CODESCROLL
CONTROLLER TESTER

Suresoft

CODESCROLL CONTROLLER TESTER MANUAL

Software for safe world

www.suresofttech.com

CODESCROLL
CONTROLLER TESTER

User Manual

3.7 — Last update: Jun 09, 2022

Suresofttech

Copyright © 2022 Suresofttech

Table of Contents

1. Before Starting **4**

2. Overview **5**

3. Install **7**

4. Uninstall **11**

5. Run **12**

6. Set License **14**

7. Set a Toolchain (Analyzer) **16**

7.1. Auto-registration of Toolchain 17

7.2. Add a Toolchain 18

7.3. Edit a Toolchain 21

7.4. Duplicate a Toolchain 22

7.5. Remove a Toolchain 23

7.6. Export a Toolchain 24

7.7. Import a Toolchain 26

8. Create a Project **27**

8.1. C/C++ Project with Source Files 28

8.2. C/C++ Project from Embedded 30

8.3. C/C++ Project from Visual Studio Project 31

8.4. C/C++ Project form Existing CodeScroll Project 33

8.5. Create a C/C++ Project with CPI File 34

8.6. Create a C/C++ Project with Build Information 35

9. Create a Test **37**

10. Test Editor **40**

10.1. Test Info 41

10.2. Test Case 43

10.3. Test Code 45

10.4. Configuration 46

10.5. Test Macro 47

11. Test Run **49**

12. Inspect Debug Information **51**

13. Generate Function Call Flow Contents **53**

14. Properties **56**

14.1. Project Properties 57

14.2. Module Properties 66

14.3. Source File Properties 68

15. Preferences 70
15.1. Analysis 71
15.2. Exclusion 77
15.3. Performance 78
15.4. Source File Types 79
15.5. Language 80
15.6. Test 82
15.7. Toochain 88
15.8. Editor 89

16. Test Perspective 90
16.1. Dashboard 92
16.2. Test Navigator View 95
16.2.1. Merge Project Coverages 98
16.3. Unit Test View 101
16.4. Integration Test View 117
16.5. Coverage View 124
16.6. MC/DC View 128
16.7. Stub View 130
16.8. Class Factory View 141
16.8.1. Using class codes 143
16.8.2. Using mock objects 148
16.9. Control Flow Graph View 150
16.10. Call Graph View 153
16.11. Error View 156
16.12. Debug Information View 158
16.13. Fault Injection View 162
16.14. Input/output Data Graph View 168
16.15. Console View 170
16.16. Requirement View 172

17. Analysis Perspective 178
17.1. Metrics View 179
17.2. Metrics Chart View 181
17.3. Metrics Top Chart View 183
17.4. Metrics Bar Chart View 184
17.5. Metrics Diagnosis Chart View 186
17.6. Unused Function View 188
17.7. Source-Header Relation View 189
17.8. Global Variable Relation View 190
17.9. Function Call Hierarchy View 192
17.10. Source Code Editor section 194

18. File Menu 195
18.1. Create a Module 196
18.2. Switch Workspace 197
18.3. Import 198

18.3.1. Requirement − Import from CSV file ... 200
18.3.2. Requirement − Import from Polarion ... 202
18.3.3. Requirement – Import from V-SPICE ... 205
18.3.4. General − File System ... 208
18.3.5. General − Import project ... 211
18.3.6. Coverage − Import Coverage ... 214
18.3.7. Test − Import test ... 216
18.3.8. Test − Import test using test code file ... 220
18.3.9. Preferences − Virtual Memory Address ... 222
18.3.10. Preferences − ToolChain ... 224
18.4. Export ... 226
18.4.1. Metrics − Metrics ... 228
18.4.2. Requirement − Export test results connected to requirements to Polarion ... 230
18.4.3. Requirement − Export test results to V-SPICE ... 232
18.4.4. Requirement − Requirement test coverage report ... 234
18.4.5. General − Export project ... 236
18.4.6. Coverage − Export Coverage ... 238
18.4.7. Test − Export test ... 240
18.4.8. Test − Generate test report ... 243
18.4.9. Preferences − Virtual Memory Address ... 246
18.4.10. Preferences − ToolChain ... 248

19. Edit Menu ... 249

20. Search Menu ... 250

21. Project Menu ... 252

22. Window Menu ... 254

23. Help Menu ... 255

24. Troubleshooting ... 260

25. CONTROLLER TESTER Target Plug-in ... 262
25.1. Target Environment Settings ... 263
25.2. C/C++ Target Test Project ... 268
25.3. Target Test Run Settings ... 273
25.4. Target Log Collector ... 274
25.5. Preferences ... 276

26. CONTROLLER TESTER RTV(Remote Target Verifier) ... 278
26.1. RTV Server Settings ... 279
26.2. RTV Toolchain Settings ... 283
26.3. Create a RTV Project ... 286
26.4. Refresh/Add RTV Source Files ... 291
26.5. RTV Project Information ... 295
26.6. Run RTV Test ... 297
26.7. RTV Test Results ... 299

27. CONTROLLER TESTER RTV(Remote Target Verifier) Target Plug-in 300
27.1. RTV Server Settings 301
27.2. RTV Toolchain Settings 302
27.3. RTV Target Environment Settings 303
27.4. Create a RTV Target Project 306
27.5. Refresh/Add RTV Target Source Files 312
27.6. RTV Target Project Information 313
27.7. Run RTV Target Test 315

28. DISCOVERY Plug-in 316
28.1. DISCOVERY Install 317
28.2. DISCOVERY Execution 319

29. CONTROLLER TESTER CLI(Command Line Interface) 321
29.1. CLI Install 322
29.2. Project Import/Export 324
29.3. CLI Settings 327
29.4. CLI Execution 332

30. EULA(End-User License Agreement) 334

1. Before Starting

Purpose of Document

This document provides the information on how to use the CodeScroll Controller Tester Product.

> Some images contained in this document may vary slightly depending on the version, but the image corresponding to the feature described is the same as the version specified in the documentation.

2. Overview

Introduction

Controller Tester is a test automation solution for unit and integration tests of software developed and executed in various environments.

Toolchain supported in Controller Tester

Controller Tester supports about 100 toolchain. For more information, please contact the Technical support contact at the bottom of this document Troubleshooting page.

Features

Controller Tester is a source code-based software test automation tool for unit and integration tests and the major features are as follows.

Test design

1. Automatically create tests and test data
2. Intuitive test design interface
3. Ability to import test data in various format
4. Ability to design of integration test for complex nested structures

Achieving coverage goals

1. Supports statement, branch, MC/DC and function call coverage
2. Provides guides to achieve the goal of MC/DC
3. Checks the coverage results by linking the control flow graph with the source code
4. The coverage goal can be achieved quickly by testing only uncovered section after measuring the system coverage (QualityScroll COVER product).

Others

1. Supports the simulation and the actual target environment tests
2. Links with DevOps(issue management + continuous integration + configuration management) tools
3. Shares project settings with CodeScroll product family: By carrying out the project settings only once, coding rule inspection + runtime error detection + unit/integration test can be possible.
4. Reports in various formats
5. Interaction with requirements management tools to support traceability of requirements to tests and results

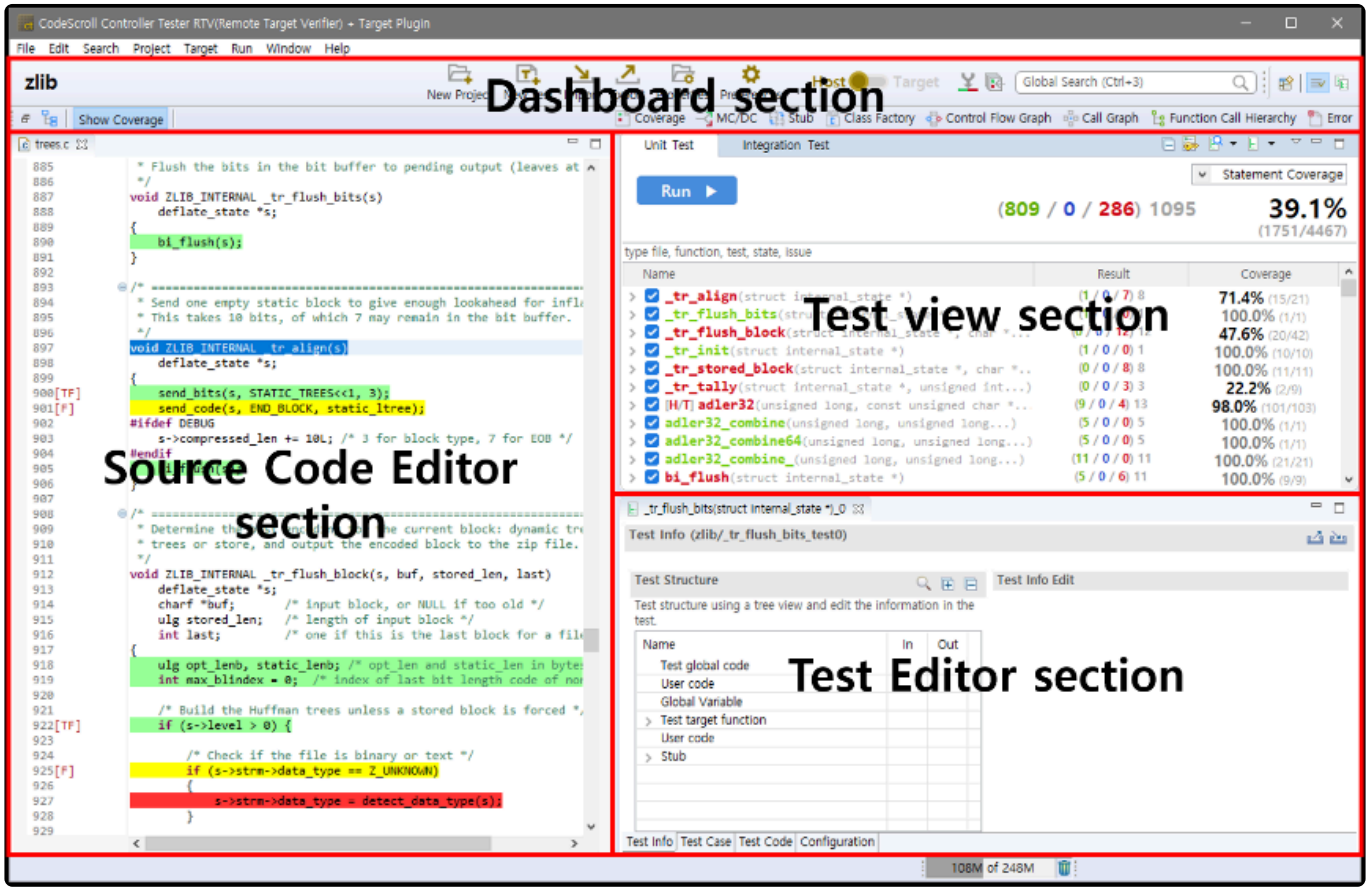
Standard certification

1. IEC 61508-3 (Electrical/Electronic)
2. ISO 26262-8 (Automotive)
3. IEC 60880 (Nuclear)

4. IEC 62279 / EN 50128 (Railway)
5. DO-178C / DO-330 (Aviation)
6. IEC 62304 (Medical device)

Overall screen of Controller Tester

It provides high DOF (degrees of freedom) windows made by Eclipse RCP. User can set them in [Window] menu and each view.



3. Install

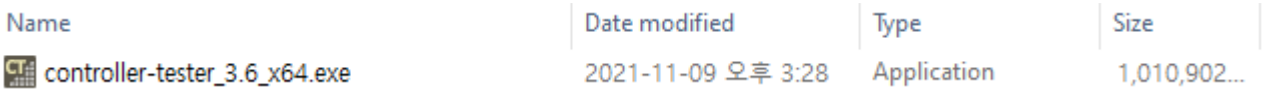
Installation requirements

OS	Microsoft Windows 7/10 (64bit)
RAM	512MB or more
HDD	Free space about 1GB (GCC Compiler installed) (When carrying out tests actually, HDD usage may be increased due to the test results.)

Install Controller Tester for Windows

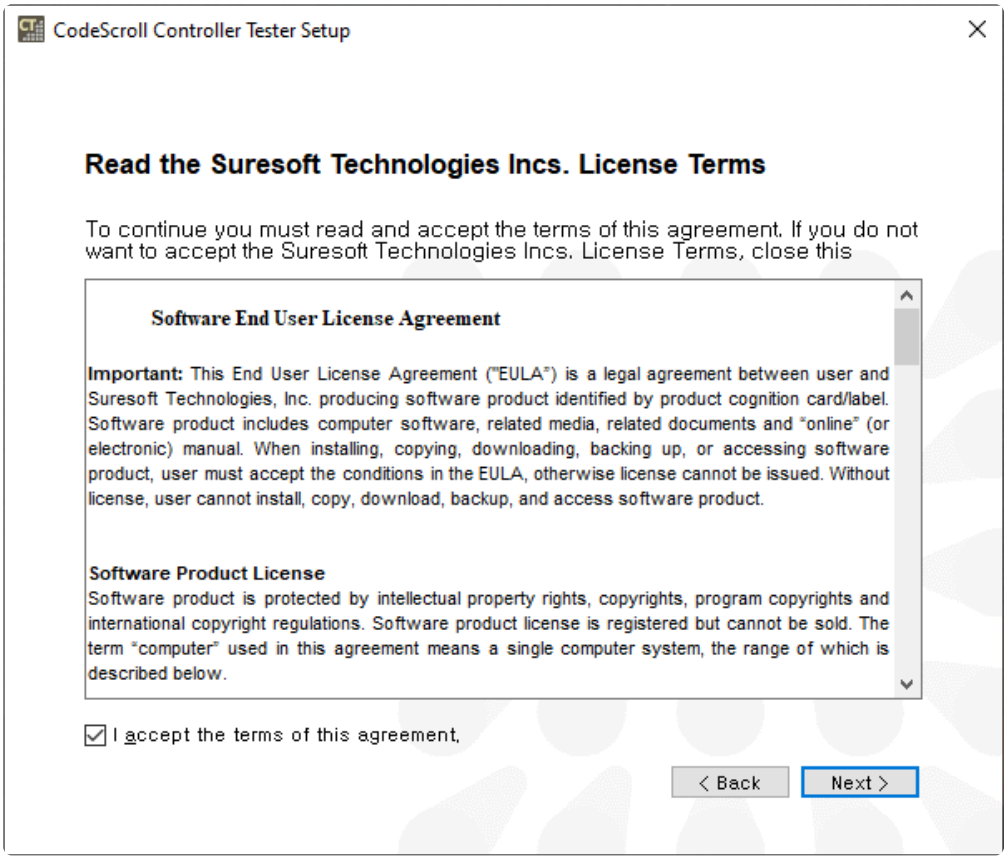
By using the install package, you can install Controller Tester as follows:

1. Execute controller-tester_3.x_xxx.exe file.

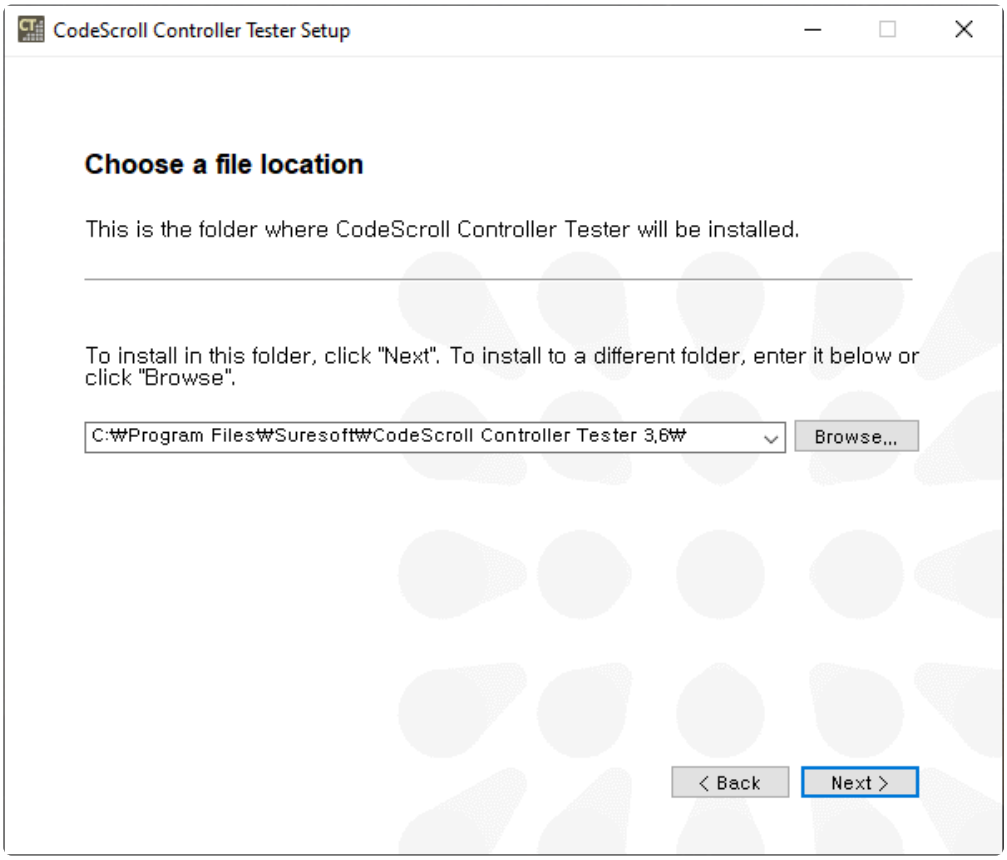


2. Click [Next] after the installation wizard runs and gathers installation information.

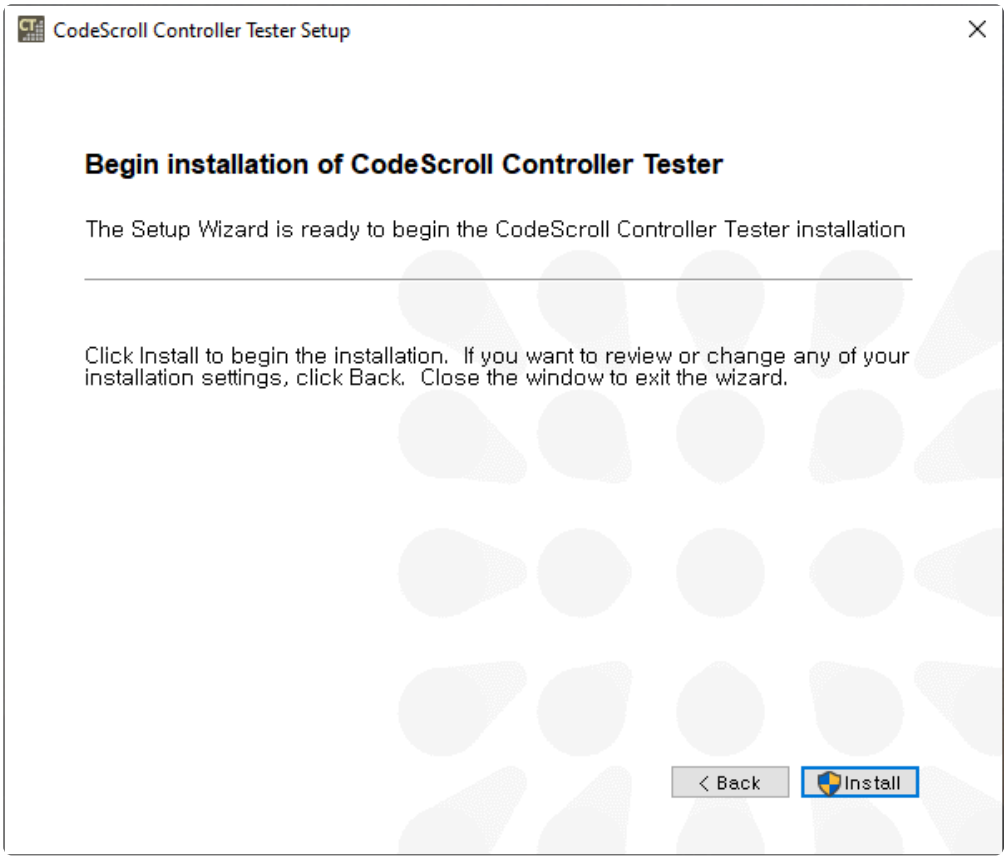
3. Accept the end-user license and click [Next].



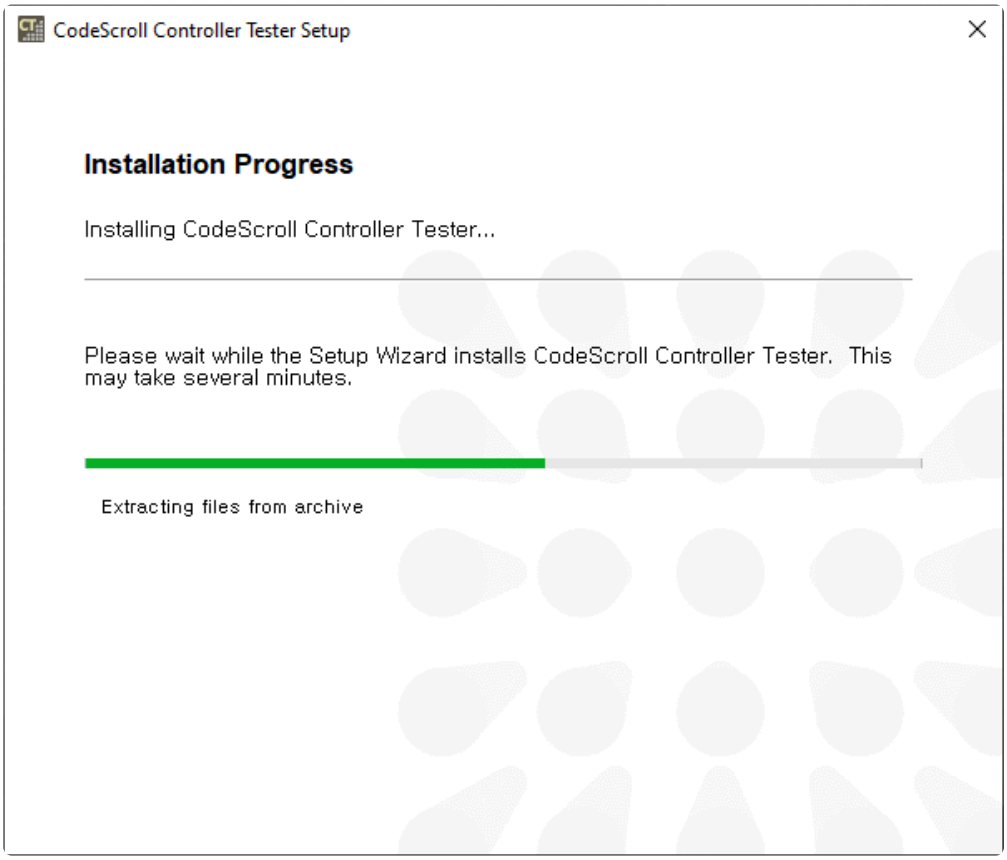
4. Set the path to install Controller Tester and click [Next].



5. All required information for the installation has been collected. Click [Install].



6. Install Controller Tester.



7. Click [Run CodeScroll Controller Tester] to run Controller Tester immediately after installation is complete, and click [Finish] to end the installation.

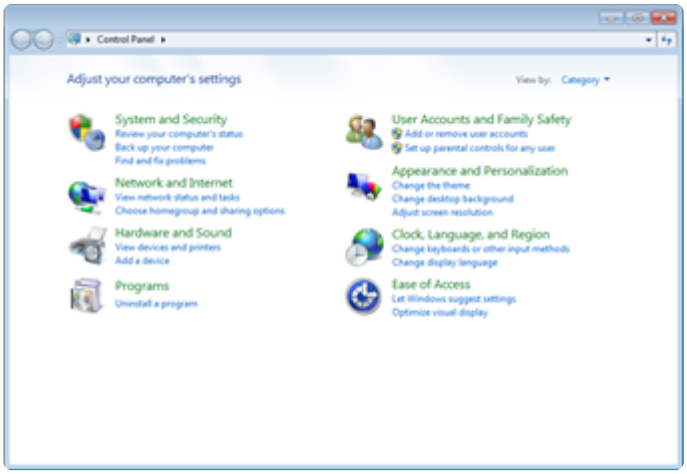
CodeScroll Controller Tester Setup
CODESCROLL
CONTROLLER TESTER
CodeScroll Controller Tester has been successfully installed.
Run CodeScroll Controller Tester
Close

4. Uninstall

Uninstall Controller Tester

How to uninstall Controller Tester is as follows.

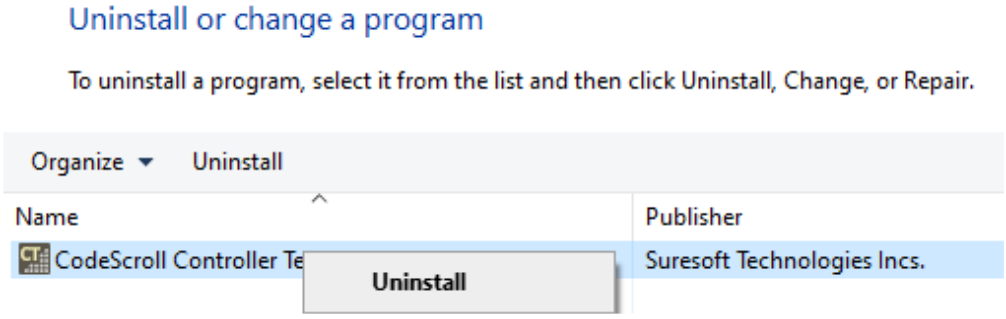
1. Uninstall using [Control Panel] > [Uninstall a program]



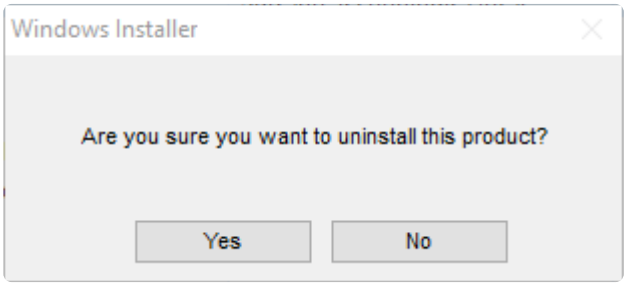
> “Uninstall Controller Tester” removes the files necessary to operate the Controller Tester. The workspaces set by users are not removed.

The procedures for [Uninstall a program]

1. Click [Control Panel] > [Uninstall a program].



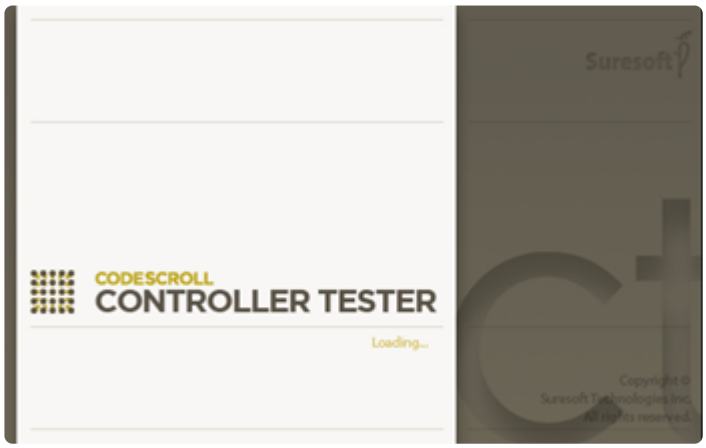
2. Click [Yes] to uninstall CodeScroll Controller Tester



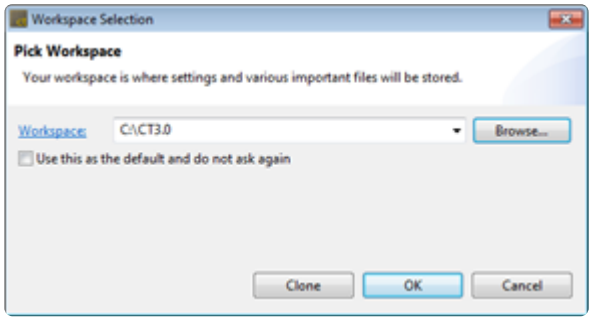
3. All the necessary information for uninstalling has been collected. When the Controller Tester is successfully uninstalled, the window closes.

5. Run

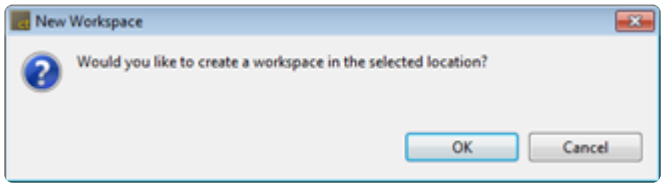
Select the shortcut icon or [Start] -> [All programs] -> [CodeScroll Controller Tester 3.x] -> [CodeScroll Controller Tester 3.x].



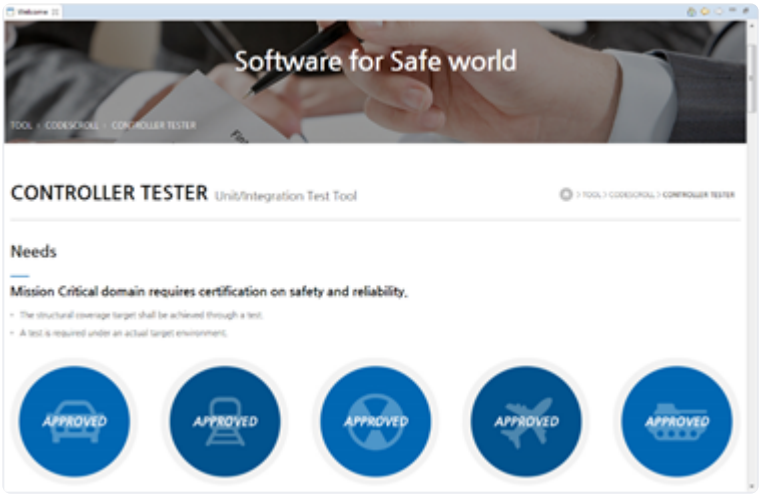
Select the workspace (working space) following the splash screen of the Controller Tester. You can select either the already existed directory or enter a new directory as the workspace path. You can also use the currently selected workspace as the default so that you do not need to reselect the workspace the next time you run again. When the path has been set, click the [OK] button.



If the selected directory does not exist, confirm the user whether to create the directory.



Create a directory in the workspace you have set and the welcome page of the tool and [Create a Project] wizard is displayed. The welcome page is displayed only when creating a workspace newly, and is not displayed when selecting the workspace created previously.

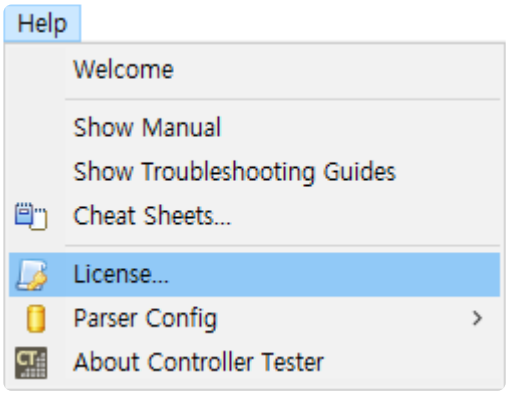


Close the welcome page and the product screen is displayed.

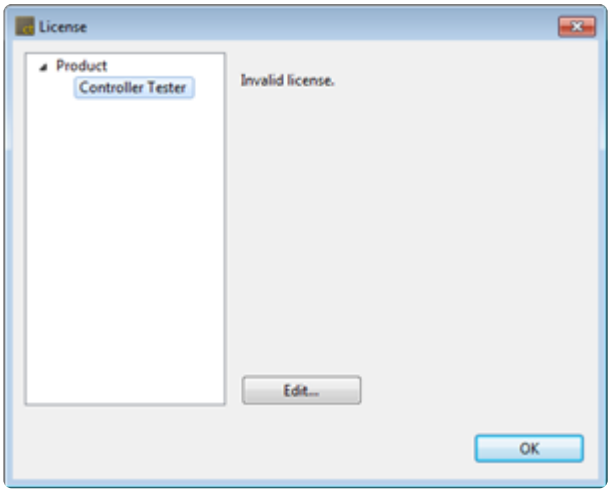
6. Set License

To use Controller Tester, you must register the license first. The procedures for registering the license are as follows.

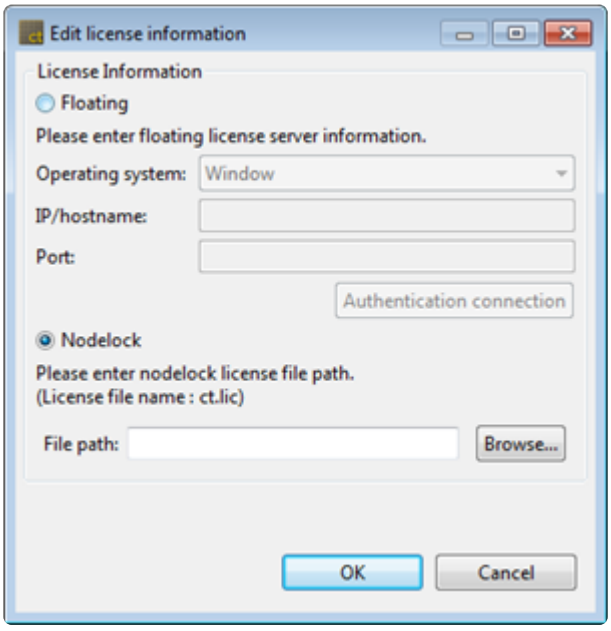
1. Select [Help] > [License] in Menu.



2. Select [Product] > [Controller Tester] and click [Edit].



3. Enter the provided license in [Editing License Information] and click [OK] button to register the license.

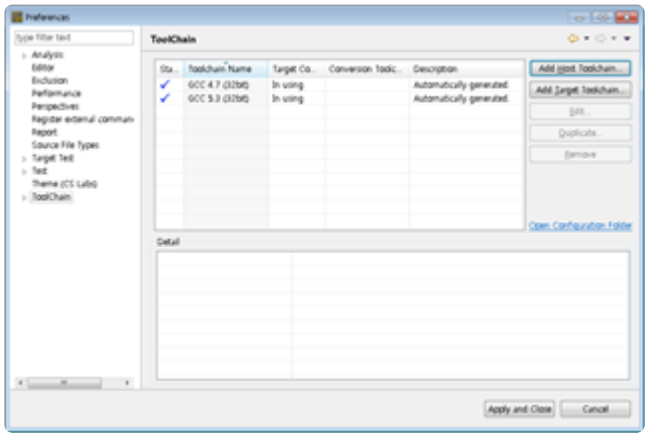


- Floating: Register it through the license server. Enter the server information (operating system, IP, port) and click [Authentication connection] button.
- Nodelock: Register the license file provided.

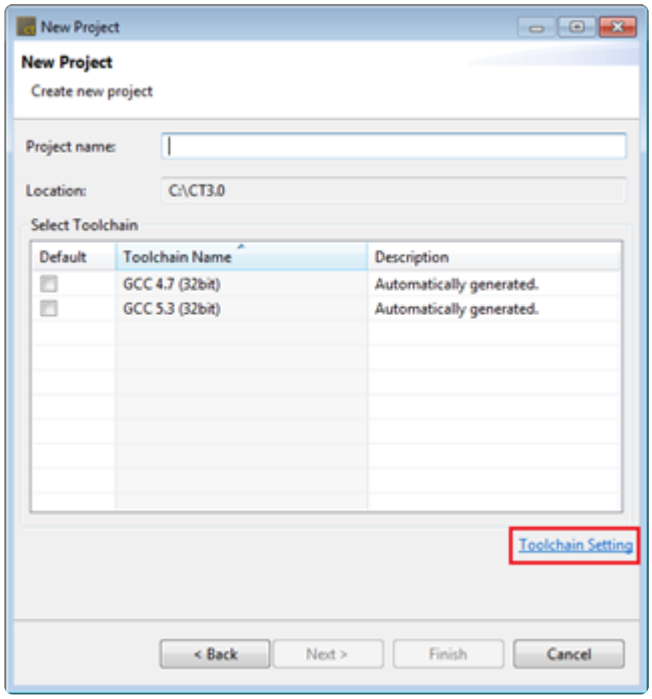
7. Set a Toolchain (Analyzer)

To perform the test using Controller Tester, the toolchain setting is required.
To create a project, you must have a toolchain(compiler information) of the sources to test.

The toolchain can be set in [Window] -> [Preferences] -> [Toolchain].



And it can also be set through [Toolchain setting] in Create a source file C/C++ project wizard or in Create a C/C++ Project from Embedded wizard.

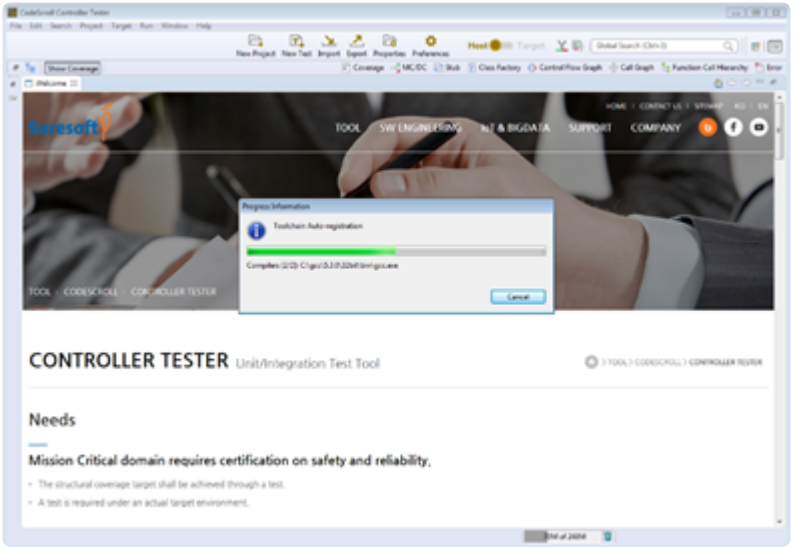


The functions related to the toolchain setting provided by Controller Tester are as follows.

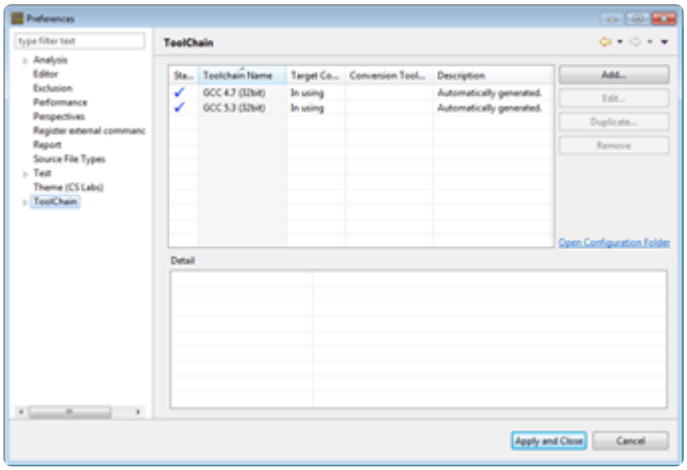
- Auto-registration of Toolchain
- Add a Toolchain
- Edit a Toolchain
- Duplicate a Toolchain
- Remove a Toolchain
- Export a Toolchain
- Import a Toolchain

7.1. Auto-registration of Toolchain

When executing the Controller Tester, it extracts the information of Visual Studio compiler installed in user PC and the information GCC compiler installed along with the Controller Tester and registers the toolchain automatically (the registered toolchain is not registered again.).

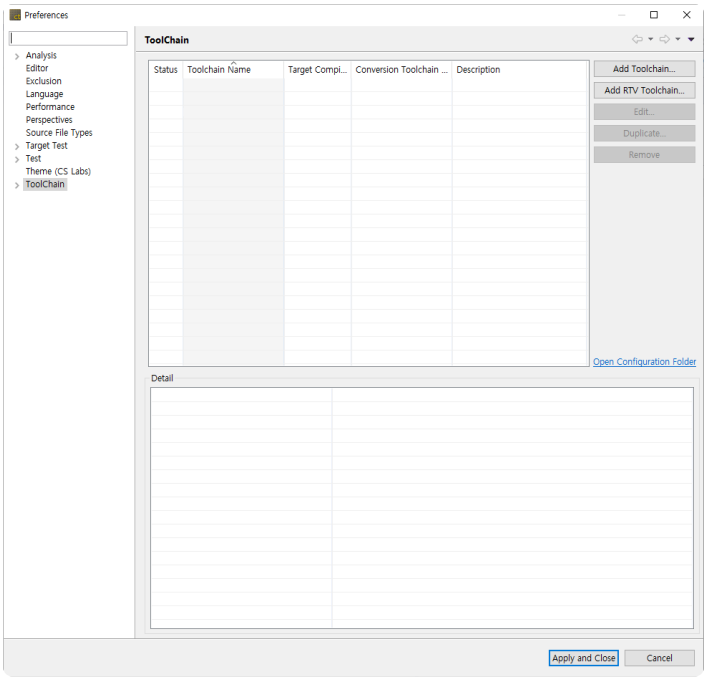


The auto-registered toolchain can be checked in [Window] -> [Preferences] -> [Toolchain] window, and it displays the message “Automatically generated.” in the description.

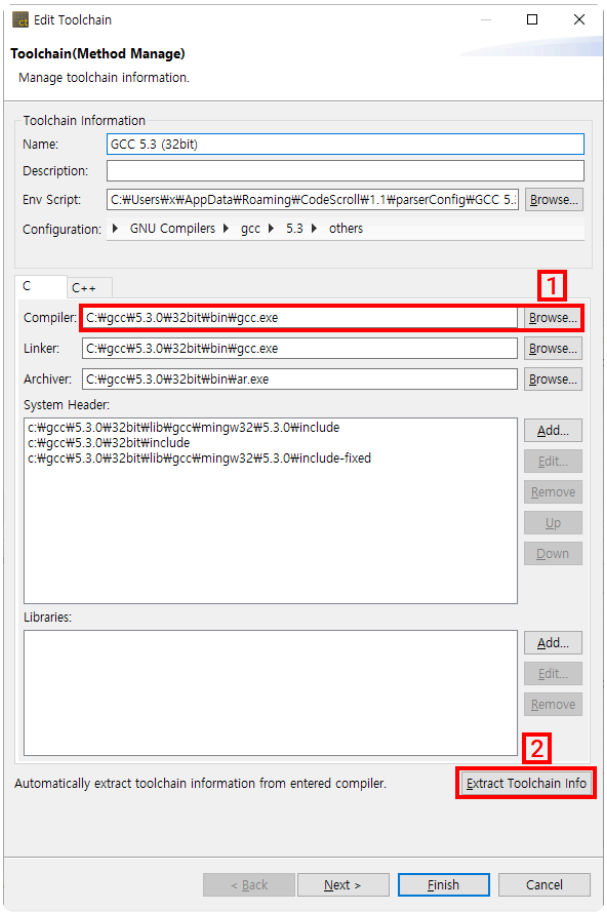


7.2. Add a Toolchain

1. In [Window] -> [Preferences] -> [Toolchain] window, click the [Add Toolchain] button.



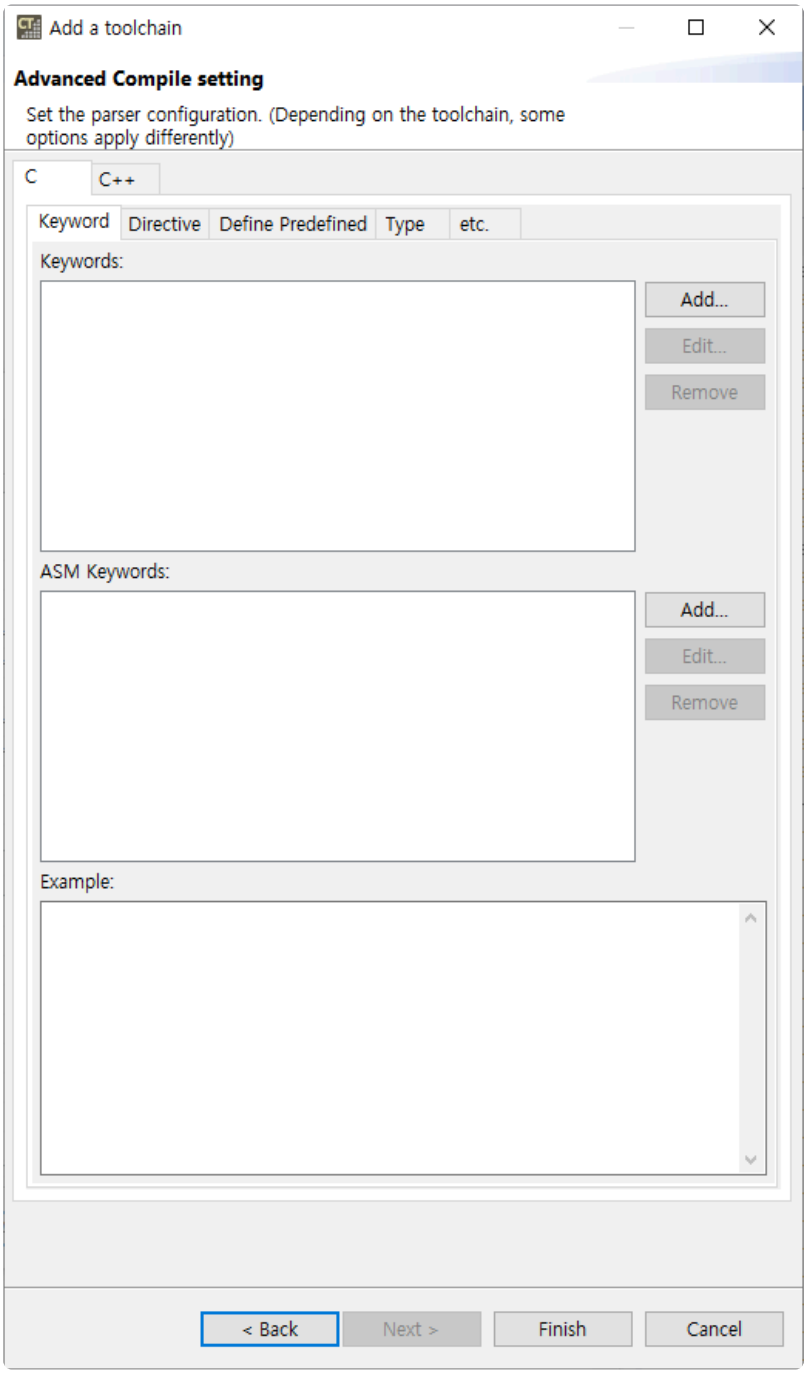
2. In [Add a toolchain] window, enter the toolchain information.



   - When you click [Extract toolchain Info]2 button after [Compiler]1, the toolchain information is extracted automatically from the compiler entered.
   - In [Compiler]1, enter the compiler path.
     - Ex:

- If it is based on Visual Studio:
  C:\program Files (x86)\Microsoft Visual Studio 10.0\VS\bin\cl.exe
- If it is based on GCC:
  C:\MinGW\bin\gcc.exe
- CodeWarrior 5.x or higher for Freeescale HC (s) 12:
  C:\Program Files (x86)\Freescale\CWS12v5.1\Prog\chc12.exe

3. In [Name], enter the name of toolchain to be created.
4. In [Description], enter the description of toolchain to be created.
5. The entry information of [Env Script], [Configuration] and [C/C++] is set automatically by extracting the information from the compiler or can be entered directly by user.
6. After entering the toolchain information, click the [Next] or [Finish] button.

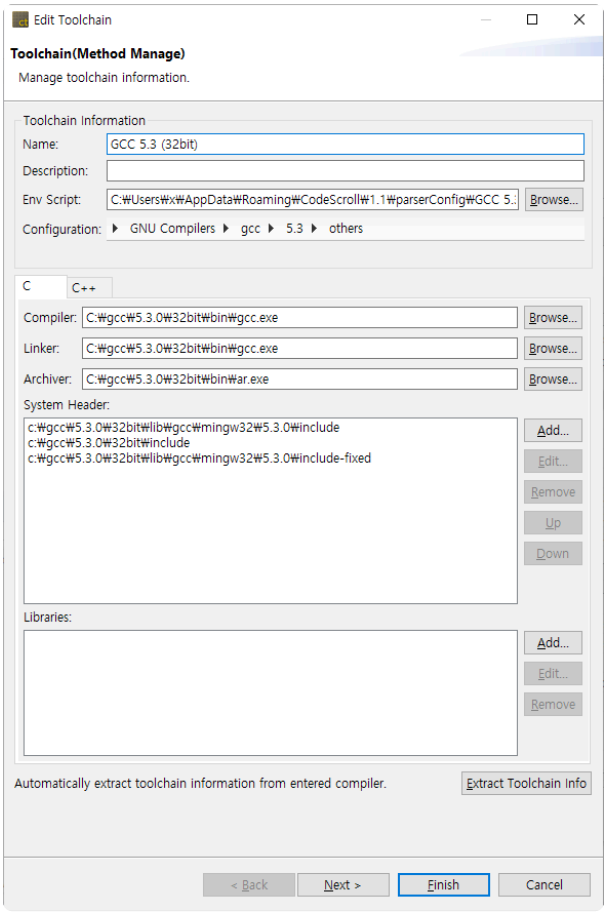


7. In [Advanced Compile setting] window displayed when clicking the [Next] button, the detailed items related to the configurations selected in the [Configuration] are shown. Each item can be changed by the user. The application of the changed items can be checked in [Example].

For the detailed setting procedure for the other toolchain detail, please contact the Technical support contact at the bottom of this document Troubleshooting page.

7.3. Edit a Toolchain

1. Select the toolchain to be edited and click the [Edit] button.



2. Edit the information of toolchain.
3. Click the [Finish] button.

7.4. Duplicate a Toolchain

1. Select the toolchain to be copied and click the [Duplicate] button.
   (It can copy only a toolchain which a compiler is not entered.)
2. Edit the toolchain information that you want to change.
3. Click the [Finish] button.

7.5. Remove a Toolchain

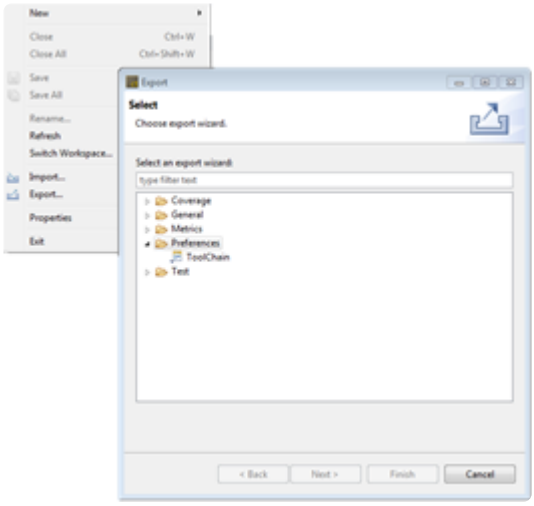
Select the toolchain to remove and click the [Remove] button.

> ! If there is a project using the toolchain you want to remove, the project will not work properly.

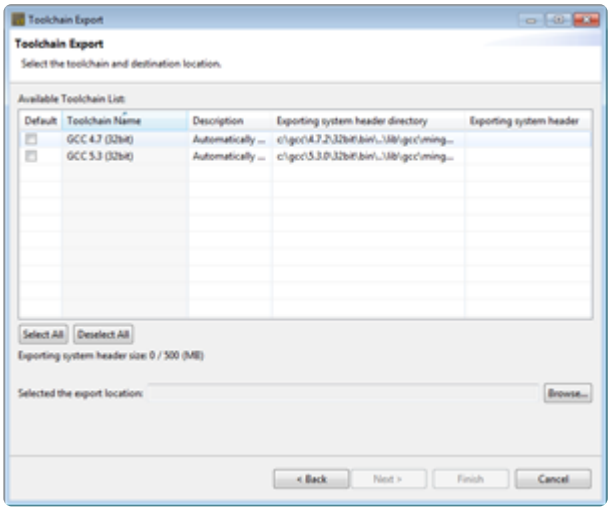
7.6. Export a Toolchain

The toolchain information can be exported via ‘Export’ function.

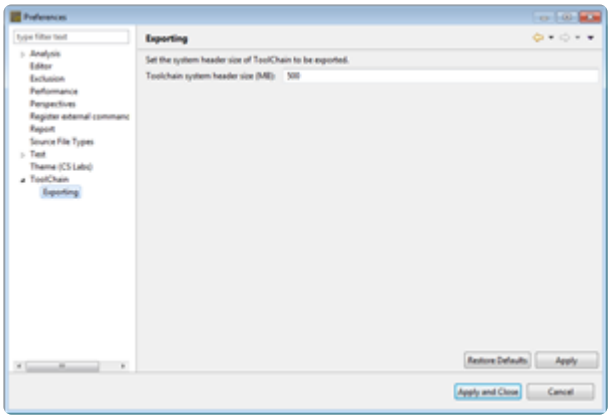
1. Click [File] > [Export] in menu and select [Toolchain] in [Preferences].



2. Select the toolchain to be exported and click [Browse] button to specify the path to be exported. If you set the system header file in the toolchain settings, a checkbox will be displayed in the [Exporting system header] column, where you can choose whether to export or not.



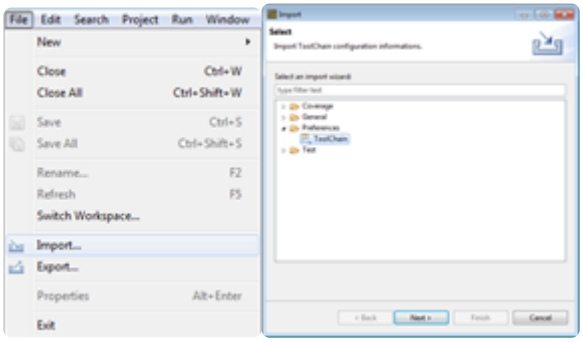
3. Click [Window] > [Preferences] > [Toolchain] > [Exporting] in menu to set the system header size to be exported. If the size of the system header is larger than the set value, the system header cannot be exported.


Preferences
type filter text
Analysis
Editor
Exclusion
Performance
Perspectives
Register external commanc
Report
Source File Types
Test
Theme (CS Labs)
ToolChain
Exporting
Exporting
Set the system header size of ToolChain to be exported.
Toolchain system header size (MB):
500
Restore Defaults
Apply
Apply and Close
Cancel

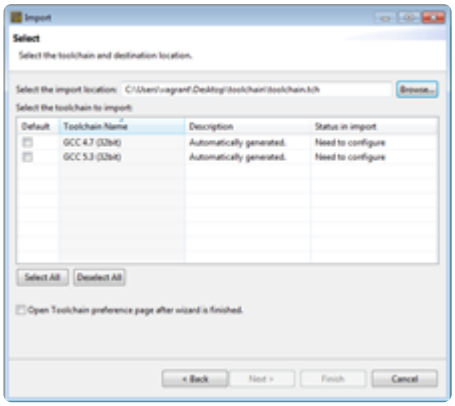
7.7. Import a Toolchain

The exported toolchain can be imported via the [Import] function.

1. Select [File] > [Import] in menu and click [Toolchain] in [Preferences].



2. Select the file to import and check the toolchain to import when the toolchain of the file is displayed.



The name of the toolchain to import is duplicated with the existing toolchain name, the status of ‘import’ becomes “Need to configure”. The name of the toolchain to import can be modified in [Select how to resolve] dialog shown when clicking the [Next] button.

8. Create a Project

Checklist before creating a project

Since Controller Tester is a tool that detects errors by actually executing source codes, the source code under test must be executable. In other words, the source code under test must be buildable for normal testing.

Before creating and testing a project, you need to set up a 'toolchain' that takes into account the development environment of the target software. Please refer to Set a toolchain for details.

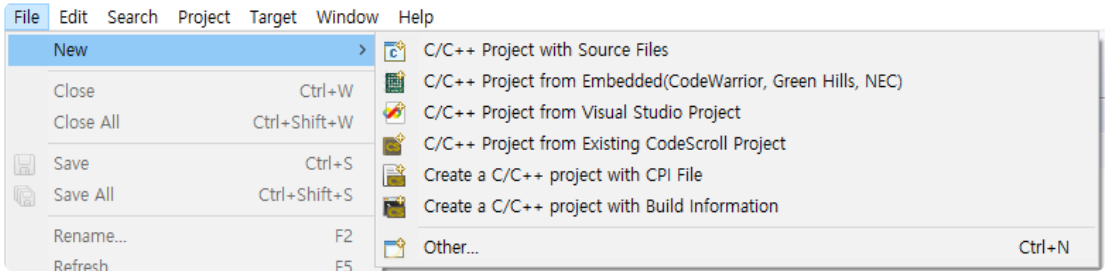
Controller Tester automatically generates a stub if the target object to link does not exist during the build process. However, the automatic test stub generation feature may have limitations in performing tests, so it is recommended to prepare an appropriate test strategy in advance.

Also, the name of the project cannot be more than 100 characters.

Create a project

To create a Controller Tester project, execute [Create a Project wizard].

Select [File] -> [New] or click the shortcut icon.



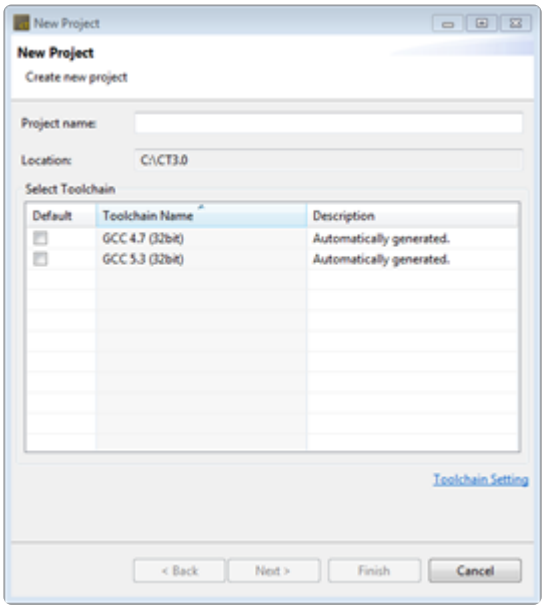
The following types of project creation wizards are provided.

- C/C++ Project with Source Files
  Creates a project by selecting the source file directly.
- C/C++ Project from Embedded
  Creates a project by using an embedded project file.
- C/C++ Project from Visual Studio Project
  Creates a project by using Visual Studio dsw, sln, vcxproj and vcproj files.
- C/C++ Project form Existing CodeScroll Project
  Creates a project by using the CodeScroll project created previously.
- Creates a C/C++ Project with CPI File
  Creates a project by using project creation information located in a CPI file.
- Creates a C/C++ Project with Build Information
  Creates a project by using a makefile and build script.

8.1. C/C++ Project with Source Files

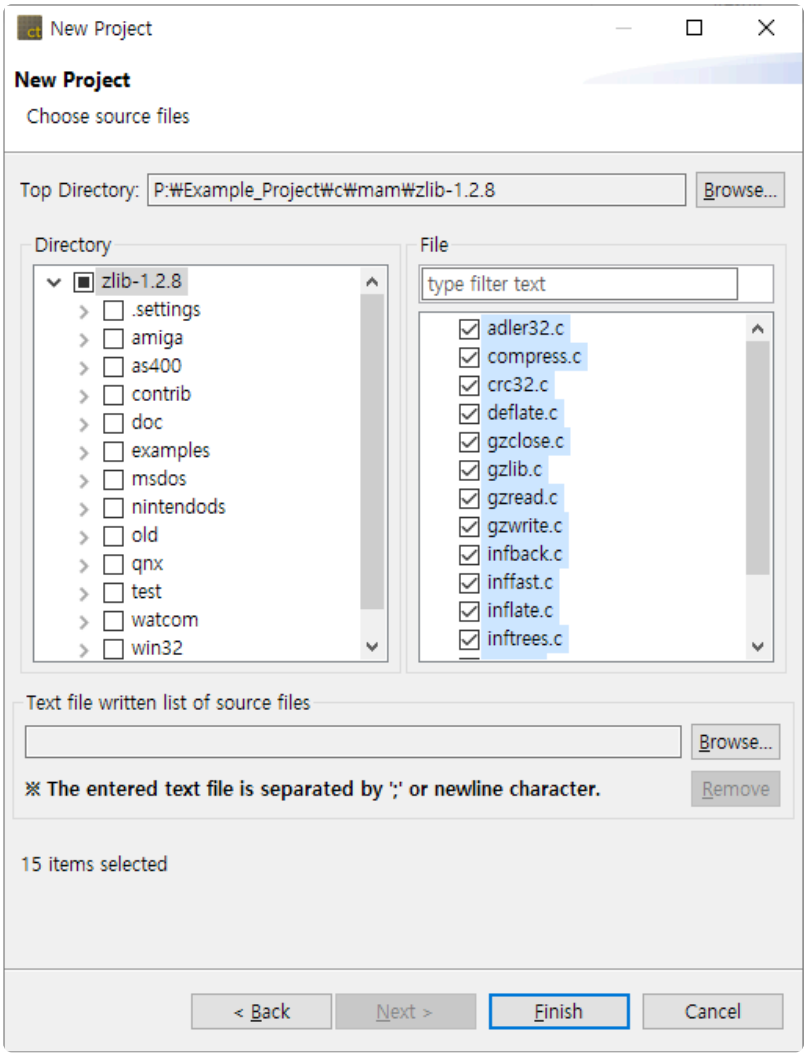
Create a project with C/C ++ source code files. The user needs to enter the necessary information to build the source code.

1. Enter the project name in [Project name].



2. In [Select Toolchain], select the toolchain to use in the project. If there is no created toolchain, create a toolchain through [Toolchain Setting]. Please refer to Set a Toolchain for details.

3. Click the [Next] button to move to the next screen. If you click the [Finish] button, an empty project containing no source file is created.



- You can select the source file directly, or through a text file with a file path.

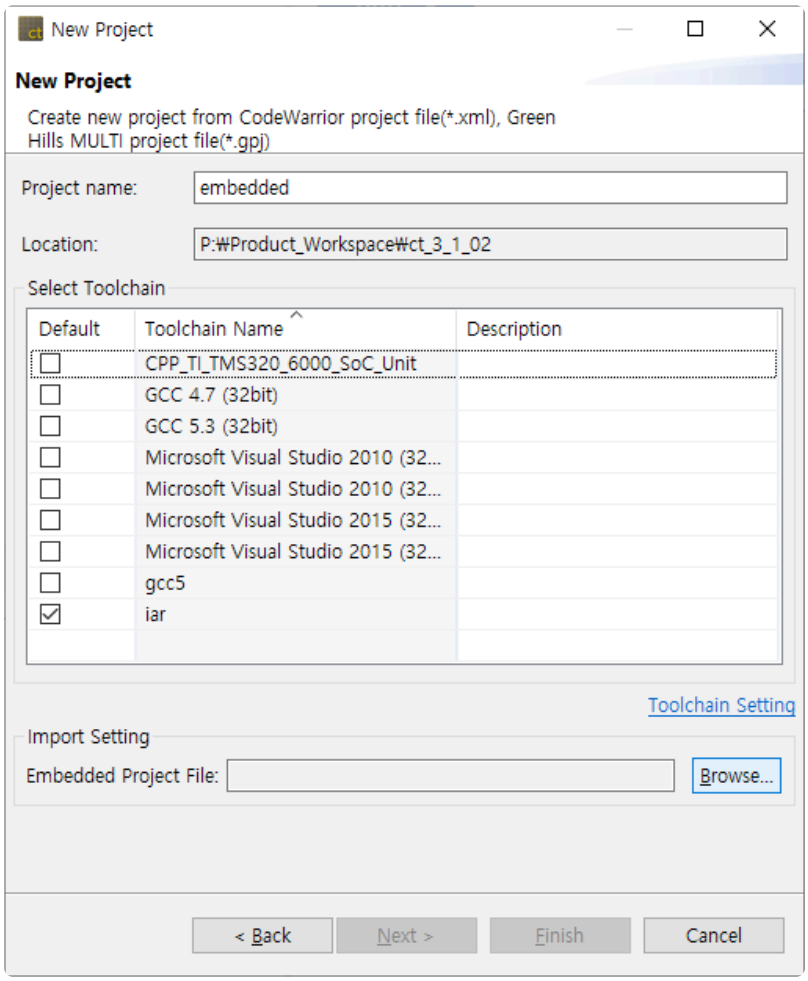
4. Select source files directly.
   a. Click [Browse …] to specify the [Top Directory] to be displayed in the directory screen below.
      - Except in special cases, make the directory one level higher than the directory containing the source files you want to select as the top-level directory.
   b. On the [Directory] screen shown on the left, select the directory containing the source files to be used for project creation.
   c. The files in the selected directory appear on the [File] screen shown on the right. Check the files to add.

5. Select by using a text file with the file paths.
   a. On the [Text file written list of source files] screen shown below, click [Browse …] to select a text file with a list of source files(absolute file path).
   b. Clicking the [Remove] button deselects the selected text file and the source file selected through the file.

6. After all settings are completed, click the [Finish] button to create the project.

8.2. C/C++ Project from Embedded

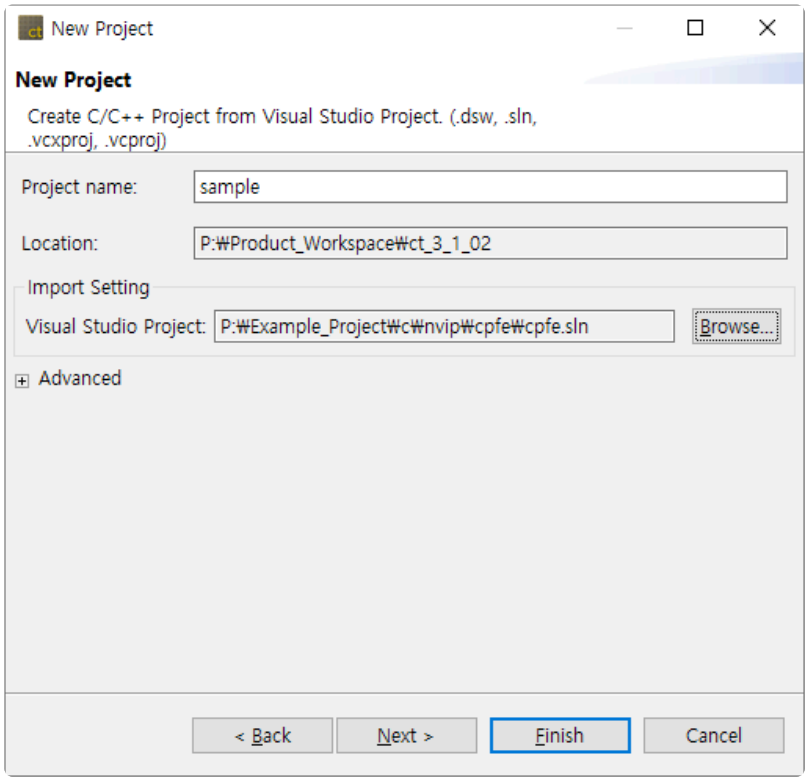
Create a project with an embedded project file(.xml, .gpj, .prw, .prj). Except in special cases, the information required for the build is automatically entered, so there is nothing to specify by the user.



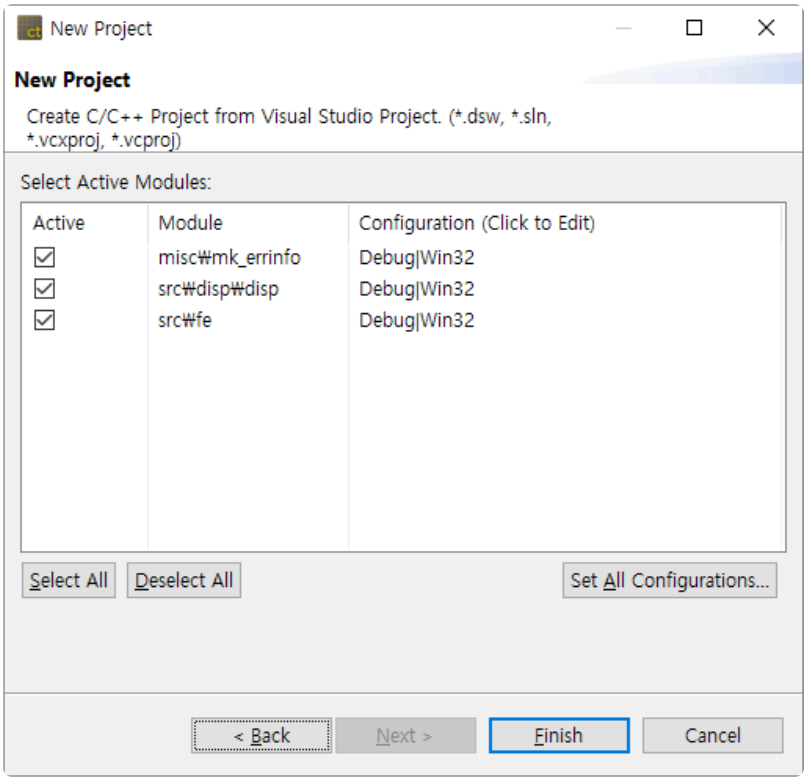
1. Enter a project name in [Project name].
2. Choose a toolchain that fits the embedded project you select.
3. In [Import Setting] -> [Embedded Project File], specify the embedded project file of the software under test.
4. After completing all settings, click [Finish] to create the project.

8.3. C/C++ Project from Visual Studio Project

Create a project with a Microsoft Visual Studio project file(.dsw, .sln, .vcxproj, .vcproj). Except in special cases, the information required for the build is automatically entered, so there is nothing to specify by the user.



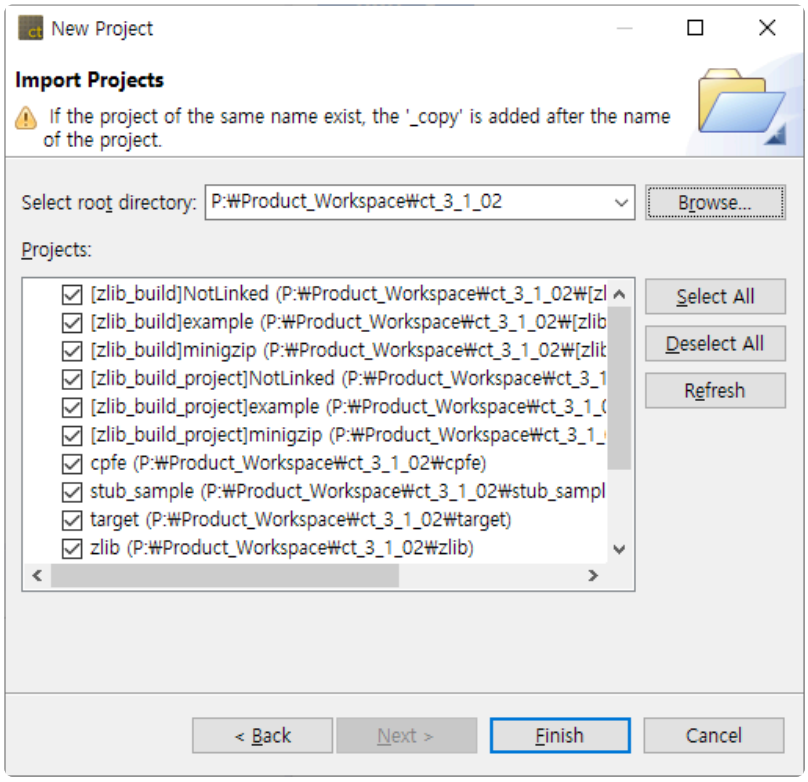
1. Enter the project name in [Project name].
2. Specify the Visual Studio project file to import in [Visual Studio Project].
3. If there are values that must be set before calling the compiler of the toolchain to be registered, add a script file to set the values to [Advanced]-> [Environment Script File].
4. Click [Next] to move to the next screen. When you click Done, module settings are randomly selected for the imported Visual Studio project.



5. Select the module to be activated from the modules included in the Visual Studio project.
6. Select the configuration of each module.
    - [Set All Configurations …] allows you to change the configuration for all modules in a batch.
7. After all settings are completed, click the [Finish] button to create the project.

8.4. C/C++ Project form Existing CodeScroll Project

Import an existing CodeScroll project and create a new project with that information. You can easily create a new project with the same configuration(source file, exclude/include analysis, compile flags) as the existing project.

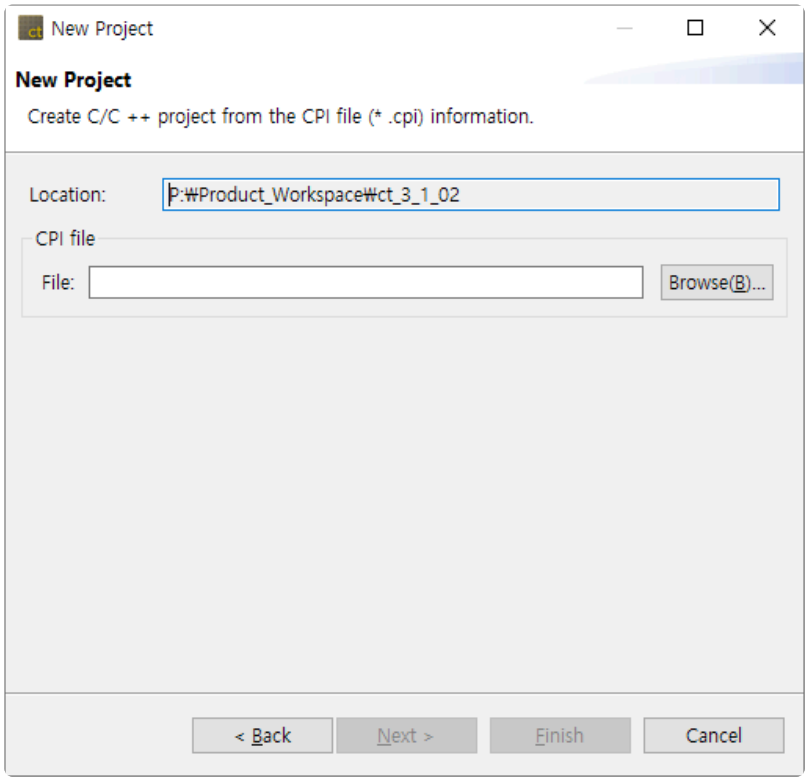


1. Use the [Browse …] button to select the CodeScroll workspace or CodeScroll project path.
2. Projects existing in the selected path are displayed in the [Projects] list.
    - When the CodeScroll workspace is selected, all the projects under it are displayed.
3. Click [Finish] to create the project.

8.5. Create a C/C++ Project with CPI File

Create a project from a CPI file. The CPI file is an information file for project creation in the command line interface.

The CPI file must be created by the user using the template file in the {installation path}\plugins\com.codescroll.gp.cli_x.x.x.x\cpi folder.



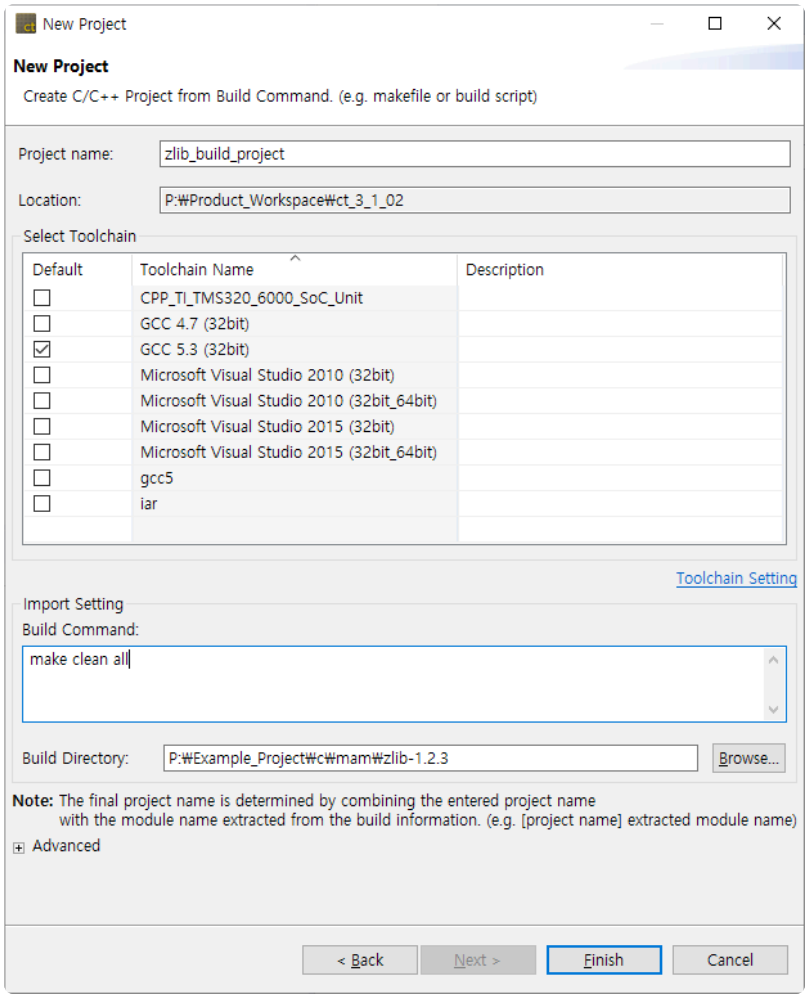
1. Select a CPI file with the [Browse…] button.
2. Click the [Finish] button to create the project.

You can also create a project by dragging and dropping a CPI file into the test navigator.

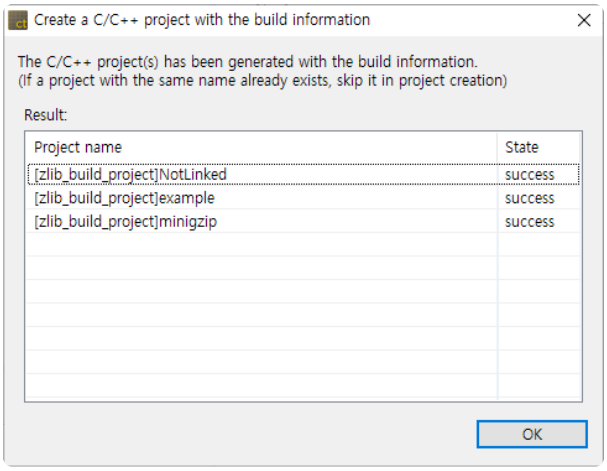
8.6. Create a C/C++ Project with Build Information

Create a C / C ++ project from build information. The project is created as many as the number of modules extracted from the build information.

1. Enter a project name in [Project name]. The final project name is determined by combining the entered project name with the module name extracted from the build information.
    - Ex) [Project name]Extracted module name



2. Choose the same toolchain as the compiler you use for the build.
3. In [Import Setting]-> [Build Command], enter the command required for the build.
    - Ex) Make all / make clean
4. Specify the path where the makefile is located in [Import Setting]-> [Build Directory].
5. If there are values that must be set before calling the compiler of the toolchain to be registered, add a script file to set the values to [Advanced]-> [Environment Script File].
6. After completing all settings, click [Finish] to create the project.
7. When the project creation is completed after building, the result screen appears.



- Success: Project creation complete.
- Error: If the creation failed due to an error during project creation.
- Exclude: If the same project name exists in the workspace.

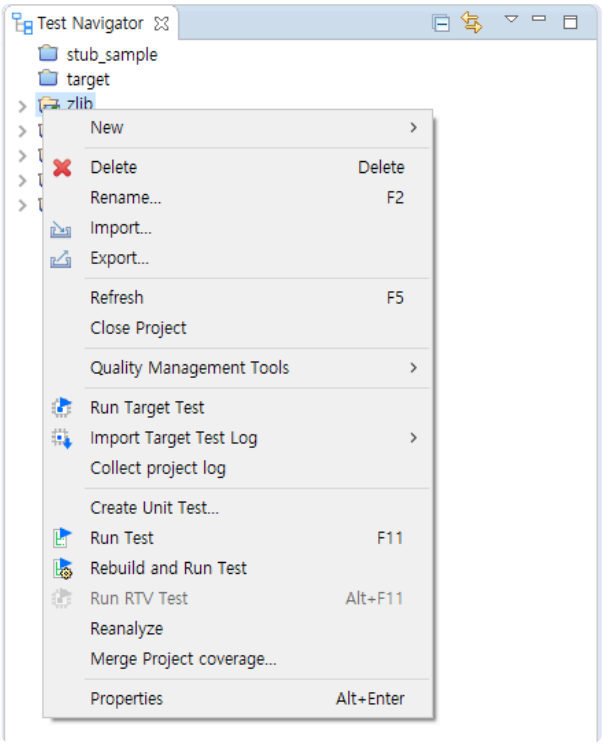
9. Create a Test

Generate test data and test code to test the functions of the source code. You can create individual tests for each function.

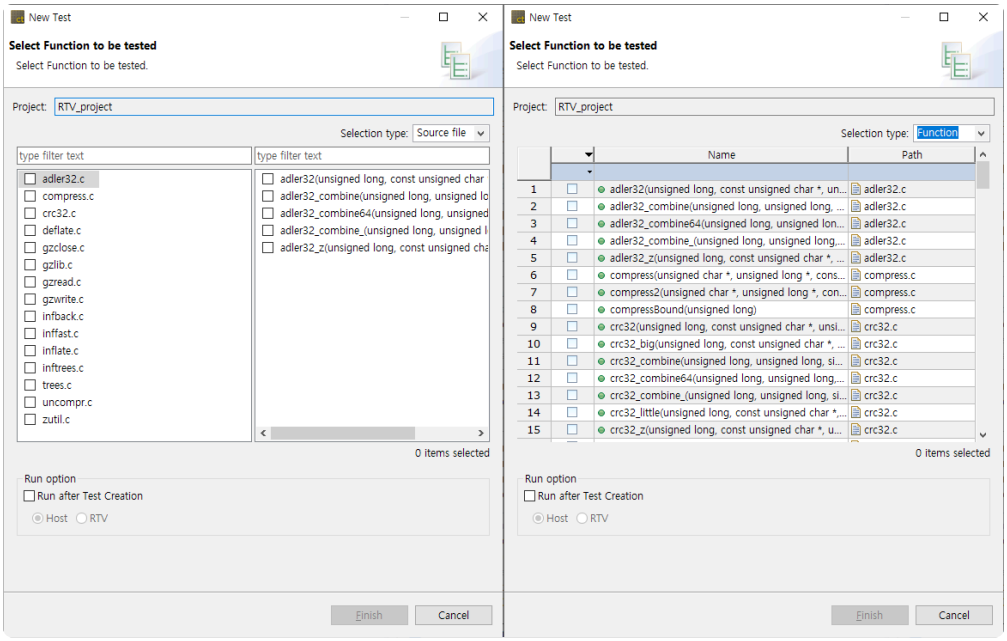
Create unit tests

Generate test data and test code for each function under test.

1. Select the project or module to create unit tests, right-click and click the [Create Unit Test] menu.



2. After the analysis is finished, in [Select function to be tested], select the function to perform the unit test.



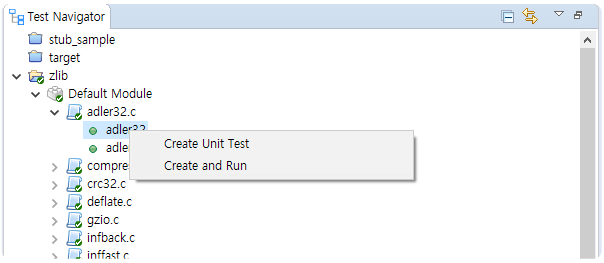
a. You can filter the function under test using [type filter text].
b. You can use the [Selection type] combo box to select the function to be tested based on the source file or function.
c. You can easily select and deselect all functions under test with the right-click menu.
d. When [Run after Test Creation] is selected, the test is executed after the test data is generated.

3. Click [Finish] to create the unit test.

Create and run tests

The [Create and Run] function allows you to create and run unit tests at once without using the [Create Unit Test] function.

1. Select the function to be tested in the test navigator and right-click.

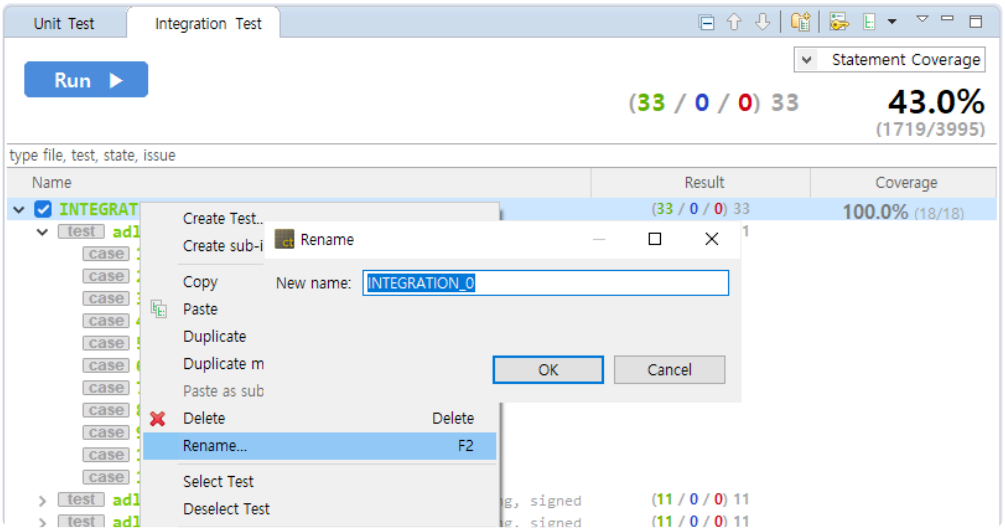


2. If you select the [Create and Run] menu, it creates a test for the target function and then runs the test.

Create an integration test

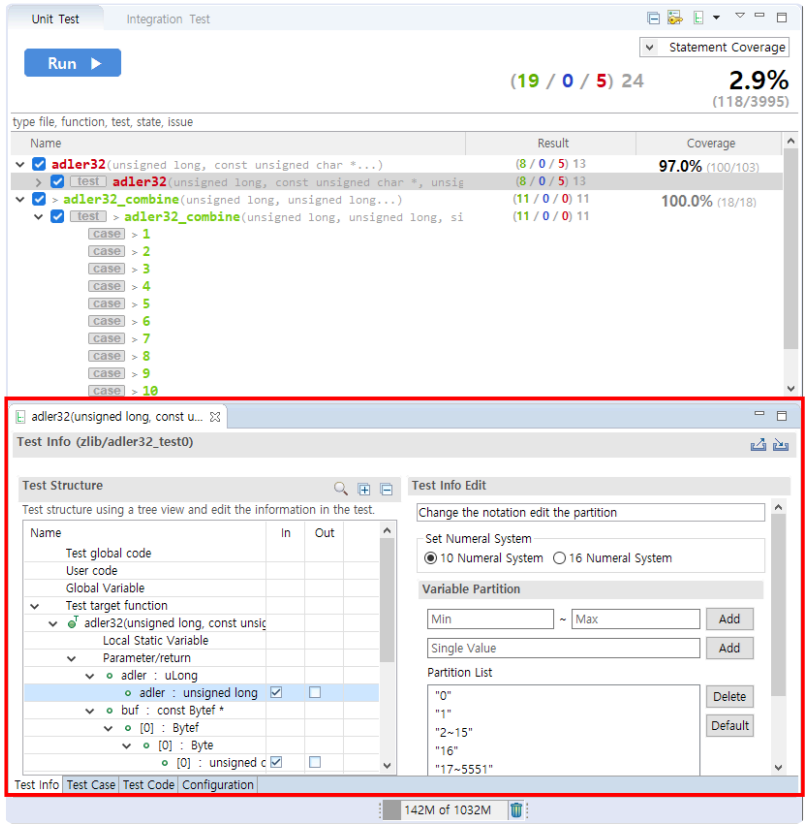
You can use the toolbar menu to create an integrated test.

The integration test name is automatically assigned and can be renamed using the [Rename] context menu.


Unit Test
Integration Test
Run
Statement Coverage
(33 / 0 / 0) 33
43.0%
(1719/3995)
type file, test, state, issue
Name
Result
Coverage
INTEGRAT
(33 / 0 / 0) 33
100.0% (18/18)
Create Test..
Create sub-i
Copy
Paste
Duplicate
Duplicate m
Paste as sub
Delete
Delete
Rename...
F2
Select Test
Deselect Test
Rename
New name:
INTEGRATION_0
OK
Cancel
(11 / 0 / 0) 11
(11 / 0 / 0) 11

10. Test Editor

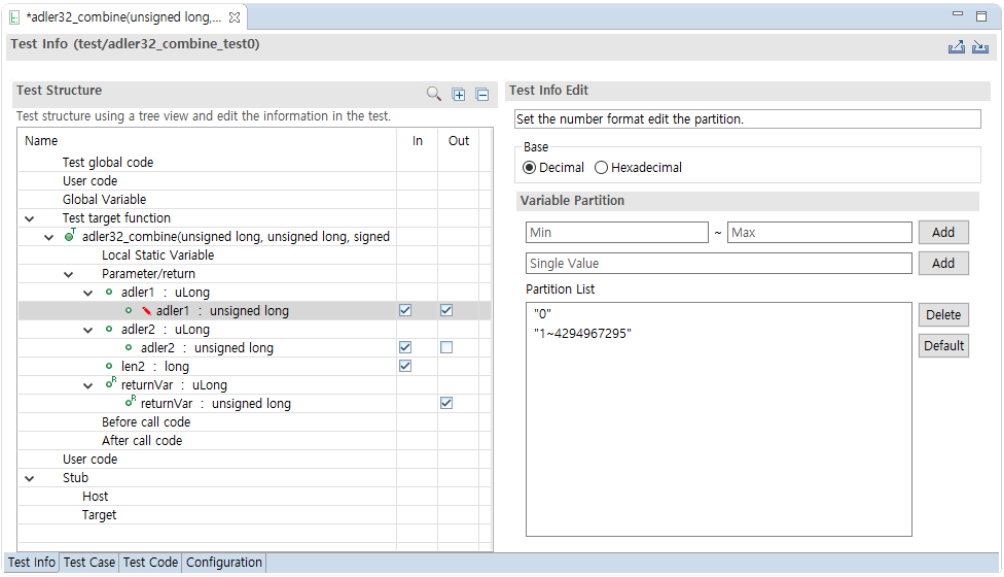
The test editor can be opened by double-clicking the test or test case in each test view, located at the bottom of the unit/integrated test view.



You can move the editor and open it in a separate window.

10.1. Test Info

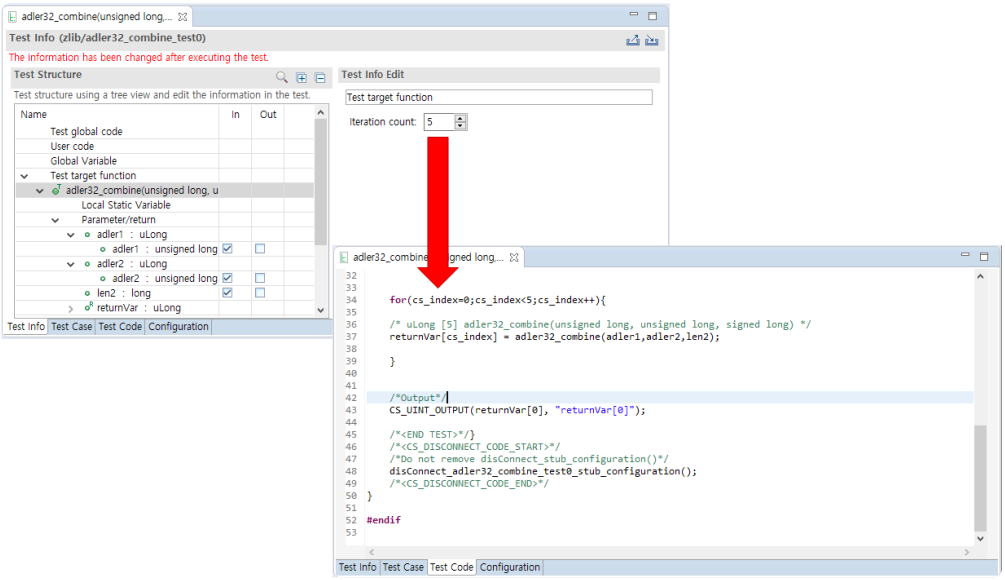
The Test Information tab is a feature that presents the test code as a tree of test structures, making it easy for users to modify the test code.



- User code can be inserted into the test code.
- You can control global variables related to the function under test.
- You can control the input and output of the function parameters and return values.
- You can enter the code to be executed just before calling the function under test.
- markers appear at modified items. When you save modifications, the markers disappear.

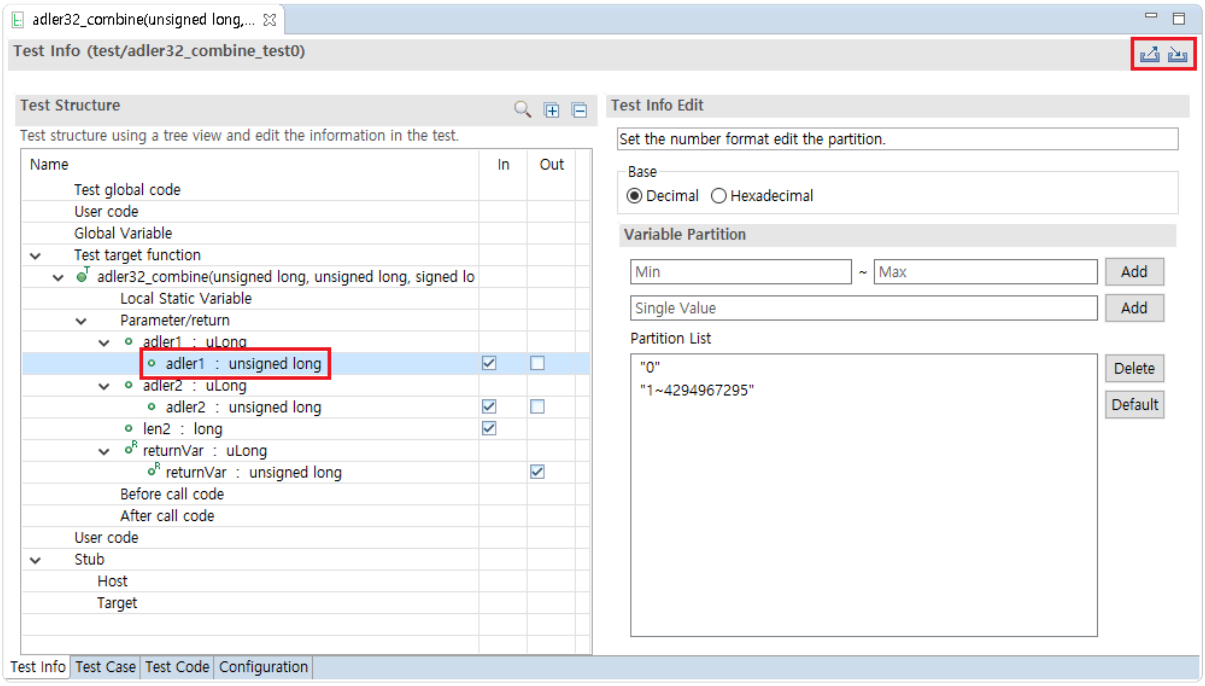
Repeat function calls

You can have the function under test be called multiple times.



Export/Import variable partition

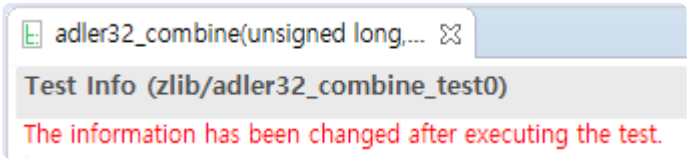
You can export/import variable partition information by selecting a variable on the Test Information tab.



Menu	Description
Export Variable Partition	Export the partition information of the selected variable.
Import Variable Partition	Import the partition information of the selected variable.

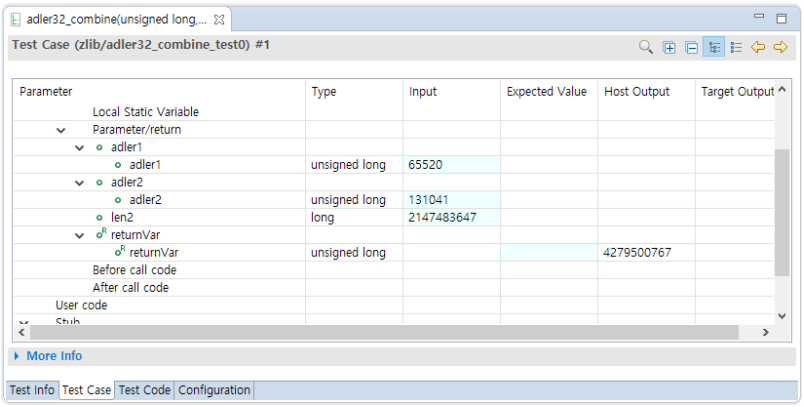
Change detection

If you edit and save the information of a test that has already been executed, “The information has been changed after executing the test.” is displayed at the top of the test editor.



10.2. Test Case

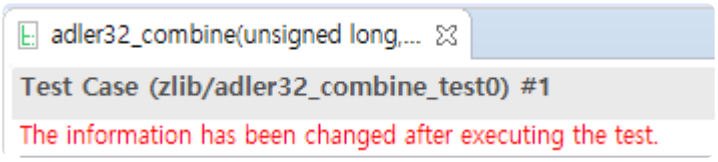
Test Case Tab provide the ability to edit inputs for a function under test and the expected value for the execution result.



- In the Test Structure tree, you can enter input values for the variables specified as input.
  - **If there is no input value, the value of the variable is 'undefined value', and the tool initializes the variable with no value entered as 0 (Blank if a string) for the convenience of the user.**
- You can switch the representation of the Test Structure as a table or tree with the [Show as Tree], [Show as Table] toolbar menu.
- With the [Apply the test case in a lump] context menu, you can apply the input and expected values of the test case to other test cases in the same test at once.
- If you select the 'Out' of a variable in the Test Structure tree, You can check the value of the variable after the test is performed.
- Editable cells are displayed in a pale sky color.
- If the expected value is different from the host/target output, it is displayed in blue.
- If the host and target output values are different, it is displayed in yellow.
- You can use ~ (range) and logical operators! (Not), & (and), and | (or) in expected values.
- The following shortcuts are available.
  - Undo : `Ctrl` + `Z`
  - Move row: arrow keys (↑, ↓)
  - Editable cell focus: `Enter`
  - Editable cell focus out: `Esc`
  - Move column (editable cell): `Tab`(right), `Shift` + `Tab`(left)
  - Move test cases: `Ctrl` + ← / →

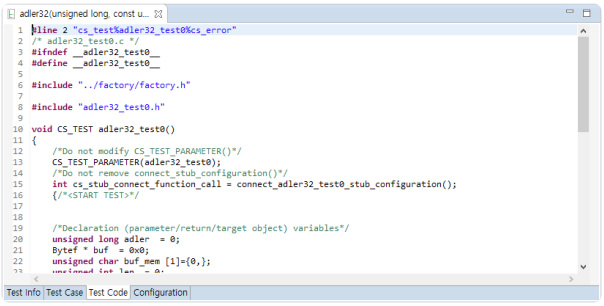
Change detection

If you edit and save the information of a test case that has already been executed, "The information has been changed after executing the test." is displayed at the top of the test editor.


adler32_combine(unsigned long,...
Test Case (zlib/adler32_combine_test0) #1
The information has been changed after executing the test.

