Getting started with DevOps Test Embedded for Eclipse IDE

Special notice

Before using this information and the product it supports, read the information in Notices on page xxxii.

Contents

Chapter 1. Overview

HCL DevOps Test Embedded (**Test Embedded**) is delivered with some examples. For the HCL DevOps Test Embedded for Eclipse IDE (Test Embedded for Eclipse IDE), they are in the `<installation folder>/examplesEclipse`. The following guide uses the example MUIproj to display various features. The guide covers the following features:

- The application build
- Code coverage
- MISRA rules review
- The call graph visualization
- The test generation
- The stub creation
- The execution of the test

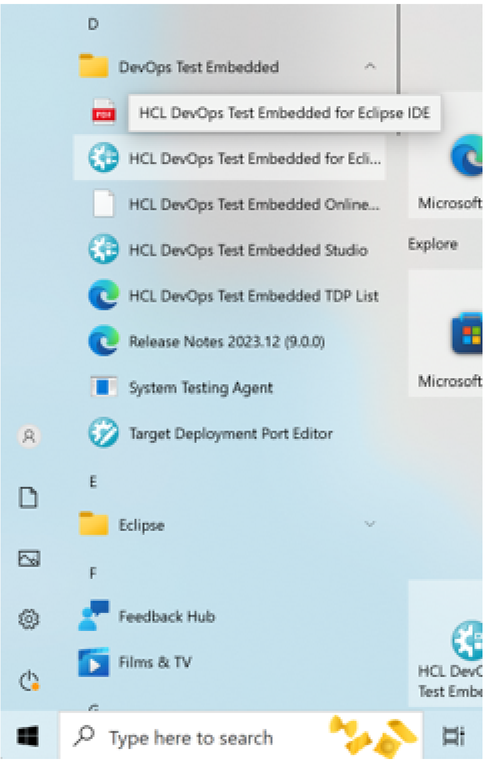
Chapter 2. Importing the project in DevOps Test Embedded for Eclipse IDE

You can either create a new project or you can import your existing Eclipse projects in your HCL DevOps Test Embedded (**Test Embedded**) workspace.

Starting Test Embedded for Eclipse IDE

On Windows:

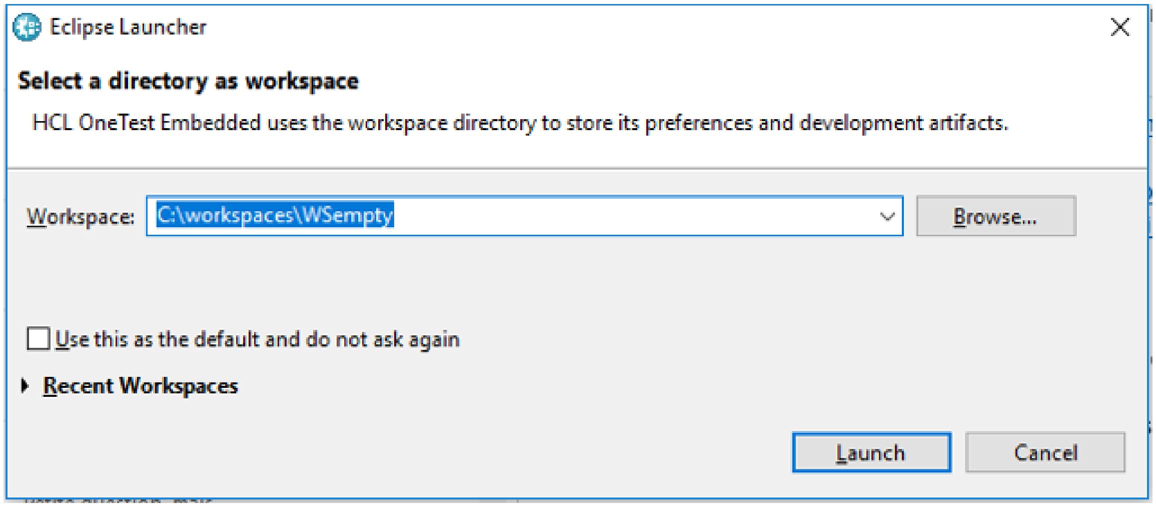
- Before stating, set the environment variables *HCL_LICENSING_URL* and *HCL_LICENSING_ID* with the information provided by HCL.
- Open the Windows start menu and select from the menu HCL DevOps Test Embedded for Eclipse IDE in the group DevOps Test Embedded


D
DevOps Test Embedded
HCL DevOps Test Embedded for Eclipse IDE
HCL DevOps Test Embedded for Ecli...
HCL DevOps Test Embedded Online...
HCL DevOps Test Embedded Studio
HCL DevOps Test Embedded TDP List
Release Notes 2023.12 (9.0.0)
System Testing Agent
Target Deployment Port Editor
E
Eclipse
F
Feedback Hub
Films & TV
Microsoft
Explore
Microsoft
HCL Dev
Test Embe
Type here to search

On Linux

- In the installation folder, edit the file `testrtinit.sh` and update the following environment variables:
  - *TESTRTDIR* with the correct installation folder
  - *HCL_LICENSING_URL* and *HCL_LICENSING_ID* with the information provided by HCL
- Run the command: `.testrtinit.sh`
- Then run the command: `.start_visualtest.sh &`

Create your own workspace. Alternatively you can select an existing workspace.


Eclipse Launcher

Select a directory as workspace

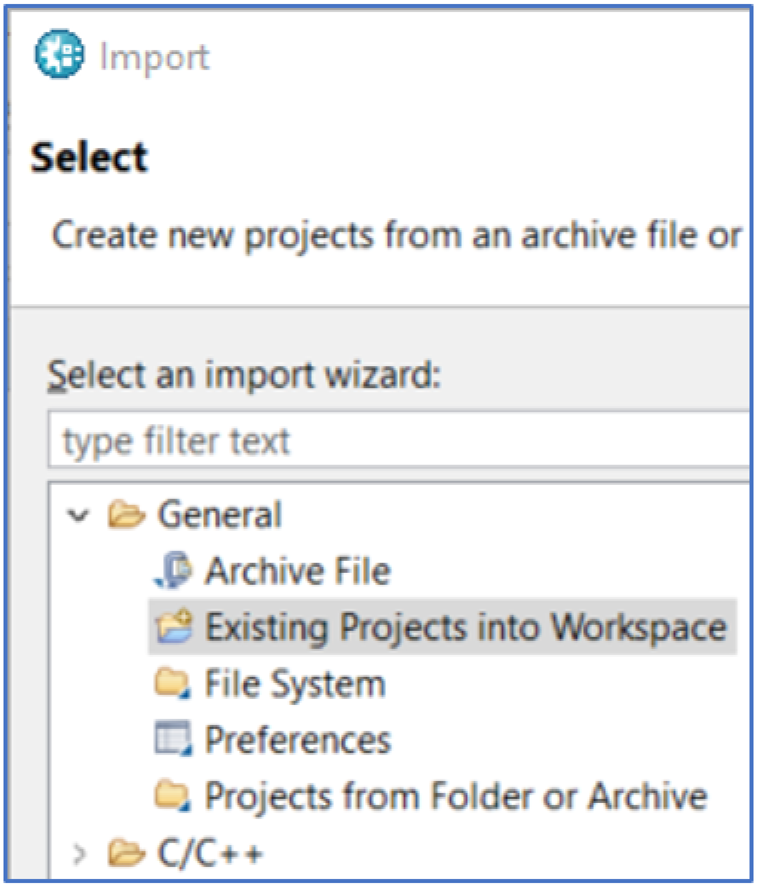
HCL OneTest Embedded uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\workspaces\WSempty

□ Use this as the default and do not ask again

Importing the project MUIproj

1. Select the menu **File > Import**.
2. Select **General > Existing Projects into Workspace** in the opened wizard, and then click **Next**.



Import

Select

Create new projects from an archive file or

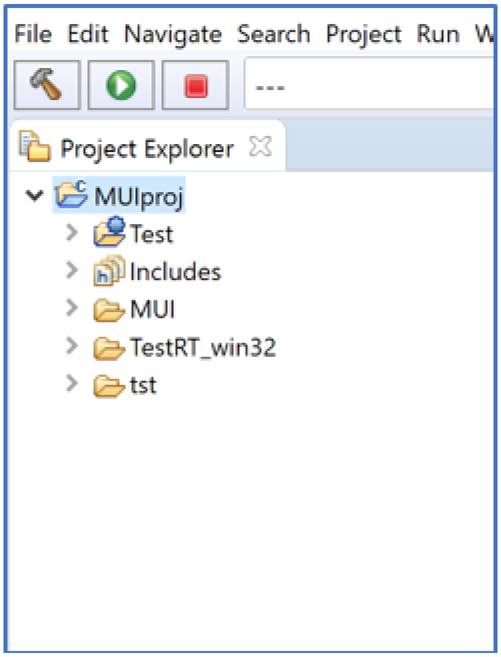
Select an import wizard:

type filter text

General
Archive File
Existing Projects into Workspace
File System
Preferences
Projects from Folder or Archive
C/C++

3. Click **Select archive file**, and then click **Browse...** on the same line and select the file `MUIproj.zip` in the folder **`<installation folder>/examplesEclipse`**.
4. Click **Finish**.

A new project `MUIproj` is created. You can see it in the project explorer (if this view is not already open, you can open it by selecting the menu **Window > Show View > Project Explorer**.



Chapter 3. Building and running the application

DevOps Test Embedded comes with many Target Deployment Port (TDP) for different compilers. For more information about Target Deployment Port, see Target deployment port overview.

This project was initially designed for **C Visual Studio 2019.** If you do not have this compiler installed or if you are on Linux, you need to modify the Target Deployment Port accordingly.

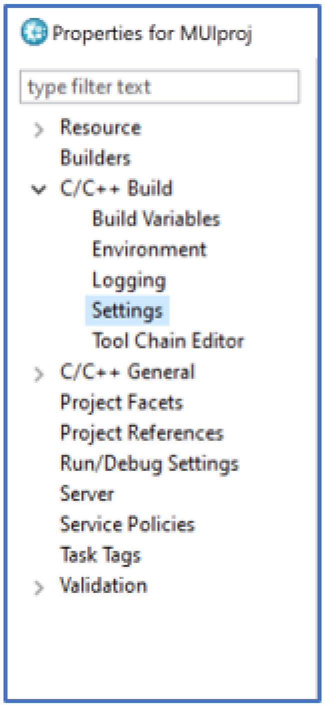
When you install DevOps Test Embeddedon your computer, the installer checks your compilers installation and create the following TDP for you:

- **C GNU** if it finds a gcc native compiler (Cygwin or MinGW)
- **C Visual** if it finds a Microsoft Visual compilers

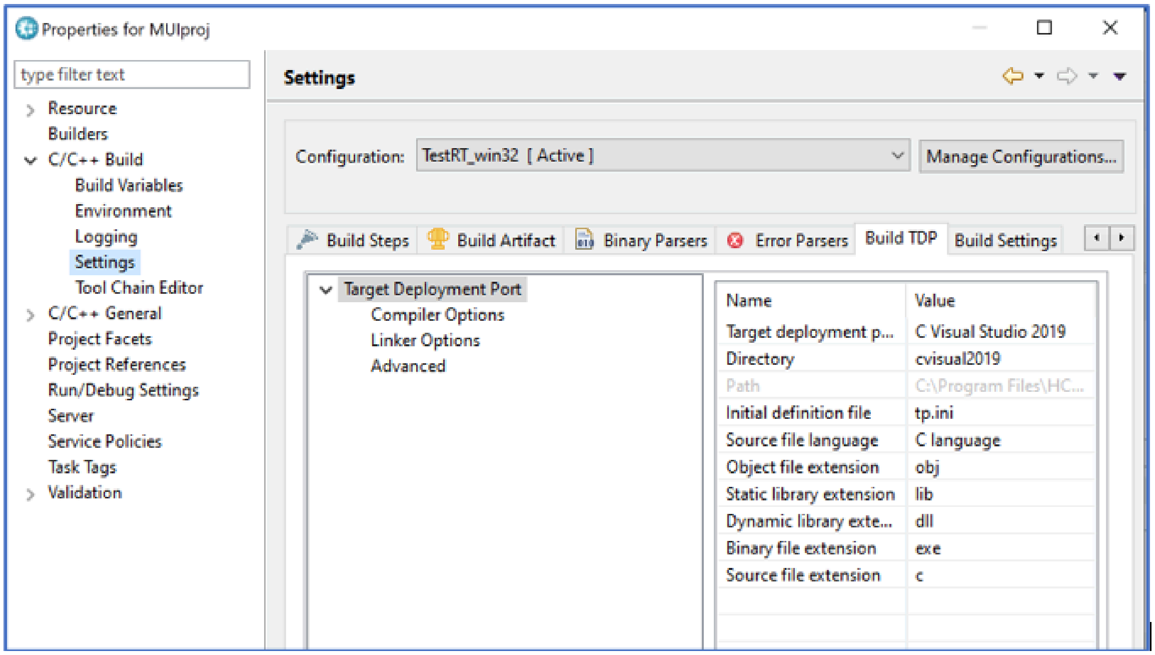
  You can update this project with any one of them.

