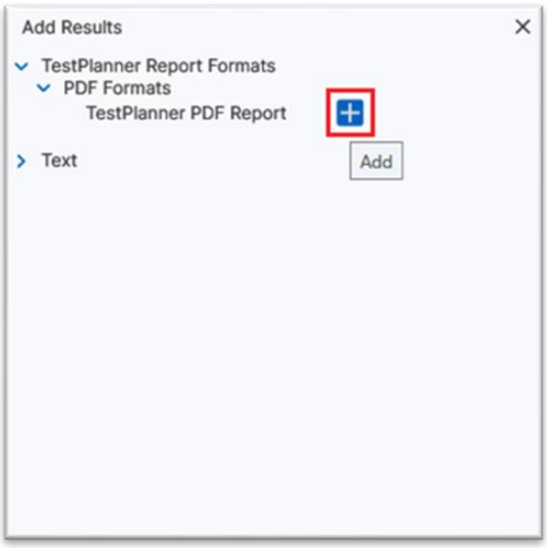


Figure 72: Adding Result Resource from Settings

2. Navigate **Resources** page and click on the **Settings** button in **Results** tag.

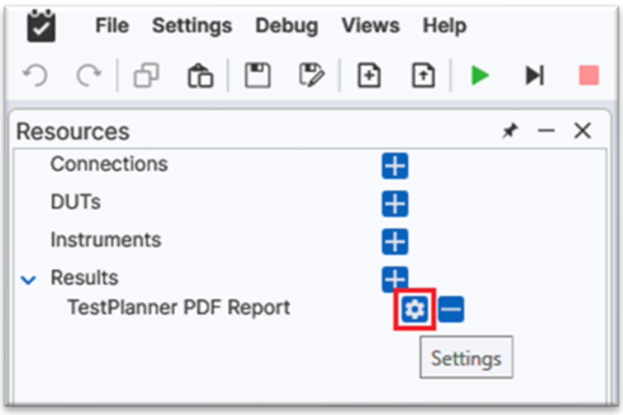
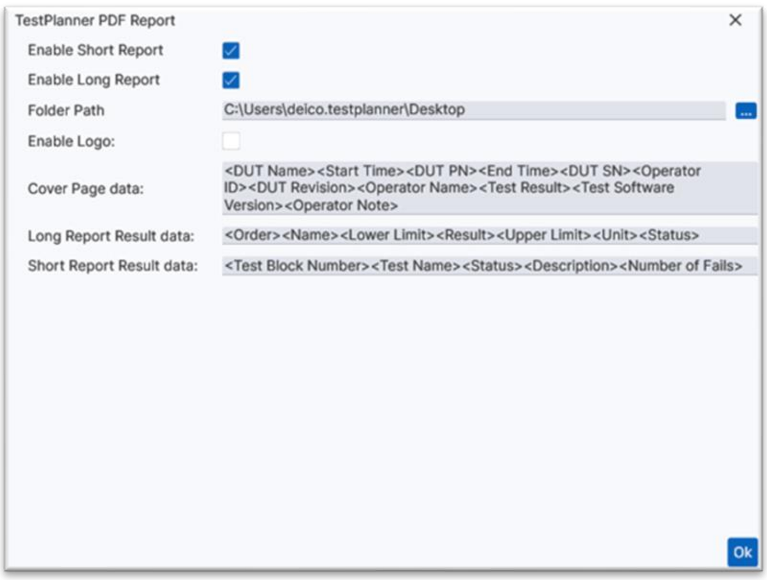


Figure 73: PDF Report Settings

3. In **Settings** page, the report format and the path can be set within given information accordingly.



TestPlanner PDF Report

Enable Short Report	✓
Enable Long Report	✓
Folder Path	C:\Users\deico.testplanner\Desktop
Enable Logo:	
Cover Page data:	<DUT Name><Start Time><DUT PN><End Time><DUT SN><Operator ID><DUT Revision><Operator Name><Test Result><Test Software Version><Operator Note>
Long Report Result data:	<Order><Name><Lower Limit><Result><Upper Limit><Unit><Status>
Short Report Result data:	<Test Block Number><Test Name><Status><Description><Number of Fails>