10.3. Test Code

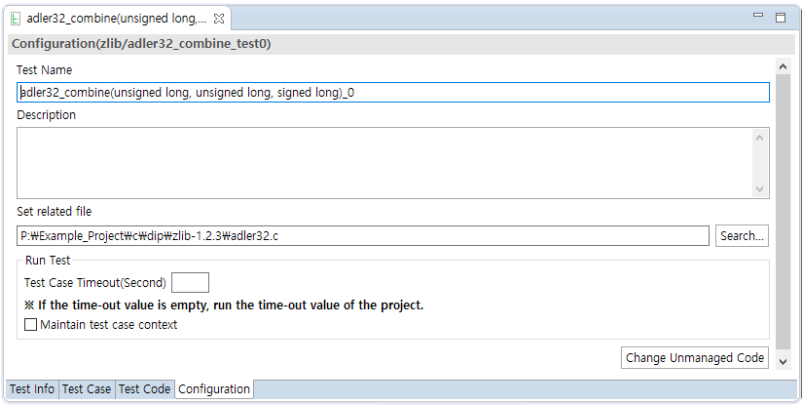
You can check the test code by clicking the Test Code tab in the test editor.



- If the user edits and saves the test structure tree, the changes are automatically reflected in the test code.
- If the return type of the function under test is a primitive type, a macro that outputs the return value is automatically generated.
- If you want to modify the test code yourself, you must switch the test to unmanaged code.
- The test execution function must have a return type and an input parameter of a void type.

10.4. Configuration

On the configuration tab, you can make various settings for unit testing.



- If you modify and save the test name, the test name that appears in the unit test view changes.
- You can enter a description for the test.
- With the related file settings, you can select the file to which the function under test belongs.
- You can set the timeout for the test.
- You can choose whether to keep the context for the test.
- If you want to write test code yourself, you need to do [Change Unmanaged Code].

10.5. Test Macro

Controller Tester provides several macros to help users write test code more easily. You can check and write the macro provided using the shortcut(`Ctrl` + `Spacebar`) in the Test Editor.

ASSERT macro

Examine the conditional expression and print the success/failure in the Test Case Tab.

Macro	Parameter	Example
CS_ASSERT(_b)	_b: conditional expression	`CS_ASSERT(val!=1);`
CS_ASSERT_MSG(_B, _msg)	_B: conditional expression _mgs: message to print when the conditional expression is false	`CS_ASSERT_MSG(val==1, "val is not 1!");`

Output macro

Print the values of specific variables in the Test Case Tab.

Macro	Parameter	Example

CS_INT_OUTPUT(_v, _s)
CS_UINT_OUTPUT(_v, _s)
CS_FLT_OUTPUT(_v, _s)
CS_STR_OUTPUT(_v, _s)
	v: a variable to print the value
	_s: a name to show in the Test Case Tab
		`CS_INT_OUTPUT(int_var, "int_var_name");`
		`CS_UINT_OUTPUT(unsigned_int_var, "unsigned_int_var_name");`
		`CS_FLT_OUTPUT(float_var, "float_var_name");`
		`CS_STR_OUTPUT(string_var, "string_var_name");`

Input macro

Pass test data to the function under test.

Macro	Parameter	Example

CS_INT_INPUT(_t, _s)
CS_UINT_INPUT(_t, _s)
CS_FLT_INPUT(_t, _s)
CS_STR_INPUT(_t, _s)
	_t: a type of the variable
	_s: a name to show in the Test Case Tab
		`CS_INT_INPUT(int, "int_var_name");`
		`CS_UINT_INPUT(unsigned int, "unsigned_int_var_name");`
		`CS_FLT_INPUT(float, "float_var_name");`
		`CS_STR_INPUT(char*, "string_var_name");`

Address-related macros

If there is a part that directly assigns or fetches the value in the embedded address in the converted source code, it may not operate normally when executed on the local computer. In this case, you can use an address-related macro for the virtual address.

Macro	Description
CS_VIRTUAL_ADDR(_b,_e)	Creates the space from the address (_b) to the address (_e)
CS_ADDR_ASSIGN(_t,_a,_v) CS_ADDR_SET(_t,_a,_v)	Assigns the value (_v) of type (_t) to the address (_a)
CS_ADDR_GET(_t,_a)	Fetches the value of type (_t) from the address (_a)
CS_VIRTUAL_ADDR_CLEAR()	Frees the memory space created

Address-related macro example

```
// Create a virtual memory area of size 100 from 0xFFE40000U.
CS_VIRTUAL_ADDR(0xFFE40000U, 0xFFE40000U+100);

// Assign value 10 of int type to 0xFFE40000U
CS_ADDR_ASSIGN(int ,0xFFE40000U, 10);
CS_ADDR_SET(int ,0xFFE40000U, 10);

// Return the value from 0xFFE40000U to the variable a.
int a = CS_ADDR_GET(int ,0xFFE40000U);

// Return an address of 0xFFE40000U into variable ptr of pointer type.
int* ptr = CS_ADDR_PTR(0xFFE40000U);

// Frees the memory space created.
CS_VIRTUAL_ADDR_CLEAR();
```

For more information about address-related macros, see the Virtual Address Usage Guide page in the User Guides.

Other macros

Macro	Parameter	Description	Example
CS_LOG(_msg)	_msg: log message	Outputs user log	`CS_LOG("User Log");`
CS_TESTCASENO()		Returns the number of the test currently running	`int testCaseNum = CS_TESTCASENO();`

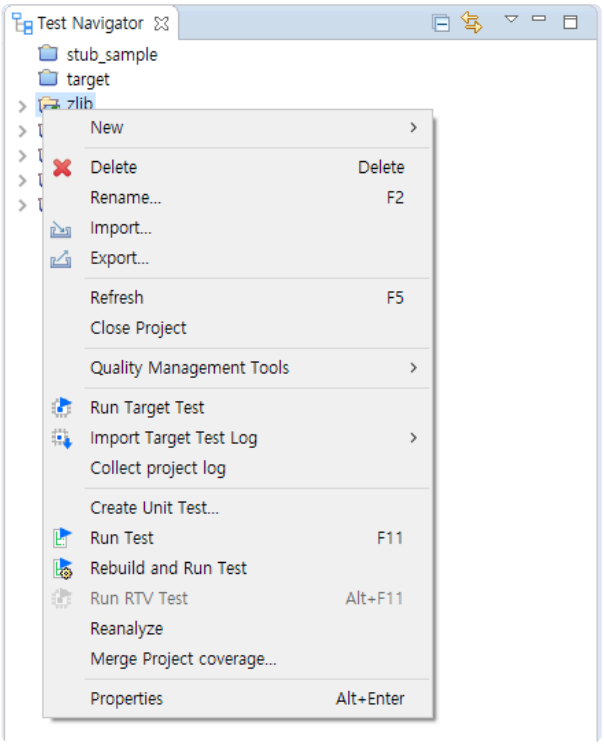
11. Test Run

Testing can be performed in a variety of ways.

Run all tests

Run the selected tests in the [Unit Test] view and [Integration Test] view.

1. Select the project to run the test, right-click ans slect [Run test]



2. A progress monitor of the test opens. When the test is completed, you can check the results through each test view.

Run unit tests

Run the selected unit tests in the [Unit Test] view.

1. Click the [Run] button on the dashboard in the [Unit Test] view.

2. A progress monitor of the test opens. When the test is completed, you can check the results through each test view.

To perform only a specific test case, select the test case you want to run and right-click the mouse and click the [Run Test Case] menu.

Run integration tests

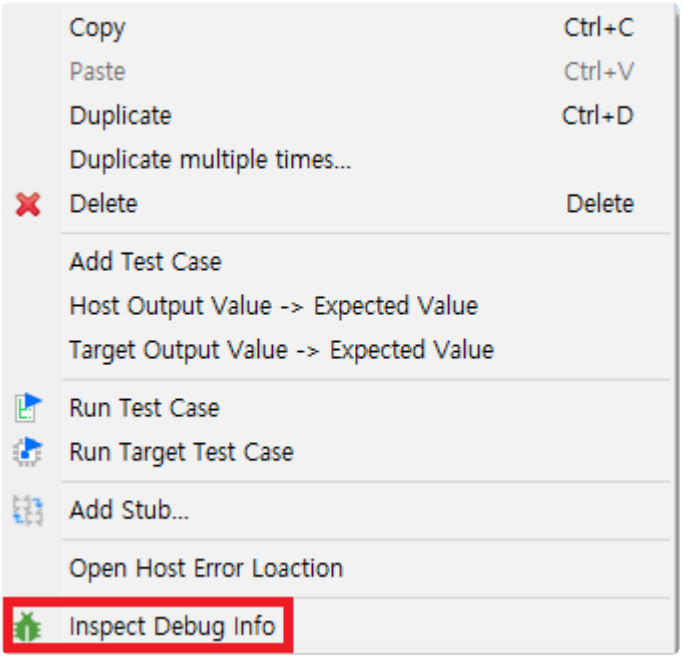
Run the selected tests in the [Integration Test] view.

1. Click the [Run] button on the dashboard in the [Integration Test] view.

2. A progress monitor of the test opens. When the test is completed, you can check the results through each test view.

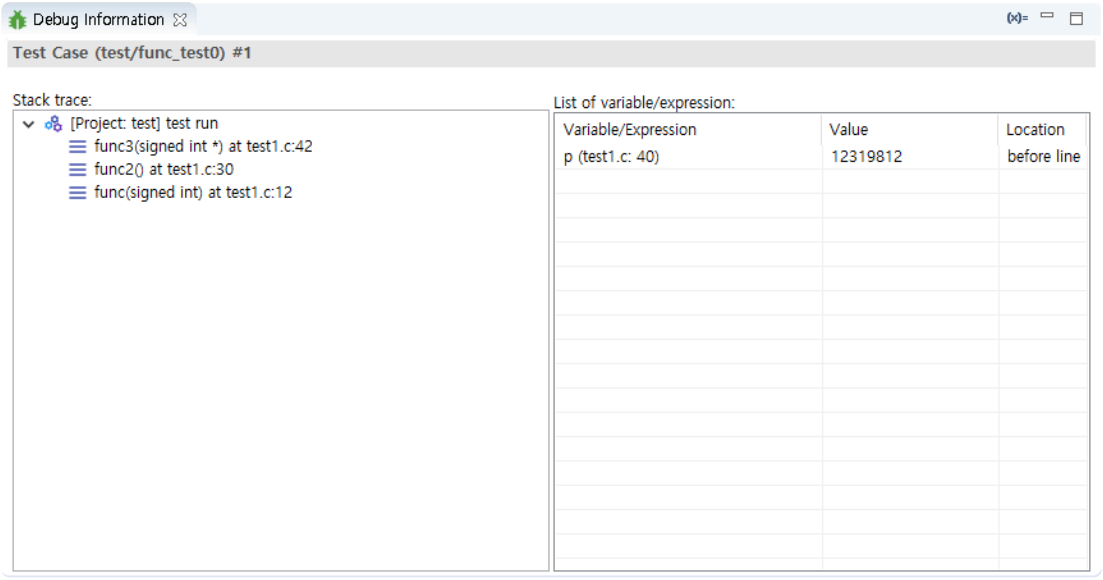
12. Inspect Debug Information

If an error occurs while running the test case, you can use the [Inspect Debug Info] function to determine the cause of the error.
[Inspect Debug Info] function can be checked in the unit test view with the context menu of the test case.

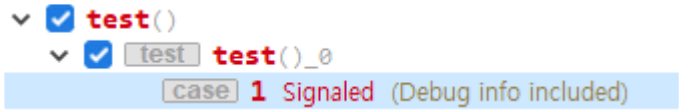


> Test cases that have not been executed are automatically executed before [Check Debug Information] is performed.

When [Inspect Debug Info] is performed, debug information is displayed in the [Inspect Debug Info].



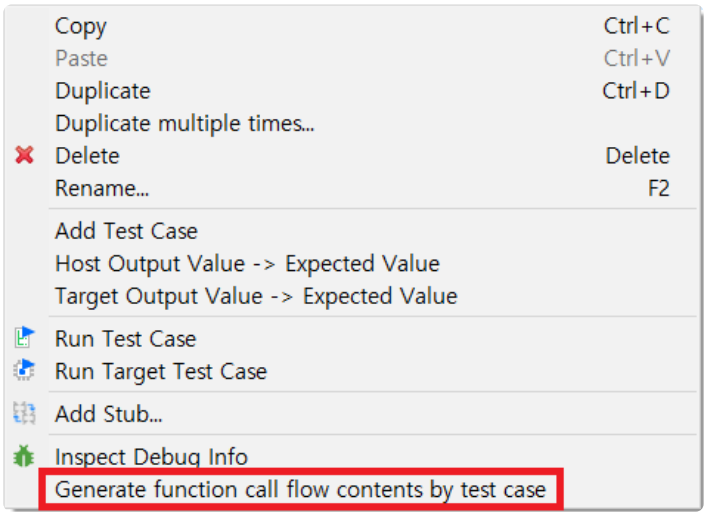
The test case (Debug info included) is displayed in the test case where the [Inspect Debug Info] was performed.


test()
test test()_0
case 1 Signaled (Debug info included)

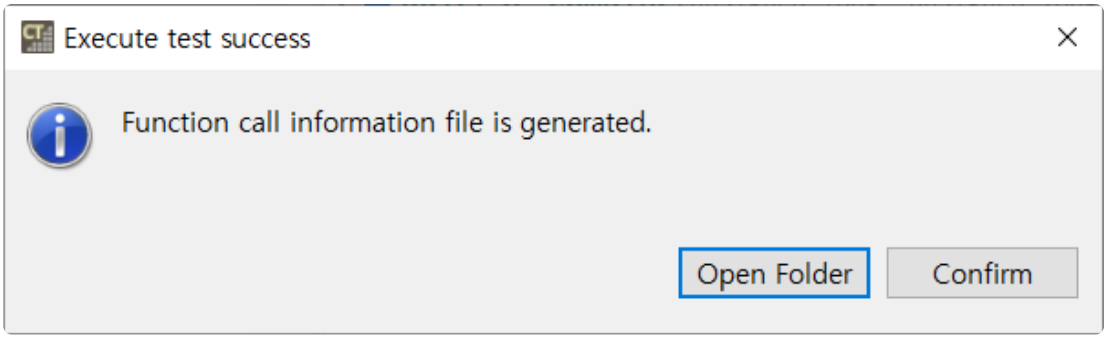
13. Generate Function Call Flow Contents

Using [Generate Function Call Flow Contents by Test case] feature, you can print the called functions sequentially when you run the test case.

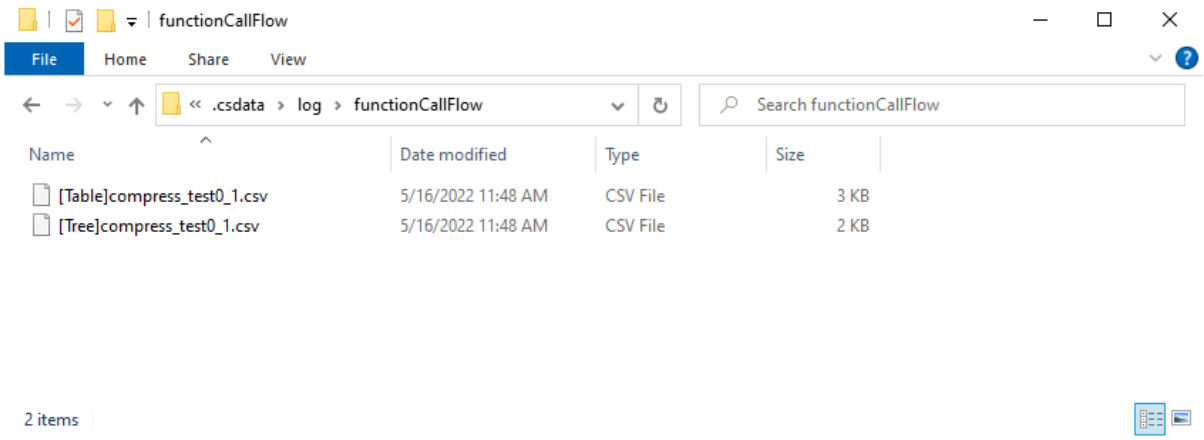
1. At the Unit Test View, right-click the test case and select [Generate Function Call Flow Contents by Test case].



2. When [Generate Function Call Flow Contents by Test case] is executed, the [Execute Test Success] window pops up.



3. Click the [Open Folder] button in [Execute Test Success] window. The directory containing the function call flow content file is opened.



> The output file name is in the format of `[type]test name_test case number.csv`.

- The file includes name and parameters of the functions. There are two types in displaying function call flow.
  - Table
    - The calling relationship is displayed in two columns.
    - The caller functions are in the first column, and the callee functions are in the second column.
  - Tree
    - The function call information is displayed in multiple columns depending on the call depth.
    - The function call information is displayed on different lines if the call is located at different locations.

- If a function is consecutively called several times, calls are printed in an abbreviated way.
  - The number of calls is indicated by appending `[called # times]` after the function signature.
  - e.g., If the a function is consecutively called 5 times, `[called 5 times]` is appended to the first function signature.

Example

```
// example code
void callee() { /*doing something*/ }
void callee2() { /*doing something*/ }
void func1() { callee(); callee2(); }
void func2() { callee(); callee(); callee(); }
void caller() { func1(); func2(); }
```

- Table

caller-function	callee-function
caller()	func1()
func1()	callee()
func1()	callee2()
caller()	func2()
func2()	callee() [called 3 times]

- Tree

caller()	func1()	callee()
		callee2()
	func2()	callee() [called 3 times]

! Constructors, destructors, operators and functions without definition (e.g., system functions) are excluded from the output. Functions called from ternary operators or from try and catch statements may not output properly.

14. Properties

Controller Tester provides a page that enables to set the properties of the project, module, and source files.

The property page types are:

- Project Properties
- Module Properties
- Source File Properties

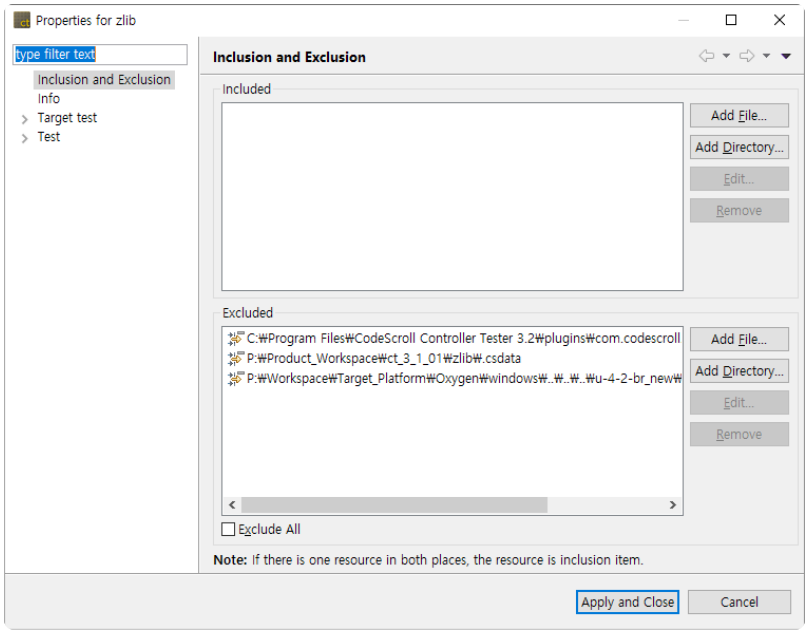
14.1. Project Properties

You can edit the properties of the created project.
Select [Project]-> [Properties] from the menu or right-click the project in the Test Navigator view and click the [Properties] menu.

Inclusion and Exclusion Info

You can set what to exclude or include when analyzing the project.

> To analyze only selected directories or files, add the targets to be analyzed to [Included] and check [Exclude All].

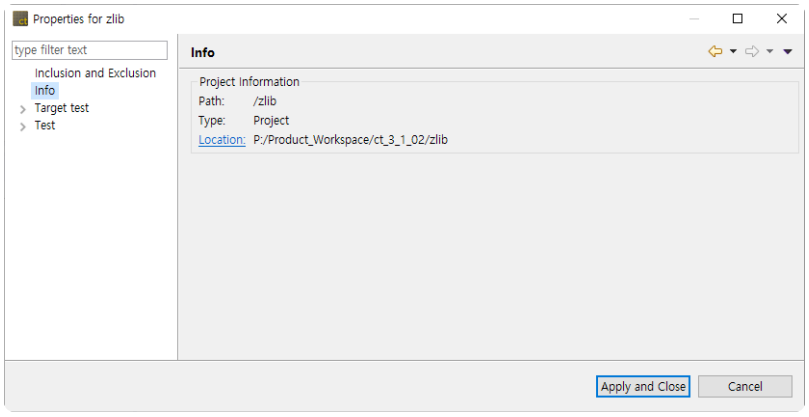


- Add targets to include in the analysis.
    1. Click the [Add File] or [Add Directory] button.
    2. Select files or a directory to analyze.
    3. Click the [Open] or [OK] button.

- Add targets to exclude from the analysis.
    1. Click the [Add File] or [Add Directory] button.
    2. Select files or a directory to exclude from the analysis.
    3. Click the [Open] or [OK] button.

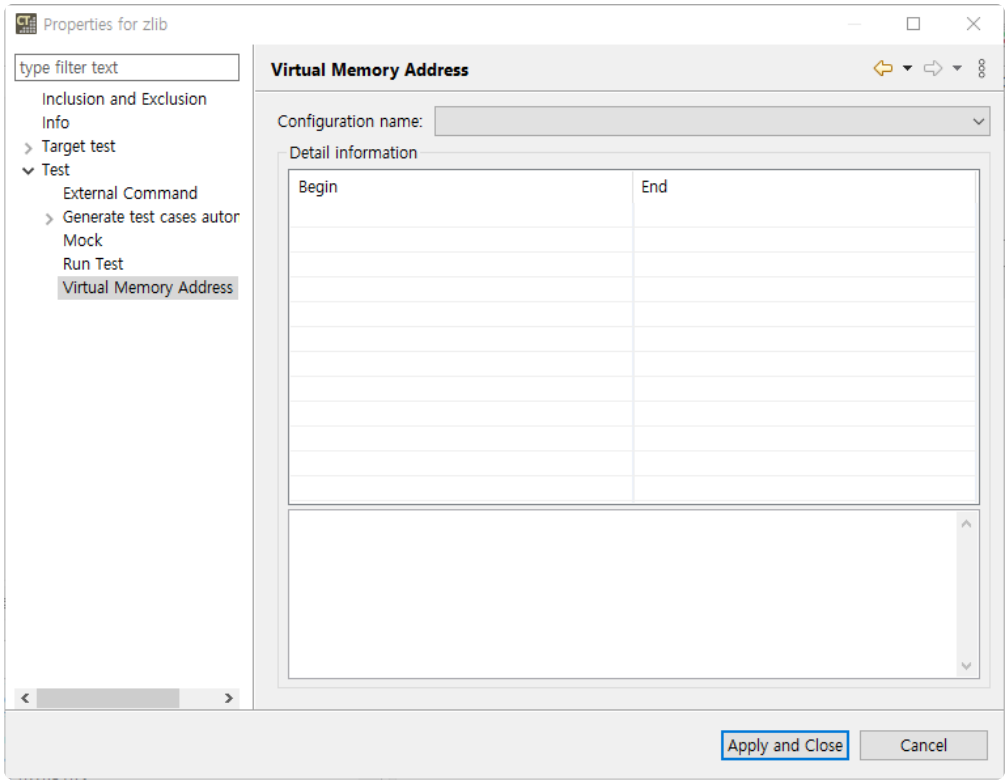
Info

Shows brief information(path, type, location) about the project.



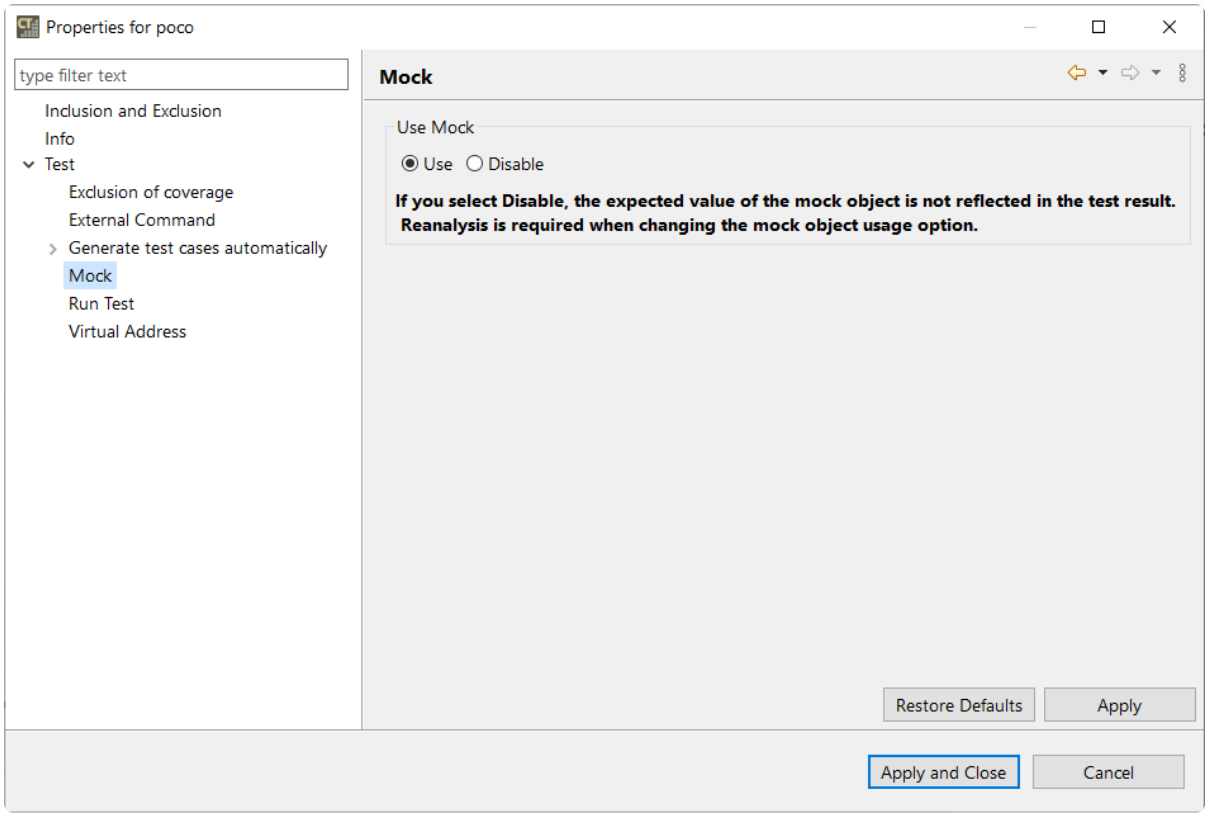
Virtual Memory Address

You can select a memory setting managed by the Virtual Memory Address preference.



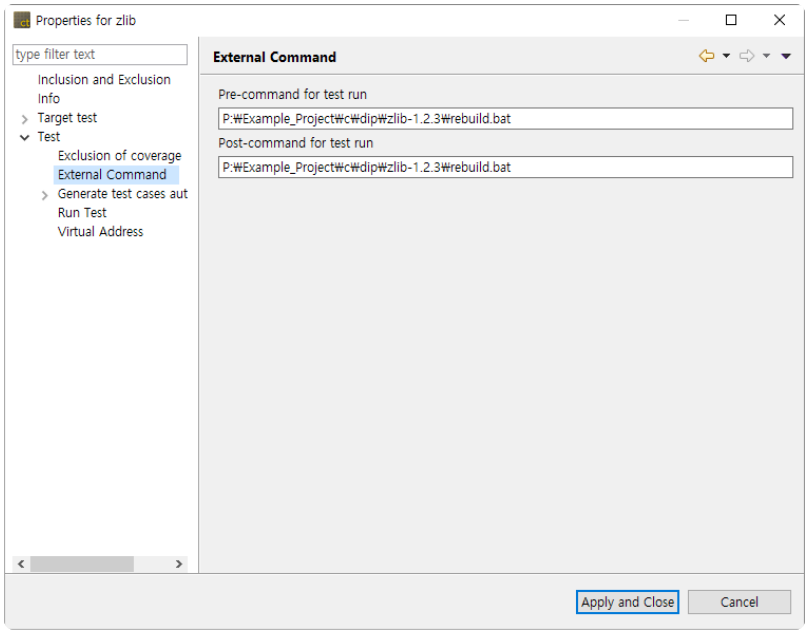
Mock Object

For C++ project, you can select an option to use mock objects. [Use] is selected by default for toolchains(GCC 6.0 or later, Visual Studio 2015 or later) that can use mock objects.



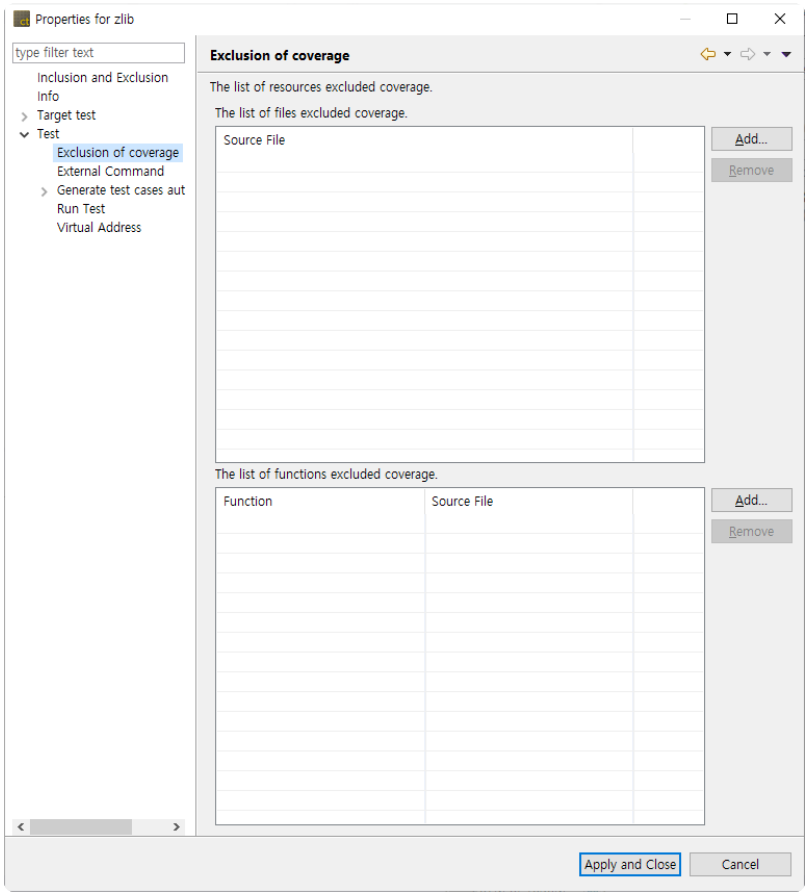
External Command

You can enter external commands to run before or after the test run.
In [Pre-command for test run], you can enter a command or batch file to execute an external command before running the test.
In [Post-command for test run], you can enter a command or batch file to execute an external command after running the test.



Exclusion of coverage

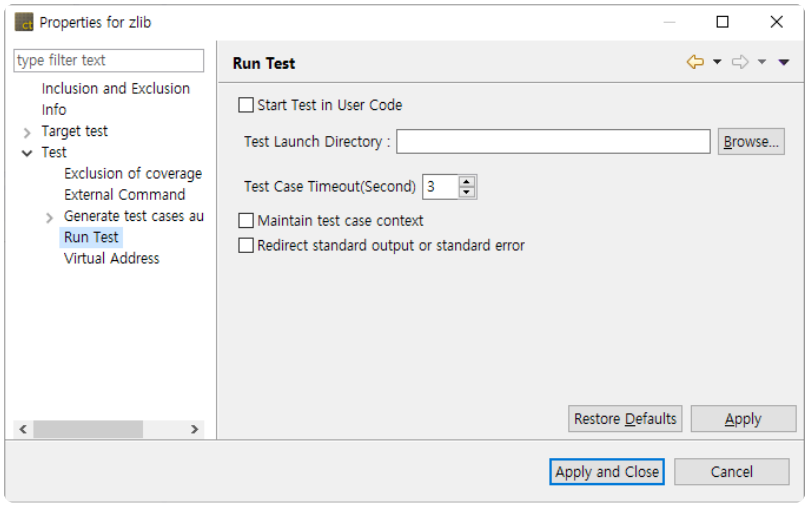
You can set coverage exclusions on a per-file or per-function basis.
You can see that the coverage of the excluded function is excluded from the coverage results obtained by running the test.



Run Test

Set the generated tests to run in a user-specified directory, not in a workspace. In this case, you do not need to copy the library or header files used by the target under test to the workspace or change the link settings.

Click the [Start Test in User Code] checkbox.
Select the home directory to run the test and click the [Apply] button.



[Test Case Timeout] Set the time to be used as the timeout judgment when executing the test case. If the test execution time exceeds the test case timeout time, the test for that test case ends and the result is reported as a timeout.

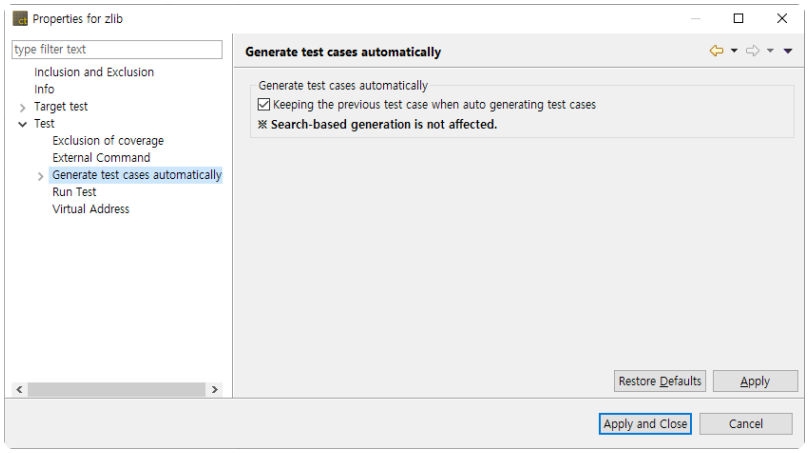
[Maintain test case context] is a feature that performs all test cases with one launcher. The previous test execution result (static, global variable) is maintained during the test, and each test case result value does not appear independently. This makes the relationship between the test case result values clear.

[Redirect standard output or standard error] is a feature to record standard output and standard error in the log file for each test case.

Generate test cases automatically

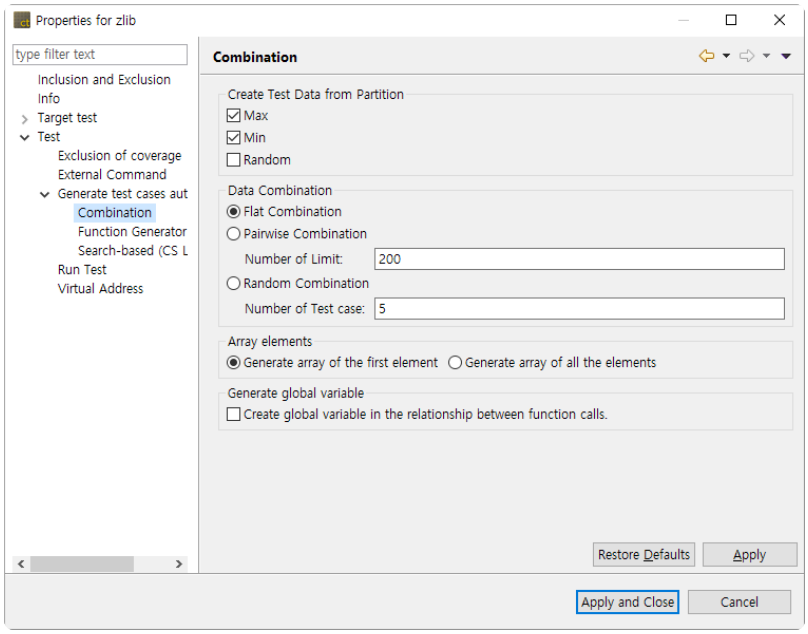
You can set to keep previous test cases when automatically generating test cases.

> ! The search-based generation is not affected.



Combination

You can set up how to extract test data if the variable partition is range.



Select the desired test data extraction method by checking the checkbox. The meaning of each value is as follows.

Method	Description
Max value	Selects the maximum value of the partition section.
Min value	Selects the minimum value of the partition section.
Random value	Selects a random value among partition sections.

You can select multiple test methods of data extraction.
When all are unchecked, the default is automatically set to the maximum and minimum values.
[Data combination] supports Flat, Pairwise and Random. The meaning of the combination is shown in the following table.

Combination	Description
Flat	Combines simply based on variables having the largest number of test data.
Pairwise	Combines so that each selected parameter data is paired at least once with the parameter data other than itself. Pairwise is performed as many as the maximum number(default 200) of symbols used for the combination, and when the number of symbols is exceeded, Flat is performed.
Random	Combines the test data as many as the number of test cases that the user defines any value between the minimum value and the maximum value for the variable partition of input parameters(default 5).

[Array elements] is a feature to select how to generate test data for each array element.

Method	Description

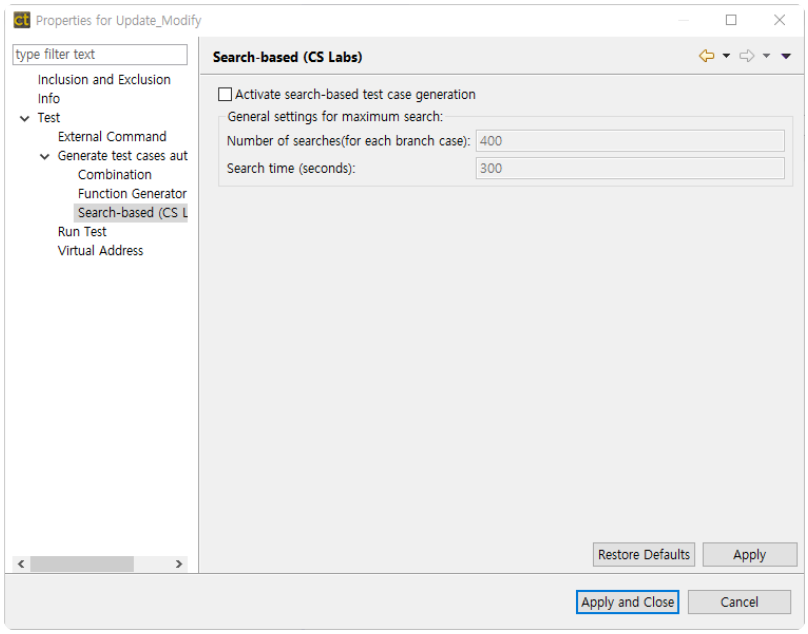
Generate an array of the first element	Creates only the first element regardless of the array size.
Generate an array of all the elements	Creates elements as much as the array size.

When creating a test, global variable creation is an option to create all global variables used by all functions called from the function under test.

Search-based (CS Labs)

> CS Labs allows you to preview new features that will be released in CodeScroll tools later. Based on feedback for the new features, CS Labs features may or may not be applied as formal features.

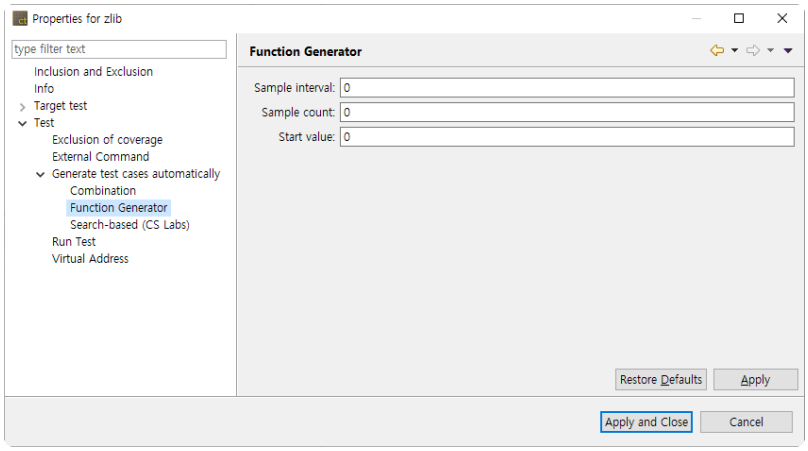
You can set the maximum search when automatically generating test cases based on search.
You can turn this feature on or off with [activating search-based test case generation] option. Reanalysis is required after turning the feature on.



Option	Description
Number of searches(for each branch)	Searches for the number of times entered for each branch combination.
Search time(seconds)	Searches for the time entered.

Function Generator

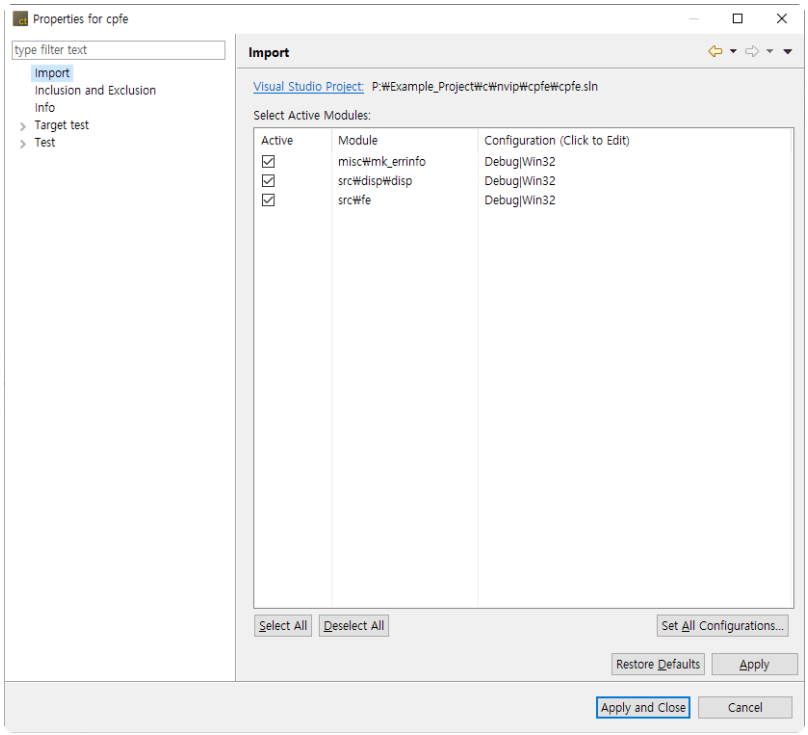
When using the function generator to automatically generate test cases, specify common setting values that each function has in common.



Option	Description
Sample interval	Interval of samples when sampling from function
Sample count	Number of samples when sampling from function (Number of test cases)
Start value	Default value at which the function starts, and generates a value based on the value

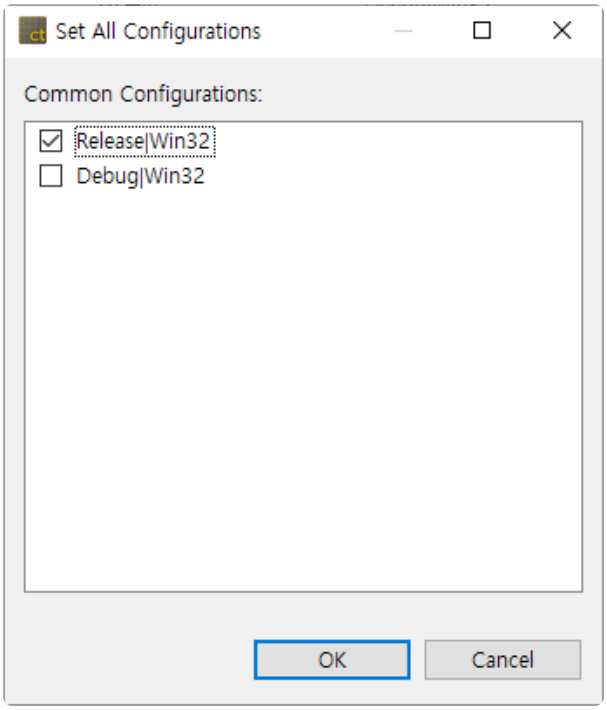
Import settings

Project created by Visual Studio project



You can see the path of the Visual Studio project used for creating the project and the modules active in the selected project.
You can change whether each module is active or not and the configuration.
You can change the configuration of all modules collectively through the [Set All Configurations] menu.

1. If you click the [Set All Configurations] button, a dialog box where you can select one of the configurations common to all modules opens.

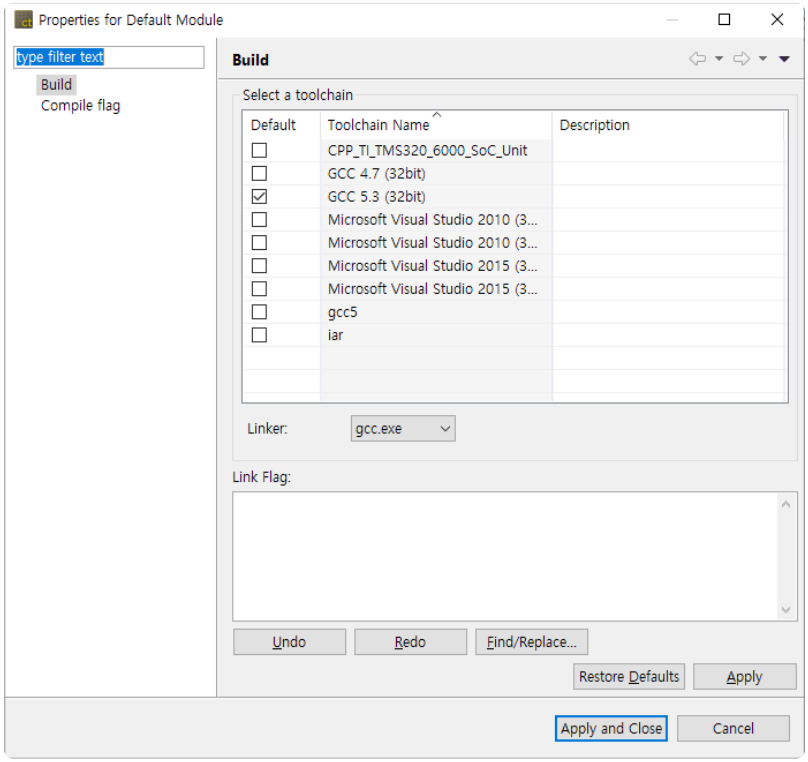


2. Select the configuration to be applied to all modules in common and click the [OK] button.

14.2. Module Properties

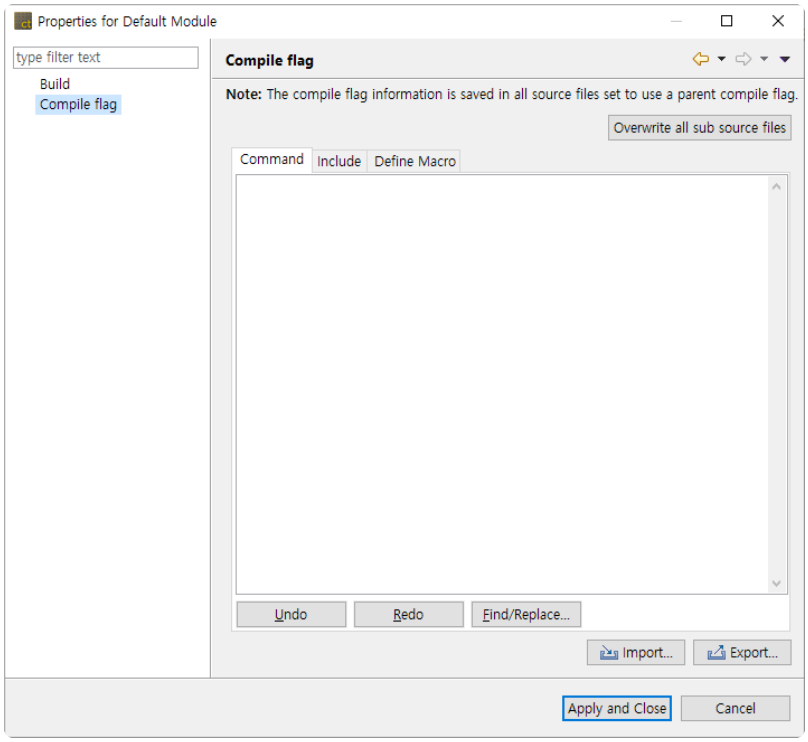
Build

You can set toolchain related information (toolchain, linker) and link flags for the module. Changing the toolchain also changes the toolchain set in all source files under the module.



Compile flag

You can set compile flag information for that module.

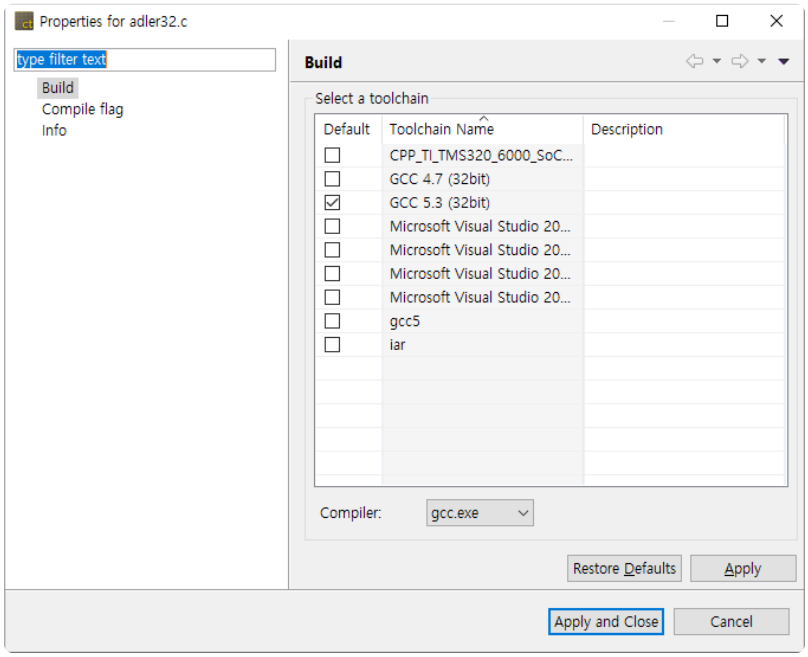


- Enter compile flags in [Command] tab. [Include] tab and [Define Macro] tab help to enter compile flags easily.
- [Overwrite all sub source files] overwrites the compile flag entered to all source files under the module.
- [Import] imports an external compile flag. [Export] saves the current compile flag externally.
    - Import
        1. Click the [Import] button.
        2. Select the file (.cf file) which has compile flags from the file open dialog.
    - Export
        1. Click the [Export] button.
        2. In the save dialog, enter the name and location of the file to enter the compile flag and save.

14.3. Source File Properties

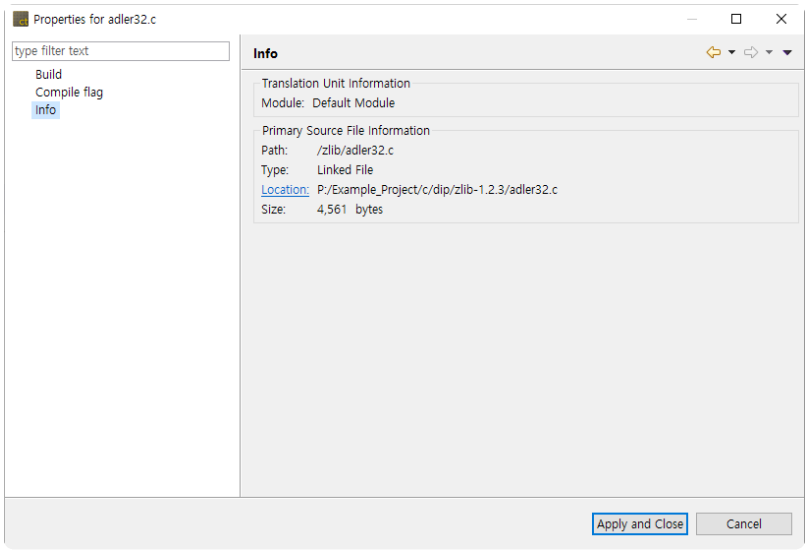
Build

You can set toolchain related information (toolchain, compiler) for the source file.



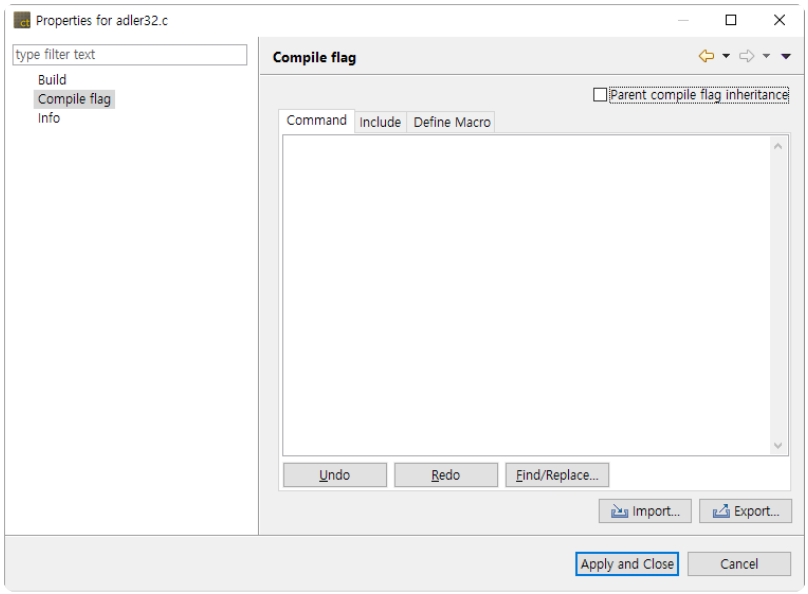
Info

Shows the information of the source file and the module containing the source file.



Compile flag

You can set compile flags for source files.



- If you check the [Parent compile flag inheritance] checkbox, the compile flag of the module containing the source file is applied. Conversely, you can change the current compile flag information by unchecking the checkbox.
- [Import] imports an external compile flag. [Export] saves the current compile flag externally.
    - Import
        1. Click the [Import] button.
        2. Select the file (.cf file) which has compile flags from the file open dialog.
    - Export
        1. Click the [Export] button.
        2. In the save dialog, enter the name and location of the file to enter the compile flag and save.

15. Preferences

In the preference menu, you can check or change the current settings of the tool.

The preferences provided by Controller Tester are as follows.

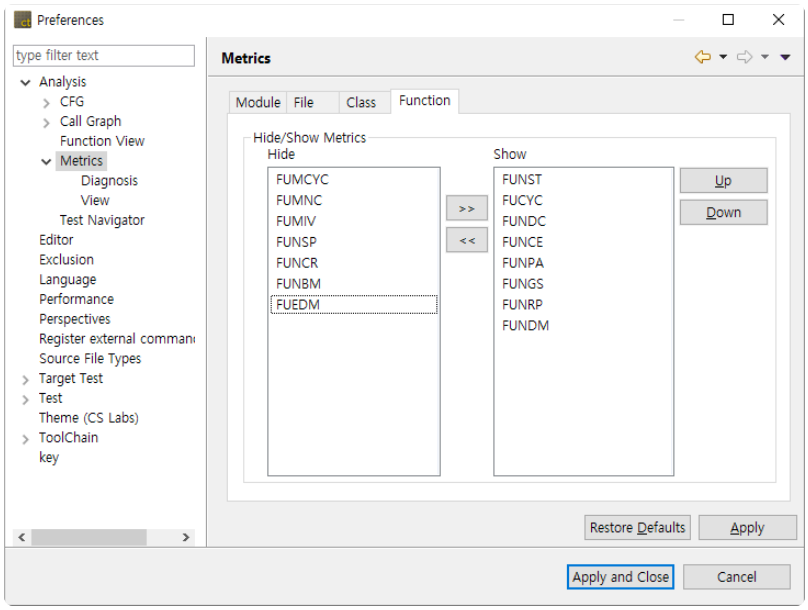
- Analysis
- Exclusion
- Performance
- Source File Types
- Language
- Test
- Toochain
- Editor

15.1. Analysis

You can check or change the settings related to the analysis.

Metric

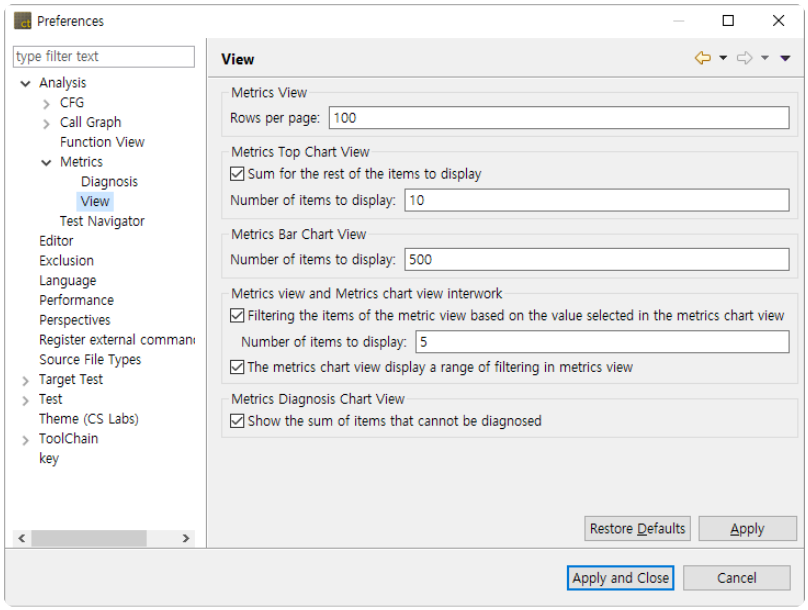
For any view showing metric data, you can set which metrics are visible or hidden and the order in which they are displayed.
The name of the selected metric is displayed as a tooltip message.



After selecting a metric, you can move to the ‘Hide’ list or the ‘Show’ list with the [>>] or [<<] button.
After selecting a metric in the ‘Show’ list, you can change the order in which the metrics are displayed with the [Up] or [Down] button.

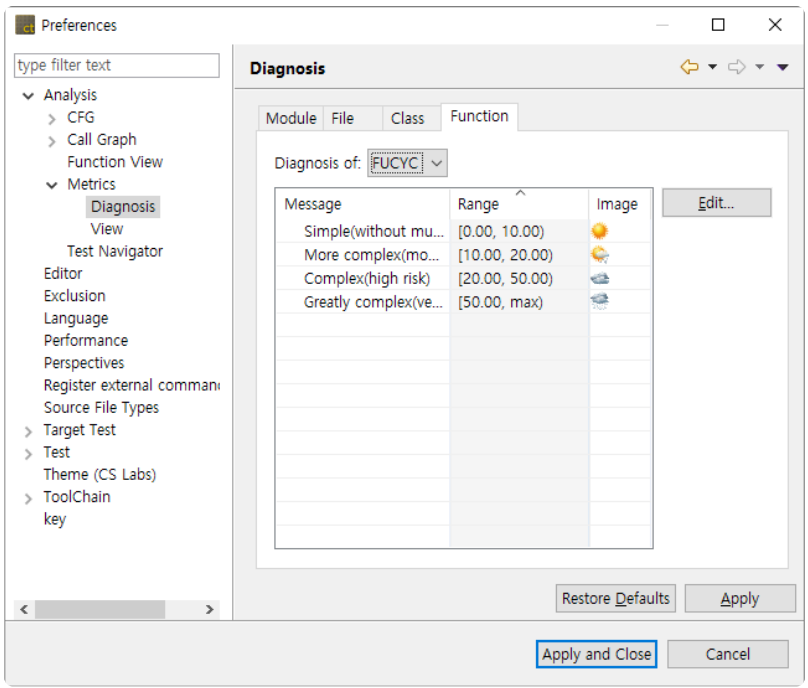
Metrics View

You can change the settings of the views showing metric data.

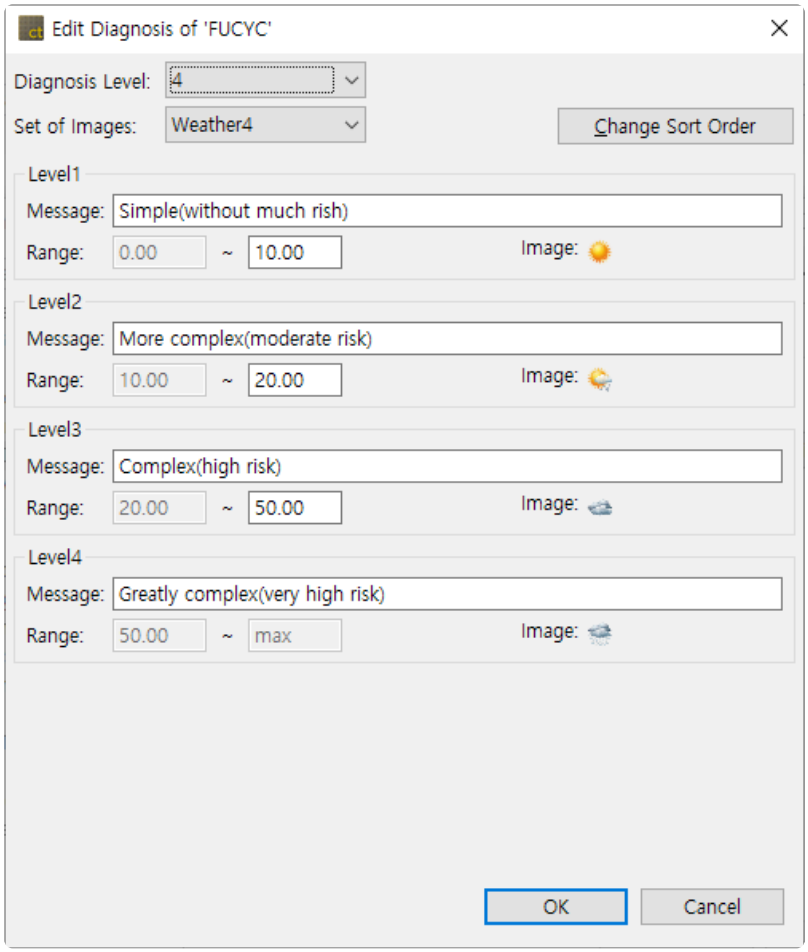


Diagnosis

You can check or set diagnostic information for metrics.
Set the message and diagnostic image (shown in the metrics view) to be displayed when the selected metric value is included in each range.



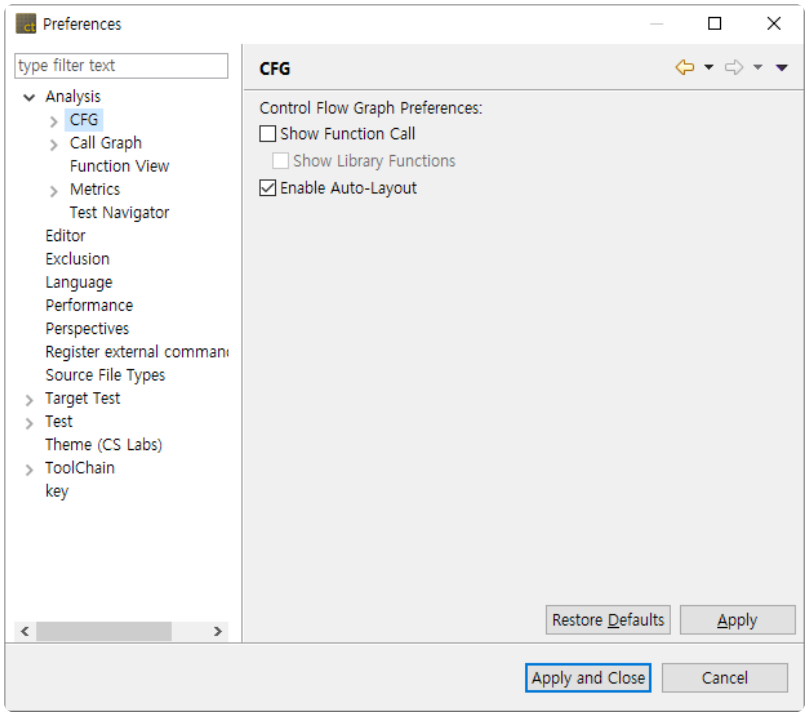
To edit the diagnostic information, click the [Edit] button.
A dialog box appears where you can edit the diagnostic information for the selected metric.



1. Set the diagnostic stage (2 ~ 5).
2. Sets the diagnostic image to show in the metrics view.
3. Press the [Change Sort Order] button to change the sort order of diagnostic images.
4. Enter the diagnostic message and diagnostic range for each step.
5. Select the [OK] button.

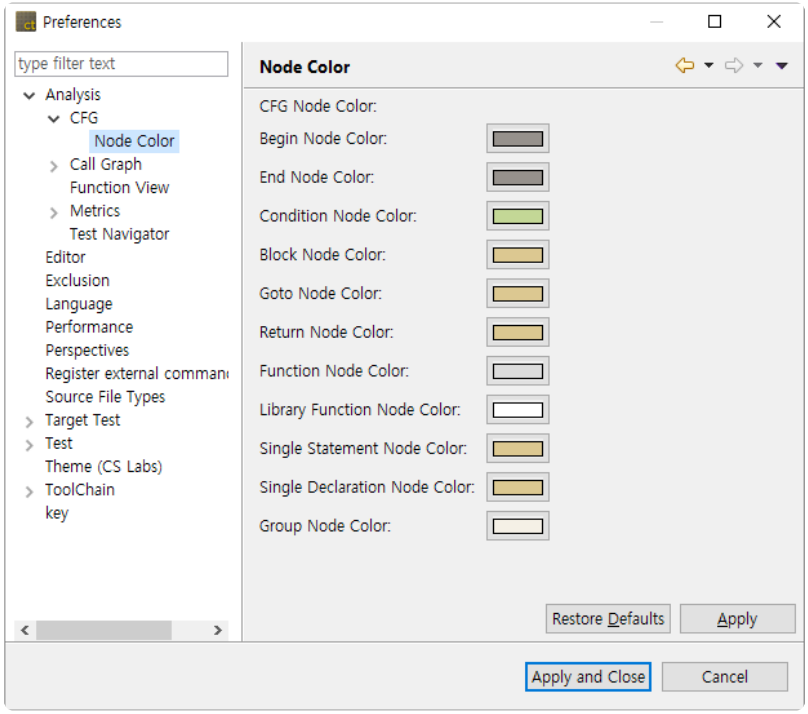
CFG

You can change the settings for the control flow graph displayed in the control flow graph view.



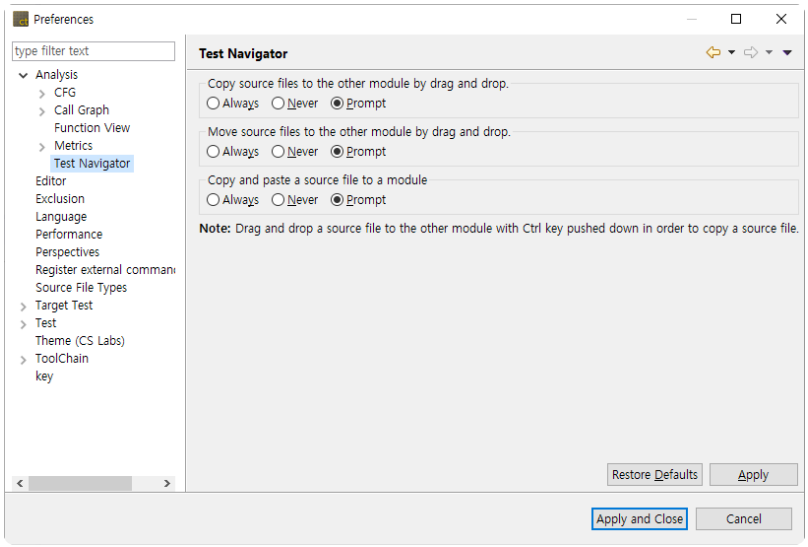
Node Color

You can change the color of nodes in the control flow graph.

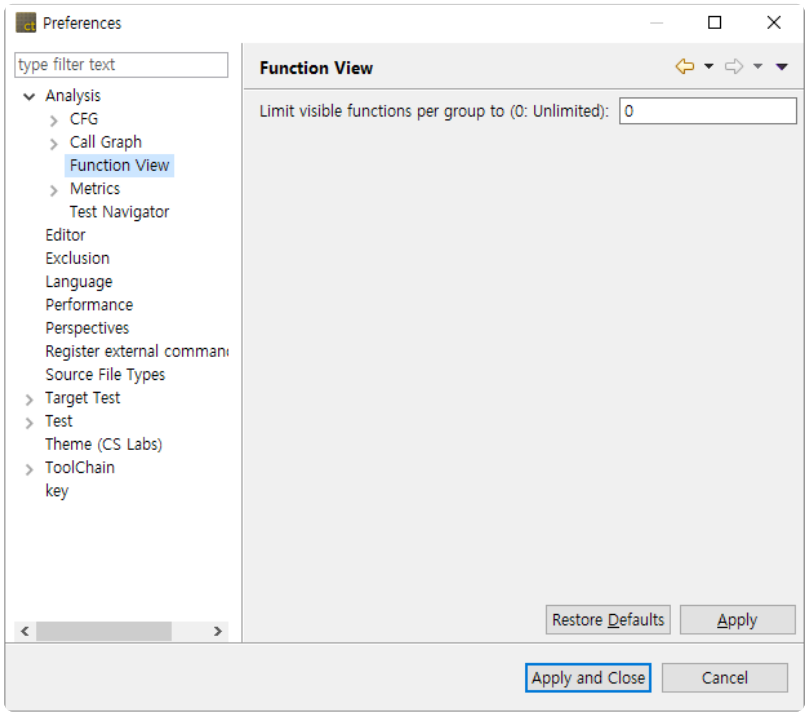


Test Navigator

For each operation that copies or moves source files in the Test Navigator view, you can set options to ask whether or not to continue.

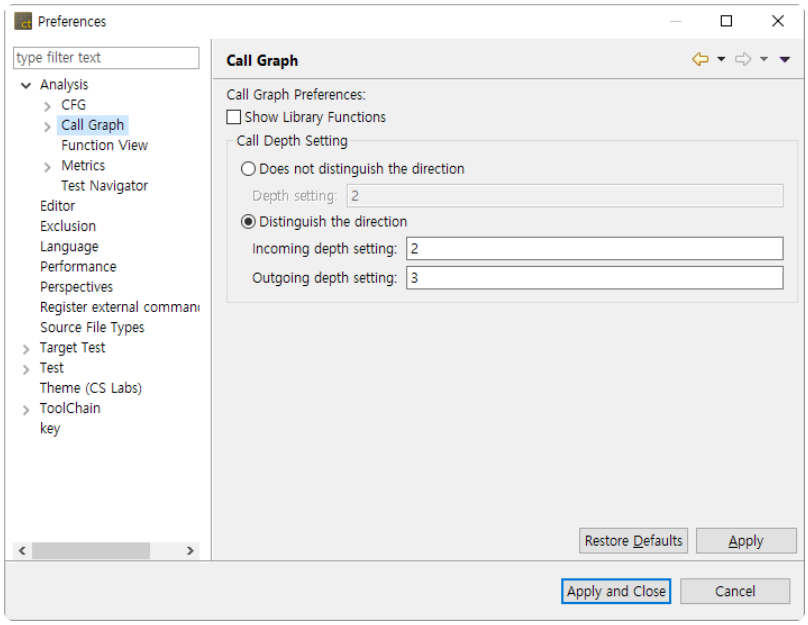


Function View



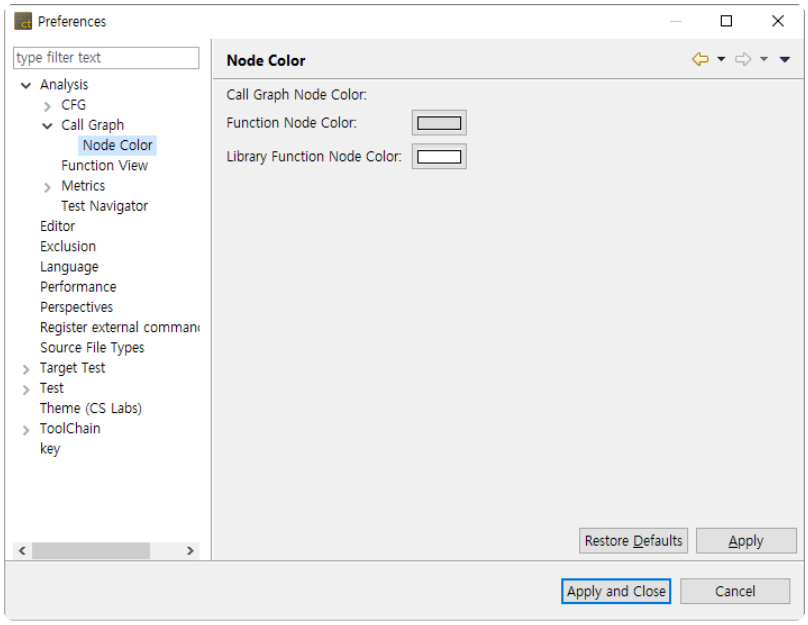
Call Graph

You can change the settings of the function call graph displayed in the call graph view.



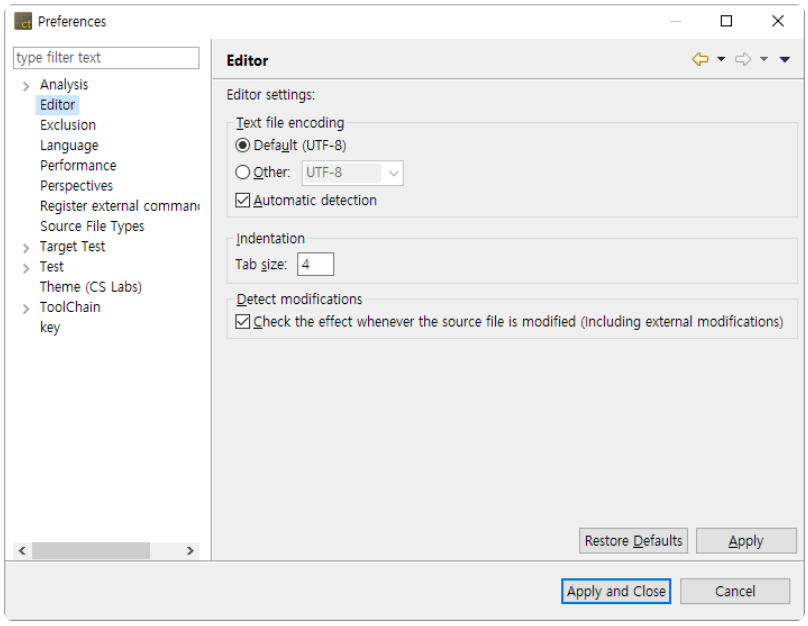
The [Show Library Functions] checkbox allows you to set whether or not to display library functions.
If you set the value of ‘Does not distinguish the direction’ to ‘2’ in ‘Call Depth Setting’, it shows the call information of one depth for both the caller and callee, centering on the selected function(Just the callers or callee of the selected function shown).

If you want to set the caller and callee separately, you can specify the value of each step by selecting the sub-option. You can change the color of nodes visible in the function call graph.



15.2. Exclusion

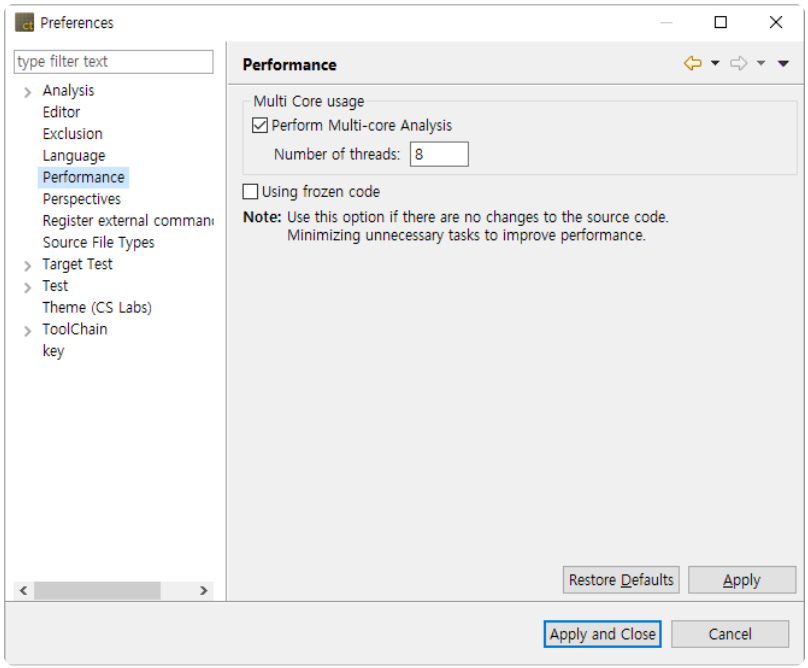
You can set the directories and files to exclude from the analysis.



- Add the analysis exclusion target.
    1. Click [Add].
    2. Select the directories or files to be excluded.
    3. Click [OK] or [Open].
- Delete the analysis exclusion target.
    1. Select the directories or files to be deleted.
    2. Click [Remove].

15.3. Performance

It provides the settings that can improve the work performance suitably for the user environment.



Multi-Core Usage

The option [Multi Core usage] improves performance by using multiple cores when executing the project analysis. (If PC has not enough CPU resource, it can affect the performance of the other tasks.)

- Perform Multi-core Analysis: Enter the number of the core to be used when analyzing the source code. (The range of the core number: 2 ~ the maximum number of cores in the execution environment)

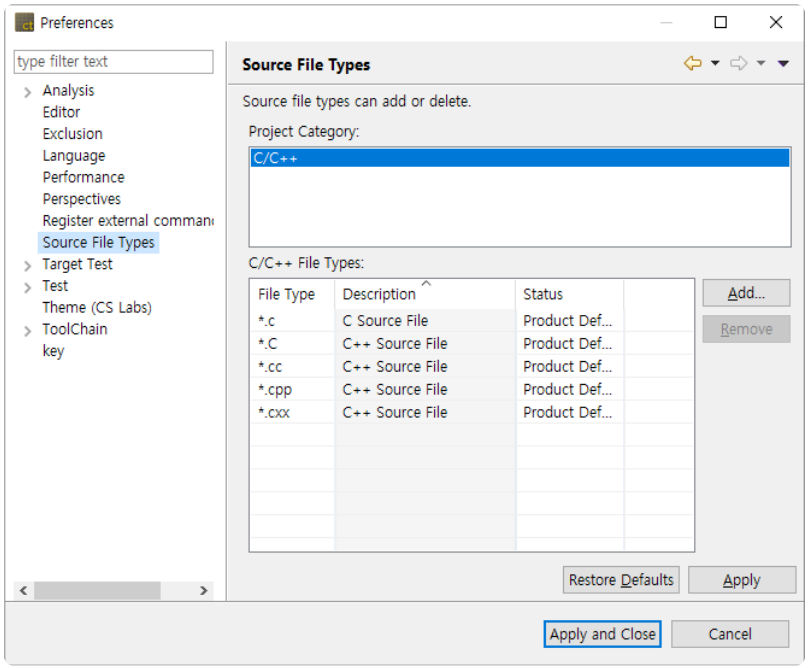
Using frozen code

The option [Using frozen code] improves the performance by minimizing unnecessary tasks (detecting and re-analyzing source code modifications etc.) assuming that there is no change in the source code configuration.
Use this option only if the PC performance is very slow and the source code configuration is never changed during testing. (If the source code or analysis setting is changed while the option is enabled, it may cause unexpected problems.)

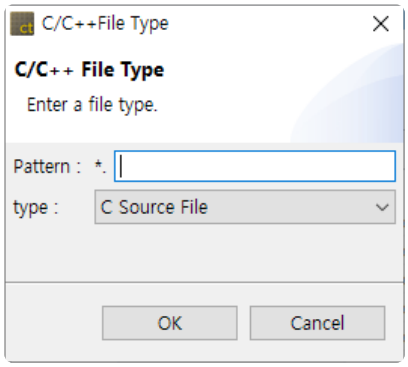
Difference with Editor – [Detect modifications] option
While if [Detect modifications] option of Editor is enabled, it analyzes the modified source code when executing [Analyze], if [Using frozen code] option of analysis is enabled, it does not analyze the modified source code until re-analyzed even though the source codes are modified. In addition, Editor – [Detect modifications] option is ignored.

15.4. Source File Types

You can set the extension to be used as a source file for each project type.



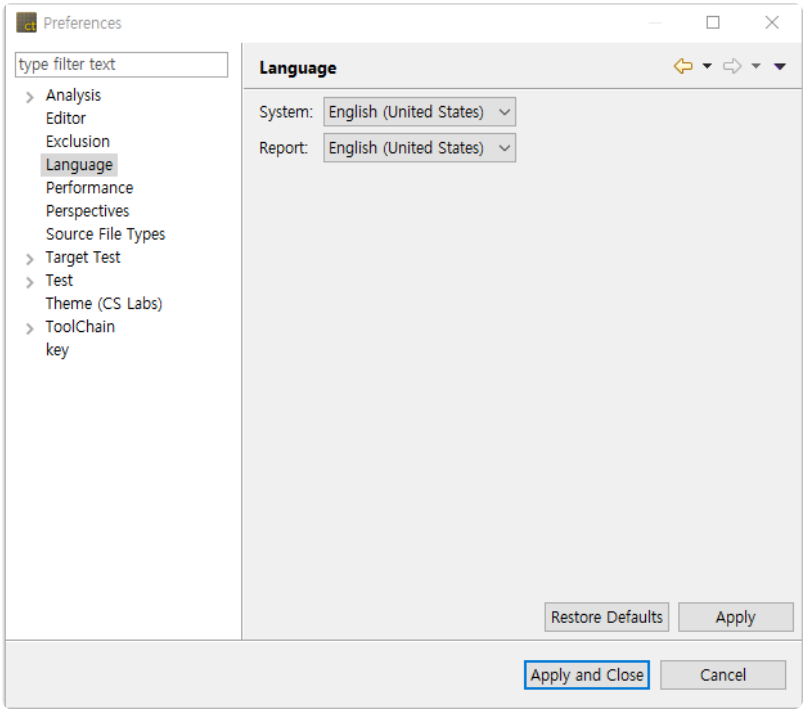
- Add a source file extension.
  1. Select the project type.
  2. Click [Add].
  3. Enter an extension, select the type and click [OK].



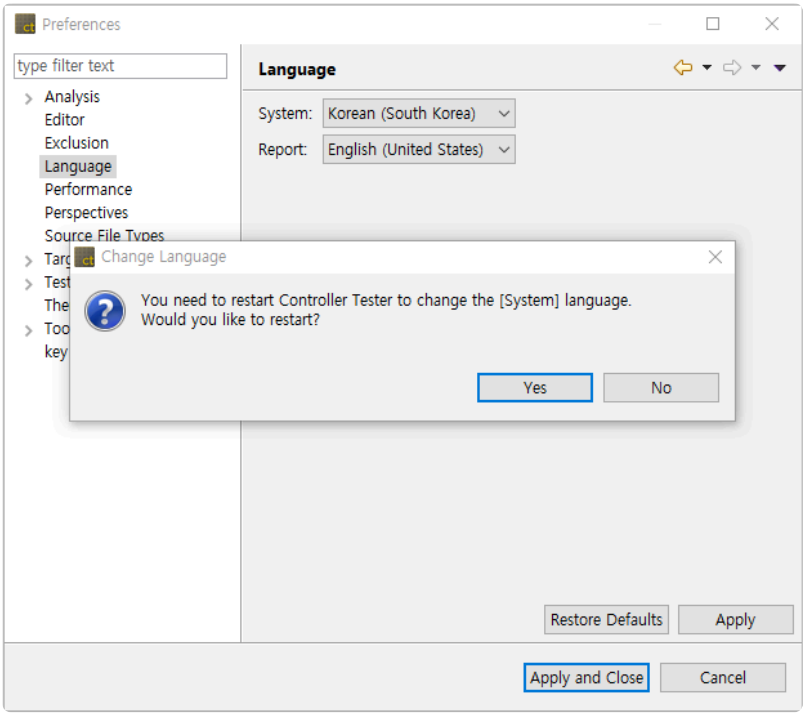
- Remove a source file extension.
  1. Select the project type.
  2. Select the extension to be removed.
  3. Click [Remove].

15.5. Language

Sets the language to be displayed in tools and reports.
You can select a language from the drop-down menu for [System] and [Report].



If you click the [Apply] button after changing the [System], a dialog box asking you to restart Controller Tester will open.



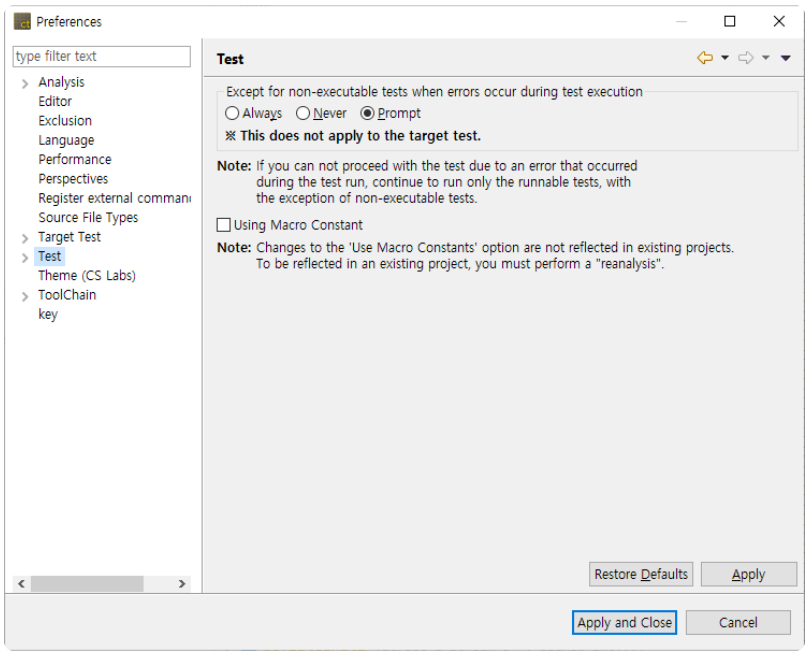
When you change the system language, the report language also changes automatically.

To set the report language differently from the system language, change the report language after restart.

15.6. Test

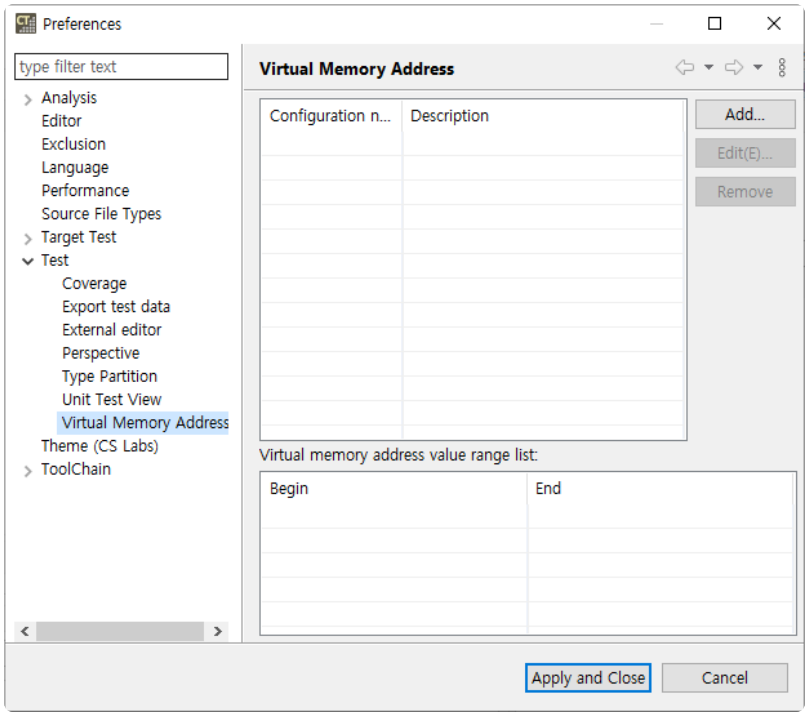
Provides settings related to editing test and execution.
[Except for non-executable tests when errors occur during test execution] is an option that allows you to execute only executable tests, excluding non-executable tests, when the whole test cannot be performed due to an error that occurs during test execution.
The [Using Macro Constant] option allows you to edit the test using macro constants defined in the source file (ex: using macro constants for input/expected values of array indexes and test cases).

> ! This setting only applies to converted projects.

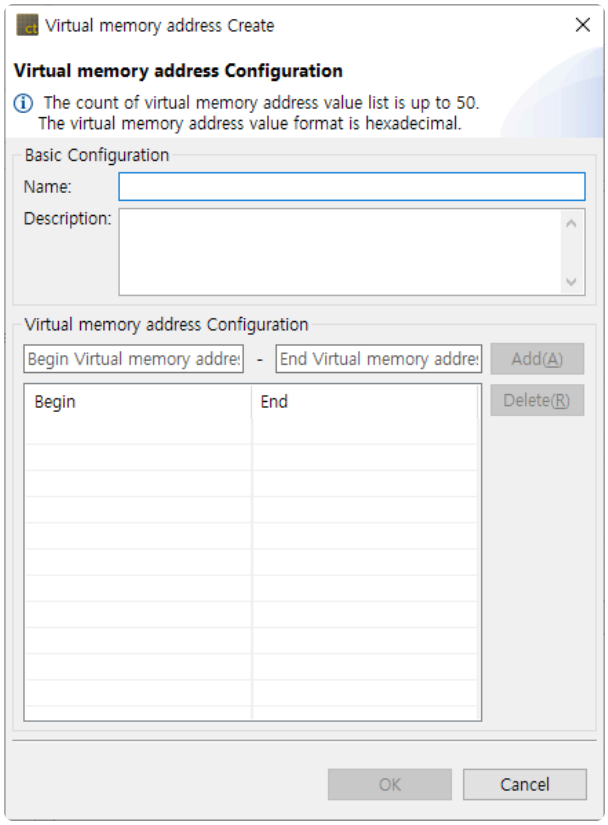


Virtual Memory Address

It provides the memory settings for the embedded environment test. You can manage the memory address group to be used in the project properties.



In the screen above, you can check the virtual memory address information set. Click [Add…] button to add the virtual memory address.

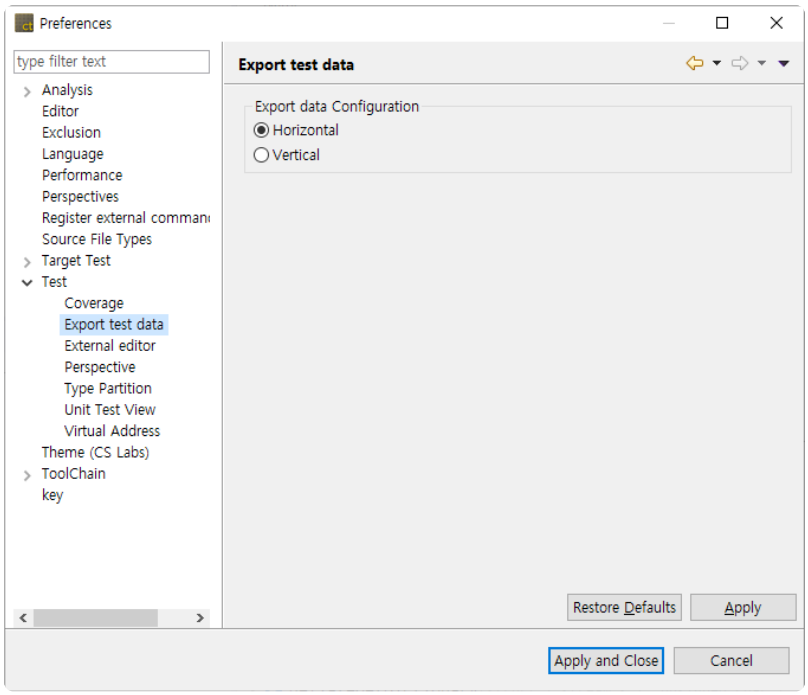


The virtual memory address areas can be specified up to 50 areas and it can be entered in hexadecimal.

Export test data

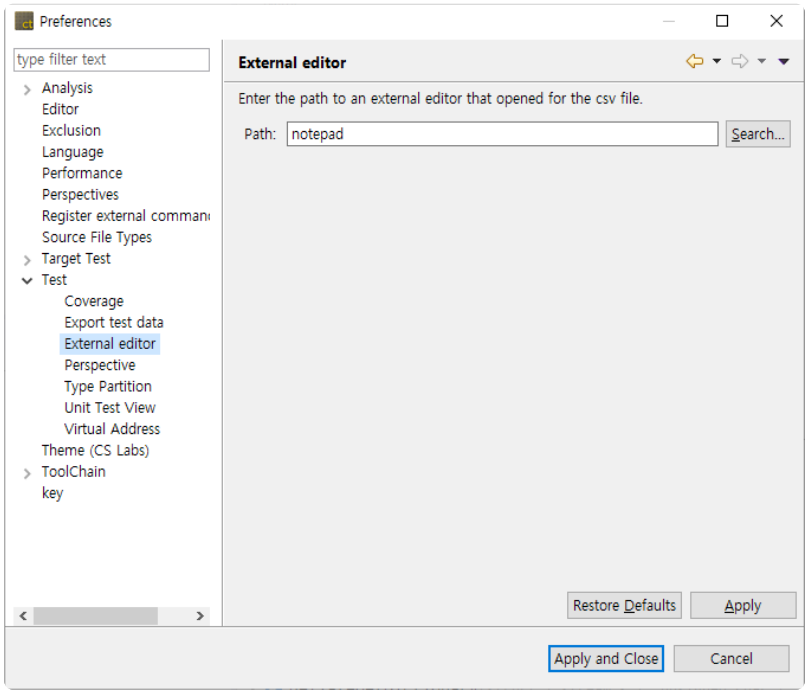
When exporting data, you can set the output direction of variables to either horizontal or vertical

Select the output direction of the export variable and click [Apply].



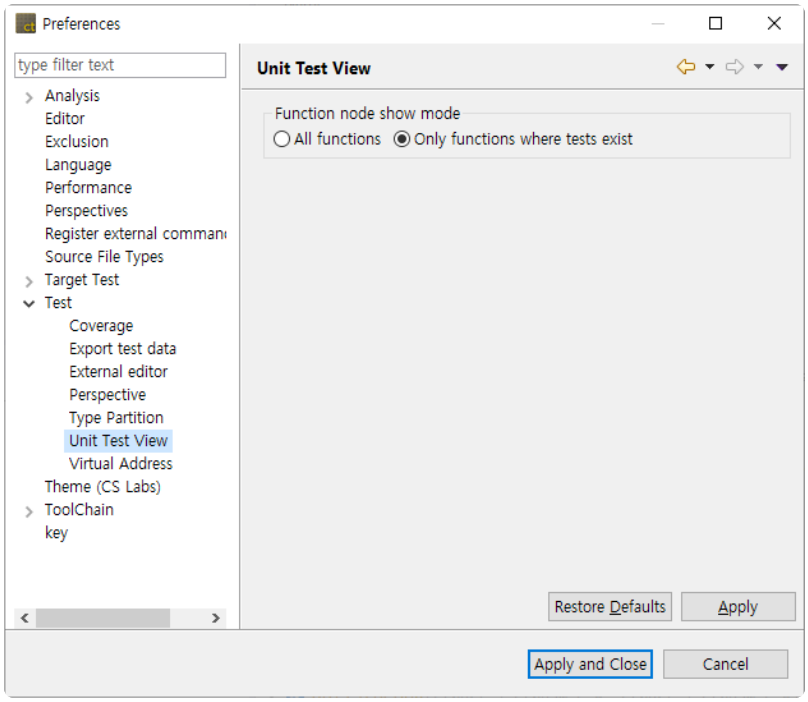
External Editor

You can set the external editor to open CSV file.
Enter the path of the external editor and click [Apply].



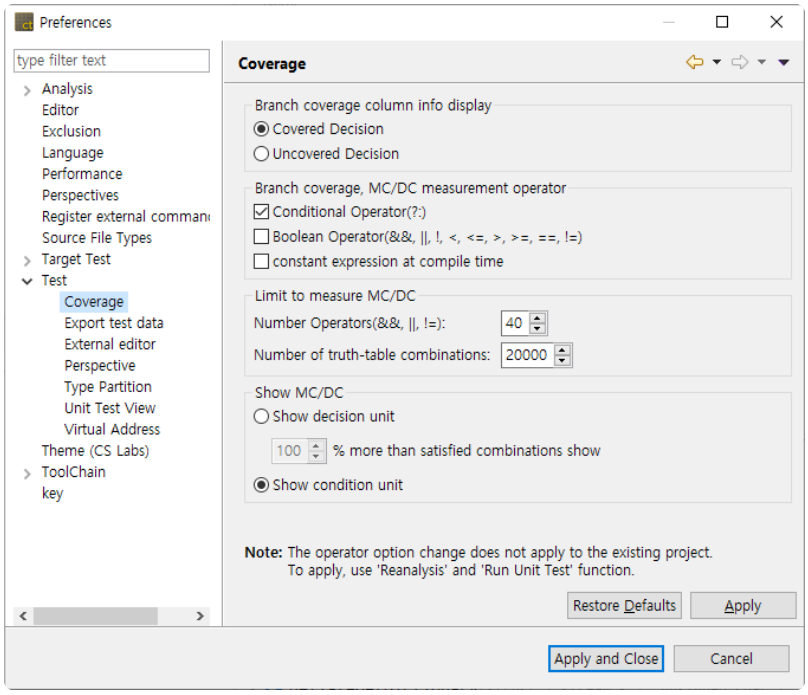
Unit Test View

You can set the function node display in the Unit Test view included in the test perspective that UI/UX is improved.



Coverage

You can set the branch coverage measurement and whether to display the covered/uncovered.



[Branch coverage Column Info display] radio button allows you to select the information of the marker to be displayed in the left column of the editor.

- T/F marker option is shown in the source code editor.
- Covered Decision: T/F indication of a covered decision
- Uncovered Decision: T/F indication of an uncovered decision

The checkbox [Branch coverage, MC/DC measurement operator] sets the branch coverage measuring

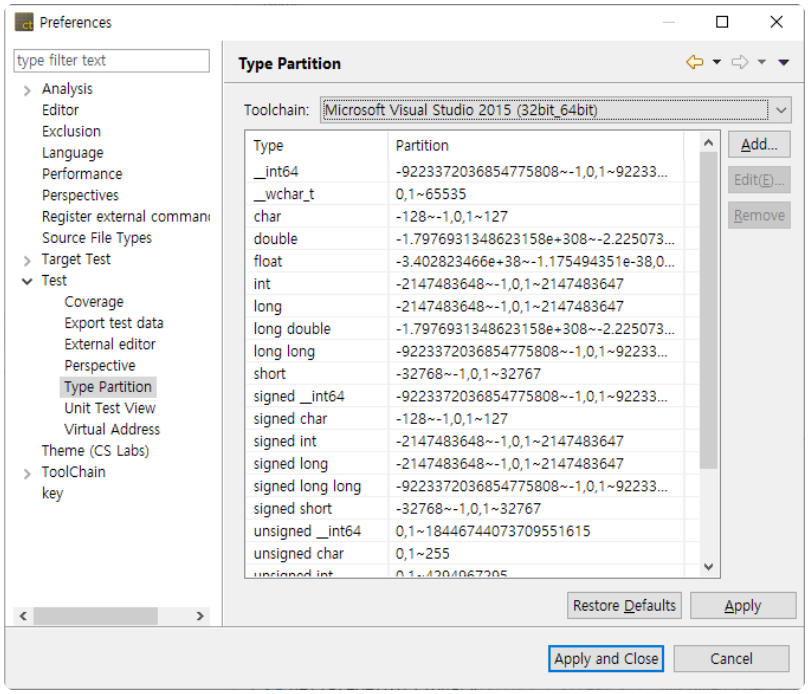
target. If it is re-analyzed and executed, the settings are applied.