Modifying the TDP in the settings

1. Right click the project **MUIproj** and select from the menu **Properties** in the Project Explorer.
2. Select **C/C++ Build > Settings** from the left menu tree in the wizard.



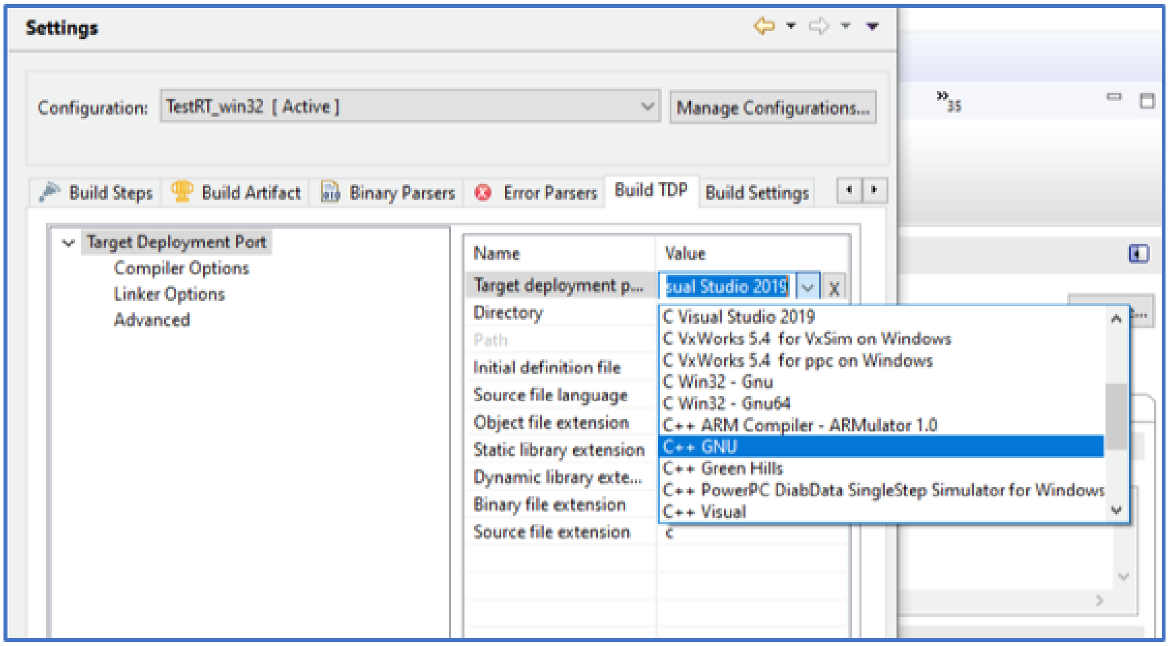
3. Alternatively, you can click the settings icon ⚙ on the toolbar (make sure that you are in the perspective **C/C++** or **DevOps Test Embedded**). This open the Properties settings directly.
4. In the right panel of the wizard, select the tab **Build TDP** (if this tab is not displayed, increase the width of the wizard or use the right arrow to make it appear).



Configuration: TestRT_win32 [Active]

Name	Value
Target deployment p...	C Visual Studio 2019
Directory	cvisual2019
Path	C:\Program Files\HC...
Initial definition file	tp.ini
Source file language	C language
Object file extension	obj
Static library extension	lib
Dynamic library exte...	dll
Binary file extension	exe
Source file extension	c

5. In the **Target Deployment Port property,** click **C Visual Studio 2019** and select **C GNU** from the drop-down list (or **C Visual** depending on the compiler you have on your computer).



Configuration: TestRT_win32 [Active]

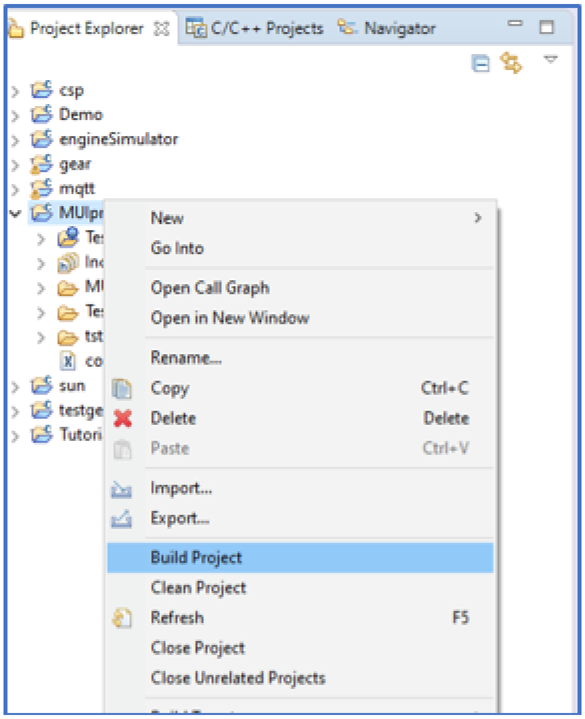
Name	Value
Target deployment p...	sual Studio 2019
Directory	
Path	
Initial definition file	
Source file language	
Object file extension	
Static library extension	
Dynamic library exte...	
Binary file extension	
Source file extension	c

6. Click **Apply**, and then **Close**.

Building the project

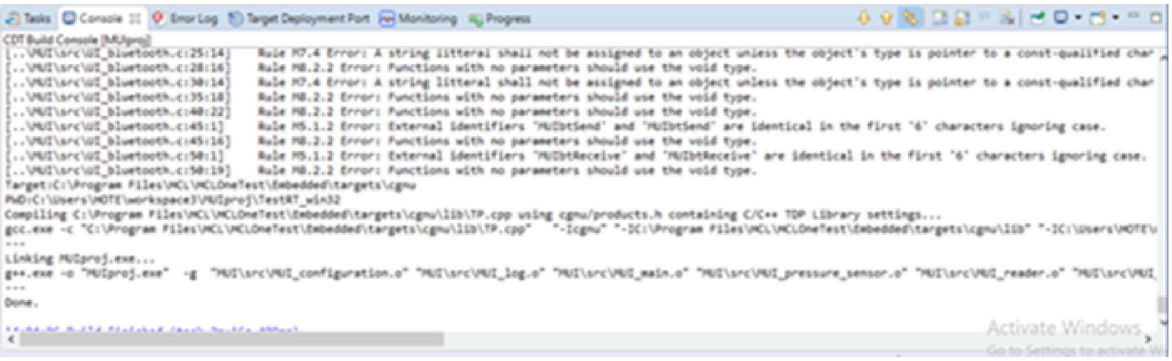
Now this project can be built by using this TDP. The default configuration involves performing the build with both Coverage and MISRA Code Review options enabled.

1. Right click the project **MUIproj** and select the option **Build Project** from the menu in the **Project Explorer**.



The console view displays the build log. The build must successfully complete up to the linking phase.

Note: It can take some minutes to complete.



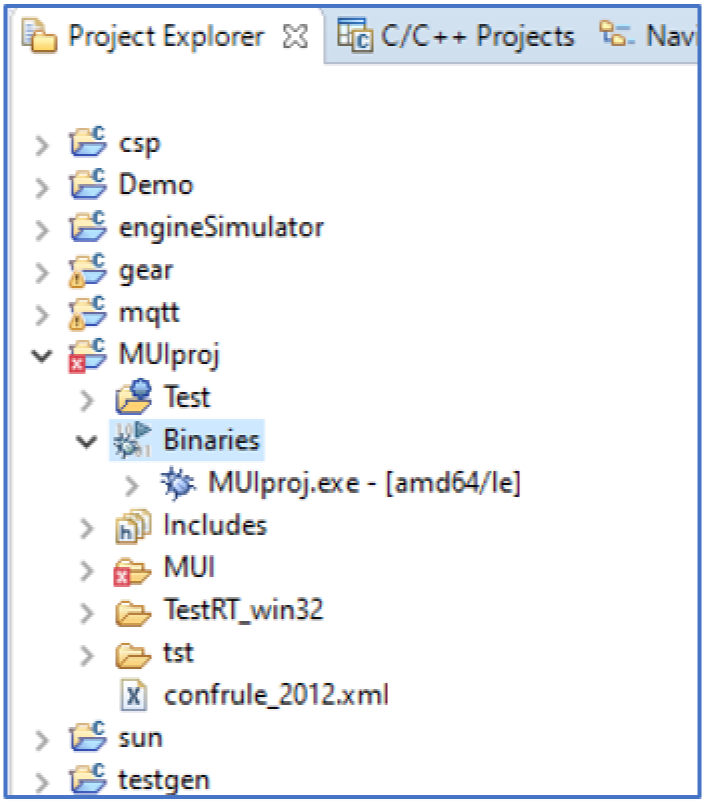
```
[..\MUI\src\UI_bluetooth.c:25:14]    Rule M7.4 Error: A string litteral shall not be assigned to an object unless the object's type is pointer to a const-qualified char
[..\MUI\src\UI_bluetooth.c:28:16]    Rule M8.2.2 Error: Functions with no parameters should use the void type.
[..\MUI\src\UI_bluetooth.c:30:14]    Rule M7.4 Error: A string litteral shall not be assigned to an object unless the object's type is pointer to a const-qualified char
[..\MUI\src\UI_bluetooth.c:35:18]    Rule M8.2.2 Error: Functions with no parameters should use the void type.
[..\MUI\src\UI_bluetooth.c:40:22]    Rule M8.2.2 Error: Functions with no parameters should use the void type.
[..\MUI\src\UI_bluetooth.c:45:1]     Rule M5.1.2 Error: External identifiers 'MUIbtSend' and 'MUIbtSend' are identical in the first '6' characters ignoring case.
[..\MUI\src\UI_bluetooth.c:45:16]    Rule M8.2.2 Error: Functions with no parameters should use the void type.
[..\MUI\src\UI_bluetooth.c:50:1]     Rule M5.1.2 Error: External identifiers 'MUIbtReceive' and 'MUIbtReceive' are identical in the first '6' characters ignoring case.
[..\MUI\src\UI_bluetooth.c:50:19]    Rule M8.2.2 Error: Functions with no parameters should use the void type.
Target:C:\Program Files\HCL\HCLOneTest\Embedded\targets\cgnu
PWD:C:\Users\HOTE\workspace3\MUIproj\TestRT_win32
Compiling C:\Program Files\HCL\HCLOneTest\Embedded\targets\cgnu\lib\TP.cpp using cgnu/products.h containing C/C++ TDP Library settings...
gcc.exe -c "C:\Program Files\HCL\HCLOneTest\Embedded\targets\cgnu\lib\TP.cpp"   "-Icgnu" "-IC:\Program Files\HCL\HCLOneTest\Embedded\targets\cgnu\lib" "-IC:\Users\HOTE\
...
Linking MUIproj.exe...
g++.exe -o "MUIproj.exe"  -g  "MUI\src\MUI_configuration.o" "MUI\src\MUI_log.o" "MUI\src\MUI_main.o" "MUI\src\MUI_pressure_sensor.o" "MUI\src\MUI_reader.o" "MUI\src\MUI_
...
Done.

[illegible]
```