Ok

Figure 74: PDF Report Settings

4. Navigate **Settings > PDF Report Settings > Execution Settings** to open execution settings of PDF report plugin.

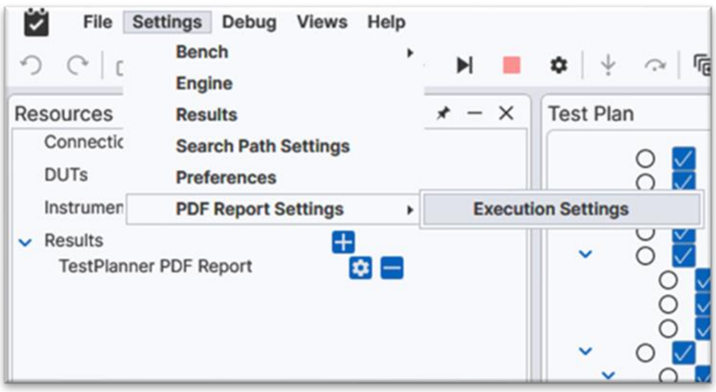
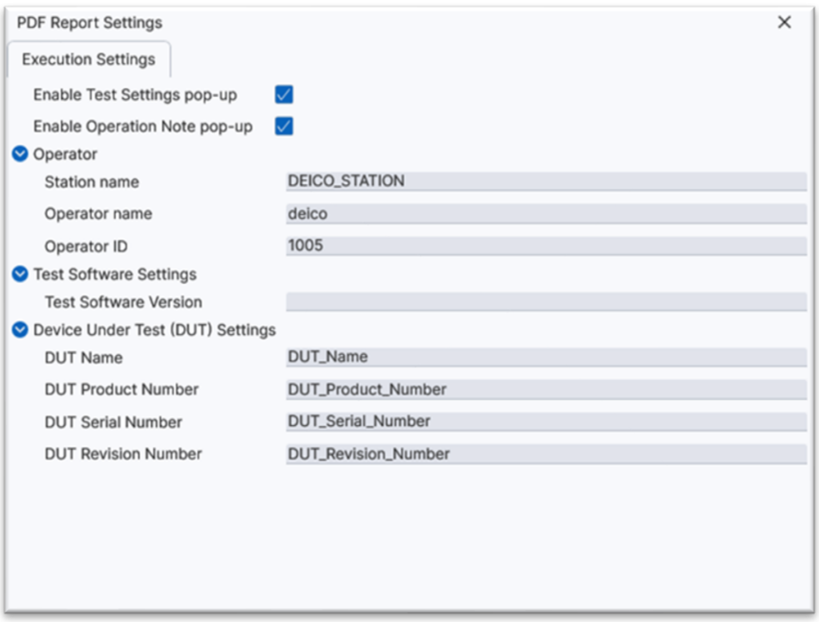


Figure 75: Execution Settings

Note In Execution Settings:

- ⇨ **Test Settings Pop-Up:** When this option is enabled, a pop-up window appears at the start of the test, allowing the operator to enter information about the test operator and the product under test from the **Execution Settings** section.
- ⇨ **Operation Note Pop-Up:** When this option is enabled, a pop-up window is displayed at the end of the test, providing a field for entering notes that will be included at the end of the test report.
- ⇨ **Operator and Product Information:** This section is dedicated to allow users to input details about the operator and the product under test, which will be displayed in the report. If the **Test Settings Pop-Up** is activated, these values can be updated by the user at the beginning of the test.



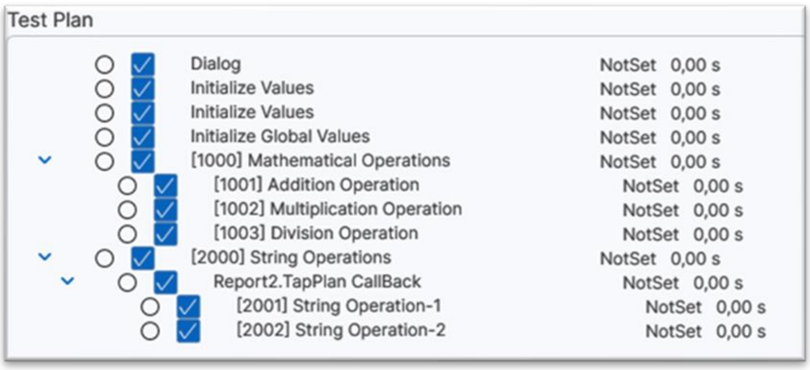
PDF Report Settings

Execution Settings

Enable Test Settings pop-up	✓
Enable Operation Note pop-up	✓
Operator	
Station name	DEICO_STATION
Operator name	deico
Operator ID	1005
Test Software Settings	
Test Software Version	
Device Under Test (DUT) Settings	
DUT Name	DUT_Name
DUT Product Number	DUT_Product_Number
DUT Serial Number	DUT_Serial_Number
DUT Revision Number	DUT_Revision_Number