- Conditional Operator(?:) expression: Measures the branch coverage for ternary operator expression.
- Boolean Operator(&,||,!,<,<=,>,>=,==,!=): Measures the branch coverage for boolean operator expression.
- If unchecked all, measures the branch coverage for ‘if’, ‘for’, ‘while’, ‘do-while’ and ‘switch’ statements only.

> ! When changing the ‘Conditional Operator(?:)’ setting, the COVER’s [Toolchain] > [Standard] must be set identically to share coverage with COVER. For details, please refer to the Import Coverage by Ternary Operator Settings page in User Guides.

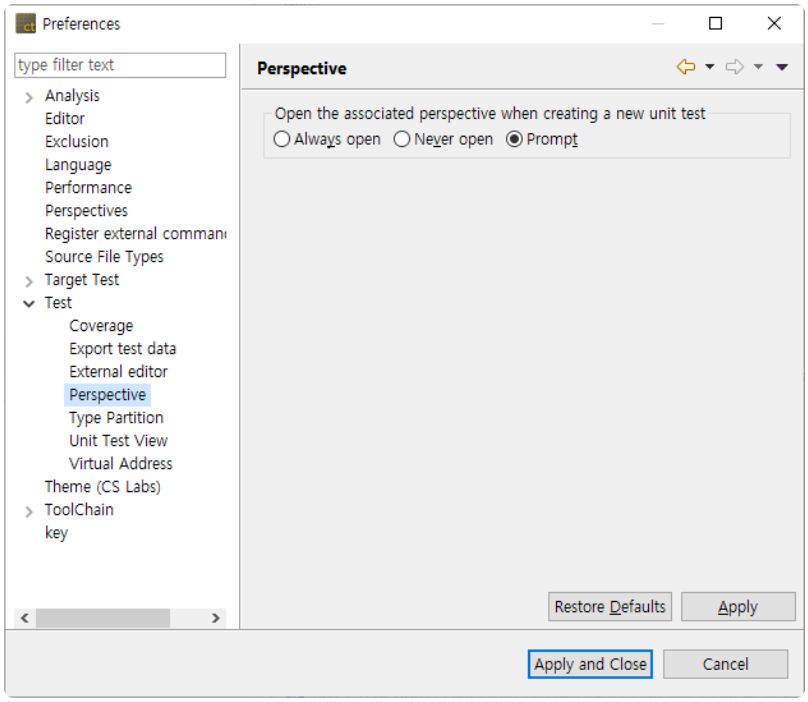
Type Partition

You can edit the basic type partition for each toolchain and restore it into the default value provided. In Toolchain combo box, select the toolchain to be modified, edit the partition and click [Apply].



Perspective

After the test has created, you can set the opening option for the perspective associated.
Select one among [Always open], [Never open] and [Prompt] and click [Apply].


Preferences
type filter text
Analysis
Editor
Exclusion
Language
Performance
Perspectives
Register external comman
Source File Types
Target Test
Test
Coverage
Export test data
External editor
Perspective
Type Partition
Unit Test View
Virtual Address
Theme (CS Labs)
ToolChain
key
Perspective
Open the associated perspective when creating a new unit test
Always open
Never open
Prompt
Restore Defaults
Apply
Apply and Close
Cancel

15.7. Toochain

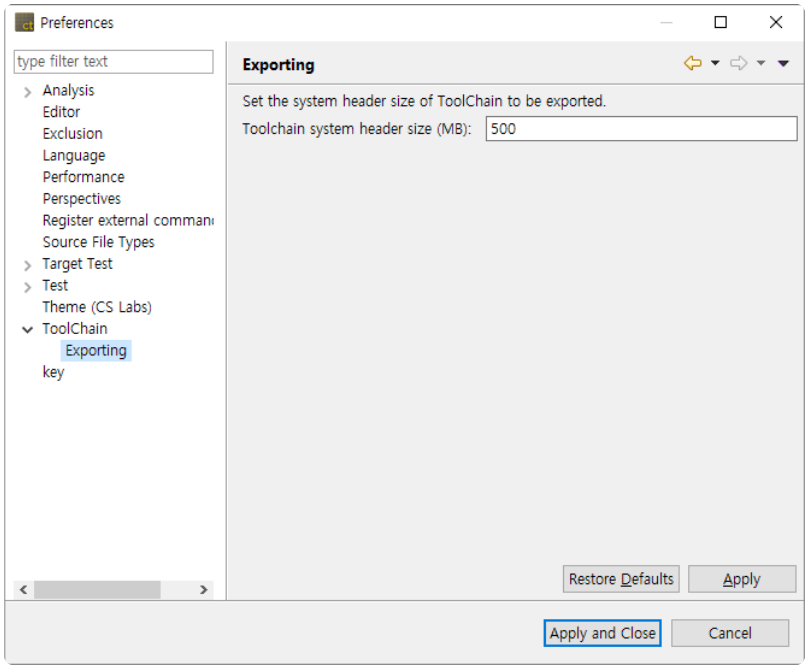
You can set the information for the toolchain to be used in the tool.

You must have a toolchain (compiler information) for the source to be tested in order to create or analyze the project.

The detail description of the toolchain settings can be found in the 'Set a Toolchain (Analyzer)'.

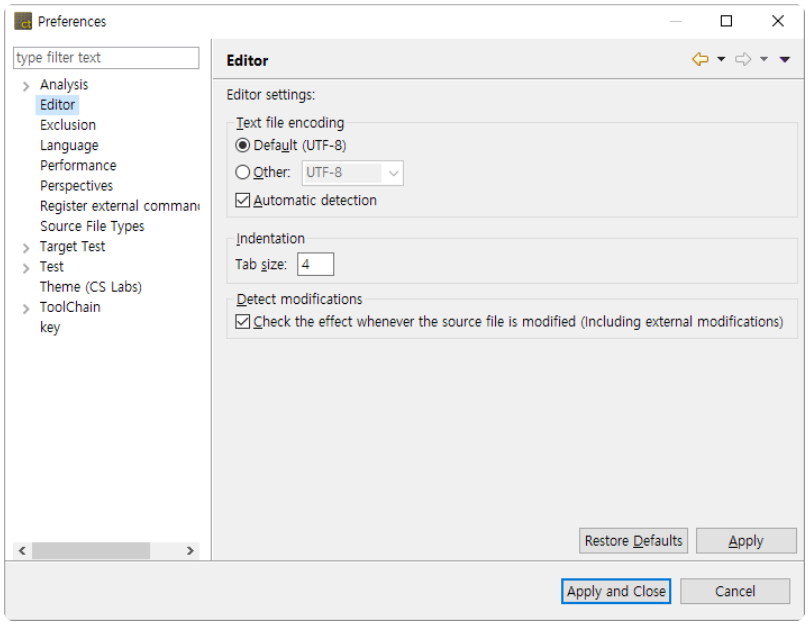
Export Toolchain

It sets the system header size of the toolchain to be exported. The system headers larger than the size you set cannot be exported.



15.8. Editor

You can change the settings related to the editor.



Text file encoding

You can set the encoding to be used when opening a text file in the editor. If the [Automatic detection] option is on, it will automatically detect the encoding of the text file. (If it cannot be detected automatically, it will open with the encoding you set.)

Indentation

If you change the tab size, the tab size shown in the editor is changed too.

Detect modifications

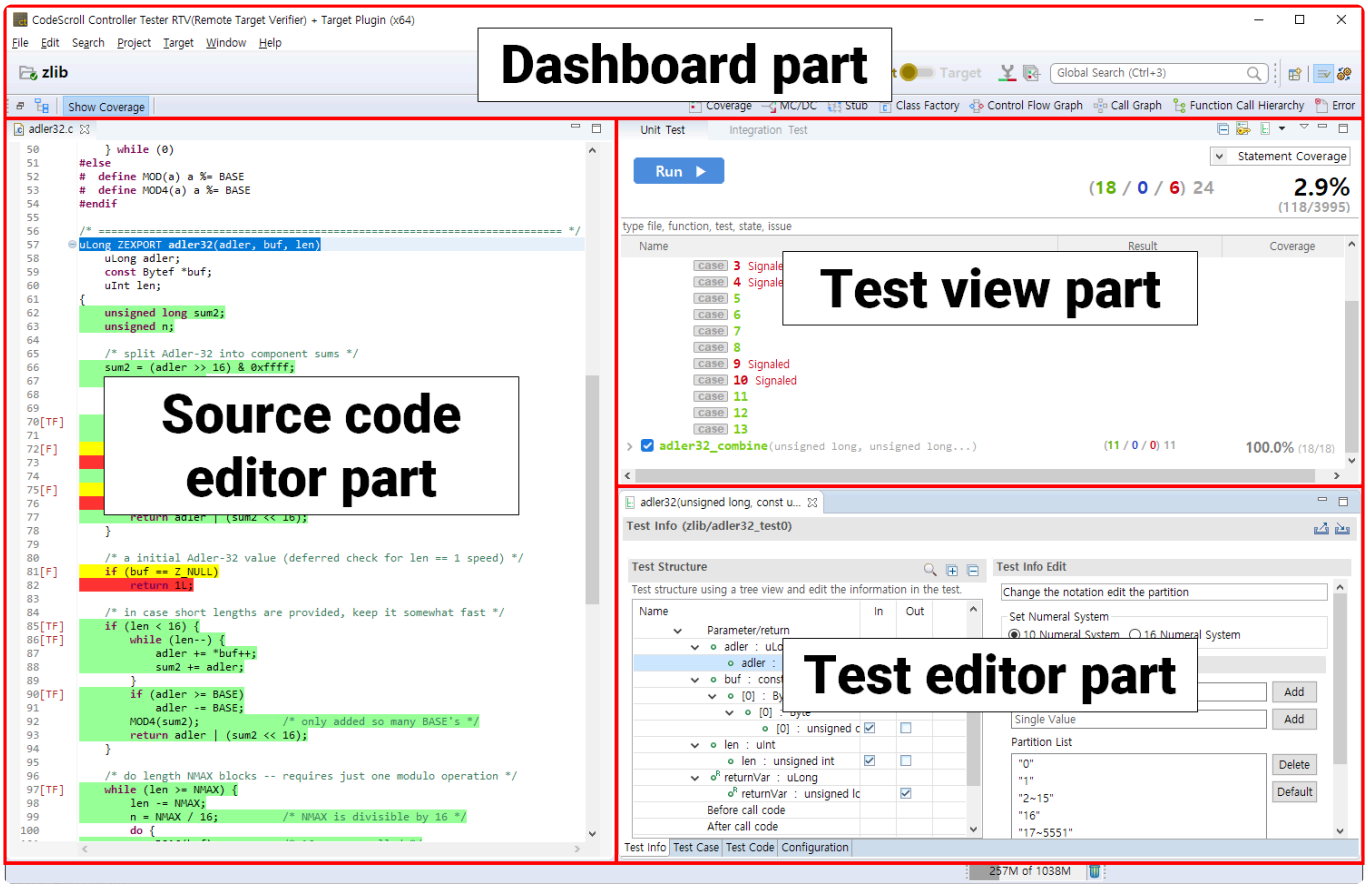
If the [Detect modifications] option is on, it checks the impact of the changes each time the source file is modified.

If the disc speed if slow, deselect this checkbox to improve the performance.

16. Test Perspective

The Test Perspective provides UI for displaying only the necessary information so as to keep the flow of the testing task and focus on the goal.

It provides a high DOF screen with Eclipse RCP. You can set it in the [Window] menu and in each view.



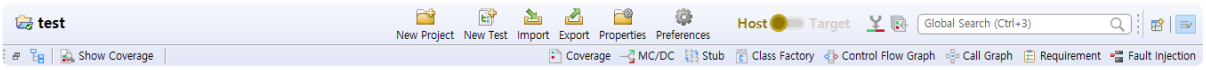
The components that make up the test perspective are:

- Dashboard
- Test Navigator View
- Unit Test View
- Integration Test View
- Coverage View
- MC/DC View
- Stub View
- Class Factory View
- Control Flow Graph View
- Call Graph View
- Error View
- Debug Information View
- Fault Injection View
- Input/output Data Graph View
- Console View
- Requirement View

Views that are not included by default can be opened from [Window]> [Show View]> [Other…] on the top menu.

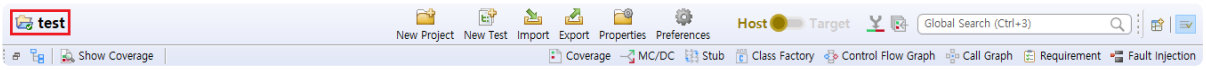
16.1. Dashboard

The dashboard section provides a summary of frequently used functions or tests that are currently in progress.



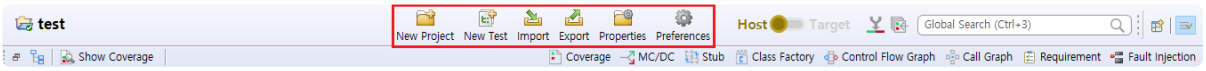
Project name

It shows the name of the project selected in the Test Navigator View.



Bookmark menu

You can use the [New Project], [New Test], [Import], [Export], [Properties] and [Preferences] features.

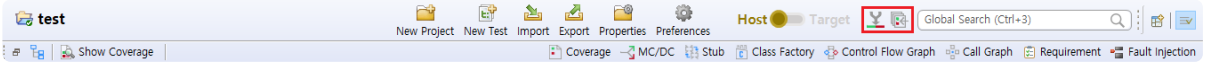


Host/Target setting menu

You can change the test environment to the host or target.

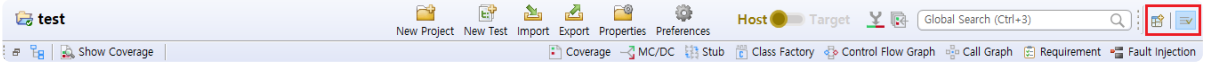
Coverage-related menu

You can use the Show the [Merger coverage between host and target] and the [Show full coverage (Include External Coverage)] features.



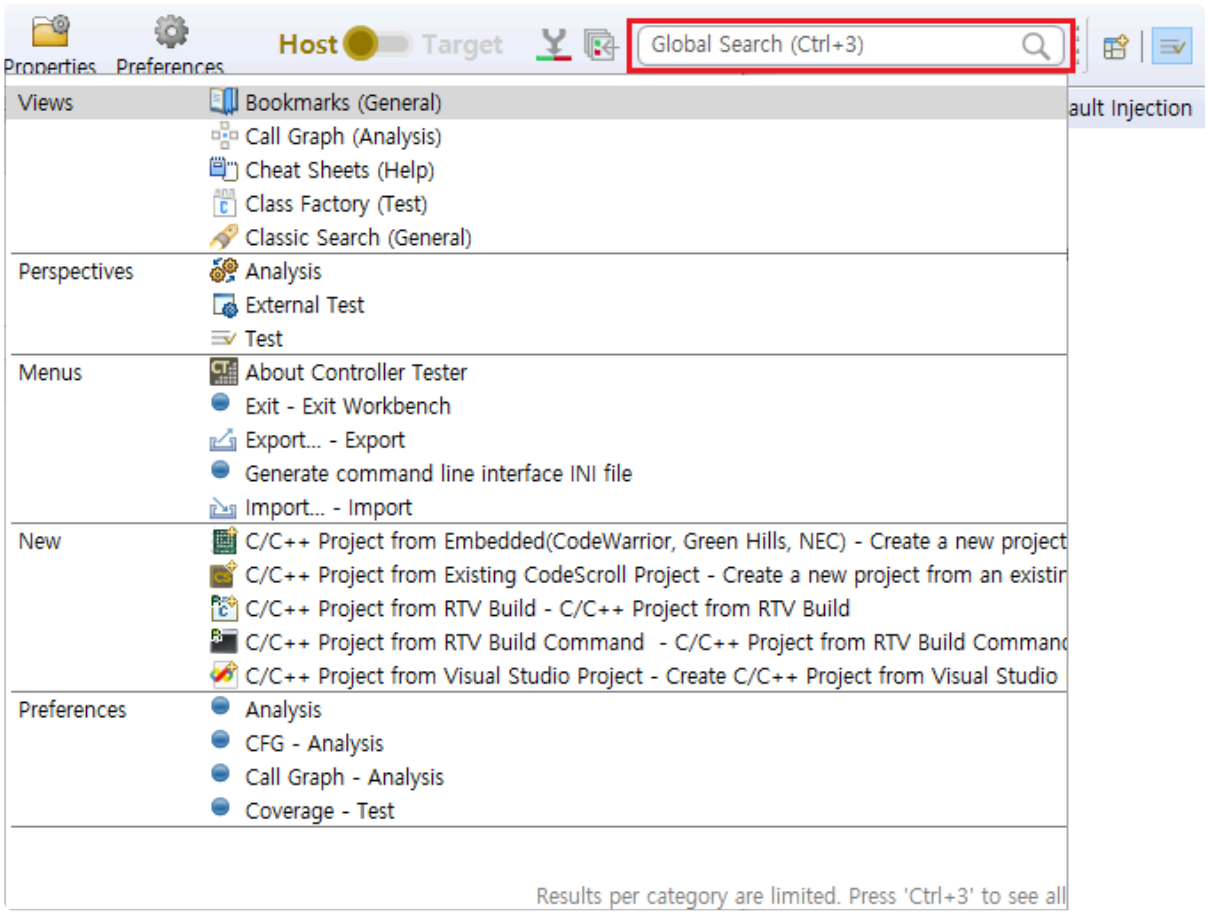
Open Perspective menu

You can select and open the perspective located in Controller Tester

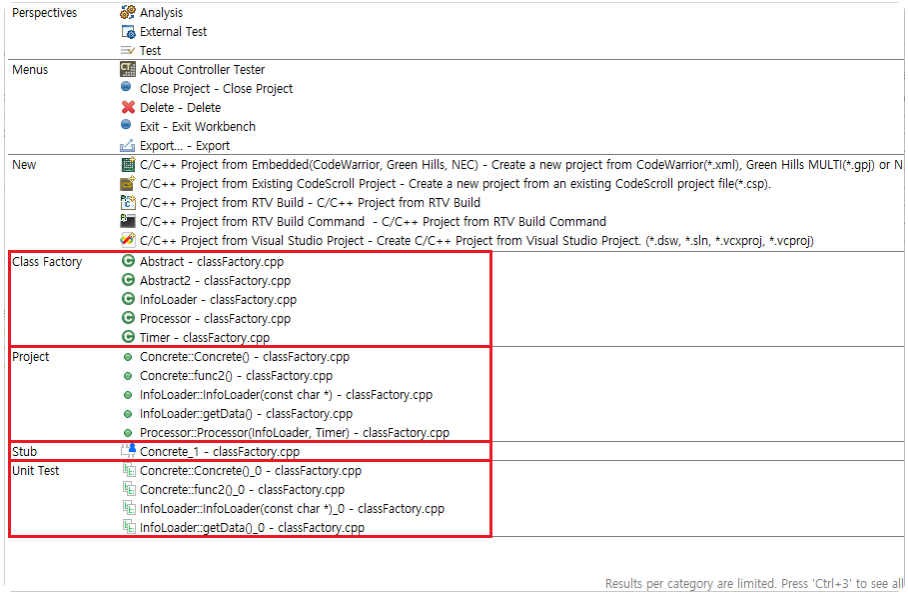


Global Search

The Global Search allows you to search New, Views, Editor, Menus, Properties, Preferences, Project resource (source file, function), Test, Stub and Class Factory, etc. by keywords. Select the Global Search window or press the shortcut key (`Ctrl` + `3`) to use the Global Search feature.

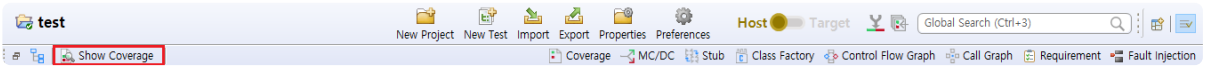


> The project resource (source file, function, etc.), test, stub and class factory can be searched by selecting the analyzed project.



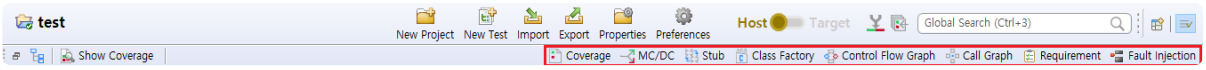
Show Coverage

It allows you to set the covered area in the source code editor is marked or not.



Open/Close the test-related view menu

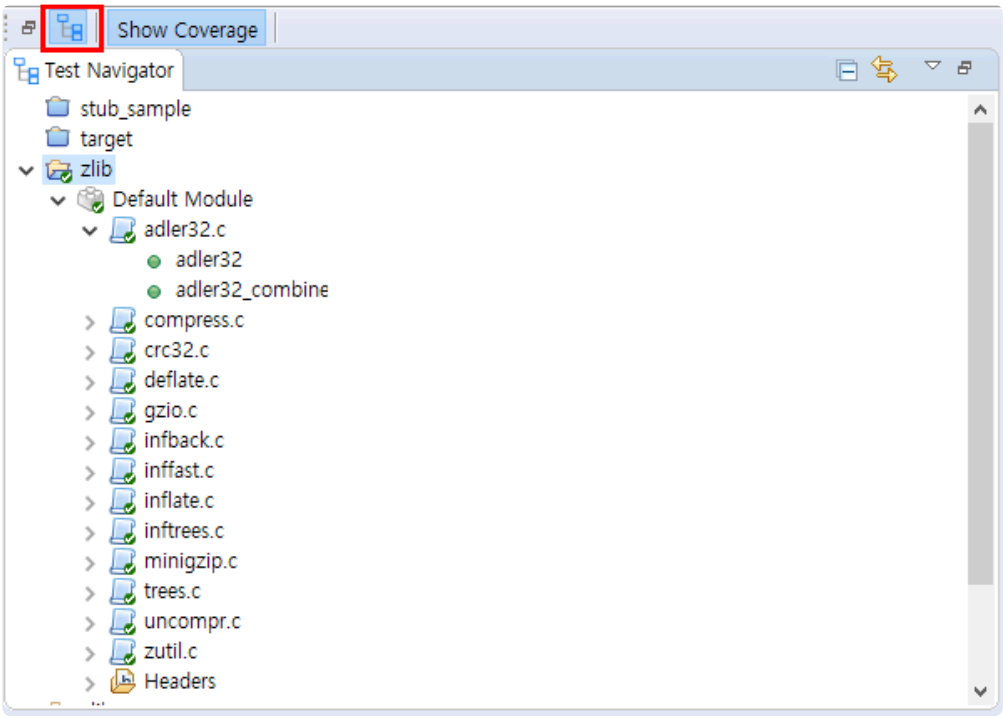
You can open and close the test-related views.



16.2. Test Navigator View

The Test Navigator View shows the hierarchy structure of the projects and the test models under the projects included in the workspace.

The Test Navigator View does not occupy any space but remains the minimized state. If you select [Test Navigator] in the dashboard, the Test Navigator View is located above the Source Code Editor.



Icon

Icon	Description(*: created after analysis)
	Open project
	Closed project
	Module
	Source file
	*Global variable
	*Function
	*Header folder
	*Header file
	*System header file

Toolbar menu

Menu	Description
Link with Selection	Highlights the items selected in the other view.
Collapse All	Hides all tree nodes.

Icon overlay

Icon overlay	Description
	Analyzed
	Changed after analysis

Copy the source file into the other module

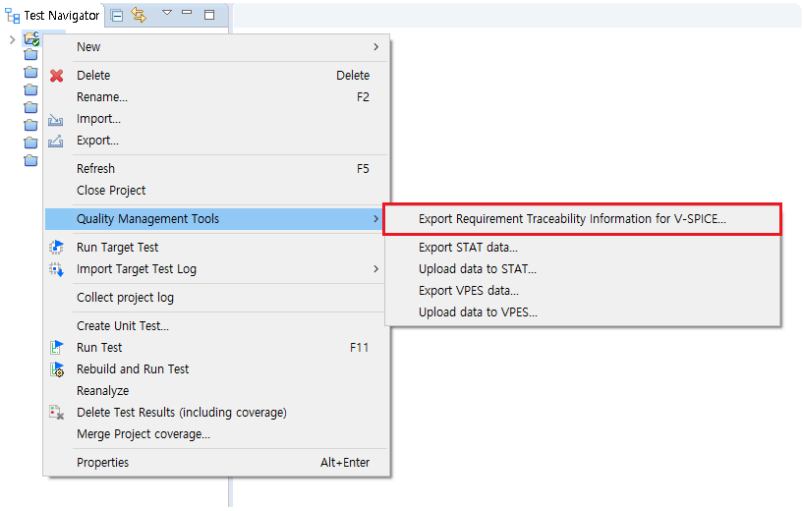
To copy the source file into the other module, with `Ctrl` key pressed, drag and drop the source file to be copied into the target module.
Alternatively, right-click the source file to be copied, select [Copy], right-click the target module and select [Paste]. (Shortcut keys: `Ctrl` + `C` and `Ctrl` + `V`)

Move the source file to the other module

To move the source file to the other module, drag and drop the source file to be moved into the target module.

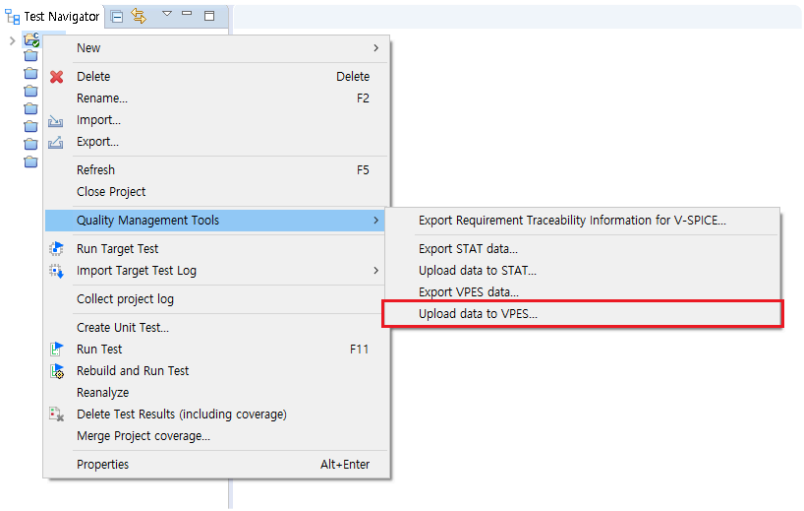
Export Requirement Traceability Information for V-SPICE

Connection information between requirements and tests can be exported to an xml file that conforms to the V-SPICE format.



Select a project and right-click on the menu to select [Quality Management Tools] > [Export Requirements Traceability Information for V-SPICE…] function.

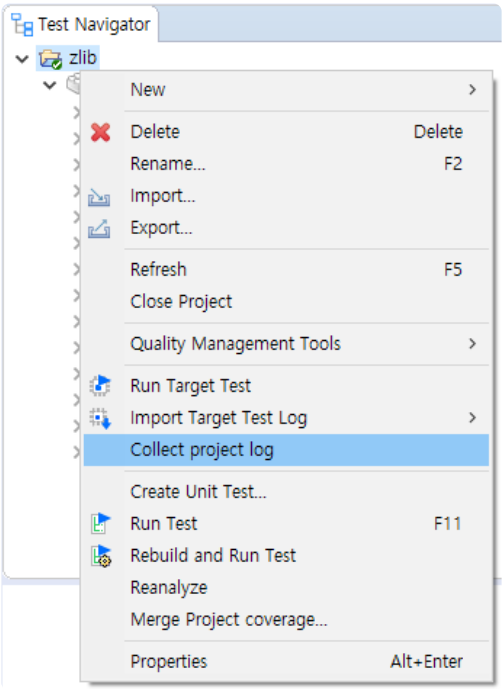
Upload data to VPES



When you select and right-click a project, [Upload data to VPES…] is indicated in the menu. For this feature, please contact Technical support.

Collect Project Log

This feature collects the logs left in use of Controller Tester on a project basis in batch and exports them. When you contact Technical support due to an error occurred, please send us the collected compressed file so that we can solve the problem more quickly.



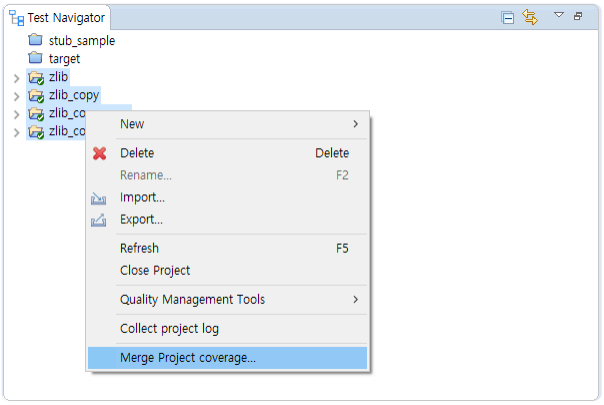
When you select and right-click a project, [Collect project log] item is indicated.

16.2.1. Merge Project Coverages

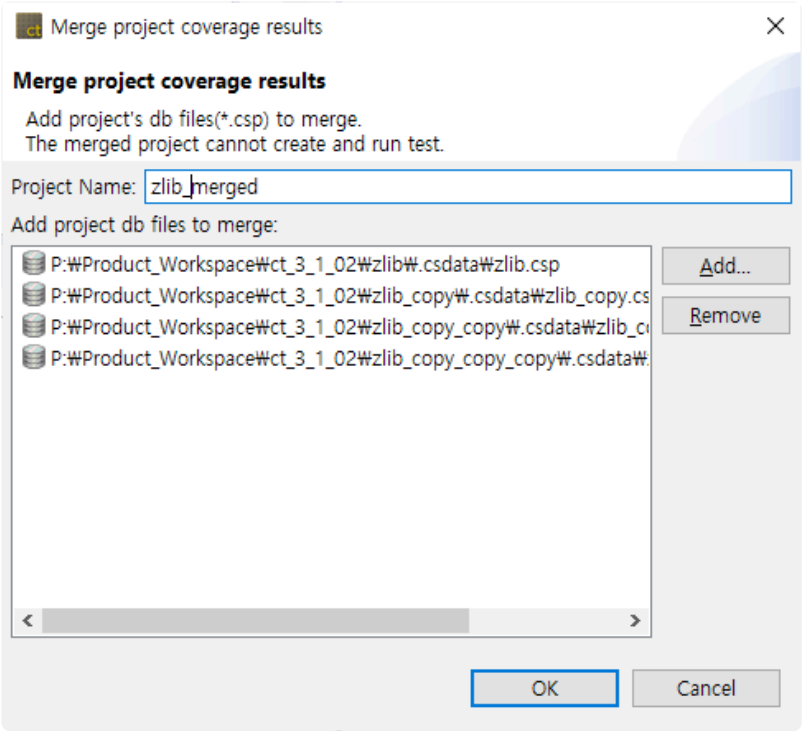
If a project size is too large to run the test at once, you can run the test by dividing it into multiple projects. [Merge Project Coverage] function runs the test by dividing one project into multiple projects and then shows the overall coverage results by merging the result of the test for these projects.

Merge Project Coverage

1. After selecting all the projects that you want to merge the coverage results, right-click it and select [Merge Project Coverage].

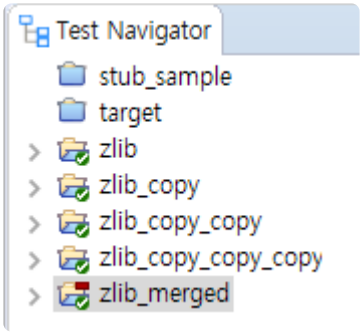


2. Select [Merge Project Coverage] menu to open for creating the project name. Enter the project name to be created in [Project Name].

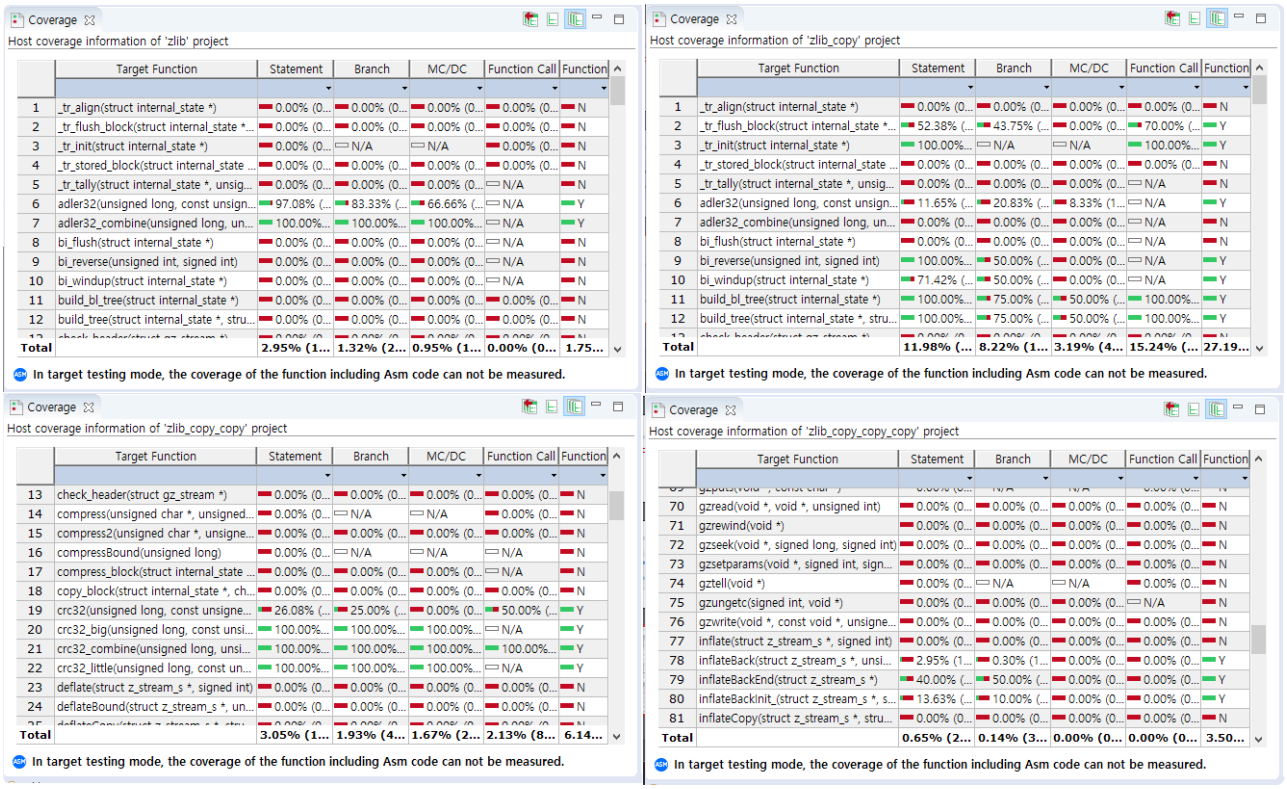


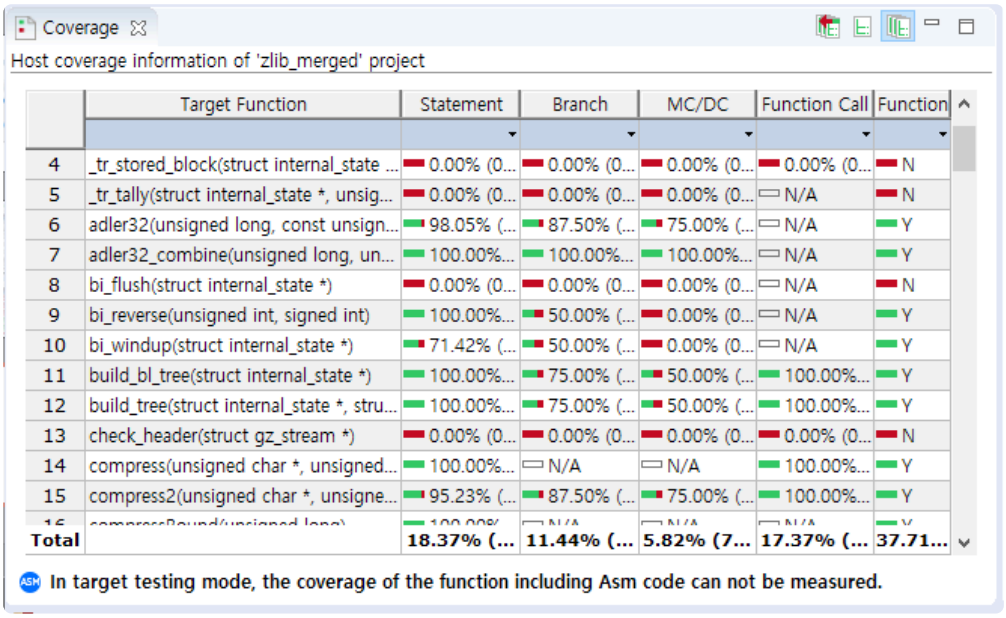
   - By clicking the [Add] button, you can perform a merge coverage function for projects not included in the workspace.

3. Click [OK] to check that a new project that the project results had been merged has created in the Test Navigator view. The merged project is displayed with a red mark on the upper right corner.



4. Select the project that the results had been merged and check the coverage view to see the overall coverages.

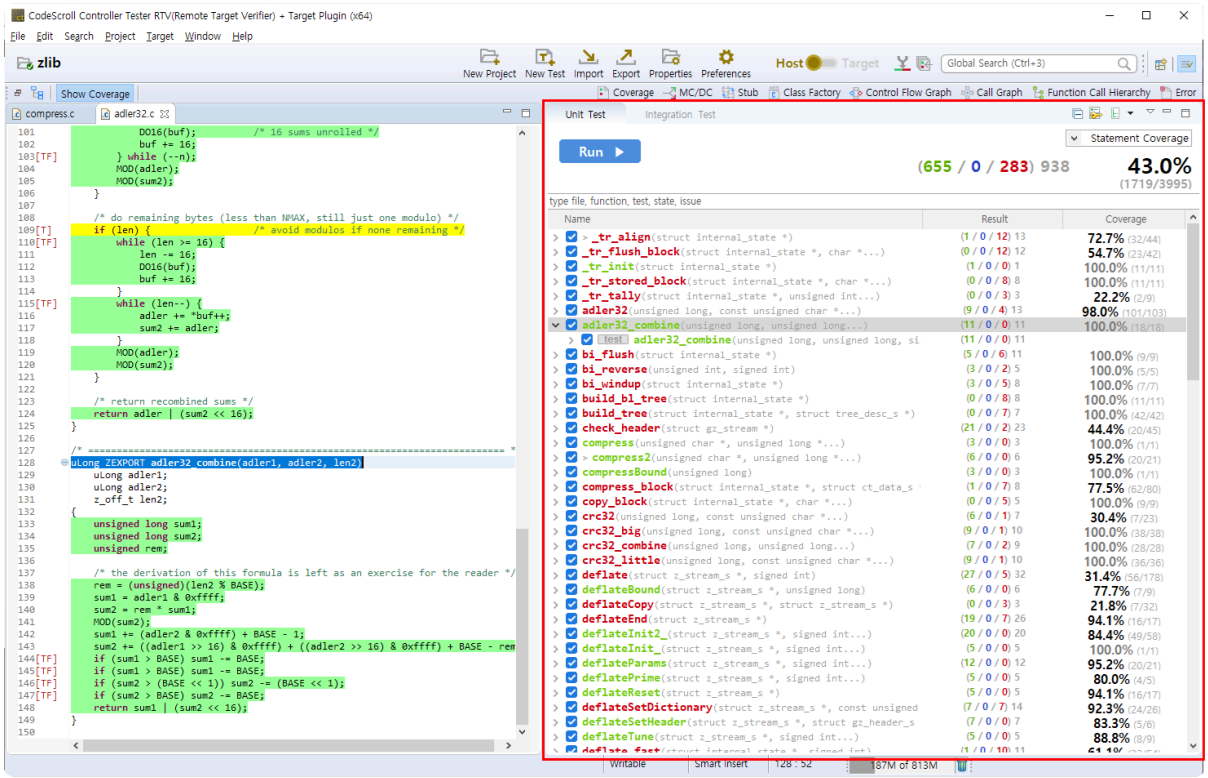




! The [Merge Project Coverage] function may not work properly when two or more different functions in the project to be merged have the same function name.

16.3. Unit Test View

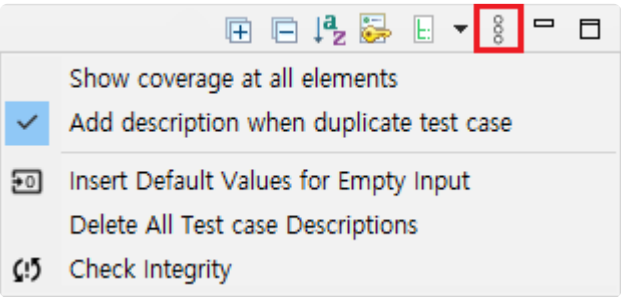
The Unit Test View section is located on the right of the screen. The Unit Test View combines the test and coverage information and provides it in a single view.



Toolbar menu in the Unit Test View

Toolbar icon	Description
Expand All	Shows all test view tree.
Collapse All	Hides all test view trees.
Sort by Unique Test Name	Sort by unique test name.
Show as Unique Test Name	Displays by unique test name.
Total	Displays all the tests.
Failure/Error	Displays only the tests that the test result is Failure/Error.
Failure	Displays only the tests that the test result is Failure.
Error	Displays only the tests that the test result is Error.
Success	Displays only the tests that the test result is Success.
Function changed	Displays only the tests that the function to be tested has been changed.
Run not guaranteed	Displays only the tests that the test execution is not ensured.
Host/Target result different	Displays only the test that the host result and the target result are different.

Pull-down menu in the Unit Test View



Menu name	Description
Show coverage at all elements	Displays the coverage in all items. (function, test, test case)
Add description when duplicate test case	Adds original test case number on duplicated test case.
Insert Default Values for Empty Input	When input data are empty, inserts default values.
Delete All Test case Descriptions	Delete test case descriptions all together.
Check Integrity	Checks integrity about functions and stubs. Then, reset test if it needs.

- Insert Default Values for Empty Input

Variable type	Default value
string	*(empty string)*
without string	0

For [Test reconfiguration], refer to [Source Code Modification and Test Reconfiguration]

Dashboard in the Unit Test View

Menu	Description
Run	Runs unit tests.
(878 / 0 / 359) 1237	Shows success, failure, error and total of test case.

Statement Coverage	Selects the coverage type to be displayed in the Unit Test View. (statement, branch, MC/DC, function call)
43.7% (1952/4466)	Shows the selected kind of full coverage.

Search

In the Unit Test View, you can search functions, tests and test cases by file, function, test, status (success, failure, error) and issue name.

Status search keyword

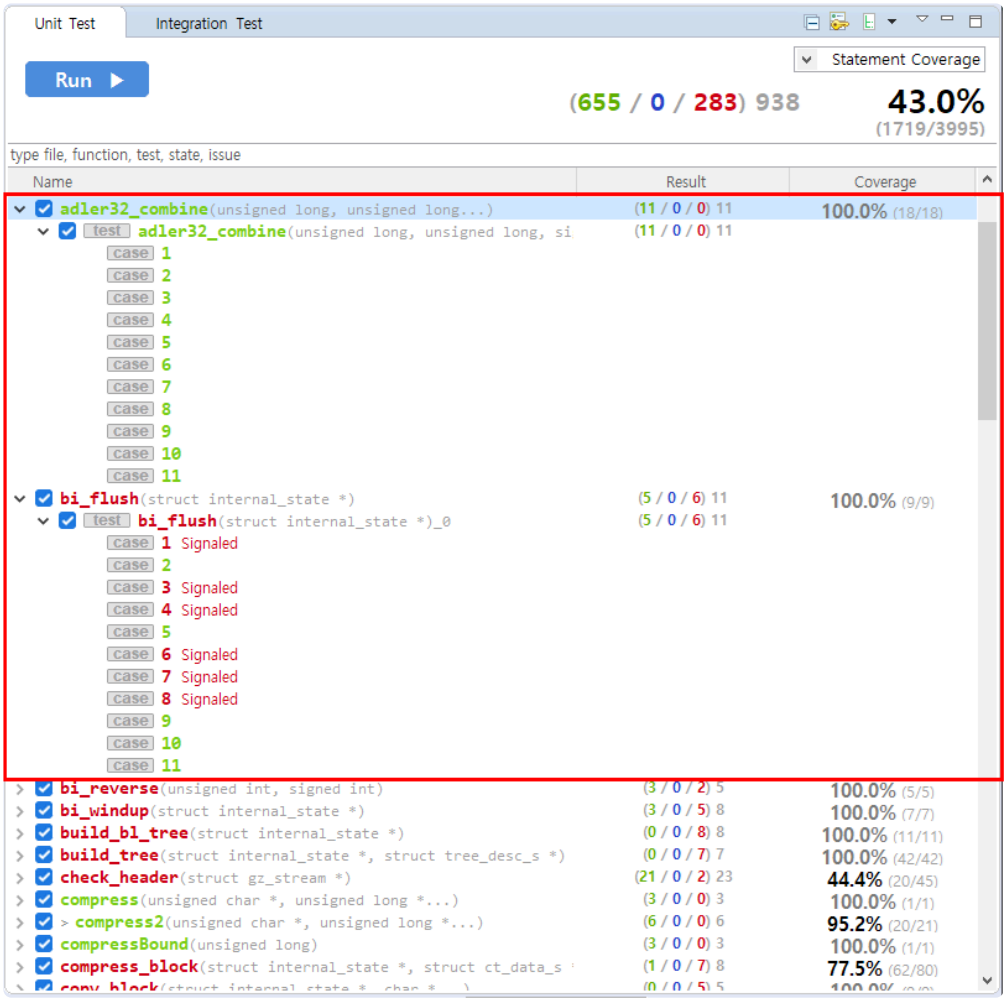
In the Unit Test View, you can filter the tests that are displayed by the status search keyword. It provides searchable keyword recommendation and auto-complete function.

Keyword	Description
%TEST_SUCCESS%	Test success
%TEST_FAILURE% %TEST_ERROR%	Test failure/error
%TEST_FAILURE%	Test failure
%TEST_ERROR%	Test error
%TEST_HAS_NOT_FUNCTION%	Function changed
%TEST_NOT_GUARANTEE%	Run not guaranteed
%TEST_RESULT_DIFFERENT%	Host/Target result different

‘Run not guaranteed’ is a case where test code is generated, but there is no guarantee that it will run normally. This is the case when using a type that is difficult to specify as a parameter or return value.

Structure of the Unit Test View

The Unit Test View presents the hierarchy structure of [Function] > [Test] > [Test case]. If there are more than 100 test cases in a test, they are grouped in groups of 100 to represent a group.



Item icons in the Unit Test View

Item icon	Description
None	Function
test	Test
case	Test case, Test case group

Item status color in the Unit Test View

The execution information of functions, tests, and test cases in the Unit Test View is represented by colors.

Color	Description
Green	Function/Test: If the execution result of all the test cases under it is success. Test case: The result of execution is success.
Blue	Function/Test: If only test cases with failure and no test cases with errors exist under it. Test case: The result of execution is failure.
Red	Function/Test: If test cases with error exist under it. Test case: The result of execution is error.

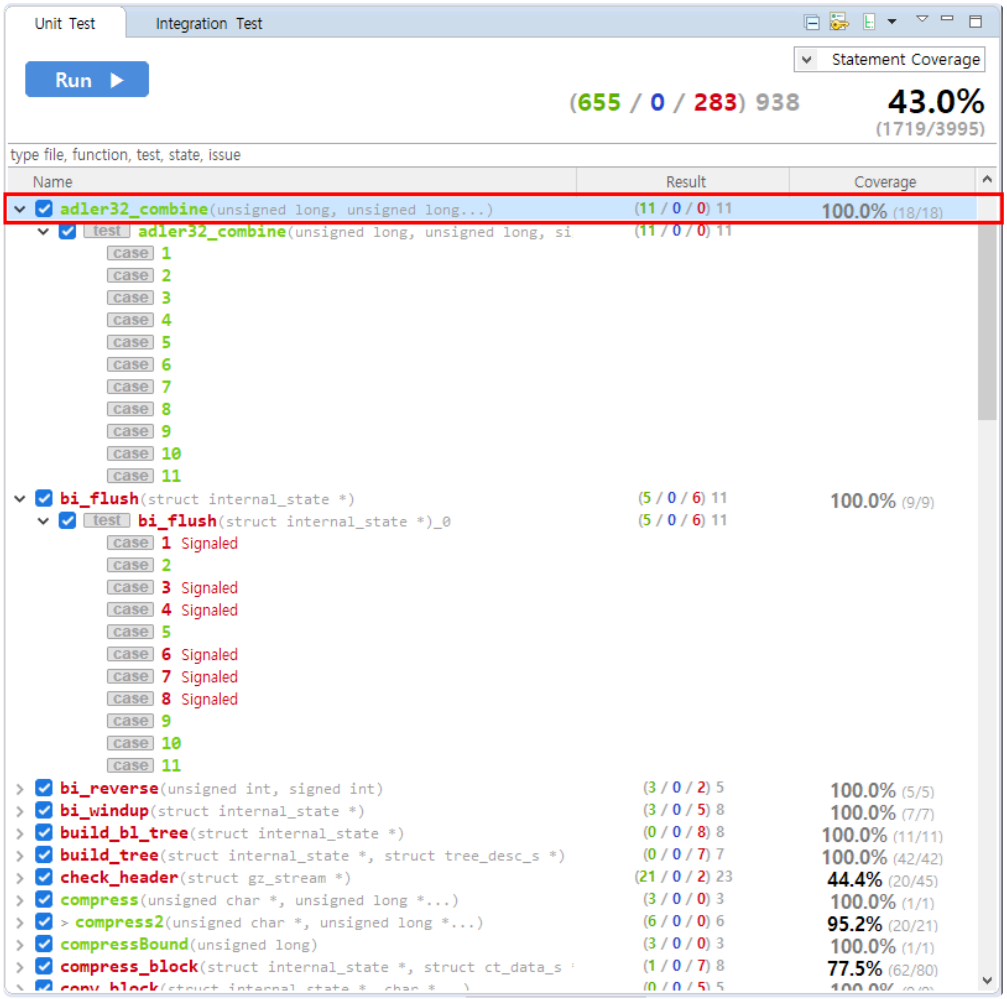
Orange	Function/Test: All test cases have not been executed under it and test cases that are not guaranteed to run exist. Test case: Run not guaranteed.
Black	Function/Test: All test cases under it have not been executed. Test case: Not been executed.

Function

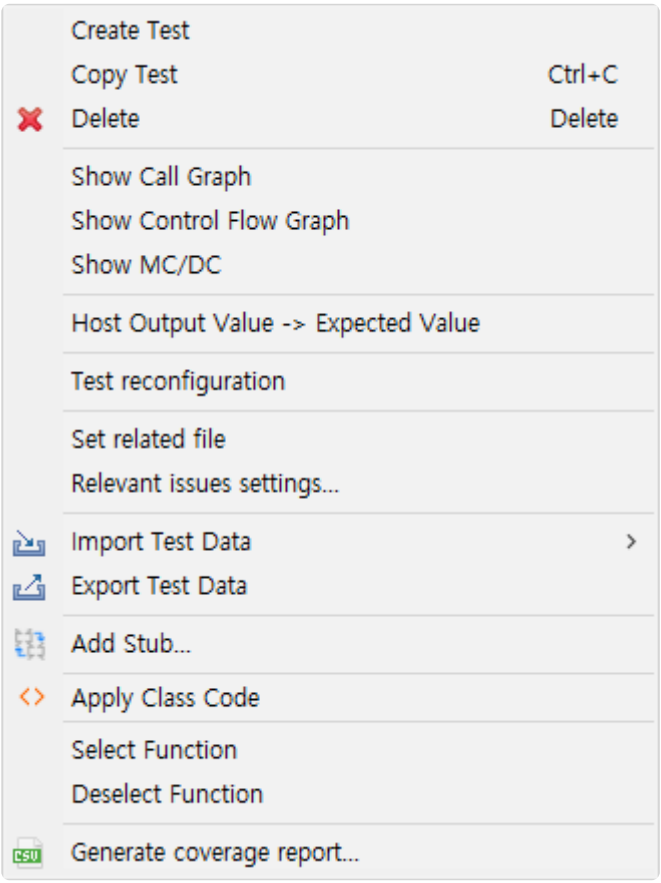
The functions provide the function name, the coverage and the test case execution results (success, failure, error and total).

> For the coverage for each function, the test execution result of the other function may be applied depending on the call relationship between functions.

Double-click on a function to check the location of the function in the Source Code Editor.



Context menu of the function



Context menu	Description
Create Test	Creates a test of the selected function.
Copy Test	Copies tests of the selected function.
Delete	Deletes tests and test cases of the selected function.
Show call graph	Shows the function call graph of the selected function.
Show Control Flow Graph	Shows the control flow graph of the selected function.
Show MC/DC	Shows MC/DC of the selected function.
Host Output Value -> Expected Value	Pastes the host output value to the expected value.
Test reconfiguration	Reconfigure test information.
Set related file	Selects the files containing that test.
Relevant issues settings	Associates the selected test with the issue of the management tool.
Import Test Data	Imports the test data saved in local.
Export Test Data	Exports the test data to local.
Add Stub	Adds the stub to the selected test.
Apply Class Code	Apply the class code generated in the Class Factory View to the

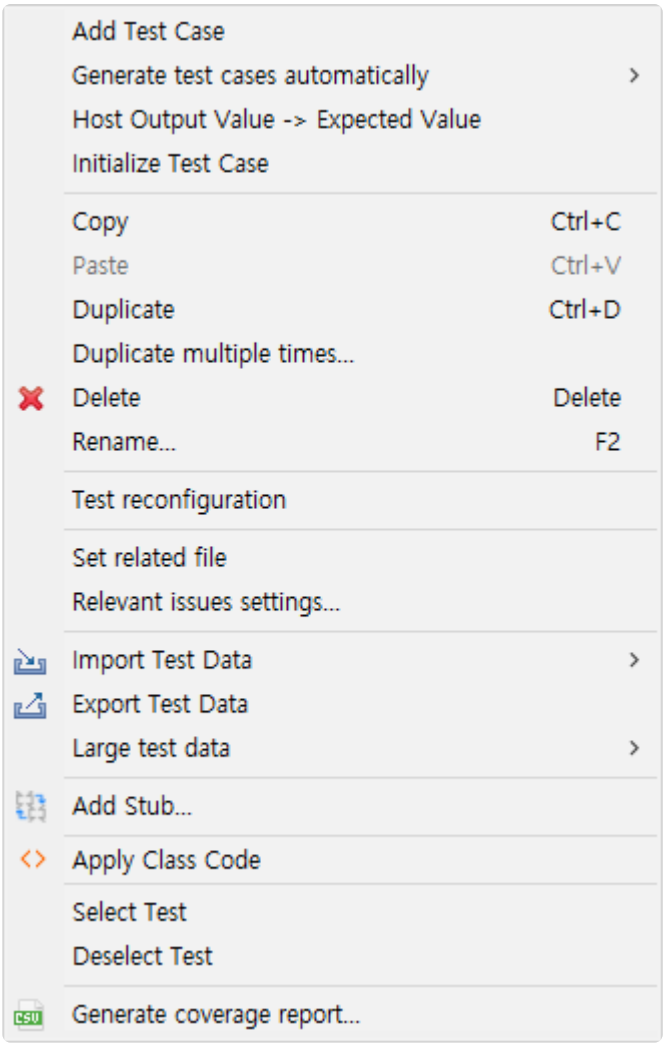
	function.
Select Function	Selects the checkbox of all the functions selected by a mouse.
Deselect Function	Deselects the checkbox of all functions selected by a mouse.
Generate coverage report	Exports the selected test coverage to the selected path.

For [Test reconfiguration], refer to [Source Code Modification and Test Reconfiguration]

Test

The test provides coverage, test case execution results(success, failure, error, total).
Double-click on a test to open its Test Editor.

Context menu of the test



Context menu	Description
Add Test Case	Adds the test case to the unit test selected.
Generate test cases	Creates the test case in various ways.

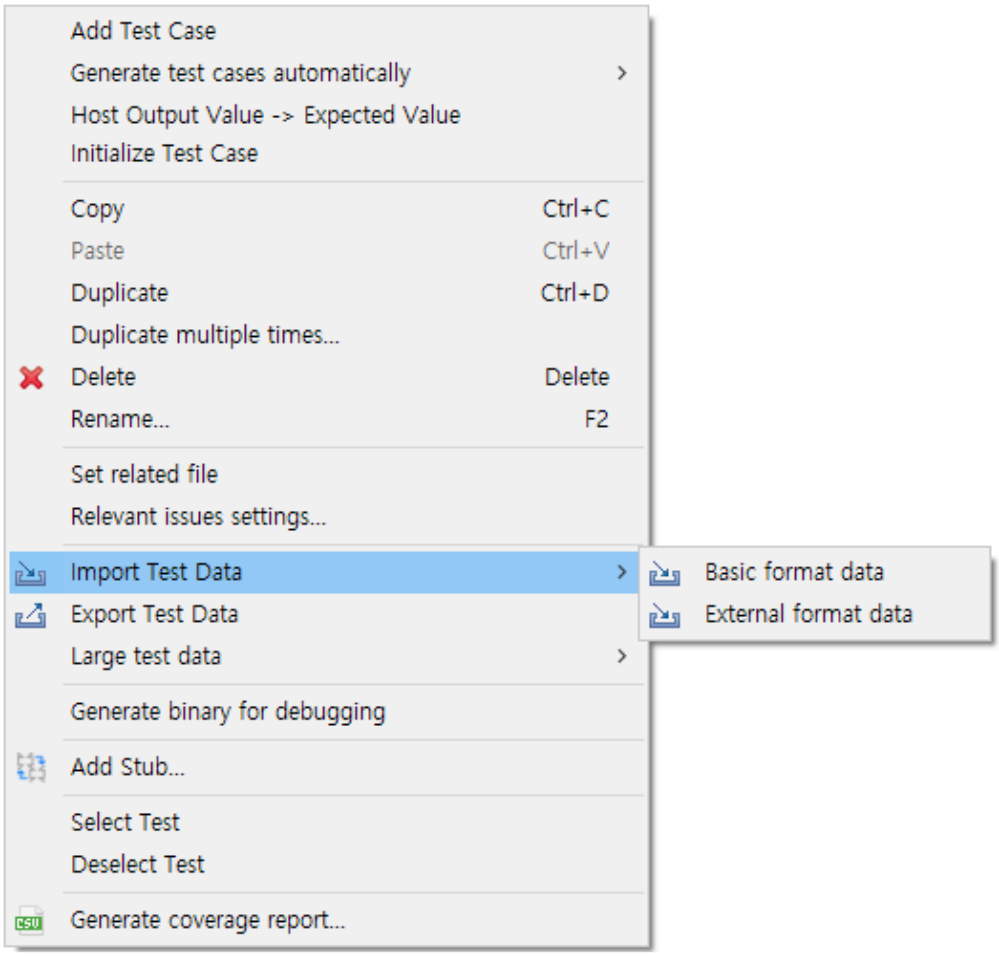
automatically	
Host Output Value -> Expected Value	Pastes the host output value to the expected value.
Initialize Test case	Delete all the test cases.
Copy	Copies the test and test case.
Paste	Pastes the test and test case.
Duplicate	Duplicates the test and test case.
Duplicate multiple times	Duplicates tests and test cases as many as the number entered.
Delete	Deletes the test and test case.
Rename	Modifies the test name.
Test reconfiguration	Reconfigure test information.
Set related file	Selects the files containing that test.
Relevant issues settings	Associates the selected test with the issue of the management tool.
Import Test Data	Imports the test data saved in local.
Export Test Data	Exports the test data in local.
Large test data	Exports it locally or registers the file written by the user to the target test as a test case.
Add Stub	Adds the stub to the selected test.
Apply Class Code	Apply the class code generated in the Class Factory View to the test.
Select Test	Checks all checkbox of the tests selected.
Deselect Test	Checks all checkbox of the tests selected.
Generate coverage report	Exports the selected test coverage to the selected path.

- Rule for importing test data
  When importing the test data, if you select multiple test data numbered with the test name, it imports those files by merging them.

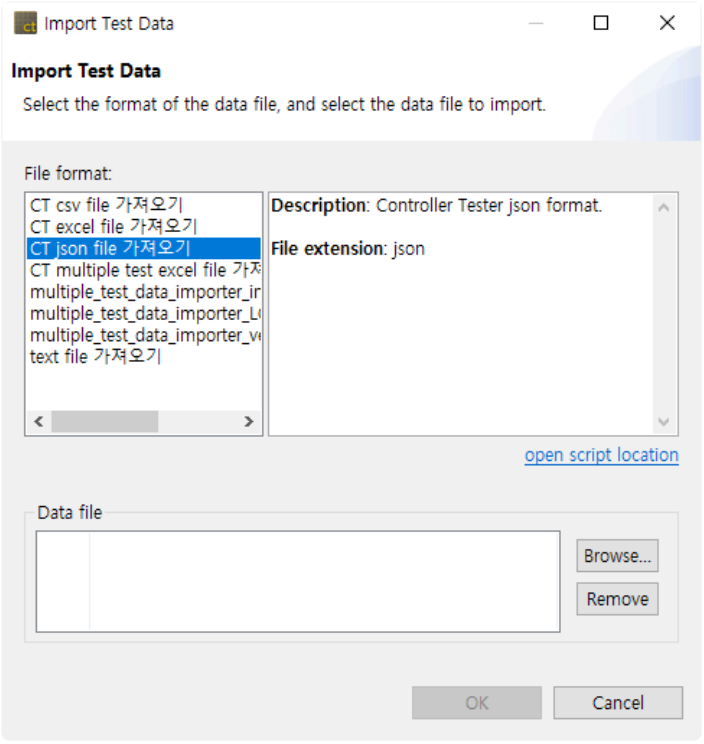
For [Test reconfiguration], refer to [Source Code Modification and Test Reconfiguration]

Import test data

You can import the test data in various formats (csv, xlsx, txt, json).

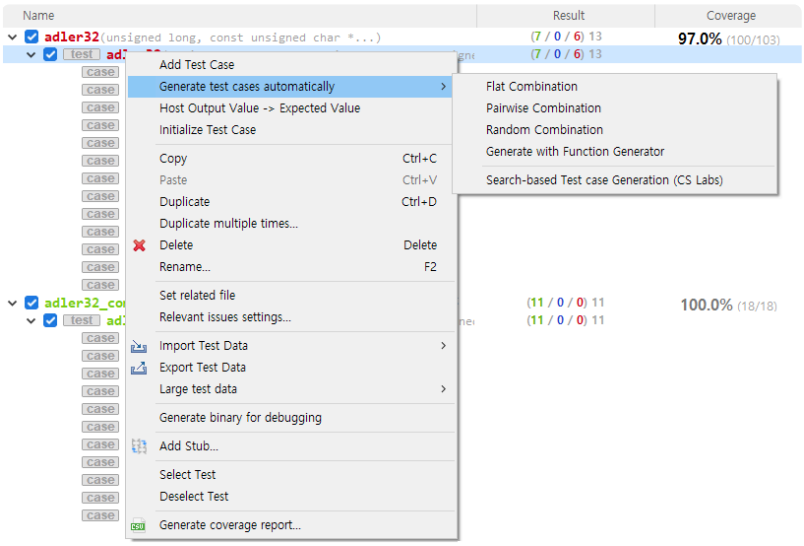


1. In the context menu of the Unit Test View, click [Import Test Data] and select either [Basic format data] or [External format data].
2. If you select [Basic format data], the test data with the format (csv) exported from the Controller Tester is imported.
3. If you select [External format data], the test data is imported from the files in various formats (csv, xlsx, txt, json).



Generate test cases automatically

It creates a test case by using [Flat/Pairwise/Random combination], [Generates with Function Generator], and [Search-based Test case Generation (CS Labs)].

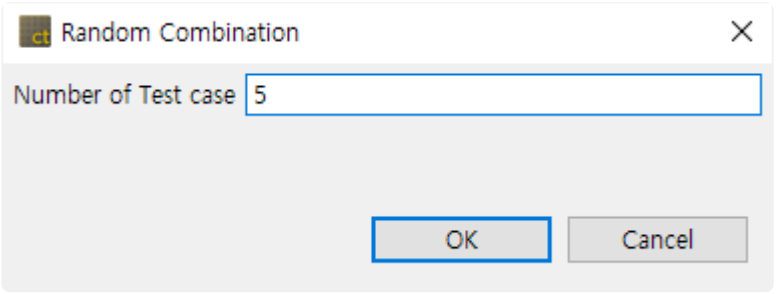


1. In the context menu of the Unit or the Integration Test View, click [Generate test cases automatically] and select one among [Flat Combination], [Pairwise Combination], [Random Combination], [Generate with Function Generator] and [Search-based Test case Generation (CS Labs)].

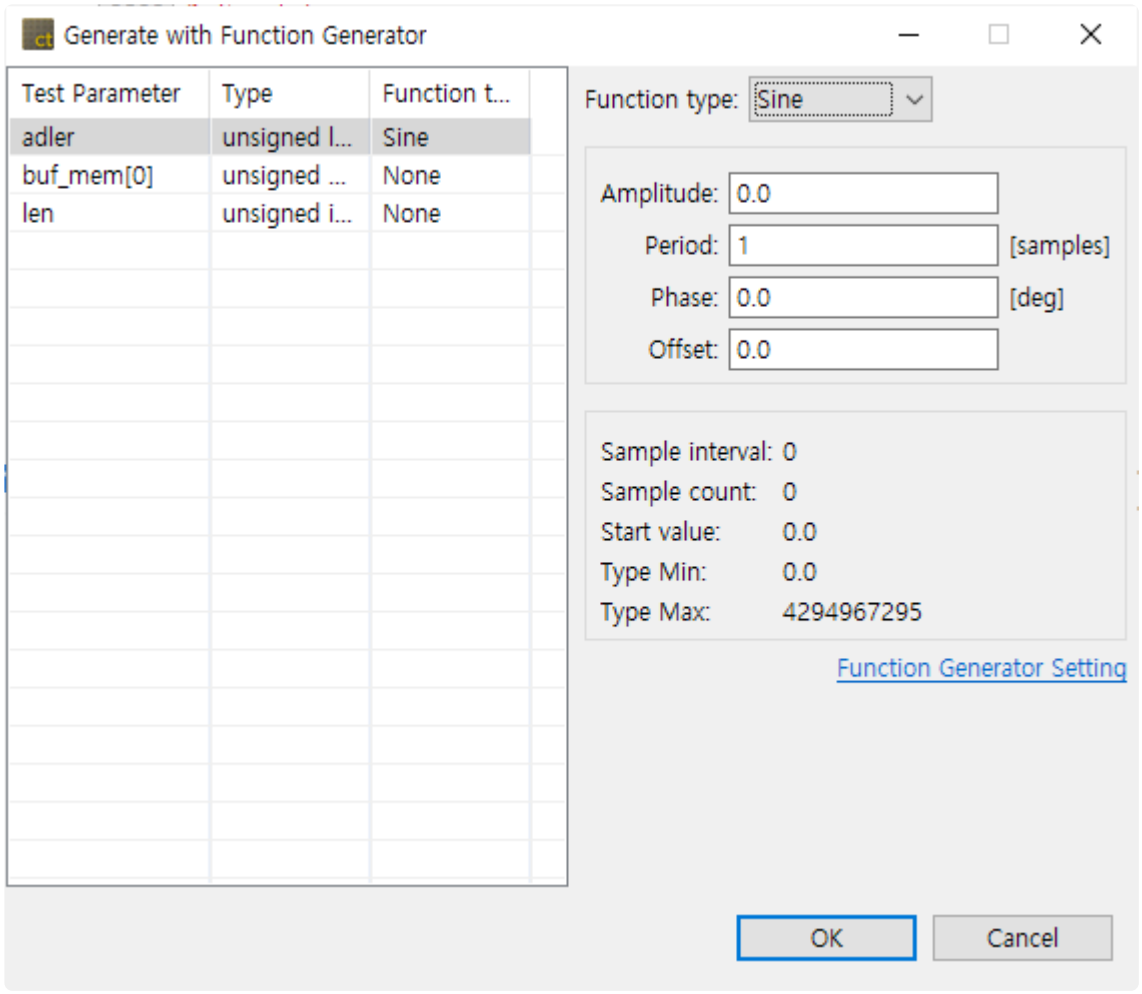
Method	Description
Flat	Combines simply based on variables having the largest number of test data.
Pairwise	Combines so that each selected parameter data is paired at least once with the

	parameter data other than itself. ※ The number of variable partitions must be more than 2 and less than 52.
Random	Combines the test data as many as the number of test cases that the user defines any value between the minimum value and the maximum value for the variable partition of input parameters.
Generate with Function Generator	Creates a test case by using six function types. (Ramps, Random, Range, Sine, Toggle, and Single Value)
Search-based Test case Generation (CS Labs)	Creates a test case by using AVM, which is a local search algorithm. ※ Currently, only C language is supported.

2. If you select [Random combination], the Random combination window is displayed so that the user can enter the number of test cases.



3. If you select [Generate with Function generator], you can select the function type (Ramps, Sine, Random, Toggle, Range, Single value and None) and change the settings of the function selected via the settings of the function generator. In the properties window of the project, you can set the function generator information in [Test] > [Generate test cases automatically] > [Function Generator].



a. Common settings

You can change the value in [Project] > [Properties] with the setting value common to each function.

Setting	Description
Sample interval	The interval of samples to be sampled from the function.
Number of sample	The number of samples to be sampled from the function(The number of test cases).
Start value	The default value at which the function is started (the values are created based on the start value).
Minimum value of type	The minimum value of variable partition. (If the return value of the function is less than the minimum value of type, the minimum value of type is returned.)
Maximum value of type	The maximum value of variable partition. (If the return value of the function is greater than the maximum value of type, the maximum value of type is returned.)

b. Ramps function

It is a function that creates a pulse by using Pre, Post and Hold values. If the number of samples is greater than the period of the function, the function is called recursively.

Setting	Description
Pre delay	The time to last Pre/Post sample value.
Rise samples	The time to rise from Pre/Post sample value to Hold sample value.
Hold samples	The time to last Hold sample value.
Fall samples	The time to fall from Hold sample value to Pre/Post sample value.
Post delay	The time to last Pre/Post sample value.
Pre/Post delay value	Pre/Post delay sample value.
Hold value	Hold sample value.

c. Random function

It is a function that creates a random value between the minimum value (Min) and the maximum value (Max).

Setting	Description
Min	The minimum value of random range.
Max	The maximum value of random range.

d. Range function

It is a function that creates values that increments or decrements by a certain interval (Step Size) between the minimum value of type and the maximum value of type.

Setting	Description
Step size	The size of increment or decrement value.
Hold	The number of times that the Step size holds.
Rising	The function type that rises by the Step size. Ex) if the step size is 30, 0,30,60……
Falling	The function type that falls by the Step size. Ex) if the step size is 30, 100,70,40……
Alternate	If the result value of function meets the minimum/maximum value of type, the function type that changes to rise or fall. Ex) if the step size is 30 and the maximum/minimum value are 0~100, 0,30,60,90,60,30,0……

e. Sine function

It is a function that creates Sine value. If the number of samples is greater than the period of the function, the function is called recursively.

Setting	Description
Amplitude	The amplitude of Sine function.
Period	The period of Sine function.
Phase	The phase of Sine function.
Offset	The offset of Sine function.

f. Toggle function

It is a function that creates FirstValue and SecondValue repeatedly.

Setting	Description
First value	The first value that is repeated in Toggle function.
Second value	The second value that is repeated in Toggle function.

g. SingleValue function

It is a function that returns only a constant single value.

Setting	Description
Value	The value to be created.

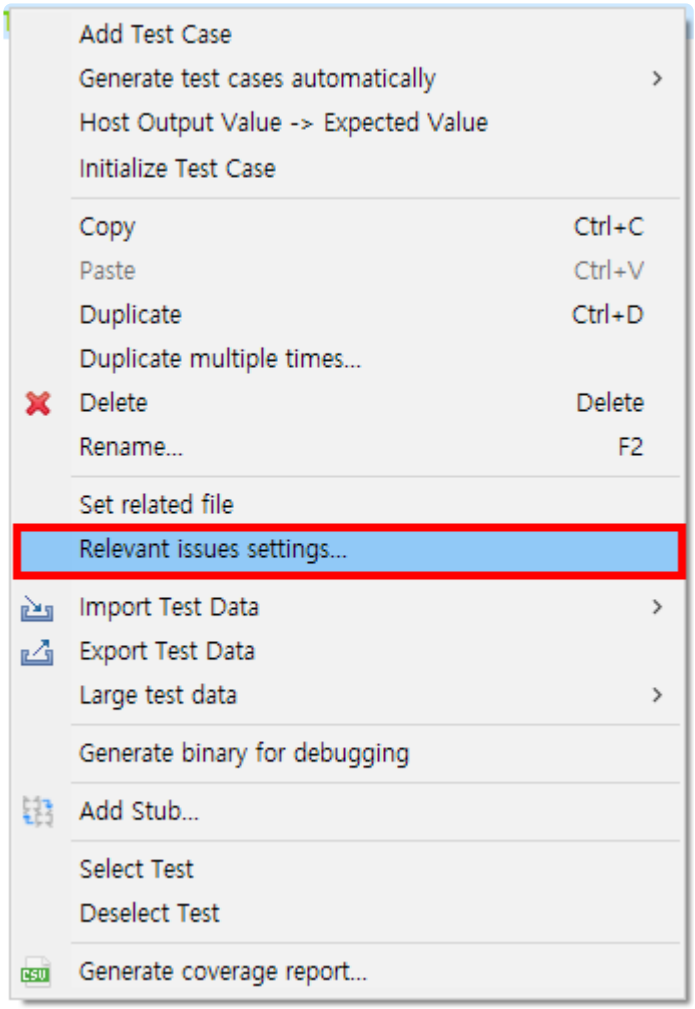
h. None

It does not create a function.

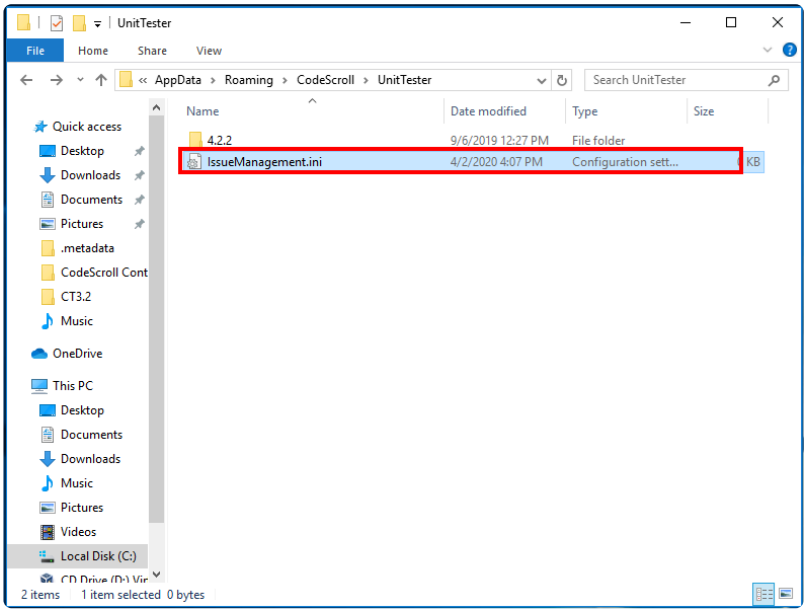
Relevant issues settings

It associates the selected test with the issue in the issue management tool registered. The Controller Tester supports the following issue management tools: JIRA, Trac, Redmine, Mantis, Bugzilla.

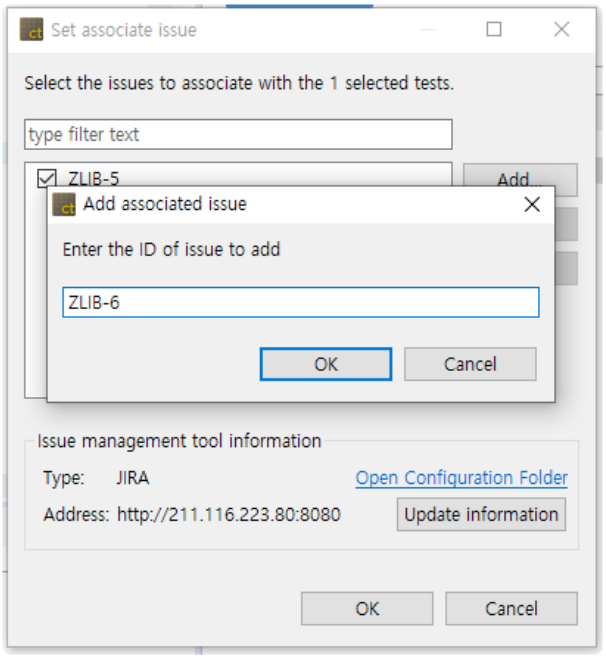
1. In the context menu of Unit Test View, click [Relevant issues settings…].



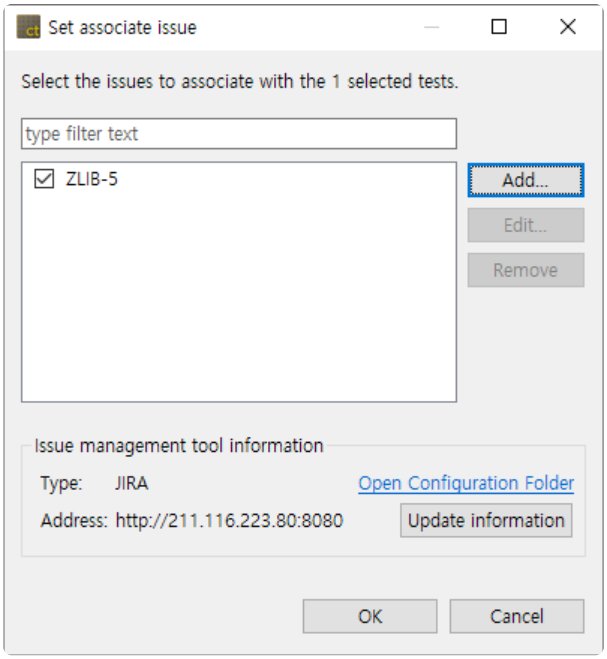
2. Enter the information of the issue management tool in the configuration file (IssueManagement.ini).



3. Add the issues to be associated with the test.

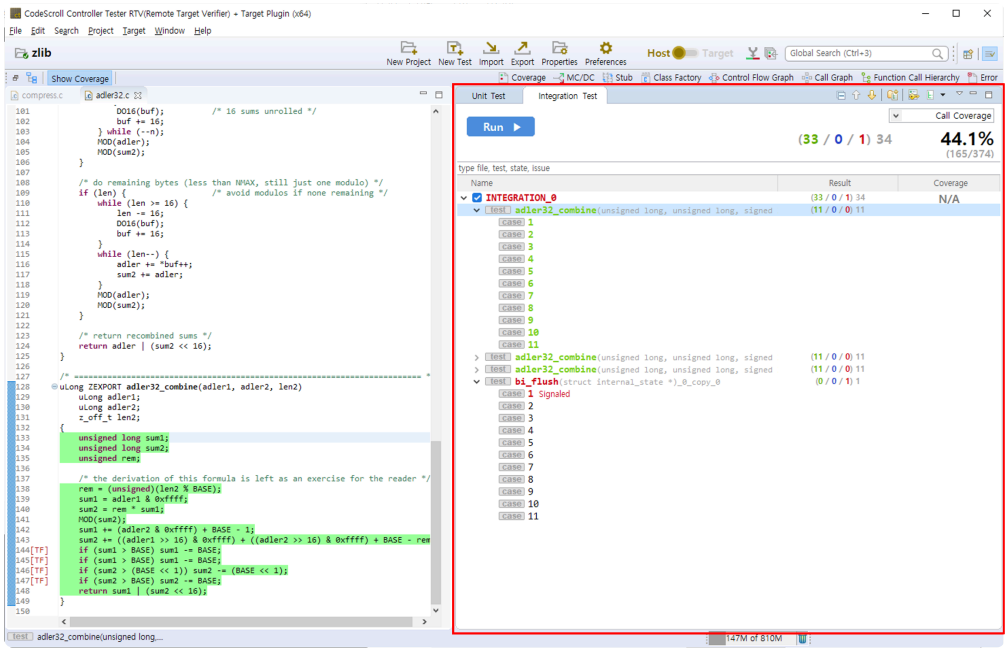


4. Check the checked issue list and click [OK].



16.4. Integration Test View

The Integration Test View provides the integration test unit that can group the unit tests. The unit tests under the integration test run in order with the context maintained.

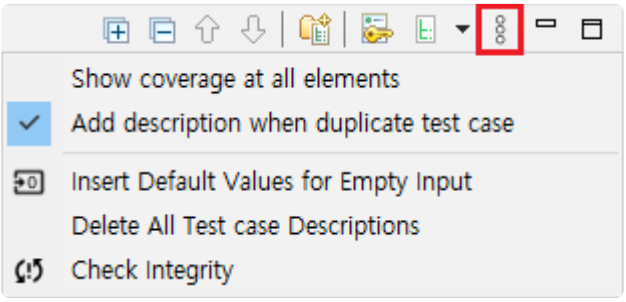


Toolbar menu in the Integration Test View

Toolbar icon	Description
Expand All	Shows all test view tree nodes.
Collapse All	Hides all test view tree nodes.
Move Up Test	Moves the selected test up.
Move Down Test	Moves the selected test down.
Create	Creates an integration test.
Shows as Unique Test Name	Displays with unique test name.
Total	Displays all the tests.
Failure/Error	Displays only the tests that the test result is Failure/Error.
Failure	Displays only the tests that the test result is Failure.
Error	Displays only the tests that the test result is Error.
Success	Displays only the tests that the test result is Success.
Function changed	Displays only the tests that the function to be tested has been changed.
Run not guaranteed	Displays only the tests that the test execution is not ensured.

Host/Target result different	Displays only the test that the host result and the target result are different.

Pull-down menu in the Integration Test View



Menu name	Description
Show coverage at all elements	Displays the coverage in all items.(integration test, test, test case)
Add description when duplicate test case	Adds original test case number on duplicated test case.
Insert Default Values for Empty Input	When input data are empty, inserts default values.
Delete All Test case Descriptions	Delete test case descriptions all together.
Check Integrity	Checks integrity about functions and stubs. Then, reset test if it needs.

- Insert Default Values for Empty Input

Variable type	Default value
string	(empty string)
without string	0

For [Test reconfiguration], refer to [Source Code Modification and Test Reconfiguration]

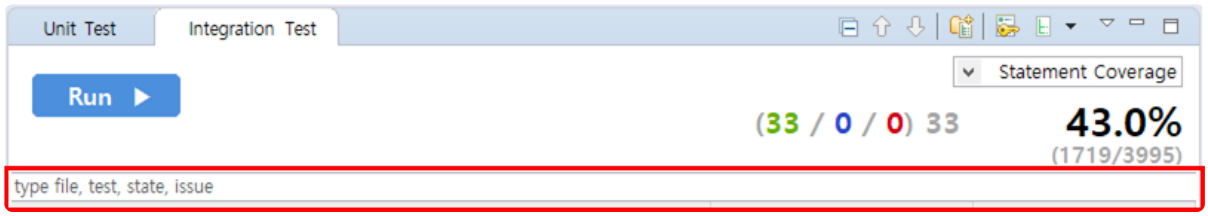
Dashboard in the Integration Test View

Menu	Description

Run ▶	Runs integration tests.
(878 / 0 / 359) 1237	Shows success, failure, error and total of test case.
Statement Coverage	Selects the coverage type to be displayed in the Integration Test View. (statement, branch, MC/DC, function call)
43.7% (1952/4466)	Shows the selected kind of full coverage.

Search

In the Integration Test View, you can search the integration tests, tests and test cases by file, function, test, status (success, failure, error) and issue name.



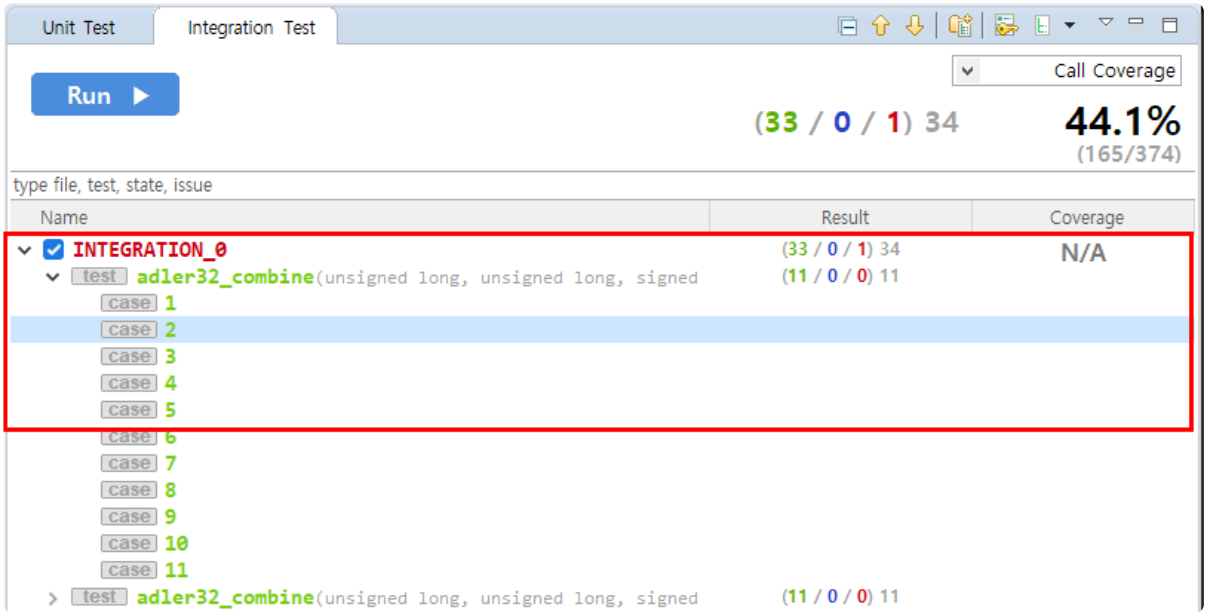
Status search keyword

In the Integration Test View, you can filter the tests displayed by the status search keyword. It provides searchable keyword recommendation and auto-complete function.

Keyword	Description
`%TEST_SUCCESS%`	Test success
`%TEST_FAILURE% %TEST_ERROR%`	Test failure/error
`%TEST_FAILURE%`	Test failure
`%TEST_ERROR%`	Test error
`%TEST_HAS_NOT_FUNCTION%`	Function changed
`%TEST_NOT_GUARANTEE%`	Run not guaranted
`%TEST_RESULT_DIFFERENT%`	Host/Target result different

Structure of the Integration Test View

The Integration Test View presents the hierarchy structure of [Integartion tes] > [Test] > [Test case].



Item icons in the Integration Test View

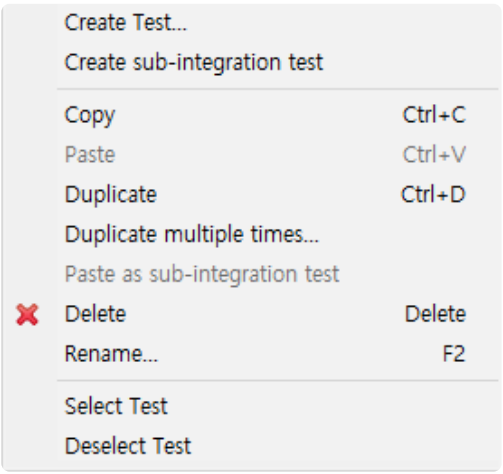
Item icon	Description
None	Integration Test
test	Test
test	Global variable test
case	Test case, Test case group

Item status color in the Integration Test View

The execution information of functions, tests, and test cases in the Integration Test View is represented by colors.

Color	Description
Green	Integration Test/Test: If the execution result of all the test cases under it is success. Test case: If the result of execution is success.
Blue	Function/Test: If only test cases with failure and no test cases with errors exist under it. Test case: If the result of execution is failure.
Red	Function/Test: If test cases with error exist under it. Test case: If the result of execution is error.
Orange	Function/Test: All test cases have not been executed under it and test cases that are not guaranteed to run exist. Test case: Run not guaranteed
Black	Function/Test: All test cases under it have not been executed. Test case: Not been executed

Context menu in the Integration Test View



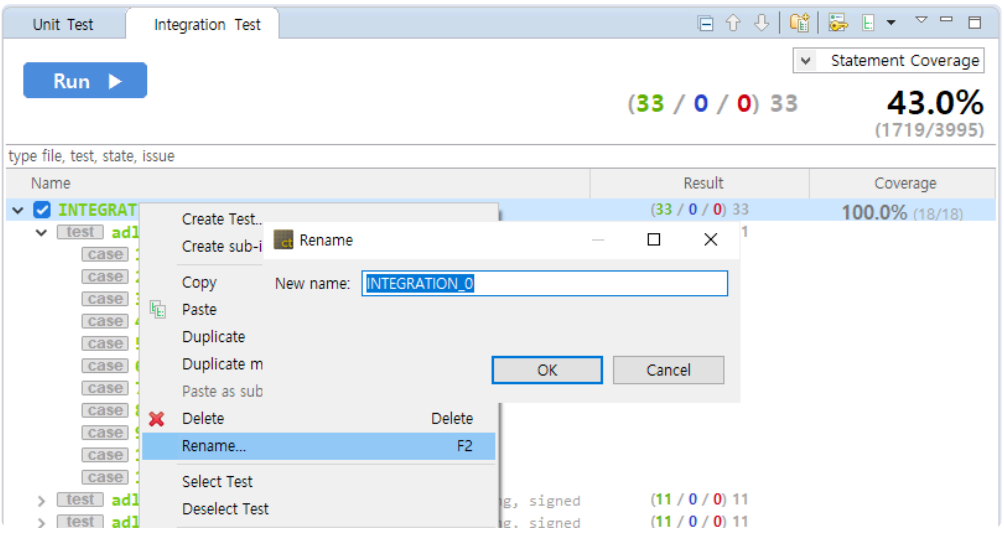
Context menu	Description
Create Test	Creates the dialog for creating a test
Create sub-integration test	Creates a sub-integration test
Copy	Copies the test and test case
Paste	Pastes the test and test case
Duplicate	Duplicates the test and test case
Duplicate multiple times	Duplicates tests and test cases as many as the number entered
Paste as sub-integration test	Pastes the integration test copied into the integration test selected as a sub-integration test
Delete	Deletes the test and test case
Rename	Modifies the test name
Select tests	Checks all checkbox of the tests selected
Deselect tests	Checks all checkbox of the tests selected

Create Integration test and Rename

The integration test can be created in the toolbar menu. The name of the integration test is assigned automatically and can be changed by using [Rename] context menu.

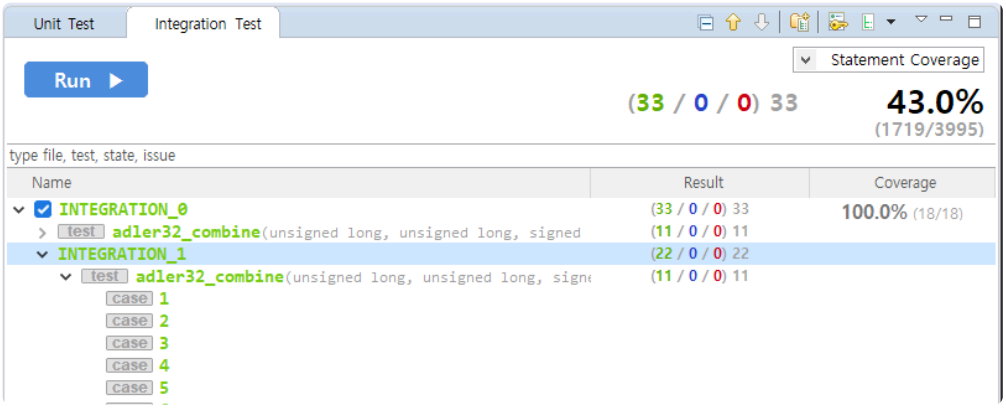
- Create

- Rename



Sub-integration test

You can create a sub-integration test under the integration test by using [Create sub-integration test] context menu or pasting an existing integration test into other integration test using [Paste as sub-integration test] context menu.



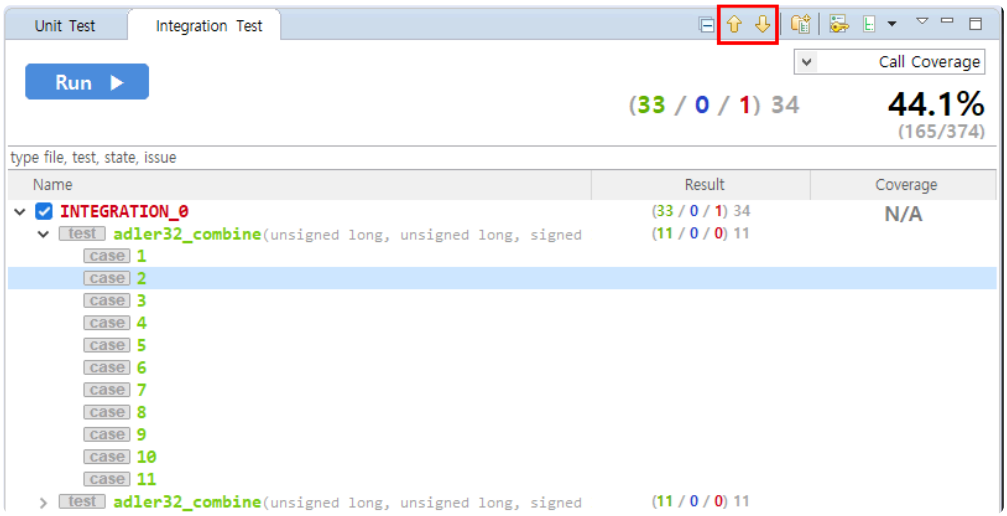
Add a test and change the order of run

There are four ways to add tests to the integration test.

1. You can select an integration test and create it by using [Create Test…] context menu.
2. You can add by dragging and dropping the test in the Unit Test View to the integration test.
3. You can select the test of the Integration Test View and copy/paste it by using the context menu.
4. You can select the test in the Unit Test View and copy/paste it by using the context menu.

There are two ways to change the execution sequence for the test.

1. You can select a test or test case and change it by moving with a mouse.
2. You can select a test or test case and change it by selecting either [Move Up Test] or [Move Down Test] in the toolbar.

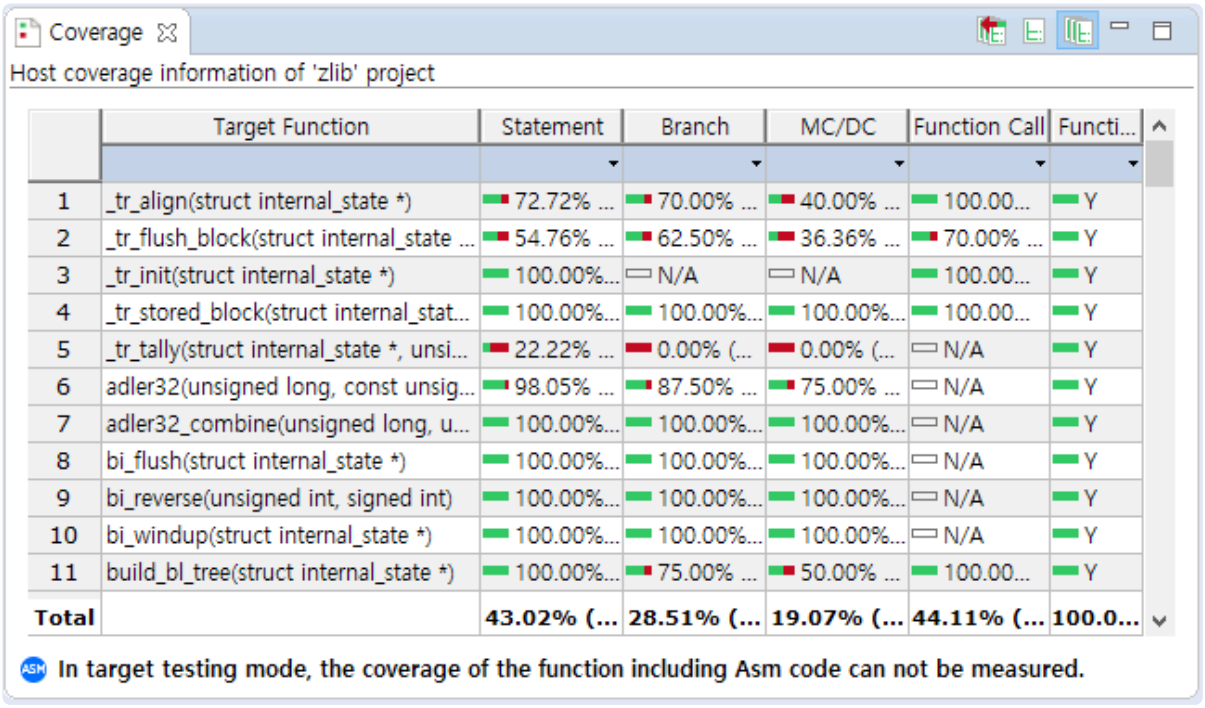


The coverage result of integration test

If you select the integration test, the result can be checked in the Coverage.

16.5. Coverage View

It shows the coverage of the function to be tested after test has been run as a percentage.



	Target Function	Statement	Branch	MC/DC	Function Call	Functi...
1	_tr_align(struct internal_state *)	72.72% ...	70.00% ...	40.00% ...	100.00...	Y
2	_tr_flush_block(struct internal_state ...	54.76% ...	62.50% ...	36.36% ...	70.00% ...	Y
3	_tr_init(struct internal_state *)	100.00%...	N/A	N/A	100.00...	Y
4	_tr_stored_block(struct internal_stat...	100.00%...	100.00%...	100.00%...	100.00...	Y
5	_tr_tally(struct internal_state *, unsi...	22.22% ...	0.00% (...	0.00% (...	N/A	Y
6	adler32(unsigned long, const unsig...	98.05% ...	87.50% ...	75.00% ...	N/A	Y
7	adler32_combine(unsigned long, u...	100.00%...	100.00%...	100.00%...	N/A	Y
8	bi_flush(struct internal_state *)	100.00%...	100.00%...	100.00%...	N/A	Y
9	bi_reverse(unsigned int, signed int)	100.00%...	100.00%...	100.00%...	N/A	Y
10	bi_windup(struct internal_state *)	100.00%...	100.00%...	100.00%...	N/A	Y
11	build_bl_tree(struct internal_state *)	100.00%...	75.00% ...	50.00% ...	100.00...	Y
Total		**43.02% (...**	**28.51% (...**	**19.07% (...**	**44.11% (...**	**100.0...**

Coverage type provided in Controller Tester

Coverage	Description
Statement	Percentage of source code statement executed by the test.
Branch	Percentage of branch executed by the test. 100% branch coverage includes 100% decision coverage and 100% statement coverage.
MC/DC	Coverage considering the condition coverage and the decision coverage complexly(described in MC/DC view).
Function call	Percentage of the function called by the test among all the functions.
Function	Percentage of functions that have been called at least once in all functions.

Label icon in the Coverage View

Icon	Description
	Function changed.
	Function containing Asm code.

Toolbar menu in the Coverage View

Toolbar icon	Description
	Shows coverage for each test.
	Shows full coverage.
	Shows full coverage(including the external coverage).

Show Coverage

You can check the coverage of the tests in the Coverage View or in the Source Code Editor.