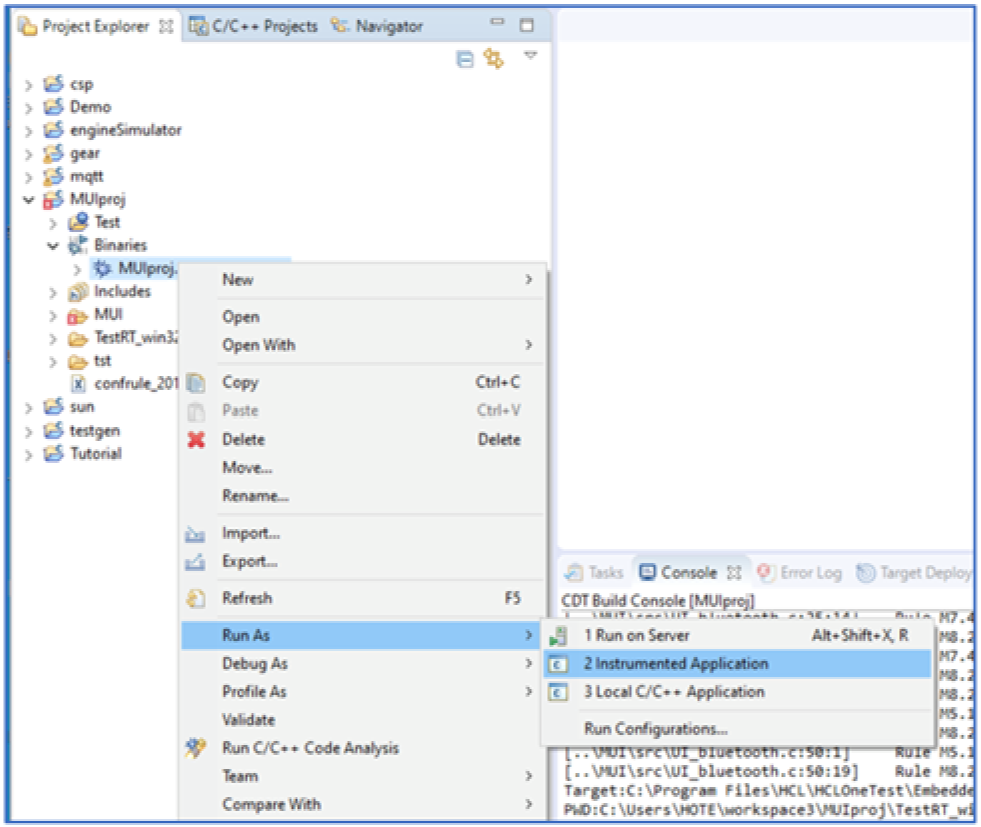
Running the application

This application contains a simple main that can be run.

1. Open the node **Binaries** in the **MUIproj** project In the project explorer.



2. Right-click **MUIproj.exe** and from the menu select **Run As** > **Instrumented Application**.



CDT Build Console [MUIproj]

```
[..\MUI\src\UI_bluetooth.c:50:1]     Rule M5.1
[..\MUI\src\UI_bluetooth.c:50:19]    Rule M8.2
Target:C:\Program Files\HCL\HCLOneTest\Embedde
PWD:C:\Users\HOTE\workspace3\MUIproj\TestRT_wi
```

This **Instrumented Application** option runs the just-compiled application and then launch different tools to generate reports depending on the settings.

Note: If Eclipse prompts you in case of errors in the project, ignore it and click **Yes.**

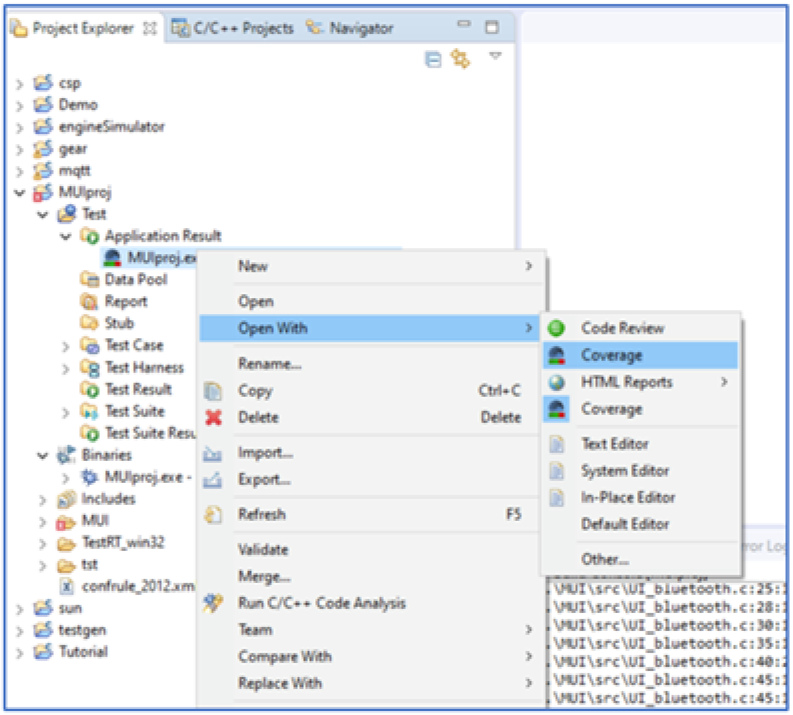
Viewing code coverage report

After the test run, all the reports are collected in a single file that you can find under the virtual node **Test**.

1. In the **Project Explorer,** expand the node **Test** > **Application Result** in the **MUIproj** project.

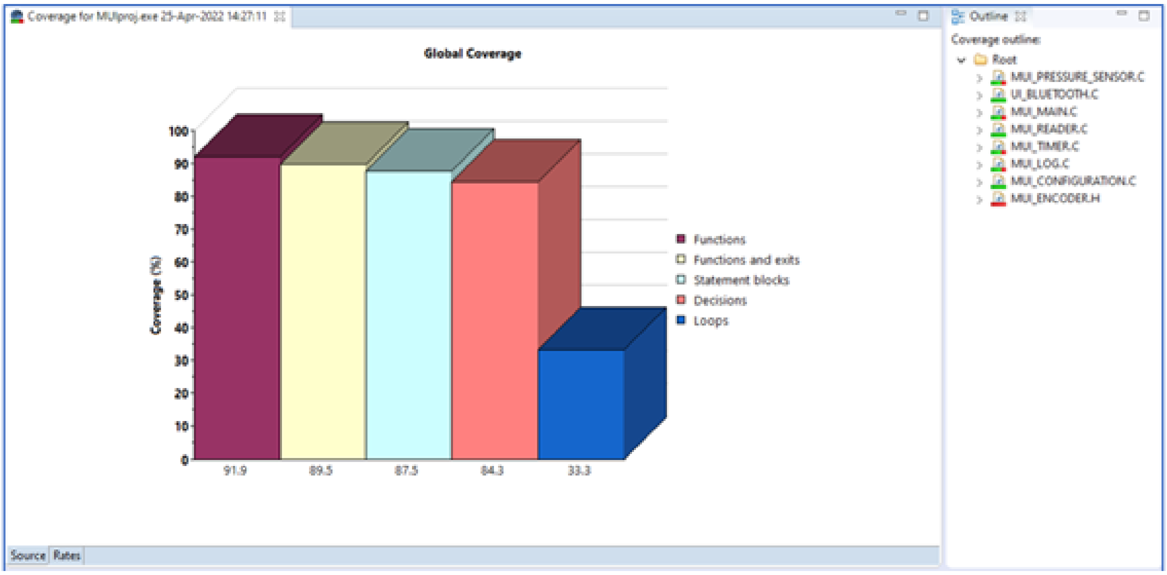
   There is a new file called `MUIproj.exe` with the date of the test run and a status.

2. Right-click the file and from the menu, select the option **Open With** > **Coverage**.



```
\MUI\src\UI_bluetooth.c:25:
\MUI\src\UI_bluetooth.c:28:
\MUI\src\UI_bluetooth.c:30:
\MUI\src\UI_bluetooth.c:35:
\MUI\src\UI_bluetooth.c:40:
\MUI\src\UI_bluetooth.c:45:
\MUI\src\UI_bluetooth.c:45:
```

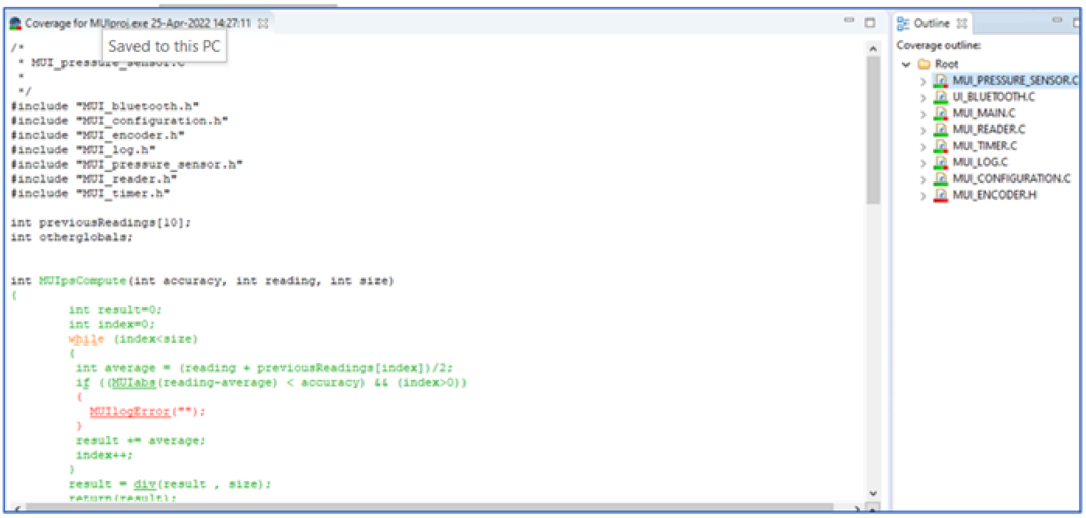
The Coverage viewer is opened and displays a graph with the different coverage level percentages. For more information, see Code coverage overview and Coverage levels.



The **Outline** option on the Coverage viewer allows you to navigate on the source code for each compilation unit.

3. Click on any one of them. A copy of the source code is now displayed with different colors:

- **Green**: the code is covered
- **Red**: the code is not covered
- **Orange**: the code is partially covered
- **Black**: no code



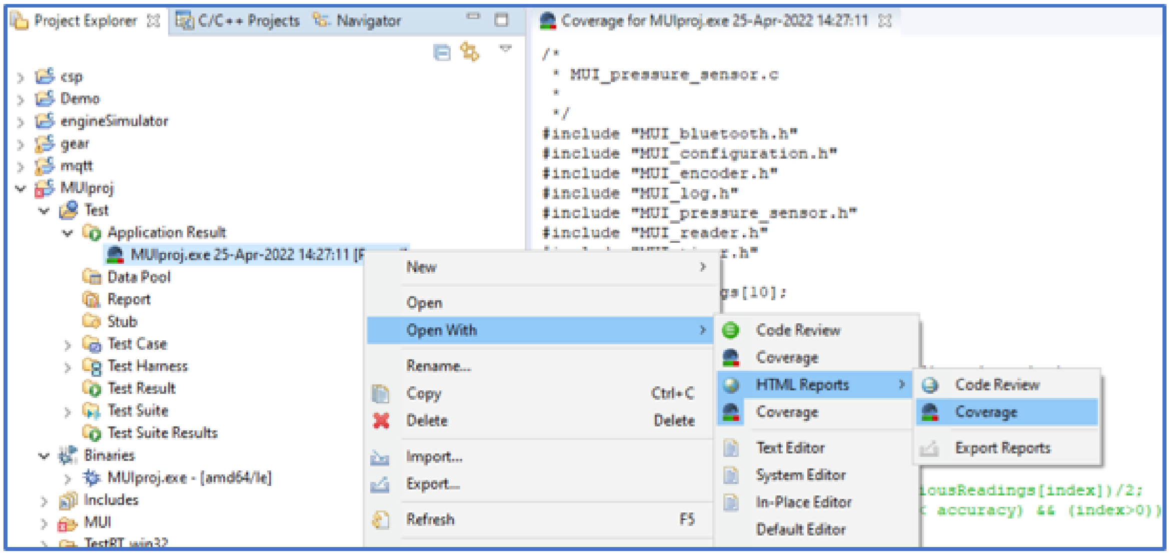
```
/*
 * MUI_pressure_sensor.c
 *
 */
#include "MUI_bluetooth.h"
#include "MUI_configuration.h"
#include "MUI_encoder.h"
#include "MUI_log.h"
#include "MUI_pressure_sensor.h"
#include "MUI_reader.h"
#include "MUI_timer.h"

int previousReadings[10];
int otherglobals;


int MUIpsCompute(int accuracy, int reading, int size)
{
        int result=0;
        int index=0;
        while (index<size)
        {
         int average = (reading + previousReadings[index])/2;
         if ((MUIabs(reading-average) < accuracy) && (index>0))
         {
           MUIlogError("");
         }
         result += average;
         index++;
        }
        result = div(result , size);
        return(result);
```