Figure 76: Execution Settings

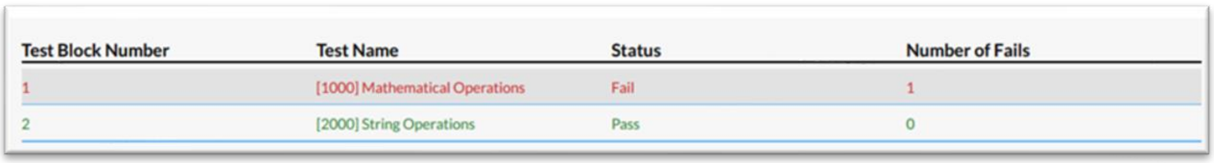
Important Note This report plugin uses the **Parent<>Child** relationship when generating the report. Therefore, test steps that produce results must be placed within a **Sequence**. It is important to ensure this structure is correctly followed. An example structure can be seen in Figure 77. The example report of this test plan can be seen in Figure 78.



Test Plan

```
✓ Dialog                                   NotSet  0,00 s
✓ Initialize Values                        NotSet  0,00 s
✓ Initialize Values                        NotSet  0,00 s
✓ Initialize Global Values                 NotSet  0,00 s
✓ [1000] Mathematical Operations           NotSet  0,00 s
    ✓ [1001] Addition Operation              NotSet  0,00 s
    ✓ [1002] Multiplication Operation        NotSet  0,00 s
    ✓ [1003] Division Operation              NotSet  0,00 s
✓ [2000] String Operations                 NotSet  0,00 s
    ✓ Report2.TapPlan CallBack               NotSet  0,00 s
        ✓ [2001] String Operation-1            NotSet  0,00 s
        ✓ [2002] String Operation-2            NotSet  0,00 s
```

Figure 77: .tapplan Example



Test Block Number	Test Name	Status	Number of Fails
1	[1000] Mathematical Operations	Fail	1
2	[2000] String Operations	Pass	0

Figure 78: Report Result Example

User Configuration

Several types of users with different permissions are defined in TestPlanner.

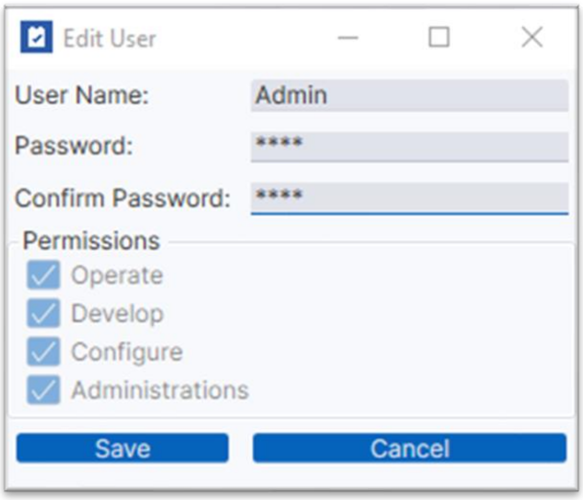
- ⇨ **Operators** are only allowed to run and debug test plans.
- ⇨ **Developers** can make changes to test steps and the test plan, as well as modify the configuration.
- ⇨ **Admins** can add new users and change their permission.

To change user configuration in TestPlanner:

1. Navigate to **File > Configure Users** in Menu Bar.

Note Default passwords for all users are empty.

2. Click on the **Edit** button (marked with a pen icon) to edit a user's settings.



Edit User

User Name: Admin

Password: ****

Confirm Password: ****

Permissions

- ✓ Operate
- ✓ Develop
- ✓ Configure
- ✓ Administrations

Figure 79: User Configuration

Note The permissions for the user can be seen below.

3. Change the password by typing the new password to given boxes.
4. Click on the **Save** button to save the changes, or the **Cancel** button to discard them.

To add/delete a user in TestPlanner:

- ⇨ Right click on a user and click on **Add User** to add a new user.

Note User types that are created this way can have their permissions changed.

- ⇨ Click on the minus icon next to users to delete a user type.

Note Note that only non-default users can be deleted this way.

DEICO

Contact

DEICO Head Office

Teknopark Ankara, Serhat Mah.,
2224 Cad., No:1 F Blok, Z-12,
Yenimahalle, Ankara, Türkiye

support@deico.com.tr

+90 312 395 68 44

www.deico.com.tr

DEICO