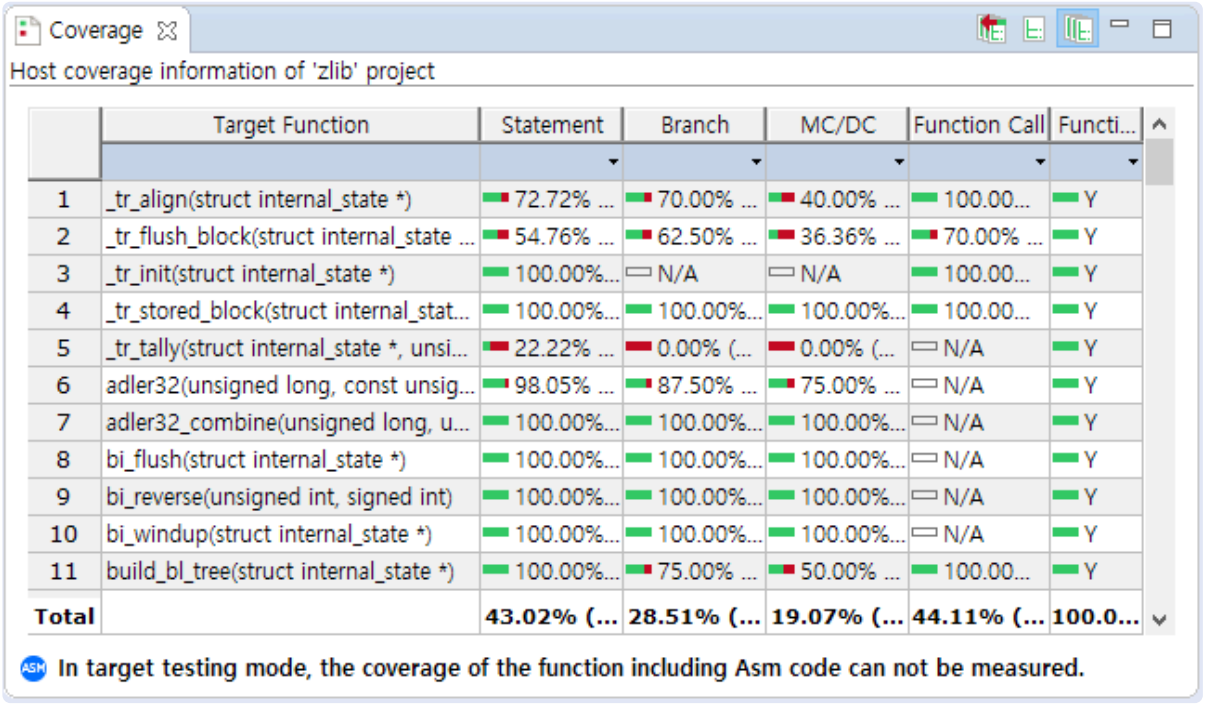
Show coverage in the Coverage View

The Coverage View shows the coverage information of the test or test case selected in the Unit Test View and Integration Test View. If you click [Show full coverage], the coverage information for all tests can be checked by merging them. And if you click [Show full coverage (Include External Coverage)], it shows also the coverage imported externally by merging them.

The coverage for the items selected in the tree is displayed at the bottom of the Coverage View.

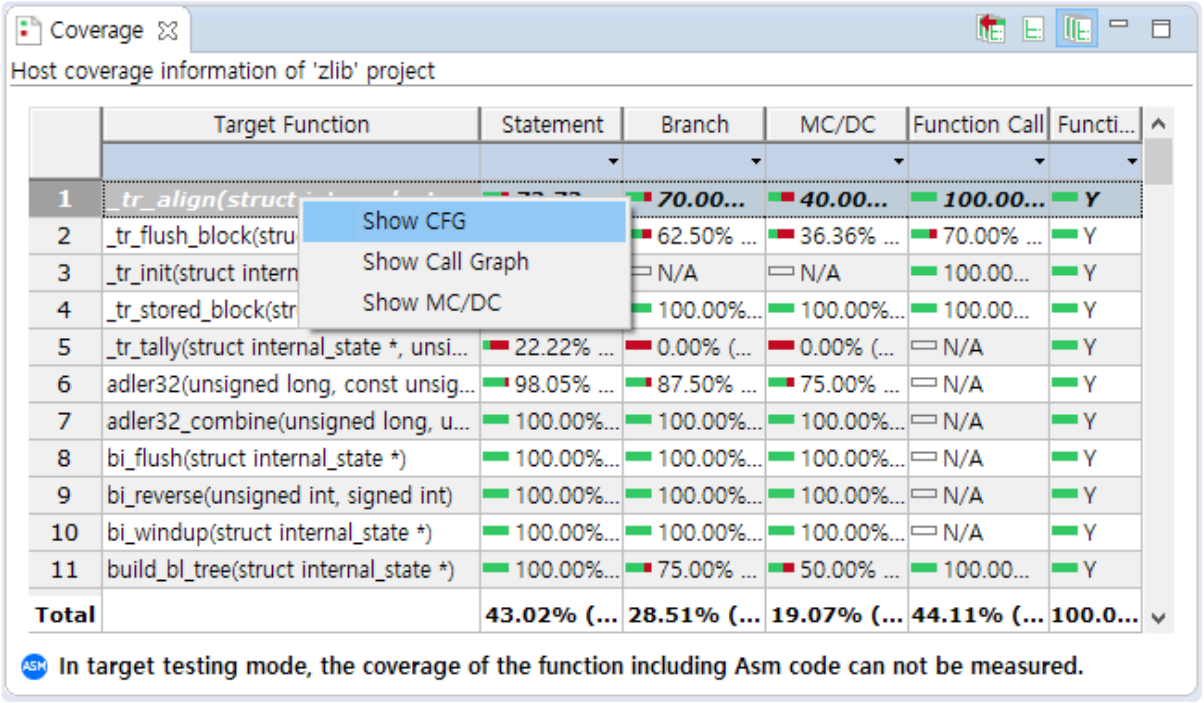
Total		**43.02% (171...**	**28.51% (603/2115)**	**19.07% (239/1253)**	**44.11% (165/...**	**100.00% (114/...**

In the Coverage View, the coverage for each function is displayed for the items selected in the Unit or Integration Test View.



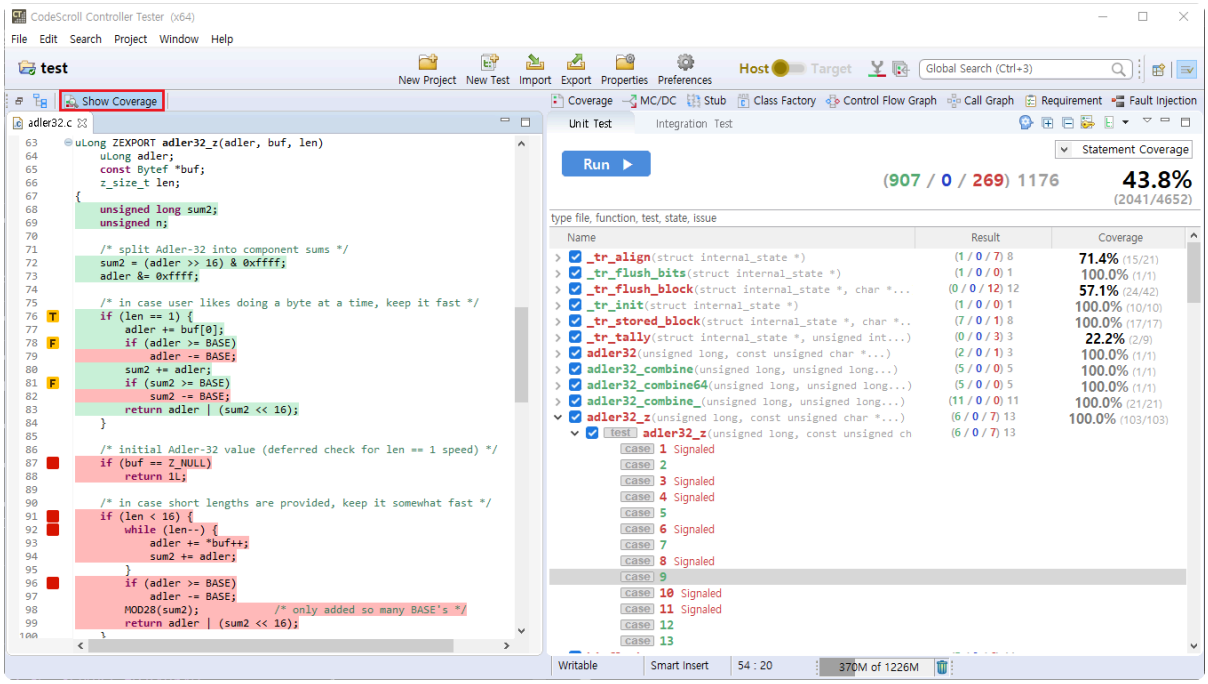
	Target Function	Statement	Branch	MC/DC	Function Call	Functi...
1	_tr_align(struct internal_state *)	72.72% ...	70.00% ...	40.00% ...	100.00...	Y
2	_tr_flush_block(struct internal_state ...	54.76% ...	62.50% ...	36.36% ...	70.00% ...	Y
3	_tr_init(struct internal_state *)	100.00%...	N/A	N/A	100.00...	Y
4	_tr_stored_block(struct internal_stat...	100.00%...	100.00%...	100.00%...	100.00...	Y
5	_tr_tally(struct internal_state *, unsi...	22.22% ...	0.00% (...	0.00% (...	N/A	Y
6	adler32(unsigned long, const unsig...	98.05% ...	87.50% ...	75.00% ...	N/A	Y
7	adler32_combine(unsigned long, u...	100.00%...	100.00%...	100.00%...	N/A	Y
8	bi_flush(struct internal_state *)	100.00%...	100.00%...	100.00%...	N/A	Y
9	bi_reverse(unsigned int, signed int)	100.00%...	100.00%...	100.00%...	N/A	Y
10	bi_windup(struct internal_state *)	100.00%...	100.00%...	100.00%...	N/A	Y
11	build_bl_tree(struct internal_state *)	100.00%...	75.00% ...	50.00% ...	100.00...	Y
Total		**43.02% (...**	**28.51% (...**	**19.07% (...**	**44.11% (...**	**100.0...**

In the Coverage View table, you can the control flow graph, the function call graph and MC/DC for each function via the context menu.



Show coverage in the Source Code Editor

If you enable [Show Coverage] icon in the main toolbar and click the wanted fucntion/test/test case in the Unit/Integration Test View, it shows which part of the source code is covered by using the color information. The green color indicates that the code is covered, the red color indicates that the code is not covered, and yellow color indicates that the portion of the code is covered only.



In the vertical column on the left of the Source Code Editor, the marker indicate whether the branch statement is true or false.

Marker	Description
T	The branch is only covered as true.
F	The branch is only covered as false.

	The branches are all covered.
	The branch is not covered.
	A line with multiple branches partially covered.

16.6. MC/DC View

The [MC/DC] view shows the information of MC/DC coverage(Modified Condition/Decision Coverage).

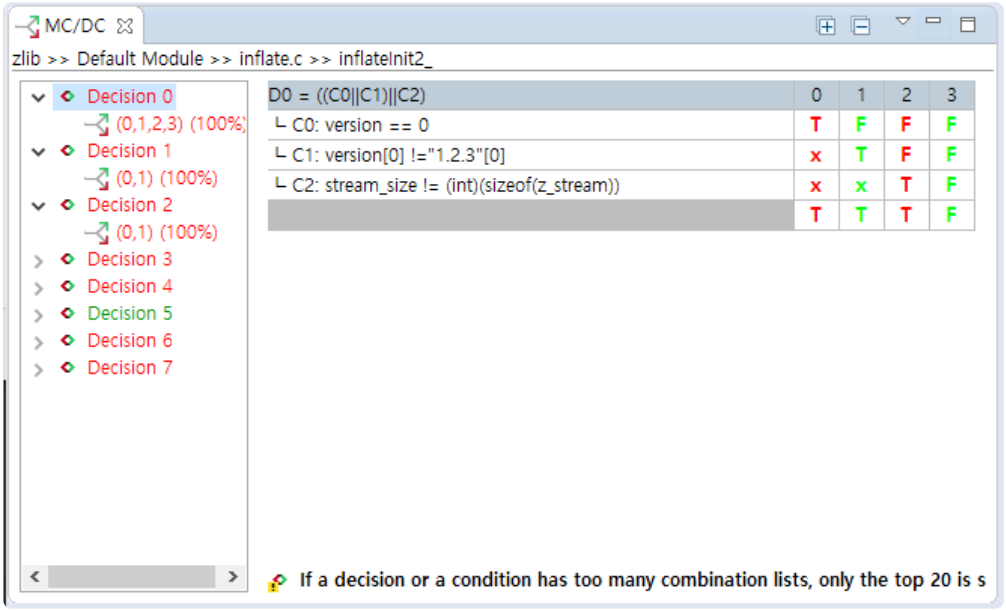
MC/DC coverage

MC/DC is the improved condition/decision coverage allowing each individual conditional expression to independently affect the result of the whole conditional expression without being affected by the other individual conditional expressions and it is stronger than the conditional/decision coverage.
If there is no change in the other conditions and it affects the result when the own state has changed, the state can be said to satisfy MC/DC, and the conditions for creating the truth table are as follows.

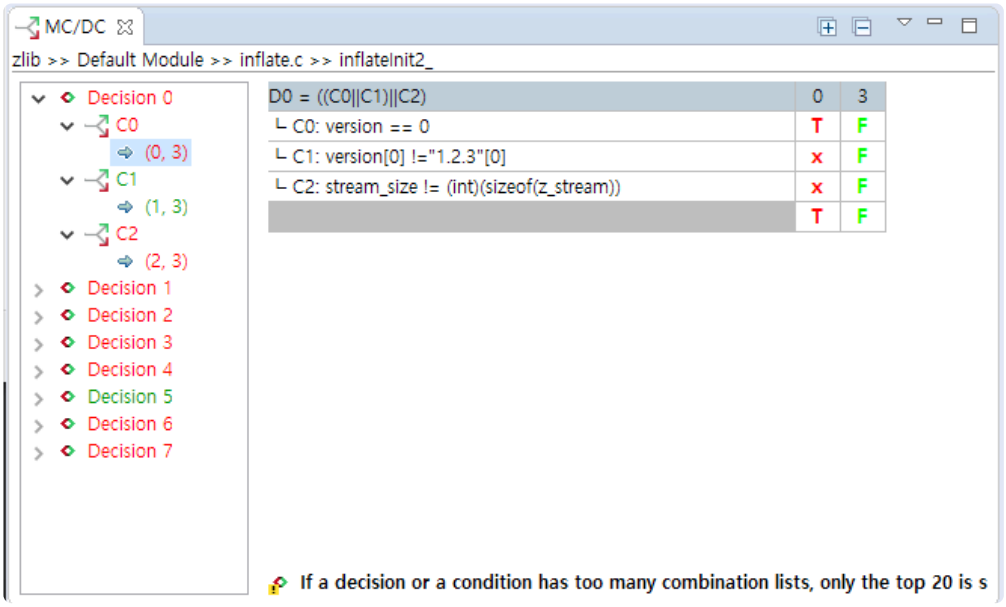
- The state of decision(decision statement) must satisfy at least once all possible results(true, false).
- All individual conditional expressions belonging to the decision must satisfy at least once all possible results(true, false).
- Each conditional expression belonging to the decision affects independently the result value of decision it belongs to without being affected by the other individual conditional expressions.

MC/DC view table

- View combination lists satisfying the target coverage
  The following figure is a view that can identify easily the combinations which need to be covered to achieve the target coverage.
  The left section shows the list of Decisions for the selected functions, below which the list of combinations satisfying the target coverage are displayed.
  The right section shows the truth table of the selected Decision or combinations and whether or not it is covered. The covered combination is displayed in green and the uncovered combination is displayed in red.
  In the Preferences page, you can change the target coverage rate and the screen view mode. And you can select all for the truth table and use ‘Ctrl + C’ or Copy clipboard function from the context menu.



- View combination lists satisfying the coverage for each condition
  The following figure is a view that can identify a one-pair combination satisfying one condition. The left section shows the list of combinations that satisfies the coverage for each condition. The right section shows the truth table for the selected combinations and whether or not it is covered.

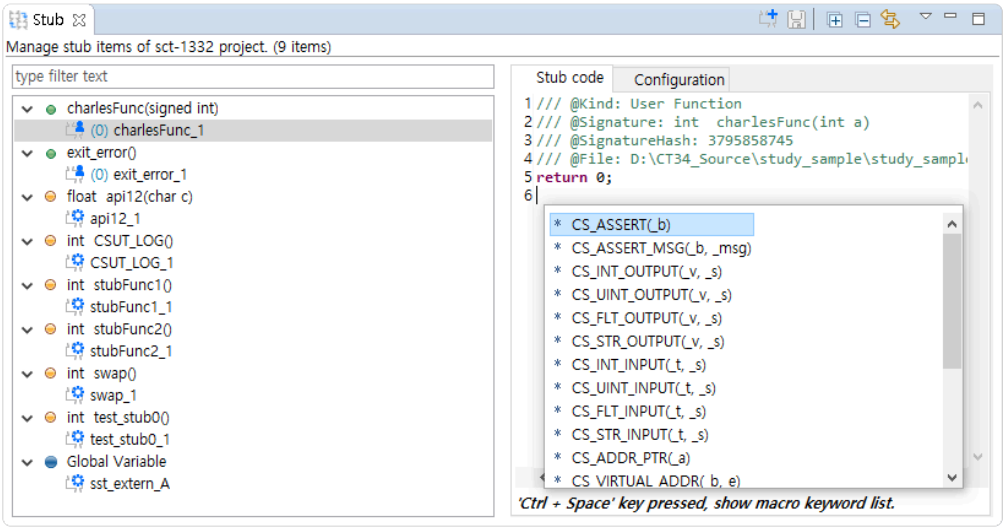


16.7. Stub View

In Controller Tester, if there are no libraries used in the target function when executing a test, if the libraries are not yet developed, or if the variables not declared are used, it creates automatically the stubs instead. The stubs are created when running the test and the body is not implemented.

You can add stubs directly in addition to the stubs created in Controller Tester. Multiple stubs can be created for the same function and the stubs created can be linked to each test.

You can control the behavior of each stub by using user macro within the stub. The value used in the user-entered macro can be controlled in Edit a test case tab for the test linked.



Function type

Type	Description
Function	Function without deficition
Function	Function with definition
Variable	Global variable

Stub type

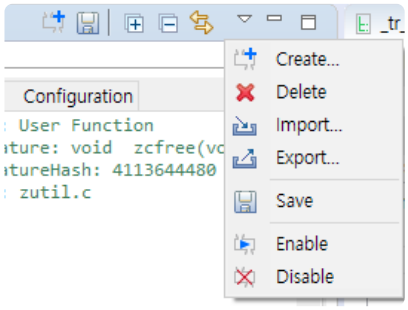
Type	Description
User stub	User-defined stub
Build stub	Auto-created stub
Old version stub	Stub with 2.3 or less version

Menu

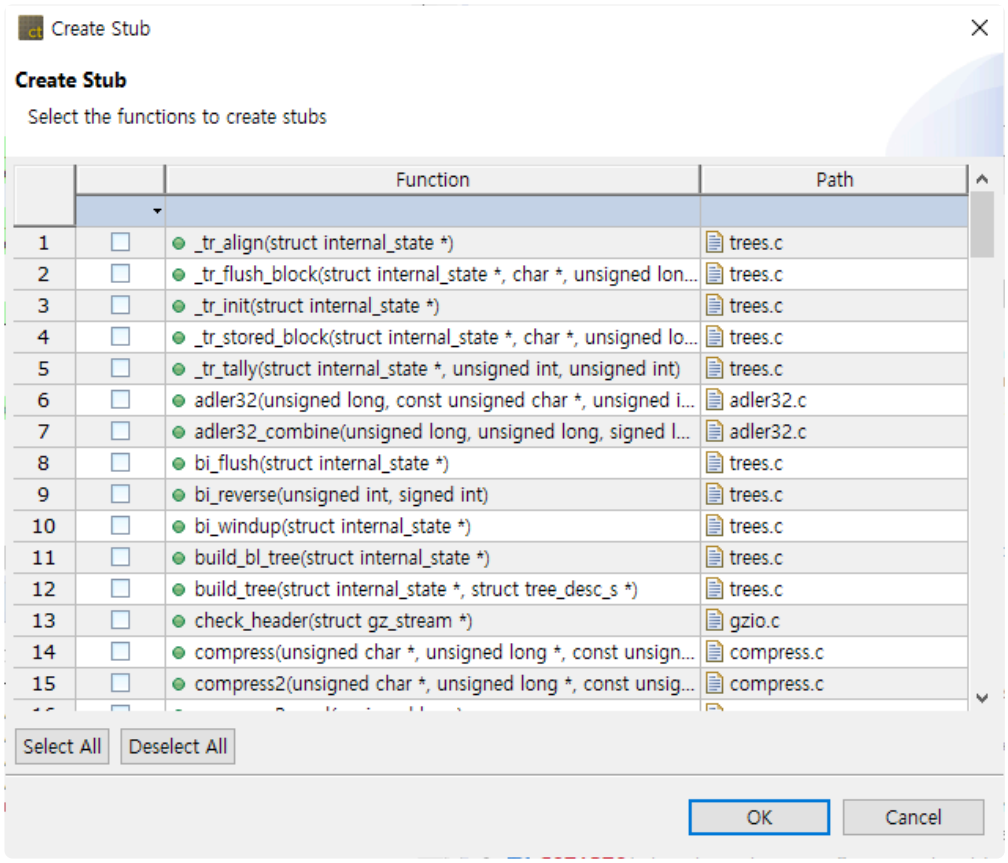
Menu	Description
Create	Creates stubs.
Delete	Deletes stubs.
Import	Imports the stub file exported.
Export	Exports a stub file.
Save	Saves the change stub information.
Enable	Enables stubs.
Disable	Disables stubs.

Create stubs

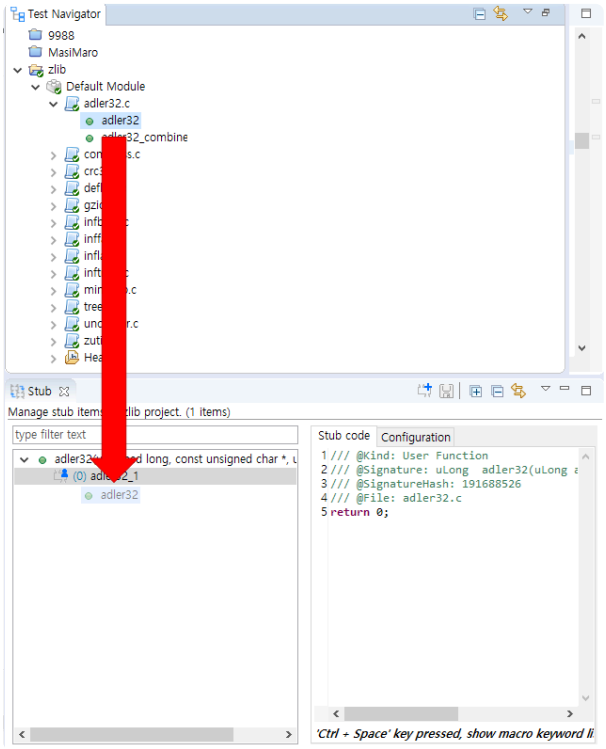
- Create stubs in Stub view
  1. Click the pull-down menu in Stub view and click [Create] menu.



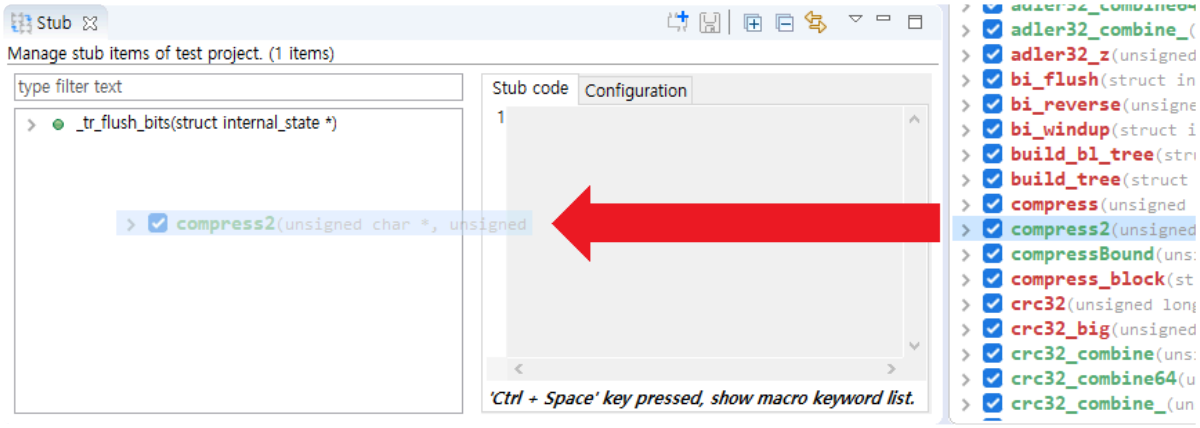
  2. Check the functions to be created as stubs and select [OK].



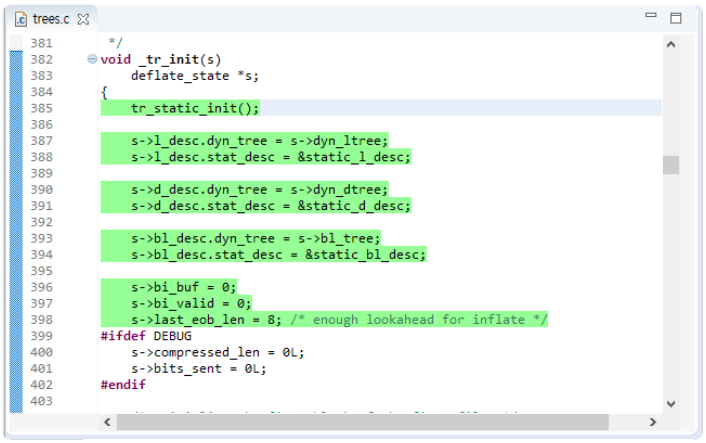
- Create stubs by drag-and-drop from Test Navigator view to Stub view.



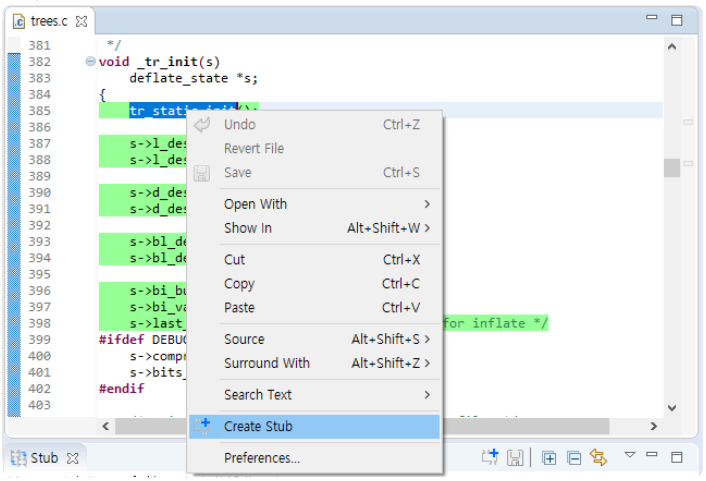
- Create stubs by drag-and-drop from Unit Test view to Stub view.



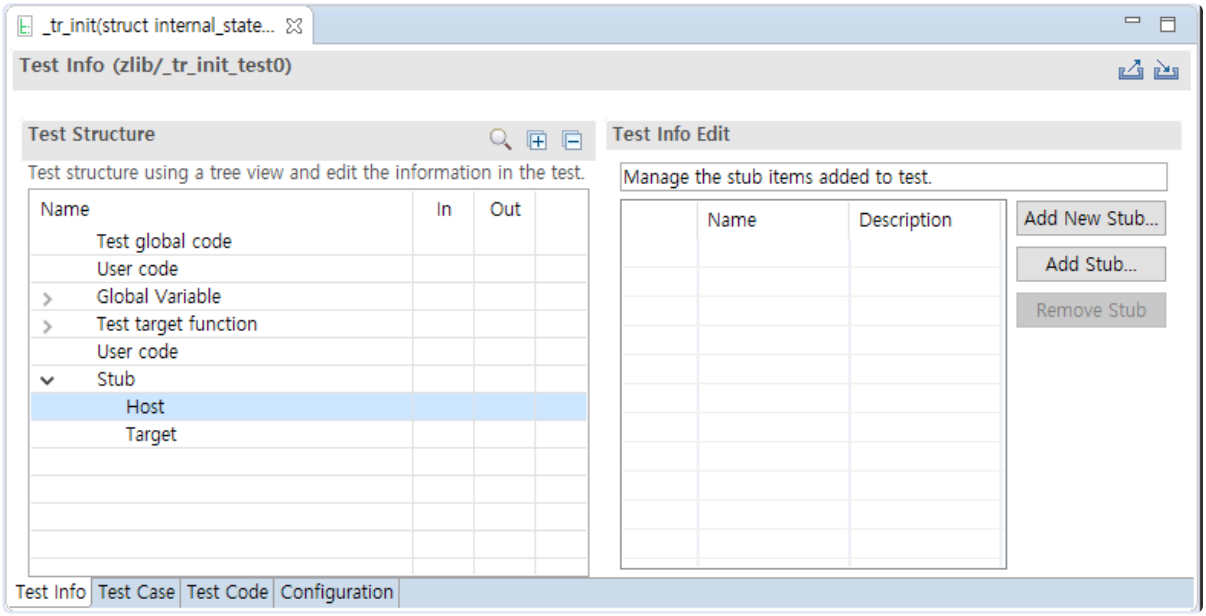
- Create stubs in source code editor
  1. Select the functions to be created as stubs.



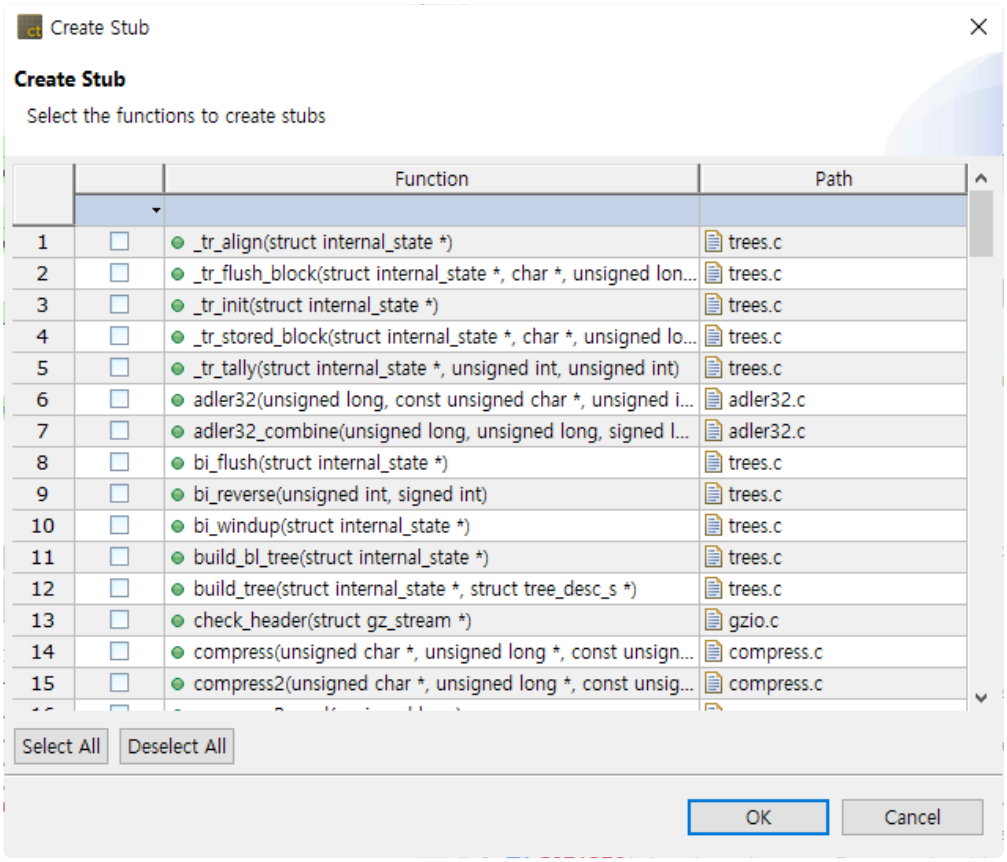
  2. Right-click it and select [Create Stub] context menu.



- Create and Link in Test Editor
  1. In the test structure of Test Editor, select the stubs or sub nodes.
  2. Select [Add New Stub…] button on the right section.

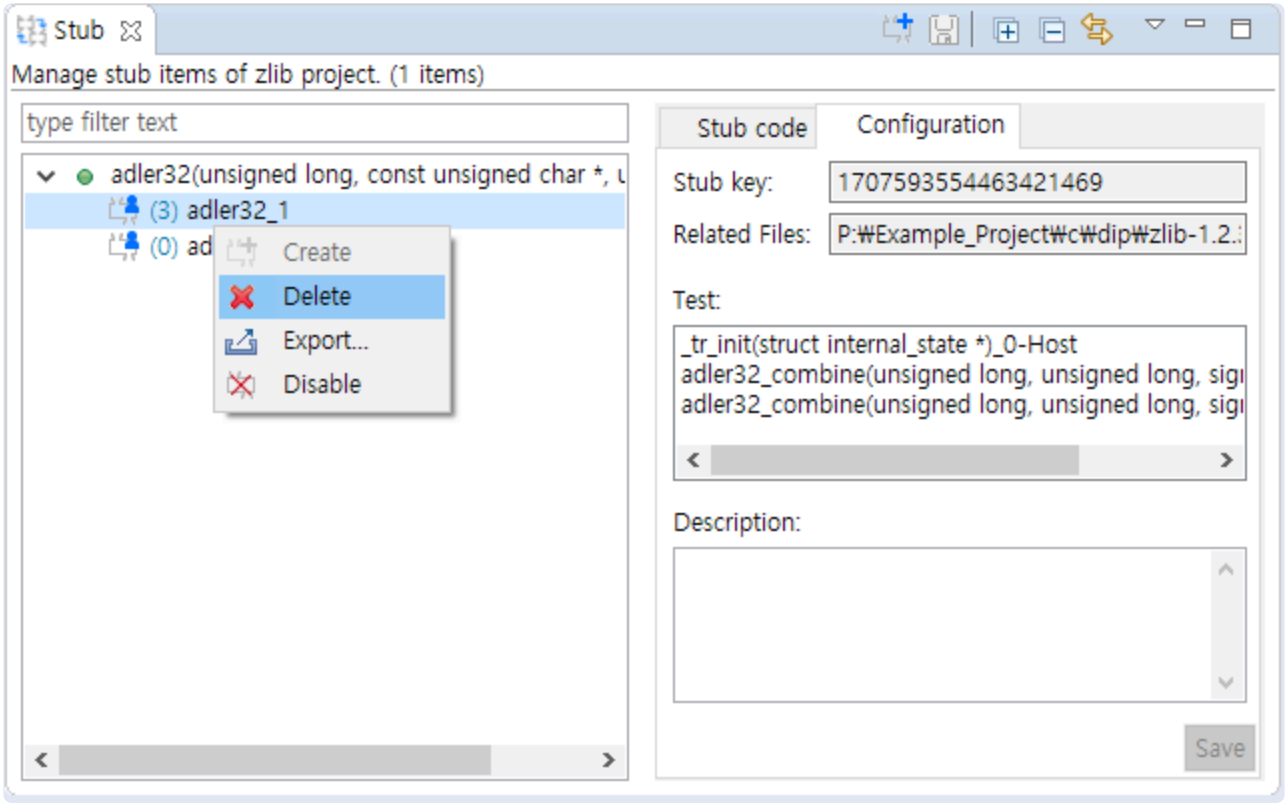


3. Check the functions to be created as stubs and select [OK].



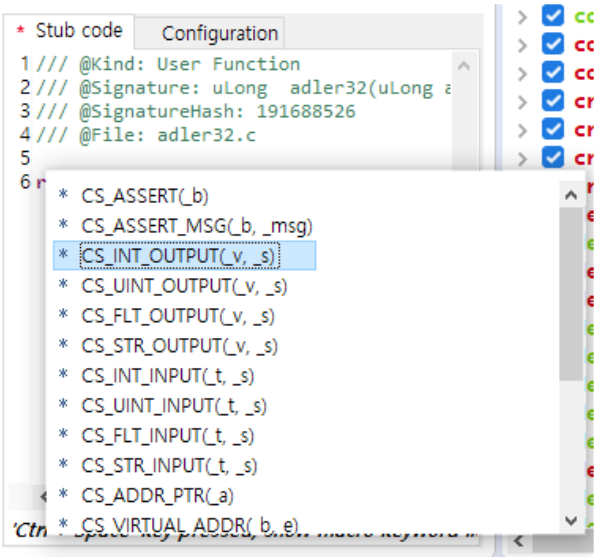
Delete stubs

- Select and right-click unnecessary stubs and select [Delete] menu.

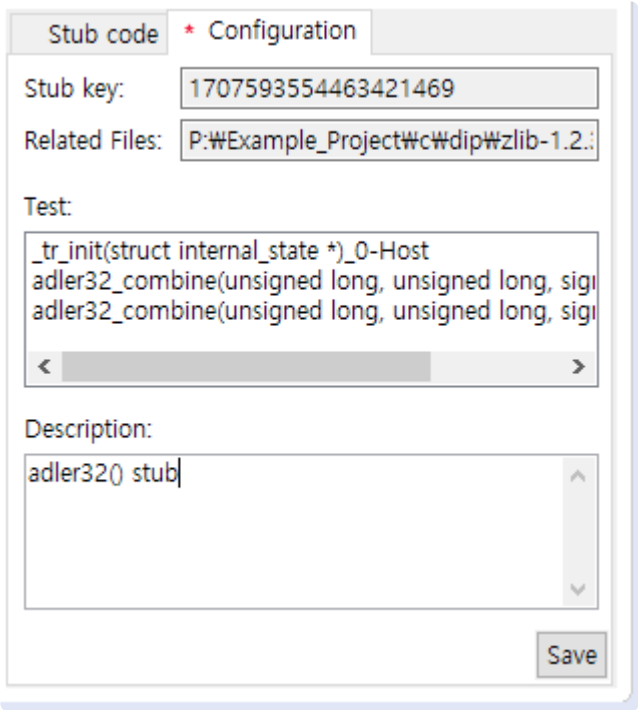


Edit stubs

- You can check the macro information by pressing ‘Ctrl + Space’ in the Stub Code screen. After editing the stub code, press ‘Ctrl + S’ or [Save] button in the toolbar to save it.

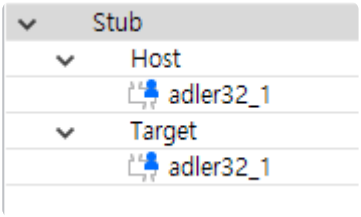


- In the stub setting screen, the description can be edited.



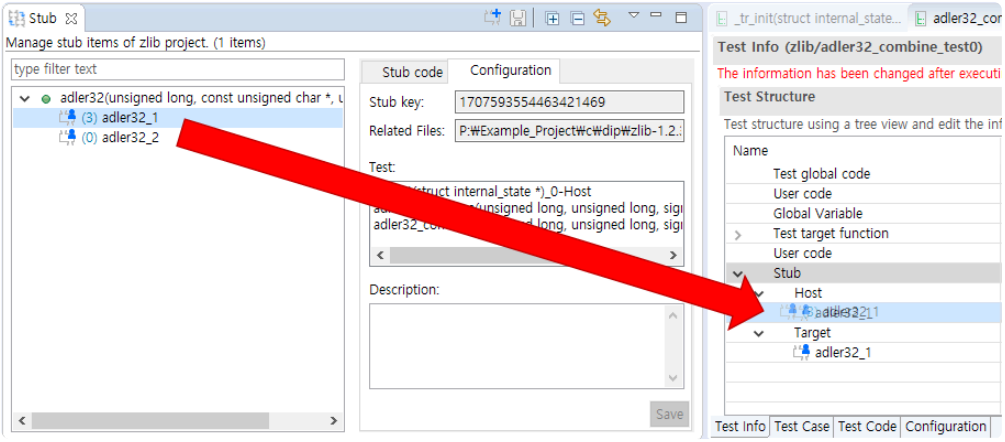
Link between stubs and tests

Each stub can be linked to the host/target test.
If you select “Stub” in the test structure editor, you can edit all the stubs included in the host and the target.



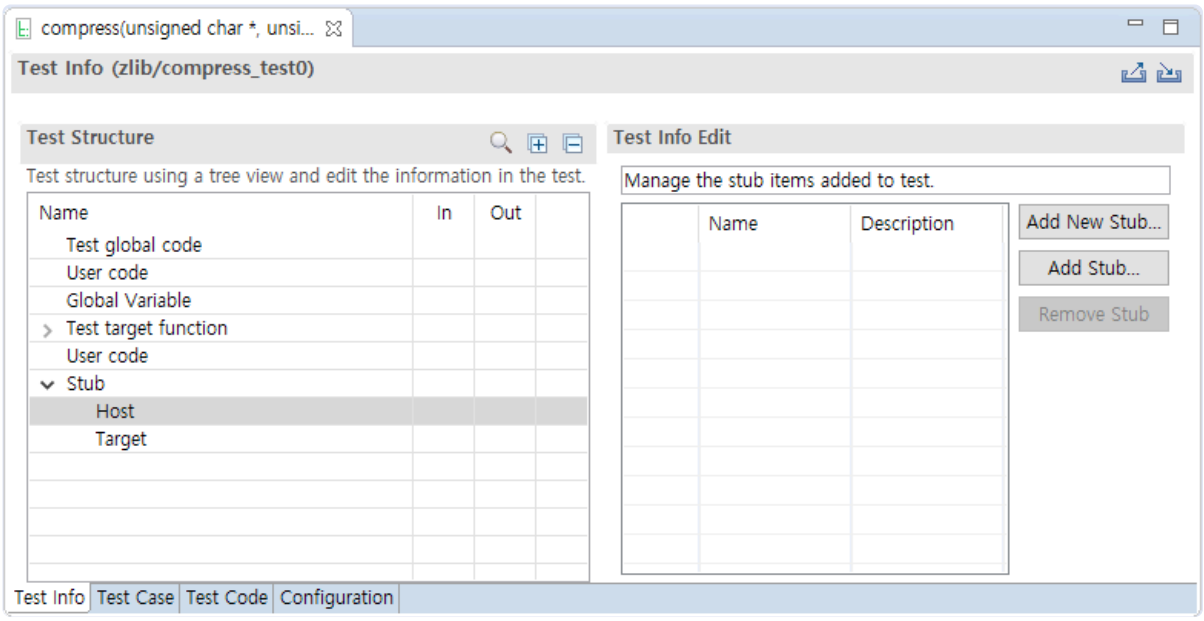
If you select “Host” or “Target”, you can only edit the stubs in the selected test environment.

- Dragging and dropping into the test editor in Stub view

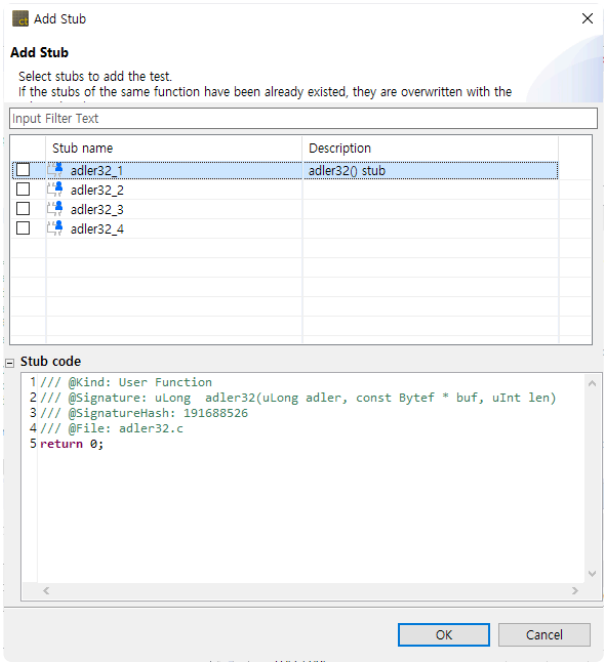


> If linked by drag & drop, it is linked to all the targets and all the hosts.

- Link with [Add Stub…] button in Test Editor.
  1. Select stubs from the test structure in Test Editor.
  2. Select [Add Stub…] button on the right section.

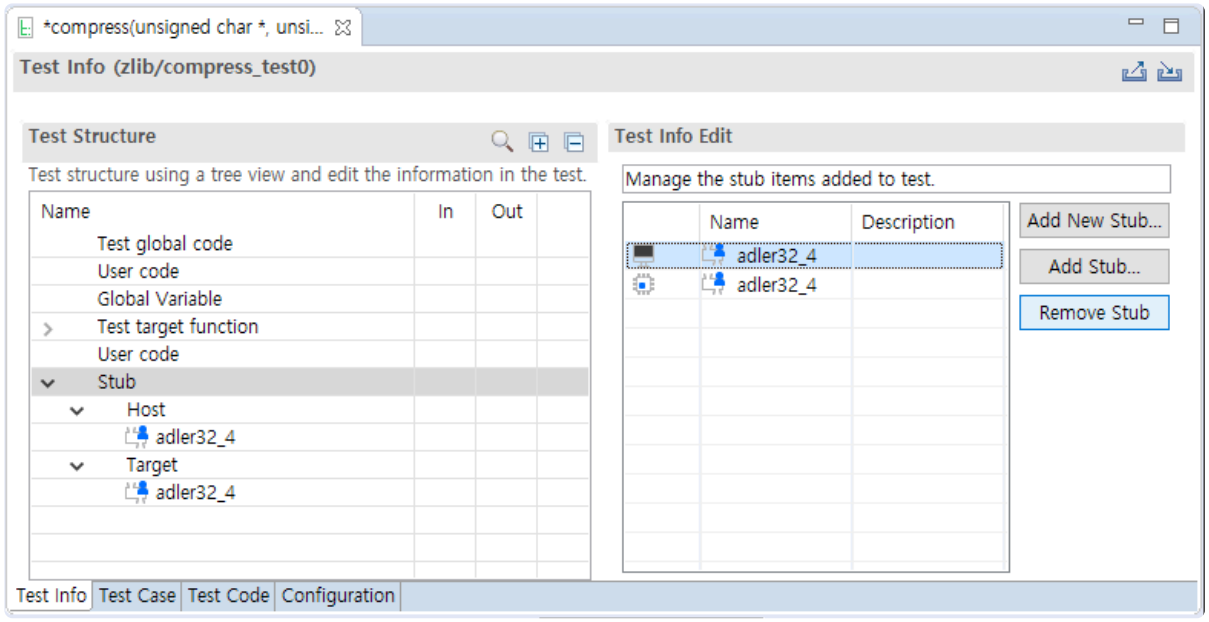


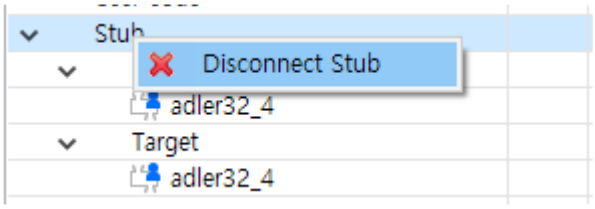
  3. Among the existing stub list, check the stubs to be linked and select [OK].



Unlink between stubs and tests

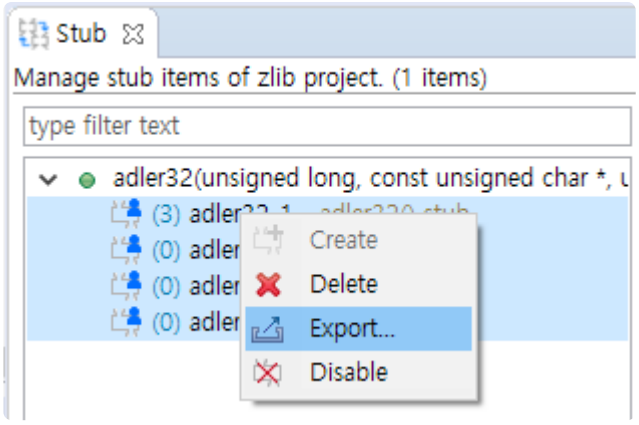
Select the stubs linked in the Test editor and select [Remove Stub] button or select the stubs in the test structure tree and right-click it to unlink them.



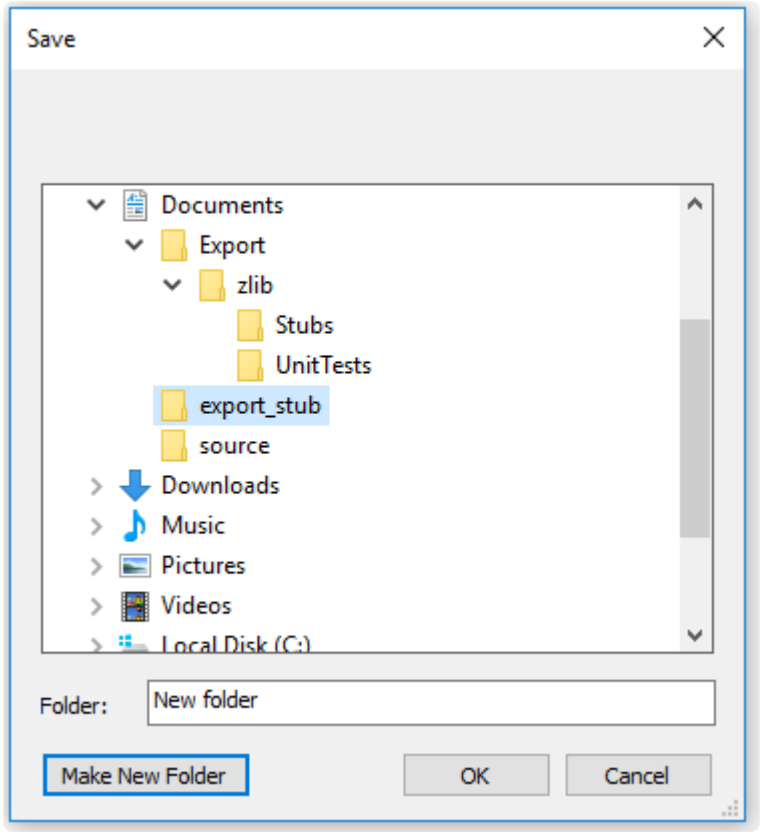


Export stubs

1. Select and the stubs to be exported and right-click and select [Export…] menu.

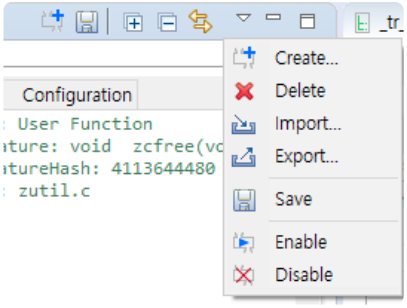


2. Specify the saving location of the stub file in the directory dialog and select [OK].

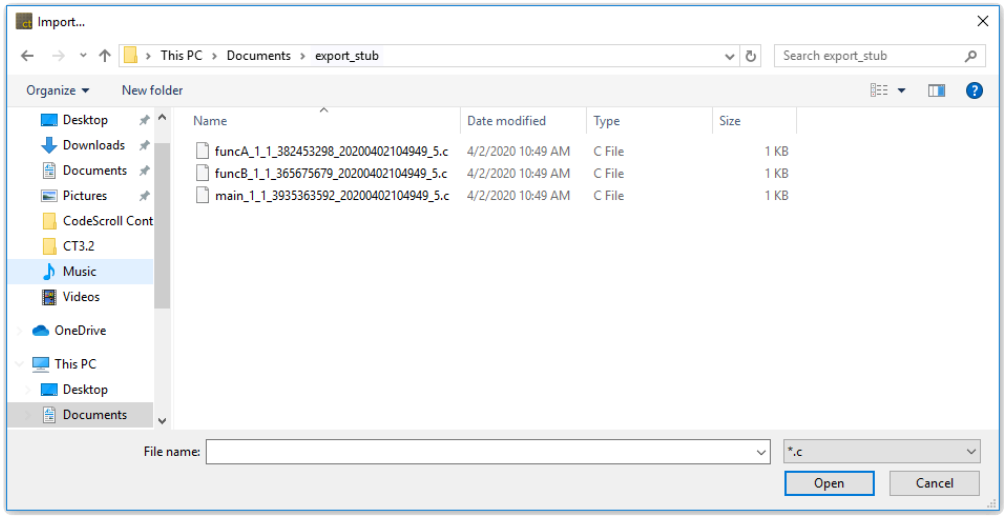


Import stubs

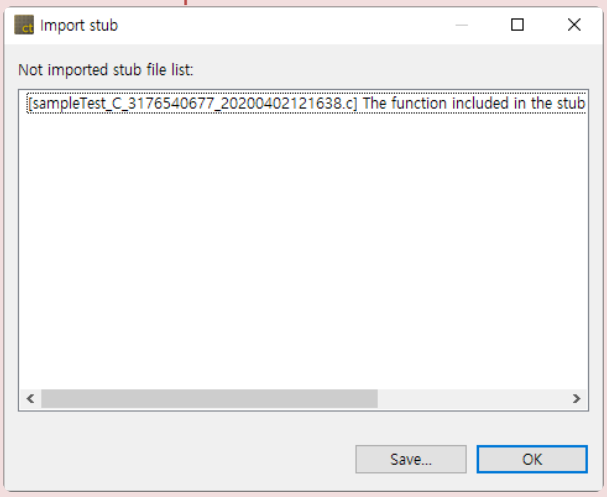
1. Select [Import…] menu from the pull-down menu in Stub view.



2. Select the stub file to be imported and select the [Open] button.



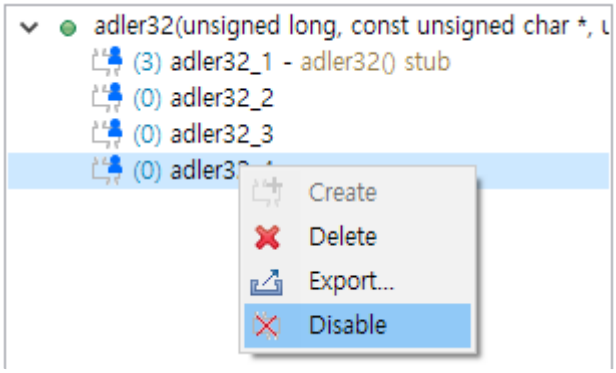
! The old version stubs or the stubs that are the same as build stubs existed in the project cannot be imported.



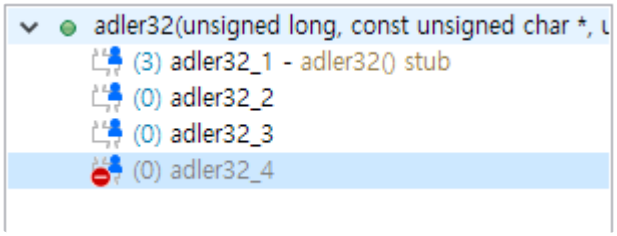
Use/Not use stubs

You can set whether to use the created stubs.

1. Select the stubs and right-click and select [Disable] menu.

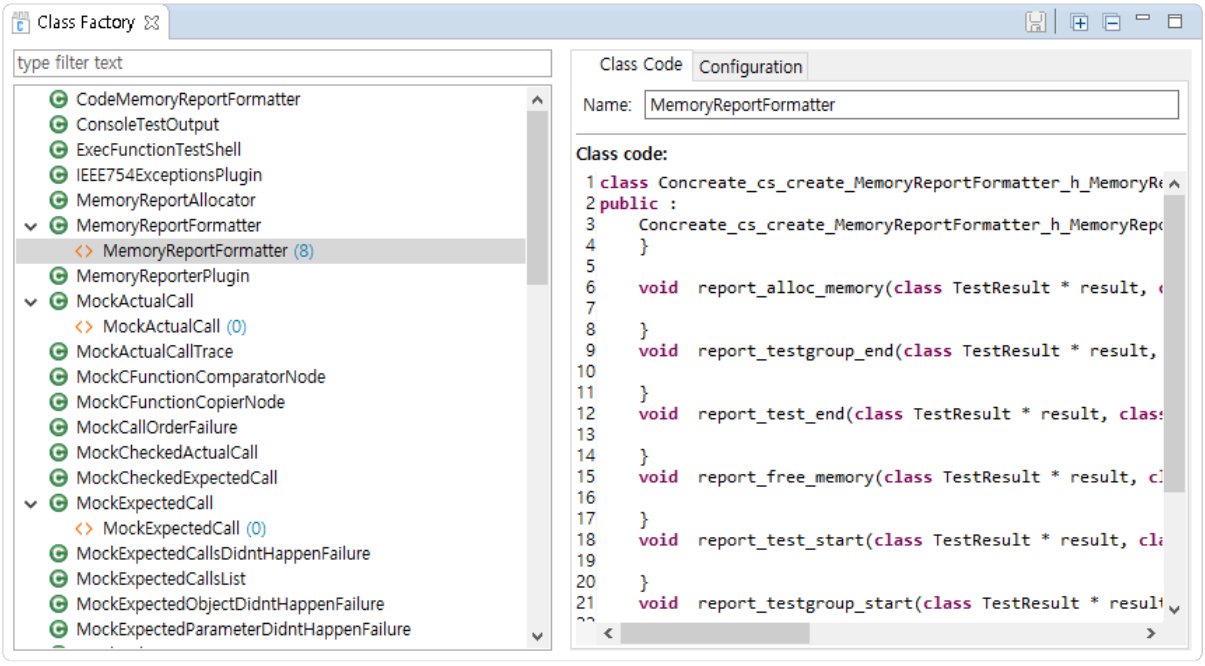


2. The stub image decorator is added and the color of the stub name changes to gray as shown in figure below.



16.8. Class Factory View

When testing C++ source code, you can use class code to create class objects. Since abstract classes cannot be instantiated, Controller Tester automatically generates instantiable class code.



- On the left side of the [Class Factory view], you can see the classes included in the project and the class code for instantiating the class in a tree structure.
- On the right side of the [Class Factory View], you can edit the class code and name in the [Class code] tab, and see the connected tests in the [Configuration] tab.

Toolbar menu of Class Factory view

Item	Description
Save	Save modification of the class code.
Expand All	Expand all class tree in Class Factory view.
Collapse All	Collapse all class tree in Class Factory view

Icon of Class Factory view

Icon	Description
Class	Class included in the project.
Abstract class	Abstract class included in the project.
Mock object	Mock object used in test.
Class code	Class code that create class objects to use in tests.

Context menu of class

Item	Description
Create	Create a class code of the selected class.

Context menu of class code

Item	Description
Delete	Delete selected class codes.

Using class factory

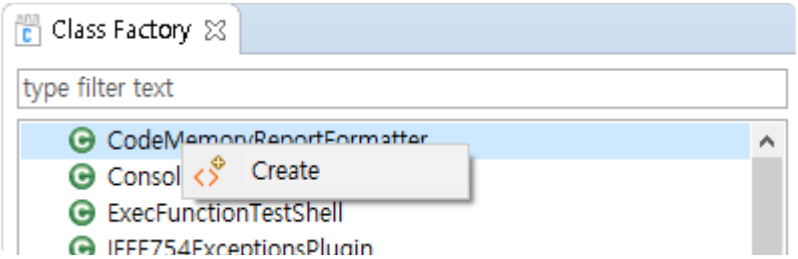
You can use the class factory in two ways.

- Using class codes
- Using mock objects

16.8.1. Using class codes

Create a class code

When analyze the project, a list of all class included in source codes appears in Class Factory view. Controller Tester automatically generates class codes of abstract classes. You can also manually create class code from the context menu of class.

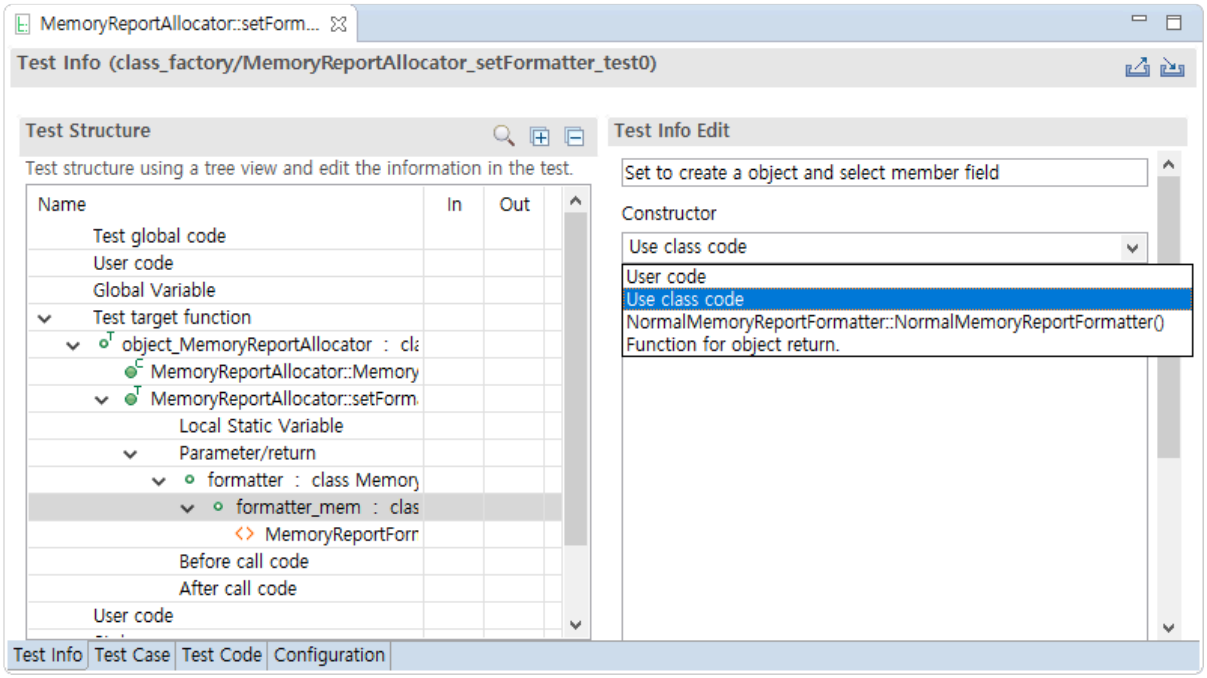


Apply class code

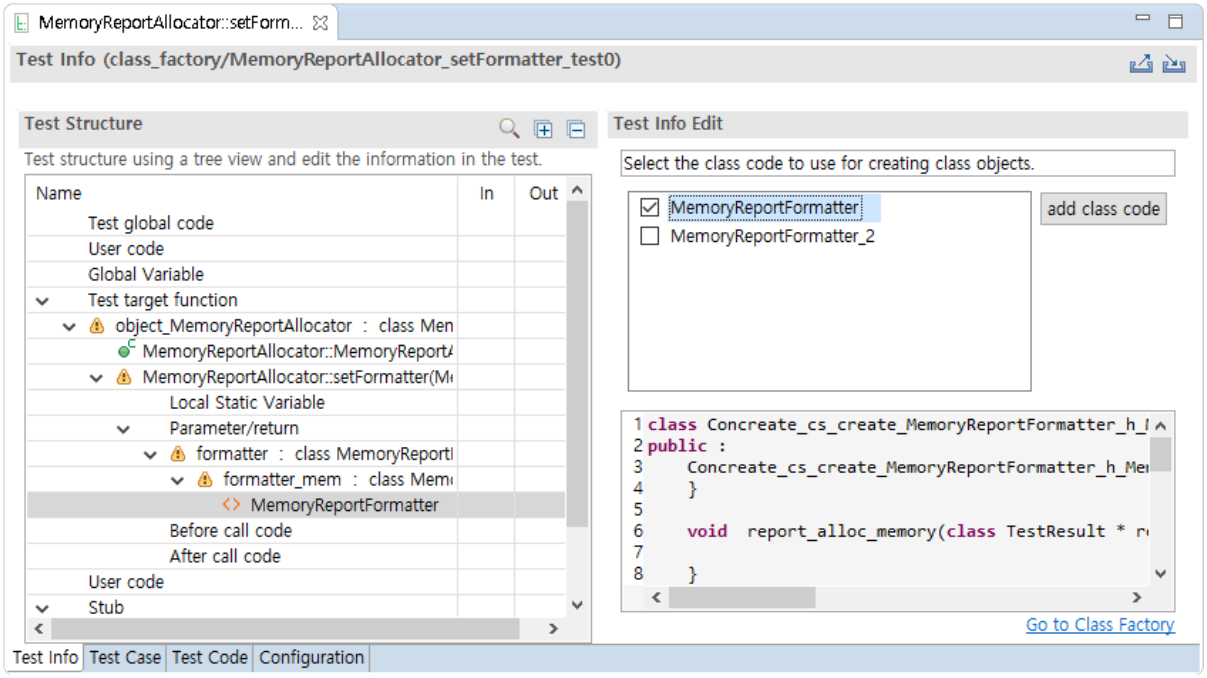
There are three ways to apply the class code to test.

Apply in Test Editor

1. Open the test editor by double-clicking on the test that you want to apply the class code. In the [Test Info] tab, expand the test structure, select the object to which the class code is to be applied, and select [Use class code] in [Test Info Edit] – [Constructor] on the right.



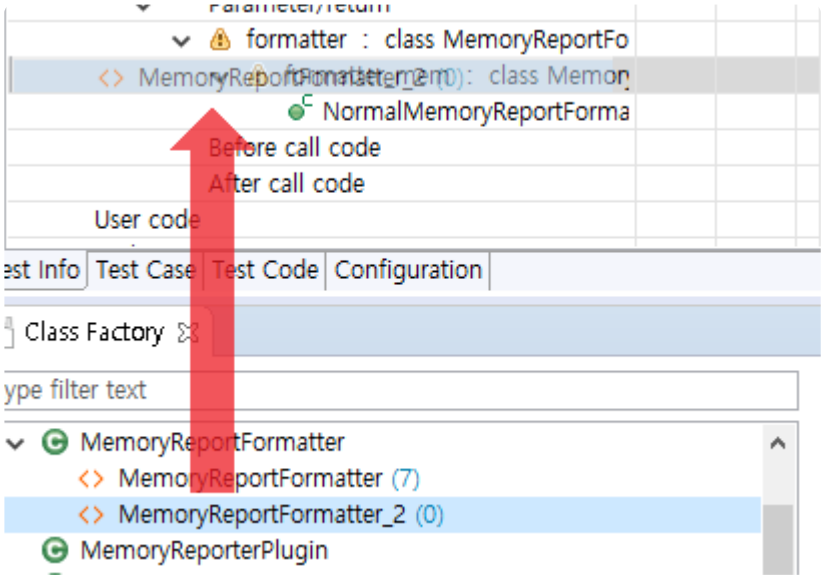
2. Select class code in [Test Structure]. In [Test Info Edit], check the class code to be applied and save. Select the class code to see the code.



- Select [Go to Class Factory] to open the class code in Class Factory view.
- Click [Add Class code] to add new class code.

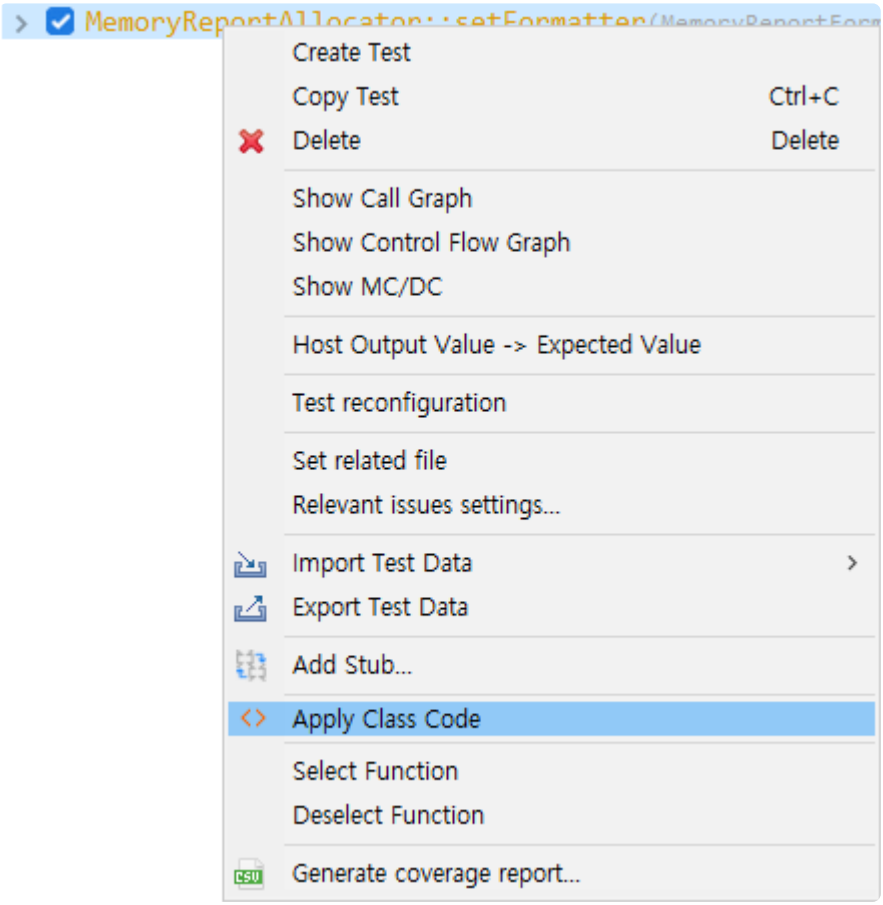
Apply by drag-and-drop

You can apply the class code by dragging and dropping the class code to be applied from Class Factory view to the class object in Test Editor.

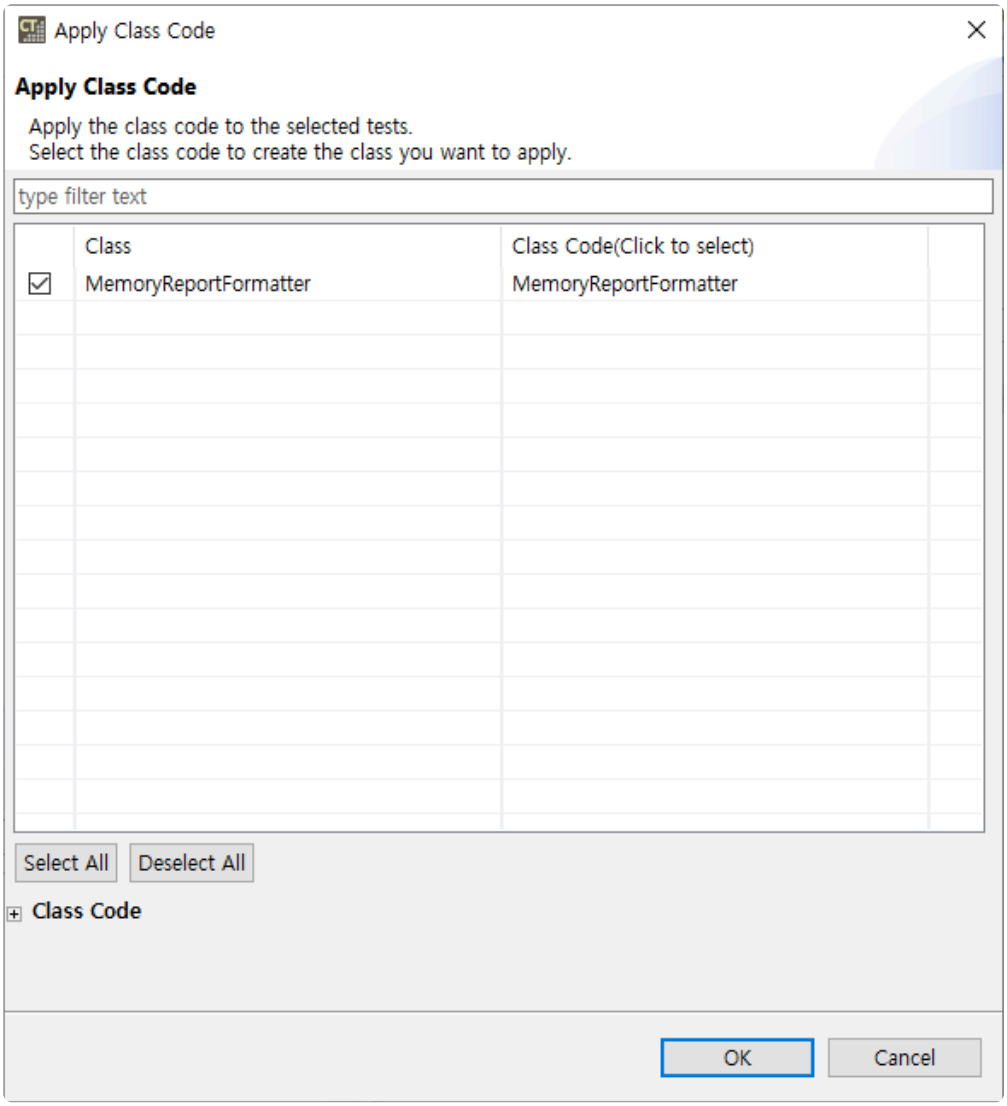


Apply using context menu in Test view

1. In Test view, select [Apply Class code] in context menu of functions or tests.



2. In [Apply Class code] dialog, select the class code to apply to selected tests and click [Ok]. Expand [Class code] at bottom to see the selected class code.



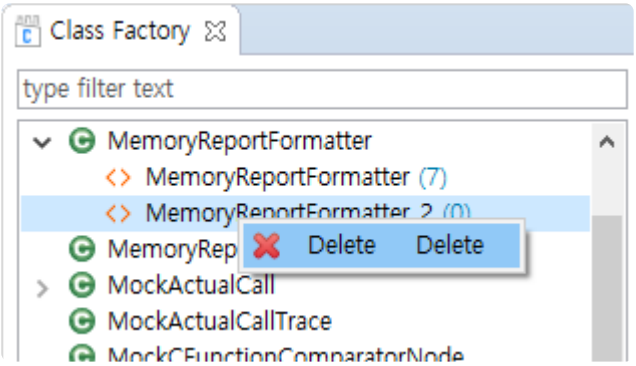
With [Apply Class code] feature, the same class code is applied wherever the class code can be applied.

Modify class code

You can modify the class code on [Class code] tab of Class Factory view. The modified class code is reflected in all connected tests.

Delete class code

You can delete class code in context menu of Class Factory view.



> ! It is impossible to delete the class code while it is connected to the test. Before deleting the class code, all connections with the test must be disconnected.

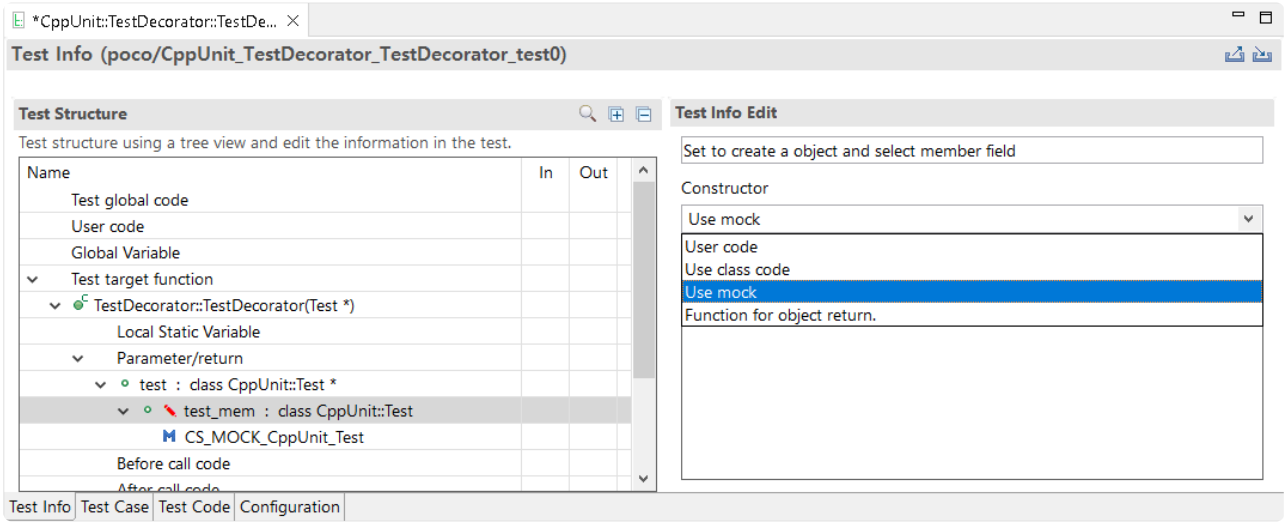
16.8.2. Using mock objects

The following toolchain is available for use with mock objects:

- GCC 6.0 or later
- Visual Studio 2015 or later

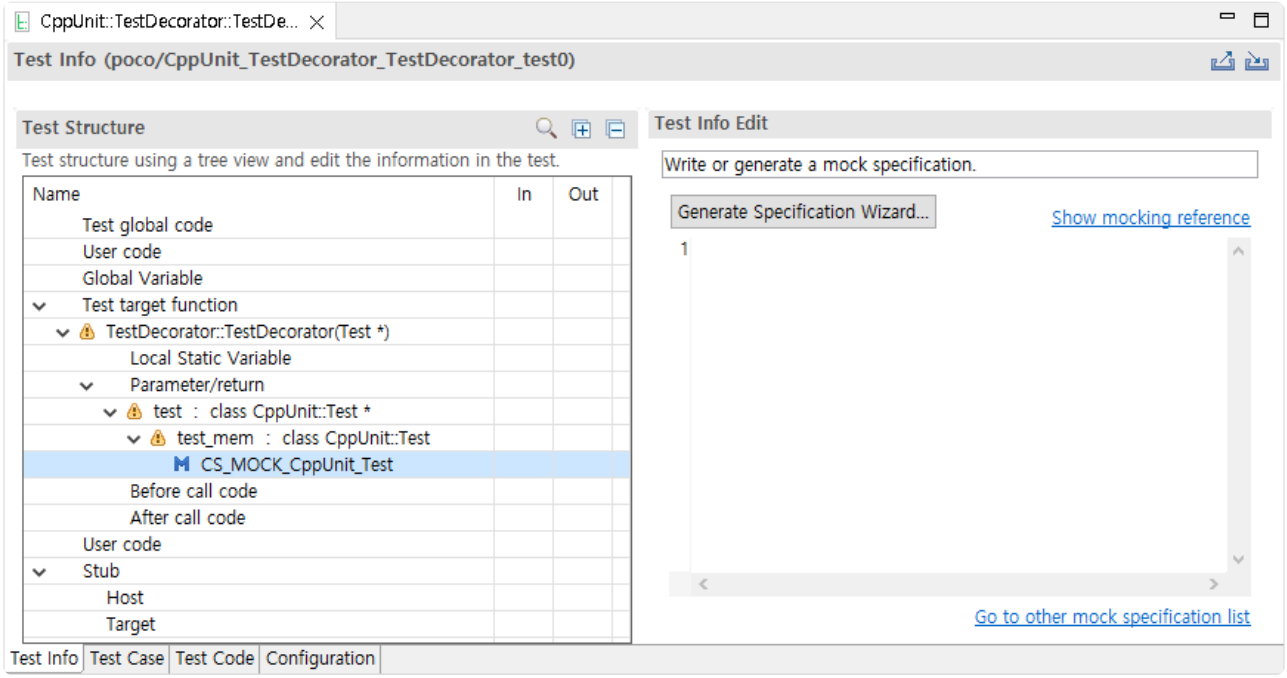
Creating and Using mock objects

1. Open [Test Editor] by double-clicking a test you want to create a mock object for.
2. In the Test Info Tab, expand the test structure tree and select the object to create a mock object. Select [Use mock] at the constructor in the Test Info Edit area on the right.



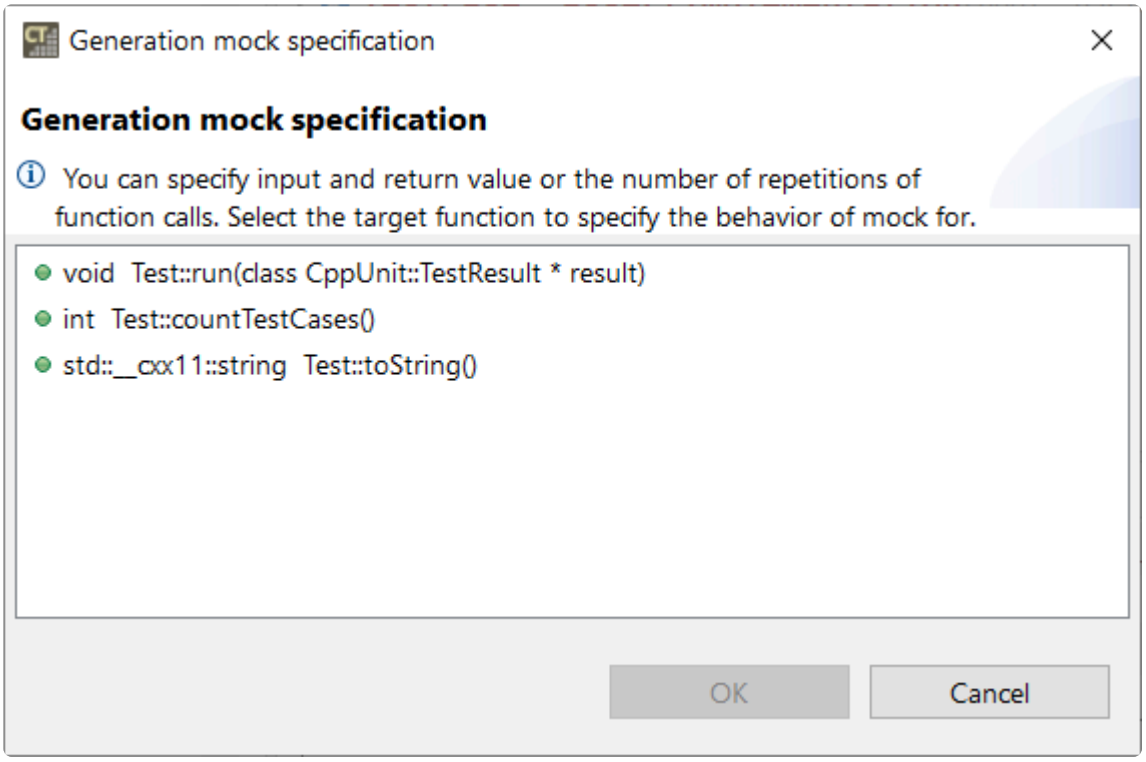
   - Selecting [Use mock], the class code of the mock object is generated in the Class Factory View.

3. Click the created mock object in the Test Info Tab.



   - [Show mocking reference]: You can see the description about specifications of mock objects.

4. You can generate mock specification in the wizard, by clicking [Generate Specification Wizard…] button in the Test Info Edit area.

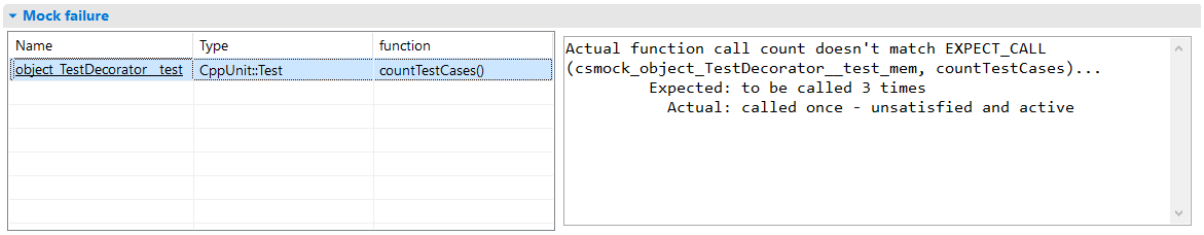


For details, refer to the Using Mock Objects in C++ Tests page of User Guides.

Check the result of mock objects.

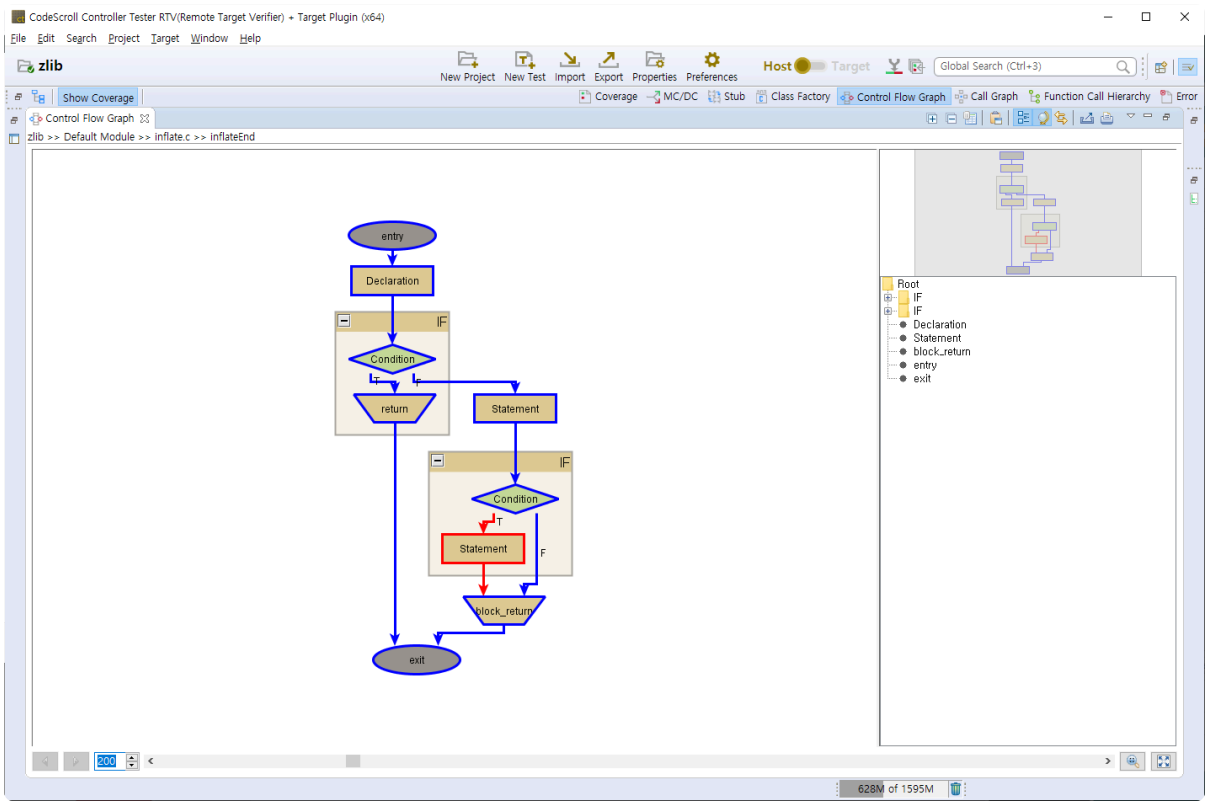
1. Click the Run button in the Test View.
2. If a mock object does not match the specifications, the test fails and displays the message [Mock Failed] next to the testcase.

3. You can check the failure information in [Test Editor] > [Test Case Tab].
   - Expand [▶ Mock failure] and check why the mock object failed.



16.9. Control Flow Graph View

The [Control Flow Graph] view shows the control flow information for the function selected in graph format.
If you click [Show Coverage] button in the main toolbar, the covered section is displayed in blue color and the uncovered section is displayed in red color.



Toolbar menu

Menu	Description
Unfold All	Shows all group nodes.
Fold All	Hide all group nodes.
Show Legend	Shows the legend(the node type in the currently displayed graph).
Copy to System Clipboard	Copies the currently displayed graph into clipboard.
Show Outline	Displays the graphs in a tree format.
Show Overview	Shows the overview of a graph.
Link with Editor	Shows the selected items in the graph in the editor by one-click.
Export View Content	Exports the contents of view in a report.
Print View Content	Prints the contents of view.

Pull-down menu

Menu	Description
Unfold All	Shows all group nodes.
Fold All	Hides all group nodes.
Show Call Graph	Shows the function call graph for the function selected in the current view.
Show Legend	Shows the legend(the node type in the currently displayed graph).
Show Outline	Displays the graphs in a tree format.
Show Overview	Shows the overview of a graph.
Link with Editor	Shows the selected items in the graph in the editor by one-click.
Save as Graph Format	Creates the graph model file for the graph displayed on the current screen. Four kinds of formats supported: • Graph Modeling Language XML (*.xgml) • Graph Modeling Language(*.gml) • yWorks Binary Graph Format(.ygf) • Trivial Graph Format(*.tgf)
Save as Image Format	Saves the currently displayed graph as an image format file(jpg, gif).
Copy to System Clipboard	Copies the currently displayed graph into clipboard.
Preferences	Opens the preferences.

Node Pop-up menu

Menu	Description
Show Call Graph	Shows the function call graph for the function selected in the current view.

History function

The node that had been selected in the current graph can be seen again by using the arrow button(Go back, Go forward) in the lower-left corner of the view.

Zoom out/in function

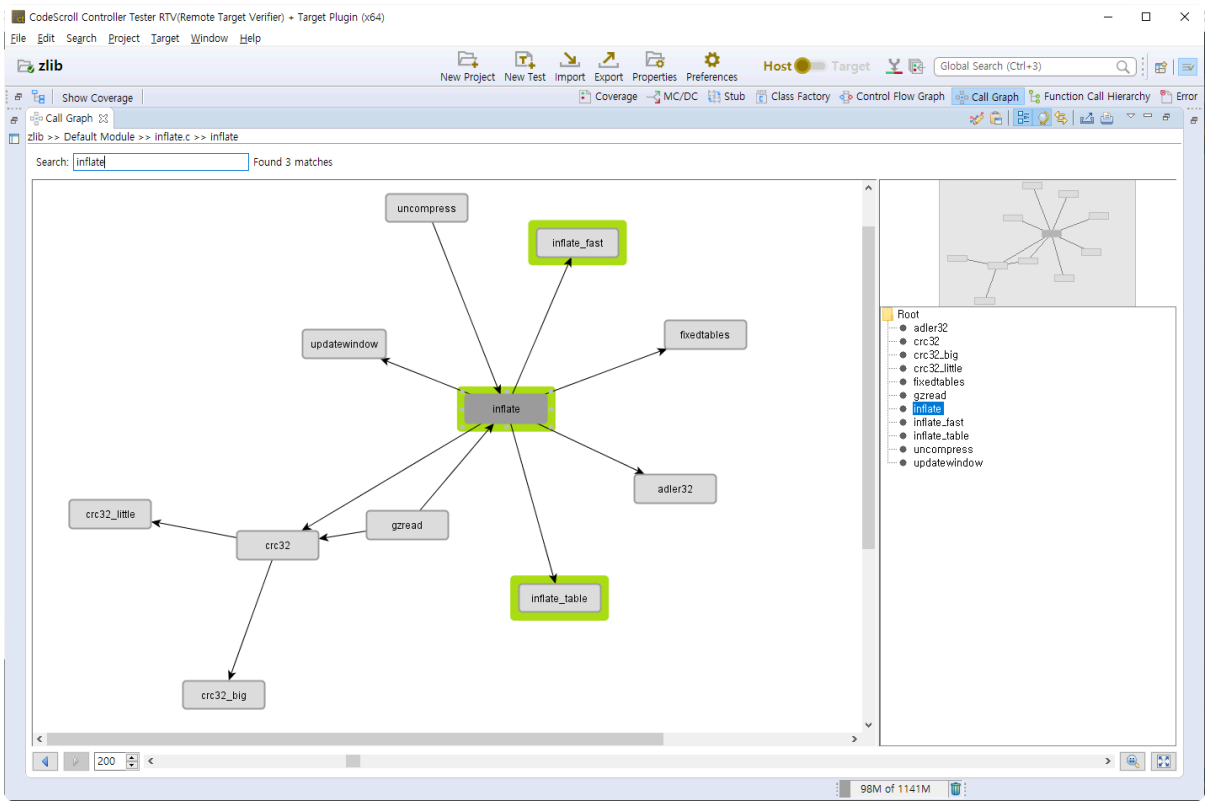
You can change the zoom out/in ratio by entering a number at the bottom of the view or by adjusting the slider.

You can reset the zoom out/in ratio by using [Initialize zoom out/in ratio] button at the lower right

corner of the view, and change the zoom out/in ratio accordingly to the view size by using [Fit to view size] button.

16.10. Call Graph View

The [Call Graph] view shows the function call information for the selected function in a report format. (Ex: When calling function ‘B’ from function ‘A’, it is expressed ‘edge from node ‘A’ to node ‘B’, and it is represented as one edge even if called multiple times.
If you select the node (function), it shows the function call information for that node.



Toolbar menu

Menu	Description
Edit Layout	Changes to make the node position of graph modifiable(it cannot be automatically edited when the graph is updated).
Copy to System Clipboard	Copies the currently displayed graph into clipboard.
Show Outline	Displays the graphs in a tree format.
Show Overview	Shows the overview of a graph.
Link with Editor	Shows the selected items in the graph in the editor by one-click.
Export View Content	Exports the contents of view in a report.
Print View Content	Prints the contents of view.

Pull-down menu

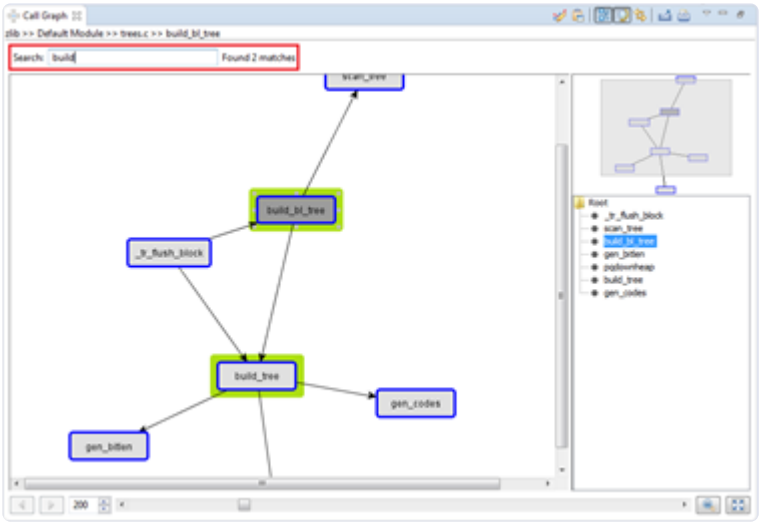
Menu	Description
Show CFG	Shows the control flow graph for the function selected in the current view.
Show Outline	Displays the graphs in a tree format.
Show Overview	Shows the overview of a graph.
Link with Editor	Shows the selected items in the graph in the editor by one-click.
Save as Graph Format	Creates the graph model file for the graph displayed in the current screen. Four kinds of formats supported: • Graph Modeling Language XML (*.xgml) • Graph Modeling Language(*.gml) • yWorks Binary Graph Format(.ygf) • Trivial Graph Format(*.tgf)
Save as Image Format	Saves the currently displayed graph as an image format file(jpg, gif).
Copy to System Clipboard	Copies the currently displayed graph into clipboard.
Preferences	Opens the preferences.

Node Pop-up menu

Menu	Description
Show CFG	Shows the control flow graph for the function node selected.
Expand	Shows the call relationship after the selected function one more step.
Collapse	Hides the call relationship after the selected function.
Set to Start Node	Specifies the start function for checking the call path for two functions.
Set to End Node	Specifies the end function for checking the call path for two functions. Shows all paths possible in the current graph.

Search function

When you enter a search word in the top of the view, the functions with the name containing the entered search word are highlighted.

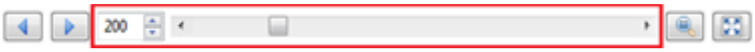


History function

The node that had been selected in the current graph can be seen again by using the arrow button(Go back, Go forward) in the lower-left corner of the view.

Zoom out/in function

You can change the zoom out/in ratio by entering a number at the bottom of the view or by adjusting the slider.



You can reset the zoom out/in ratio by using [Initialize zoom out/in ratio] button at the lower right corner of the view, and change the zoom out/in ratio accordingly to the view size by using [Fit to view size] button.

Show/Hide function

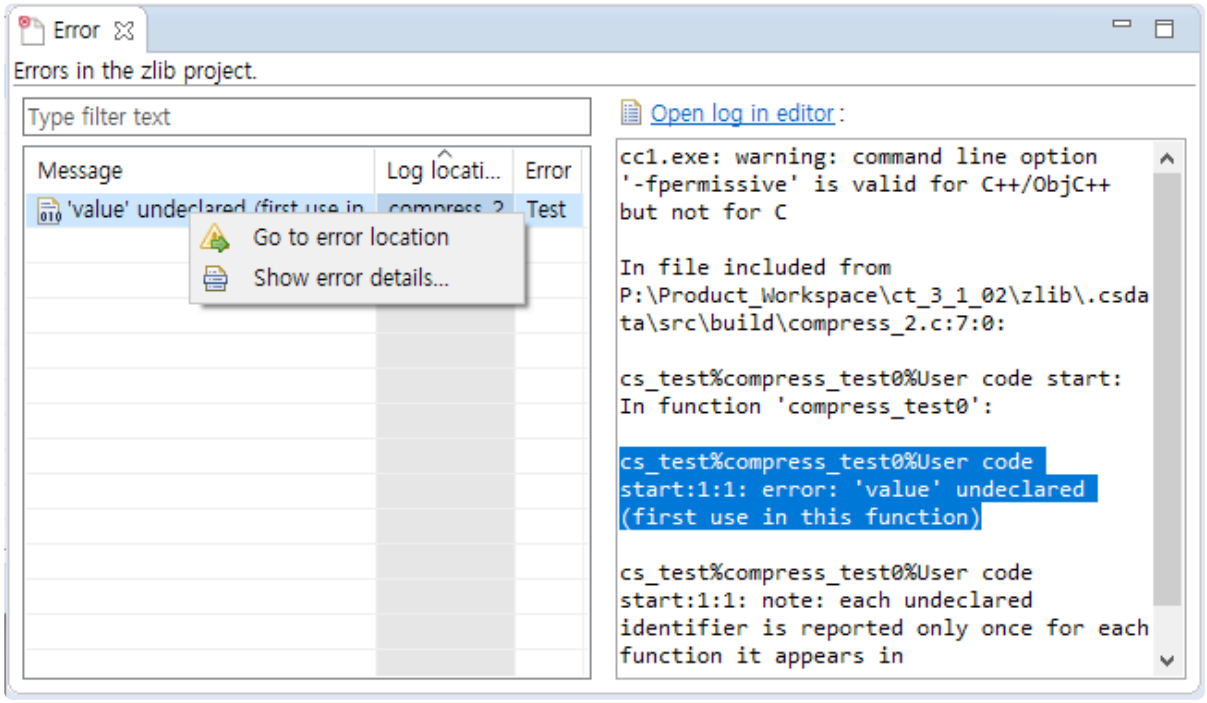
By using the menu displayed when right-clicking a node, you can show or hide the edge out from that node. If you select [Hide], it hides the call relationship for all functions called by the function corresponding to the selected node. The Hide menu of the node having no function call information is disabled. If you select [Show], it shows the functions called by the function corresponding to the selected node one more step. Likewise, if there are no call relationships, the Show menu is disabled.

Highlight Path between Nodes function

By selecting the start node and end node, it highlights all paths exited between two nodes so that you can see easily the interesting section in the complex function call relationship.

16.11. Error View

You can check the analysis error information of the project in detail.