For more information, see Using the Code Coverage Viewer to view reports. A similar report exists in HTML format. You can open it in a browser:

4. Right click on the same file and from the menu, select the option **Open With** > **HTML Reports** > **Coverage**.

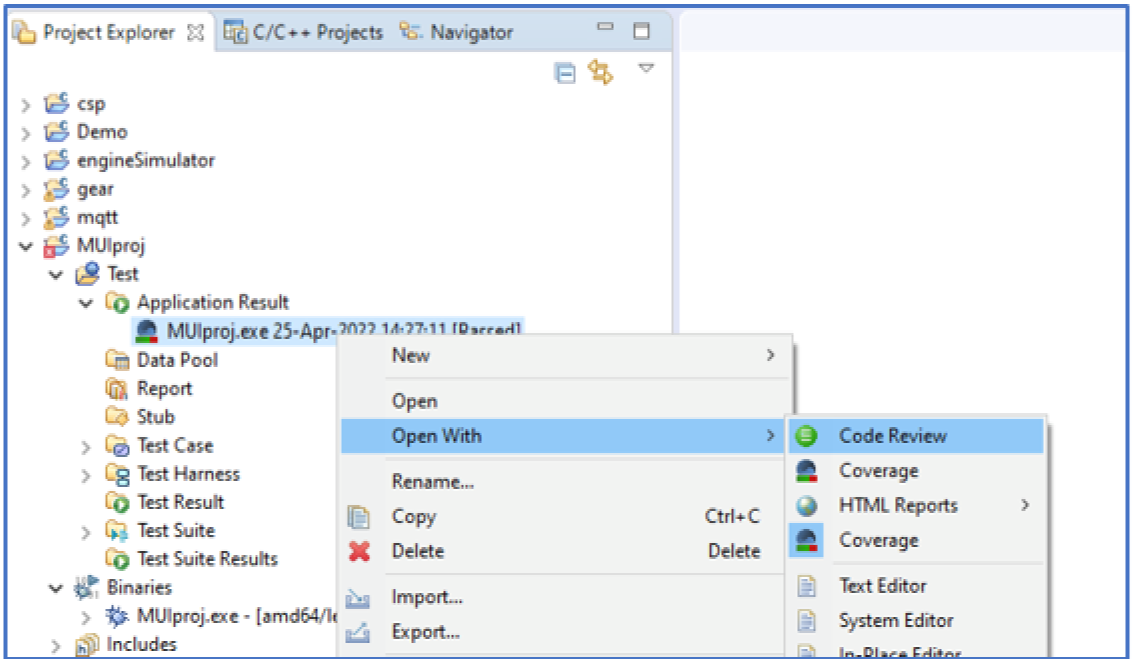


```
/*
 * MUI_pressure_sensor.c
 *
 */
#include "MUI_bluetooth.h"
#include "MUI_configuration.h"
#include "MUI_encoder.h"
#include "MUI_log.h"
#include "MUI_pressure_sensor.h"
#include "MUI_reader.h"
```

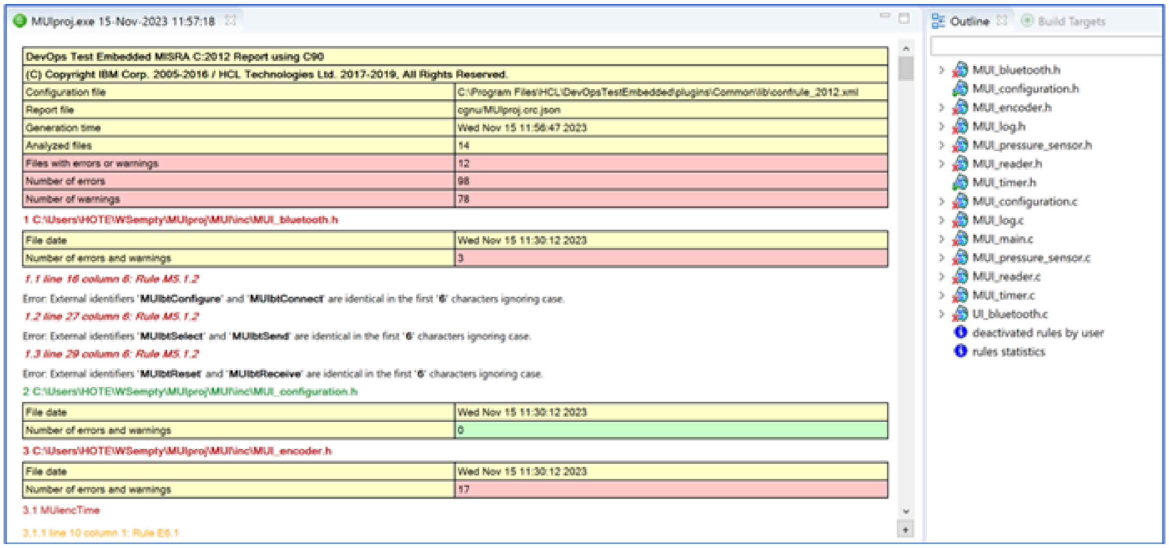
Viewing MISRA code review report

DevOps Test Embedded supports MISRA C 2004 rule, see Code review MISRA 2004, and MISRA C 2012, see Code review MISRA 2012 rules. You can open the report generated by this feature in the following way.

1. Right click the node **MUIproj.exe** in the **Application Result** node, and then select from the menu **Open With** > **Code Review** .



The code review viewer is opened. The outline view allows you to navigate to the different files of the application (.h and .c). The central panel displays the rules that is raised during the analysis for the selected file. If you click on a rule, the source code editor opens in the selected file, at the line where the error was found.



DevOps Test Embedded MISRA C:2012 Report using C90	
(C) Copyright IBM Corp. 2005-2016 / HCL Technologies Ltd. 2017-2019, All Rights Reserved.	
Configuration file	C:\Program Files\HCL\DevOpsTestEmbedded\plugins\Common\lib\confrule_2012.xml
Report file	cgnu\MUIproj.crc.json
Generation time	Wed Nov 15 11:56:47 2023
Analyzed files	14
Files with errors or warnings	12
Number of errors	98
Number of warnings	78

1 C:\Users\HOTE\WSempty\MUIproj\MUI\inc\MUI_bluetooth.h

File date	Wed Nov 15 11:30:12 2023
Number of errors and warnings	3

1.1 line 16 column 6: Rule M5.1.2

Error: External identifiers '**MUIbtConfigure**' and '**MUIbtConnect**' are identical in the first '**6**' characters ignoring case.

1.2 line 27 column 6: Rule M5.1.2

Error: External identifiers '**MUIbtSelect**' and '**MUIbtSend**' are identical in the first '**6**' characters ignoring case.

1.3 line 29 column 6: Rule M5.1.2

Error: External identifiers '**MUIbtReset**' and '**MUIbtReceive**' are identical in the first '**6**' characters ignoring case.

2 C:\Users\HOTE\WSempty\MUIproj\MUI\inc\MUI_configuration.h

File date	Wed Nov 15 11:30:12 2023
Number of errors and warnings	0

3 C:\Users\HOTE\WSempty\MUIproj\MUI\inc\MUI_encoder.h

File date	Wed Nov 15 11:30:12 2023
Number of errors and warnings	17

3.1 MUIencTime

3.1.1 line 10 column 1: Rule E6.1

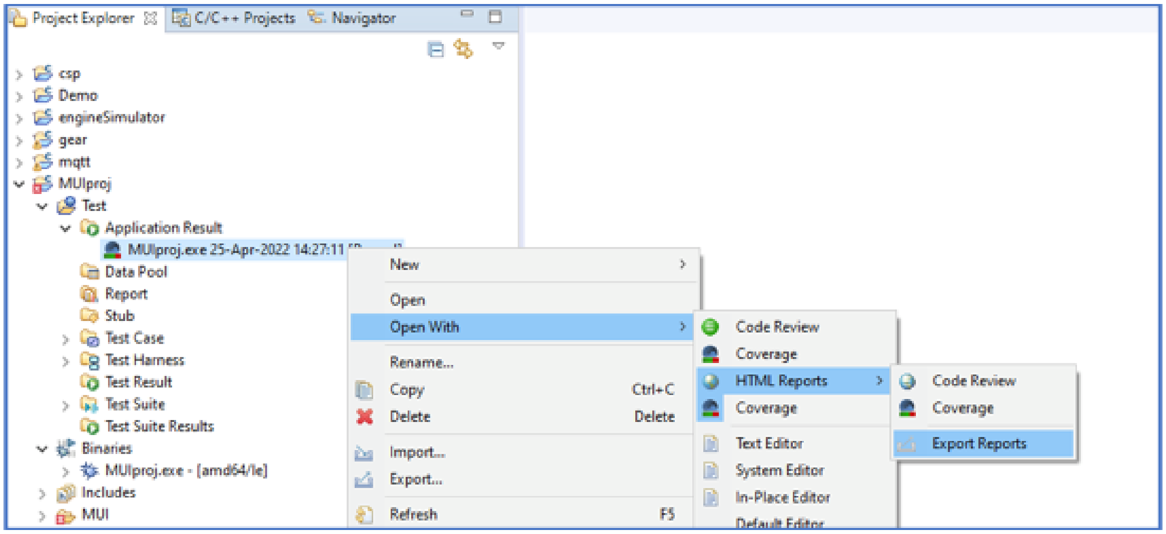
A similar report exists in HTML format. You can open it in a browser:

2. Right click on the same file and select from the menu **Open With** > **HTML Reports** > **Code Review** .

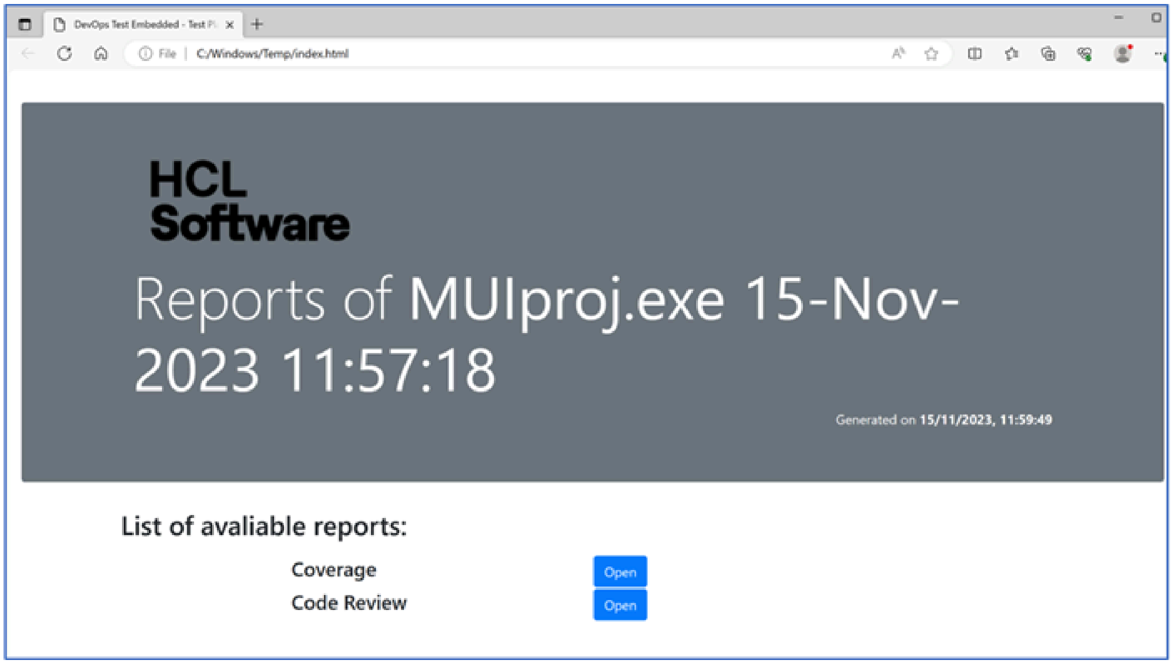
Exporting the HTML reports

All the reports, including the HTML reports, are stored in a single zip file. You can easily extract only the HTML reports in a folder with an index that lists all the exported files.