Log type

Type	Description
Build	Errors occurred during build
Statement analysis	Errors occurred during statement analysis
Link	Errors occurred during linking
Pre-processing	Errors occurred during pre-processing

Error type

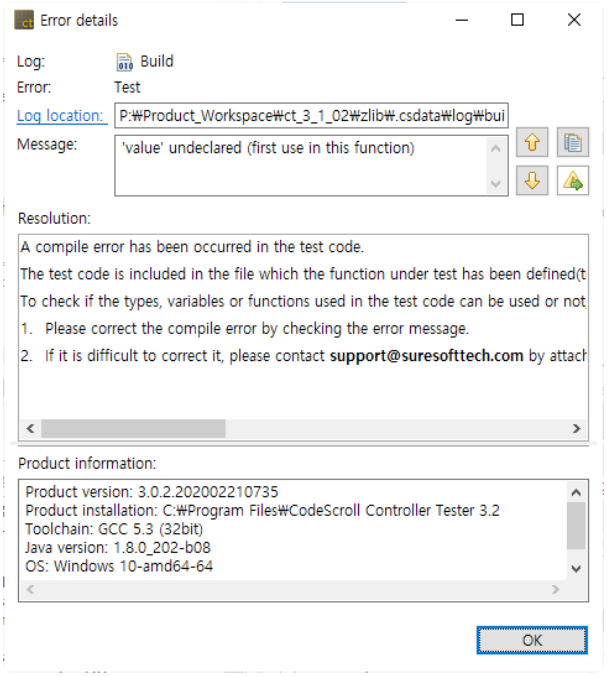
Type	Description
Source	Errors occurred int the source file
Stub	Errors occurred in the stub file
Test	Errors occurred in the test editor

Go to the error location

In the context menu, select [Go to error location] to go to the location where the error has occurred so as to correct the error easily.

Show error details

Double-click the error information or select [Show error details…] in the context menu to check the detail information for the error.



Buttons

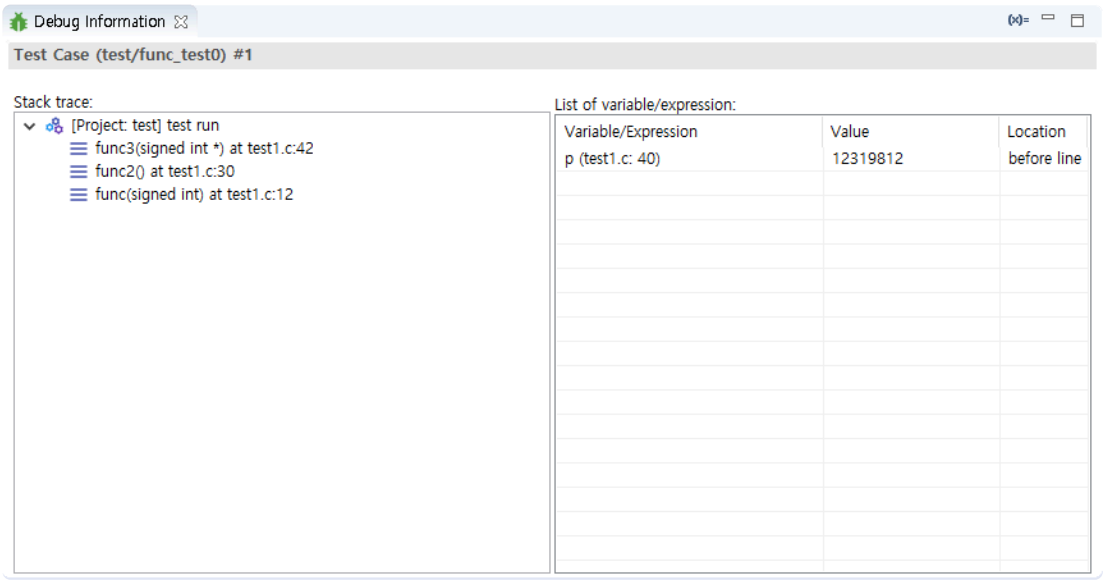
Button	Description
Previous	Shows the previous error information.
Next	Shows the next error information.
Copy	Copies the error information.
Go to Error	Points to the location of the error.

16.12. Debug Information View

The debug information view shows information to help you determine the cause of an error in a test case.

- Trace function calls in test cases with errors
- Actual value of variable/expression added to inspect debug information

If you select a test case that has [Inspect Debug Info] in the unit test view, debug information is displayed in the Debug Information view.



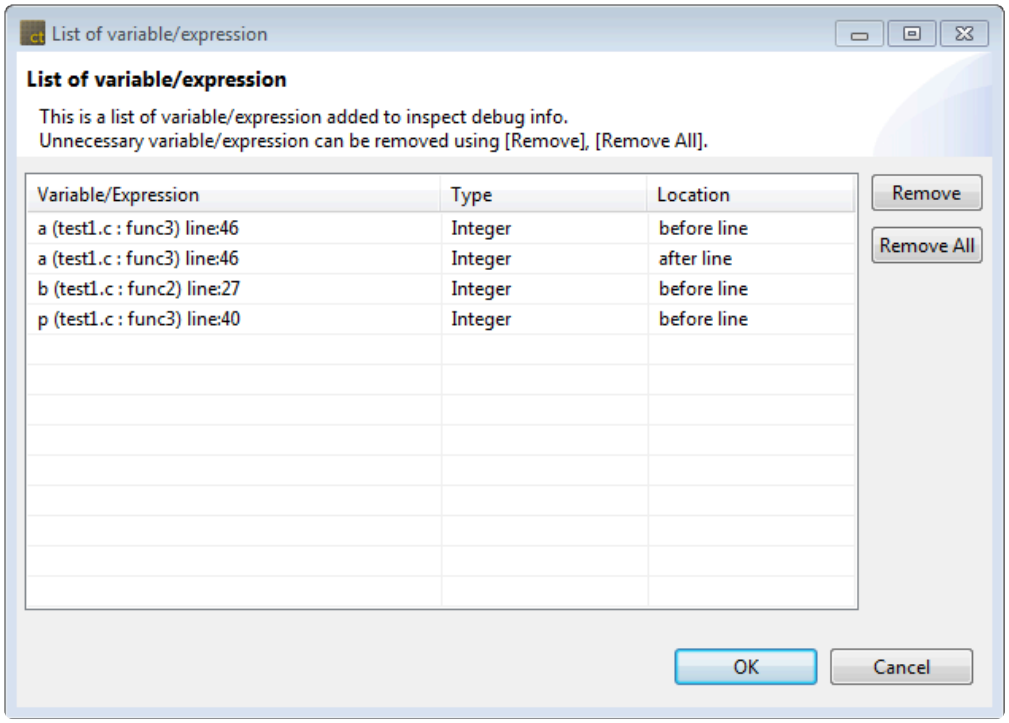
- The Stack trace shows the function call trace when the test case in error was executed. The location where the error occurred is displayed at the top.
- The variable/expression list shows the actual value of the variable/expression added to inspect the debug information.

> ! If the actual value of the added variable/expression is not displayed:
> - When [Inspect Debug Info] of the test case is not executed after adding a new variable/expression
> - When the line where the variable/expression was added was not executed when the test case was executed (it was not executed depending on the condition or an error occurred before execution)

Debug Information View toolbar menu

icon	description
(x)=	List of variable/expression to debug

In the list of debug variable/expression, you can check and remove variable/expression.

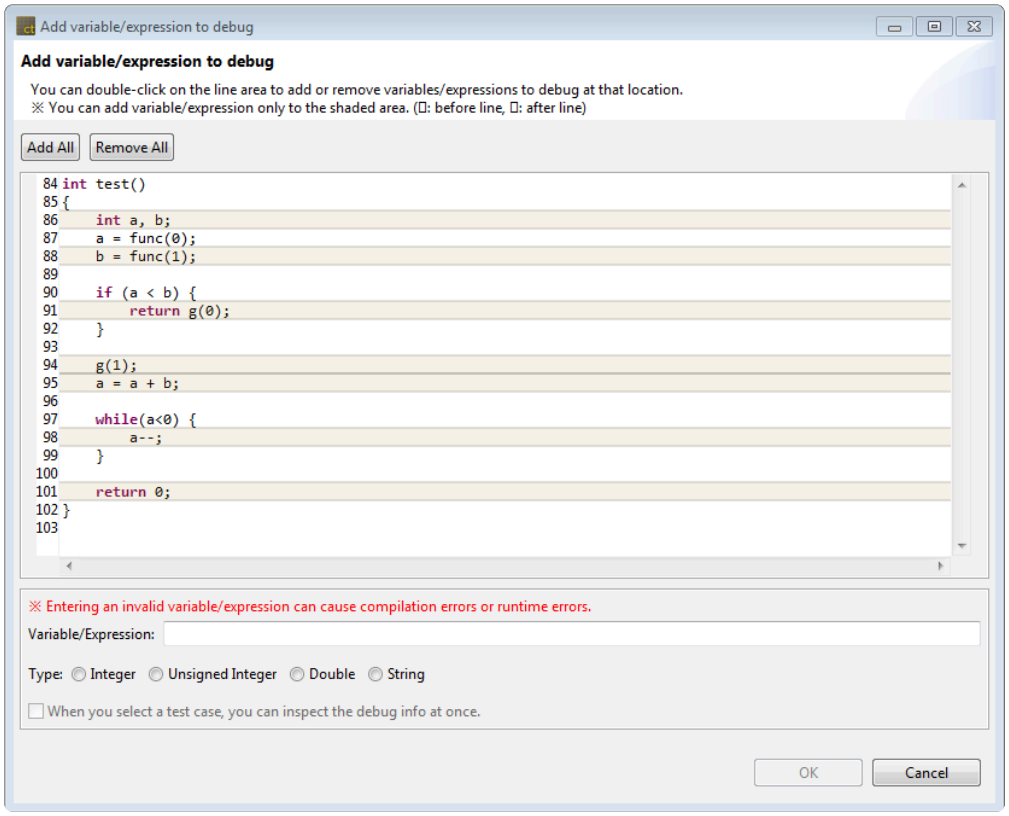


Adding variable/expression to debug

You can add variable/expression to debug in the following ways.

1. Double-click the line area in the source code editor
2. Select [Add Variable/expression to debug…] from the line area context menu in the source code editor
3. After specifying a variable or expression in the source code editor, select [Add Variable/expression to debug…] from the context menu.

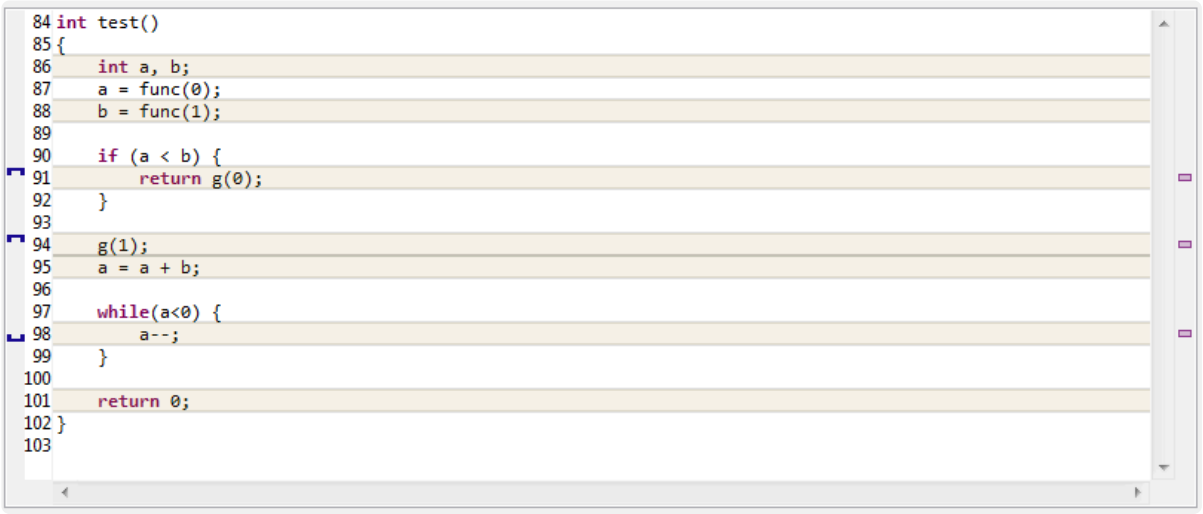
When the above operation is performed, the [Add variable/expression to debug] window appears.



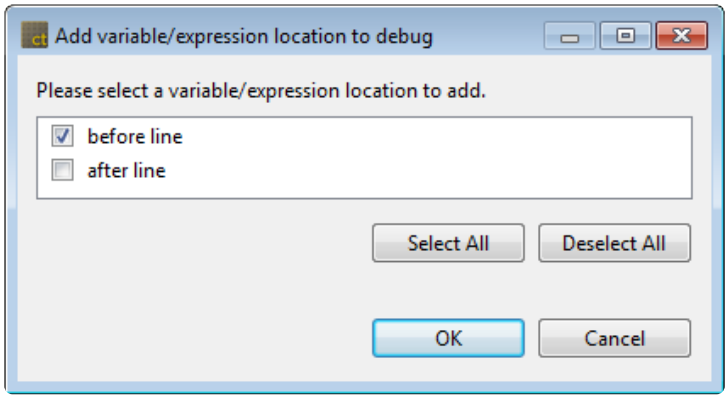
You can double-click on the line area to add or remove variable/expression to debug at that location.

marker	description
⌐	Variable/expression added before the line
⌙	Variable/expression added after the line

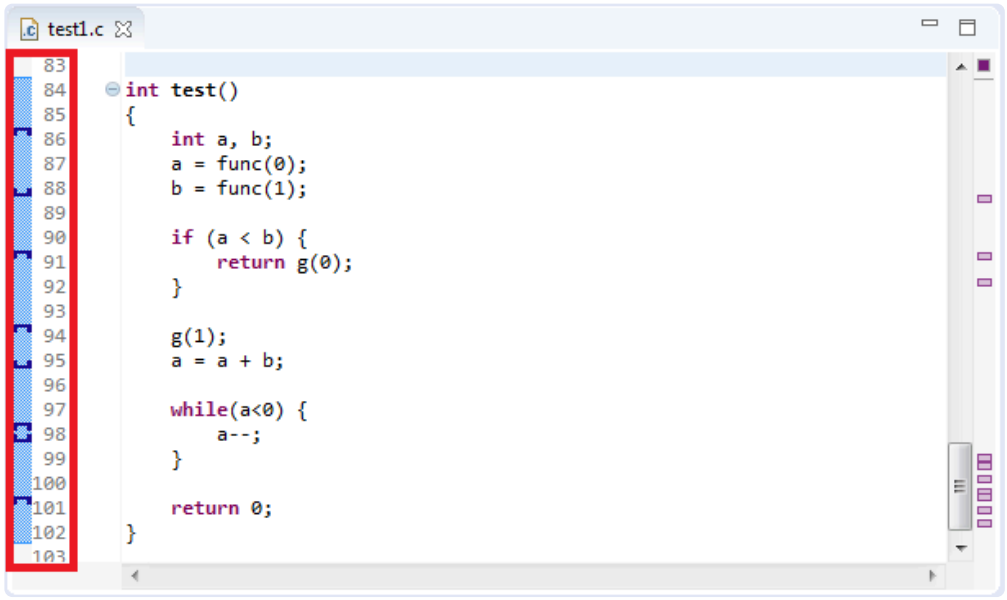
You can add markers only in the shaded area.



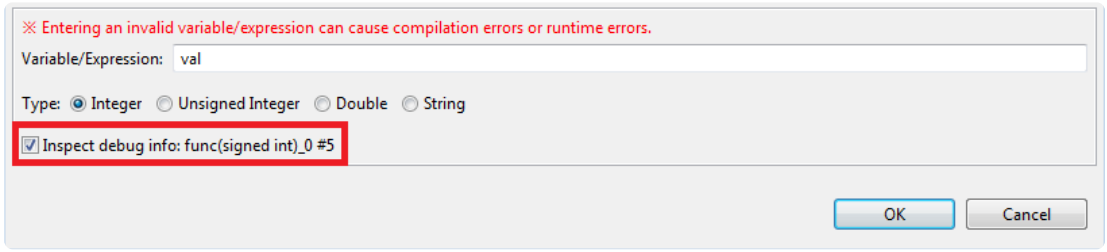
If you can add a variable/expression to debug both before and after the line in the shaded area, a dialog will pop up asking where to add it.



You can add or remove variables/expressions anywhere you can add to the target function through the [Add All], [Remove All] buttons at the top of the source viewer.
After specifying the variable/expression and type and clicking the [OK] button, you can see that the marker has been added to the source code editor.



If the test case is selected in [Unit Test View], you can immediately perform [Inspect Debug Info] while adding a variable/expression to be debug.

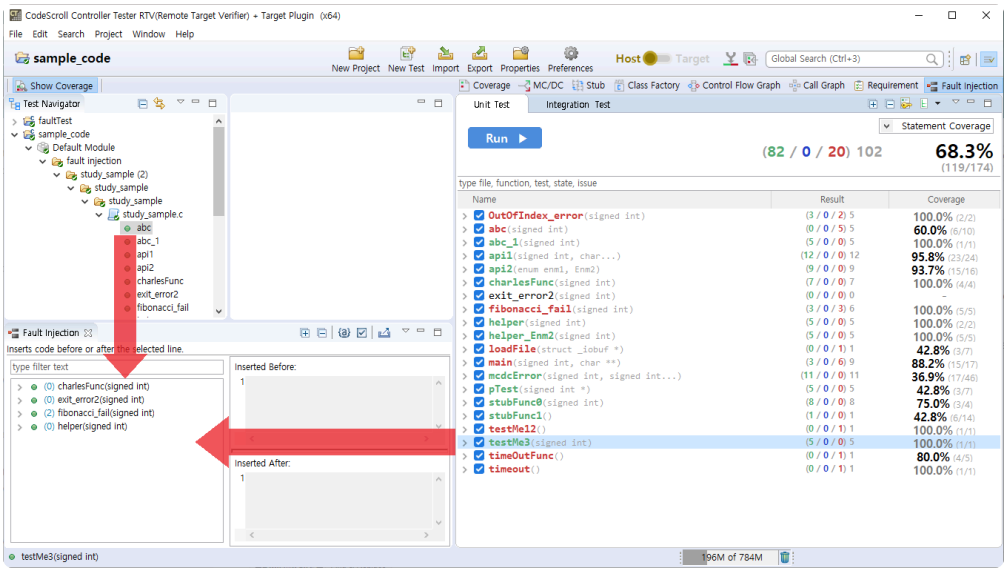


> ! If you enter an invalid variable/expression, a compilation error or a runtime error may occur when performing [Inspect Debug Info].

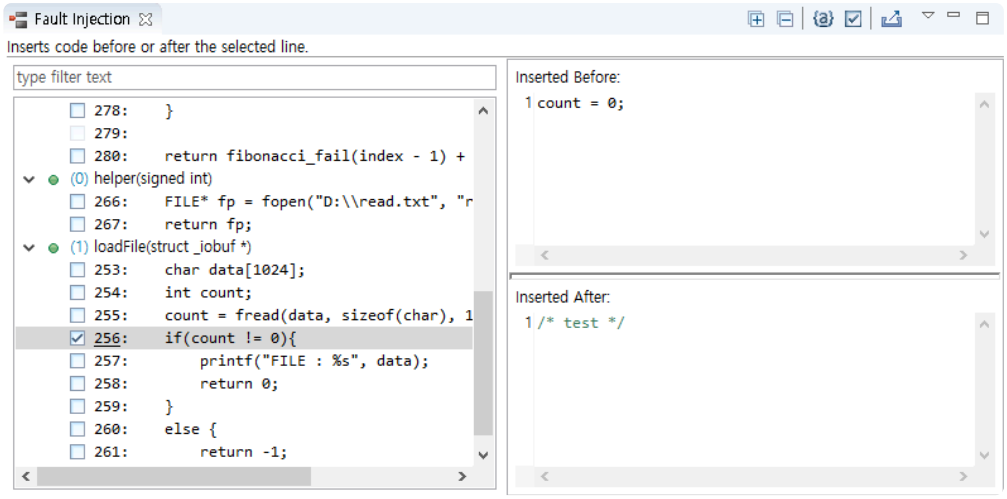
16.13. Fault Injection View

The Fault Injection View provides the feature for inserting the code needed to test additionally into the specific area of the function to be tested.

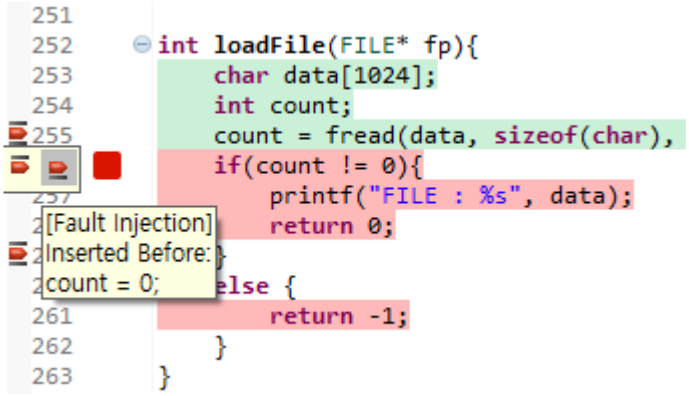
1. Drag and drop the function that you want to inject a fault from the Test Navigator View or the Unit Test View to the Fault Injection View.



2. In the tree structure on the left of the view, click the line that you want to inject a fault and, on the right of the view, enter the user code to insert before/after the line. The number of the line where user code is written is underlined.



- When check checkboxes of lines to insert, the user codes are applied. In the source code editor, (before the line), (after the line) markers appear at that lines. When mouse is on the markers, the tooltip about fault injection information and written code appears.



> ! Single-line comments (//) and input/output user macros cannot be used.

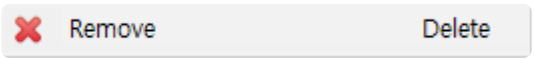
Toolbar menu in the Fault Injection view

Toolbar icon	Description
Expand All	Expand all function trees in Fault Injection view.
Collapse All	Collapse all function trees in Fault Injection view.
Show Only Non-empty Fault Injections	View only the line where the fault injection code is written among the fault injection lines.
Show Only Enable Fault Injections	View only active lines among fault injection lines.
Import Fault Injections	Import fault injection information with .xls format.
Export Fault Injections	Export fault injection information with .xls format.

Context menu in the Fault Injection view

Function nodes and lines can be multi-selected to bring up the context menu.

Context menu of function nodes



Menu	Description
Remove	Remove the function inserted by users.

Context menu of lines

Menu	Description
Copy	Copy the information inserted by user before/after the line.
Paste	Paste the information inserted by user before/after the line.
Clean	Uncheck the checkboxes of the selected lines and remove inserted code.
Enable	Check the checkboxes of the selected lines and activate fault injection.
Disable	Uncheck the checkboxes of the selected lines and deactivate fault injection.

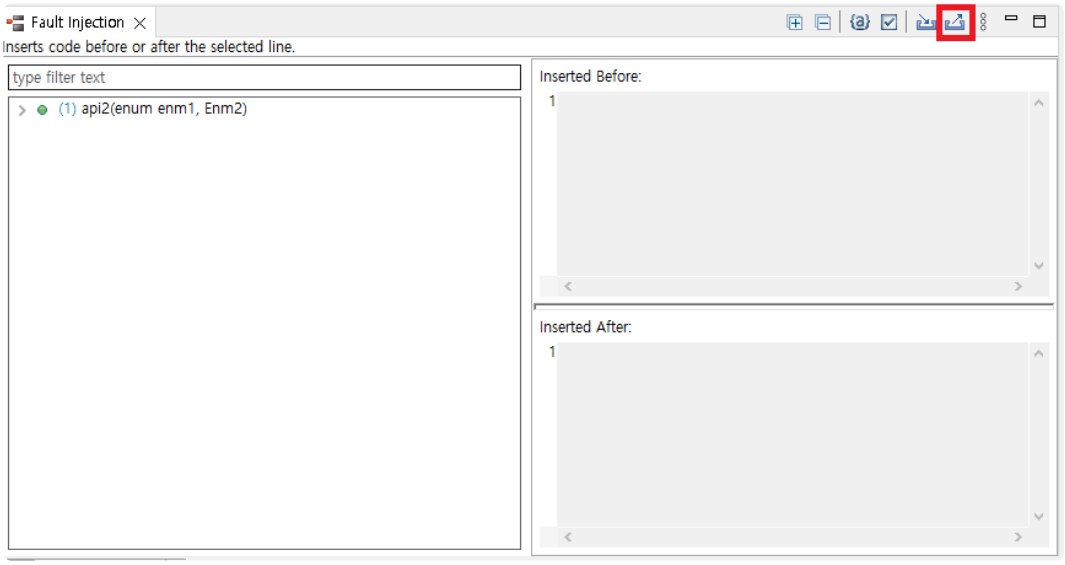
Pull-down menu

Menu	Description
Reconfiguring Fault Injection	Note the Reconfiguring Fault Injection

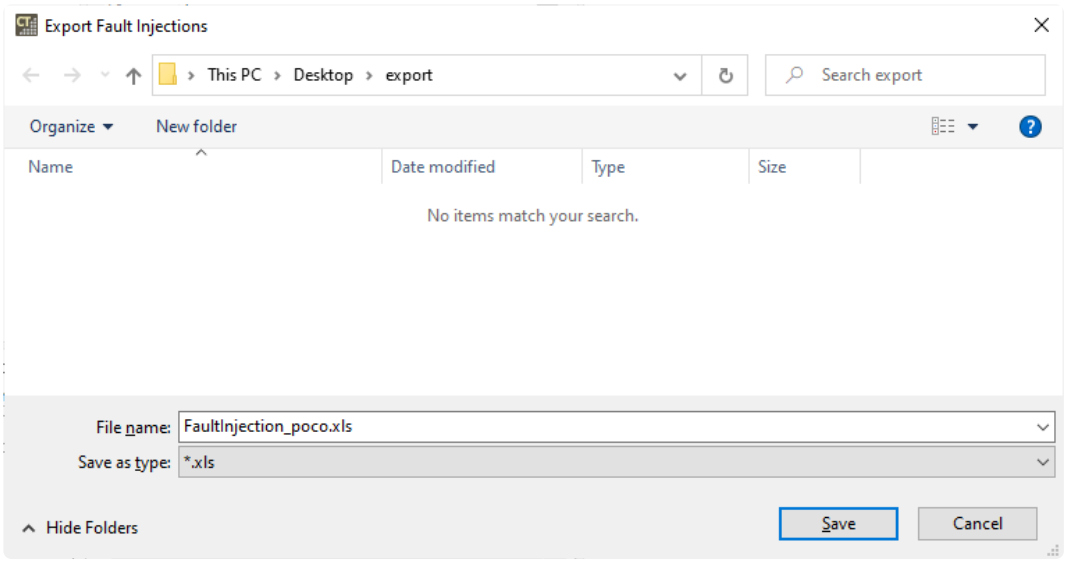
> When modifying the source code, contents of the fault injection may not be properly reflected. In case of modification, contents of the fault injection must be rewritten.

Export Fault Injections

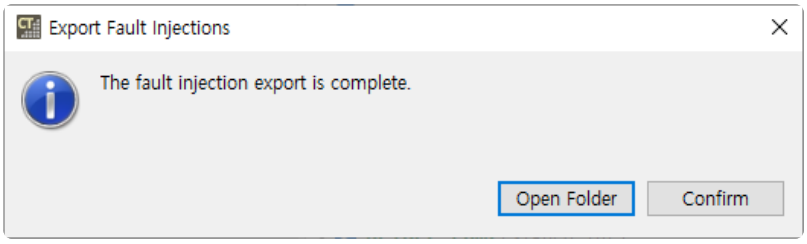
1. Select the [Export Fault Injections] icon from the toolbar menu.



2. In the directory dialog, specify the save location for the fault injection file (.xls) and then select the [Save] button.

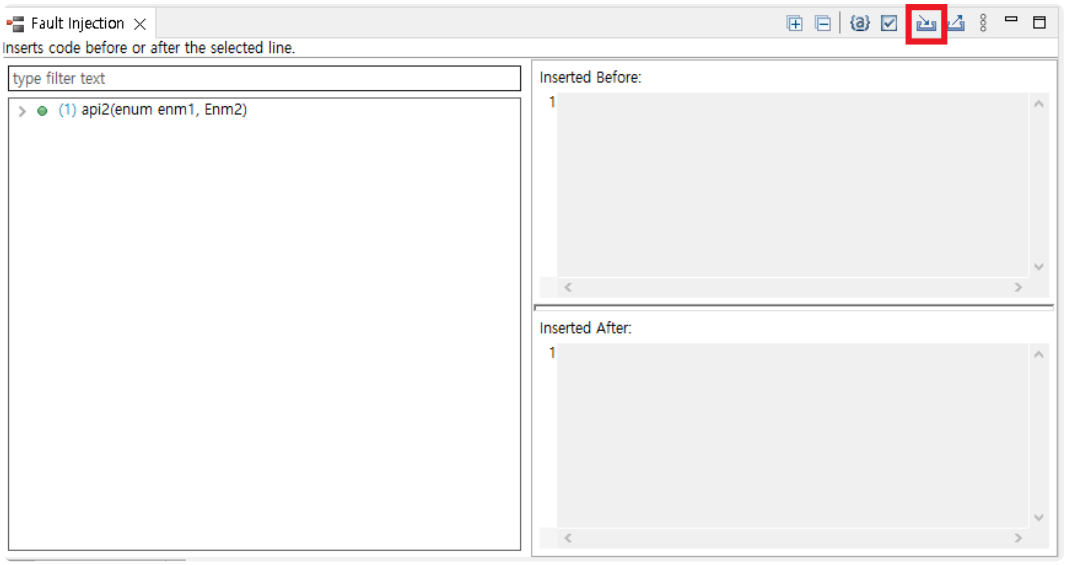


3. If the export is successful, you can go directly to that path location.

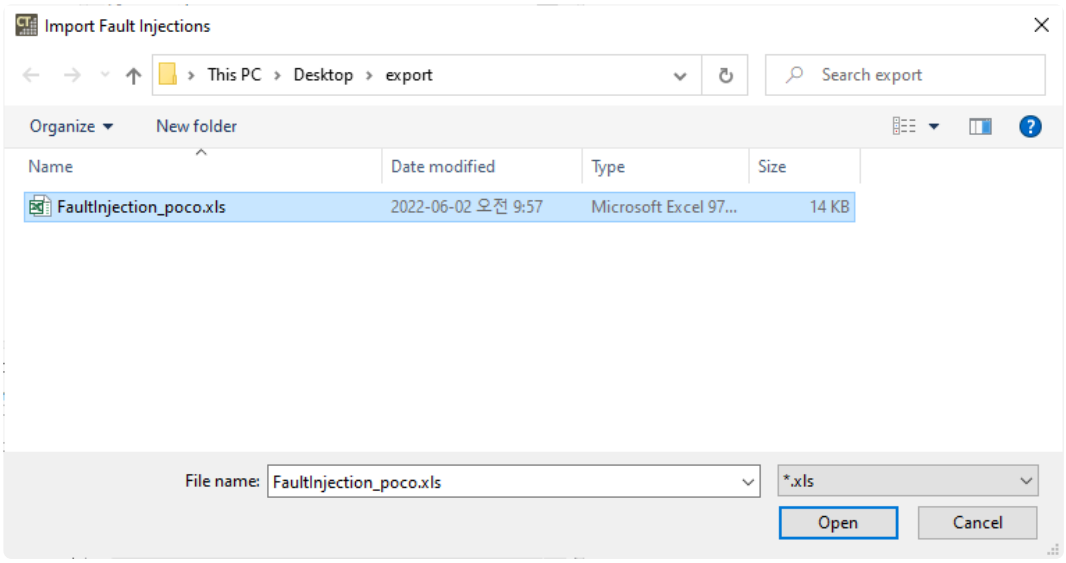


Import Fault Injections

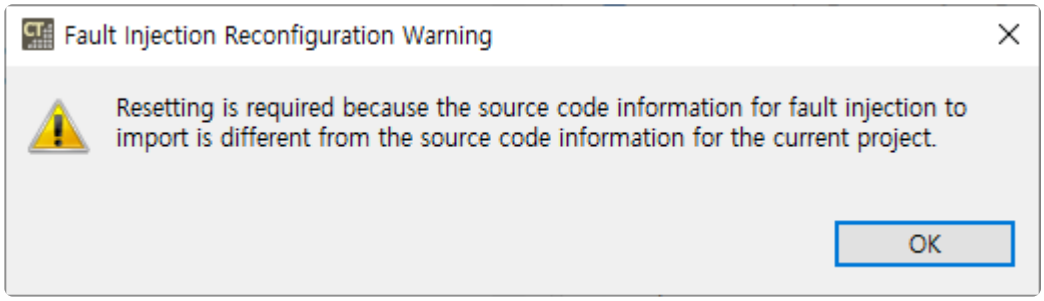
1. Select the [Import Fault Injections] icon from the toolbar menu.



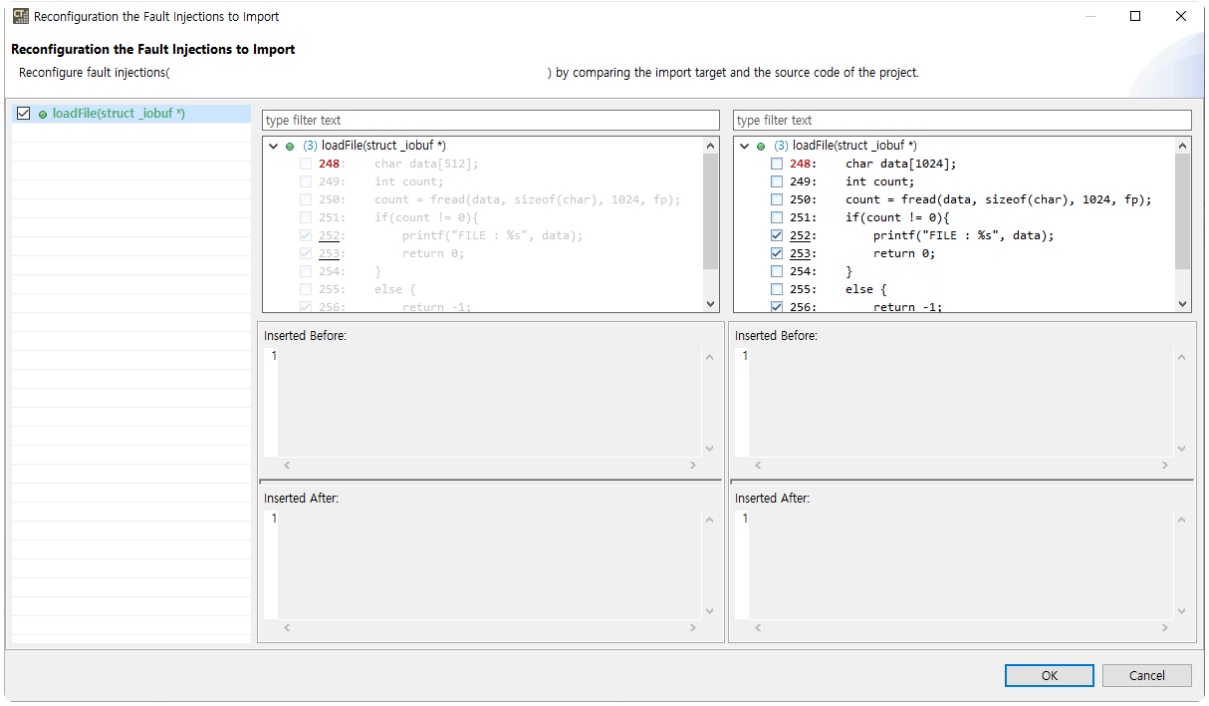
2. Select the fault injection file to import.



3. If the source code of the fault injection information you want to import is different from the source code of the Controller Tester project, fault injection can be reused to match the changed source code using [Reconfiguration the fault injection to import] feature.
    a. If the source code shape is different when performing defect injection import, the following dialog will appear.



    b. You can reconfigure the import of fault injection by clicking the [OK] button.

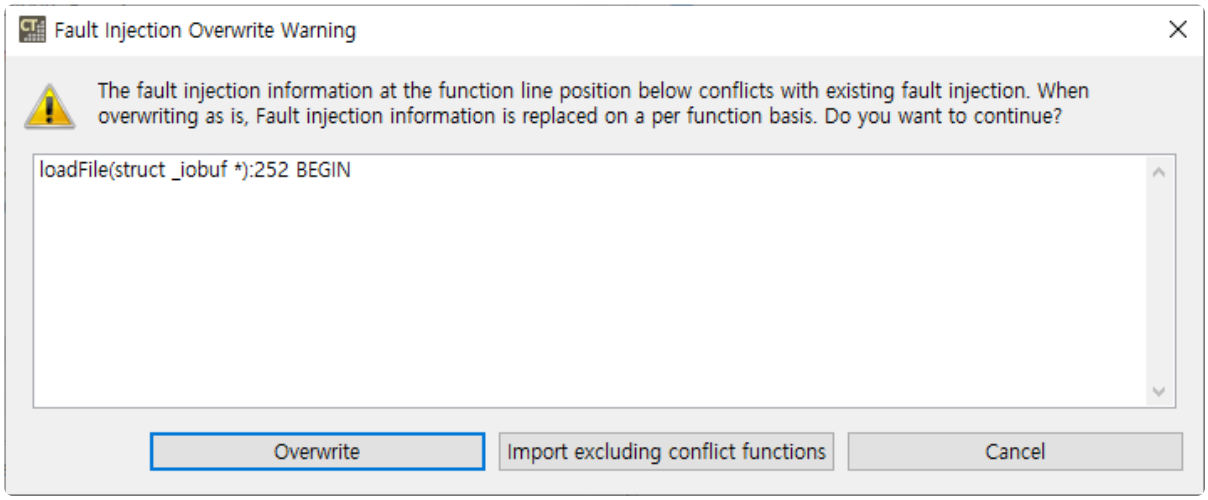


- On the left, the source code of the fault injection information you want to import is displayed.
- On the right, the source code of the current project is displayed.

c. If you click the [OK] button in the [Reconfiguration the Fault Injections to Import] window, you can import the reconfigured fault injection information.

> If the file and function information of the fault injection information you want to import are different from the project, reconfigure the function is performed first.

4. If you have fault injections already created in the same location as the fault injection information you want to import into your project, you need to decide whether to overwrite the fault injection.
   a. When importing fault injection, if there is fault injection information in the existing location, the following dialog will appear.

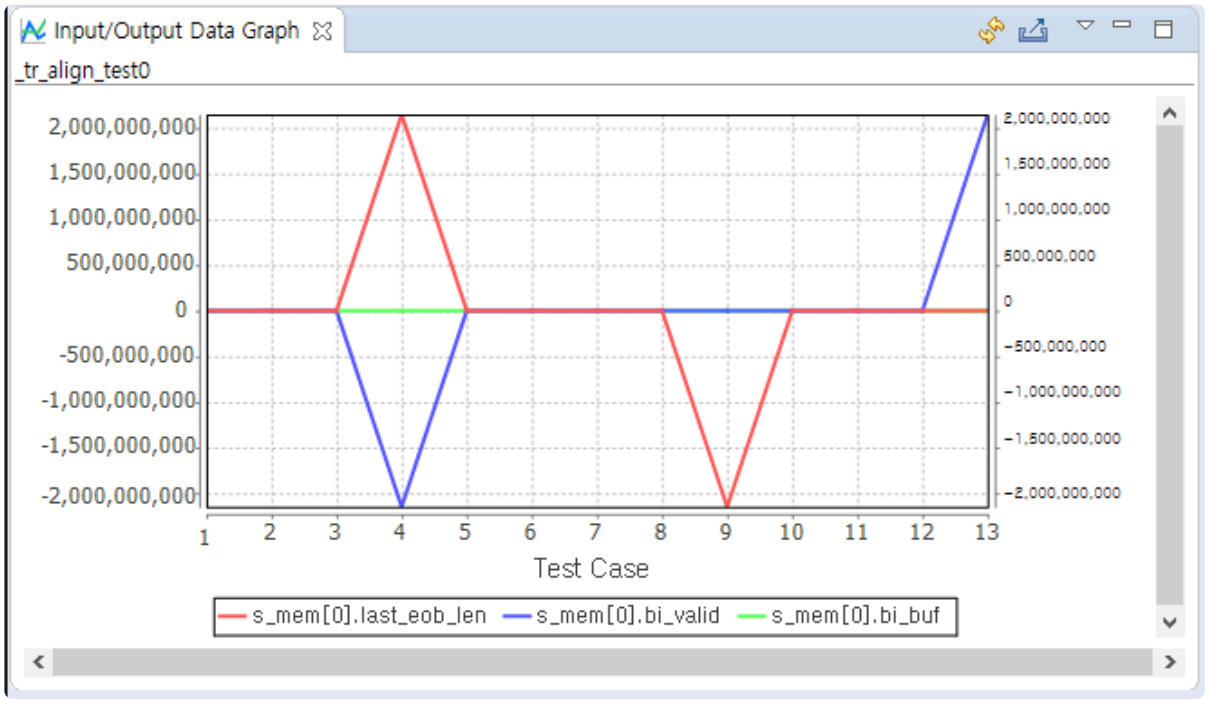


   - The [Fault Injection Overwrite Warning] dialog shows the function, line, and location of the fault injection that does not match the existing information.
   - If you select [Overwrite], the existing fault injection information will be overwritten with the fault injection information you want to import.
   - If [Import excluding conflict functions] is selected, only fault injection information for functions except for conflict functions is imported.
   - Clicking the [Cancel] button cancels the import of fault injections.

> Before Controller Tester 3.7, only lines with fault injection code are imported as enabled, as the exported fault injection file did not include enabled or disabled.

16.14. Input/output Data Graph View

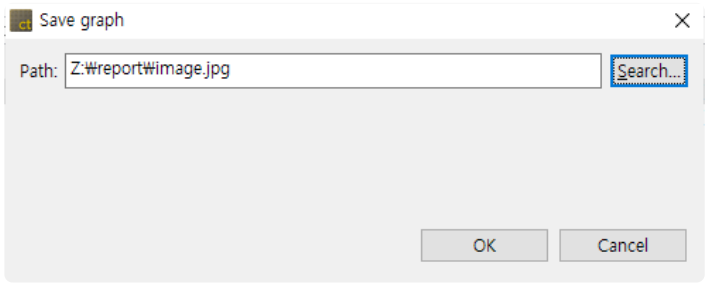
The Controller Tester provides the data for input value/expected value/output value in graph format. The horizontal axis of the graph indicates the number of the test case and the vertical axis indicates the data of the test case.



Toolbar icon in Input/output Data Graph view

Toolbar icon	Description
Refresh	Refreshes the input/output data.
Save graph	Saves the input/output data.

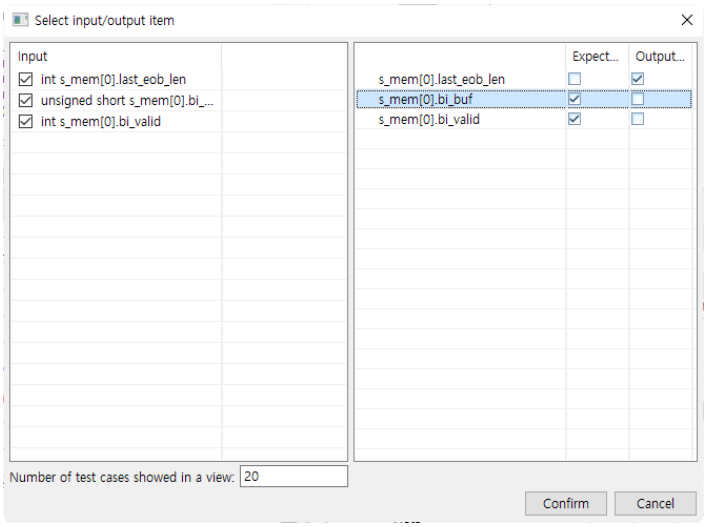
Click [Save graph] to display the notification window that can enter the path for saving the graph. Enter the path to save, the file name and the file format and click [OK] to save the graph.



Graph settings

Select [Select input/output item] in the pull-down menu to display the Graph settings window.

- For the items selected in the checkbox, the [Input], [Output value] and [Expected value] are displayed in the Input/output Data Graph. However, if the expected values are specified as ~, &, | and ! etc., the expected value is not applied to the Input/output Data Graph view.
- [Number of test cases showed in a view] allows you to specify the number of test cases to be displayed on one screen.



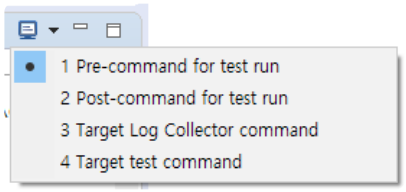
16.15. Console View

Enter the instruction or batch file in [Pre-command for test run] and [Post-command for test run] of [Project] -> [Properties] -> [Test] -> [External Command] and execute the unit test to display the execution result for the external instructions before and after the test execution in the [Console] view.

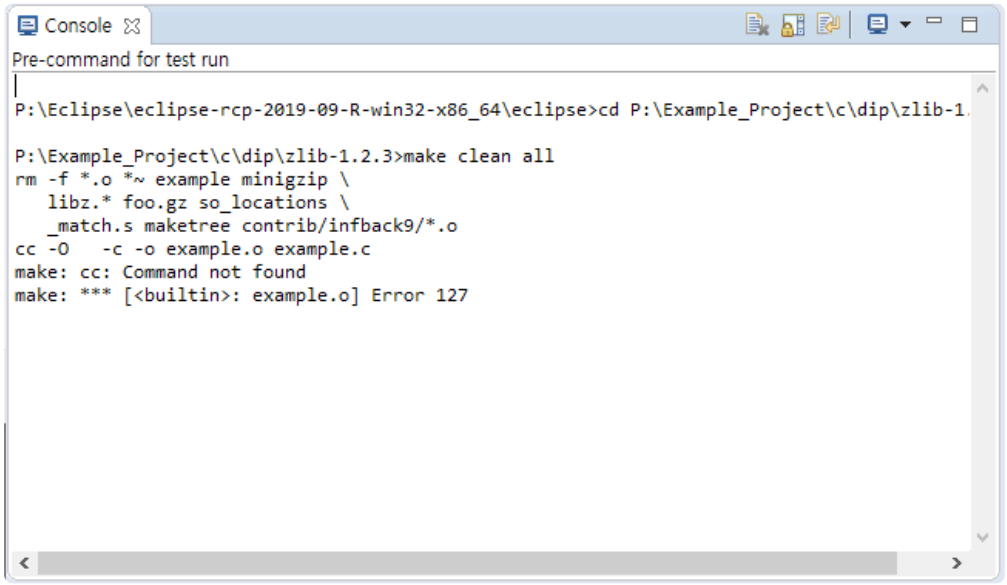
Toolbar icon in the [Console] view

Toolbar icon	Description
Clear Console	Clears the console contents.
Scroll Lock	Locks the scrolling of the [Console] view screen.
Display Selected Console	Shows the selected console.

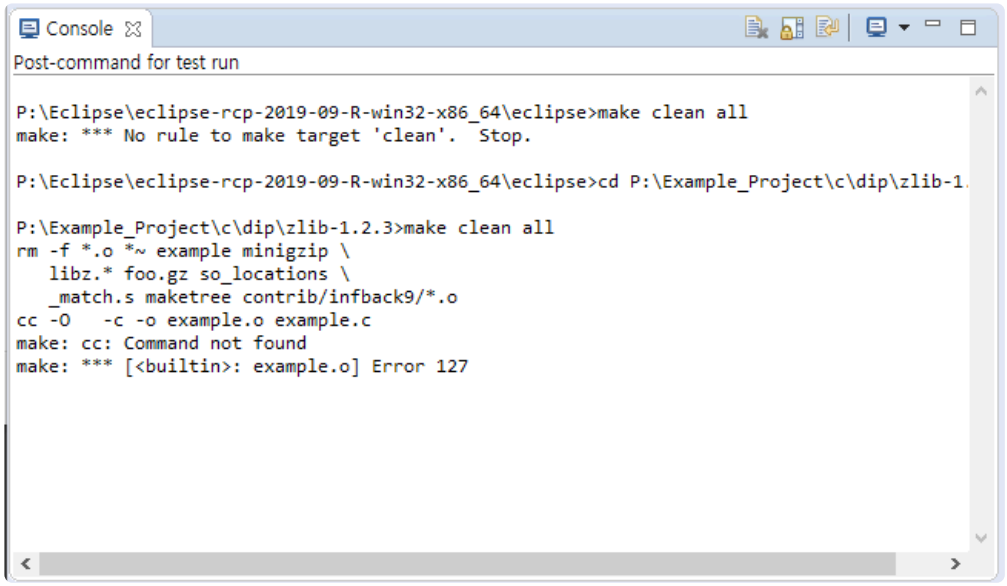
If [Display Selected Console] is selected, the following two menus are displayed: [Pre-command for test run] and [Post-command for test run].



If you select [Pre-command for test run], the execution result for the external command before running the test is displayed in the Console view.

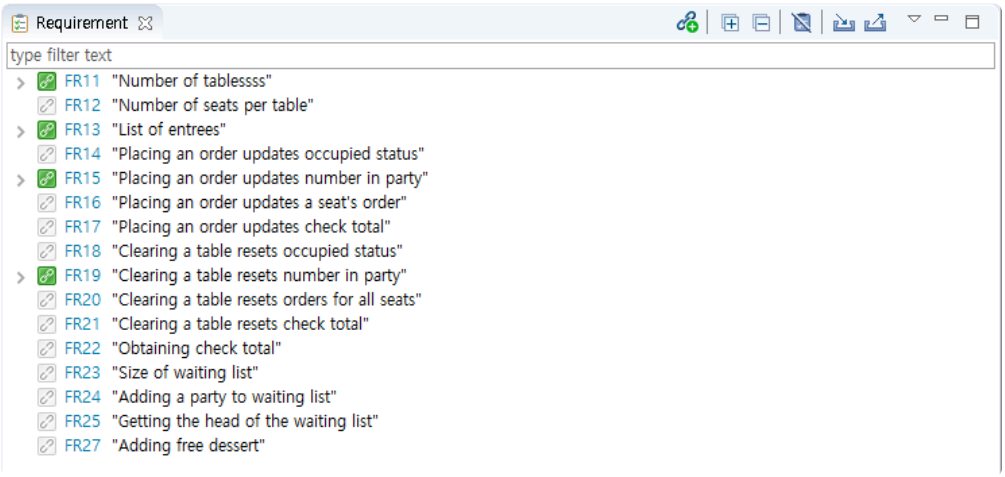


If you select [Post-command for test run], the execution result for the external instructions after the test execution is displayed in the Console view.


Console
Post-command for test run
P:\Eclipse\eclipse-rcp-2019-09-R-win32-x86_64\eclipse>make clean all
make: *** No rule to make target 'clean'. Stop.
P:\Eclipse\eclipse-rcp-2019-09-R-win32-x86_64\eclipse>cd P:\Example_Project\c\dip\zlib-1
P:\Example_Project\c\dip\zlib-1.2.3>make clean all
rm -f *.o *~ example minigzip \
libz.* foo.gz so_locations \
_match.s maketree contrib/infback9/*.o
cc -O   -c -o example.o example.c
make: cc: Command not found
make: *** [<builtin>: example.o] Error 127

16.16. Requirement View

You can import requirements from a requirements management tool or a CSV file into Controller Tester. You can connect imported requirements with tests and export the connected information.



Requirement status

Status	Description
Not connected	A requirement without connected tests
Connected	A requirement with connected tests
Needs Review	A requirement that need to be reviewed

Toolbar Menu in Requirement View

Icon	Description
Connect requirements and tests	Open the [Connect requirements and tests] dialog box.
Expand All	Expand all requirements
Collapse All	Collapse all requirements
Hide unconnected requirements	Hide unconnected requirements.
Import	Importing requirements created with a CSV file or a requirements management tool.
Export	Export requirements to a requirements test coverage report or requirements management tool.

For more information, please refer to the pages Export, Import.

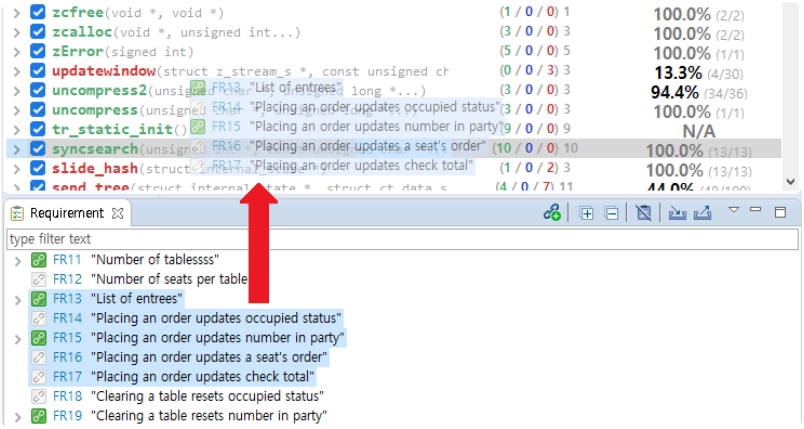
pull-down menu in Requirement View

Icon	Description
Export requirements traceability (csv)…	Exporting connection information between requirements and tests.
Export requirements traceability [Including all test cases] (csv)…	Export connection information between requirements and tests, including all test cases.
Import requirements traceability	Importing connection information between requirements and tests.
Export Requirements Traceability Information for V-SPICE	Export connection information between requirements and tests in V-SPICE's xml file format.

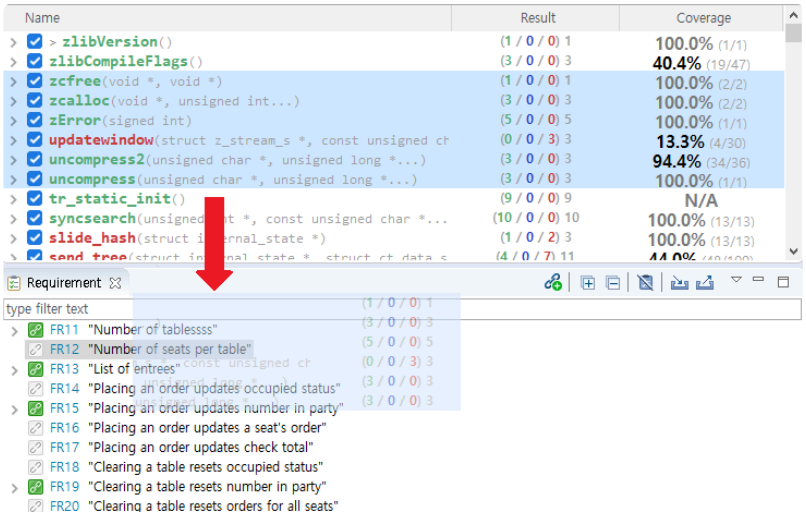
Connect tests and requirements

Requirements can be connect to test/test case/integration test.

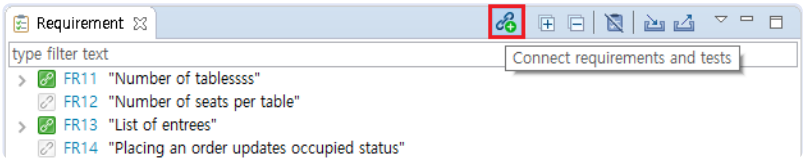
- Drag a requirement from the Requirements View and drop it on the target you want to connect to in the Test View.



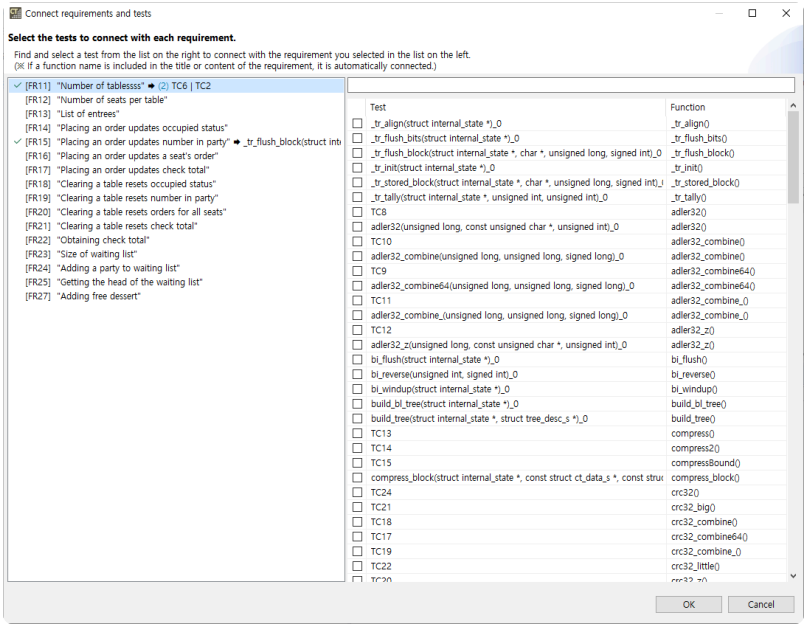
- Drag an item from the test view and drop it on the target you want to connect to in the Requirements View.



- Connecting via the Connect requirements and tests dialog
  1. Select [Connect requirements and tests] from the toolbar menu.

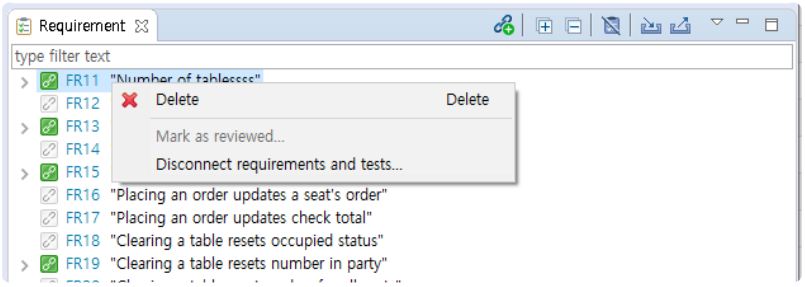


  2. You can connect requirements and tests through the [Connect requirements and tests] dialog.



Disconnect requirements and tests

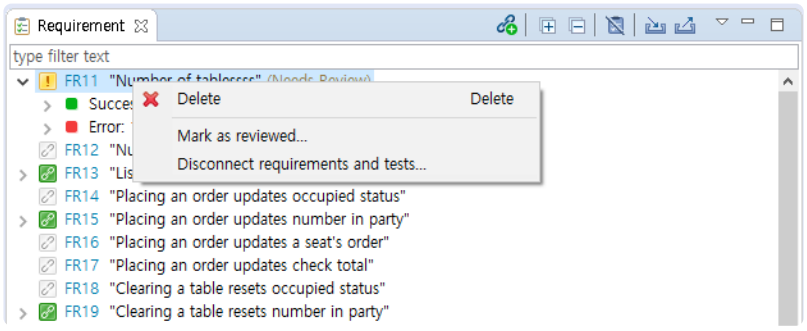
You can disconnect requirements and tests from the context menu in the Requirements View.



Review of tests connected to requirements

1. When a requirement is re-imported, it will be marked as [Needs Review] if the title or description has changed.

2. After reviewing the requirements and connected tests that need to be reviewed, you can mark them as reviewed with the context menu.

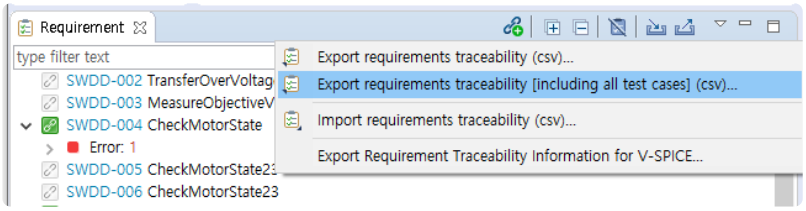


Export requirements traceability

You can export the connection information between requirements and tests to a CSV file.

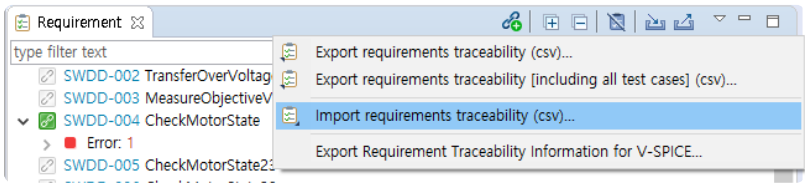
Export requirements traceability [Including all test case]

You can export the connection information between requirements and tests to a CSV file, including all test cases.



Import requirements traceability

After entering traceability information in the CSV file created by [Export requirements traceability], you can import it through the [Import requirements traceability] menu.



- Requirements traceability file description

Row	Description
TEST_TYPE	Test: 0 Integration Test: 1 Test Case: 2
TEST_NAME	Integration Test: Name Test or Test case: a unique name for the test
TESTCASE_NO	Test case: number Non Test case: 0
REQ_KEY	The key of the requirement to be connected or connected to the test
TEST_ID	Integration Test: Name Test: The name displayed in the test view Test case: The name displayed next to the number in the test view

> The unique name of the test can be found by selecting ‘Show as Unique Test Name’ from the toolbar menu of the test view or by searching.

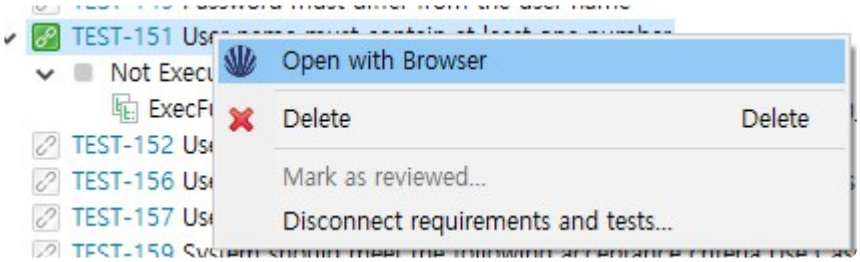
Export requirement traceability information for V-SPICE

Connection information between requirements and tests can be exported to an xml file conforming to the V-SPICE format.

Open in Requirements Management Tool

Requirements imported from the requirements management tool or tests exported to the requirements management tool can be viewed in the requirements management tool through the [Open with Browser]

context menu.



17. Analysis Perspective

The Analysis Perspective provides a UI for analysis information that can be confirmed from the test results.

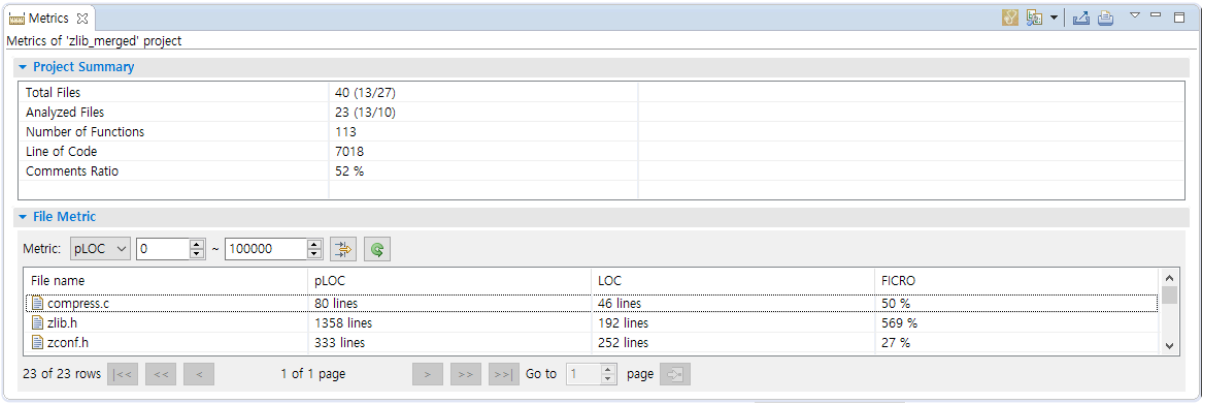
The components that make up the analysis perspective are:

- Metrics View
- Metrics Chart View
- Metrics Top Chart View
- Metrics Bar Chart View
- Metrics Diagnosis Chart View
- Unused Function View
- Source-Header Relation View
- Global Variable Relation View
- Function Call Hierarchy View
- Source Code Editor section

Views that are not included by default can be opened from [Window]> [Show View]> [Other…] on the top menu.

17.1. Metrics View

The Metrics view shows the result that measures the metric for the project. It shows the summary information for the overall metric and the metric information for each items included in the project (module, file, class, function).



Icons

Icon	Description
	Module (a logical portion that divides the project for each function)
	File (Source file and header file belonging to the project)
	Class
	Function

Toolbar menu

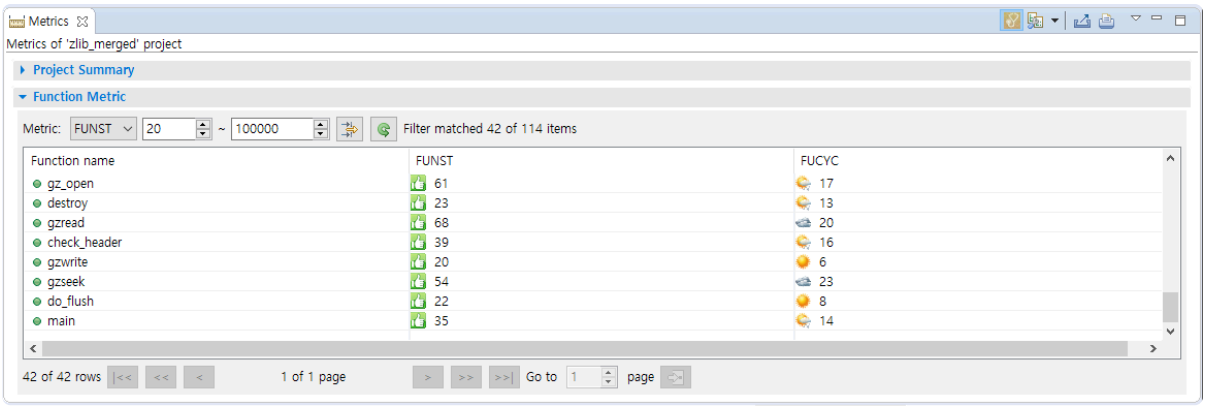
Menu	Description
Show Diagnosis	Displays a diagnostic image of the value in the left of the metric value.
Change Mode	Changes the view mode (module, file, class, function).
Export View Content	Exports the contents of view as a report.
Print View Content	Prints the contents of view.

Pull-down menu

Menu	Description
Sort by	Sorts based on the selected item (even if clicking the column name in the table, it is sorted equally).
Columns	Changes the order and width of the columns (allows you to change the order by dragging

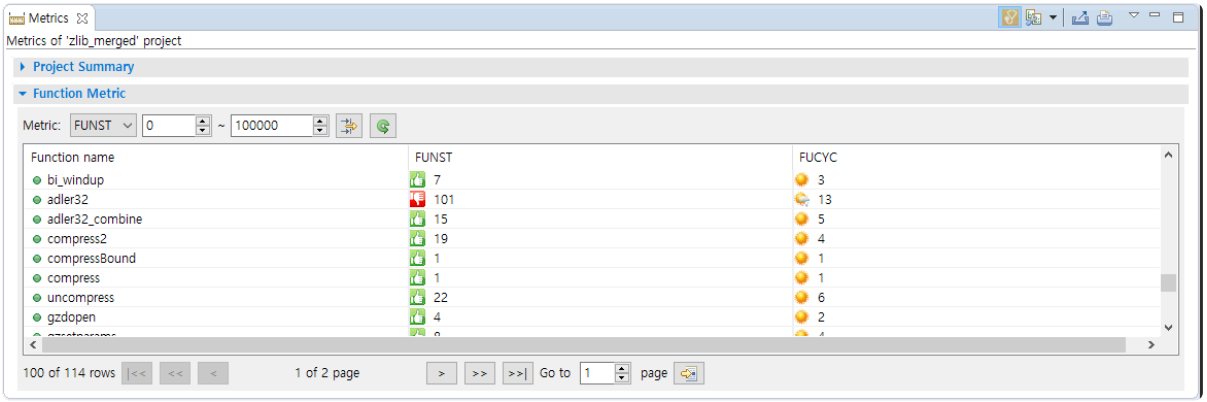
	and dropping the table's column name).
Configure diagnosis	Opens the diagnostics preferences page (allows you to set the diagnostic step and the ranges or images for each metric).
Set to Hide/ Show metric	Sets whether to hide or show the metrics.
Metric View Option	Sets the number of lines per page in the Metric view.

Filtering



The filtering allows you to view only those items within the desired range for the specific metric.
Select a metric, enter the range value and click [Filtering] button in the right section to see those values corresponding to the entered information.
If you want to return to the status before filtering, click [Initialize the Filtering Criteria] button on the right side of [Filtering] button.

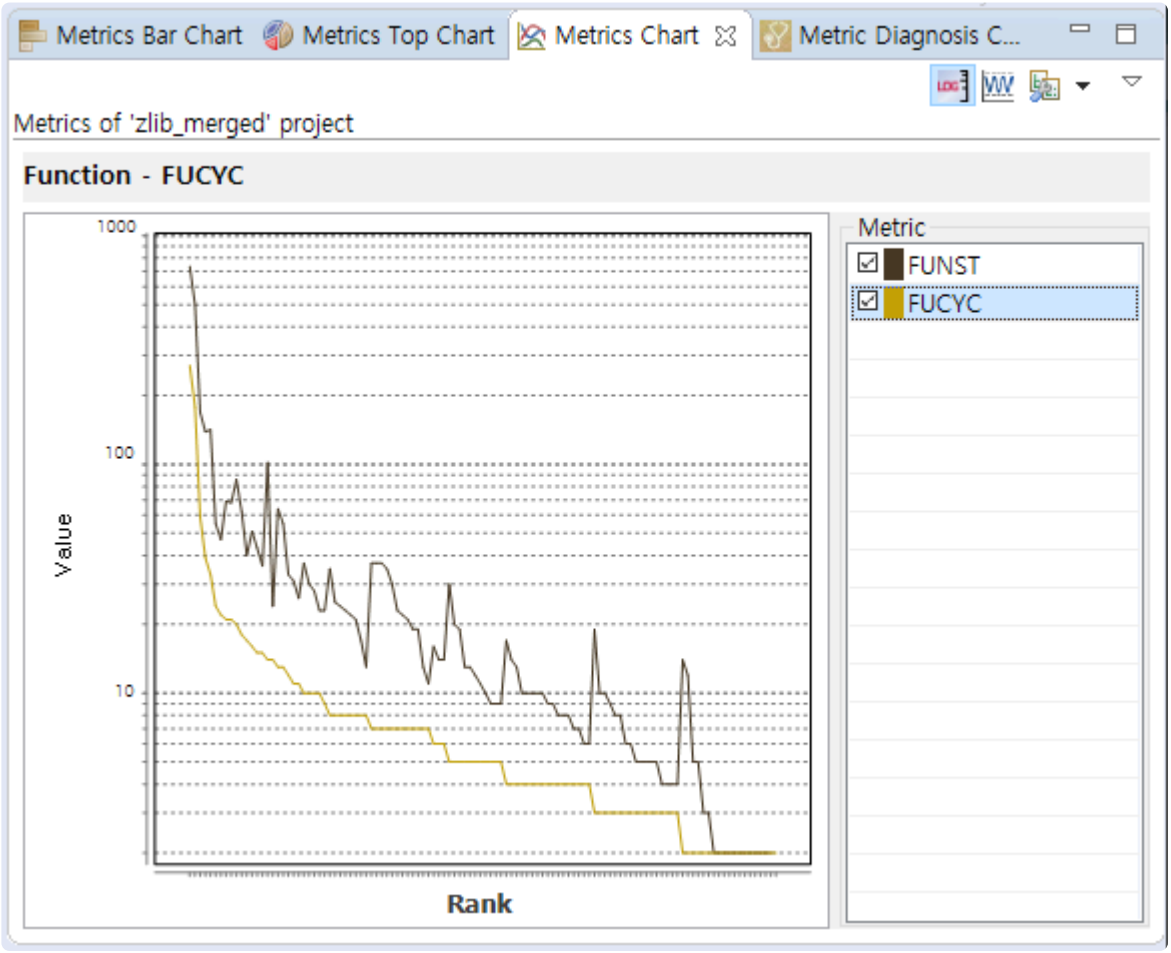
Paging



The Paging allows you to view many items conveniently.

17.2. Metrics Chart View

The Metrics Chart shows the metric view in a line chart. The x-axis is each item, and the y-axis is the metric value of the item. You can compare the distribution of values between different metrics each other in the project.



Toolbar menu

Menu	Description
Show Log Scale	Changes the y-axis of chart to log units.
Show Normalized Metric	Shows the standardized metric value.
Change Mode	Changes the view mode (module, file, class, function).

Pull-down menu

Menu	Description
Sort by	Sorts based on the selected item (even if clicking the column name in the table, it is sorted equally).
Set to Hide/Show	Sets whether to hide or show the metrics.

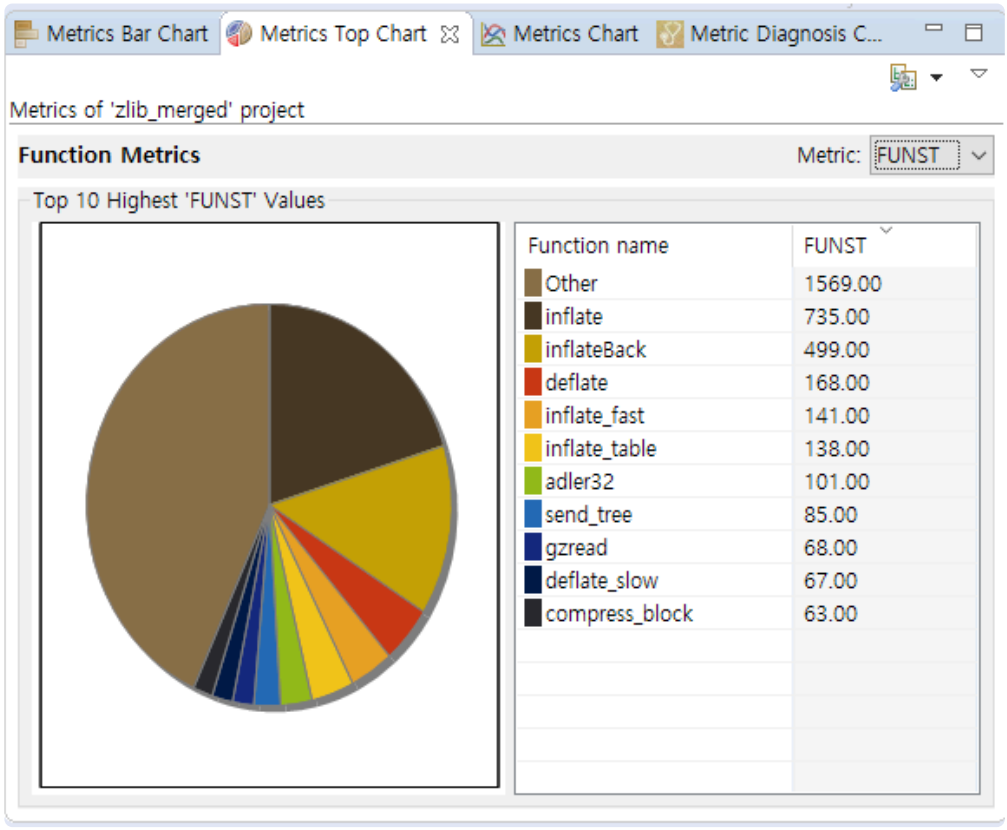
metric	
Metrics View Options	Sets the connection function with Metric view.

Select Metric

If you select the checkbox left to the metric name in the right table, the checked metrics are displayed in the chart. If you click the metric name in the checked metric, the items are sorted based on that metric value.

17.3. Metrics Top Chart View

The Metrics Top Chart view shows the top items of the metric value in a pie chart. You can see the ratio of the top items in the total.



Menu	Description
Change Mode	Changes the view mode (module, file, class, function).

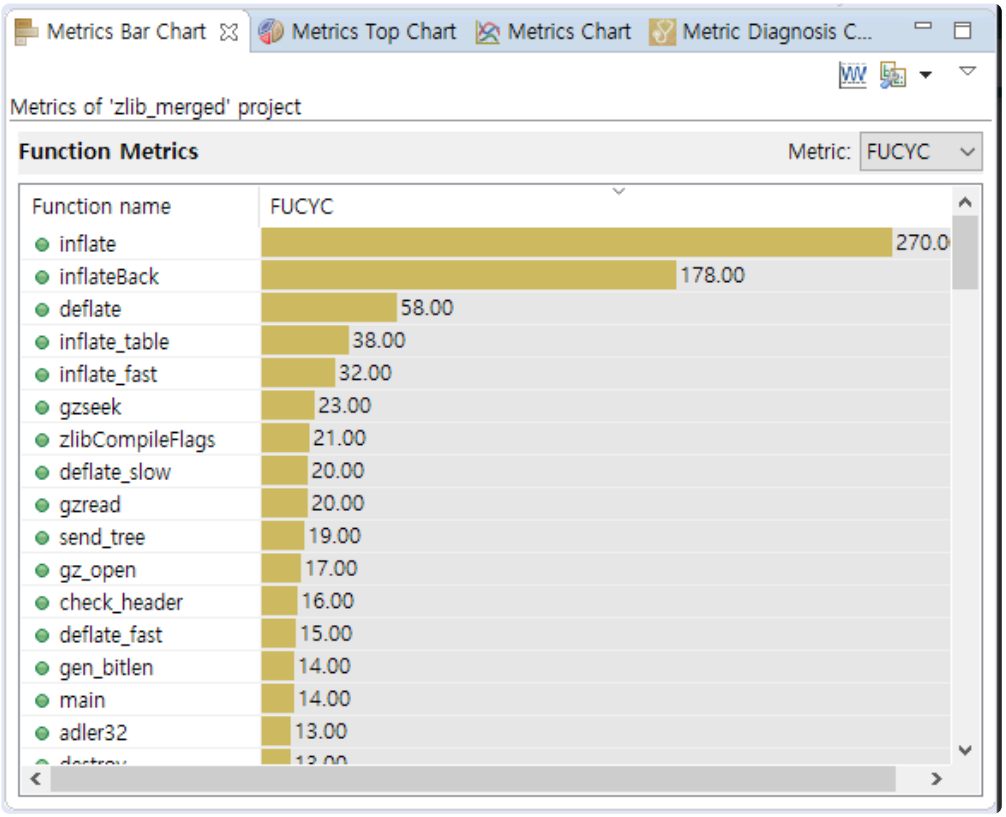
Pull-down menu

Menu	Description
Metrics Hide/Show Setting	Sets whether to hide or show the metrics.
Metrics View Options	Sets the number of top items to be displayed and sets whether to show the sum of the resting items.

The selection of the pie chart and the selected item in the table are connected together.

17.4. Metrics Bar Chart View

The Metrics Bar Chart view shows the metric values in a bar chart. You can check the value size for each item in the bar chart.



Icons

Icon	Description
	Module (a logical portion that divides the project for each function)
	File (Source file and header file belonging to the project)
	Class
	Function

Toolbar menu

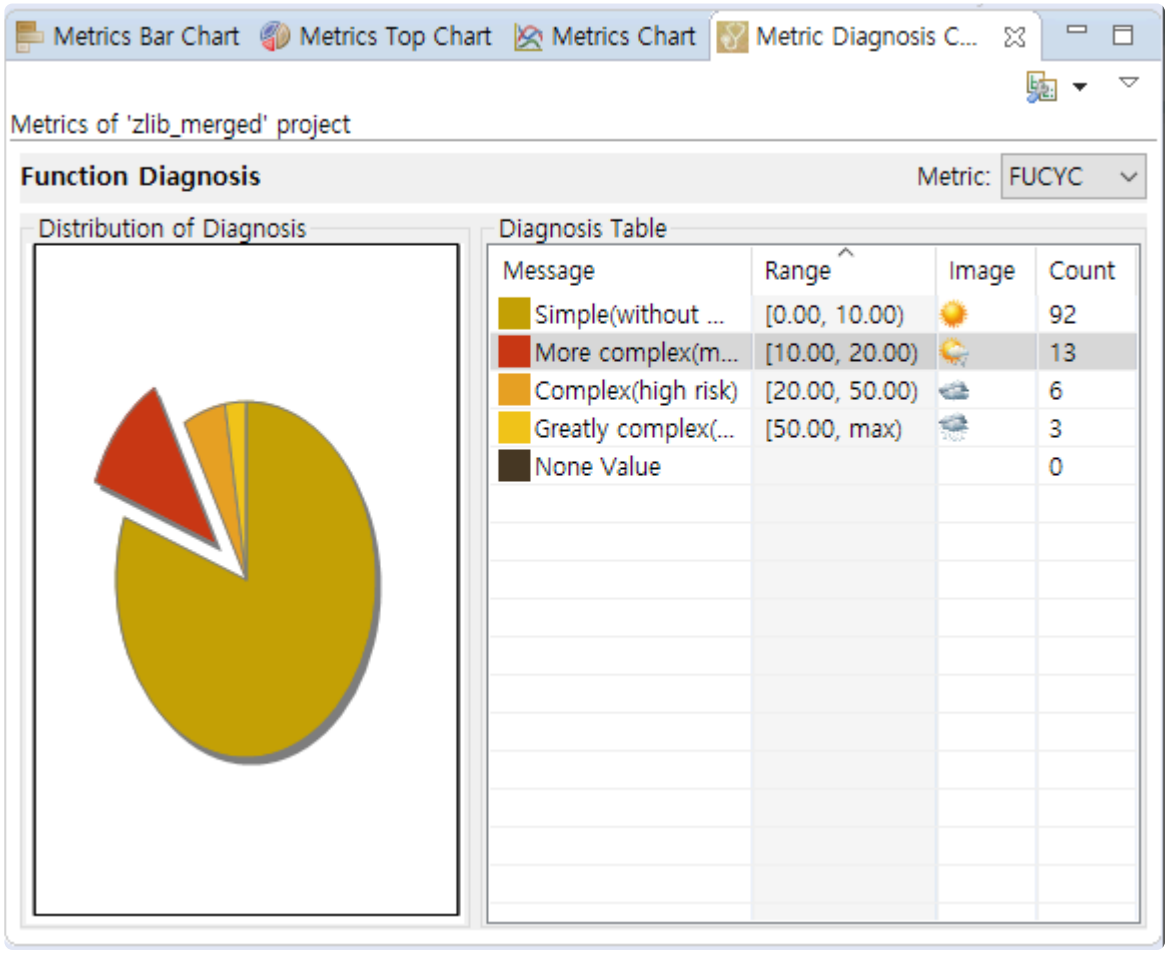
Icon	Description
Show Normalized Metric	Shows the standardized metric value.
Change Mode	Changes the view mode (module, file, class, function).

Pull-down menu

Icon	Description
Metrics Hide/Show Setting	Sets whether to hide or show the metrics.
Metric View Options	Sets the number of items to be displayed.

17.5. Metrics Diagnosis Chart View

The Metric Diagnosis Chart view shows the number of items included in the diagnostic range in a pie chart. You can check the number of items within each diagnostic range in the project.



Toolbar menu

Menu	Description
Change Mode	Changes the view mode (module, file, class, function).

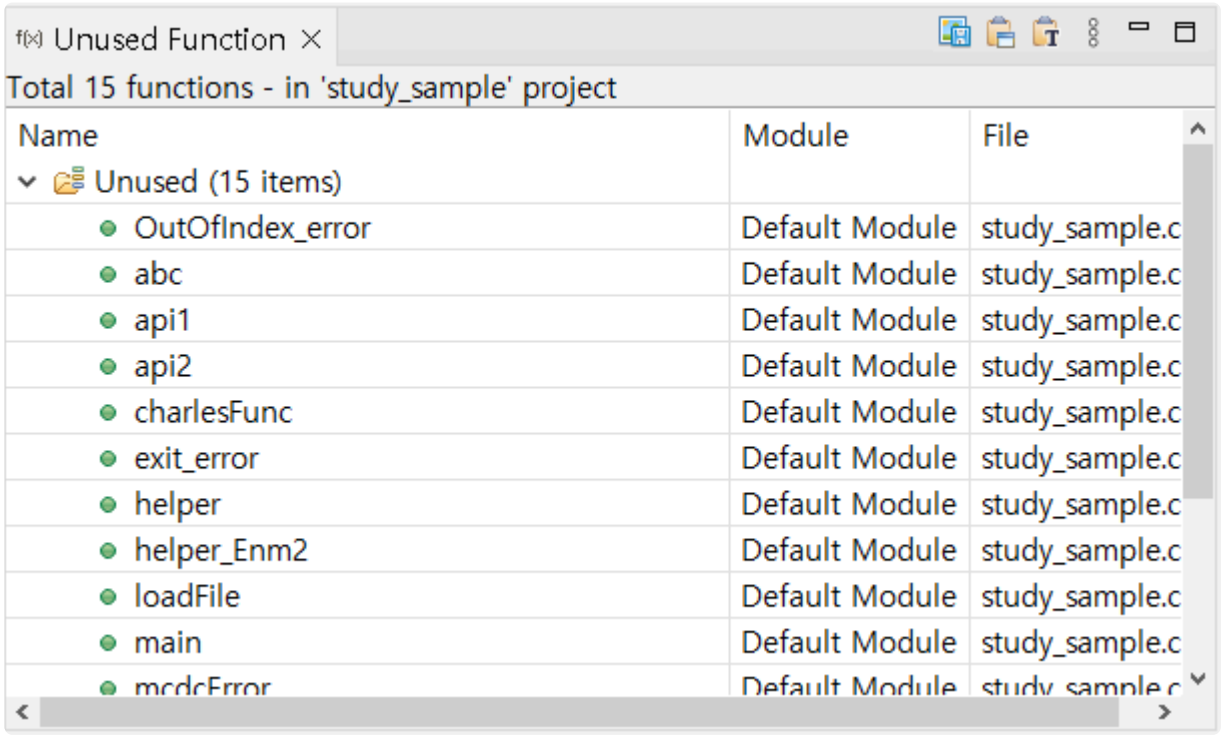
Pull-down menu

Menu	Description
Configure Diagnosis	Opens the diagnostics preferences page (allows you to set the diagnostic step and the ranges or images for each metric).
Metrics Hide/ Show	Sets whether to hide or show the metrics.
Metric View Options	Sets whether to display the items having no diagnostic value.

The selection of the pie chart and the selected item in the table are connected together.

17.6. Unused Function View

The Unused Function view shows the functions included in the selected project by classifying them according to the specified criteria. It shows the currently unused functions and the criteria for classifying can be added.



Menu	Description
Save as PNG	Save unused function view as image.
Copy to System clipboard	Copy unused function view image to system clipboard.
Copy Text to System clipboard	Copy unused function view content to system clipboard.

Icons

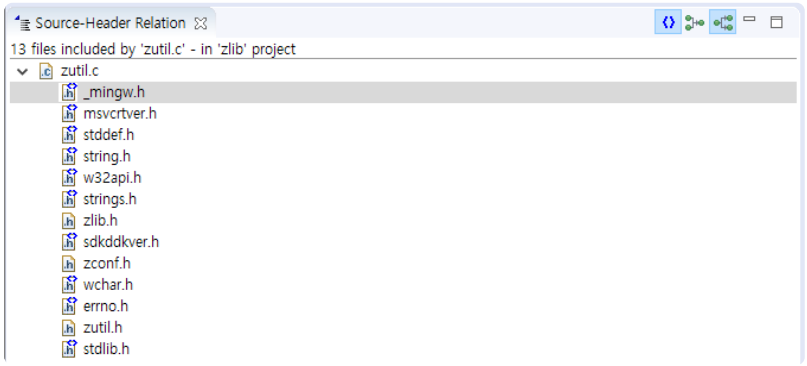
Icon	Description
	Classification criteria
	Function group
	Function

Pull-down menu

Menu	Description
Function View Options	Sets the number of functions to be displayed for each group.

17.7. Source-Header Relation View

Source-Header Relation view shows `<include>` and `#include` relations between the source file and the header file



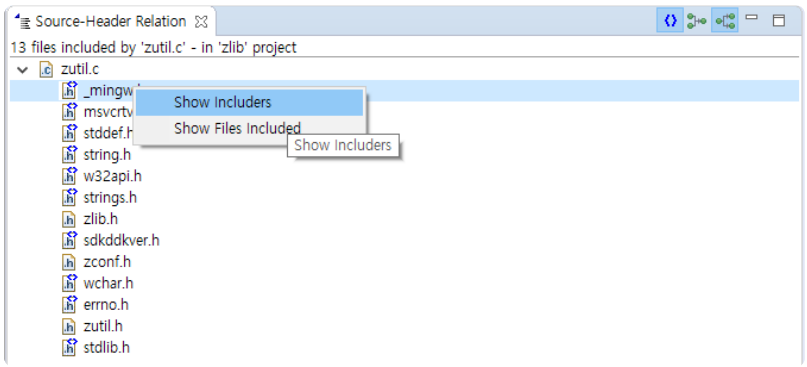
Icons

Icon	Description
	Source file
	Header file
	System header file

Toolbar menu

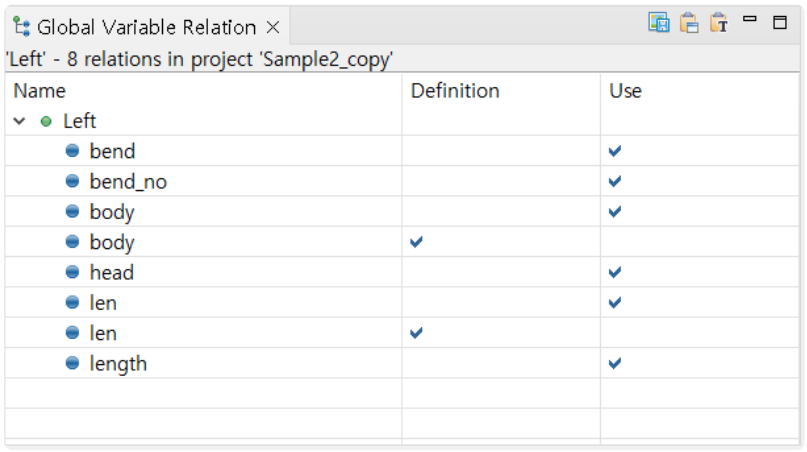
Menu	Description
Show System Header	Shows the system header.
Show Includers	Shows the files that includes the selected file.
Show Files Included	Shows the file included in the selected file.

Show Includers, Show Files Included
You can display the files that includes the selected file or the files that the selected file is included by using the menu displayed when right-clicking a file in the view.



17.8. Global Variable Relation View

The Global Variable Relation view shows the function where the global variable is defined and used. If a function is selected, it shows the global variables defined or used in the function, and if a global variable is selected, it shows the functions where the global variable is defined or used.



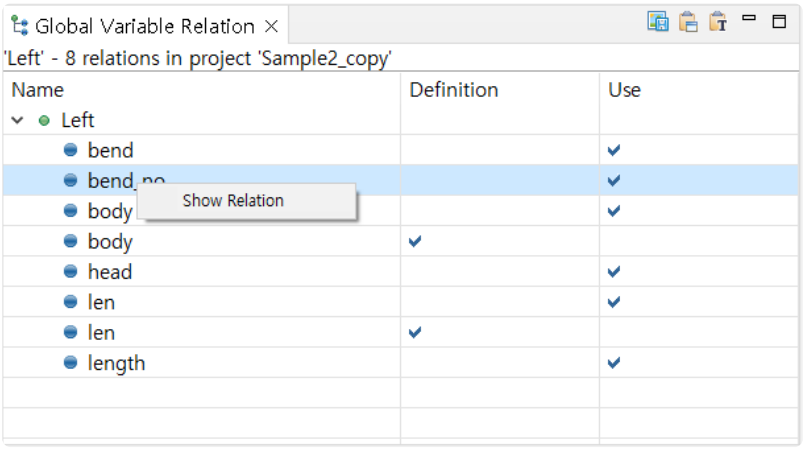
Menu	Description
Save as PNG	Save global variable relation view as image.
Copy to System clipboard	Copy global variable relation view image to system clipboard.
Copy Text to System clipboard	Copy global variable relation view content to system clipboard.

Icons

Icon	Description
	Function
	Global Variable

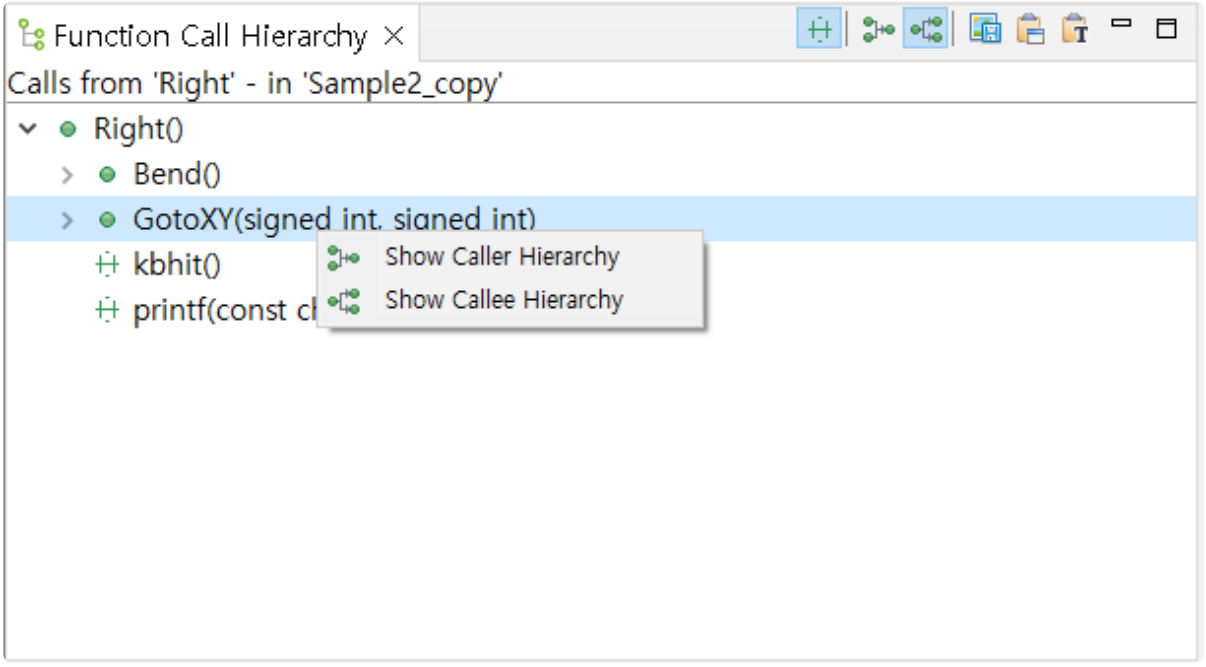
Show Relation

You can view the relation for the selected items again by using the menu displayed when right-clicking the related item in the view.


Global Variable Relation
'Left' - 8 relations in project 'Sample2_copy'
Name
Definition
Use
Left
bend
bend_no
body
body
head
len
len
length
Show Relation

17.9. Function Call Hierarchy View

The Function Call Hierarchy View shows the information for the function called by the selected function or the called function in the hierarchy structure.
You can change the call information for the selected function by using the context menu that appears when selecting and right-clicking the function displayed in the hierarchy structure and double-click it to open the editor for the source file that the function is defined and to go to the location of the selected function. (However, not included if it is a system function)

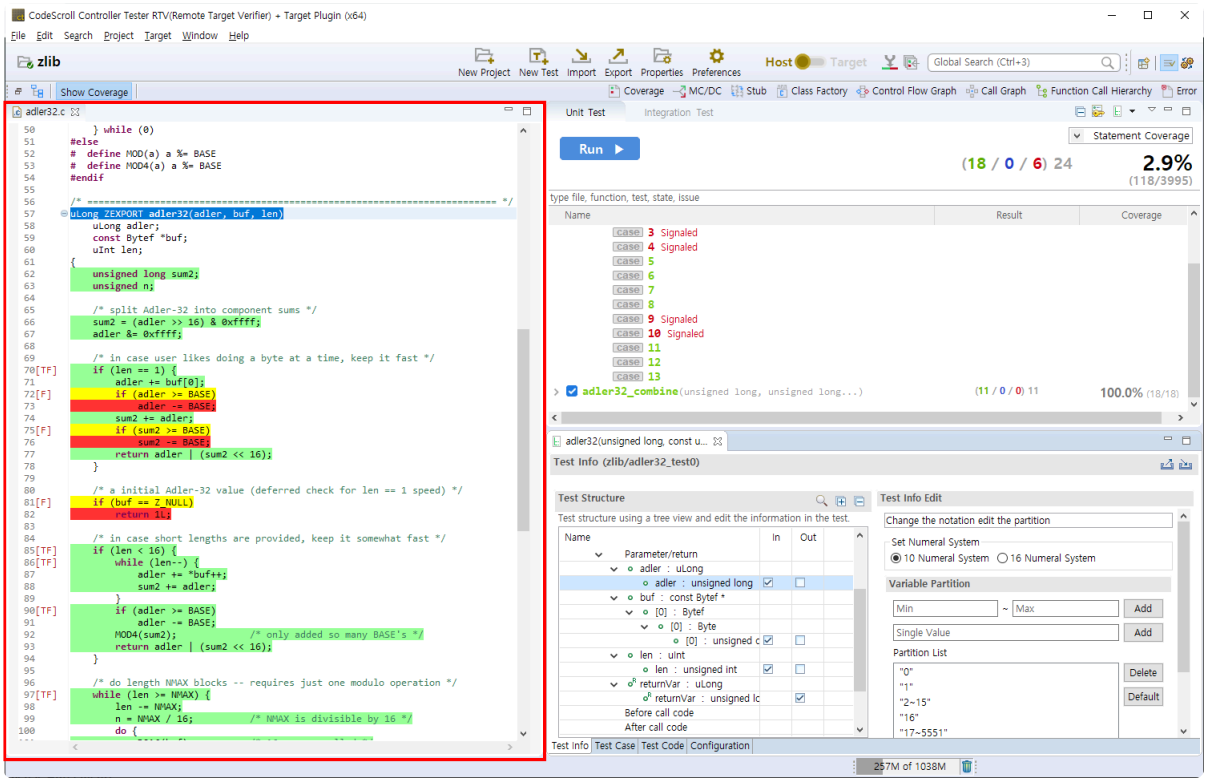


Menu	Description
Show System Function	Sets whether to indicate the call information for the system function in a hierarchical structure.

Show Caller Hierarchy	Shows the functions that call the selected functions in a hierarchical structure.
Show Callee Hierarchy	Shows the functions called by the selected function in a hierarchical structure.
Save as PNG	Save function call hierarchy layer view as image.
Copy to System clipboard	Copy function call hierarchy view image to system clipboard.
Copy Text to System clipboard	Copy the Function Call Hierarchy View contents to system clipboard as text

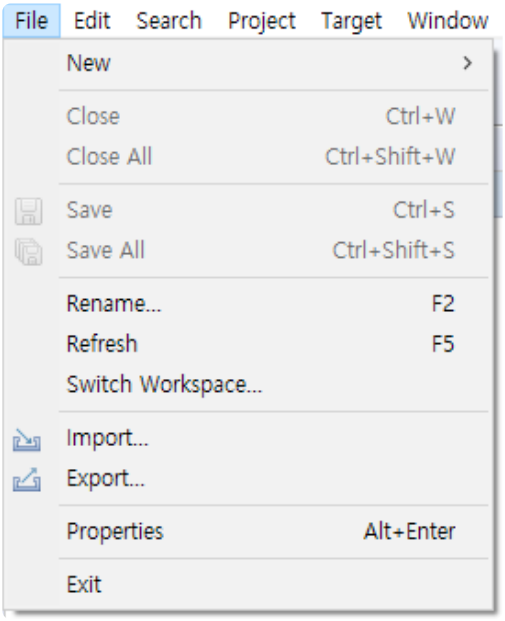
17.10. Source Code Editor section

The source code editor section is a pre-reserved area, located on the left of the screen.



18. File Menu

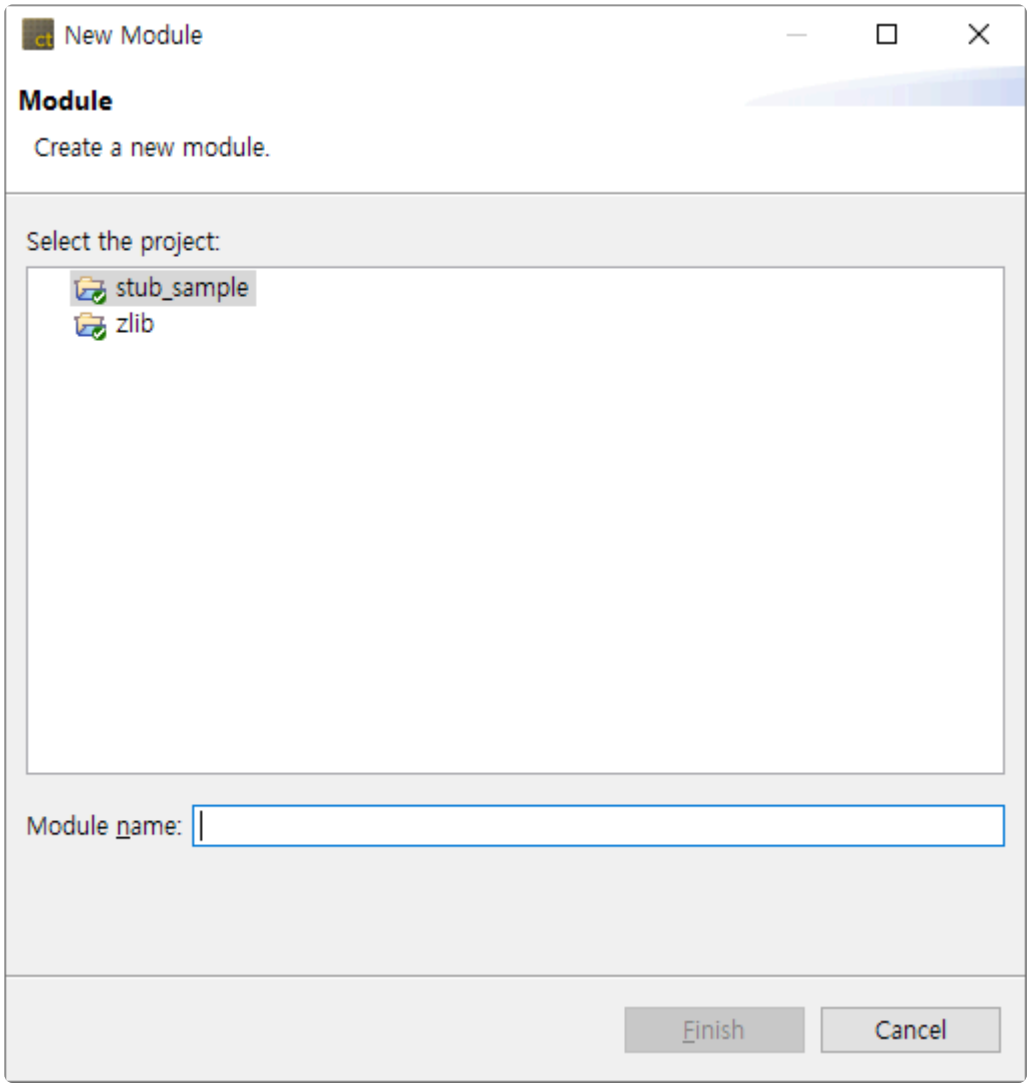
In File menu, Create a new project or module, Editor-related operation, Refresh, Change Workspace, Import and Export, View Properties of the selected item and Exit Tool etc. can be carried out.



- New:
  - Create a project
  - Create a module
- Close/Close all/Save/Save all: Editor-related menu
- Rename: You can modify the name of project or module.
- Switch workspace
- Import
- Export
- Exit: Exit tool.

18.1. Create a Module

You can create a new module in the project. The information other than the module name is created the same as the modules already existed in the target project.

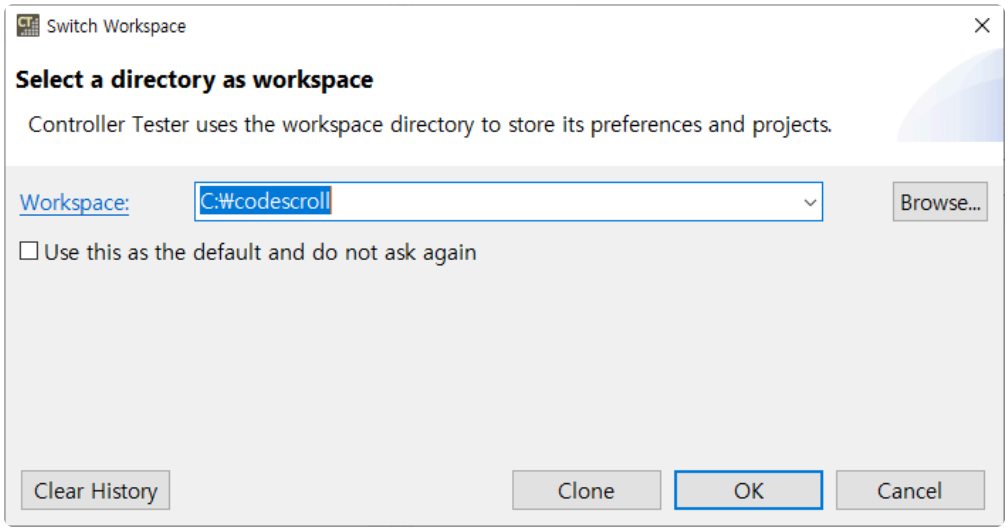


1. In [New] menu, click [Other…], select [Module] in [Other] and click [Next].
2. Select the project that you want to create a new module.
3. Enter the module name. The module name existed already cannot be used.
4. Click [Finish] to create the module.

You can also create a module by using the [New] menu displayed when right-clicking a project in Test Navigator view.

18.2. Switch Workspace

You can switch the directory of workspace. When you witch the workspace, the tool restarts with the switched workspace.

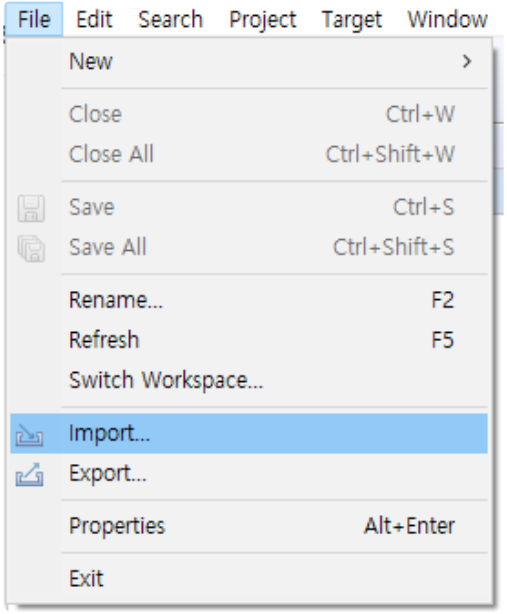


Click [Browse…] to select the workspace directory to be switched or to enter it manually. You can select the workspace directory selected previously in the list displayed when clicking [▼].
If you select the checkbox [Use this as the default and do not ask again], you are not prompted again to select a workspace directory the next time you execute the tool.
If you click [Clone], the selected workspace is copied to the other directory.
If you click [Clear History], you can delete the workspace list.

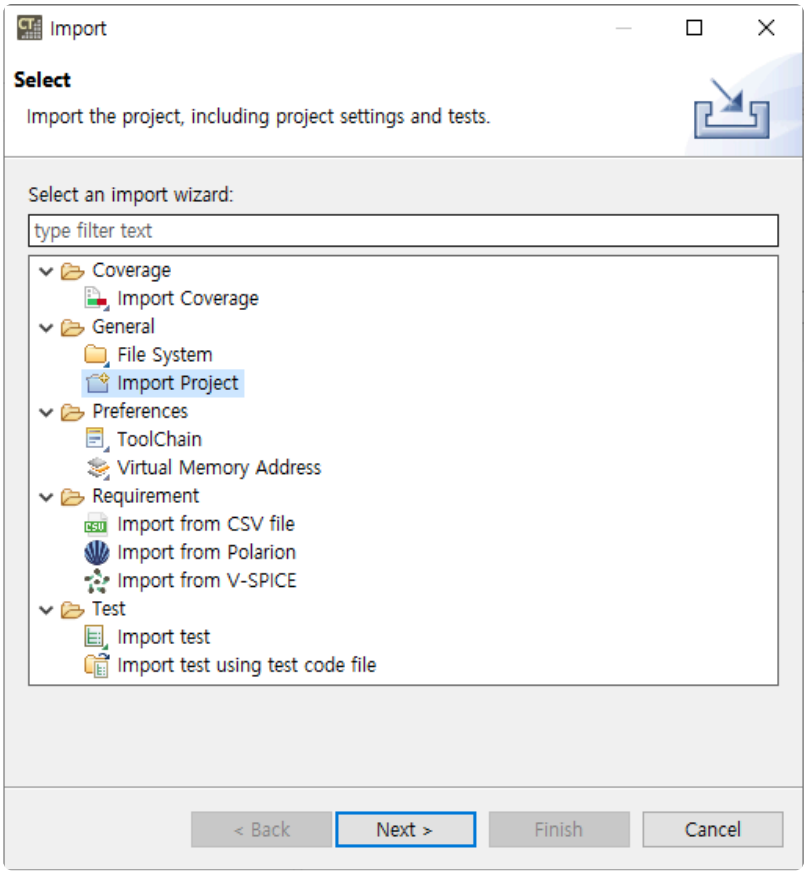
18.3. Import

Controller Tester provides a feature that import project and data used other version or other PC.

1. Select [File] > [Import] from the main menu or [Import] from the dashboard to open the Import Wizard.



2. Select a item to import and click [Next].



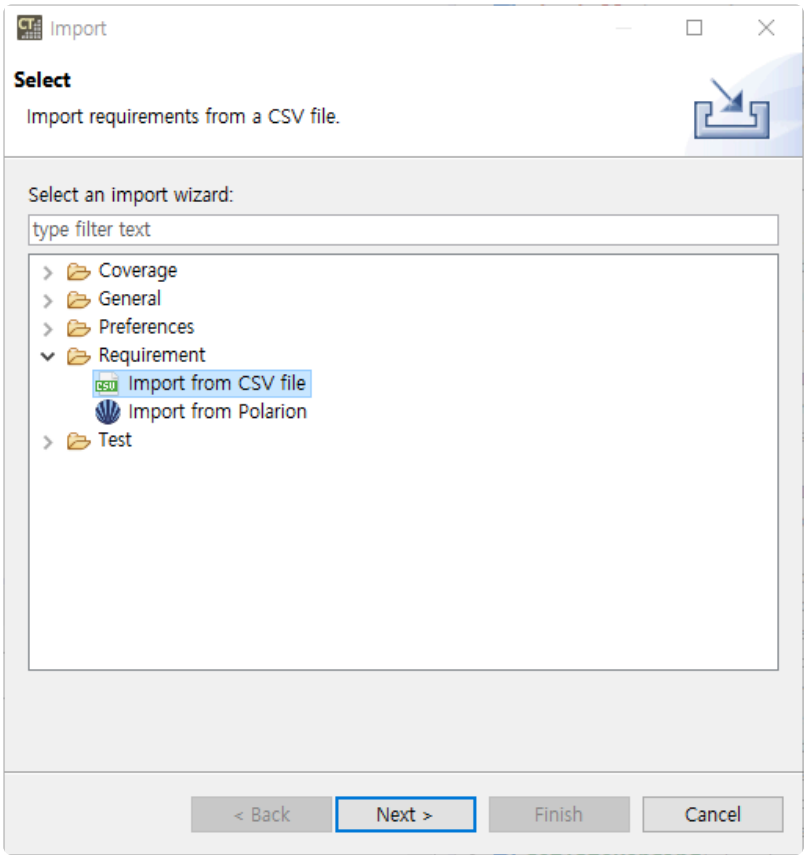
Parent item	Child item	Description

Coverage	Import Coverage	import coverages in csd format.
General	File System Import project	Import files or projects.
Preferences	ToolChain Virtual Memory Adress	Import toolchains or virtual memory adress exported before.
Requirement	Import from CSV file Import from Polarion Import from V-SPICE	Import requirements from CSV files or requirement management tools.
Test	Import test Import test using test code file	import tests exported before or tests from test codes.

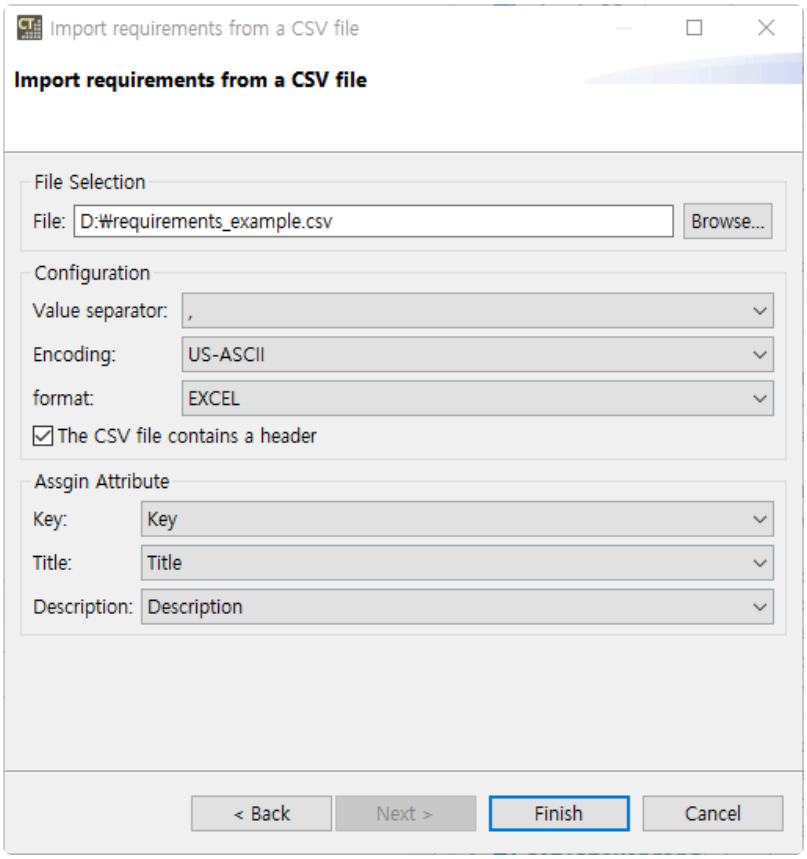
18.3.1. Requirement − Import from CSV file

You can import requirement documents written in CSV file to Controller Tester.

1. In the Import Wizard, select [Requirement] -> [Import from CSV file] and click [Next].



2. Enter informations for importing requirements and click [Finish].



a. File Selection
   - Enter the path of a file that the requirements are stored.

b. Configuration
   - **Value seperator** : Enter a separator that separate each value in the CSV file. Default value is ,(comma).
   - **Encoding** : Enter a encoding of the CSV file. Default value is US-ASCII.
   - **format** : Enter a format of the CSV file. Default value is EXCEL.
   - **The CSV file contains a header** : Check when the CSV file contains a header.

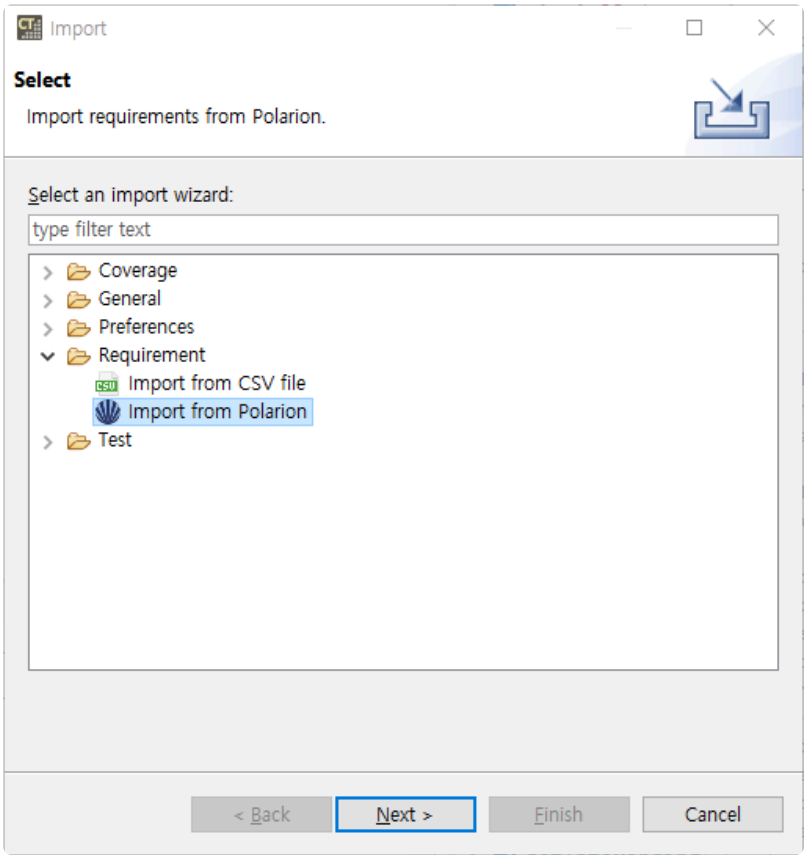
c. Assign Attribute
   - **Key** : Select the column that corresponds to the key of the requirements.
   - **Title** : Select the column that corresponds to the title of the requirements.
   - **Description** : Select the column that corresponds to the description of the requirements.

The information you entered in the previous import is automatically filled in.

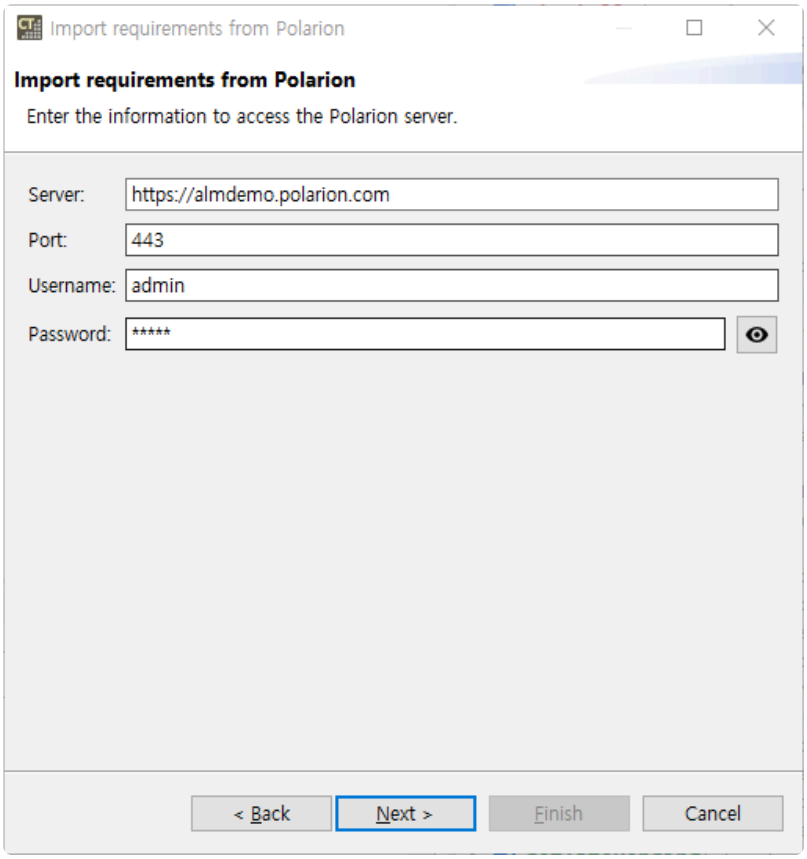
18.3.2. Requirement − Import from Polarion

You can import requirement documents from Polarion to Controller Tester.

1. In the Import Wizard, select [Requirement] -> [Import from Polarion] and click [Next].

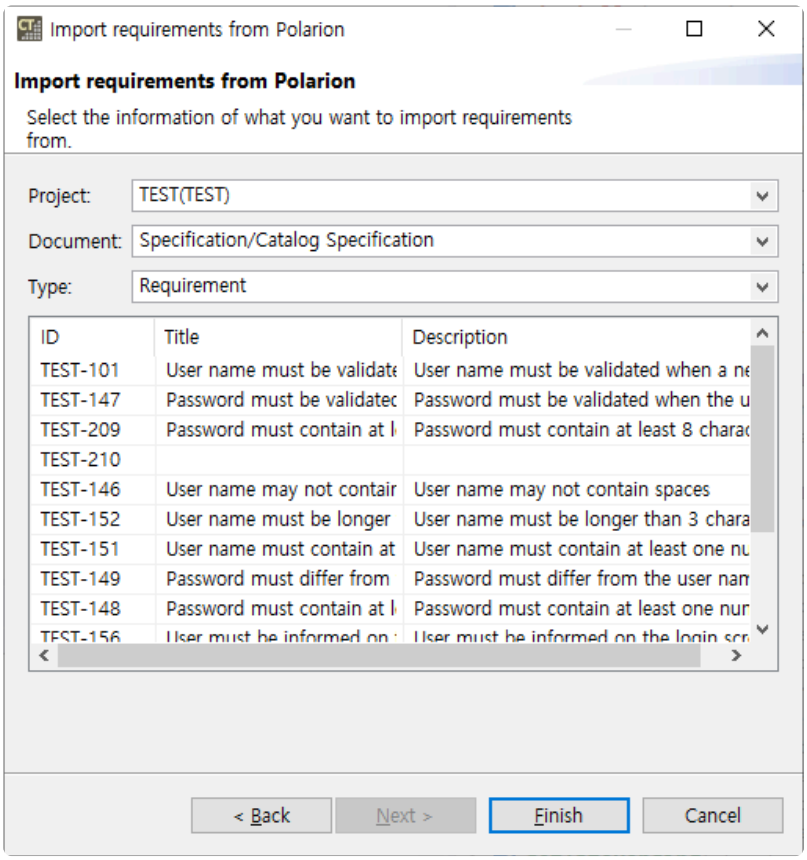


2. Enter the information to connect with Polarion server and click [Next]. You can see the entered password to click button.



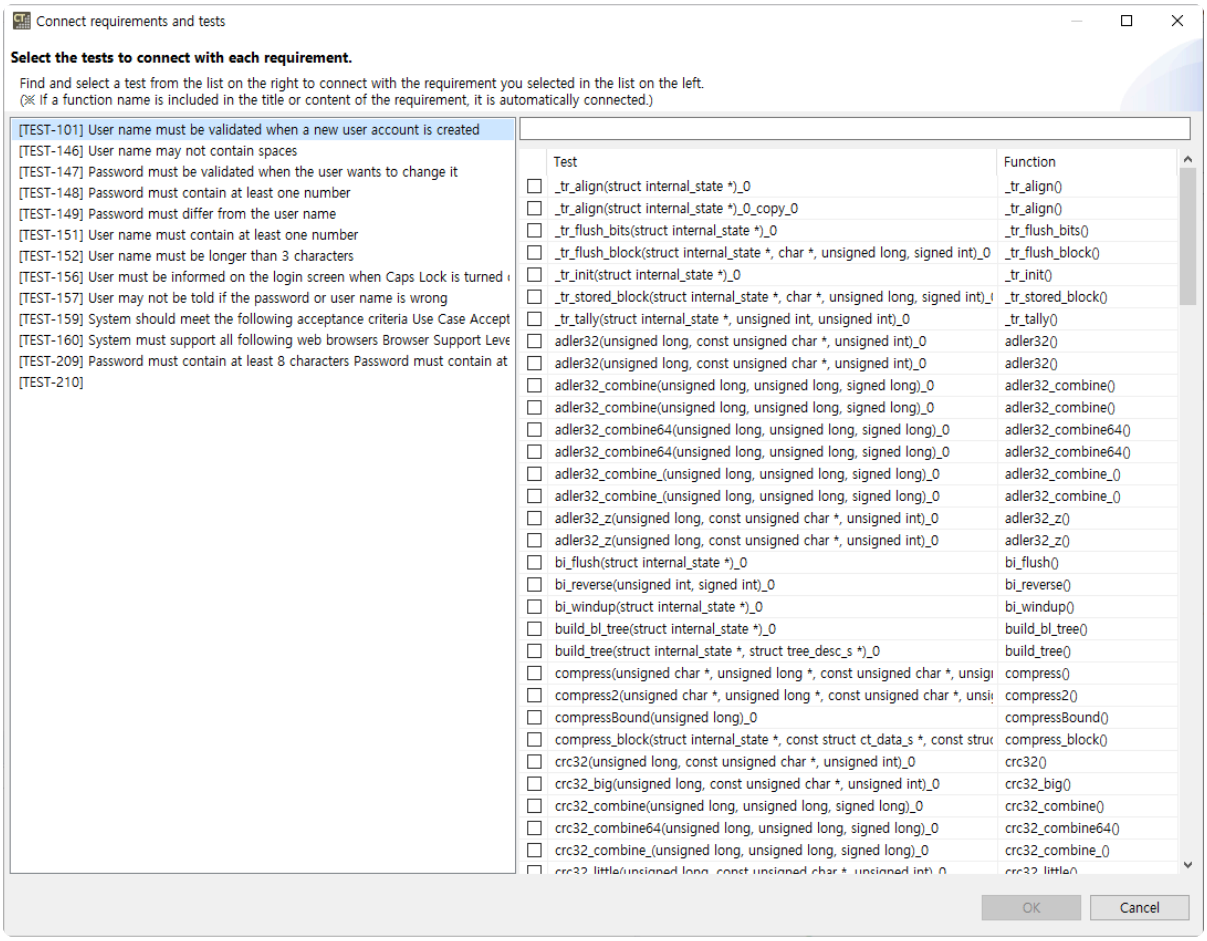
- The information of Polarion's demo server is entered in server and port.
- If you use a local server, enter `http://ip_address` in server and port number of local server in port.

3. When you enter the Project, Document, and Type to import from Polarion, the corresponding requirements list appears. Check the requirements and click the [Finish] button to import the requirements.



- If you do not select the Document, you can also import requirements that are not included in a specific document.

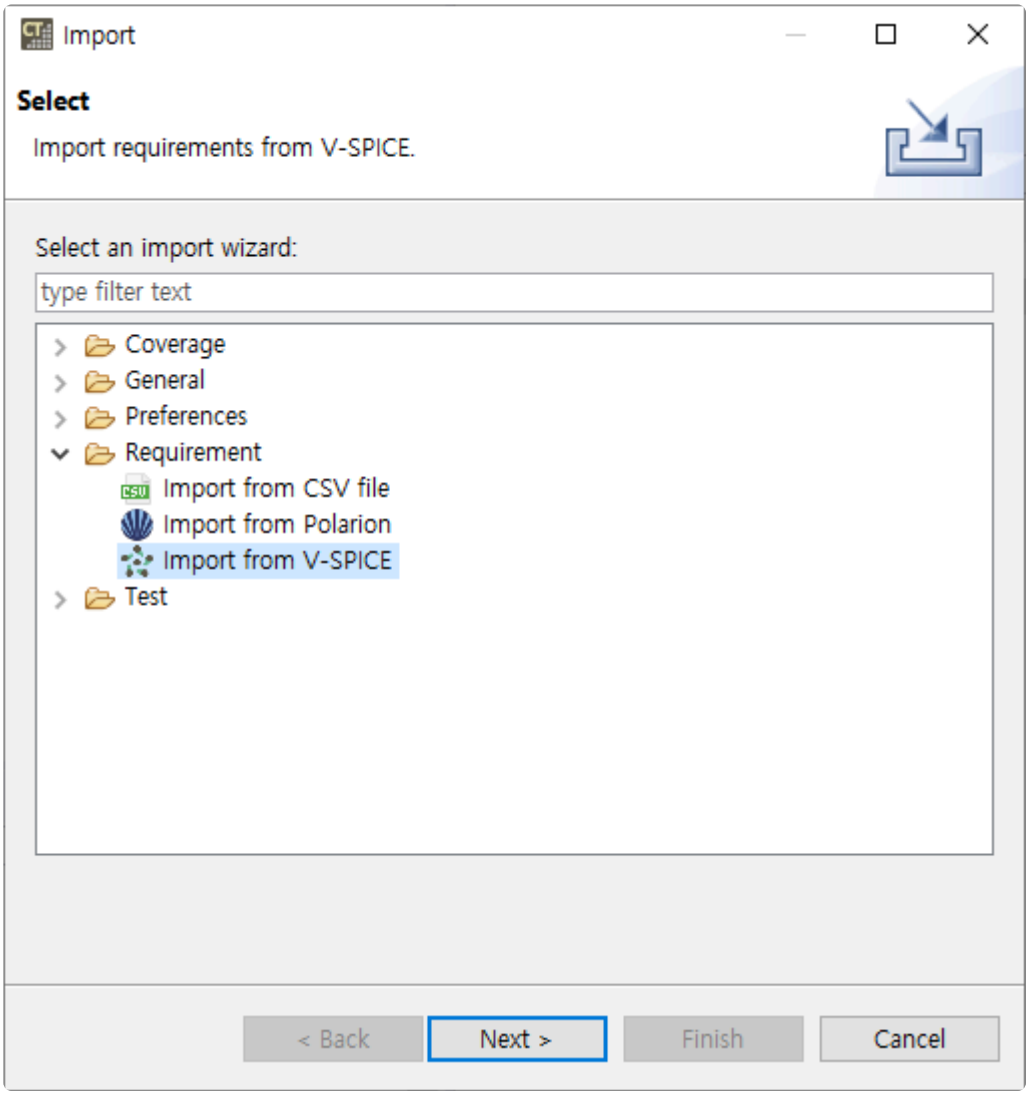
4. [Connect requirements and tests] dialog appears where you can connect the imported requirements and tests. If the title or description of the requirement contains the name of the function associated with the test, it is automatically linked. Click [Cancel] to finish [Connect requirements and tests] without linking requirements and tests.



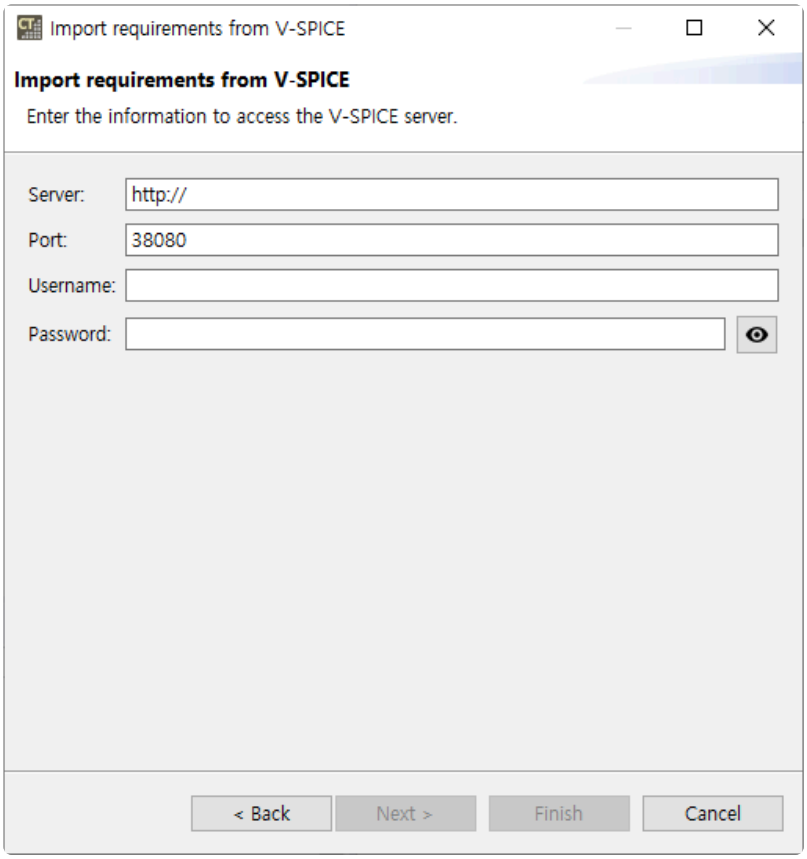
18.3.3. Requirement – Import from V-SPICE

You can import requirement documents from V-SPICE to Controller Tester.

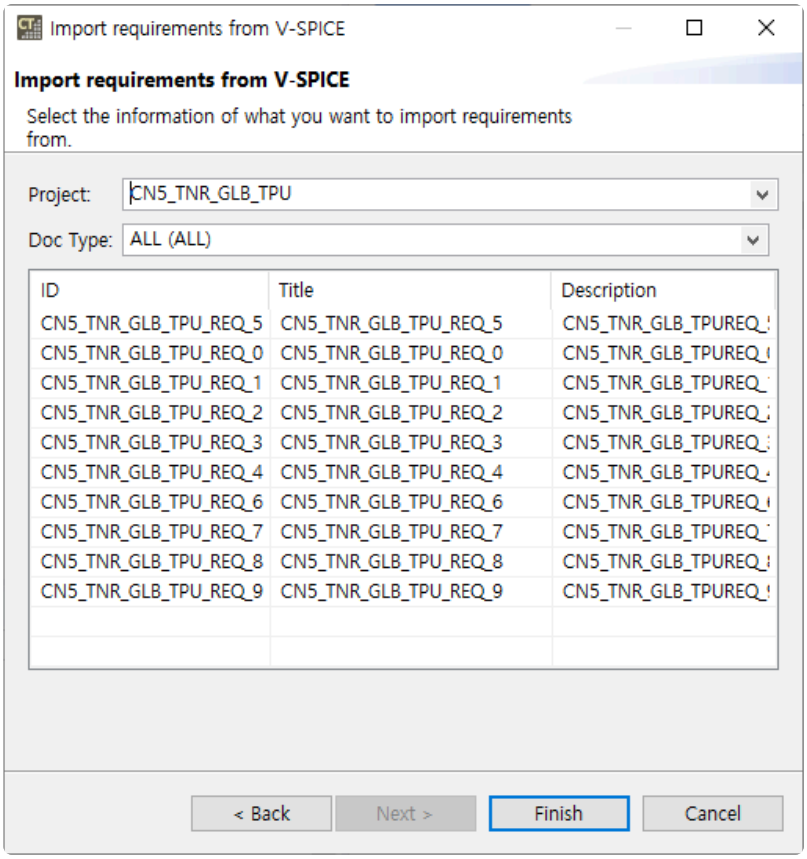
1. In the Import Wizard, select [Requirement] > [Import from V-SPICE] and click [Next].



2. Enter the information to connect with V-SPICE server and click [Next]. You can see the entered password to click button.

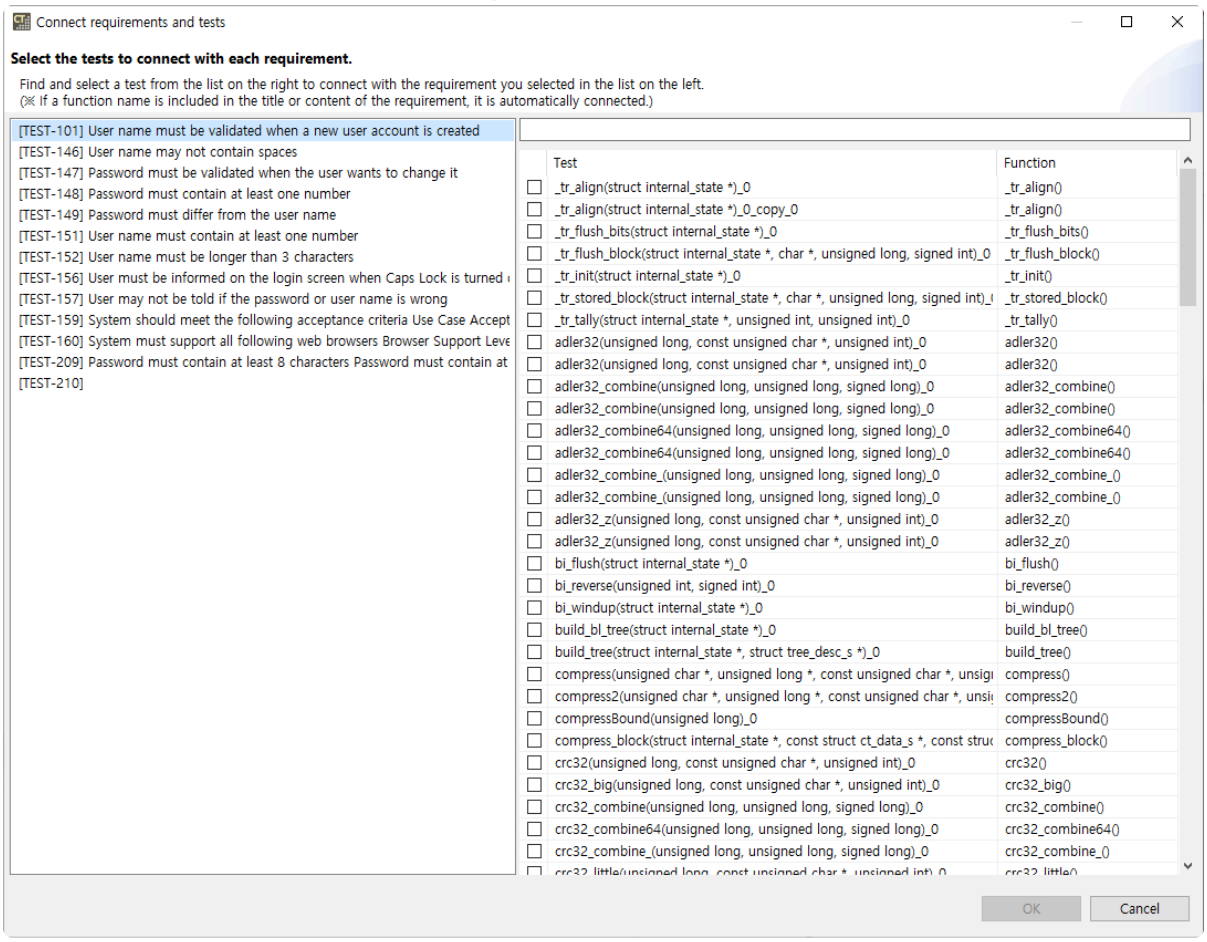


3. When you enter the Project to import from V-SPICE, the corresponding requirements list appears. Check the requirements and click the [Finish] button to import the requirements.



4. [Connect requirements and tests] dialog appears where you can connect the imported requirements and tests. If the title or description of the requirement contains the name of the function associated with the test, it is automatically linked. Click [Cancel] to finish [Connect

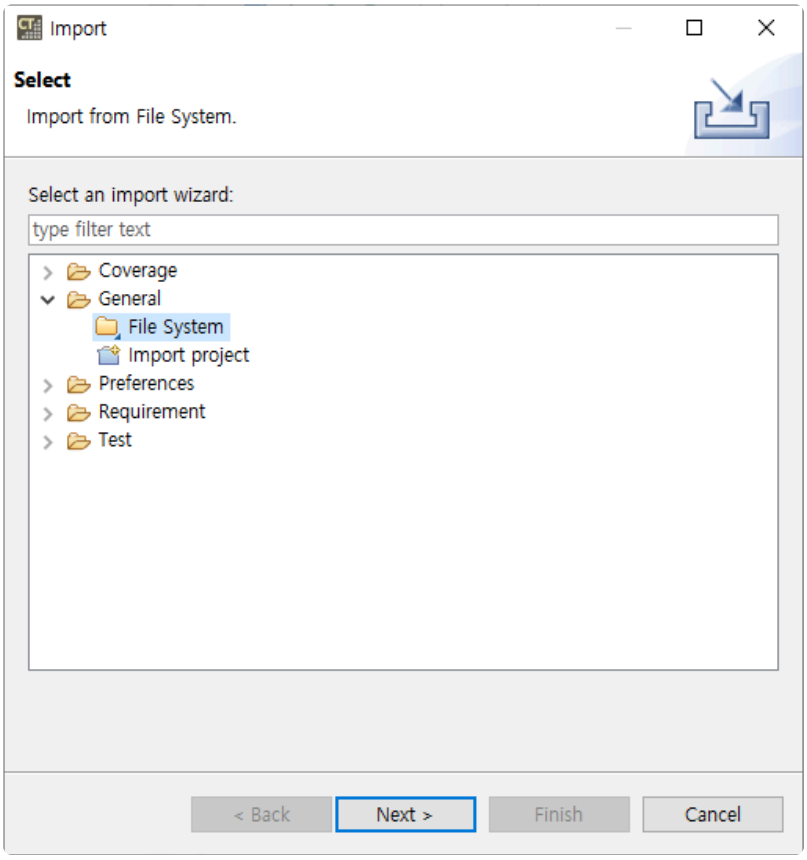
requirements and tests] without linking requirements and tests.



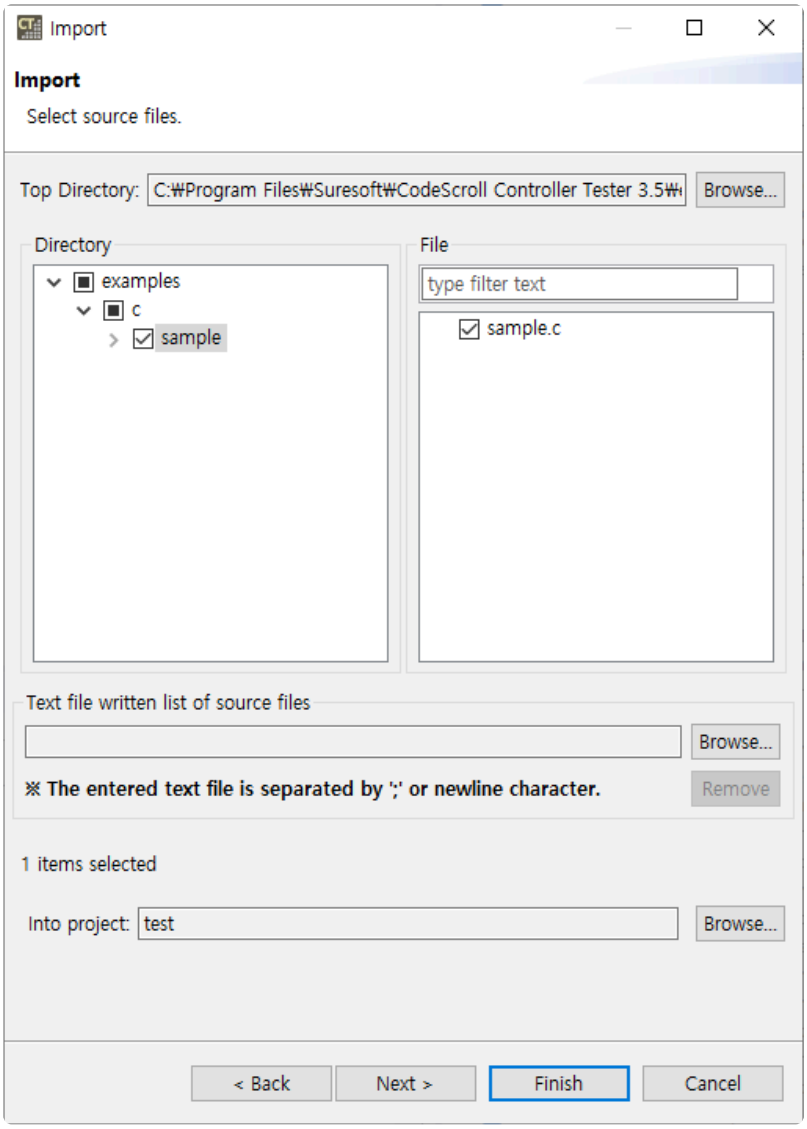
18.3.4. General − File System

You can add the source files to the project by using [Import] – [File System] feature.

1. In the Import Wizard, select [General] -> [File System] and click the [Next] button.

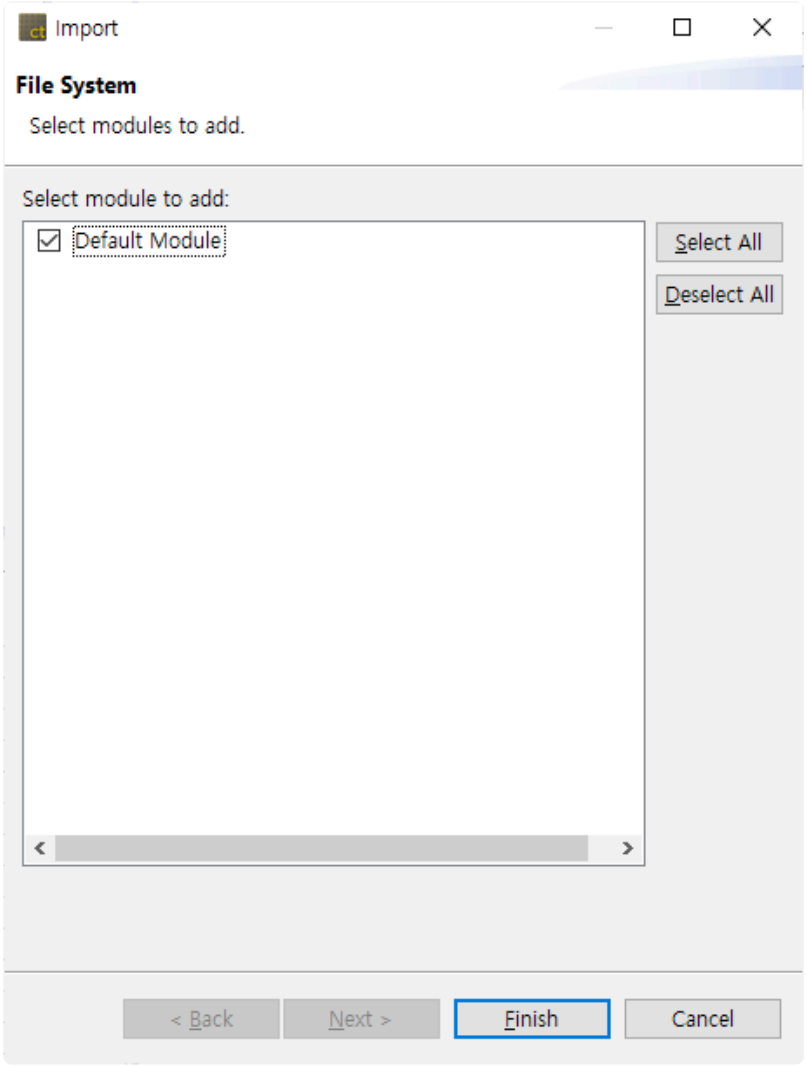


2. Enter the information of the file to be imported and click the [Next] button.



Item	Description
Top Directory	Select the directory of the source files to be imported.
Directory, File	Select the source files to be imported.
Text file written list of source files	Select the directory of the txt file containing a list of source files.(Option)
Into project	Select the project to add the imported source files.

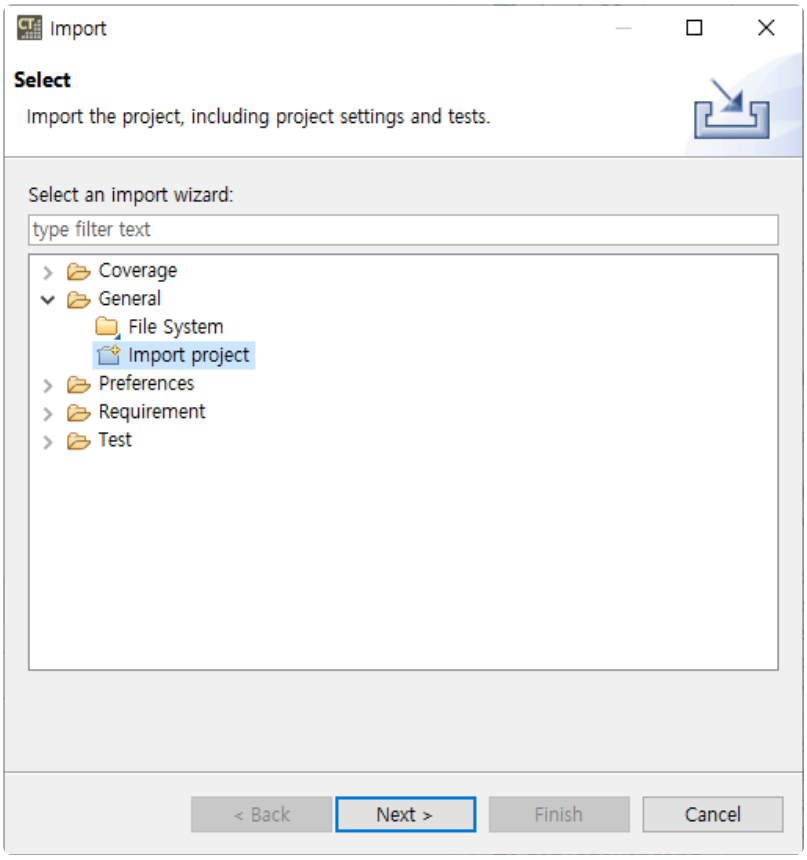
3. Select the module that the source file will be added from the list of modules included in the project and click the [Finish] button.


Import
File System
Select modules to add.
Select module to add:
Default Module
Select All
Deselect All
< Back
Next >
Finish
Cancel

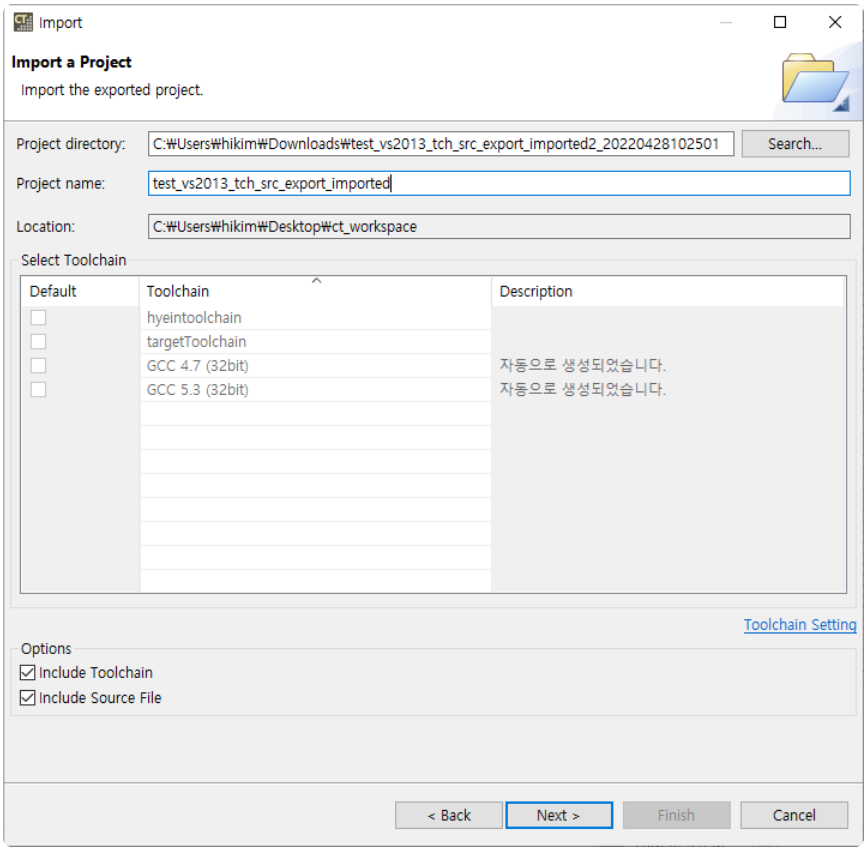
18.3.5. General − Import project

Using the Import Project feature, you can import a project exported from another PC.

1. In the Import Wizard, click [General] > [Import Project] and then click the [Next] button.

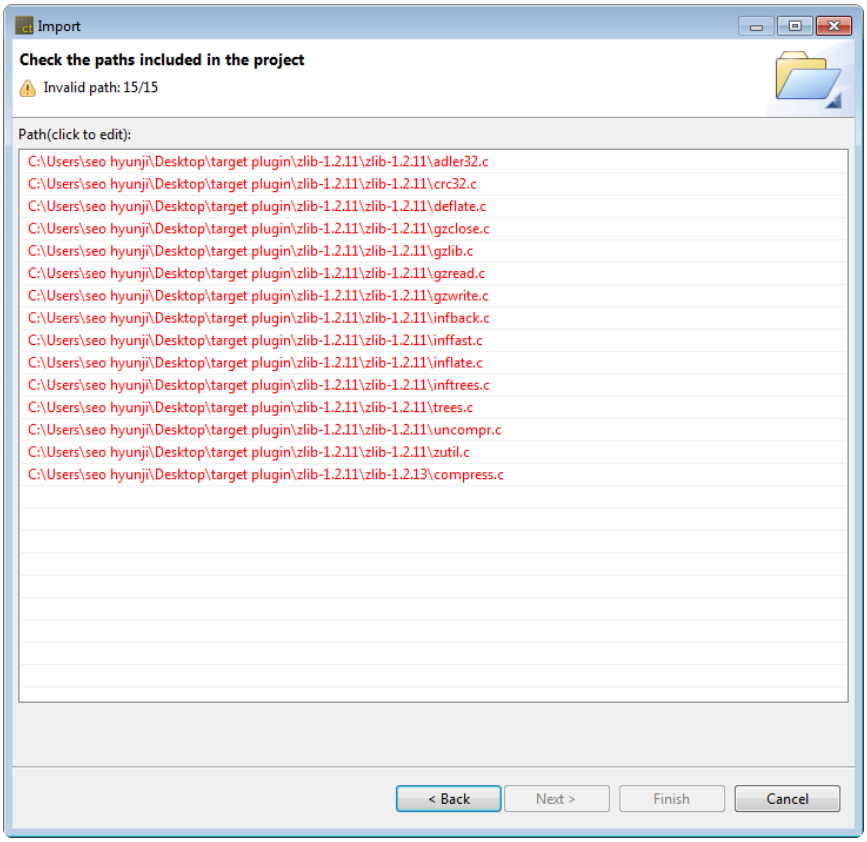


2. Click the [Next] button after entering the exported project directory and selecting the toolchain. If the exported project has toolchain and source code information, Include Toolchain and Include Source Files are automatically selected when you select the directory.

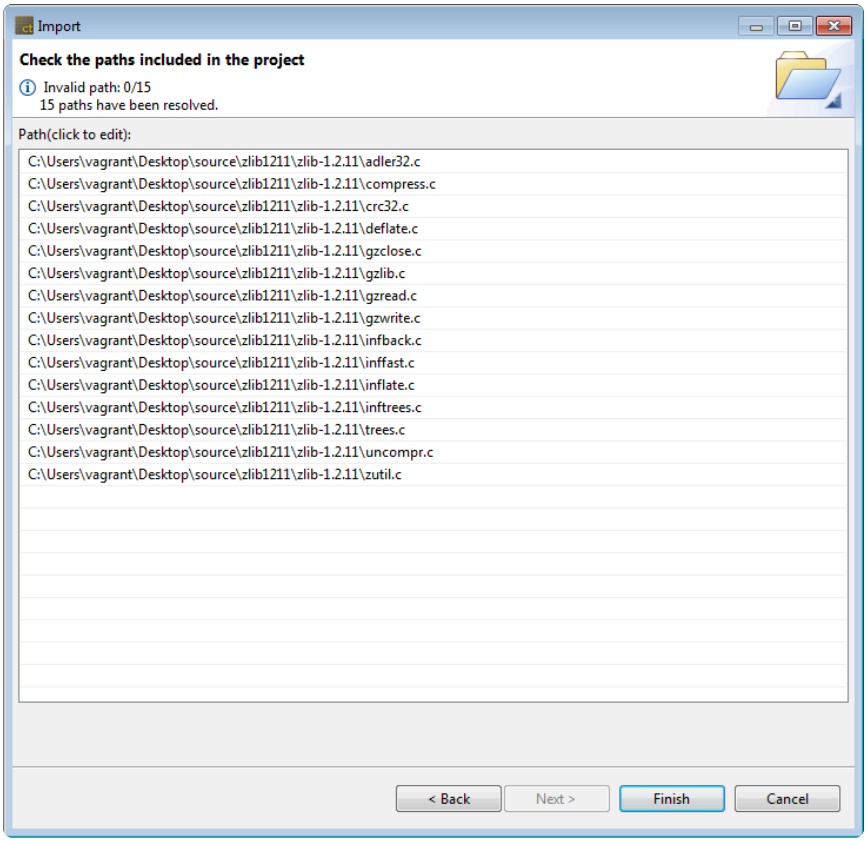


If you don't use the include toolchain option without a suitable toolchain, you need to import the toolchain using the toolchain export/import feature before importing the project.

3. You can check the path information included in the project to be imported. Invalid paths are marked in red and can be modified by clicking with the mouse or pressing the F2 key.



4. If you edit one path, all paths that can be found in the relative path are automatically corrected. The number of modified paths is displayed at the top of the wizard. After editting, click the [Finish] button.

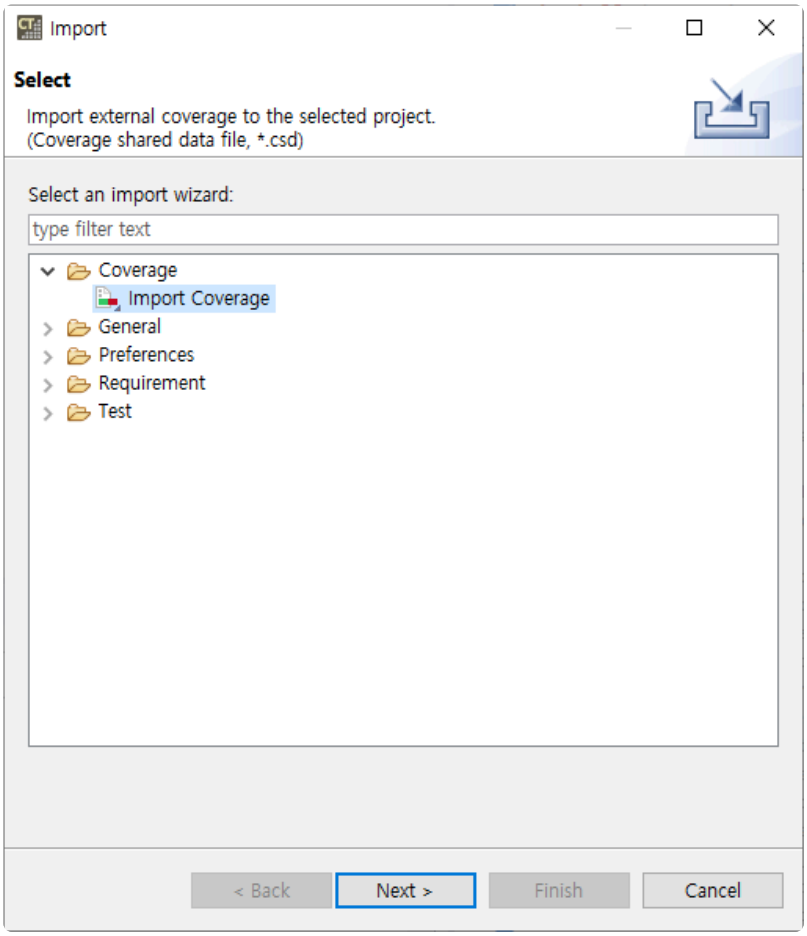


If path is not in absolute path format of Windows OS, the path is not checked for validity.

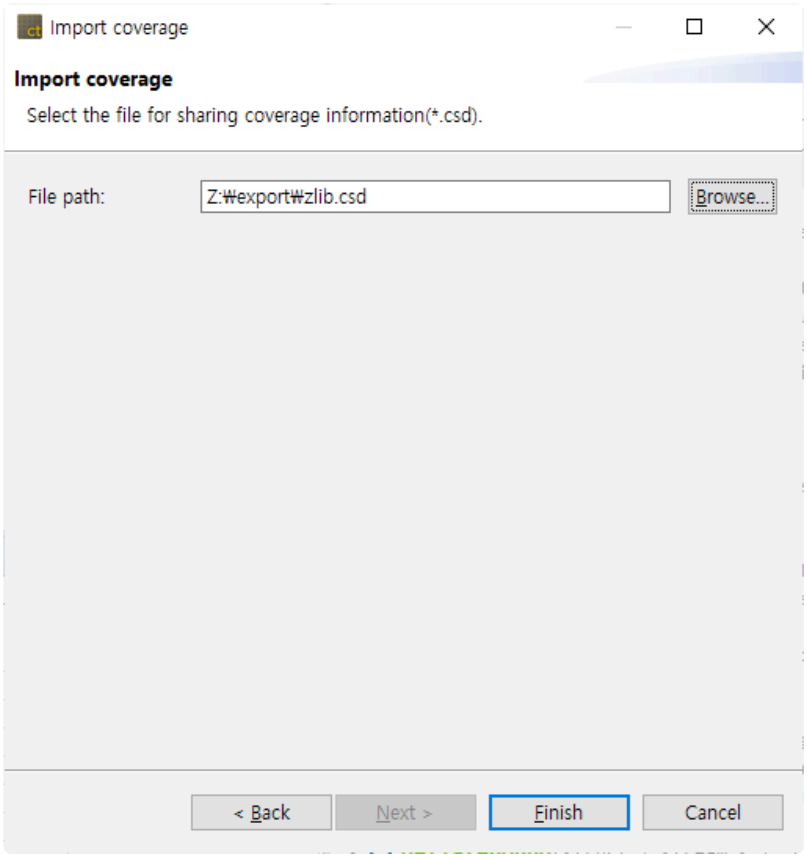
18.3.6. Coverage − Import Coverage

> ! If the version, the coverage type, and the ternary operator option of the coverage file to be imported are the same as those of Controller Tester, the coverage can be imported. For details, please refer to the Coverage Import Guide page of User Guides.

1. In the [Import] wizard, select [Coverage] > [Import Coverage] and click [Next].



2. Enter the path of coverage file to be imported and click [Finish].



The imported coverage information can be checked by selecting [Show full coverage (Include External Coverage)] in the toolbar menu of the Coverage View.

Coverage

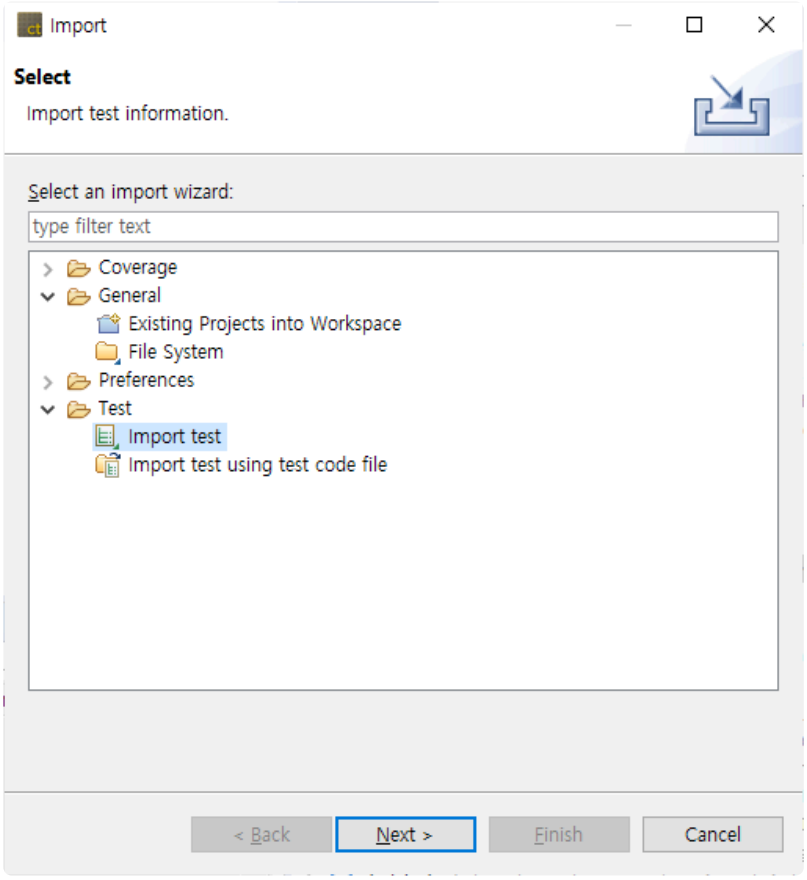
Host coverage information of 'zlib' project

	Target Function	Statement	Branch	MC/DC	Function Call	Functi...
1	_tr_align(struct internal_state *)	72.72% ...	70.00% ...	40.00% ...	100.00...	Y
2	_tr_flush_block(struct internal_state ...	54.76% ...	62.50% ...	36.36% ...	70.00% ...	Y
3	_tr_init(struct internal_state *)	100.00%...	N/A	N/A	100.00...	Y
4	_tr_stored_block(struct internal_stat...	100.00%...	100.00%...	100.00%...	100.00...	Y
5	_tr_tally(struct internal_state *, unsi...	22.22% ...	0.00% (...	0.00% (...	N/A	Y
6	adler32(unsigned long, const unsig...	98.05% ...	87.50% ...	75.00% ...	N/A	Y
7	adler32_combine(unsigned long, u...	100.00%...	100.00%...	100.00%...	N/A	Y
8	bi_flush(struct internal_state *)	100.00%...	100.00%...	100.00%...	N/A	Y
9	bi_reverse(unsigned int, signed int)	100.00%...	100.00%...	100.00%...	N/A	Y
10	bi_windup(struct internal_state *)	100.00%...	100.00%...	100.00%...	N/A	Y
11	build_bl_tree(struct internal_state *)	100.00%...	75.00% ...	50.00% ...	100.00...	Y
Total		**43.02% (...**	**28.51% (...**	**19.07% (...**	**44.11% (...**	**100.0...**

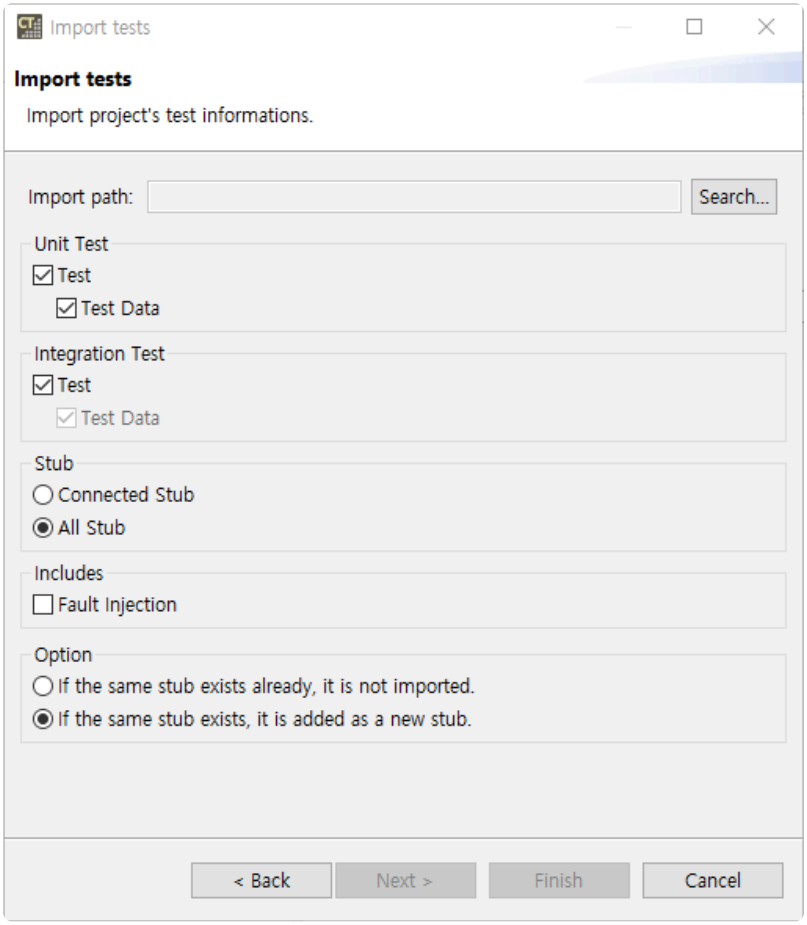
In target testing mode, the coverage of the function including Asm code can not be measured.

18.3.7. Test – Import test

1. In the Import Wizard, select [Test] > [Import Test] and click [Next].

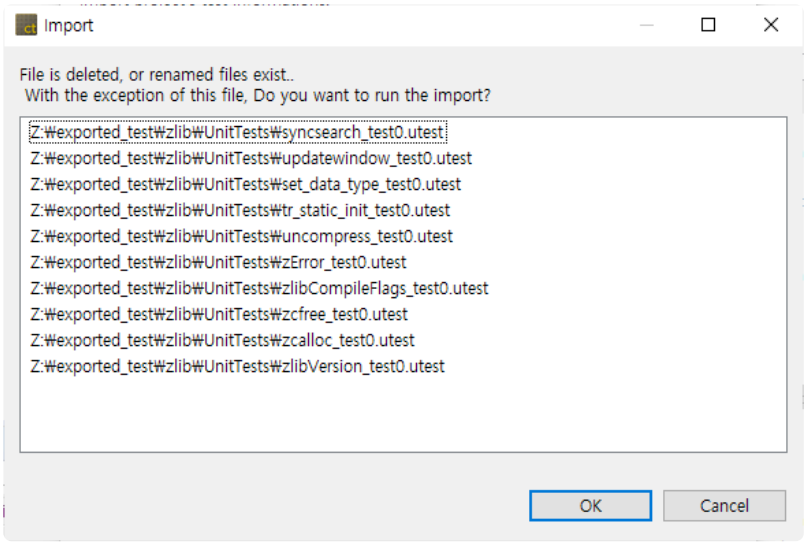


2. Enter the information for the test to be imported and click [Finish].

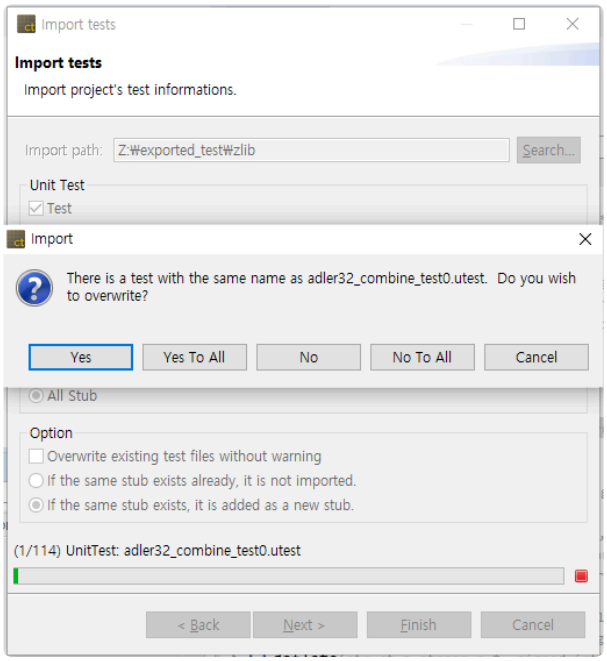


a. In [Import path], enter the path to import the test.
b. In [Unit Test] group, check whether to import [Test] and [Test Data] respectively.
c. In [Integration Test] group, check whether to import [Test].
    - When importing the integration test, it is imported with the test data in a set.
d. In [Stub] group, select the stub to be imported among [Connected stubs] and [All stubs].
    - **Connected stubs**: It imports only the stubs associated with the test you want to import.
    - **All Stub**: It imports all the stubs in the path to import.
e. In [Includes] group, select whether to import with fault injections.
f. In [Option] group, select the method to import if stubs exist.

3. If there are items that the name has been changed or the file has been deleted when importing the test information, the notification windows is displayed, allowing you to select whether to import the information except for that items. This notification window is displayed only if the information was exported from Controller Tester 2.6.14 or later version at a time. If the information is exported from the other menu and previous version, the notification is not displayed.



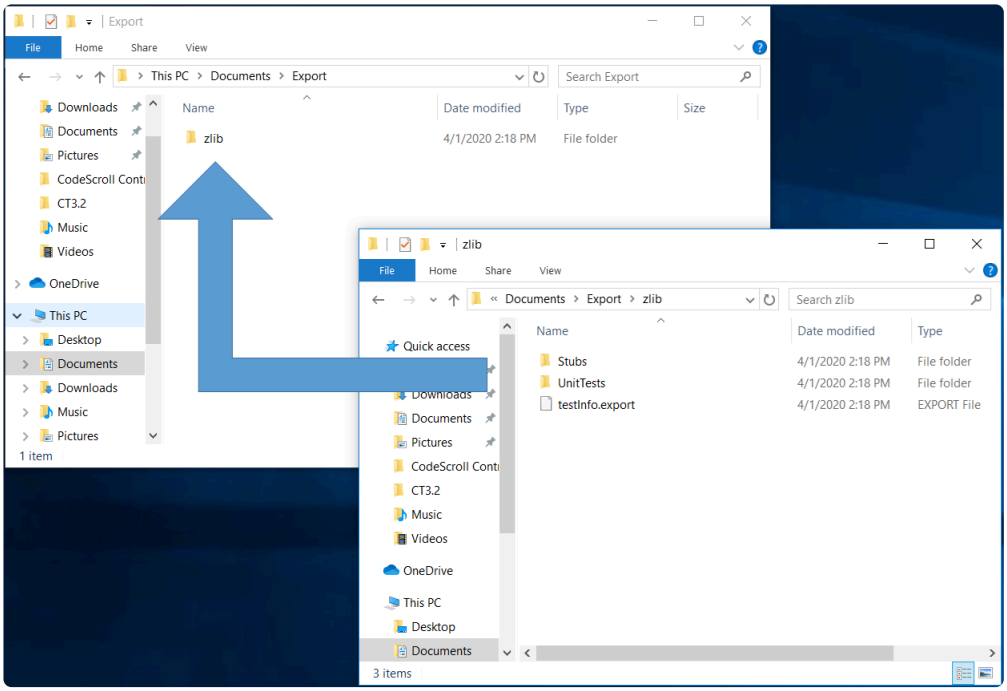
4. If the same test exists in the project, select whether to overwrite the test. Select [Yes] to overwrite and select [No] to cancel the import.



5. After the import has completed, it is applied to the selected project.

Import information exported from Controller Tester 2.6.14 or earlier

1. Enter the path to import the test in [Import path].

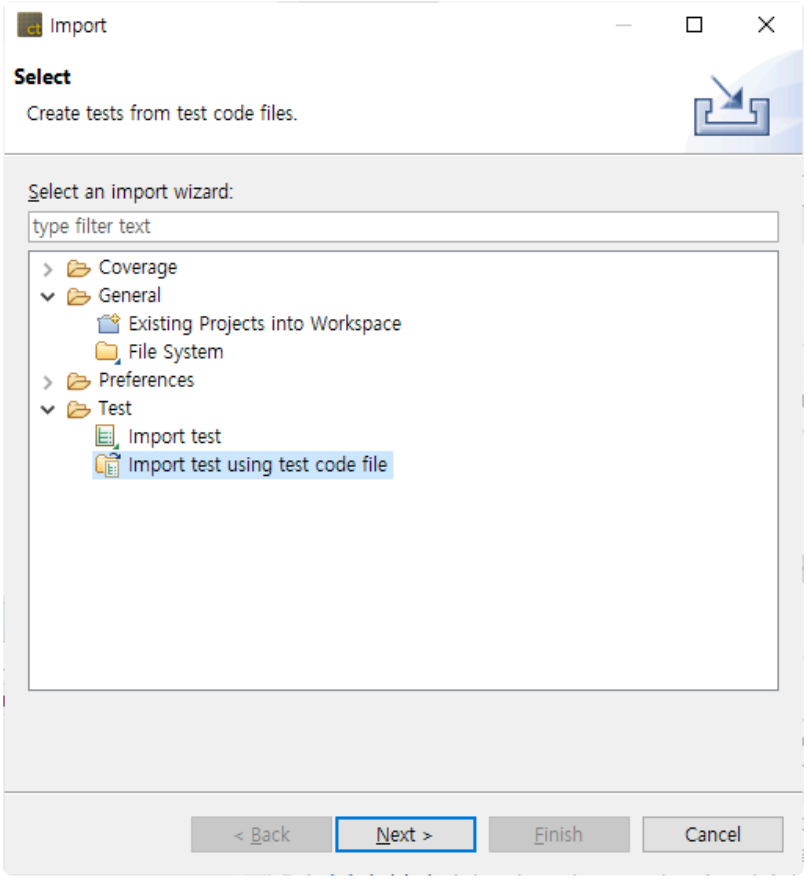


- If [testinfo.export] file exists in the test path to be imported, select the directory containing the file.
  If you import the information exported separately from Controller Tester 2.6.14 or less version, select the higher-level directory containing the information exported.

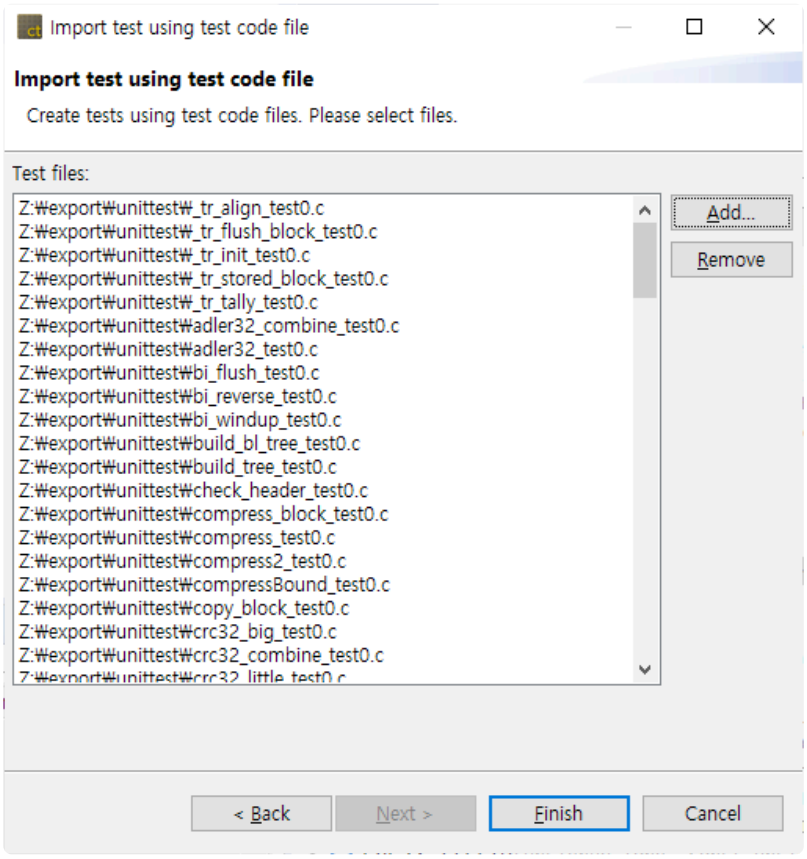
2. In [Unit Test] group, check whether to import [Test] and [Test data] respectively.
3. In [Integration Test] group, check whether to import [Test].
   - The test and data must always be in the same directory, and if it is not, the data is not imported.
4. In [Stub] group, select the stub to be imported among [Connected Stubs] and [All stubs].
   - For the stubs created in Controller Tester 2.3 version, or the stubs migrated to Controller Tester 2.6, if the same stubs exist in the project, the import cannot be carried out.
   - For the stubs created in Controller Tester 2.6 version, if the same stubs exist, you can select either not to import them optionally or to create as a new stub additionally.
   - **There is no feature to link the stubs to the test in Controller Tester 2.3 version, so you need to link directly to the test when migrating to version Controller Tester 2.6 or later.**
5. In [Option] group, select whether to overwrite the existing test files if they exist and how to handle them when the same stub exists.

18.3.8. Test – Import test using test code file

1. In the Import Wizard, select [Test] -> [Import test using test code file] and click [Next].



2. Click [Add…] and select the test to be imported.

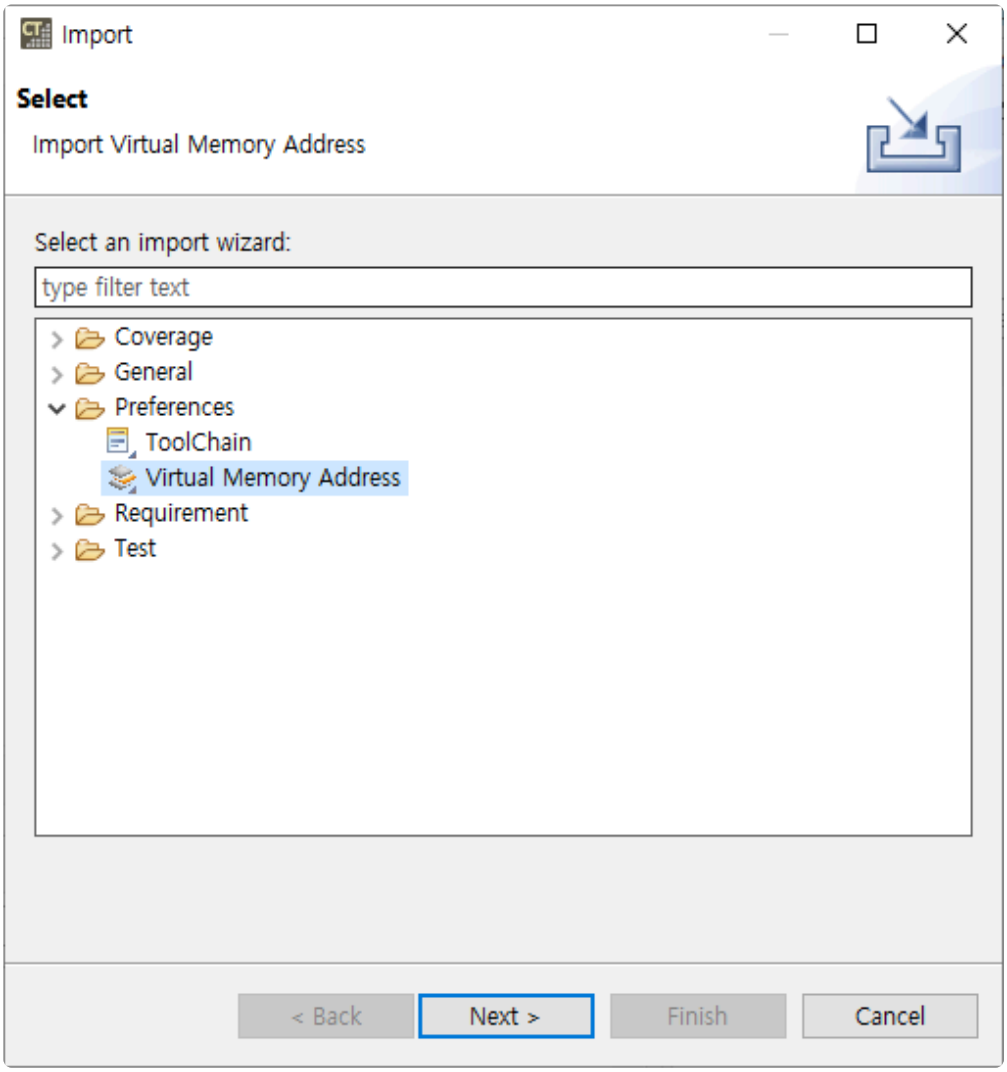


3. Click [Finish] to apply the selected test to the selected project.

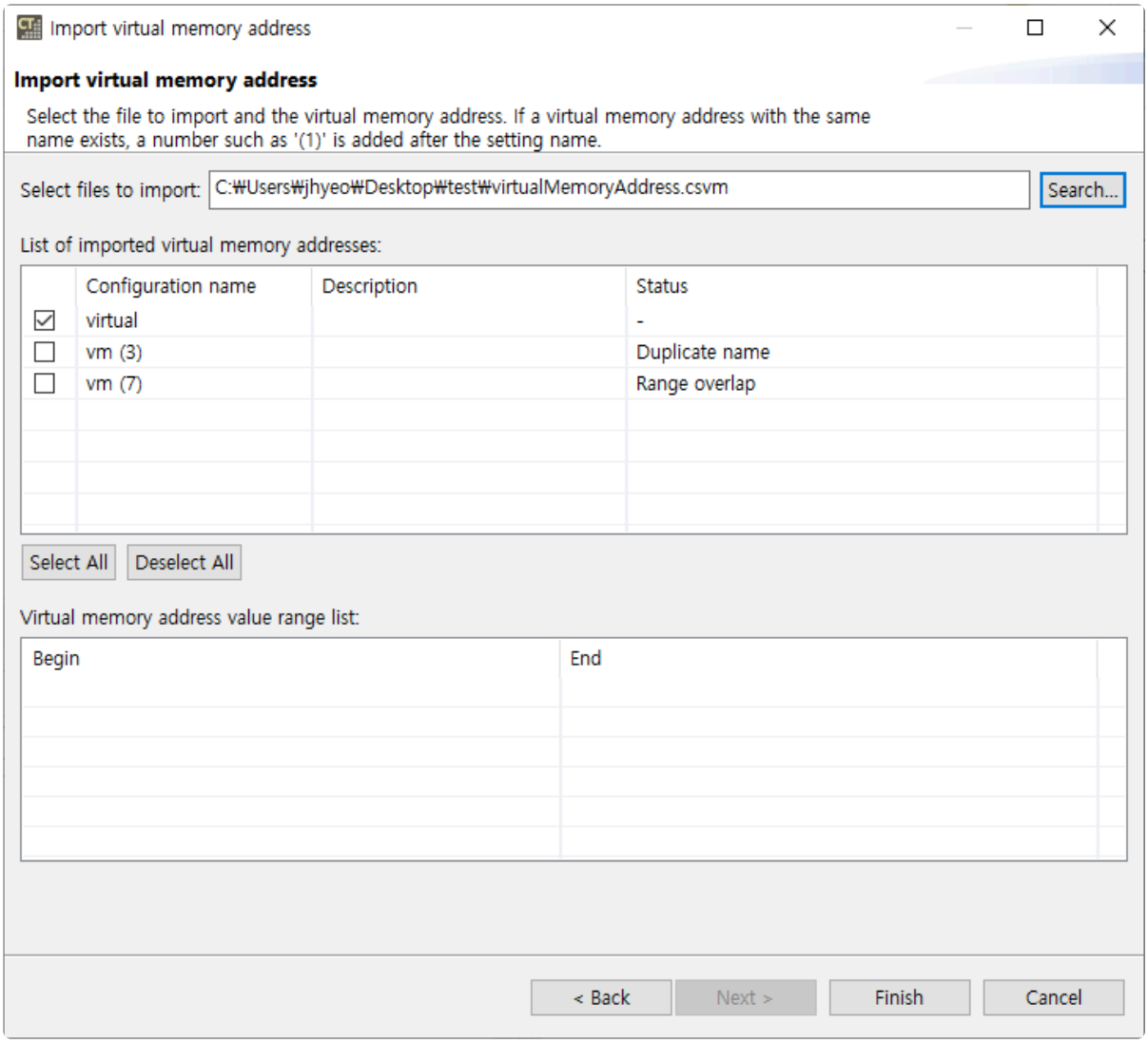
18.3.9. Preferences − Virtual Memory Address

You can import the virtual memory address exported by using the Import function.

1. In the Import Wizard, select [Preferences] > [Virtual Memory Address] and click [Next].

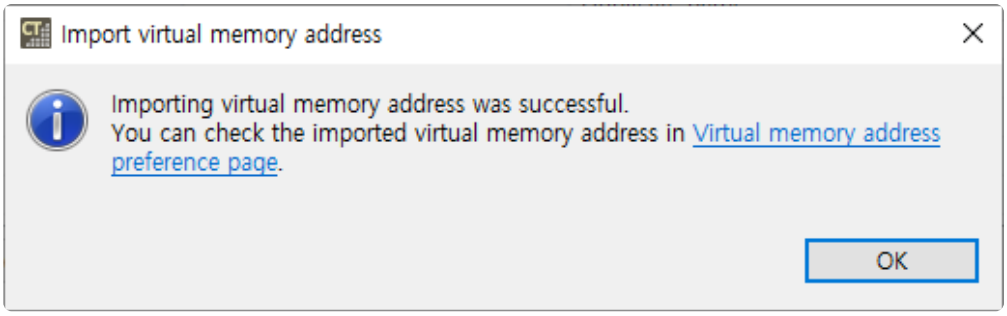


2. When you select a file(*.csvm) to import, a list of virtual memory addresses included in the selected file appears. Select virtual memory addresses to import from the list of virtual memory addresses.



- Status
  - Duplicate name: If there is a virtual memory address with the same name, the import status will be “Duplicate Name”. If you get a virtual memory address that is “Duplicate Name”, it will be imported with the displayed setting name.
  - Range overlap: If there are virtual memory addresses in the same range, the import status will be “Range overlap”. When you get a virtual memory address that is “Range overlap”, the virtual memory address in the same range is imported.

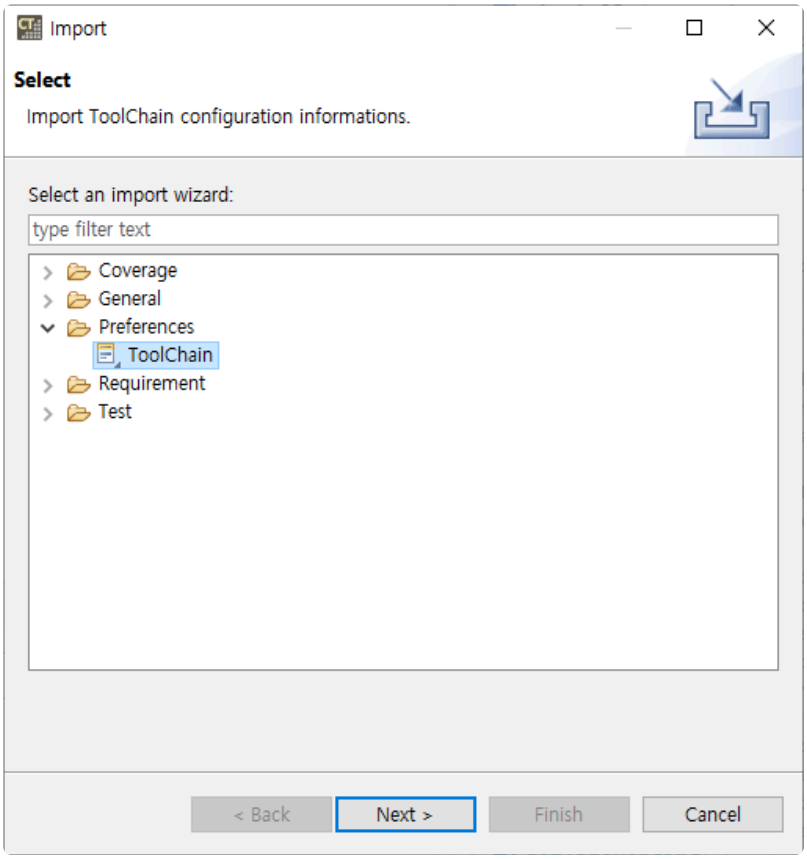
3. Click the [Finish]. You can click the link to check the imported virtual memory address on the [Preference] page.



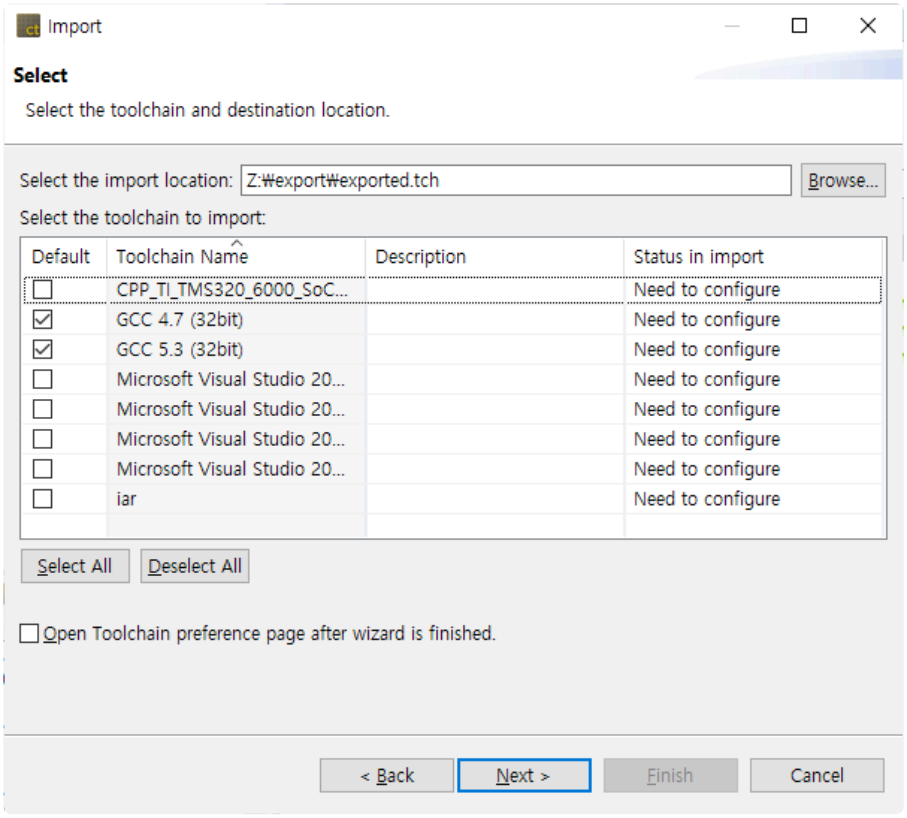
18.3.10. Preferences – ToolChain

You can import the toolchain information exported by using the Import function.

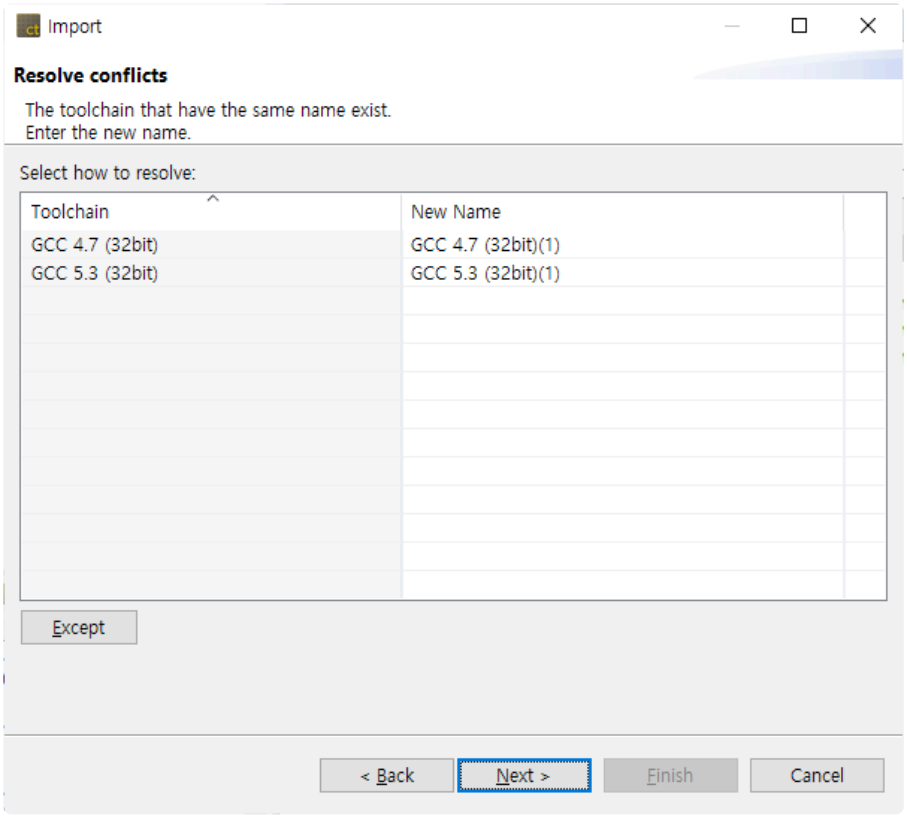
1. In the Import Wizard, select [Preferences] > [ToolChain] and click [Next].



2. If you select the file (*.tch) to be imported, the list of toolchains included in the selected file are displayed. Select the toolchains to be imported from the toolchain list.



3. If the toolchain name is duplicated with the name of the existing toolchain, the import status becomes “Need to modify”. You can change the toolchain name to be imported in the [Select how to resolve] window displayed when clicking [Next] button.

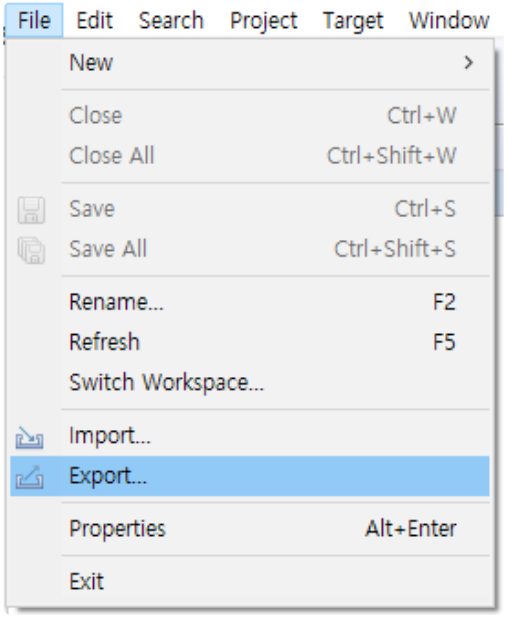


4. Click [Finish].

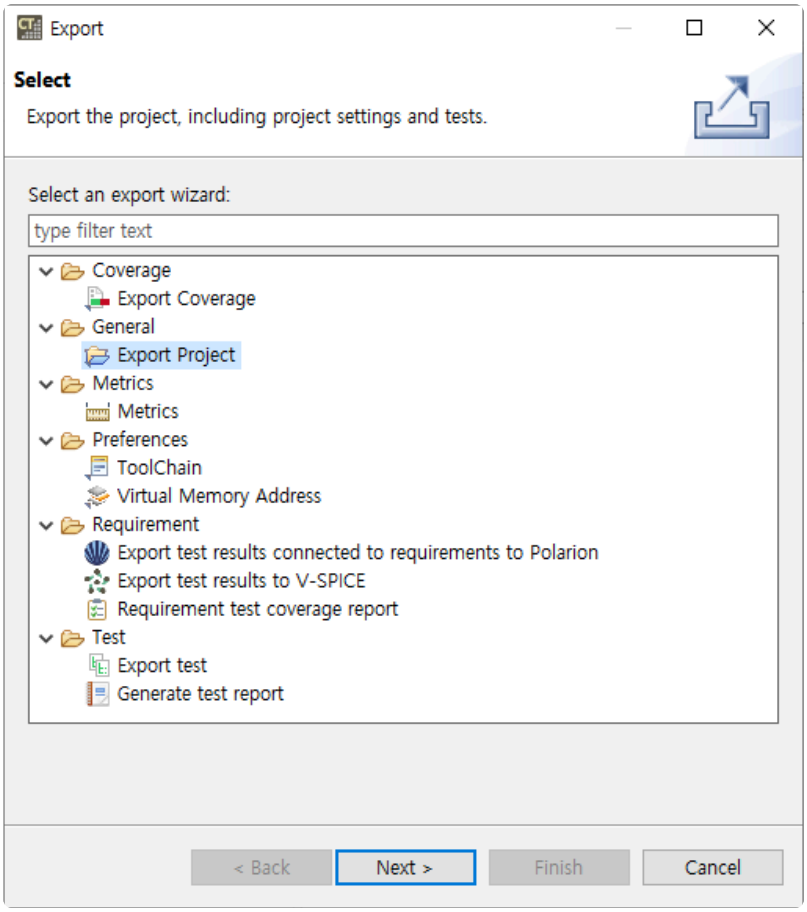
18.4. Export

You can export projects, tests, toolchains, and etc in Controller Tester.