1. Right click on the result file and select the option from the menu **Open With** > **HTML Reports** > **Export Reports**



2. Select a folder on your disk, or create a new one and click **OK**. A browser opens with the following index:


DevOps Test Embedded - Test Pl
File | C:/Windows/Temp/index.html
HCL Software

Reports of MUIproj.exe 15-Nov-2023 11:57:18

Generated on 15/11/2023, 11:59:49

List of avaliable reports:

Coverage	Open
Code Review	Open

Chapter 4. Creating a test case

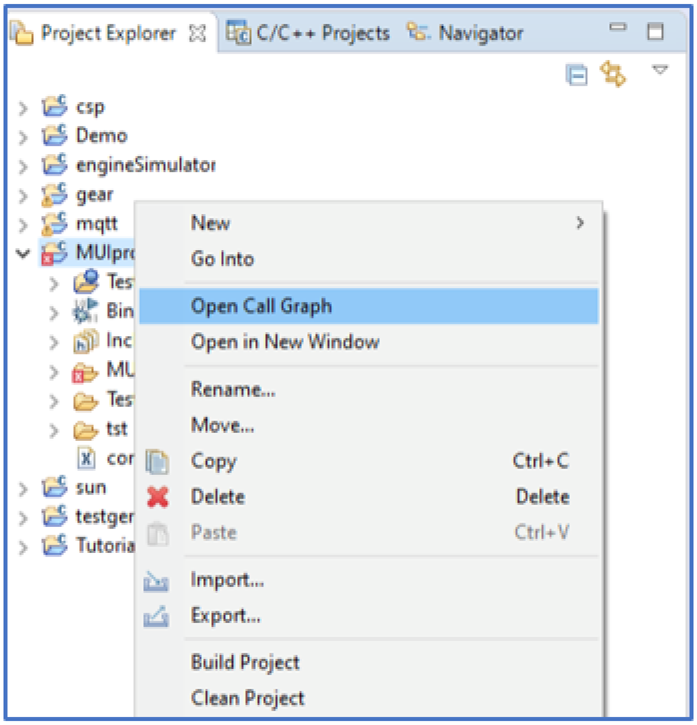
You can create a test case from the project by selecting a source file or a function. Each test case focuses on a particular function.

The following sections display how to create, update and run a test case with DevOps Test Embedded

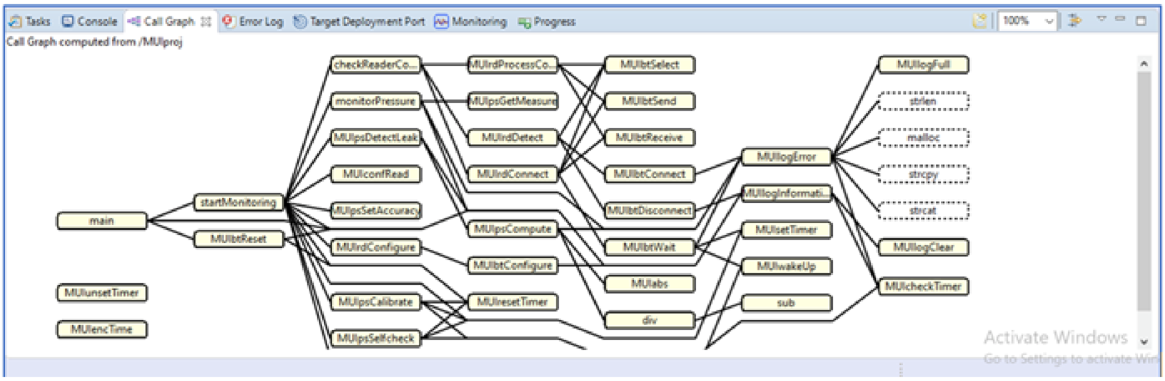
Opening the call graph

You have various ways to create a test case, and one method involves using the call graph that you can explore.

1. Right click the project **MUIproj** and from the menu, select **Open Call Graph**.

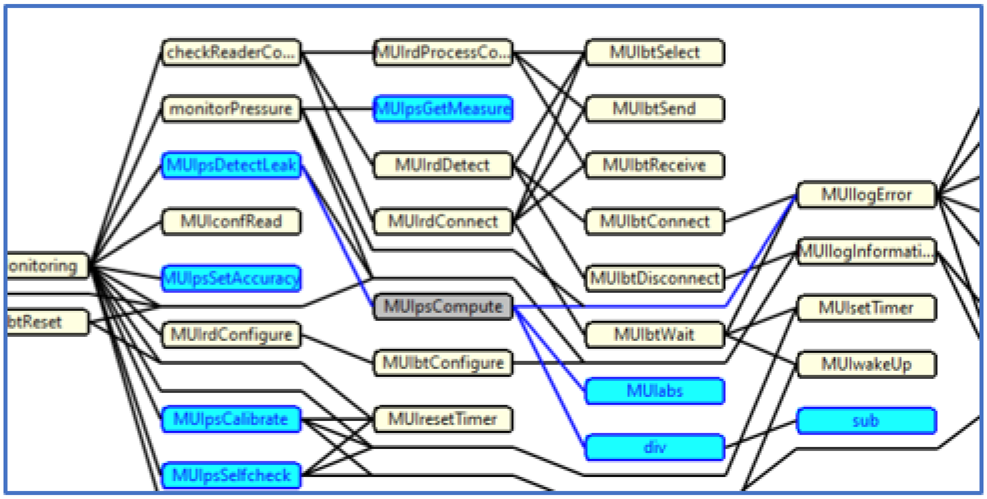


This action opens a new view that contains the call graph of the application:



The nodes in this graph represent the functions of the application. Nodes with dotted lines correspond to functions without available source code (in this example, these are functions in `libc`). The lines between two nodes are the calls between functions. The top level function is at the left of the graph (in this example it is the function `main`) and low level functions on the right.

2. Click a node inside the call graph (in the following example, the node **MUIpsCompute** is selected.



Now the call graph highlights the following:

- The grey node is the selected function
- The blue lines on the right of this node calls to other functions
- The blue lines on the left of this node are links to caller functions
- The blue nodes are functions that are in the same compilation unit

3. Double click on a node, the corresponding source code opens in the editor.

Creating a test

You can now create a test for the function **`div`**. DevOps Test Embedded, a test is composed of two parts:

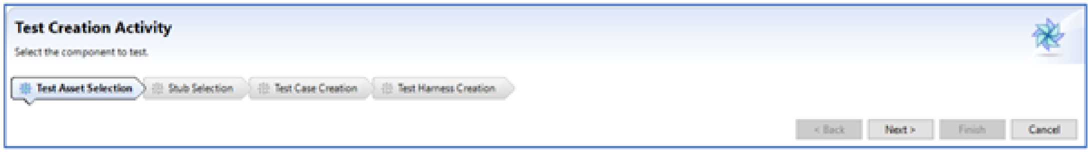
- A test case, is the test itself. The test contains the following:
  - The call of the function under test.
  - A table with the initial values and the expected values of the parameters, the global variables and local variables that you can add in this test case.
  - The stub behaviors relative to this test case.
  - Optionally, code that is added in the beginning (`#include` for example) and one or several requirements.
- A test harness is the container for test cases. One test harness can contain multiple test cases, defining how the test is built to become an executable program. A test harness:

- ◦ The list of the files under test.
- ◦ Additional source files that can be added to the test harness when linking. This option is useful for software integration test. It is also possible to add object files and libraries.
- ◦ The build settings.
- ◦ Optionally, code that is be added in the beginning (`#include` for example) and one or several requirements in such case, these requirements cover all the test cases of the test harness.

In DevOps Test Embedded, test cases and test harnesses are files with the extensions `.test_case` and `.test_harness`. There are compressed XML files.

1. Right-click the node **div** and select from the menu. the option **New test harness**

   A new panel appears on the call graph, called **Test Creation Activity**:

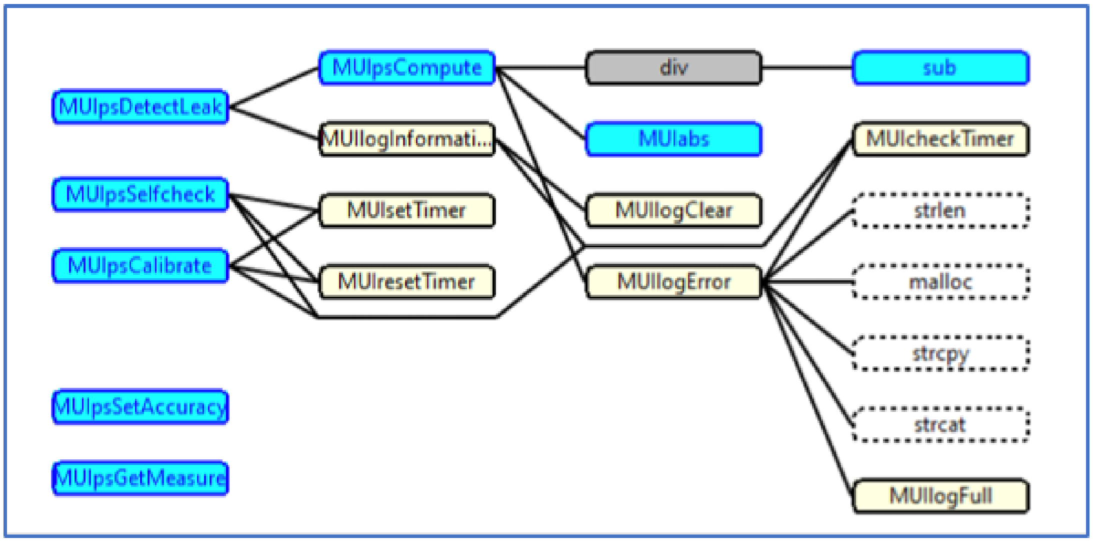
   

   Test Creation Activity

   Select the component to test.

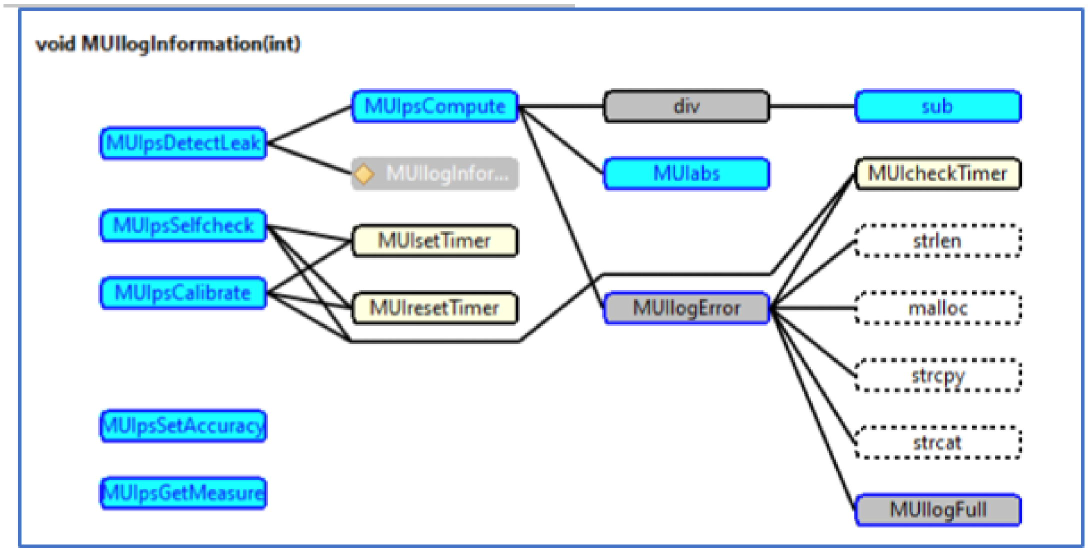
   This is a wizard that helps to create a first test case.

2. Click **Next.** This is the second page of the wizard. Now the call graph is reduced only to the functions that are in the same compilation unit (in blue) and the functions that are called by previous one.

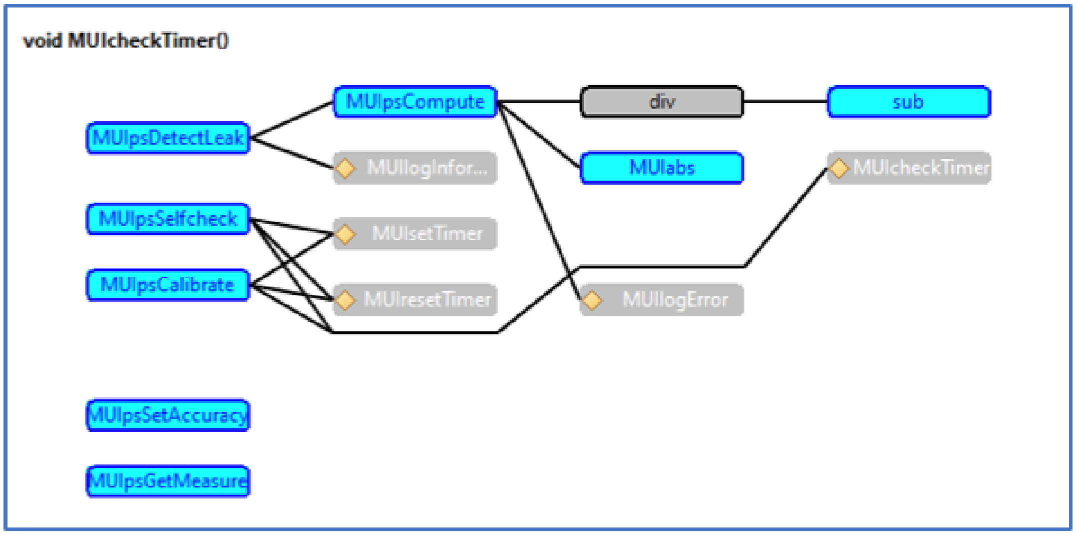
   The objective of this step is to take into account only the functions that are required in your test harness to link properly with your compilation unit without error. By default, the test harness is linked only with the compilation unit of the function under test. So, all the referenced functions (nodes in yellow) generates an error at the link phase as they are missing. The way to avoid that is to stub them.

3. Click the node **MUILogInformation**.



This node is now displayed as a stub, and the node `MUILogClear` disappears from the call graph because its caller will be stubbed.

4. Continue the sub selection by clicking the nodes **MUIsetTimer**, **MUIresetTimer, MUILogError** and **MUIcheckTimer**.



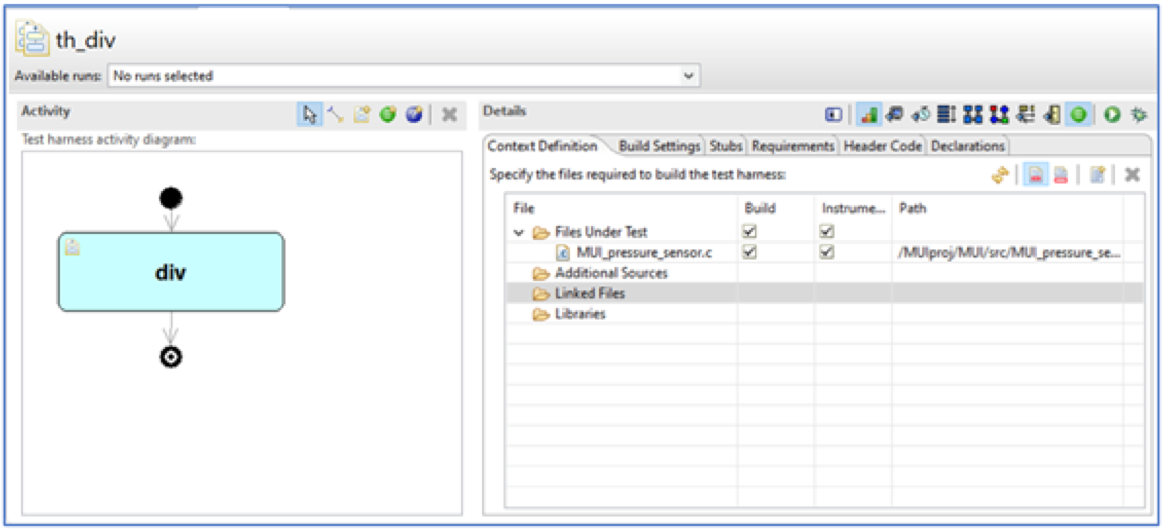
In the end, all the yellow nodes are suppressed, leaving only the blue ones.

5. Click **Next**. The next page is the test case name. By default, it is the name of the function under test. You can go with the default.
6. Click **Next**.The next page is the test harness name. By default, it is the name of the function under test with the suffix `th_`. You can go with the default.
7. Click **Finish**.

Editing a test

At the end of the wizard, a test harness that includes one test case is created. This test harness must be open. The test harness contains two parts:

- On the left panel, an activity diagram that allows to chain the test cases that is run.
- On the right panel, the configuration of the test harness.



Available runs: No runs selected

Specify the files required to build the test harness:

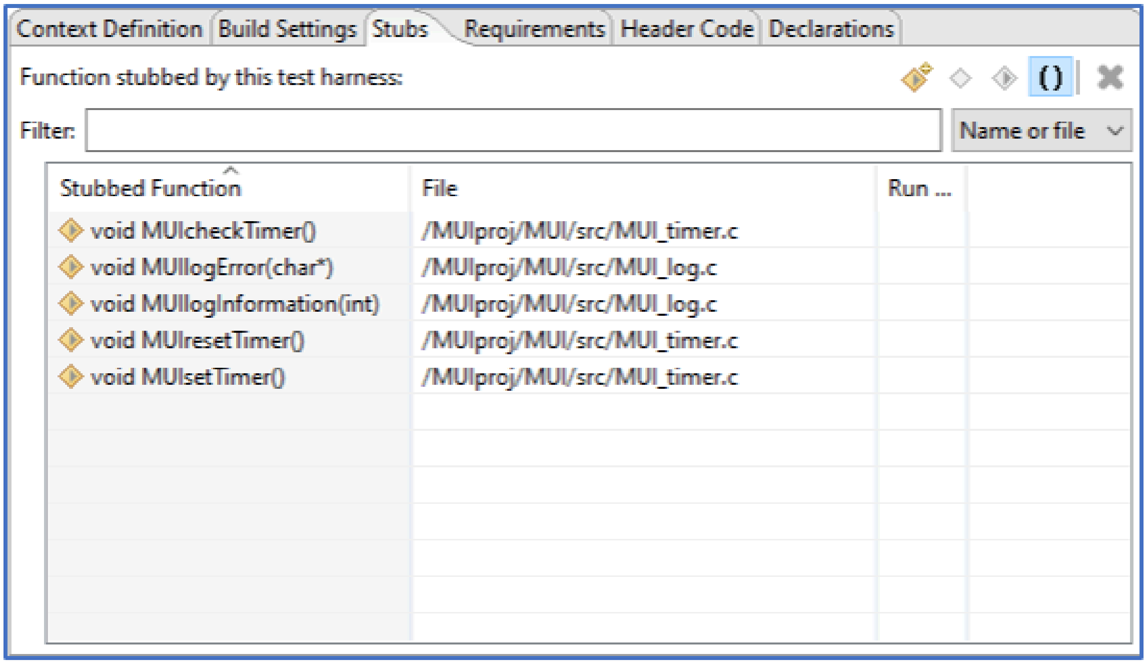
File	Build	Instrume...	Path
Files Under Test	☑	☑	
MUI_pressure_sensor.c	☑	☑	/MUIproj/MUI/src/MUI_pressure_se...
Additional Sources			
Linked Files			
Libraries			

You can go through on the following

- **Context definition**: Only one file is added as files under test, because you have stubbed all the external calls.

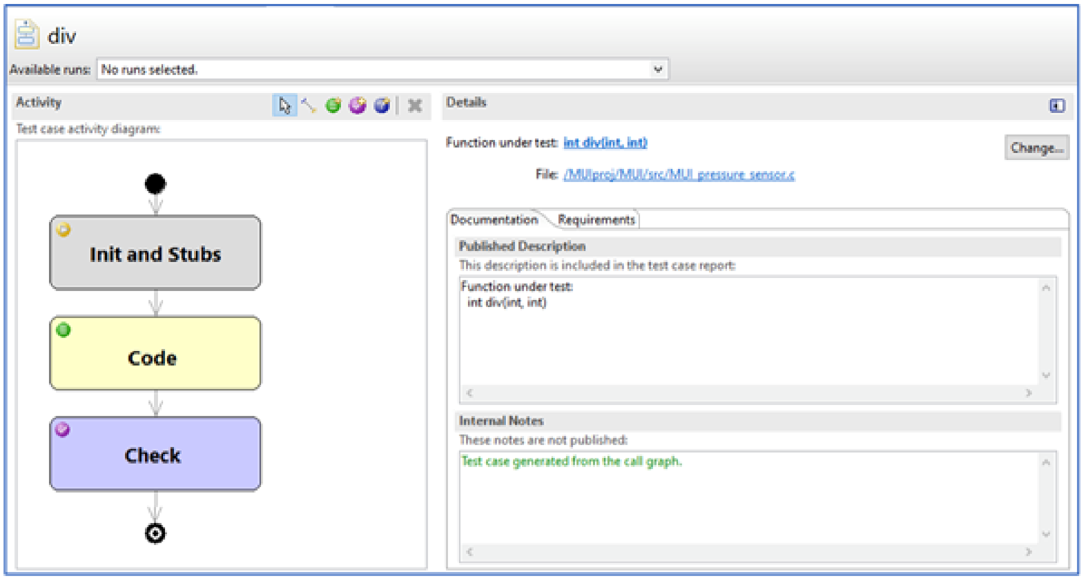
Note: This icon 📄 indicates that the files under test is included in the code generated for the test harness. This helps you to have the visibility on static variables and static functions that are hidden from external compilation units.

- **Stubs**: You can see here the five selected stubbed functions.



Stubbed Function	File	Run ...
void MUIcheckTimer()	/MUIproj/MUI/src/MUI_timer.c	
void MUIlogError(char*)	/MUIproj/MUI/src/MUI_log.c	
void MUIlogInformation(int)	/MUIproj/MUI/src/MUI_log.c	
void MUIresetTimer()	/MUIproj/MUI/src/MUI_timer.c	
void MUIsetTimer()	/MUIproj/MUI/src/MUI_timer.c	

1. Double click the **div** icon in the activity diagram to open the test case. The activity contains two parts:
   - On the left panel, an activity diagram that display the different phases of the test case
   - On the right panel, the configuration of the phase that is selected on the activity diagram (the default one when opening the test case is its general description, that you can open when clicking on the background of the activity diagram)


div
Activity
Test case activity diagram:
Init and Stubs
Code
Check
Details
Change...
Documentation
Requirements
Published Description
This description is included in the test case report:
Internal Notes
These notes are not published:

Available runs: No runs selected.

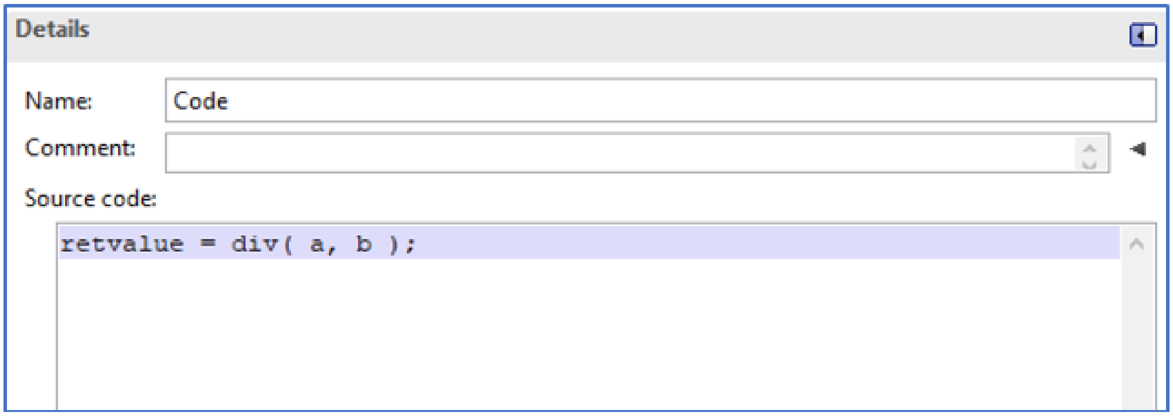
Function under test: int div(int, int)

File: /MUIproj/MUI/src/MUI_pressure_sensor.c

Function under test:
int div(int, int)

Test case generated from the call graph.

2. Click the **Code** box in the activity diagram: this shows you the generated code for calling the function under test. You can edit the generated code.