1. Select [File] > [Export] from the main menu or [Export] from the dashboard to open the Export Wizard.



2. Select a item to export and click [Next].



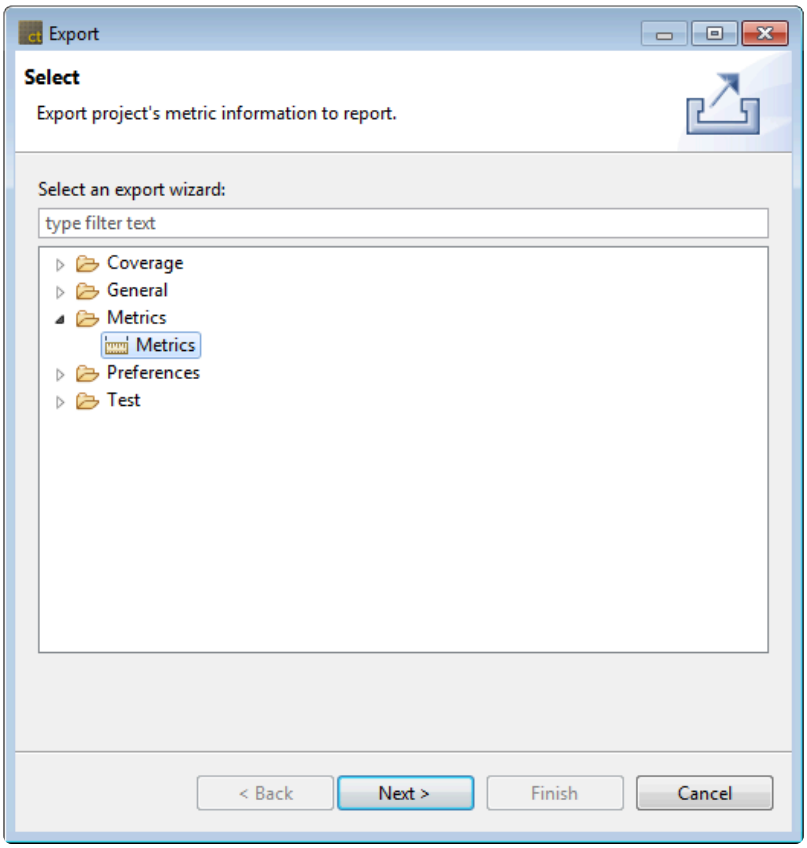
Parent item	Child item	Description

Metrics	Metrics	Export the project's metrics measurement results in a report format.
Requirement	Export test results connected to requirements to Polarion Requirement test coverage report	Export the results of tests associated with requirements to a requirement management tool or to a CSV file.
General	Export project	Export Controller Tester projects.
Coverage	Export Coverage	Export the coverage results in a csd file.
Test	Export test Generate test report	Export tests or generate test reports.
Preferences	ToolChain Virtual Memory Addres	Export virtual memory address or toolchain information.

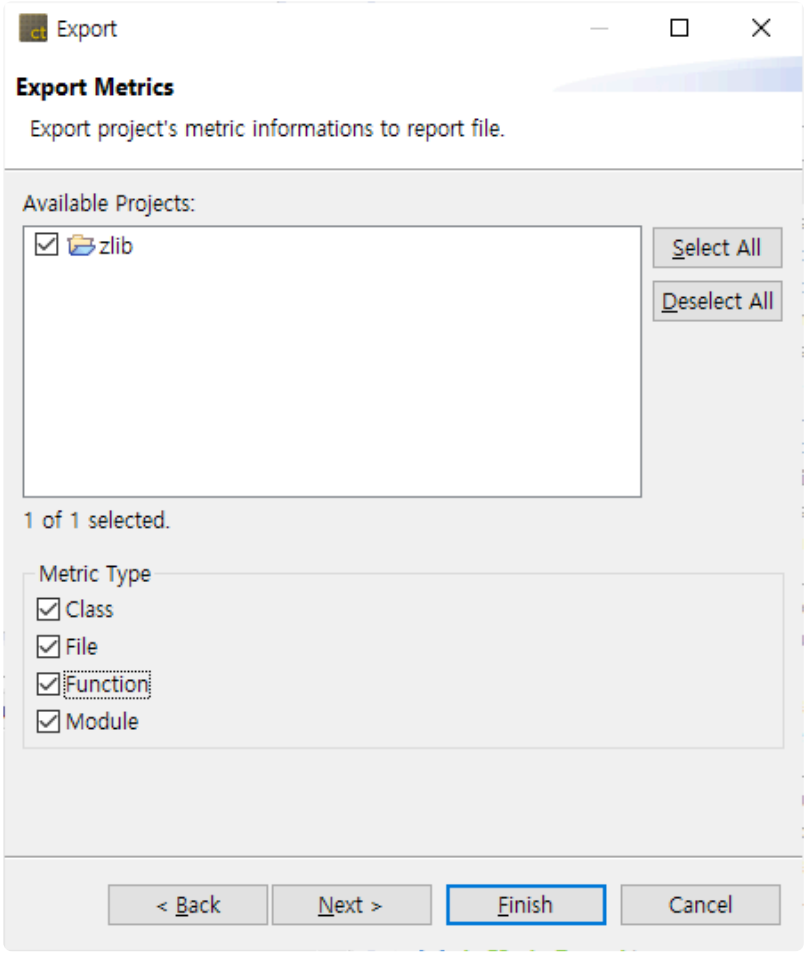
18.4.1. Metrics – Metrics

It exports the metric result of the project in a report file.

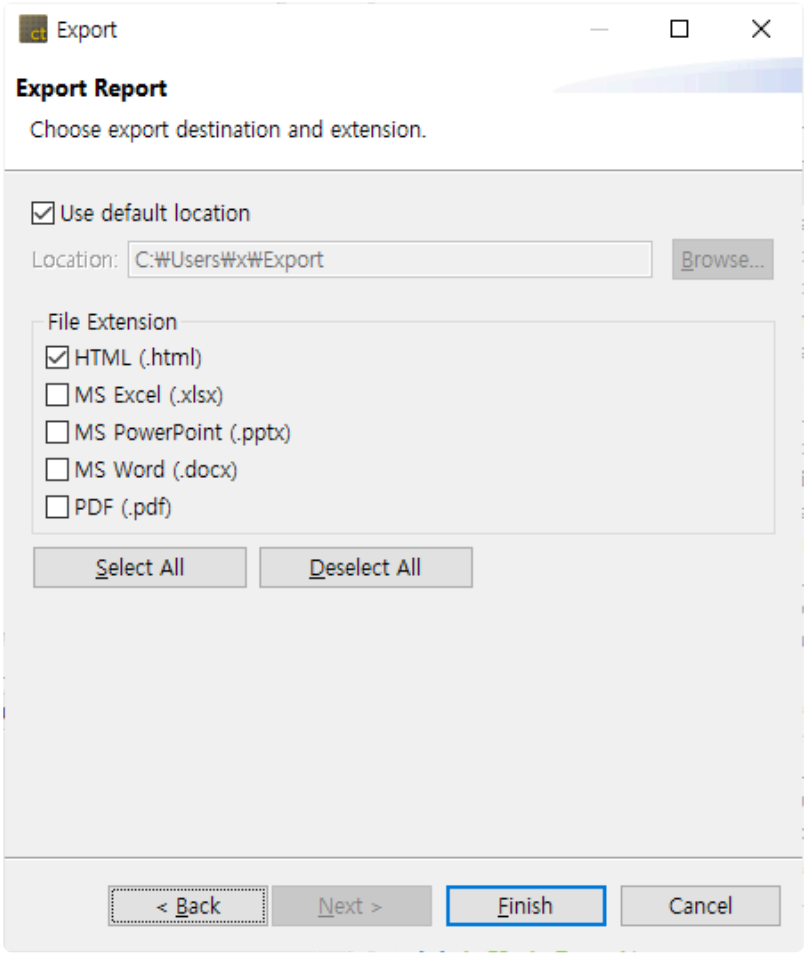
1. In the Export Wizard, click [Metric] -> [Metric] and then click the [Next] button.



2. Select the project and metric type to export metric information and click the [Next] button.



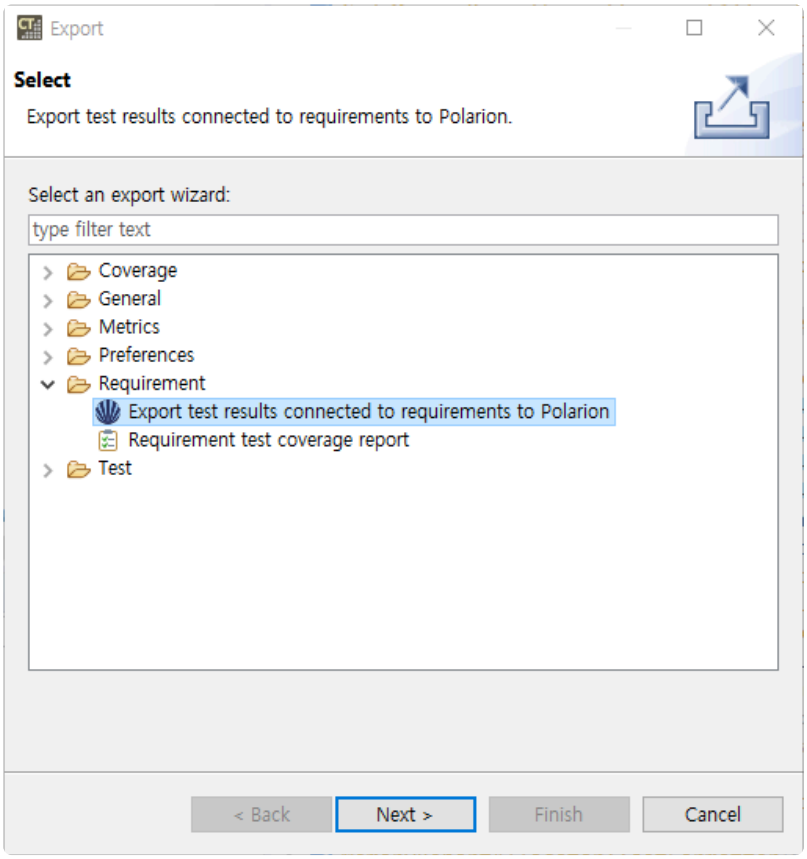
3. Select the path and report format to export the report and click the [Finish] button.



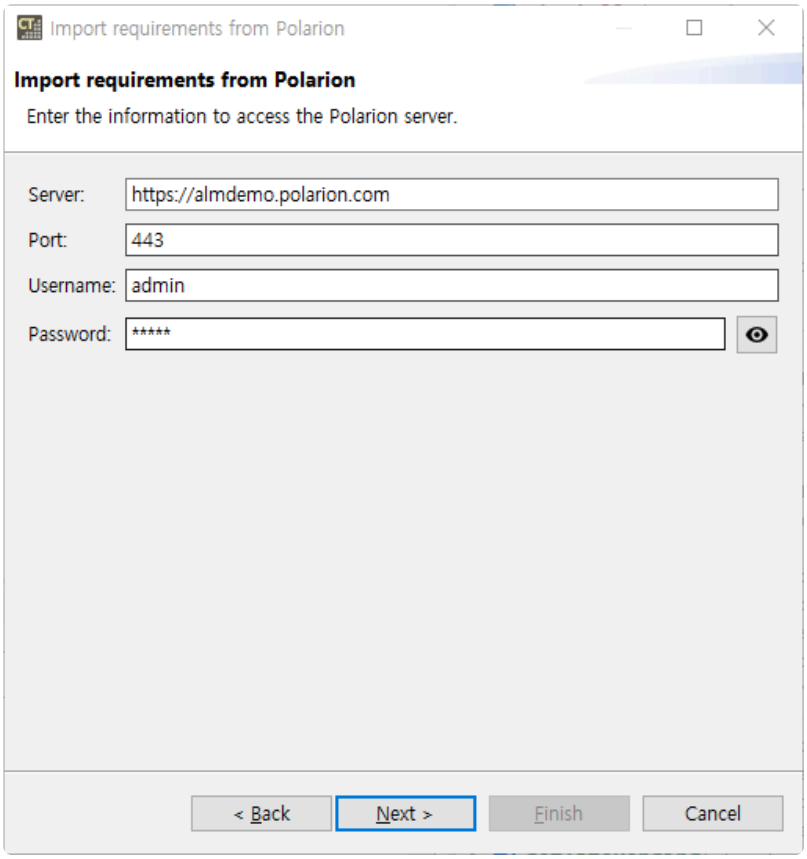
18.4.2. Requirement − Export test results connected to requirements to Polarion

You can export test results, which is connected to requirement, to Polarion.

1. In the Export Wizard, select [Requirement] -> [Export test results connected to requirements to Polarion] and click [Next].

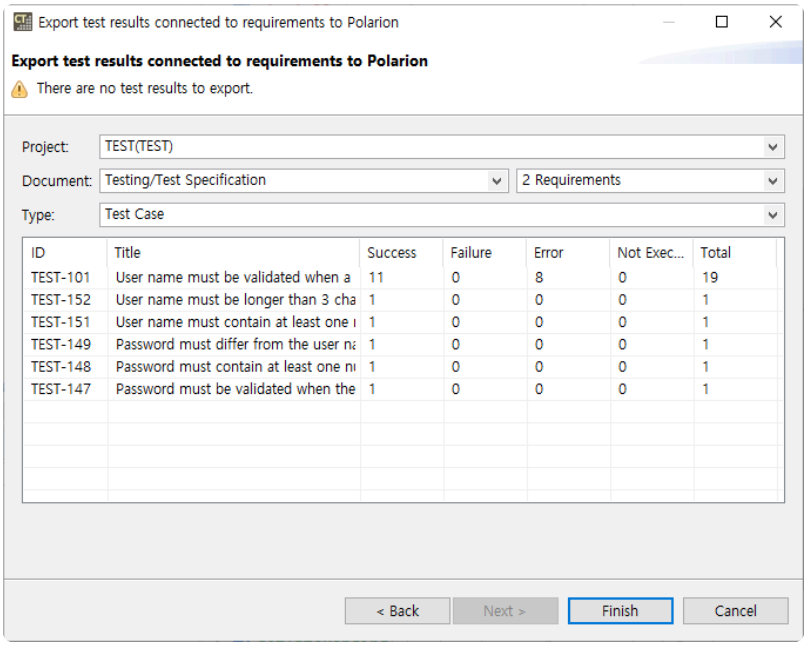


2. Enter the information to connect with Polarion server and click [Next]. You can see the entered password to click button.



- The information of Polarion's demo server is entered in server and port.
- If you use a local server, enter `http://ip_address` in server and port number of local server in port.

3. Enter the information of target to export test result and click [Finish].

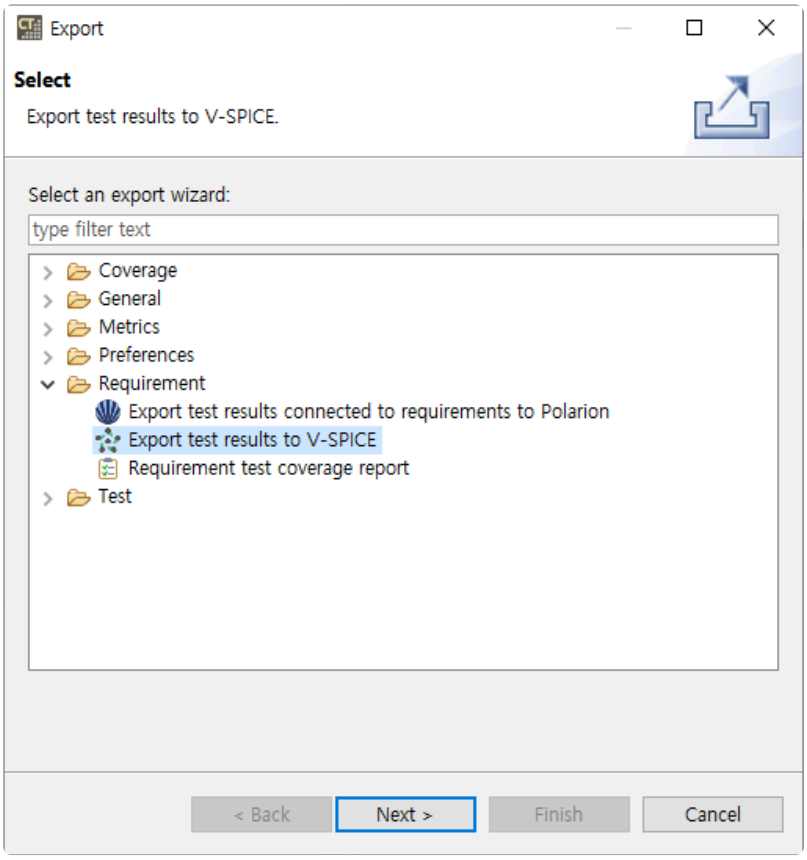


- Project : Select a project to export test results. The project from which the requirements were imported is selected by default.
- Document : Select a document and a content to export the test results. You can export test results without selecting a document.
- Type : Select a type of work item to export the test results.

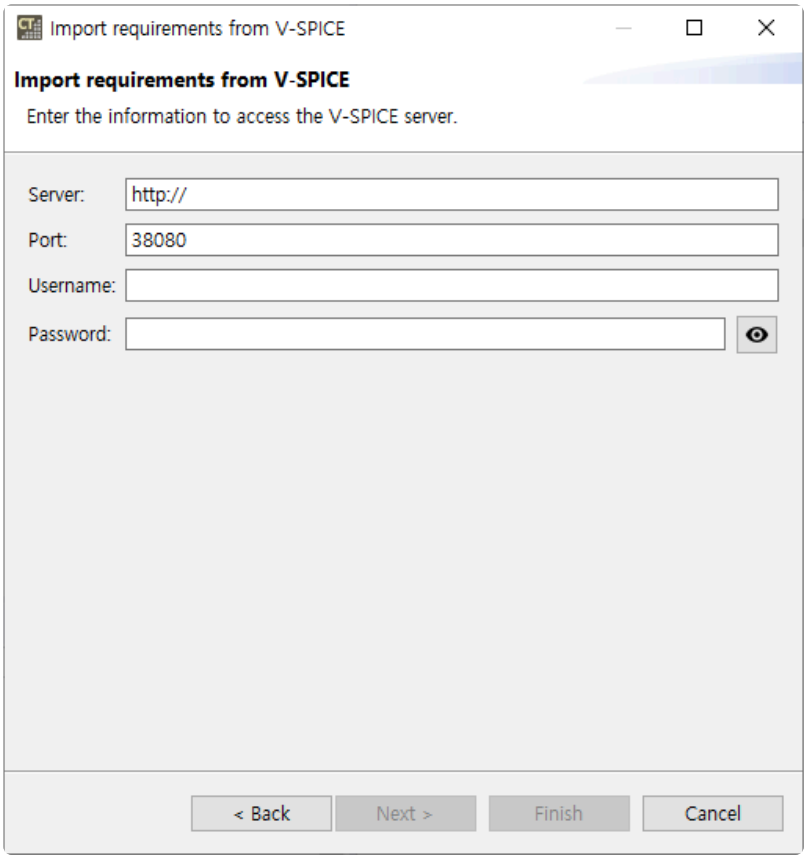
18.4.3. Requirement − Export test results to V-SPICE

You can export test results to V-SPICE.

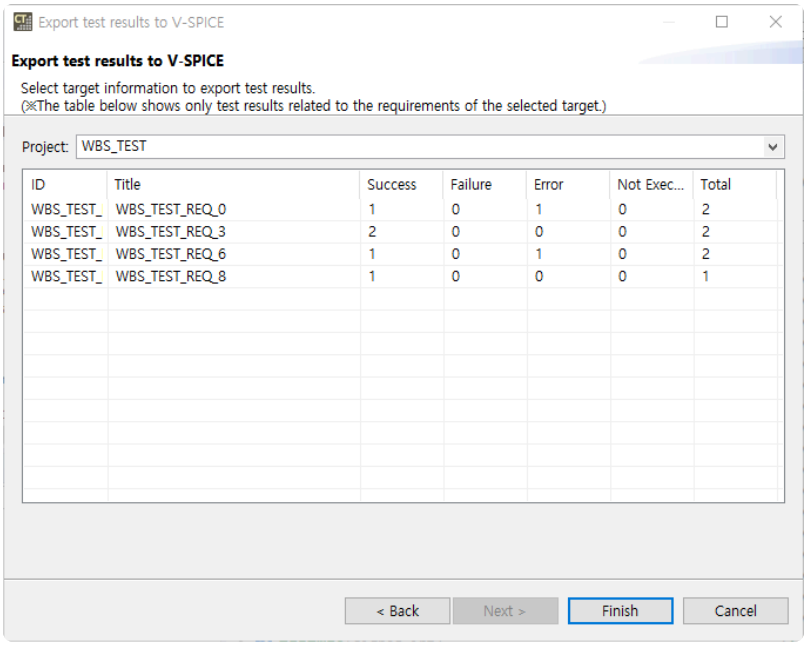
1. In the Export Wizard, select [Requirement] -> [Export test result to V-SPICE] and click [Next].



2. Enter the information to connect with V-SPICEserver and click [Next]. You can see the entered password to click button.



3. Enter the information of target to export test result and click [Finish].



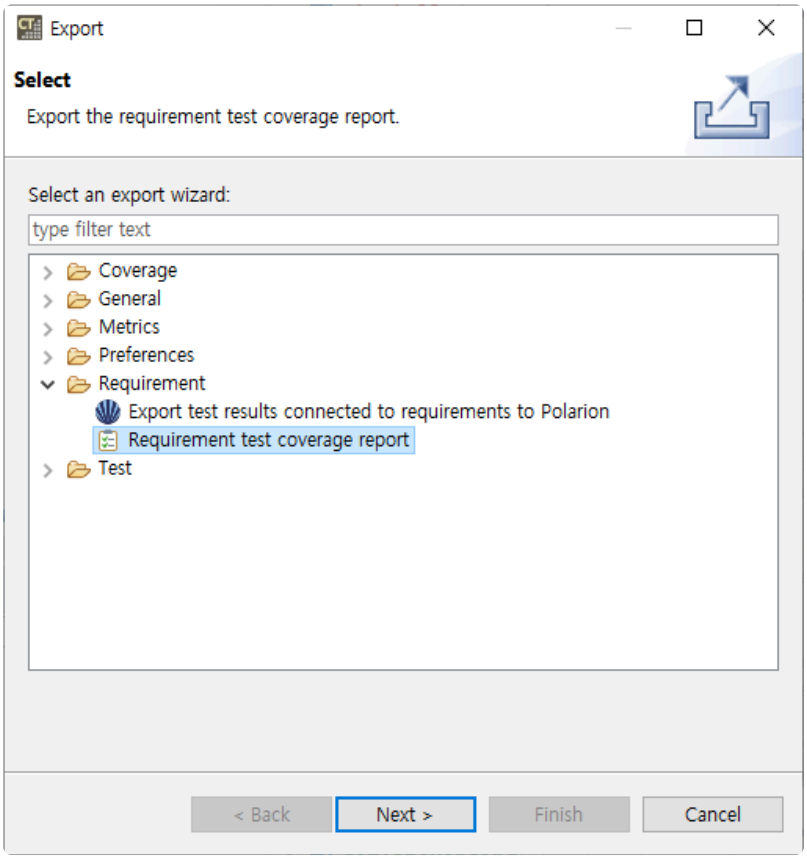
- Project : Select a project to export test results. The project from which the requirements were imported is selected by default.

If there are no tests connected with a requirement in the selected V-SPICE project, the list of requirements is not displayed. All test information is exported even if the requirements list is not displayed.

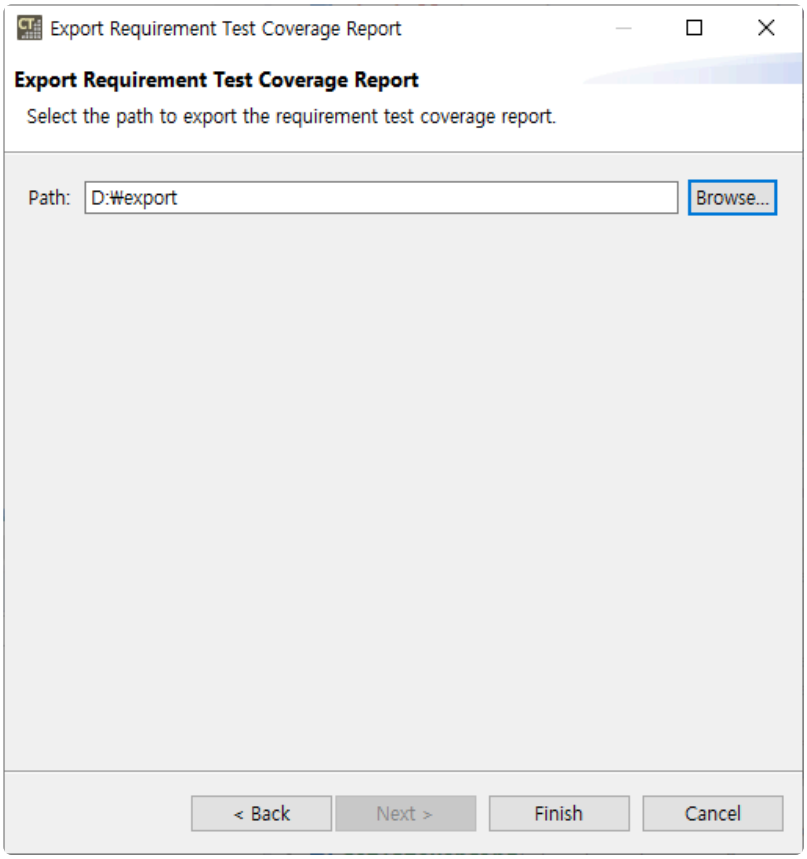
18.4.4. Requirement − Requirement test coverage report

You can generate requirement test coverage report in excel format.

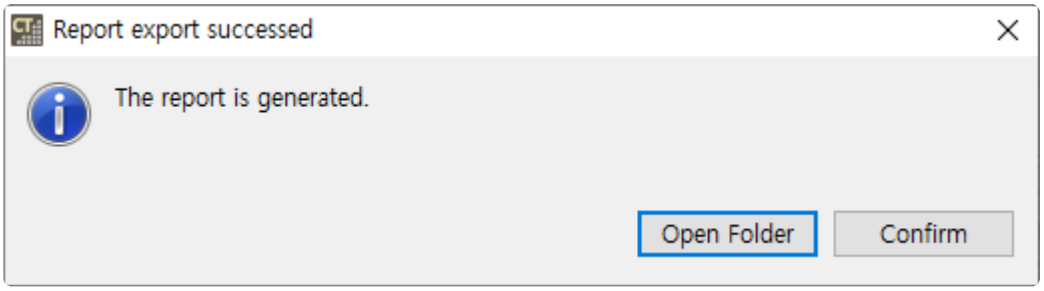
1. In the Export Wizard, select [Requirement] -> [Requirement test coverage report] and click [Next].



2. Select the path to export requirement test coverage report and click [Finish].



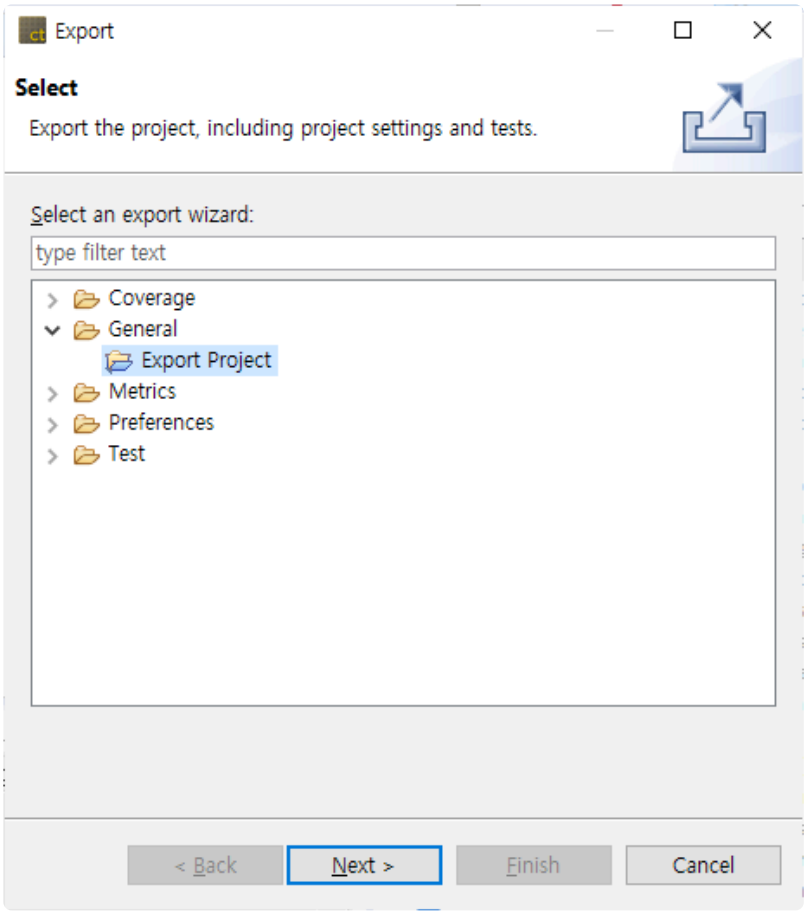
3. You can check the exported report by clicking [Open Folder] .



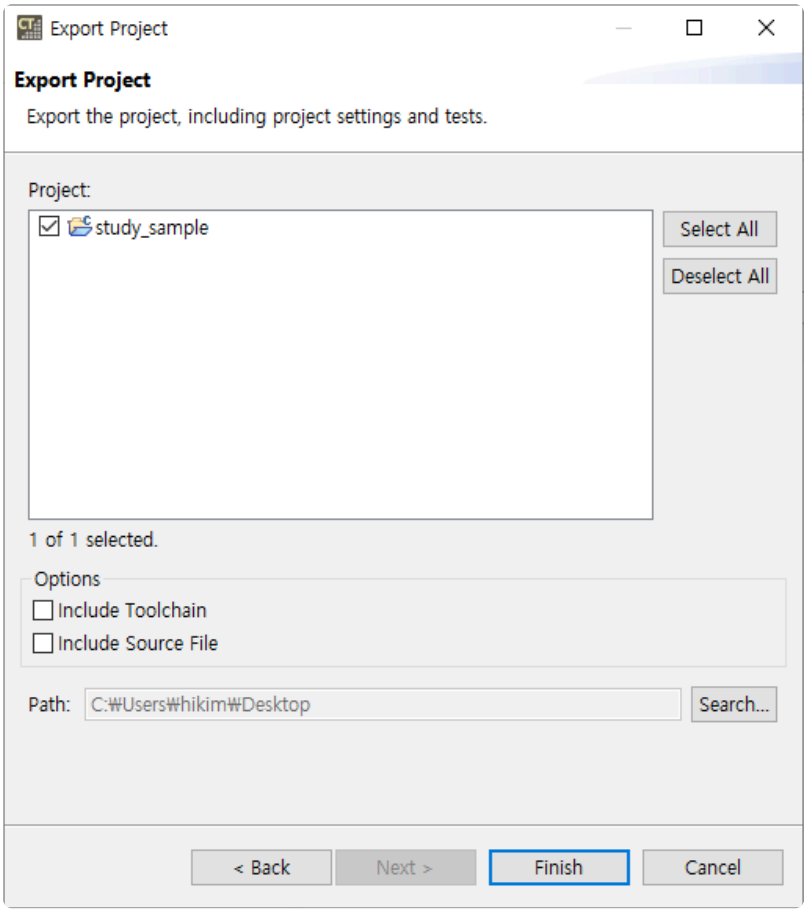
18.4.5. General − Export project

The project can be exported including the project settings and the test. The exported project can be imported with [Import Project].

1. In the Export Wizard, select [General] > [Export Project] and click [Next].



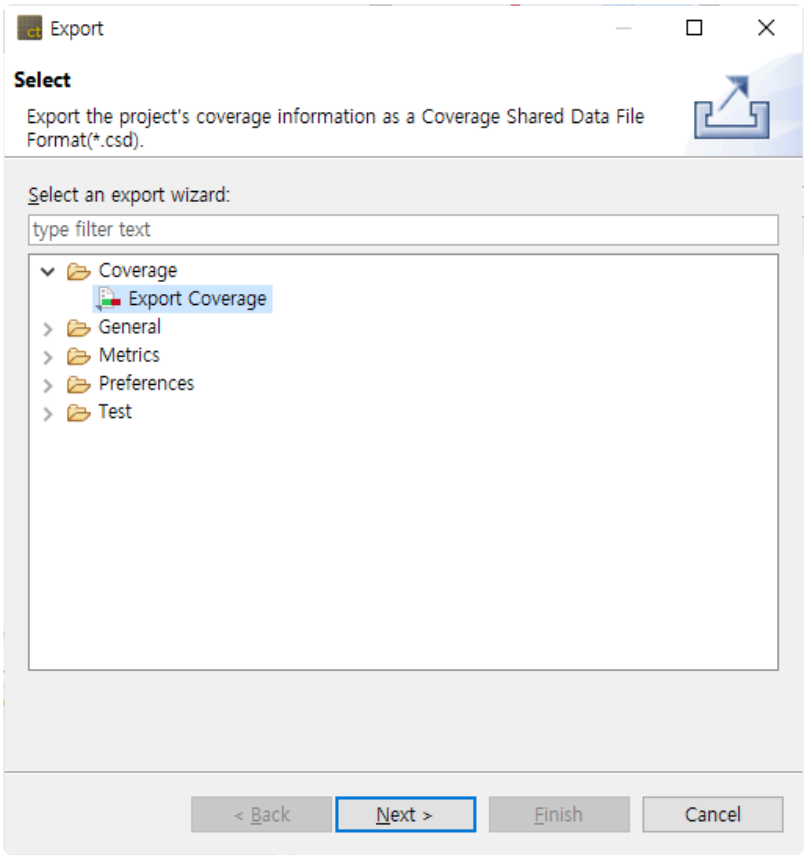
2. Select the project to be exported and the export path and click [Finish]. When you export a project, you can export it including the toolchain and source files.


Export Project
Export Project
Export the project, including project settings and tests.
Project:
study_sample
Select All
Deselect All
1 of 1 selected.
Options
Include Toolchain
Include Source File
Path:
C:₩Users₩hikim₩Desktop
Search...
< Back
Next >
Finish
Cancel

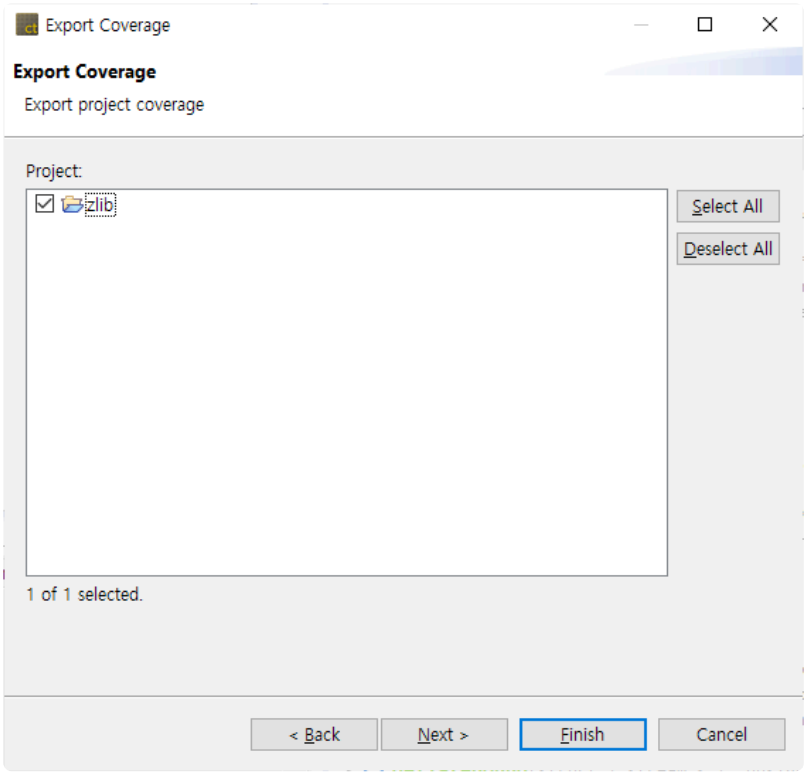
18.4.6. Coverage − Export Coverage

You can export the coverage information measured in the project as a file in Coverage Shared Data (*.csd) format. The exported coverage can be imported with [Import Coverage].

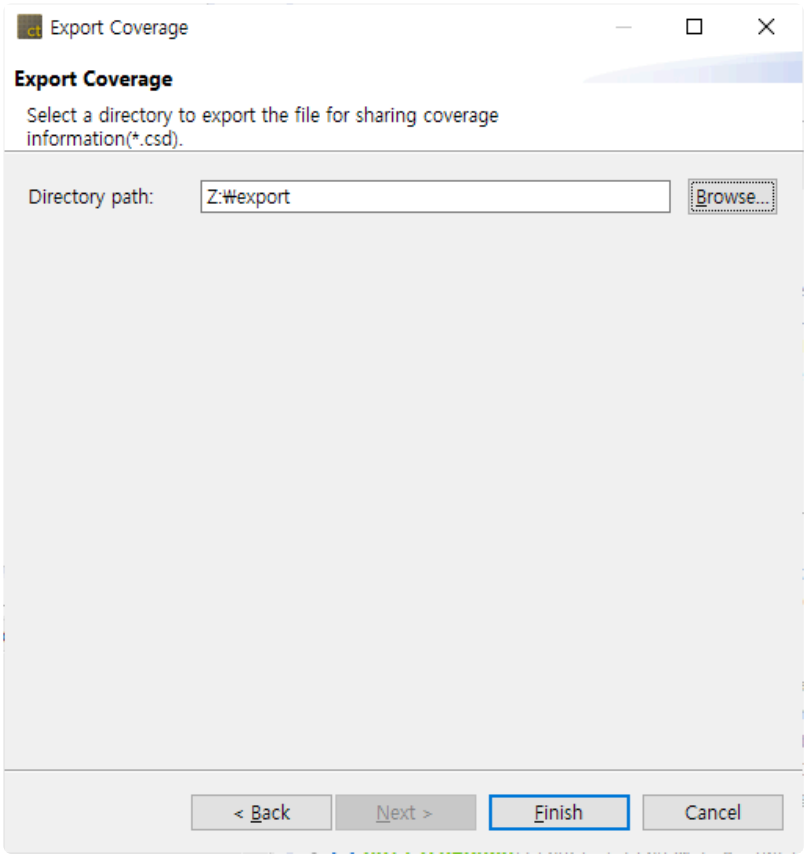
1. In the Export Wizard, select [Coverage] -> [Export Coverage] and click [Next].



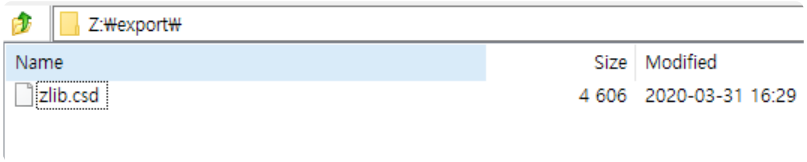
2. Select the project that you want to export coverage information and click [Next].



3. Enter the directory path to export and click [Finish].



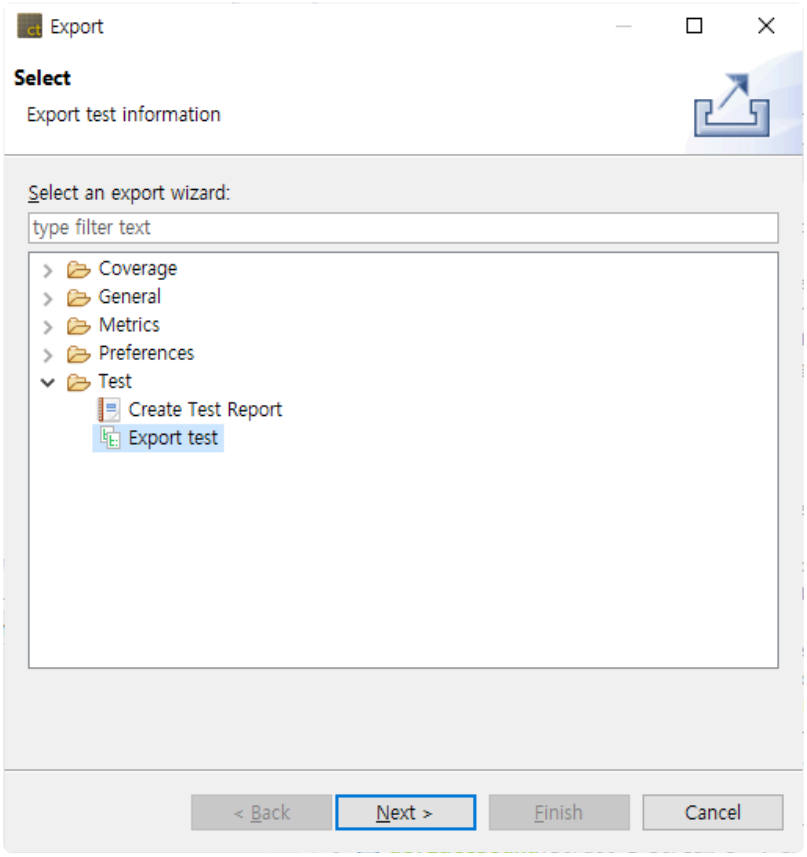
4. A file with the same name as the project name is created.



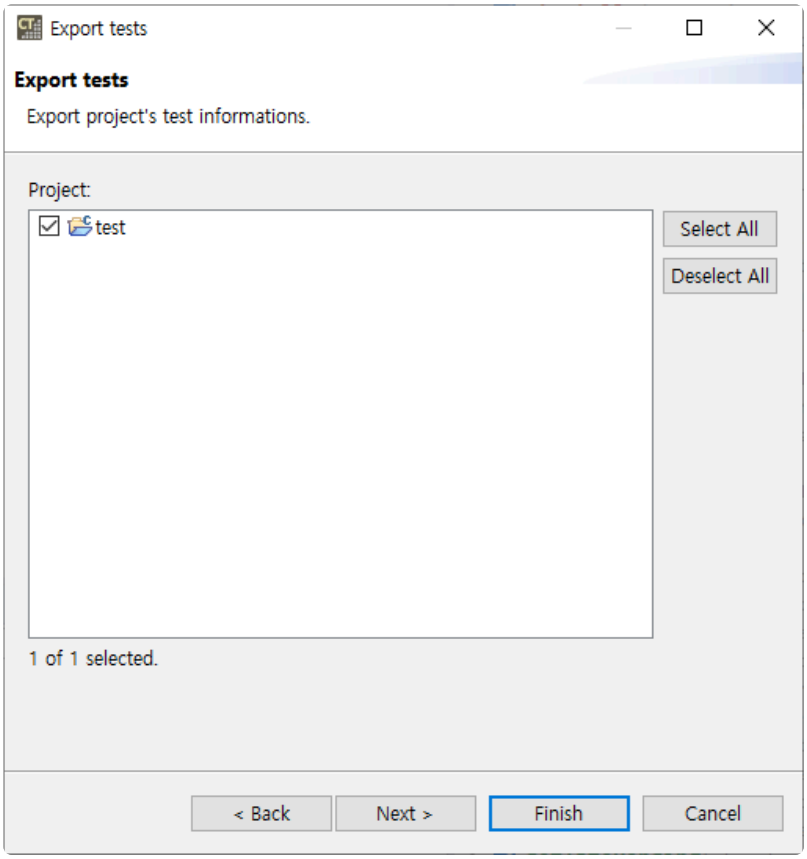
18.4.7. Test – Export test

The test information created in the project can be exported at once. Before a project is not analyzed or if there is no test information (Unit/Integration Test, Stub) created, the export cannot be carried out.

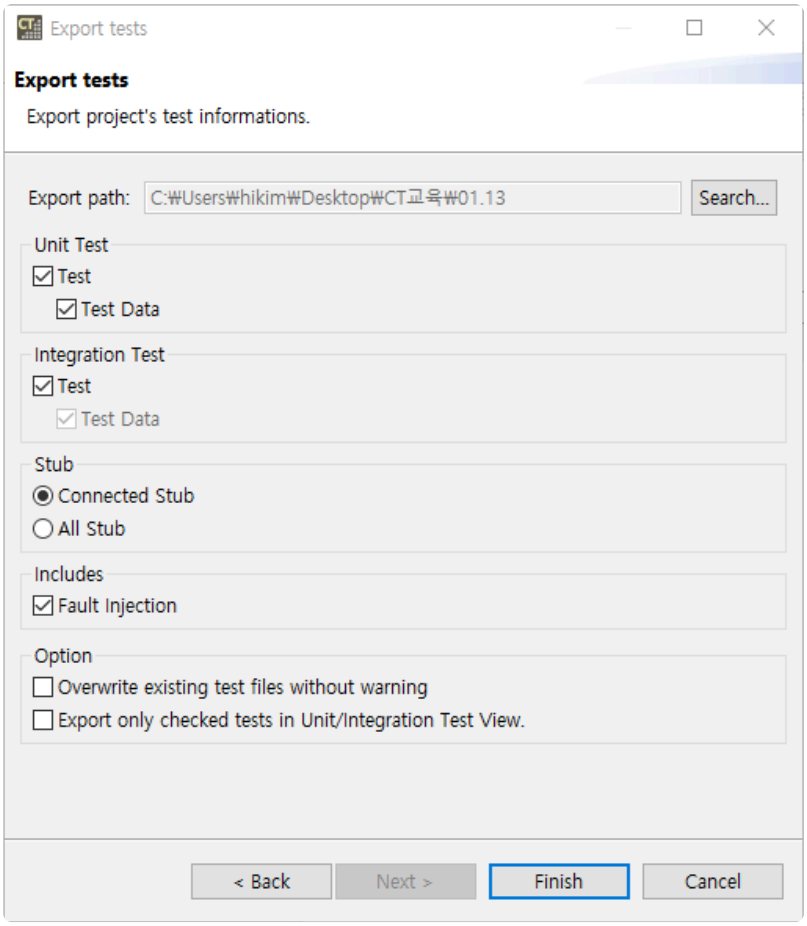
1. Select [Test] > [Export Test] in the Export Wizard and click [Next].



2. Select the project that tests are exported and click [Next].



3. Select the export path, the unit test items, the integration test items, the stub items, the fault injection and the options, and click [Finish].



   a. In [Export path], enter the path to export the test.
   b. In [Unit Test] group, check whether to export [Test] and [Test Data] respectively.

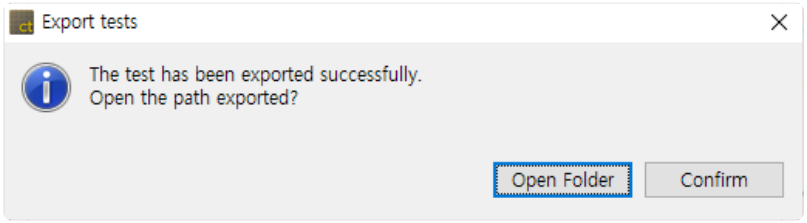
c. In [Integration Test] group, check whether to export [Test].
   - When exporting the integration test, it is exported with the test data in a set.

d. In [Stub] group, select the stub to be exported among [Connected stubs] and [All stubs].
   - **Connected stubs**: The stubs connected with the test to be exported. If the stubs have no link with the test or the test to be exported is not selected, it does not export that stubs.
   - **All Stub**: It exports all the stubs created in the project.

e. In [Includes] group, select whether to export with fault injections.

f. In [Option] group, choose whether to overwrite the existing test file when they exist, and whether to export only the tests checked in the Unit/Integration Test view.

4. Click [Open Folder] button in the message “Test has been exported successfully.” to check the test exported in the folder where it is saved, or click [Confirm] to finish the export.

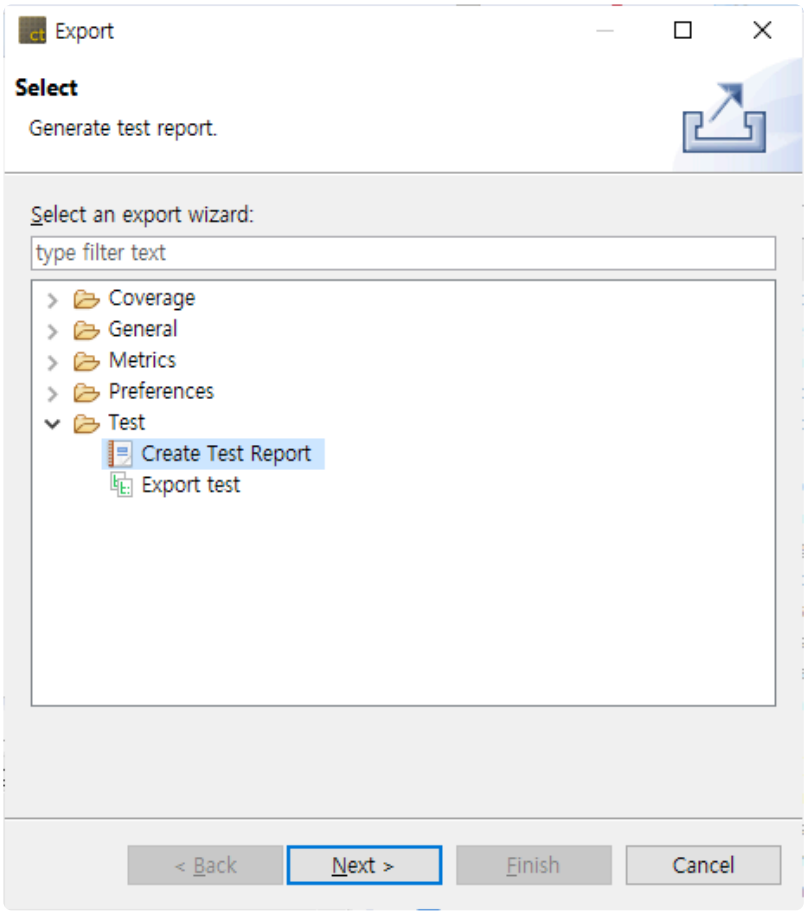


From Controller Tester 2.6.14 or later version, when exporting tests, [testinfo.export] file is created in the project directory exported.
(**testinfo.export**: the file that records the exported information)

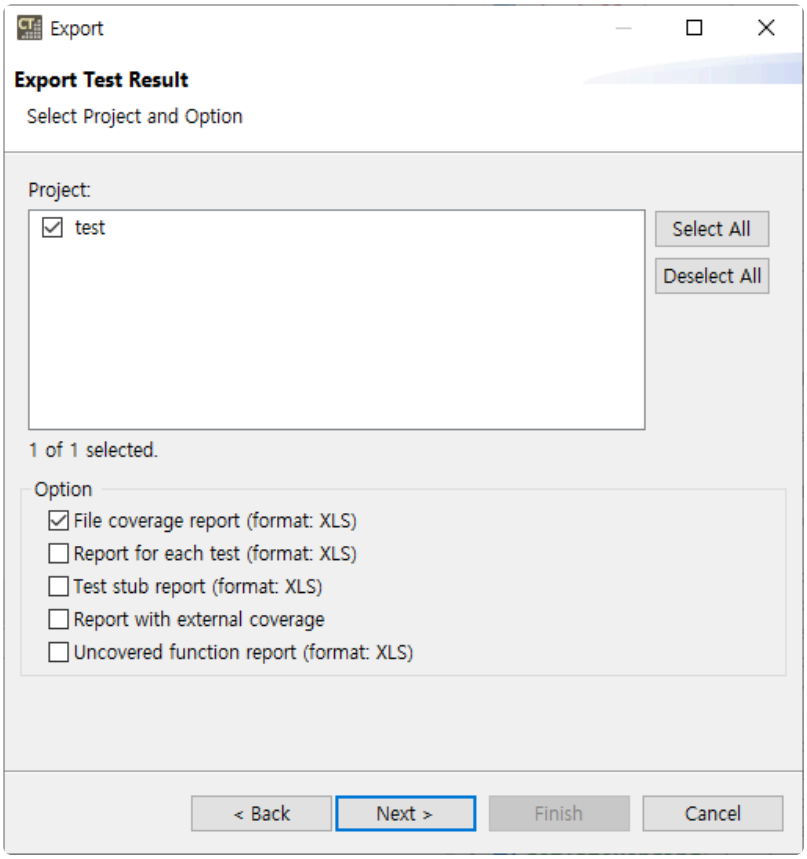
18.4.8. Test – Generate test report

You can create the test result report in various formats by using Export.

1. In the Export Wizard, click [Test] -> [Create Test Report] and click [Next].



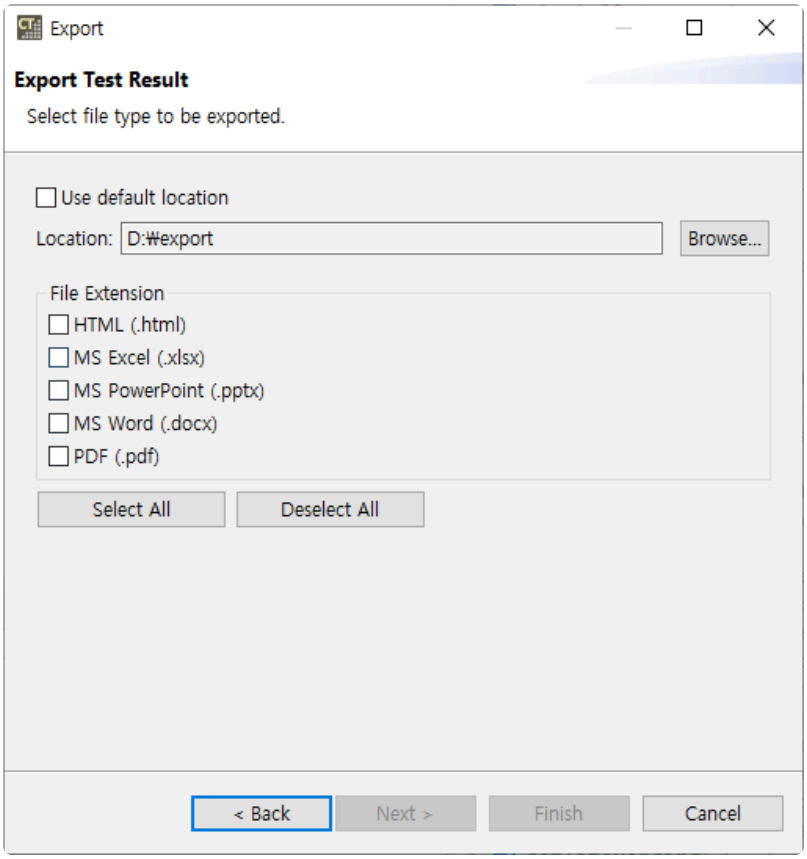
2. Select the project that you want to generate the report and the report type.



a. In [Project], select the project that you want to generate the report
b. In [Option], check each report to be exported.

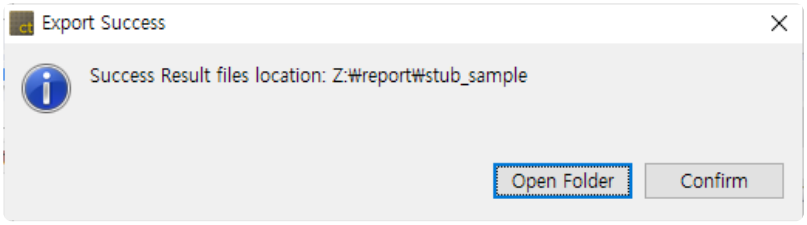
Report type	Description
File coverage report	Generate reports of covered functions for each files in the selected project.
Report for each test	Generate reports of test result and coverage for each test in the selected project.
Test stub report	Generate a report of stubs created in the selected project.
Report with external coverage	Generates a report containing the external coverage in the selected project.
Uncovered function report	Generate a report of the uncovered function in the selected project.

3. Select the location where the reports are exported and the report file extensions and click [Finish].



a. Selecting the checkbox [Use default location] specified “User Account\Export” directory by default.
b. [File Extension] provides the report file format with html, pdf, docx, pptx and xlsx. The report file extension can be selected in duplicate.

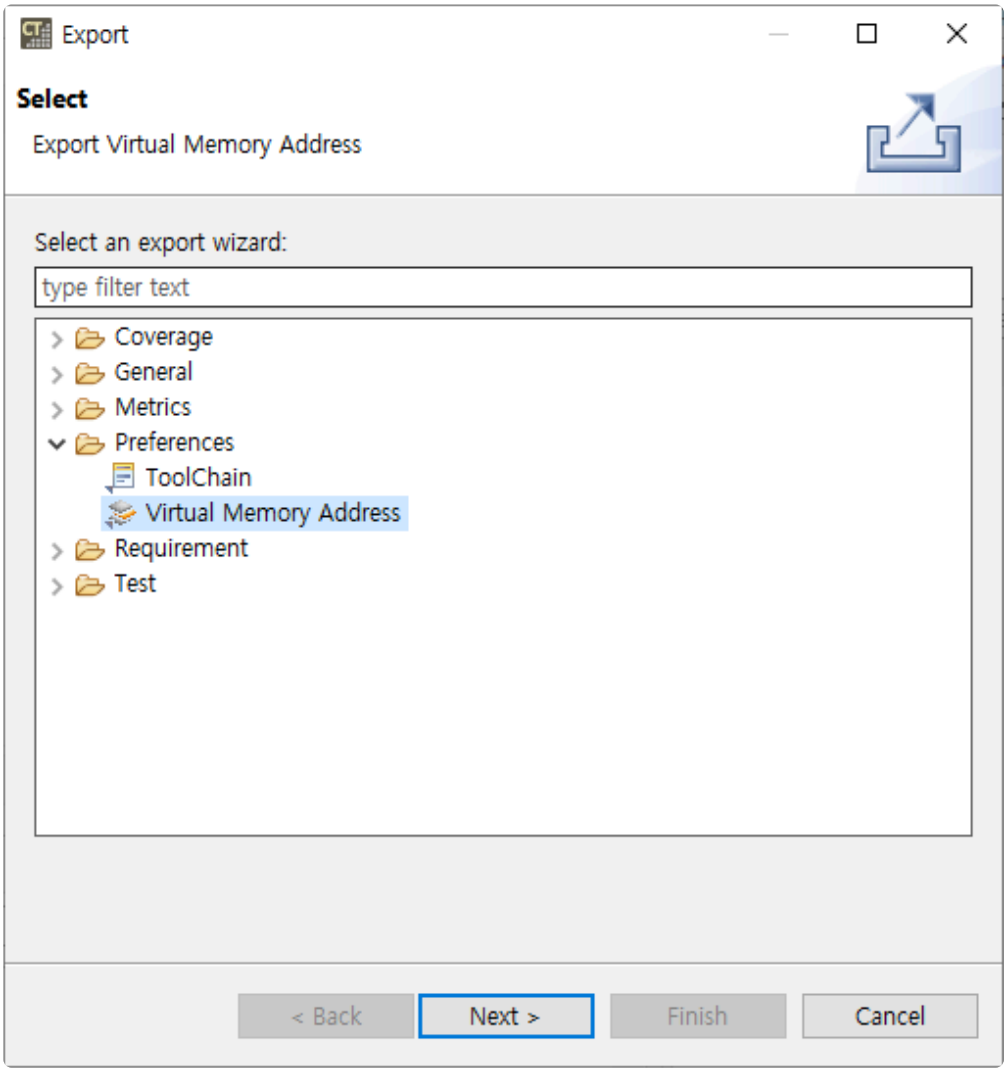
1. Click [Open Folder] button in the message “Report has been created” to check the report in the folder where it is saved.



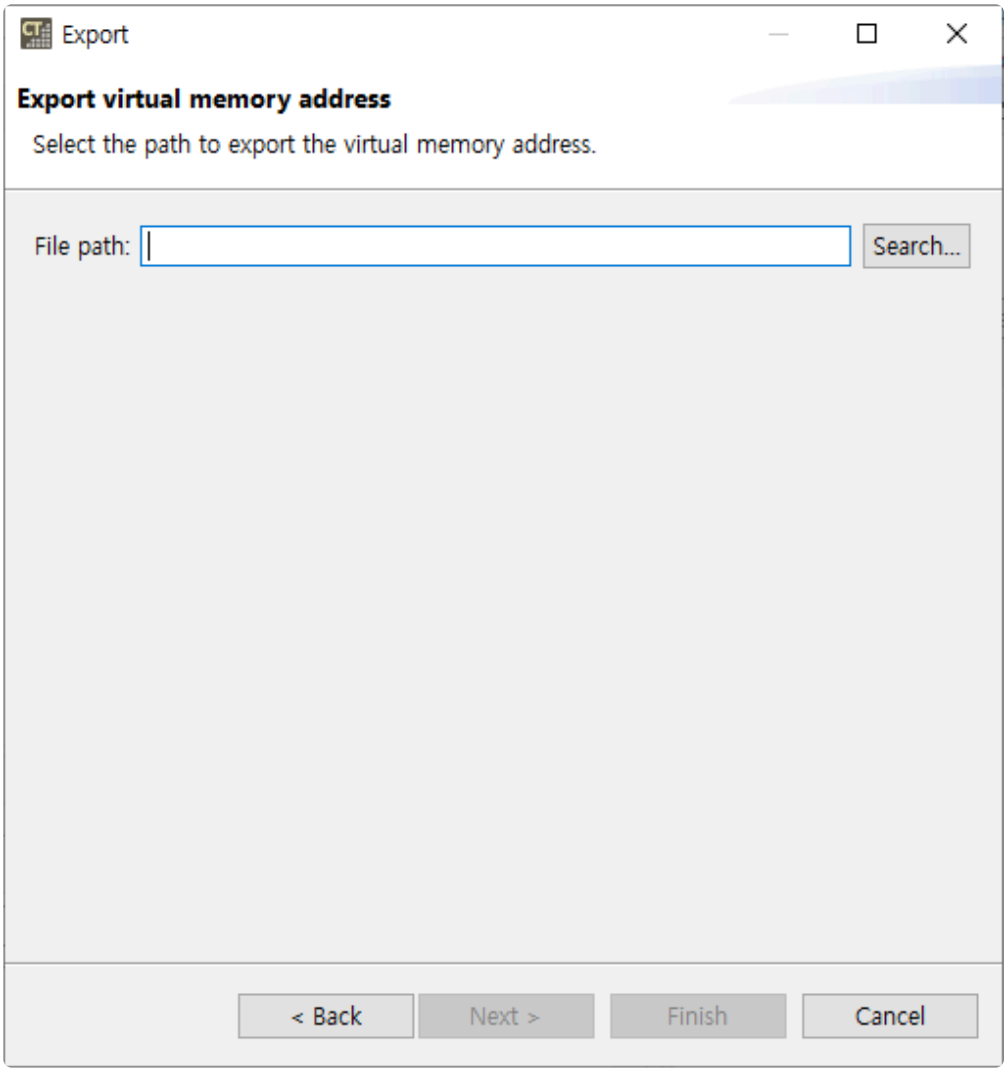
18.4.9. Preferences – Virtual Memory Address

You can export the information of the virtual memory address.

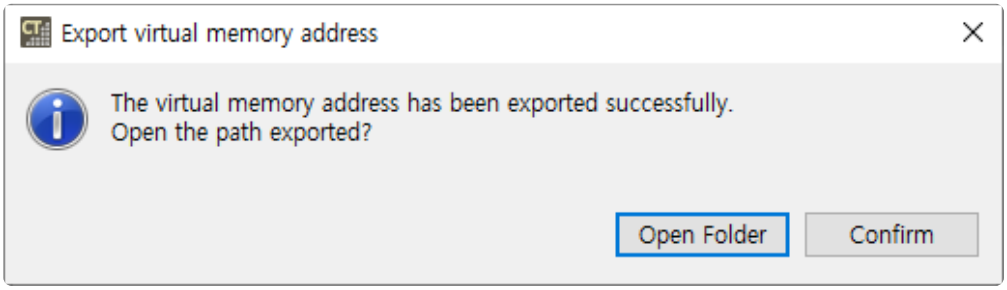
1. In the Export Wizard, click [Preferences] > [Virtual Memory Address] and click the [Next].



2. Select the path to export the virtual memory address and click the [Finish].



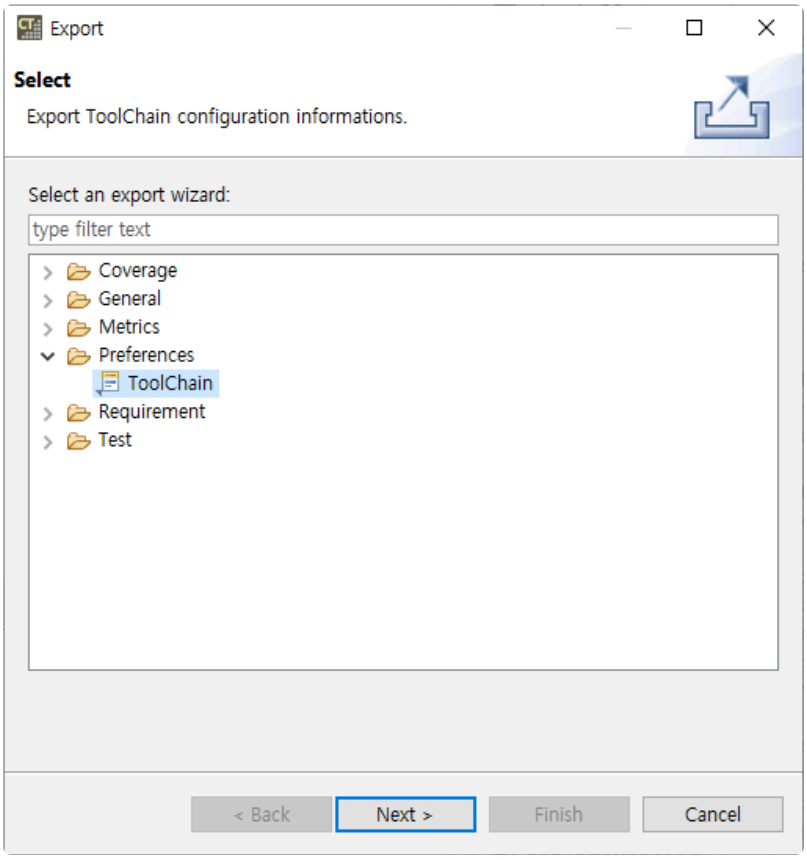
3. You can check the exported virtual memory address file by clicking the [Open Folder].



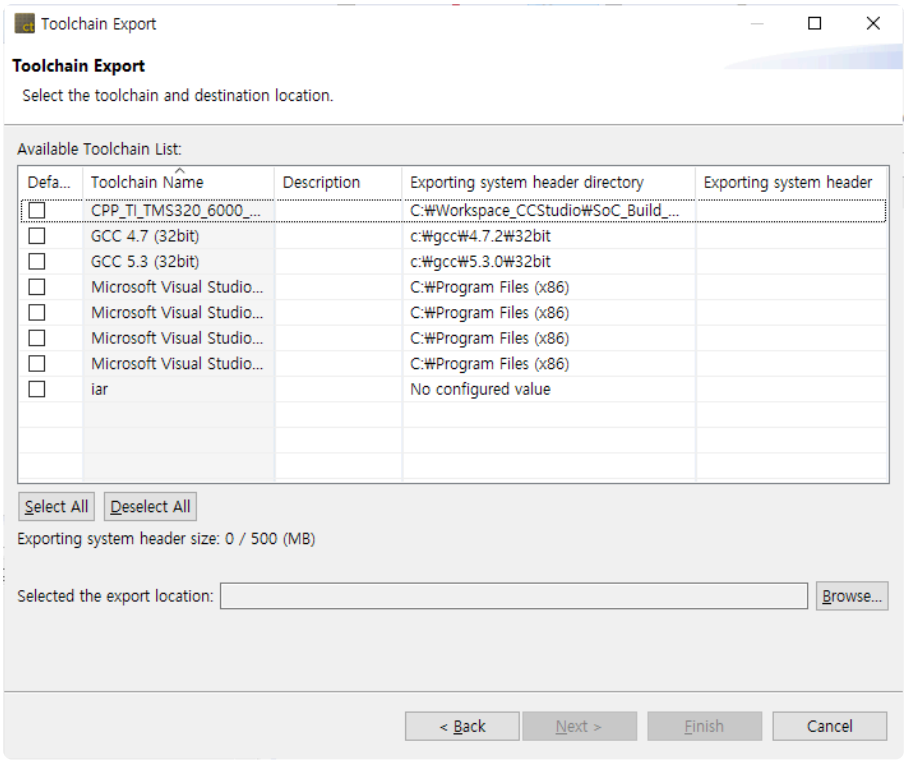
18.4.10. Preferences – ToolChain

You can export the information of the toolchain created.

1. In the Export Wizard, click [Preferences] > [ToolChain] and click [Next].

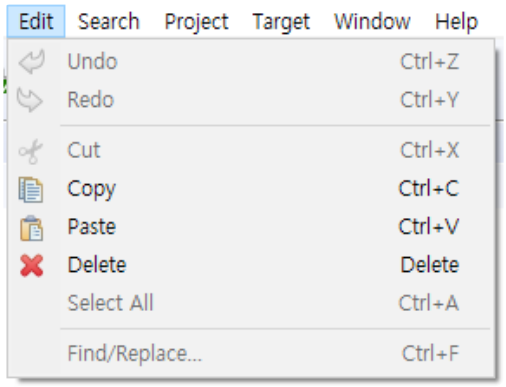


2. The list of toolchains registered currently is displayed. Check [Exporting system header] to export including system header. Enter the export path and click [Finish].



19. Edit Menu

In the Edit menu, you can carry out the functions such as Cut, Copy, Paste, Delete and Find/Replace the items selected in the view or editor, or undo or redo the last action performed.

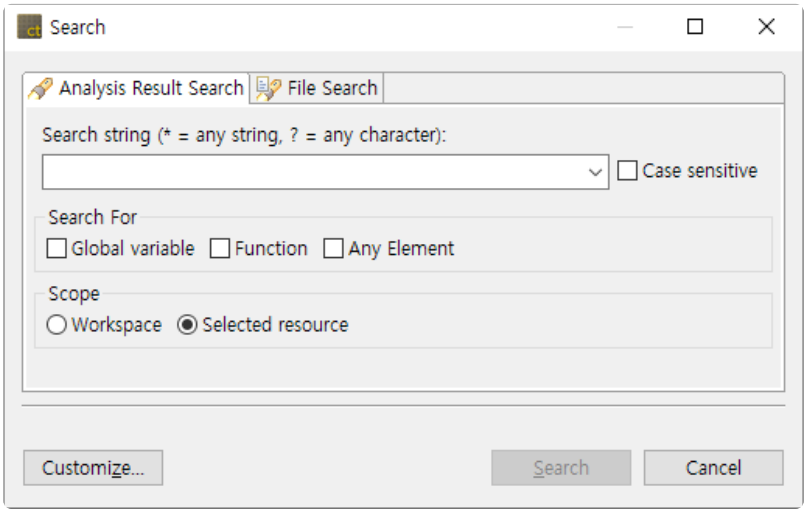


The menu may vary depending on the selected view or editor.

20. Search Menu

Search Analysis Result

You can search the result created after analysis.



Search word

Enter characters to be searched.
The available wildcards are displayed in the Search dialog box.

- “*” Wildcard string: matches the set of characters containing the empty string.
- “?” Wildcard character: matches all characters.

Search target

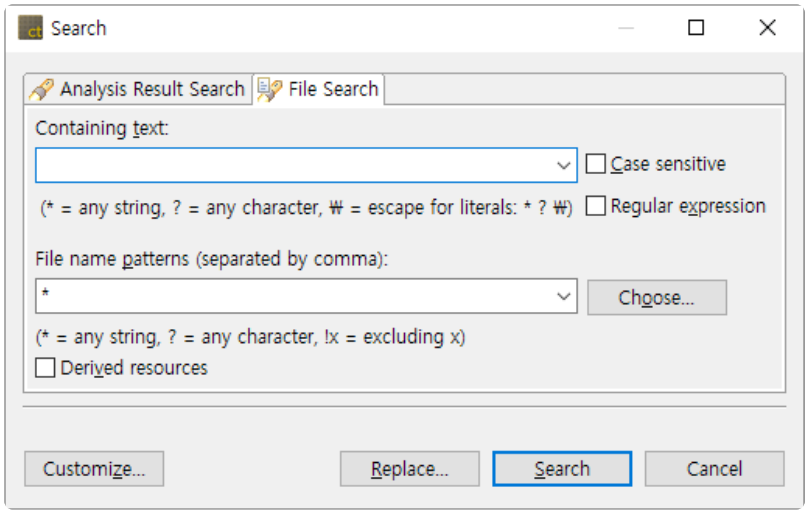
Select the target to be searched. The global variable, function or all targets can be searched.

Search range

Select the searching range. It can be searched within the workspace or the selected item range.

File search

A file can be searched.



Text included

Enter a character to be searched. To search a file, the field must be empty.
Click [▼] to select the characters searched recently.
The available wildcards are displayed in the Search dialog box.

- “*” Wildcard string: matches the set of characters containing the blank string.
- “?” Wildcard character: matches all characters.
- To search “*”, “?” or “\” character, enter a backslash before the characters in order to indicate that you do not use these characters by “\ *” wildcards. (Ex: “\?” or “\ \”)

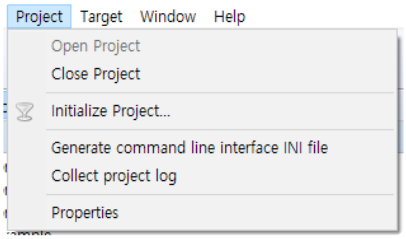
File Name Pattern

Enter the pattern of all file names for the files to be searched by using the specific expression.
The wildcards that can use the file name pattern are displayed in the Search dialog box.

- “*” Wildcard string: matches the set of characters containing the blank string.
- “?” Wildcard character: matches all characters.

21. Project Menu

In the Project menu, you can execute the operations for the project (open, close, initialize).



Open Project

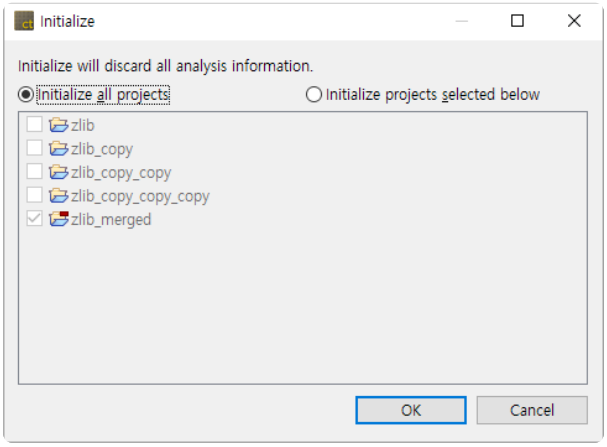
You can open a closed project selected in the Test Navigator view.

Close Project

You can close an open project selected in the Test Navigator view.

Initialize Project

You can initialize all the open projects in the workspace or the selected projects in the project list below. If you initialize the project, all analysis results are cleared.



Generate Command Line Interface INI File

Based on the project information selected in the Test Navigator, you can create the configuration file needed to execute the Controller Tester in CLI environment.

Collect Project Log

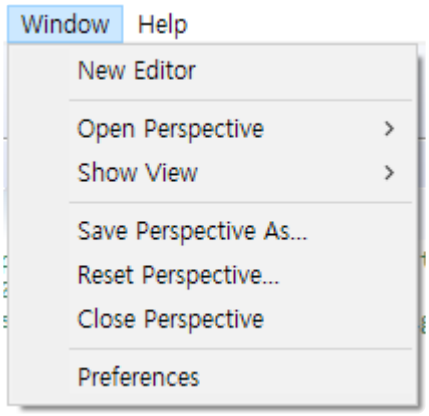
You can export the log file of the project selected in the Test Navigator view into the specified path.

Properties

You can see the information or change the settings for the project selected in the Test Navigator view.

22. Window Menu

In the Window menu, you can open the same new editor as an active editor, open a perspective or a new view, and see the preferences.



New Editor

It copies the selected source code editor or test editor.

Open Perspective

You can select and open the perspective registered in Controller Tester.

Show View

You can select and open the view registered in Controller Tester.

Save Perspective As

Saves the perspective reconfigured by the user as the other name.

Reset Perspective

Resets the perspective into the initial state.

Close Perspective

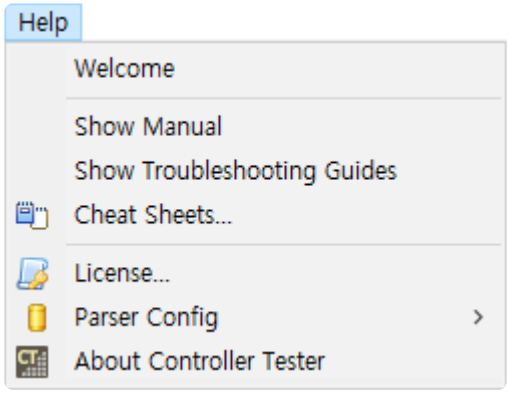
Closes the perspective currently open.

Preferences

You can check or change the settings currently applied to the tool.
For more information, please refer to Preferences

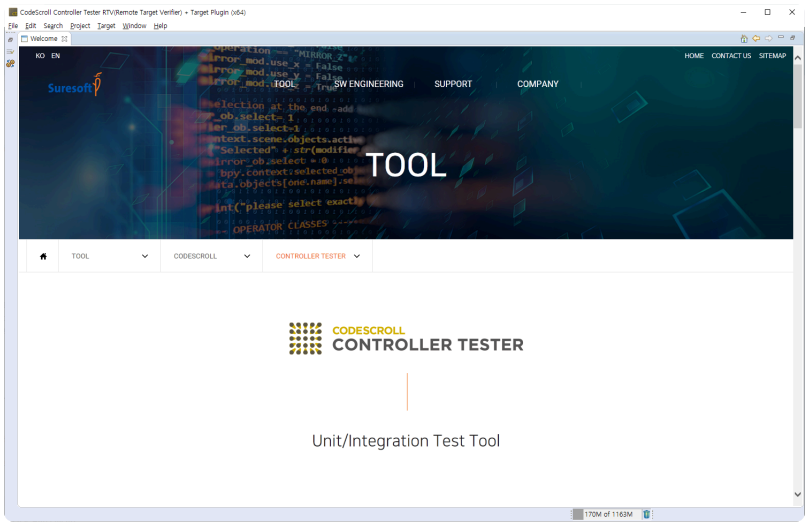
23. Help Menu

In the Help menu, you can find manuals, troubleshooting guides, tutorials, license settings, changing and reverting analysis settings, and information about the installed product.



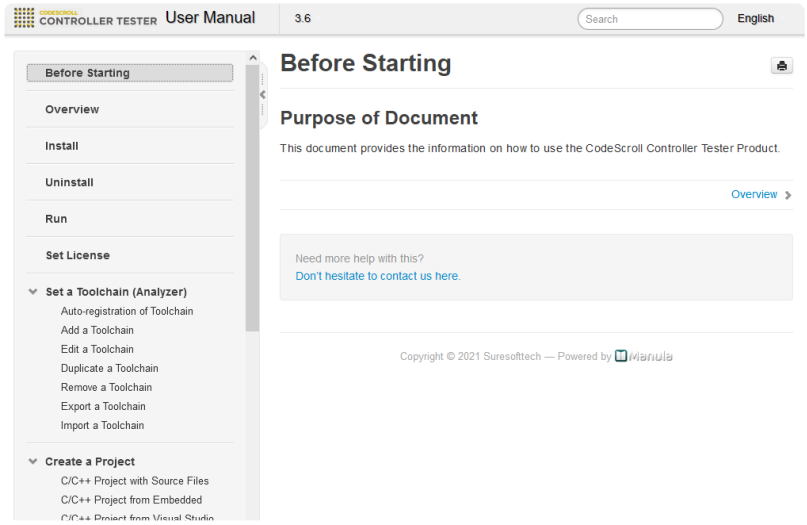
Welcome

You can see the welcome page that appears when the tool is executed for the first time.



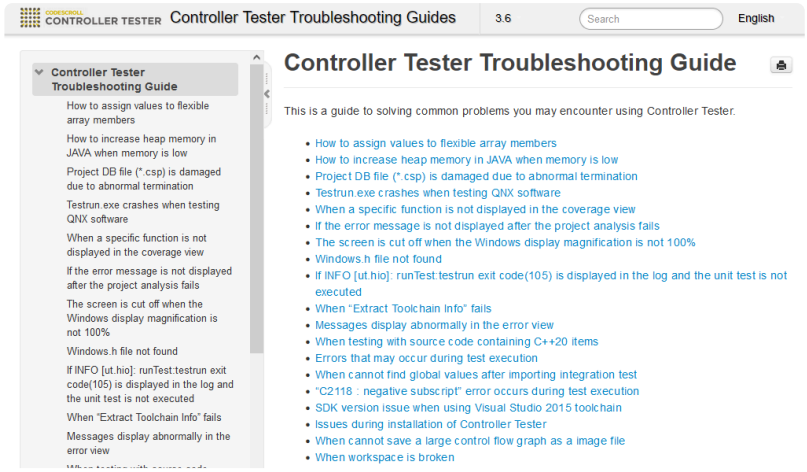
Show Manual

You can see the Controller Tester User Manual.



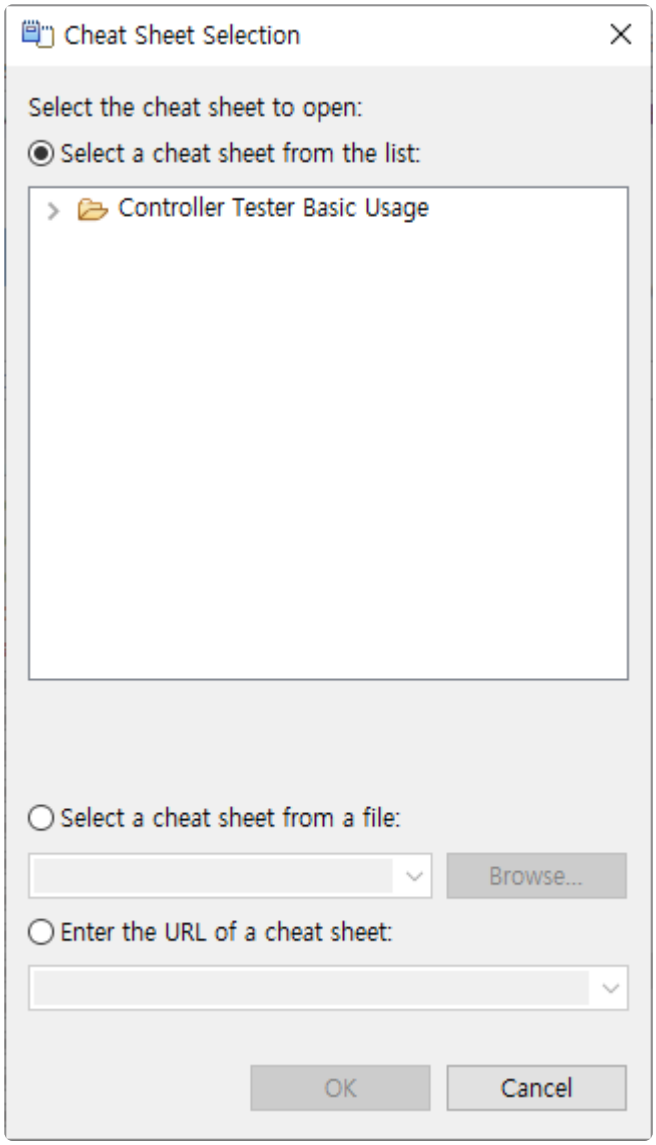
Show Troubleshooting Guides

You can see the Controller Tester Troubleshooting Guides.



Cheat Sheets

You can run cheat sheets on how to use Controller Tester.

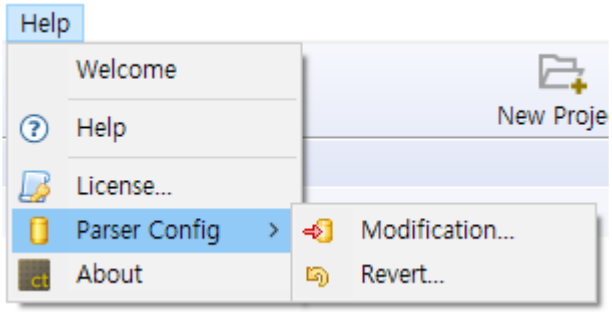


Modify Parser Config

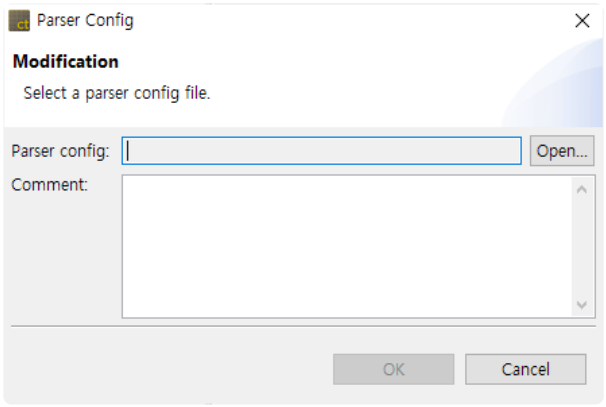
You can modify the parser config by entering the compressed file provided by SureSoft Technologies, Inc.

How to use

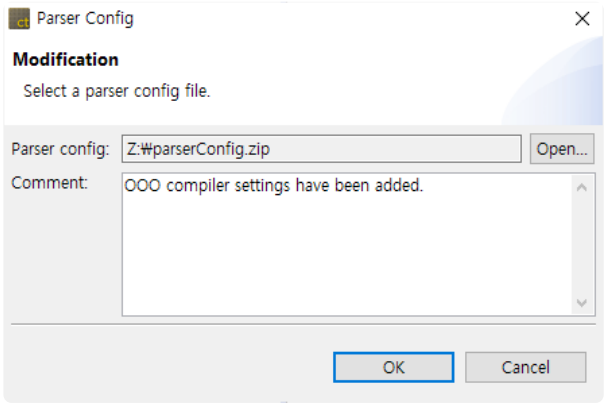
1. Select [Help] > [Parser Config] > [Modification…] menu.



2. Click [Open…] to select the compressed analysis setting file and enter the changes of analysis settings to [Comment] text as an option.



3. Click [OK].



Format of the parser config file

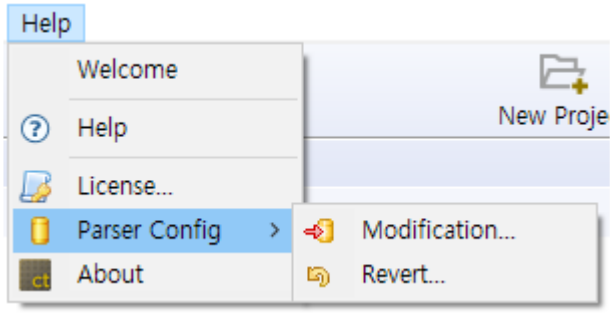
- The directory [parserConfig] must exist immediately when opening the compressed file as shown in the figure below.
- The extension is “zip”.

Revert the Parser Config

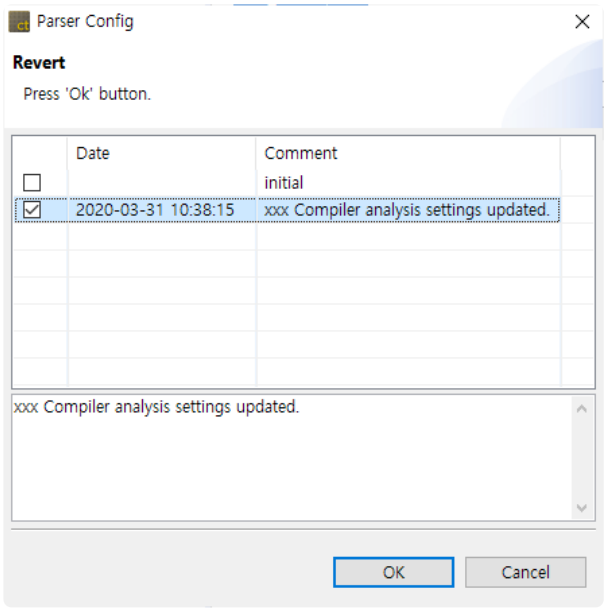
You can revert the parser config by selecting one among the previous parser configs.

How to use

1. Select [Help] > [Parser Config] > [Revert] menu.



2. Check the parser config to be reverted in the list of the existing parser configs and select [OK].



Information

You can see the information and version of the product installed.

Click [Installation Details] to see the features installed and plug-in.

24. Troubleshooting

If it conflicts with the security program at installation

Some security programs may stop running the tool installation package. If an error occurred, temporarily stop the security program operation and proceed with the installation again.

If the product does not operate after specifying the workspace when executing it

When executing the product for the first time, the global data of CodeScroll is stored in \APPDATA\ path. If access to this path is restricted, the product may not be operated.
To solve this problem, please use the feature to set the global data path.
Setting method: Enter the global data path to be set instead of –g default (default value) in CodeScroll.ini file. (the same if a case of -g or –global and csc.ini file)
If the -g option is not given, the dialog for setting any global data path when executing the product for the first time is displayed. If canceled here, it is set to the default path.

If it is installed and run in Windows Vista/7

It can only be run with the User Access Control (UAC) features disabled.
If you want to use it without disabling UAC, change the directory during installation so that it is not installed under the Program Files.
In addition, the Windows accounts for installation and execution must be the same and must be installed with an administrator account.

If an analysis is failed or Compile (pre-processing) error occurred

It is the state of source code that the test cannot be run. Please check the compile/link flag and header file/library settings.
You can check the settings in [Preferences] -> [Toolchain settings].
If all settings are correct (if it is possible to build without any problems in the actual development environment (IDE etc.)), please contact SureSoft Technologies, Inc.

If it fails to run all test cases

This is because the test engine operation is blocked by the firewall. The test engine loaded in memory when executing the test must be added as an exception to the firewall setting to run normally.

If it fails to register toolchain automatically

If it fails to register toolchain automatically due to some problem with toolchain information installed on

PC, the toolchain auto-registration feature can be turned off. (`--disableAutoToolchain option`)

- Add `--disableAutoToolchain` option to CodeScroll.ini file

Preprocessing error when running tests after connecting stub

When you create and connect a stub and run the tests, the stub may display the error `catastrophicerror:cannot open source file.`
This error occurs because the path of the stub file is too long. Windows defines the maximum length of the path as 260, so please rename the workspace and the project to reduce the path.

Etc.

For other issues, please refer to the Troubleshooting Guide .

Technical support

If you find any problems, please contact the Technical support contact below.

- help@suresofttech.com
- +82-2-6472-2800

25. CONTROLLER TESTER Target Plug-in

Introduction

Controller Tester Target Plug-in complements the functionality of the default Controller Tester. Controller Tester allows you to automate testing in your own host environment. In other words, you run the test in a software development environment (normal PC). Controller Tester Target Plug-in allows you to run tests of Controller Tester in a real embedded target environment. This makes it easy to check whether the test results are the same in the host and target environment and to run tests that cannot be executed in the host environment because of factors dependent on the target environment.

25.1. Target Environment Settings

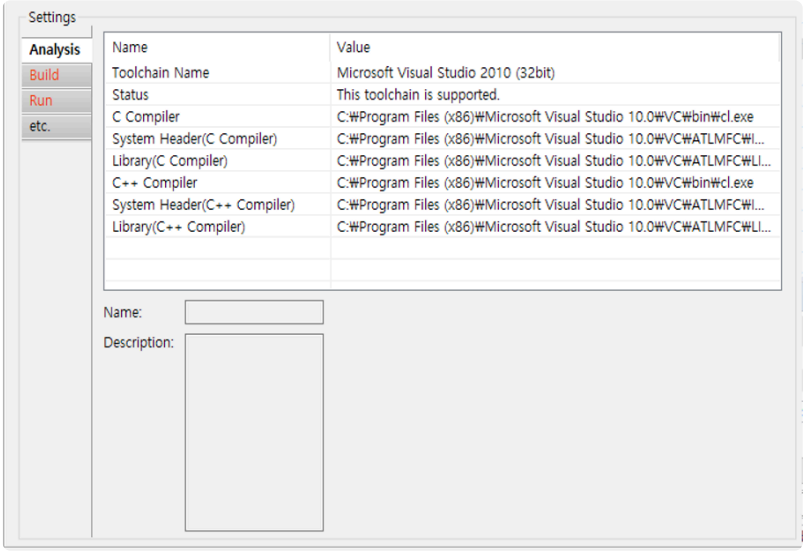
To run tests in a target environment using Controller Tester Target Plug-in, you need to enter information about the target environment.
Controller Tester Target Plug-in builds a test harness using information about the target environment entered by the user and automatically gets the results of running in the target environment.

The target environment can be set in the project properties page or in the new target test project wizard.

Target environment settings

Target environment settings are divided into basic information and detailed settings.

After entering the basic information, detailed settings that require input are displayed.



Target environment detailed settings

Target environment detailed settings are divided into the analysis, build, run, and etc.

Category	Description
Analysis	The toolchain information is displayed, and in the case of a target test project, the target compiler related settings required for analysis are displayed.
Build	The settings for building the test software are displayed.
Run	Settings for running tests and getting results in the target environment are displayed.
etc.	Other settings are displayed. (Program entry point, etc.)

Required settings for each category are indicated in red. Depending on whether the required settings for each category are entered or not, the behavior when clicking the [Run] button is different as shown

below.

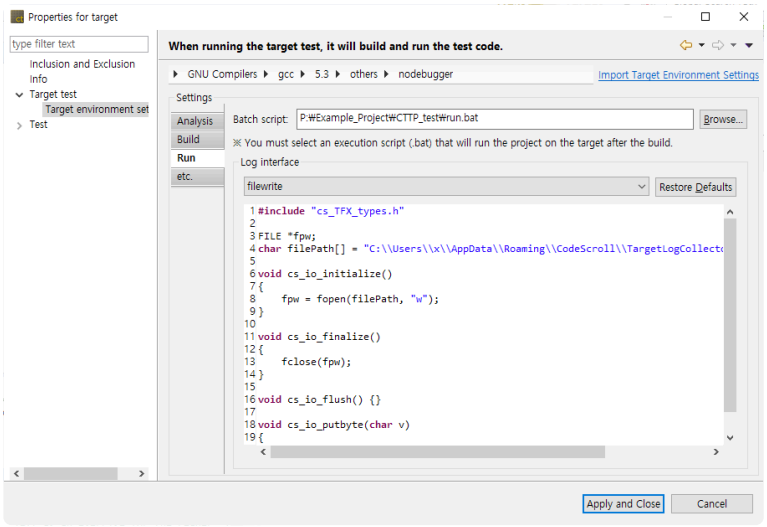
Required settings completed category	Description
None	Test run impossible
Analysis	If you click [Run], it overwrites the original source code with the test code. To perform the test, the user must manually build and run the test.
Analysis, Build	If you click [Run], it builds the test code. To perform the test, the user has to manually run the test on the target.
Analysis, Build, Run	Automatically run tests in the target environment.

Import target test results

Controller Tester Target Plug-in saves and imports the result of running the test in the target environment in log format.
If you do not use the debugger, you need the settings for saving the target test results(Log interface) and the settings for importing(Target Log Collector – Preferences).

Log interface

Log interface is a setting for saving the test results in the target environment. Log interface is written in the form of a source code that is actually run in the target environment.



Log interface structure

Function	Description
`viod cs_io_initialize()`	Initial function for transfer
`void cs_io_finalize()`	Transfer end function
`void cs_io_flush()`	Remaining data transfer function
`void cs_io_putbyte(codescroll_byte v)`	1-byte data transfer function

Import target log automatically

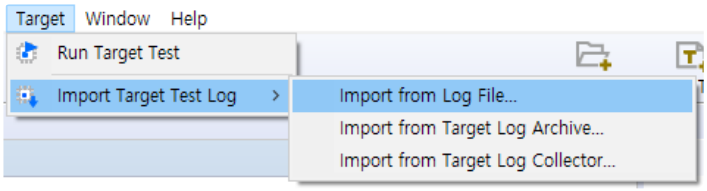
The target log, which is the result of the test execution, can be automatically imported through the target log collector. For the target log collector settings, please refer to [Target Log Collector] in Target Log Collector and Target Test Preferences.

Import target log manually

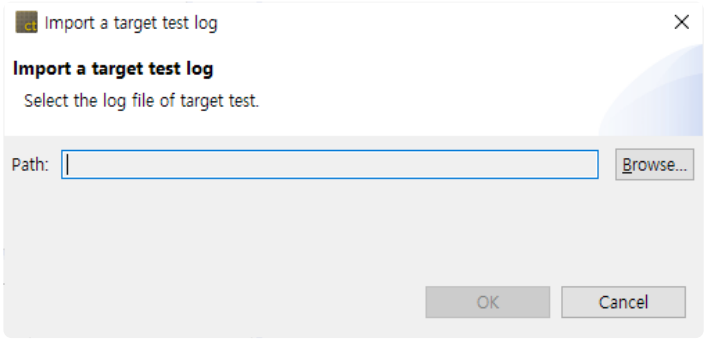
If the target log cannot be imported automatically, you can import the target log manually.

Import from a log file

1. After selecting an analyzed project, click [Target] -> [import Target Test Log] -> [Import from Log File…] in the global menu.

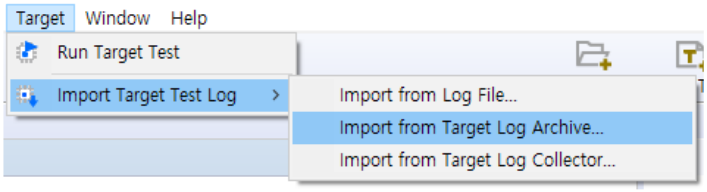


2. Select a target test log file and click the [OK] button.

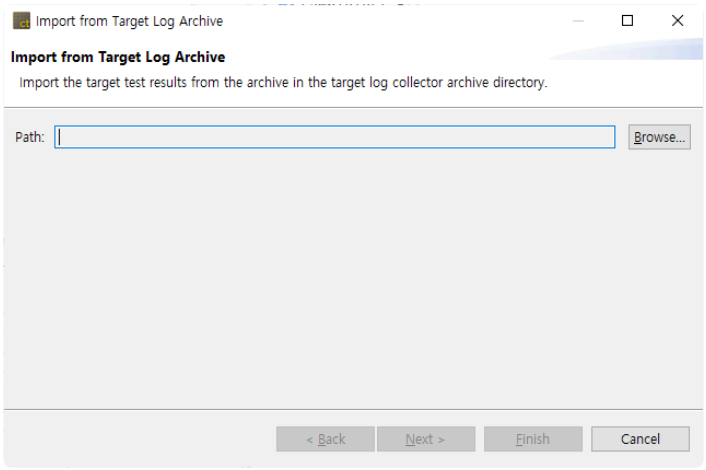


Import from a target log archive

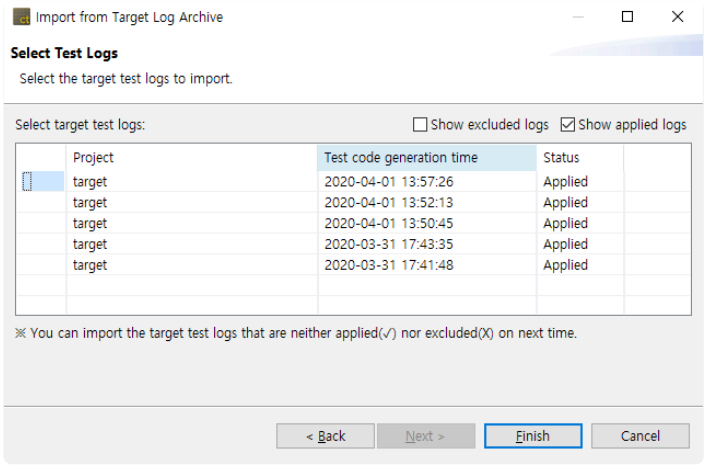
1. After selecting an analyzed project, click [Target] -> [import Target Test Log] -> [Import from Target Log Archive…] in the global menu.