Name: Code

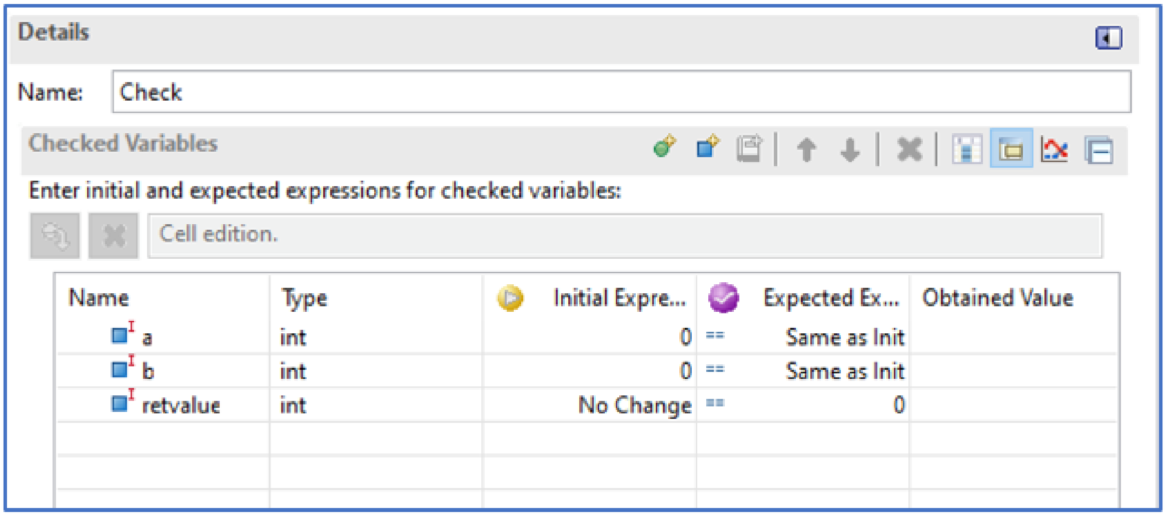
Comment:

Source code:

```
retvalue = div( a, b );
```

The variables *retValue*, *a* and *b* are created locally in the test harness and used as parameters.

3. Click the **Check** box in the activity diagram,



Details

Name: Check

Checked Variables

Enter initial and expected expressions for checked variables:

Cell edition.

Name	Type	Initial Expre...	Expected Ex...	Obtained Value
a	int	0	== Same as Init	
b	int	0	== Same as Init	
retvalue	int	No Change	== 0	

This table displays all the variables used for the test (parameters and global variables) with:

- Their type
- Their initial expression that represents the value before calling the function under test. **No Change** means that this variable is not initialized before calling the function under test.
- Their expected expression that represents the expected value after calling the function under test. **Same as Init** mean that the expected value is the same as the initialized value.
- Their obtained expression (i.e. the actual value after calling the function under test). This column is empty for now. It will be automatically filled after an execution.

Be default, the wizard generates a test case with all the parameters to 0 or null in case of pointers. You can modify this test case for having the following division 50/7 that should give 7 as result.

4. Click the cell **Initial expression** of the variable `a` and enter the value 50.
5. Press enter to validate this value, or click the icon .
6. Click the cell **Initial expression** of the variable *b*, enter the value 7 and validate it. Do not modify the expected result of `retValue` for now.

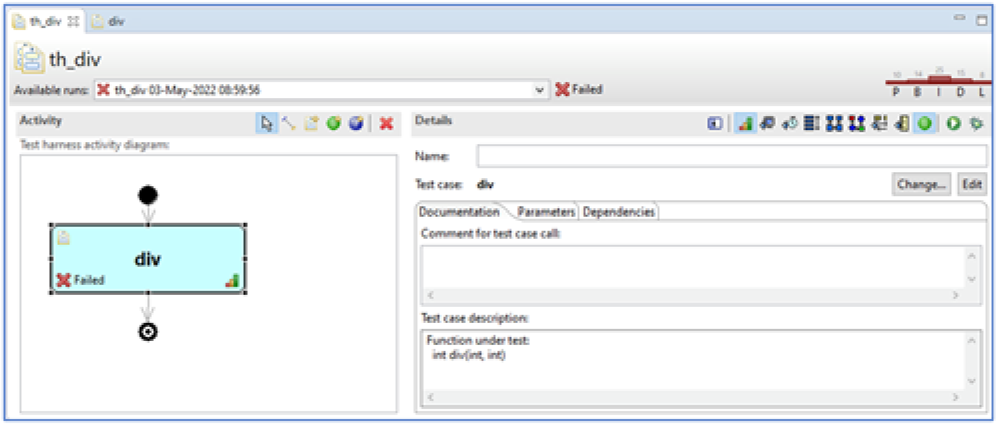
Running a test

You can run test harnesses or test scripts in a single step. You can update the list of test harnesses and test scripts to be run.

1. Go back to the test harness by clicking on the tab **th_div** in the editor panel.
2. Click the run icon on the details panel. The console view should display build log… This should take less than one minute.

Viewing a test result

After the test run is completed, the test harness editor is updated to display the following result:



th_div

Available runs: th_div 03-May-2022 08:59:56 Failed

Test harness activity diagram:

Name:

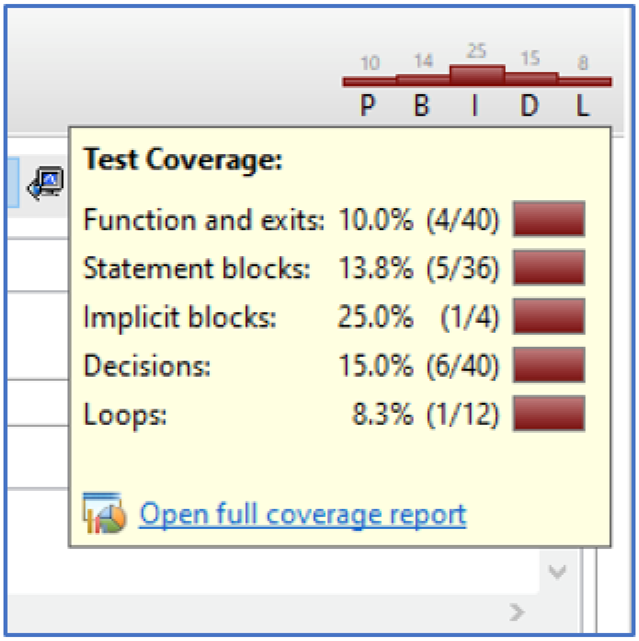
Test case: div

Comment for test case call:

Test case description:

Function under test:
 int div(int, int)

- The **Available runs** field is updated with the last test result and its status is displayed (**Failed** in this case).
- The activity diagram is updated with the status of each test case (**Failed** in this case).
- A coverage summary is displayed on the panel.
  1. You can hover over this coverage summary that displays a detailed information of the test result.

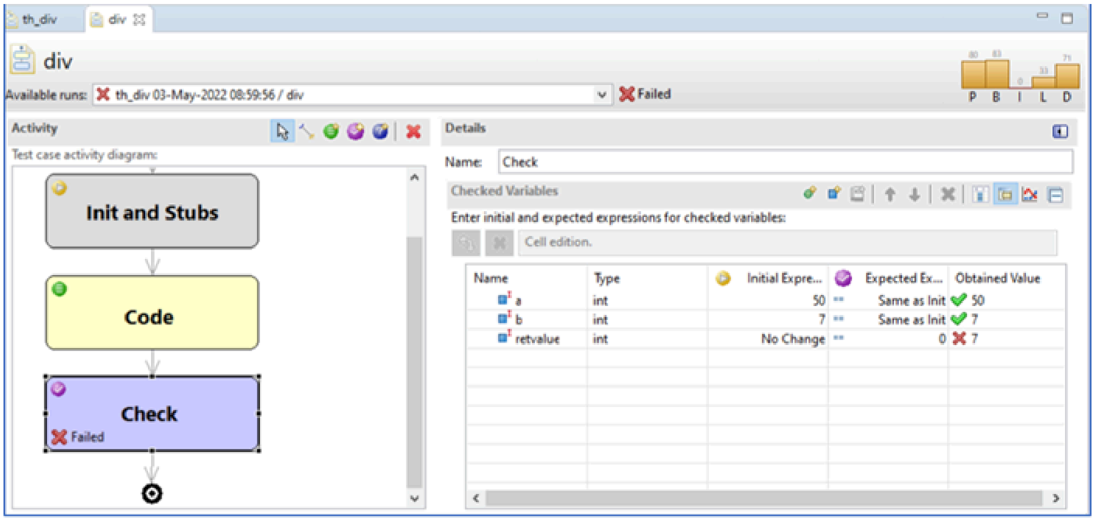


Test Coverage:

Function and exits:	10.0%	(4/40)	🟥
Statement blocks:	13.8%	(5/36)	🟥
Implicit blocks:	25.0%	(1/4)	🟥
Decisions:	15.0%	(6/40)	🟥
Loops:	8.3%	(1/12)	🟥

Open full coverage report

  2. Click the link **Open full coverage report** to open the coverage viewer with the information related to the file under test only.
  3. Double click the test case **div** inside the activity diagram to go to the test case editor that display additional information related to the last runs:



div

Available runs: ✗ th_div 03-May-2022 08:59:56 / div ✗ Failed

Test case activity diagram:

Name: Check

Checked Variables

Enter initial and expected expressions for checked variables:

Cell edition.

Name	Type	Initial Expre...		Expected Ex...	Obtained Value
a	int	50	==	Same as Init	✓ 50
b	int	7	==	Same as Init	✓ 7
retvalue	int	No Change	==	0	✗ 7

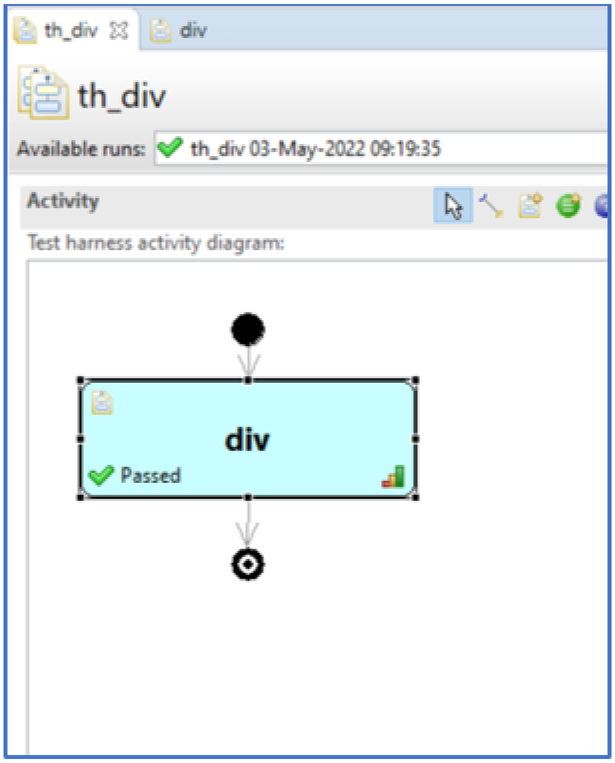
- The field **Available runs** is updated with the last execution result and the status of the test run is displayed (**Failed** in this case).
- The activity diagram is updated with the status **Failed** on the **Check** box
- A coverage summary is displayed on the panel.
- The column **Obtained value** is updated with the true values read during the execution and their status.

Fixing the test case

When you see the value of `retvalue` is wrong, you can perform the following:

1. Fix the expected value of `retvalue` to 7.
2. Save the test case **div** .
3. Go to the test harness and re-run the test.

The test is passed.



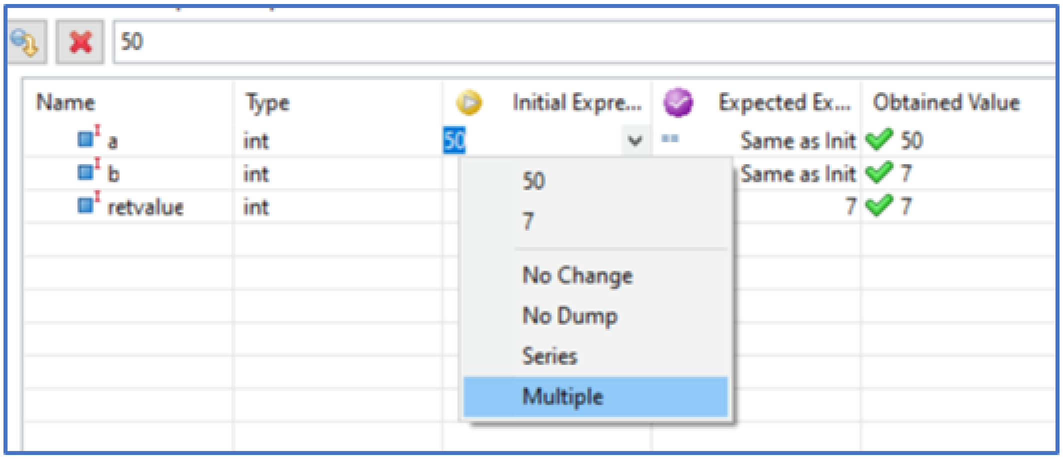
Updating the test case with multiple input values

DevOps Test Embedded allows you to define not only a single value but multiple values for an input variable, and also for the expected values of an output parameter. For multiple input values, you can:

1. Give a list of values from a minimum value to a maximum value with a defined step,
2. Give a list of values as a list,
3. Give a list of values that come from a datapool.

In the second case, illustrated here is with multiple values on parameter `a` that should modify the output value of `retvalue`.

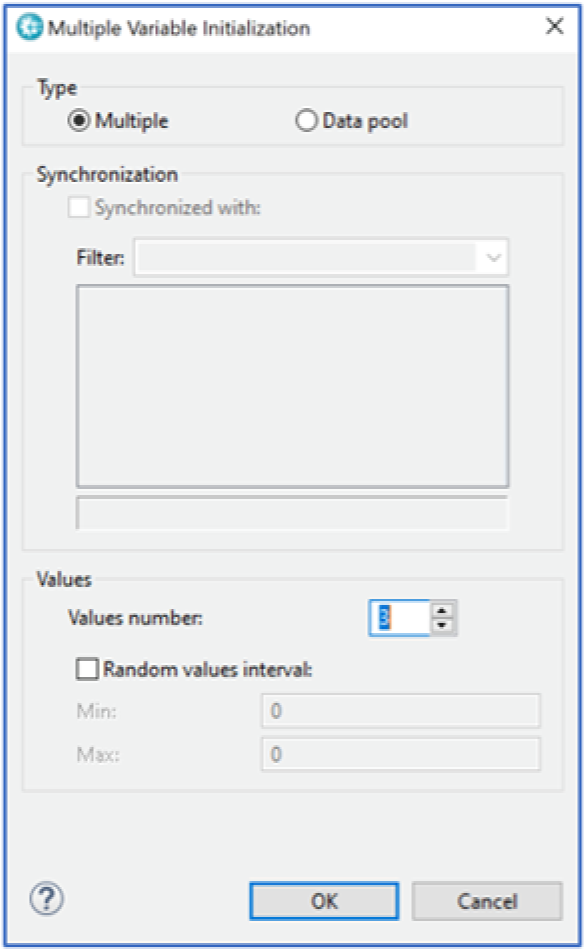
1. Click the **input Expression** of the variable `a` and select the option **Multiple**.



50

Name	Type	Initial Expre...	Expected Ex...	Obtained Value
a	int	50	Same as Init	✓ 50
b	int		Same as Init	✓ 7
retvalue	int		7	✓ 7

2. A **Multiple Variable Initialization** dialog box is opened.
3. Select the option **Multiple** option, and then select 3 in the **Values number**.



Multiple Variable Initialization

Type

● Multiple ○ Data pool

Synchronization

□ Synchronized with:

Filter:

Values

Values number: 3

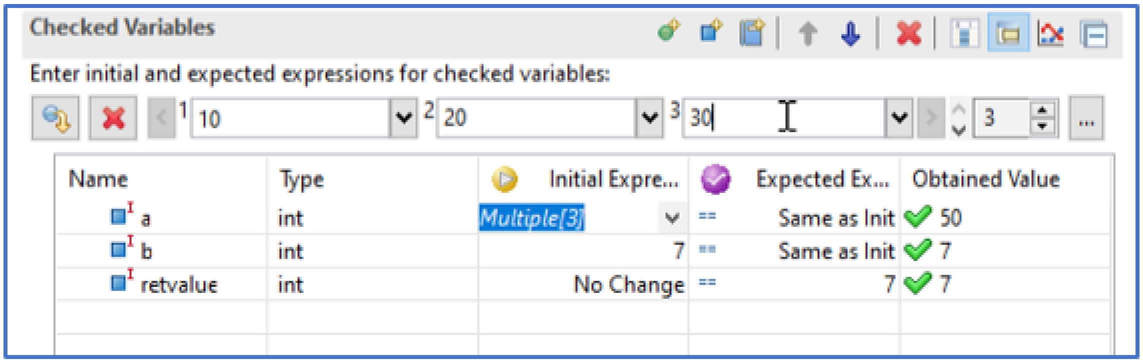
□ Random values interval:

Min: 0

Max: 0

4. Click **OK**.

A new bar appears on the table. Enter the values `10`, `20` and `30` in the three available fields.



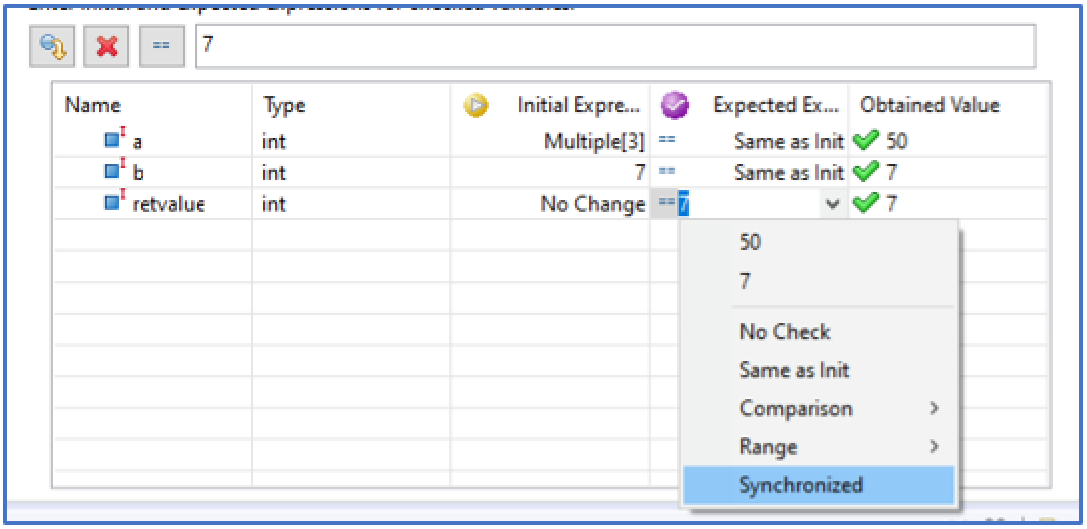
Checked Variables

Enter initial and expected expressions for checked variables:

1 10 · 2 20 · 3 30 · 3

Name	Type	Initial Expre...	Expected Ex...	Obtained Value
a	int	Multiple[3]	== Same as Init	✓ 50
b	int	7	== Same as Init	✓ 7
retvalue	int	No Change	== 7	✓ 7

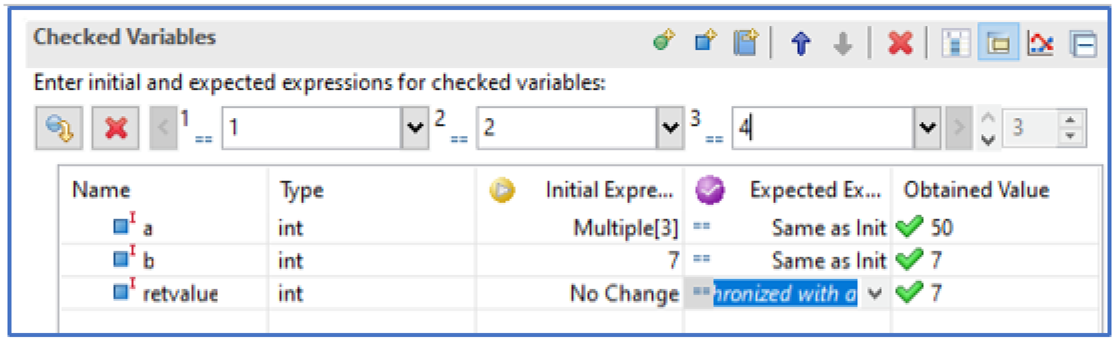
5. Press enter to validate these values, or click the icon .
6. Click the **Expected Expression** of the variable `retvalue` and select **Synchonized** from the menu.



== 7

Name	Type	Initial Expre...	Expected Ex...	Obtained Value
a	int	Multiple[3]	== Same as Init	✓ 50
b	int	7	== Same as Init	✓ 7
retvalue	int	No Change	==7	✓ 7

7. A new bar appears on the table. Enter the values `1`, `2` and `4` in the three available fields.



Checked Variables

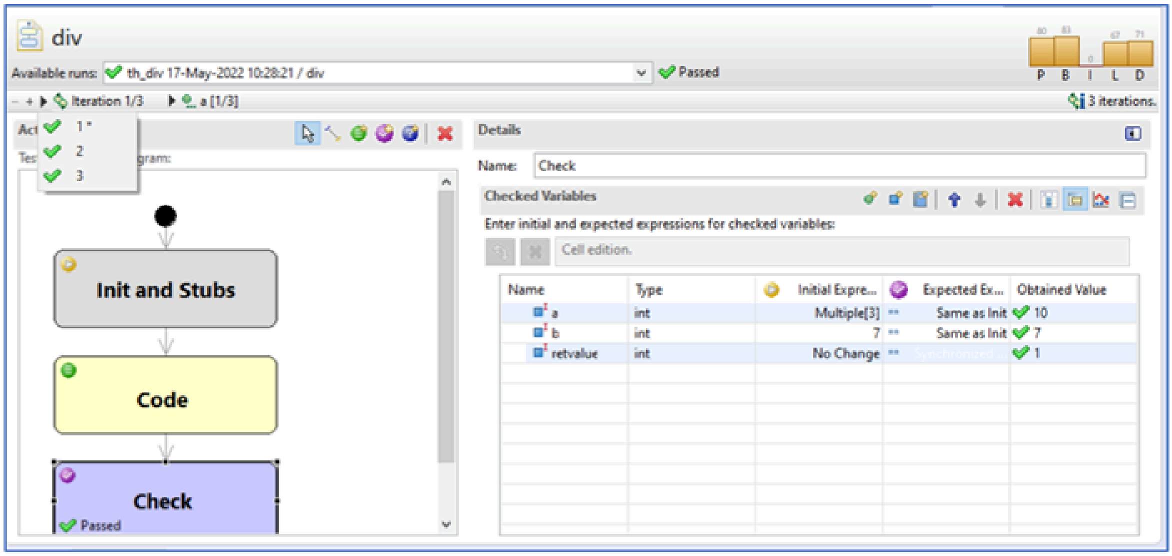
Enter initial and expected expressions for checked variables:

1 == 1 · 2 == 2 · 3 == 4 · 3

Name	Type	Initial Expre...	Expected Ex...	Obtained Value
a	int	Multiple[3]	== Same as Init	✓ 50
b	int	7	== Same as Init	✓ 7
retvalue	int	No Change	== ɔronized with a	✓ 7

8. Press enter to validate these values, or click the icon .

9. Run this test case. Go to the previous test harness and click the run icon ▶ on the details panel.
10. After the test run is completed, the test case is updated with the obtained values. You can navigate to the three iterations using the breadcrumb bar available on the test case:



div

Available runs: ✓ th_div 17-May-2022 10:28:21 / div ✓ Passed

Iteration 1/3 ▶ a [1/3] — 3 iterations.

✓ 1 *
✓ 2
✓ 3

Details

Name: Check

Checked Variables

Enter initial and expected expressions for checked variables:

Cell edition.

Name	Type	Initial Expre...		Expected Ex...	Obtained Value
a	int	Multiple[3]	==	Same as Init	✓ 10
b	int	7	==	Same as Init	✓ 7
retvalue	int	No Change	==	Synchronized ...	✓ 1

Notices

This document provides information about copyright, trademarks, terms and conditions for the product documentation.

This information was developed for products and services offered in the US.

HCL® may not offer the products, services, or features discussed in this document in other countries. Consult your local HCL® representative for information on the products and services currently available in your area. Any reference to an HCL® product, program, or service is not intended to state or imply that only that HCL® product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any HCL® intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL® product, program, or service.

HCL® may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

For license inquiries regarding double-byte character set (DBCS) information, contact the HCL® Intellectual Property Department in your country or send inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. HCL® may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-HCL® websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this HCL® product and use of those websites is at your own risk.

HCL® may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by HCL® under terms of the HCL® Customer Agreement, HCL® International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL® products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL® has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL® products. Questions on the capabilities of non-HCL® products should be addressed to the suppliers of those products.

Statements regarding HCL®'s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. HCL® shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

Trademarks

HCL®, the HCL® logo, and ibm.com® are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of HCL® or other companies.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the HCL® website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of HCL®.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of HCL®.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

HCL® reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by HCL®, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

HCL® MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.