2. Select a target log archive file and click the [Next] button.

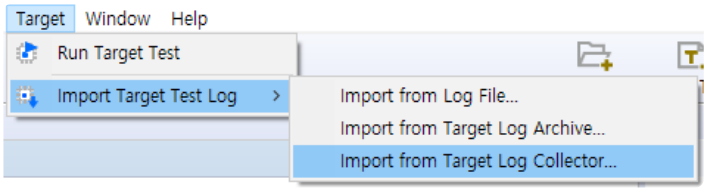


3. Check a target test log file and click the [Finish] button.

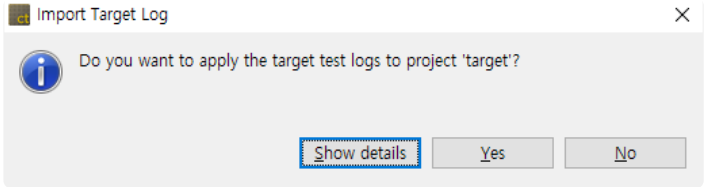


Import from target log collector

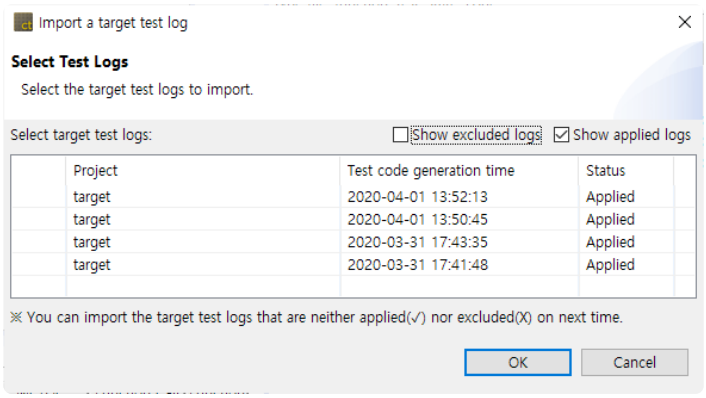
1. To get logs from the target log collector, you need to select ‘Use the default target log collector’ from [Preferences] -> [Target Test] -> [Target Log Collector].
2. After selecting an analyzed project, click [Target] -> [import Target Test Log] -> [Import from Target Log Collector…] in the global menu.



3. Click the [Yes] button to import all the new target test logs or click the [Show details] button.

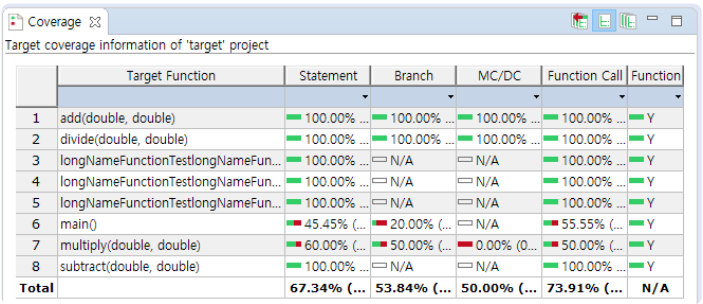


4. If you click the [Show details] button, check the target test log to be imported and click the [OK] button.



Result

After the import is completed, you can check the target coverage information in the coverage view.



	Target Function	Statement	Branch	MC/DC	Function Call	Function
1	add(double, double)	100.00% ...	100.00% ...	100.00% ...	100.00% ...	Y
2	divide(double, double)	100.00% ...	100.00% ...	100.00% ...	100.00% ...	Y
3	longNameFunctionTestlongNameFun...	100.00% ...	N/A	N/A	100.00% ...	Y
4	longNameFunctionTestlongNameFun...	100.00% ...	N/A	N/A	100.00% ...	Y
5	longNameFunctionTestlongNameFun...	100.00% ...	N/A	N/A	100.00% ...	Y
6	main()	45.45% (...	20.00% (...	N/A	55.55% (...	Y
7	multiply(double, double)	60.00% (...	50.00% (...	0.00% (0...	50.00% (...	Y
8	subtract(double, double)	100.00% ...	N/A	N/A	100.00% ...	Y
Total		**67.34% (...**	**53.84% (...**	**50.00% (...**	**73.91% (...**	**N/A**

[Import Target Test Log] may fail because the target test run information and the host test information are not valid if the host test case, source file, function information, etc. are changed before importing the log of the target test result.

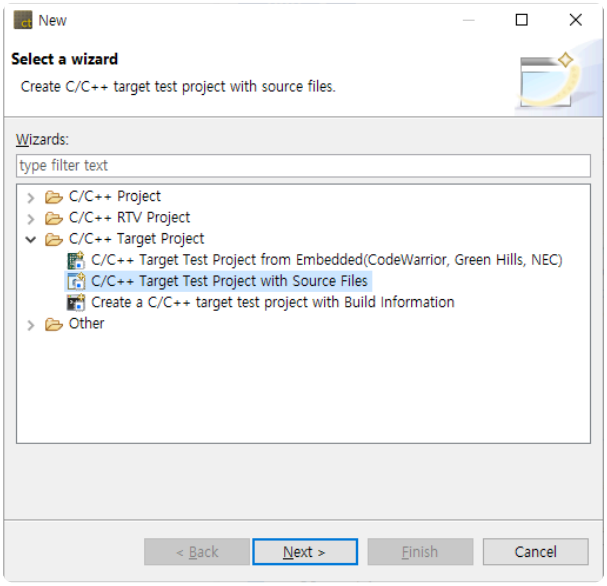
25.2. C/C++ Target Test Project

The C / C ++ target test project (hereinafter referred to as the target test project) is a project for target environment testing. Target testing projects do not support host testing.

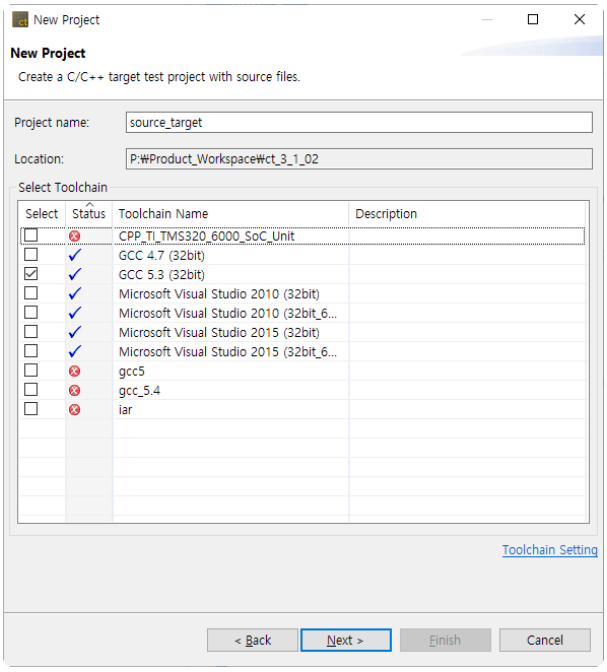
Create a target test project

C/C++ Target Test Project with Source Files

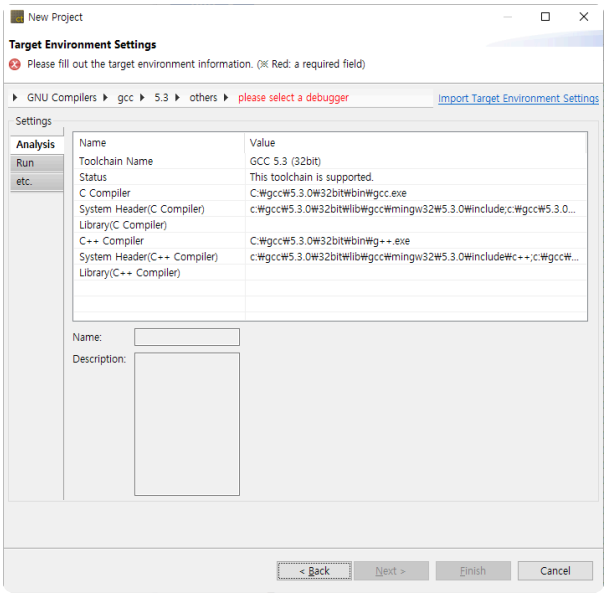
1. Create a project by selecting [File]-> [New]-> [Other …] in the global menu and clicking [C / C ++ Target Test Project with Source Files].



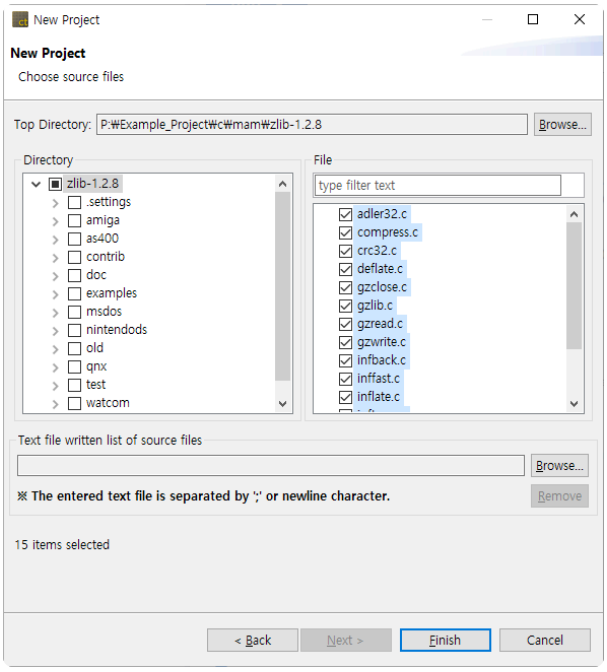
2. Enter a project name and select a toolchain to use.



3. After selecting the basic information of the target environment, enter the detailed settings for each category and click the [Next] button.

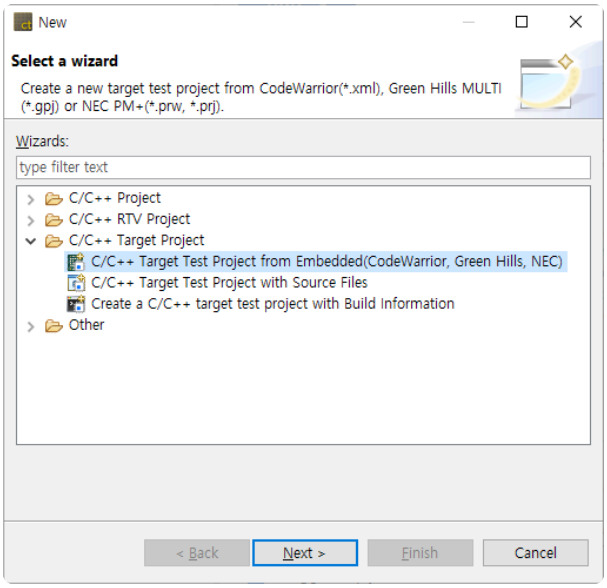


4. Select the source files under test and click the [Finish] button.

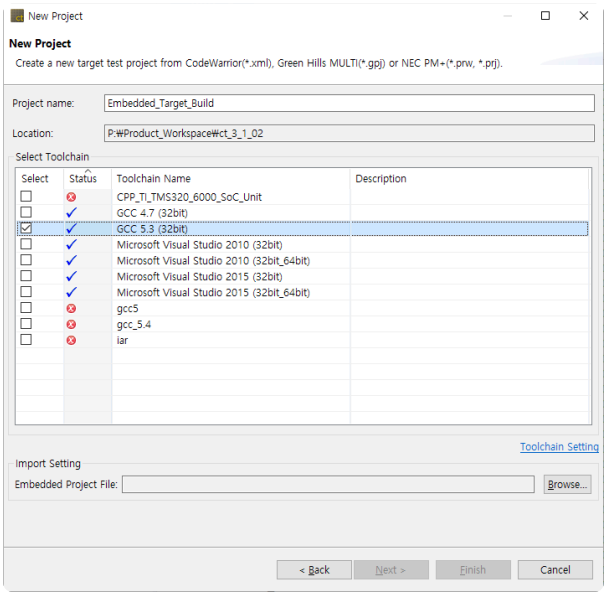


C/C++ Target Test Project from Embedded(CodeWarrior, Green Hills, NEC)

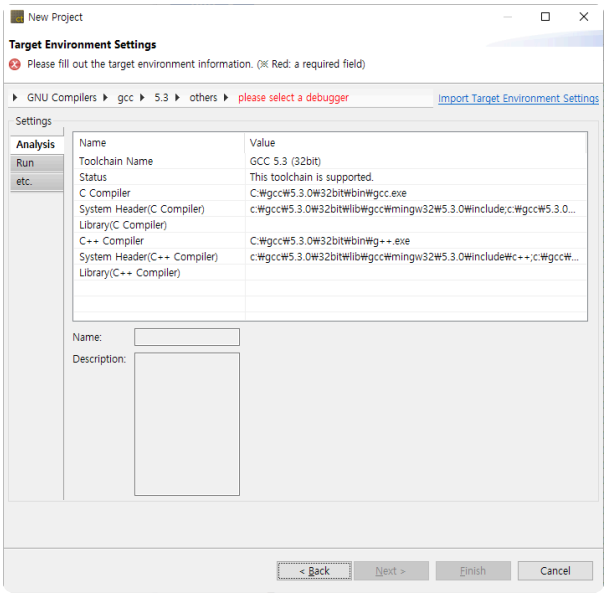
1. Create a project by selecting [File]-> [New]-> [Other …] in the global menu and clicking [C/C++ Target Test Project from Embedded(CodeWarrior, Green Hills, NEC)].



2. Enter a project name and select a toolchain to use. After selecting the toolchain, import the embedded project file to test. Currently, embedded projects supported by Controller Tester are CodeWarrior, Green Hills, and NEC.

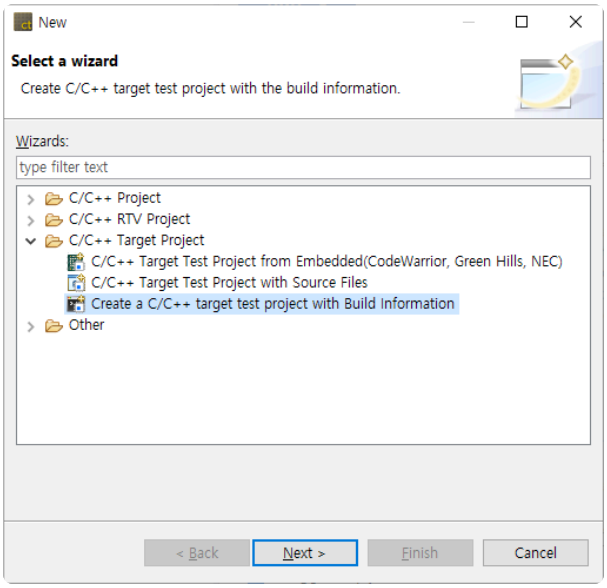


3. After selecting the basic information of the target environment, enter the detailed settings for each category.

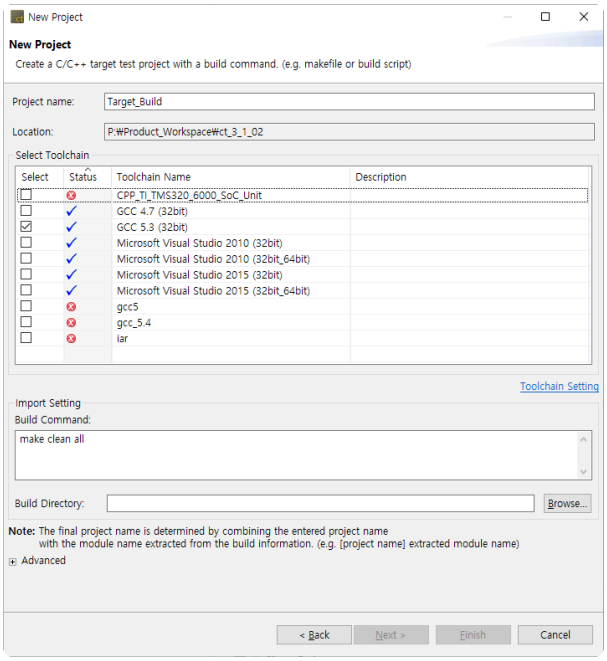


Create a C/C++ Target Test Project with Build Information

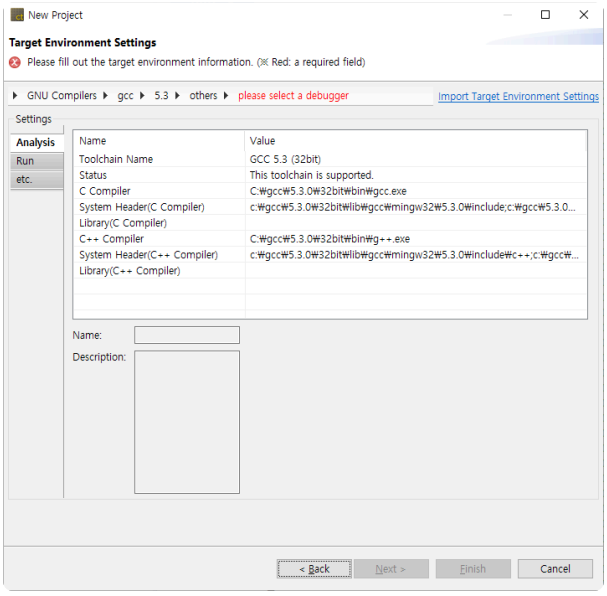
1. Create a project by selecting [File]-> [New]-> [Other …] in the global menu and clicking [Create a C/C++ Target Test Project with Build Information].



2. Enter a project name and select a toolchain to use. After selecting the toolchain, Specify the path of the created Makefile and build script as the build directory and write the build command. You can choose a script file to set the build-environment from the [Advanced] menu.



3. After selecting the basic information of the target environment, enter the detailed settings for each category.



> ! Toolchains with invalid compiler paths cannot be selected when creating a target test project.

25.3. Target Test Run Settings

You can check the target test execution settings on the target test page in [Project]-> [Properties].

Coverage kind

You can select the type of coverage(Statement, Branch, MC/DC) to be measured for the target test.

Target for coverage measurement

In addition to the functions under test, you can select functions to measure coverage.

All functions called by the target function	Measures coverages for all functions called by the function under test.
The functions called by the previous test run	Includes functions whose coverage was measured from the previous test.

Testcase run

You can choose how to run the test case.

Run at once	Loads and runs all test cases on target at once. Each test case is affected by the previous test case run results.
Run one by one	Repeats loading and running each test case on target. It takes longer than running at once but uses less memory.

25.4. Target Log Collector

The target log collector is a server that collects target test results and sends them to Controller Tester. Collecting target test results methods are largely divided into communication and file scanning. Supported communication protocols and scannable file formats are as follows.

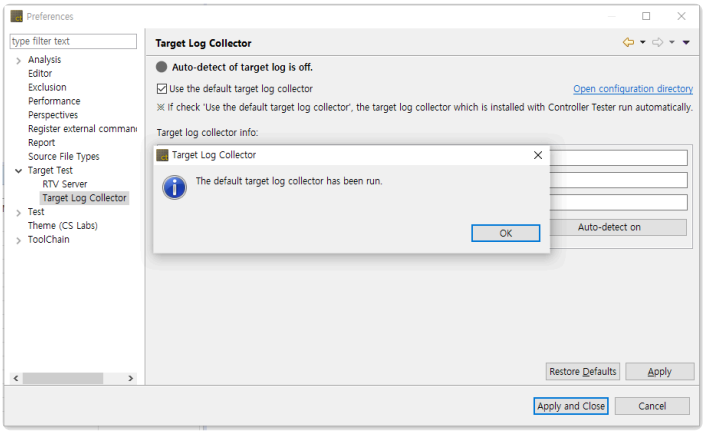
Protocol	TCP, UDP, UART
File format	Target log(text), memory dump(Hex, Intel HEX)

Installation

The target log collector is automatically installed when Controller Tester is installed.

Run

When [Use the default target log collector] is checked in [Preferences] -> [Target Test] -> [Target Log Collector], the target log collector starts collecting target test results.
If you use TargetLogCollector.zip after extracting it, run TargetLogCollector.exe in the unzipped directory at the Windows command prompt.



Settings

It can be set through the ‘settings.ini’ file in the target log collector installation path, and the options are as follows.

[LogReceiveServer]	Settings for receiving target test results
port	TCP, UDP communication port
protocol	Communication protocol for receiving target test results
timeout	Connection timeout
lastString	String representing the end of the data
serialPort	Serial communication port

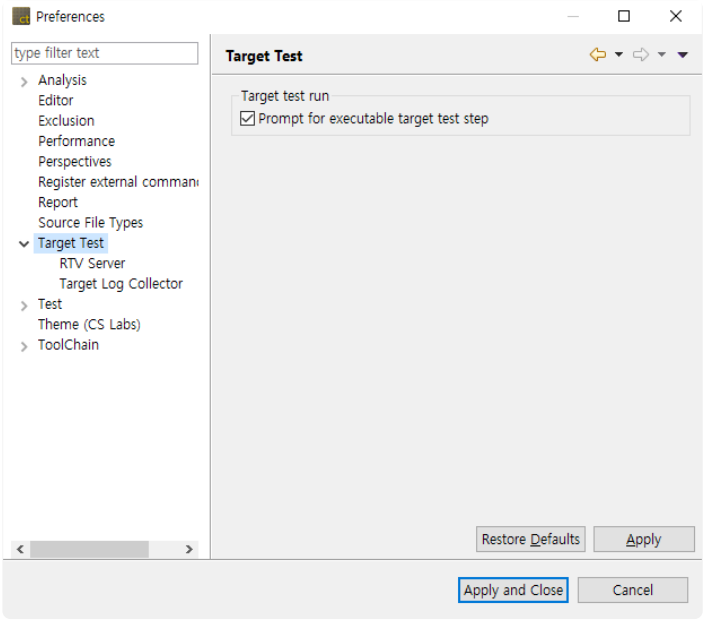
	(Windows: COM#, Linux: /dev/ttyS#)
baudRate	Serial communication settings
dataBits	Serial communication settings
stopBits	Serial communication settings
parity	Serial communication settings
flowControl	Serial communication settings
[ScanLog]	Settings for file scanning
dir	Directory to scan
fileExtension	File extensions to scan(Scan all files if blank)
[LogSendServer]	Settings for communication with Controller Tester
port	Target log file transfer port

timeout: If there is no data transmission for the specified number of seconds from the last reception, the received data is saved.
dir: When a target log file (text, hex) is entered in the specified directory, the target log collector recognizes it.

25.5. Preferences

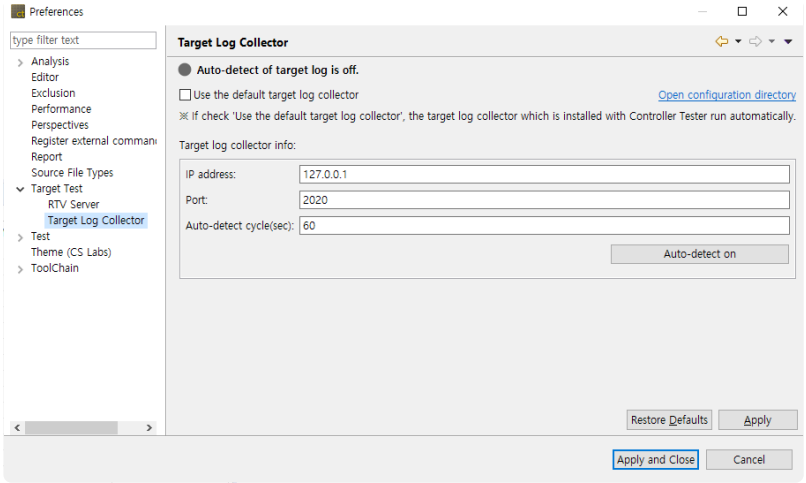
Target Test

You can set whether to open the target test step notification dialog when the target test is run.



Target Log Collector

Set target log collector information to use and whether to automatically detect target test results. If you turn on automatic target log detection by clicking the [Atuto-detect on] button, a new target test result is received from the target log collector at a specified automatic detection cycle.



Target Log Collector Setting	Description
Auto-detect cycle(sec)	Cycle to check for new target test results
IP	IP on the server or PC where the target log collector runs
Port	Target log collector’s communication port

! When using the default target log collector, set only the auto-detection interval(sec).

26. CONTROLLER TESTER RTV(Remote Target Verifier)

Introduction

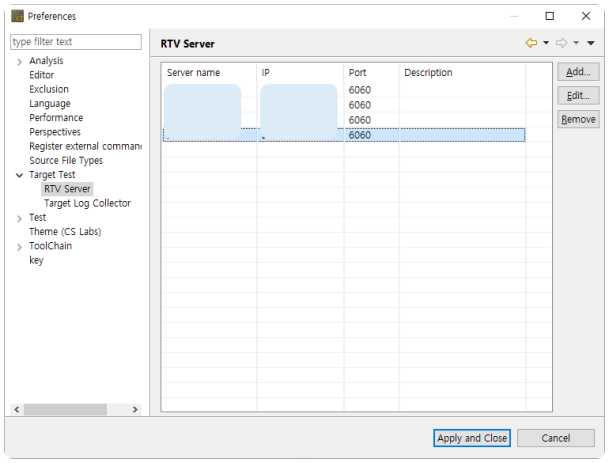
Controller Tester RTV helps you perform automated testing in your Linux environment.
After installing the Controller Tester RTV server in your Linux environment and completing the default configuration, you can use the Controller Tester RTV client to perform the RTV test as easily as you would in a hosted environment.
For information on how to install the Controller Tester RTV server in your Linux environment, please contact the Technical support contact on bottom of the Troubleshooting page of this document.

Supported Operating Systems

Server: Linux (RHEL, Ubuntu, Debian, Fedora based)
Client: Windows 7/10

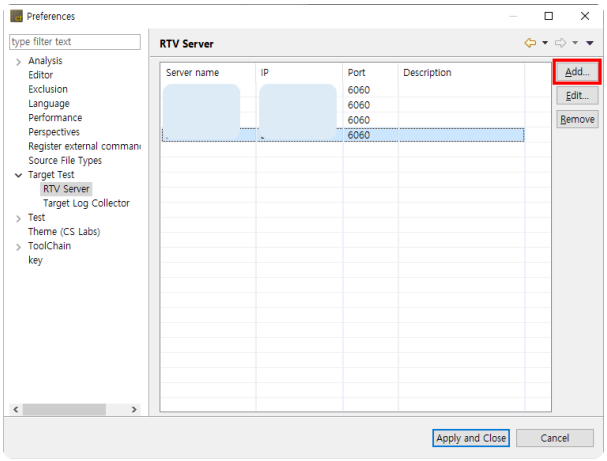
26.1. RTV Server Settings

In order to use Controller Tester RTV, you need to add a RTV server with Controller Tester RTV server installed.
The RTV server can be added in [Window] -> [Preferences] -> [RTV Server].

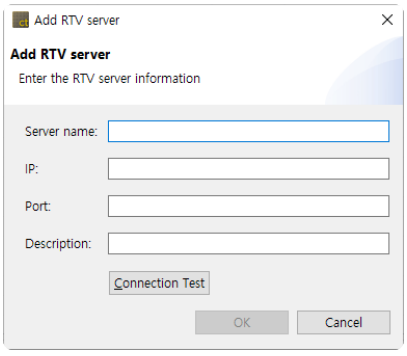


Add a RTV server

1. Click [Add] in [Window] -> [Preferences] -> [RTV Server].

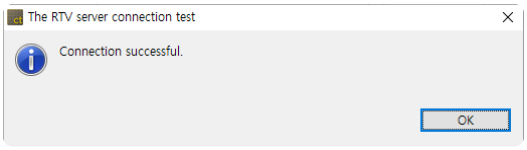


2. In the [Add RTV Server] window, enter the RTV server information.

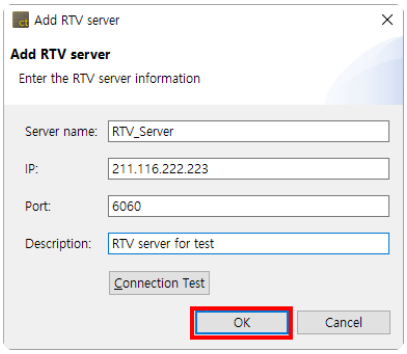


> ! The server name only accepts characters that can be used as Windows file names and can not be modified after adding the RTV server.

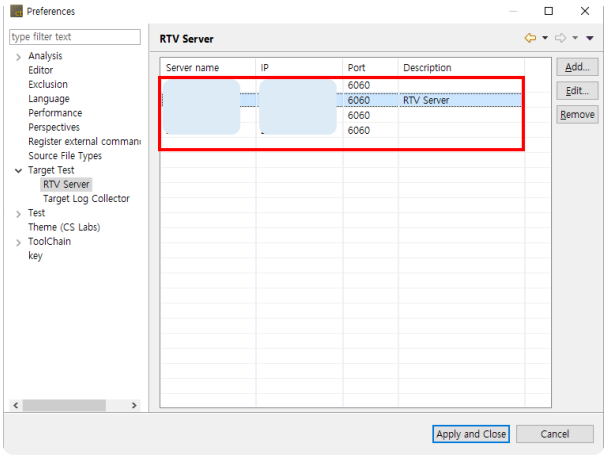
3. Click the [Connection Test] button to check the connection status with the RTV server you entered.



4. If the connection test with the RTV server is successful, click the [OK] button to add the RTV server.

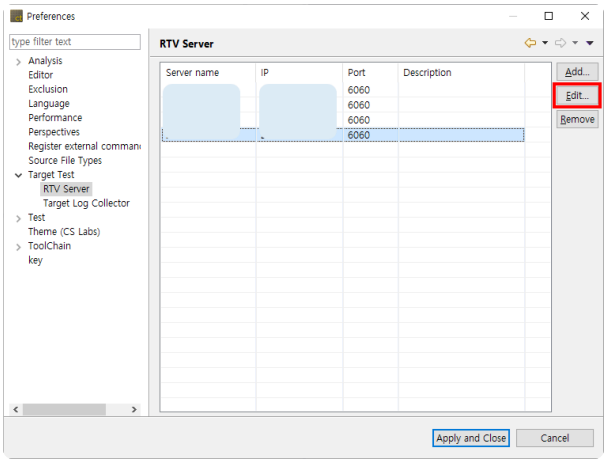


5. You can see that the RTV server has been added.

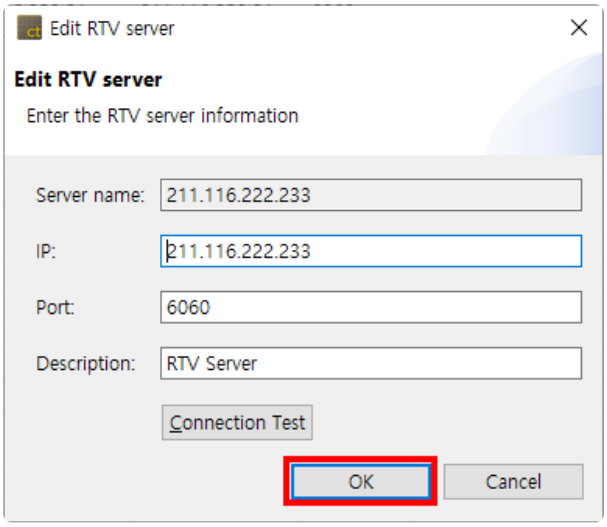


Edit a RTV server

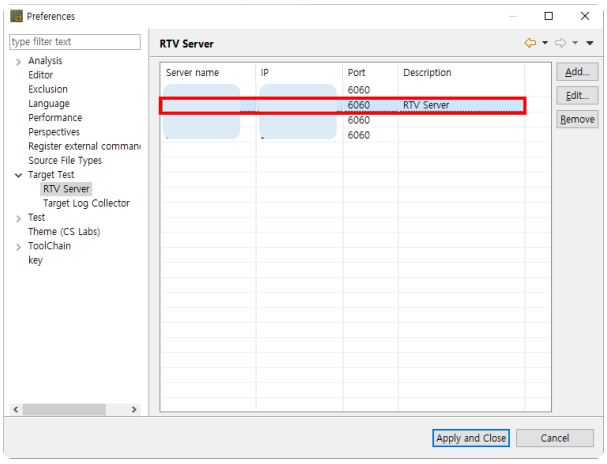
1. Double-click the server in the RTV server list, or select the server and click the [Edit] button.



2. On the [Edit RTV server] window, edit the server information and then click [OK].



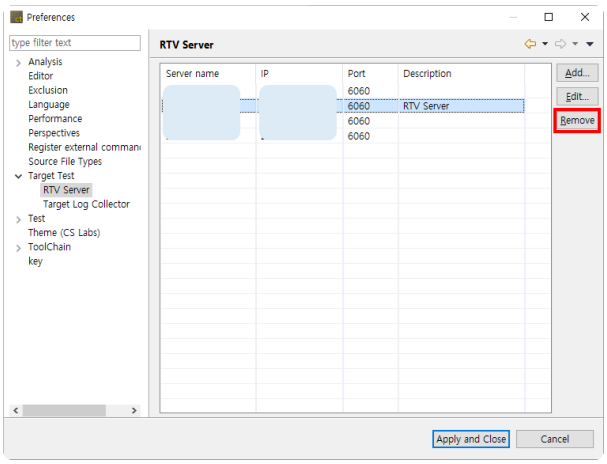
3. You can see that the RTV server information has been edited.



Remove a RTV server

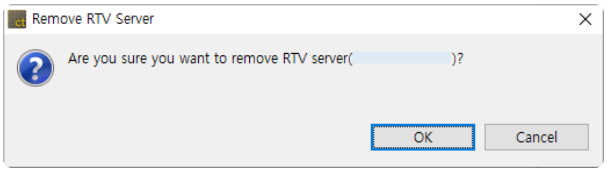
If you no longer use the RTV server, you can remove it.
When you remove a RTV server, the toolchain or project resources associated with the RTV server are deleted from the client environment(Can not be deleted if a RTV toolchain added from that RTV server exists).

1. Select a server from the list of RTV servers and click the [Remove] button.

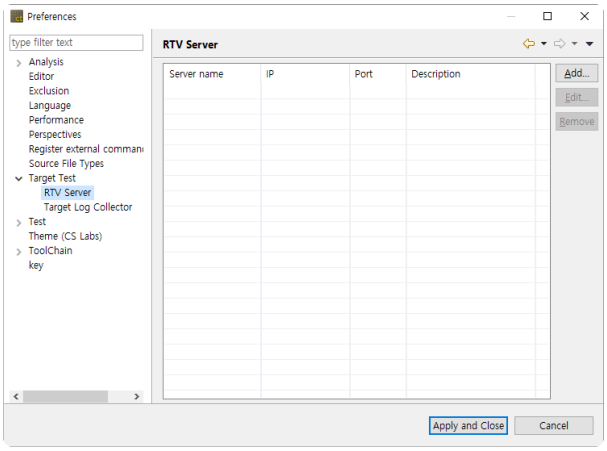


2. Click the [OK] button in the [Remove RTV Server] window(Depending on the size of the RTV

server resource, it may take a minute or two).



3. You can see that the RTV server has been removed.



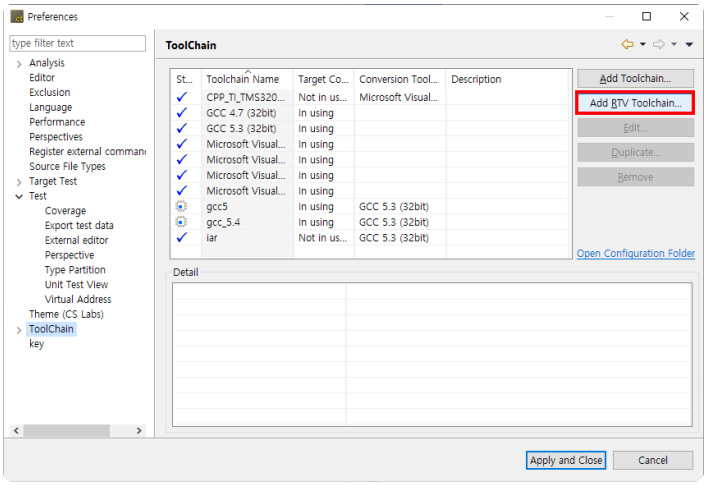
26.2. RTV Toolchain Settings

RTV toolchain

To build and run a test in a RTV environment with Controller Tester RTV, you need to set the toolchain (compiler) information for the source file being tested.
This chapter describes how to add, edit, and delete the RTV toolchain information that has previously extracted from the RTV environment where the Controller Tester RTV Server is installed.
(For extracting toolchain information from your RTV environment, please contact the Technical support contact at the bottom of this document Troubleshooting page.)

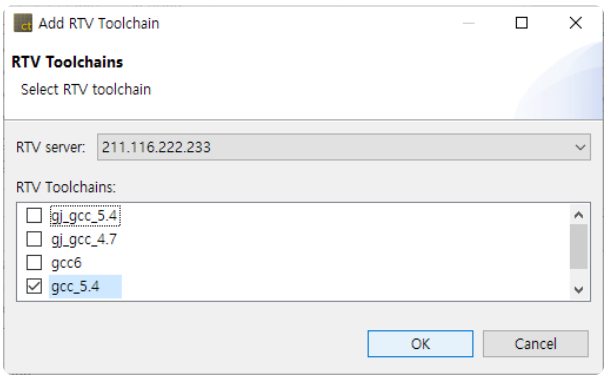
Add an RTV toolchain

1. Click the [Add RTV Toolchain] button in the [Window] -> [Preferences] -> [Toolchain] window.

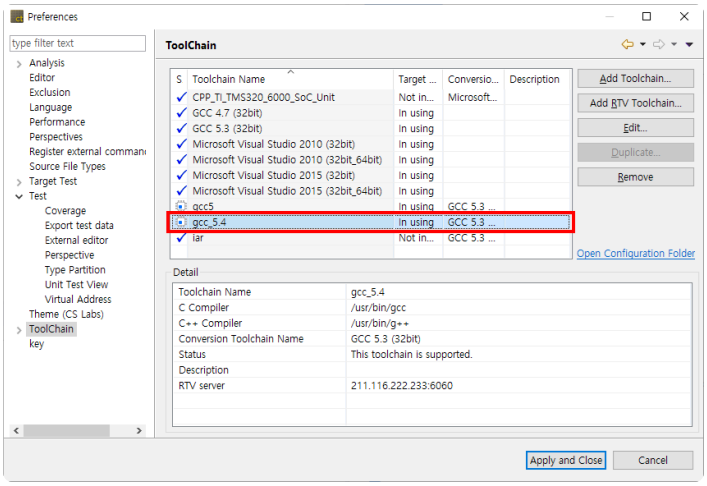


> If you do not have a RTV server, the [Add RTV Server] window will appear first.

2. If you select the RTV server in the [Add RTV Toolchain] window, you can see the list of RTV toolchain extracted from the RTV server. Select the toolchain you want to add and click the [OK] button.



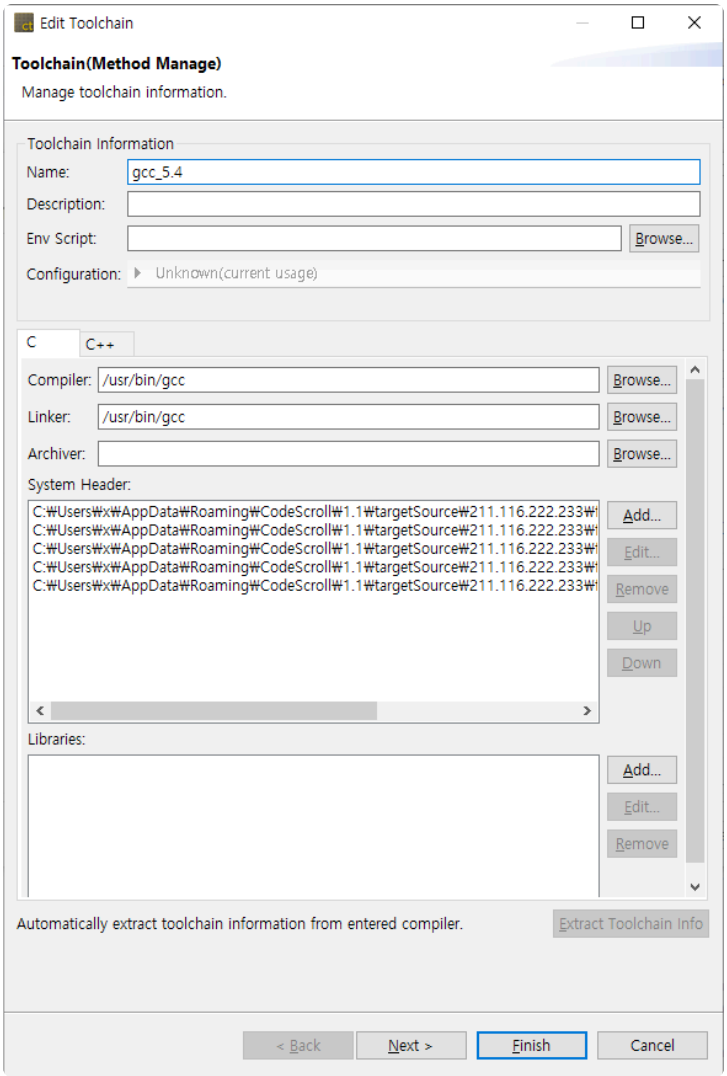
3. The toolchain information is imported from the RTV server (Depending on the size of the system header, it may take 1 to 2 minutes). When the operation is completed, you can see that the RTV toolchain has been added.



4. If you select the RTV toolchain added, you can see that the information of the RTV server is displayed on the [Detail].

Edit an RTV toolchain

1. Select the toolchain to be edited and click the [Edit] button.



2. Edit the RTV toolchain information.
3. Click the [Finish] button.

Duplicate an RTV toolchain

The duplication for an RTV toolchain is not provided.

Remove an RTV toolchain

Select the toolchain to delete and click the [Remove] button.

> ! If you have a project associated with the toolchain you want to delete, you can not use the project normally after deleting the toolchain.

Import/Export an RTV toolchain

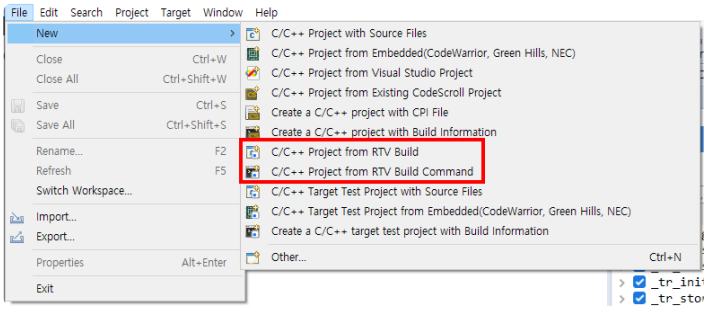
The RTV toolchain can not be imported/exported because it can be added through the RTV server.

26.3. Create a RTV Project

Create an RTV project to test the source files that exist in the RTV environment.

When building source files in the RTV environment, you can extract the information needed to create the RTV project.
This chapter describes how to create an RTV project by using pre-extracted source file build information(using the Controller Tester RTV Server Utility) in the RTV environment, or extracting information at the same time as building the source file.

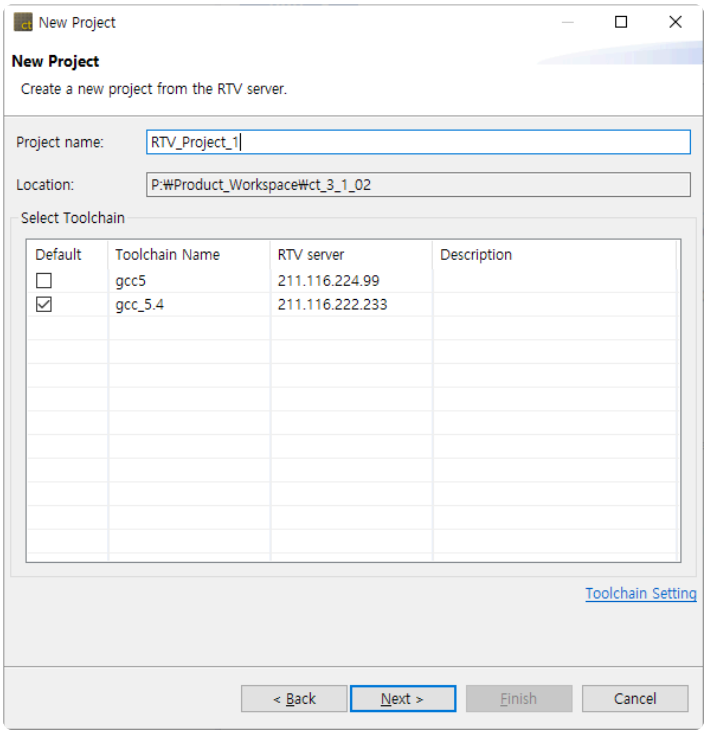
To create an RTV project, run the [Create Project Wizard].
Click [File] -> [New] or Shortcut.



C/C++ Project from RTV Build

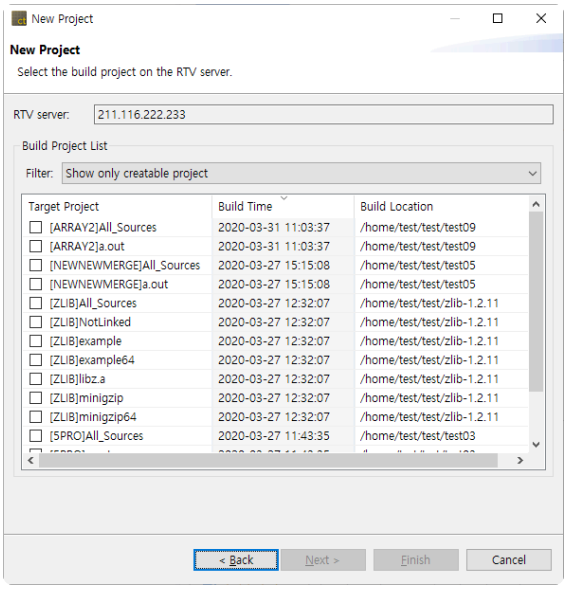
It is selected when creating an RTV project using pre-extracted source file build information (using the Controller Tester RTV Server Utility) in the RTV environment.

1. Enter the project name in [Project name].



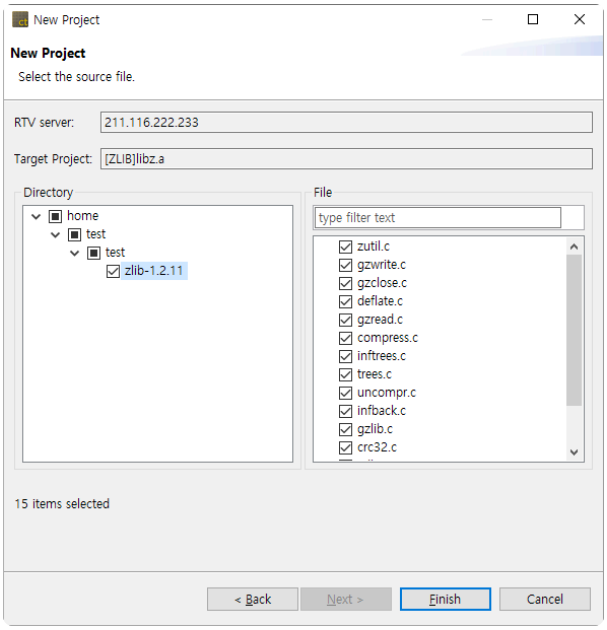
2. A list of RTV toolchains appears in the [Select Toolchain]. Select the RTV toolchain to use.

3. After selecting the toolchain, then click the [Next] button. You can see the list of source file builds pre-extracted from the RTV server.



> Among the generated projects, ‘All_Sources_Project Name’ is a project that combines all the source files of the modules captured by ctbuild. Since this project is a combined project without considering whether it is built or not, it may not be able to perform the test normally.

4. In the [Build Project List], select the item for which you want to create the RTV project and click the [Next] button.



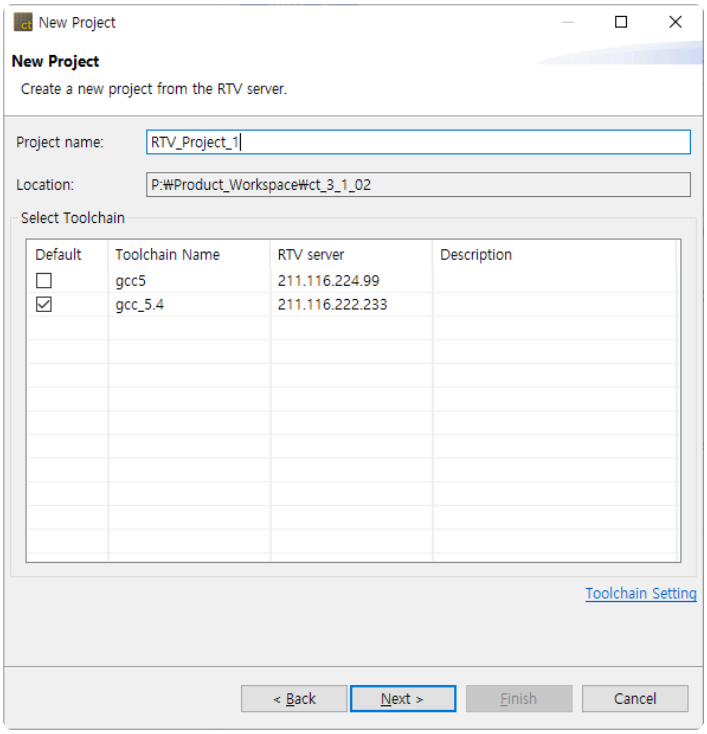
5. After selecting source files to be tested, click the [Finish] button to create the project.

C/C++ Project from RTV Build Command

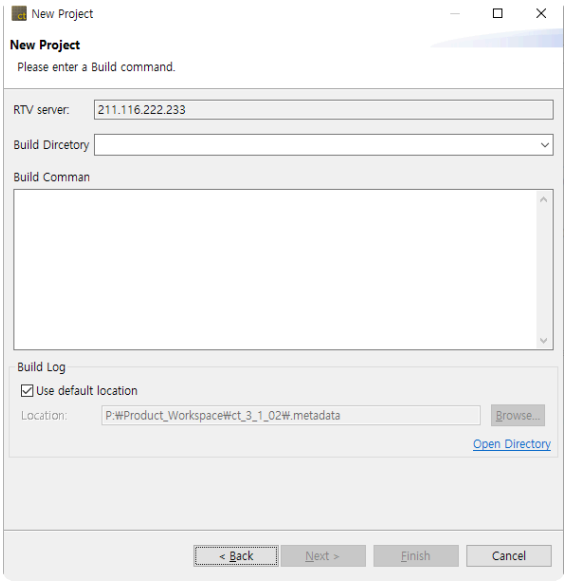
It is selected when creating an RTV project by extracting information at the same time as building the source file. The user builds the source file on the RTV server using the build command and then creates

the RTV project.

1. Enter the project name in [Project name].

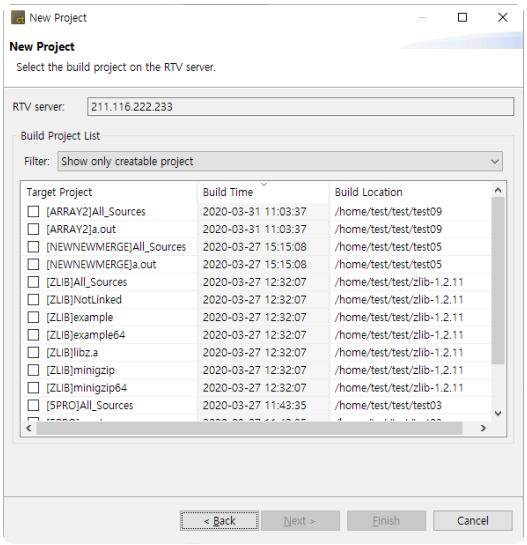


2. A list of RTV toolchains appears in the toolchain list. Select the RTV toolchain to use.
3. After selecting the toolchain, click the [Next] button. The build will be performed on the RTV server from which the selected toolchain was imported.



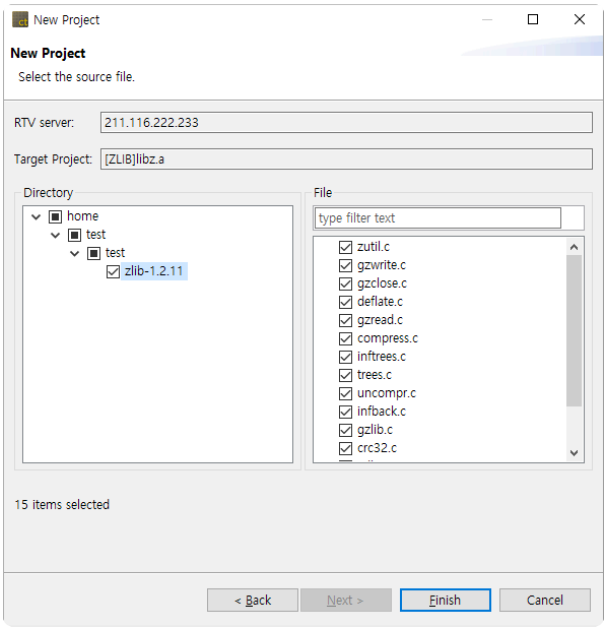
4. Enter the path to build from the RTV server (for example, the directory where the make file is located), and enter the build command.
5. In [Build Log], specify the path to the build results file.
   The default location is the .metadata directory of the workspace. If you uncheck [Use default location], you can change the path.
6. After entering the required information for the build, click the [Next] button. It builds the source file on the RTV server with the entered build command (depending on the build target, it can take a lot of time).

7. When the build is complete, a list of built projects appears.



> Among the generated projects, ‘All_Sources_Project Name’ is a project that combines all the source files of the modules captured by ctbuild. Since this project is a combined project without considering whether it is built or not, it may not be able to perform the test normally.

8. In the [Build Project List], select the item for which you want to create the RTV project and click the [Next] button.



9. When creating the RTV project, select the source file to be included and click the [Finish] button to create the project.

RTV build command

To create a project from build information, you must use the ctbuild command on the build command.

ctbuild command

The cs is a utility that automatically extracts the Controller Tester project configuration information of the source files that are built so that the Controller Tester RTV project can be created.

How to use ctbuild

Prefix the existing build command with ctbuild.

- Ex) `make` → `ctbuild capture make`

ctbuild option

new: Generate the necessary information before extracting build commands. Specify the toolchain with the `--toolchain` option and the project name with the `--project` option.

- Ex) `ctbuild new --toolchain=gcc5.4 --project=PRJ`

> ! The `--toolchain` option is required.

capture: Creates an RTV project by performing build command extraction.

- Ex) `ctbuild capture make`

convert: Convert a STATIC project to an RTV project.

- Ex) `ctbuild convert`

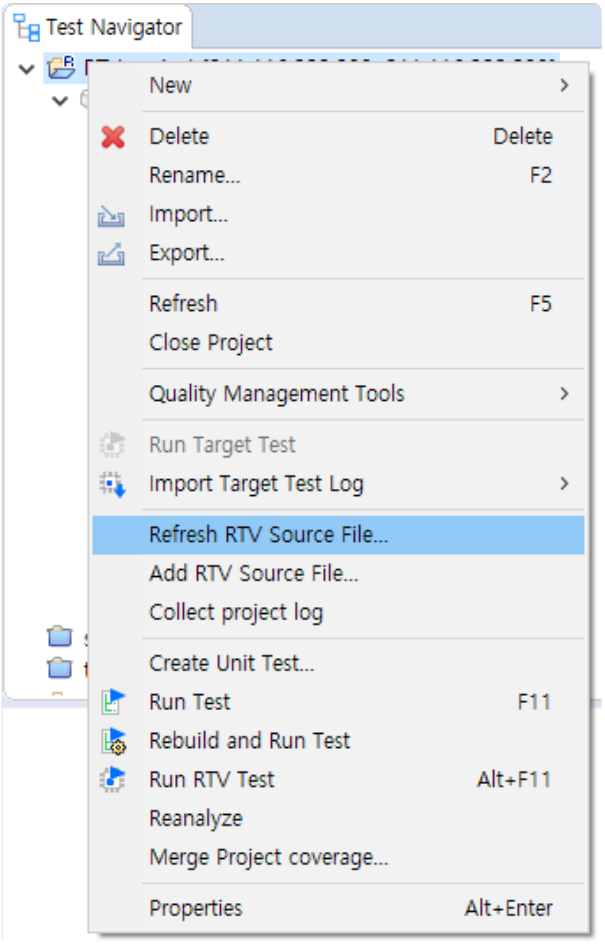
26.4. Refresh/Add RTV Source Files

Refresh RTV source file

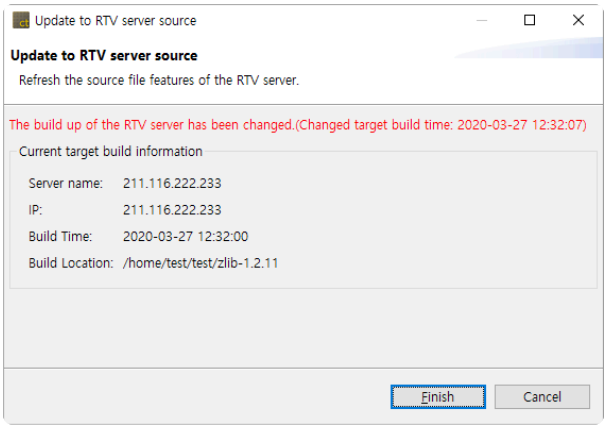
If the source file is modified and rebuilt on the RTV server, the [Refresh RTV Source File] allows the revised content to be reflected in the project.

> ! If the source file is modified but not rebuilt, the changed contents will not be reflected.
> If you modify the source files on the RTV server and then rebuild, you must perform a full build.

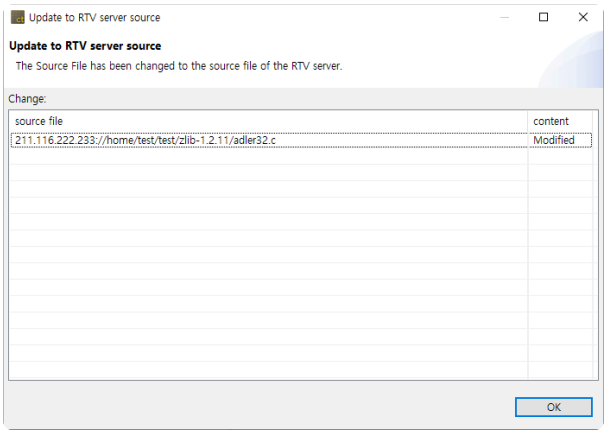
1. In the [Test Navigator] view, click the [Refresh RTV Source File] context menu.



2. When you create an RTV project, you can see the latest build information for the source files you selected (display a change notification if the build has changed). Click the [Finish] button to confirm your changes.



3. After confirming the changes, click the [OK] button.

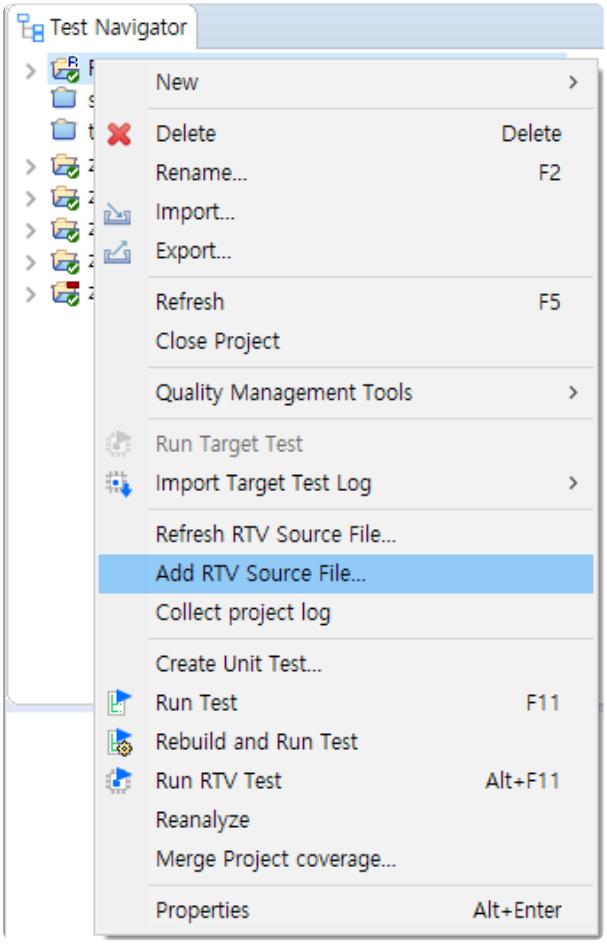


Add RTV source file

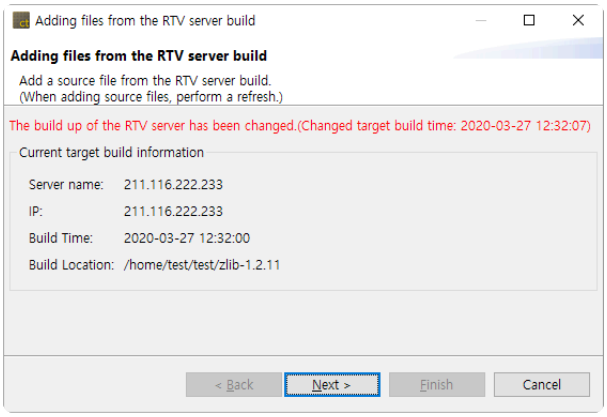
You can add source files to your RTV projects via the [Add RTV Source File] if new source files are added and built on the RTV server, or if only some source files are selected when you create the RTV project.

> ! If the source file is added but not rebuilt, the changed contents will not be reflected.
> If you build the target server after adding the source file, you must perform a full build.

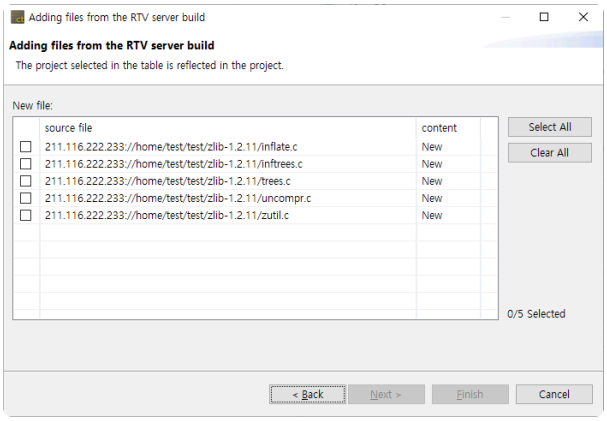
1. In the [Test Navigator] view, click the [Add RTV Source File] context menu.



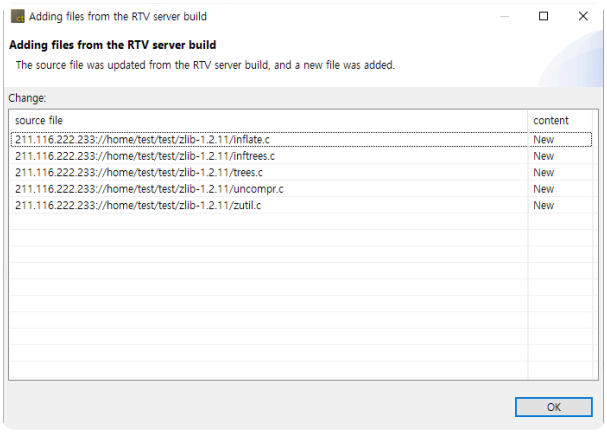
2. When you create the RTV project, you can see the latest build information for the selected source file (show change notification if the build has changed). Click the [Next] button to confirm the new file list.



3. In the [New file], select the file you want to add and click the [Finish] button.



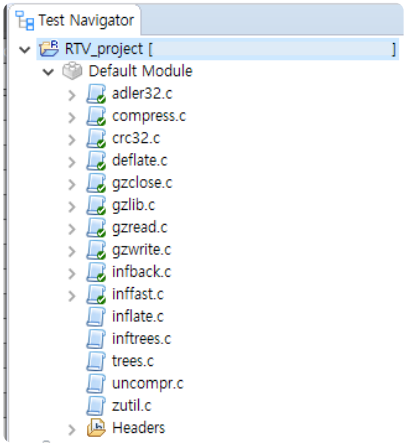
4. After confirming the changes, click the [OK] button.



26.5. RTV Project Information

Test navigator

1. The icon for the RTV project is displayed as .
2. The RTV server's information ([server name: IP]) is displayed after the name of the RTV project.

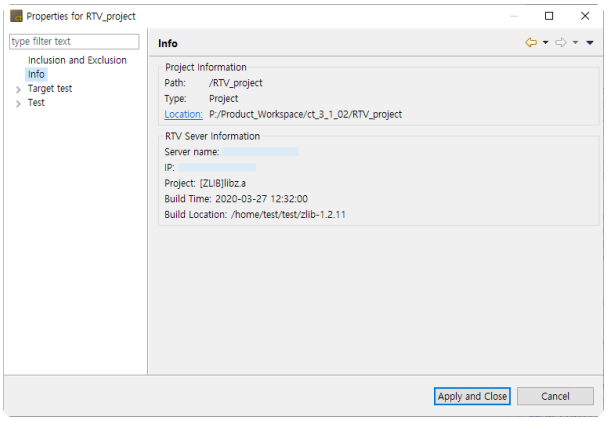


3. Source files in the RTV project can not be modified(the source file editor of the RTV project is read-only).

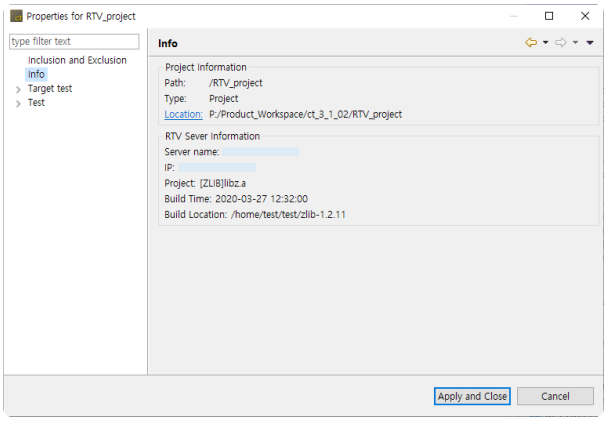
Project property

Info

1. The RTV project displays the RTV server and project information.



2. If the general project has set the toolchain as the RTV toolchain, information about the RTV server from which the RTV toolchain was imported is displayed.



Module/File property

Build

1. The RTV project can not change the RTV toolchain that was set when the project was created.
2. However, if the RTV toolchain you set originally is deleted, you can change it to another toolchain.
3. In general projects, you can change the toolchain to the RTV toolchain. Changing to the RTV toolchain changes all toolchain settings (modules and source files) under the project and allows you to run the RTV tests.
4. The project with the RTV toolchain can not change the compiler settings of the source file (the module’s linker configuration can be changed).

26.6. Run RTV Test

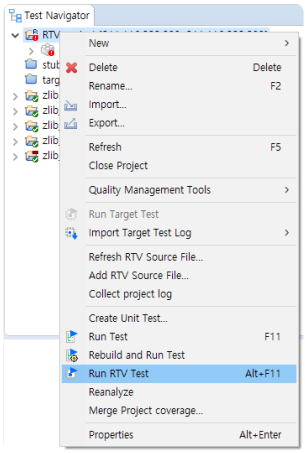
The RTV project allows you to run RTV tests without any additional setup. To test a general project in the RTV environment, you must set the toolchain of the project to the RTV toolchain created from the RTV server to be tested.

Run all RTV tests

It runs the selected tests in the [Unit Test] view and the [Integration Test] view.

RTV tests can be run in the following way.

1. Select the project to be tested, right-click it and click [Run RTV Test].

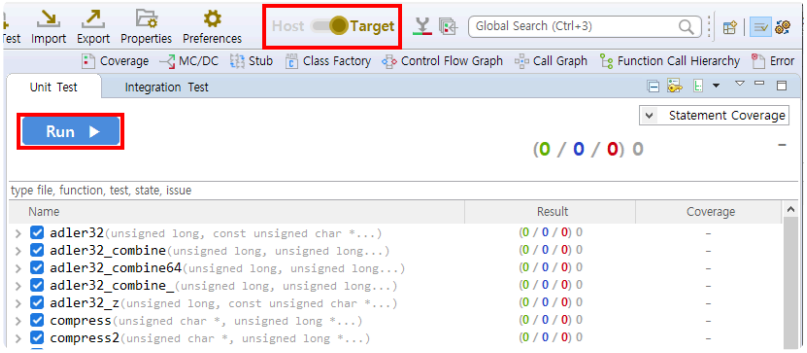


Run RTV unit test

Runs the selected test in the [Unit Test] view.

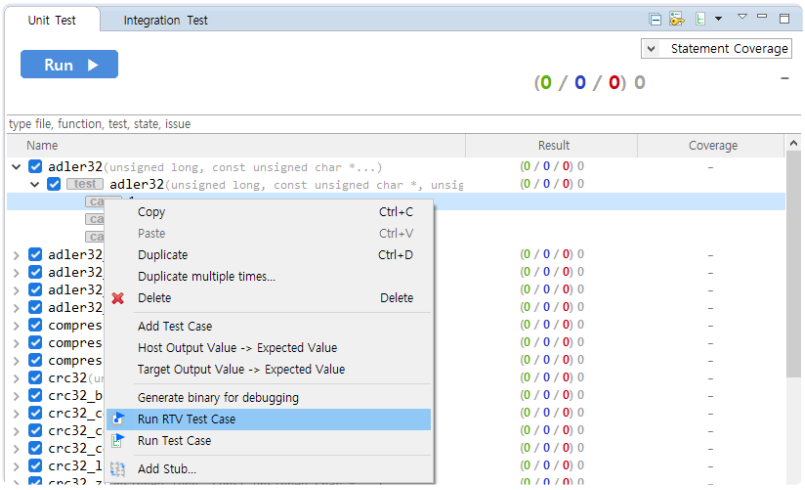
Run RTV test

[Run] button In the [Unit Test] view.



Run RTV test case

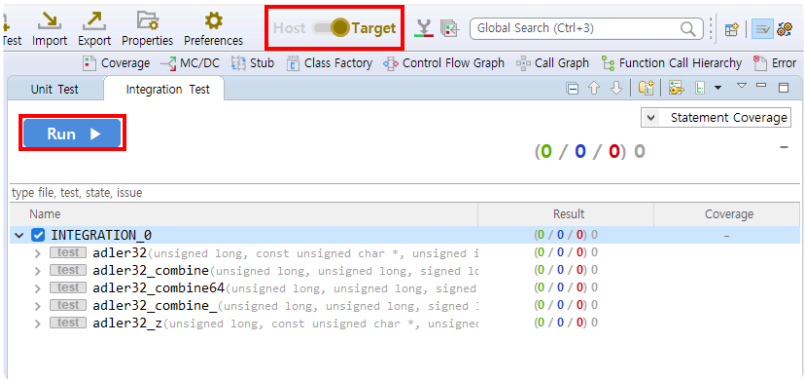
[Run RTV Test Case] context menu (multi-selectable) in the [Unit Test] view.



Run RTV integration test

Runs the selected test in the [Integration Test] view.

[Run] button In the [Integration Test] view.



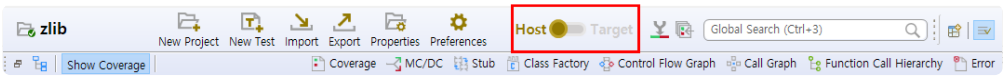
26.7. RTV Test Results

Coverage

You can change the coverage information (host, target, host/target merged) displayed in each view and editor through the coverage view menu in the top toolbar menu.
You can check the detailed information of each view in Test Perspective.

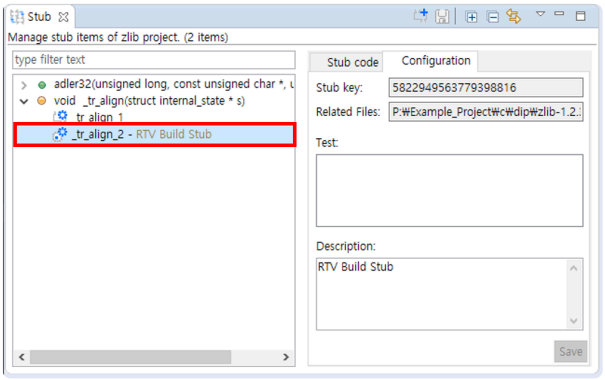
Test results

You can change the test results through the host/target settings menu on the dashboard.



Stub

The RTV build stubs that are created when running the RTV test is distinct from the build stubs which are created when running the host test.



Type	Description
Target build stub	Auto-created stub during RTV tests

27. CONTROLLER TESTER RTV(Remote Target Verifier) Target Plug-in

Using Controller Tester RTV (Remot Target Verifier) Target Plug-in, software developed in Linux environment can be tested in a real embedded target environment. You can run RTV target after installing Controller Tester RTV server in Linux and setting target environment in Controller Tester. For information on how to install the Controller Tester RTV server in your Linux environment, please contact the Technical support contact on bottom of the Troubleshooting page of this document.

The followings are how to create Controller Tester RTV Target Plug-in project and run a test.

- RTV Server Settings
- RTV Toolchain Settings
- RTV Target Environment Settings
- Create a RTV Target Project
- Refresh/Add RTV Target Source Files
- RTV Target Project Information
- Run RTV Target Test

Supported Operating Systems

Server: Linux (RHEL, Ubuntu, Debian, Fedora based)
Client: Windows 7/10

27.1. RTV Server Settings

For RTV server setting to use Controller Tester RTV Target Plug-in, refer to RTV Server Settings.

27.2. RTV Toolchain Settings

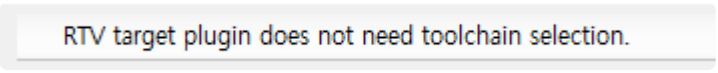
For RTV toolchain setting to use Controller Tester RTV Target Plug-in, refer to RTV Toolchain Settings.

27.3. RTV Target Environment Settings

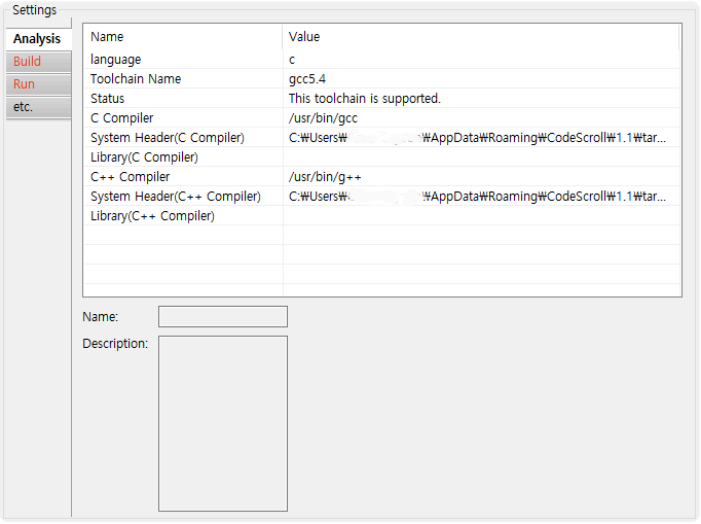
To run tests in a target environment using Controller Tester RTV Target Plug-in, you need to enter information about the target environment as with Controller Tester Target Plug-in. Using information about the target environment entered by the user, Controller Tester builds a test harness in RTV server and run in the target environment. Target log collector can get the test results automatically.
The target environment can be set in the project properties page or in the new RTV target test project wizard.

Target environment settings.

Target environment settings are divided into basic information and detailed settings.
RTV target project don't need to enter the basic information.



After entering the basic information, detailed settings that require input are displayed.



Target environment detailed settings

Target environment detailed settings are divided into the analysis, build, run, and etc.

Category	Description
Analysis	The toolchain information is displayed, and in the case of a target test project, the target compiler related settings required for analysis are displayed.
Build	The settings for building the test software are displayed.
Run	Settings for running tests and getting results in the target environment are displayed.
etc.	Other settings are displayed. (Program entry point, etc.)

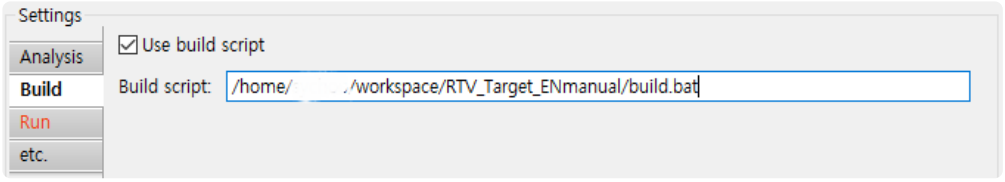
Required settings for each category are indicated in red. Depending on whether the required settings for each category are entered or not, the behavior when clicking the [Run] button is different as shown

below.

Required settings completed category	Description
None	Test run impossible
Analysis	If you click [Run], it overwrites the original source code with the test code. To perform the test, the user must manually build and run the test.
Analysis, Build	If you click [Run], it builds the test code. To perform the test, the user has to manually run the test on the target.
Analysis, Build, Run	Automatically run tests in the target environment.

Target environment detailed settings – Build

Enter a build script path of RTV server into build script area. After entering the build script path, Controller Tester checks validity of the path. If the path is invalid, build script area is displayed in red and if vaild, it is displayed in black.

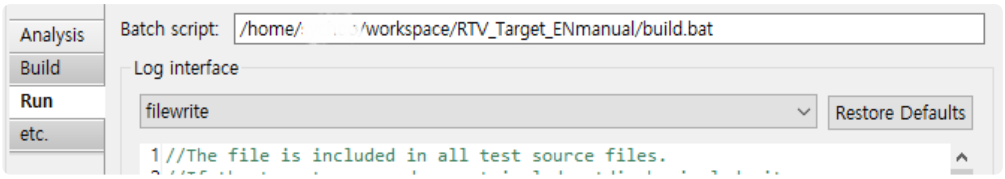


Target environment detailed settings – Run

Run setting is divided into batch script area and log interface area. Controller Tester checks validity of batch script area as with build script area. Informations to get results of RTV target test from target environment is entered in log interface area.

Batch script area

Enter a batch script path of RTV server into batch script area as with build script area.

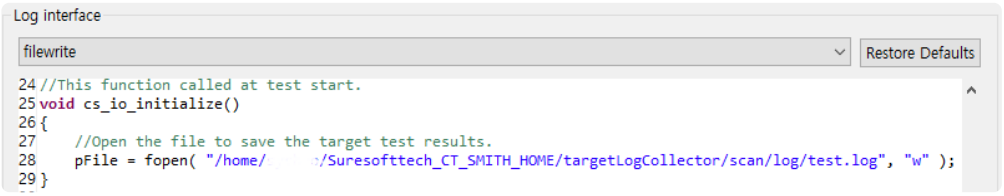


Log interface area

The log interface area is a setting to record test result from target environment. The followings are how to get target log into Target Log Collector.

- File
- Socket communication (TCP, UDP)
- Serial communication (UART)

To allow target log collector to get test results during recoding the results in file, you modify a path in log interface into `%SMITH_HOME%/targetLogCollector/scan/log/%file_name%`.

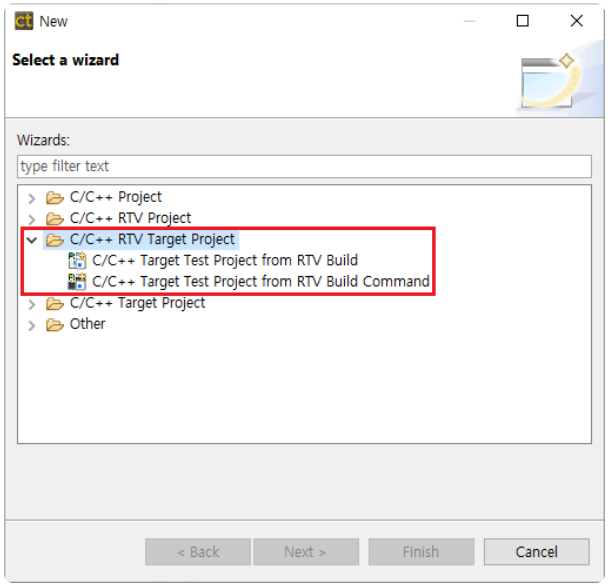


27.4. Create a RTV Target Project

Create an RTV project to test the source files that exist in the RTV environment.

When building source files in the RTV environment, you can extract the information needed to create the RTV project.
This chapter describes how to create an RTV target project by using pre-extracted source file build information(using the Controller Tester RTV Server Utility) in the RTV environment, or extracting information at the same time as building the source file.

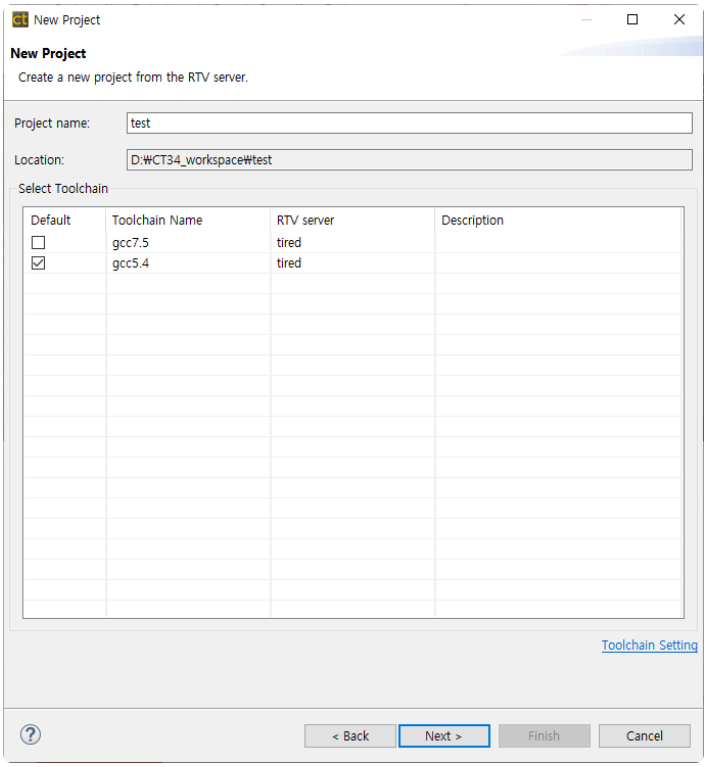
To create an RTV project, run the [Create Project Wizard].
Click [File] -> [New] or Shortcut.



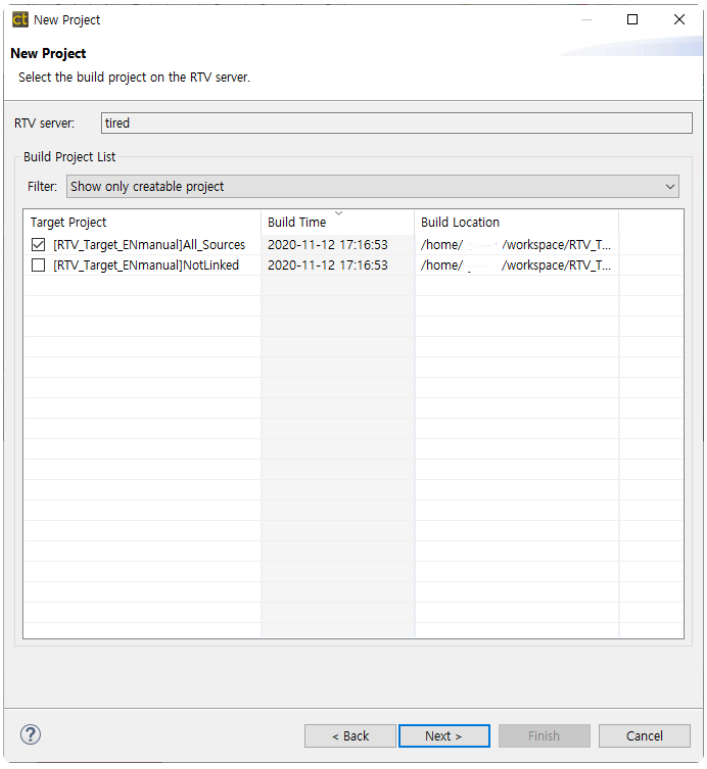
C/C++ Target Test Project from RTV Build

It is selected when creating an RTV target project using pre-extracted source file build information (using the Controller Tester RTV Server Utility) in the RTV environment.

1. Enter the project name in [Project name] and select the RTV toolchain to use in the [Select Toolchain]. After selecting the toolchain, then click the [Next] button.

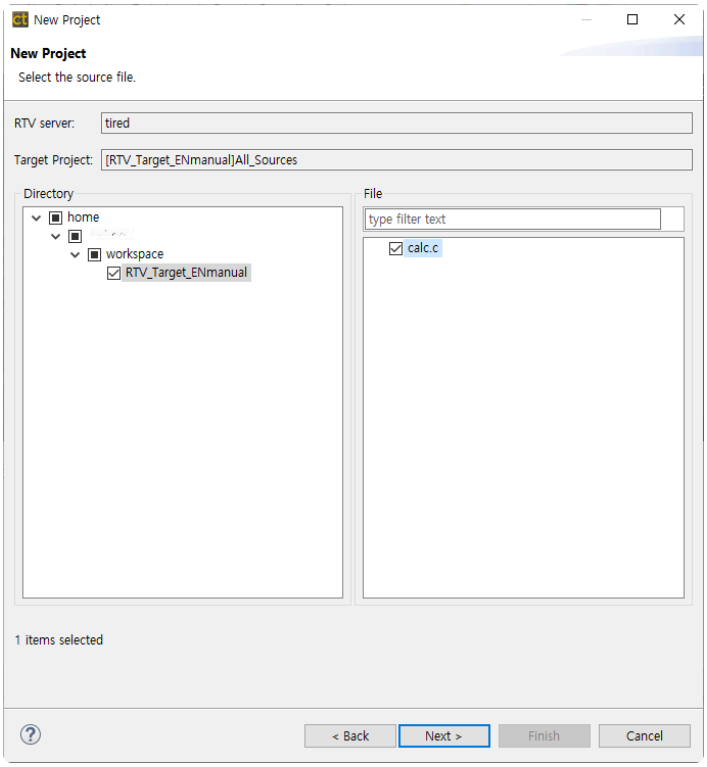


2. You can see the list of source file builds pre-extracted from the RTV server. In the [Build Project List], select the item for which you want to create the RTV project and click the [Next] button.

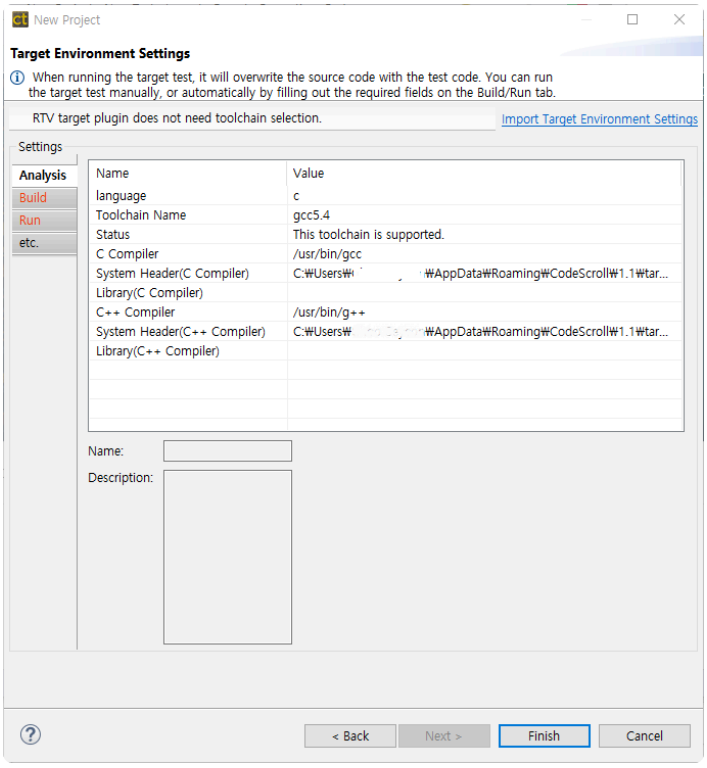


> ✻ Among the generated projects, ‘All_Sources_Project Name’ is a project that combines all the source files of the modules captured by ctbuild. Since this project is a combined project without considering whether it is built or not, it may not be able to perform the test normally.

3. After selecting source files to be tested, click the [Next] button.



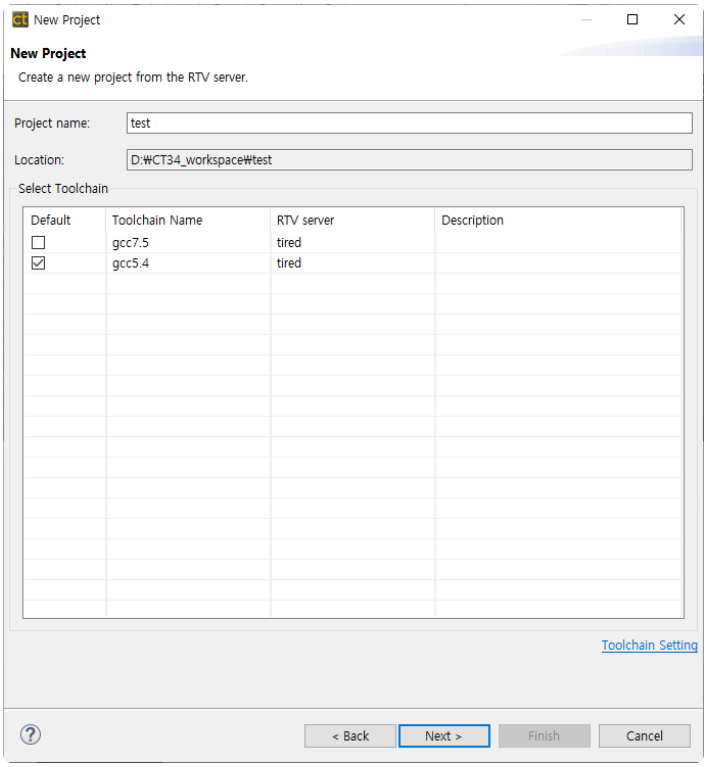
4. After entering the detailed settings for each category and click the [Next] button



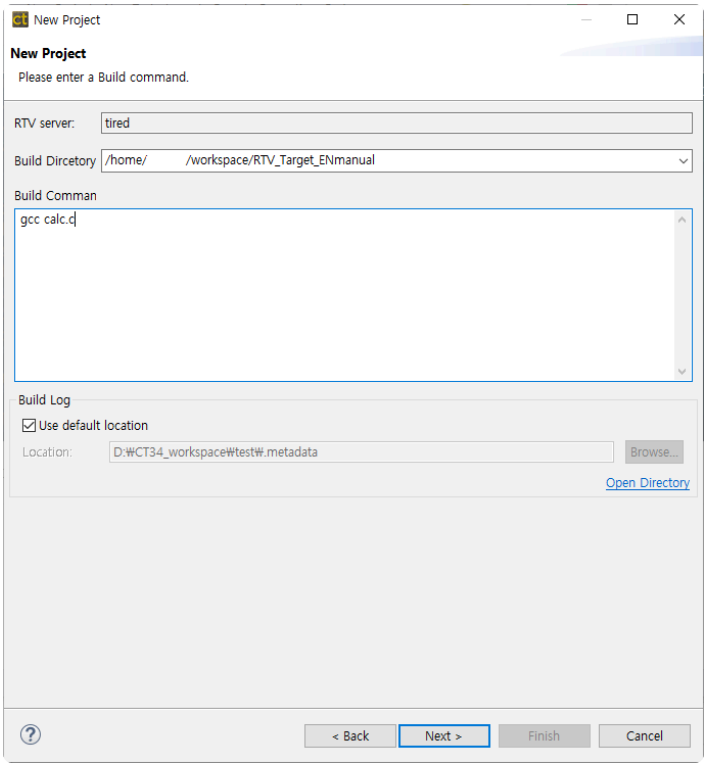
C/C++ Target Test Project from RTV Build Command

It is selected when creating an RTV target project by extracting information at the same time as building the source file. The user builds the source file on the RTV server using the build command and then creates the RTV target project.

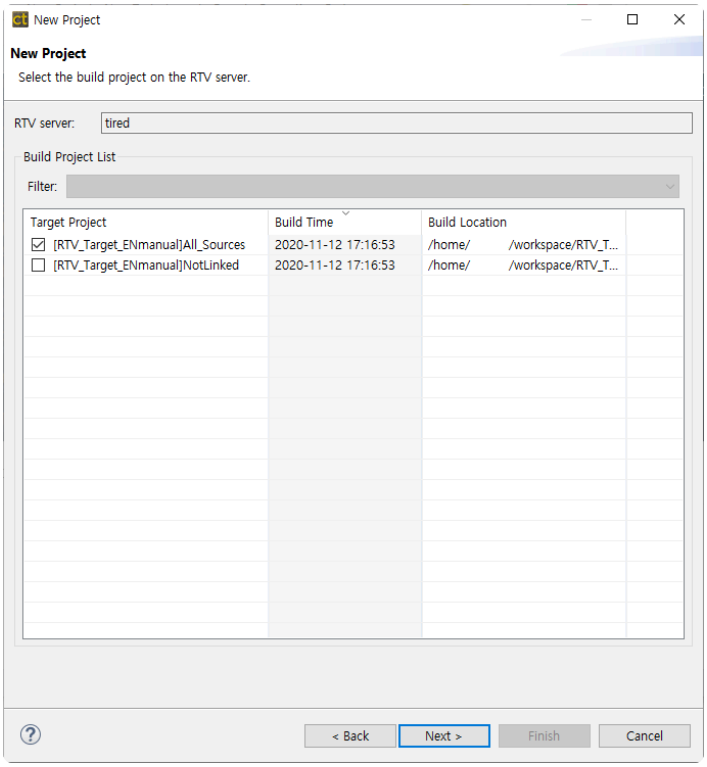
1. Enter the project name in [Project name] and select the RTV toolchain to use in the [Select Toolchain]. After selecting the toolchain, then click the [Next] button.



2. Enter the path to build from the target server (for example, the directory where the make file is located), and enter the build command. In [Build Log], specify the path to the build results file. After entering the required information for the build, click the [Next] button. It builds the source file on the target server with the entered build command (depending on the build target, it can take a lot of time).

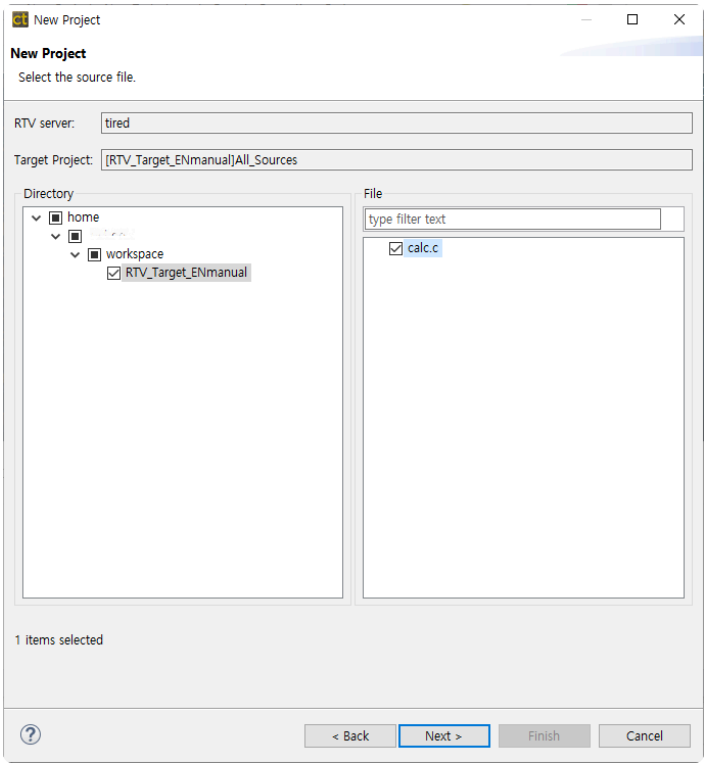


3. When the build is complete, a list of built projects appears. In the [Build Project List], select the item for which you want to create the RTV target project and click the [Next] button.

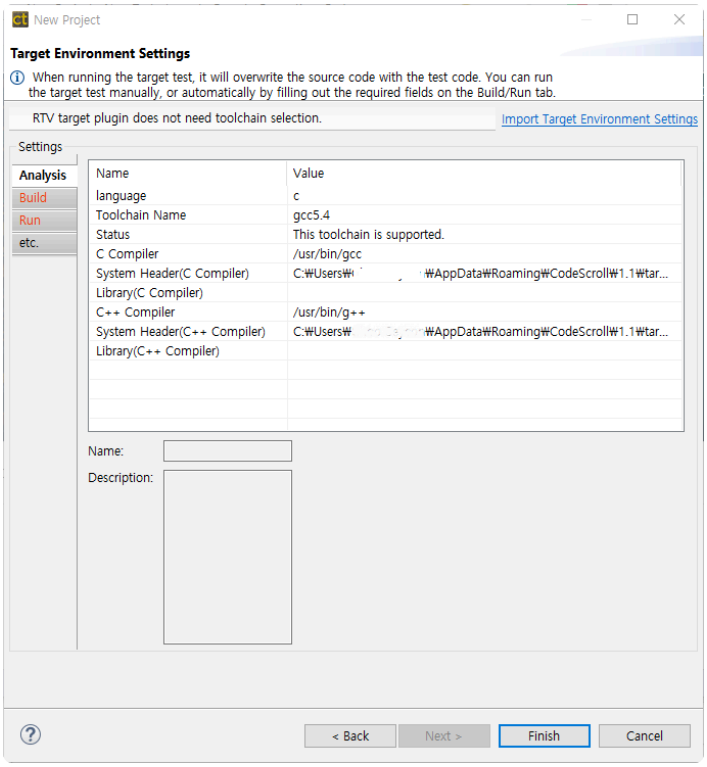


Among the generated projects, 'All_Sources_Project Name' is a project that combines all the source files of the modules captured by ctbuild. Since this project is a combined project without considering whether it is built or not, it may not be able to perform the test normally.

4. After selecting source files to be tested, click the [Next] button.



5. After entering the detailed settings for each category and click the [Next] button



RTV build command

To create a project from build information, you must use the ctbuild command on the build command. For RTV build command, refer to [RTV build command] in Create a RTV Project

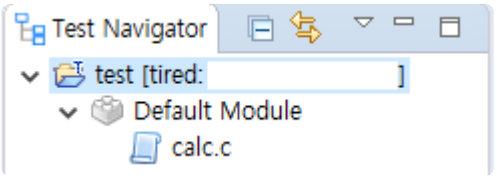
27.5. Refresh/Add RTV Target Source Files

If the source file is modified or added on the RTV server, the [Refresh RTV Source File] and [Add RTV Source File] allow the revised content to be reflected in the project. For Refresh/Add RTV Target Source Files, refer to Refresh/Add RTV Source Files.

27.6. RTV Target Project Information

Test navigator

- The icon for the RTV project is displayed as .
- The RTV server's information ([server name: IP]) is displayed after the name of the RTV target project.

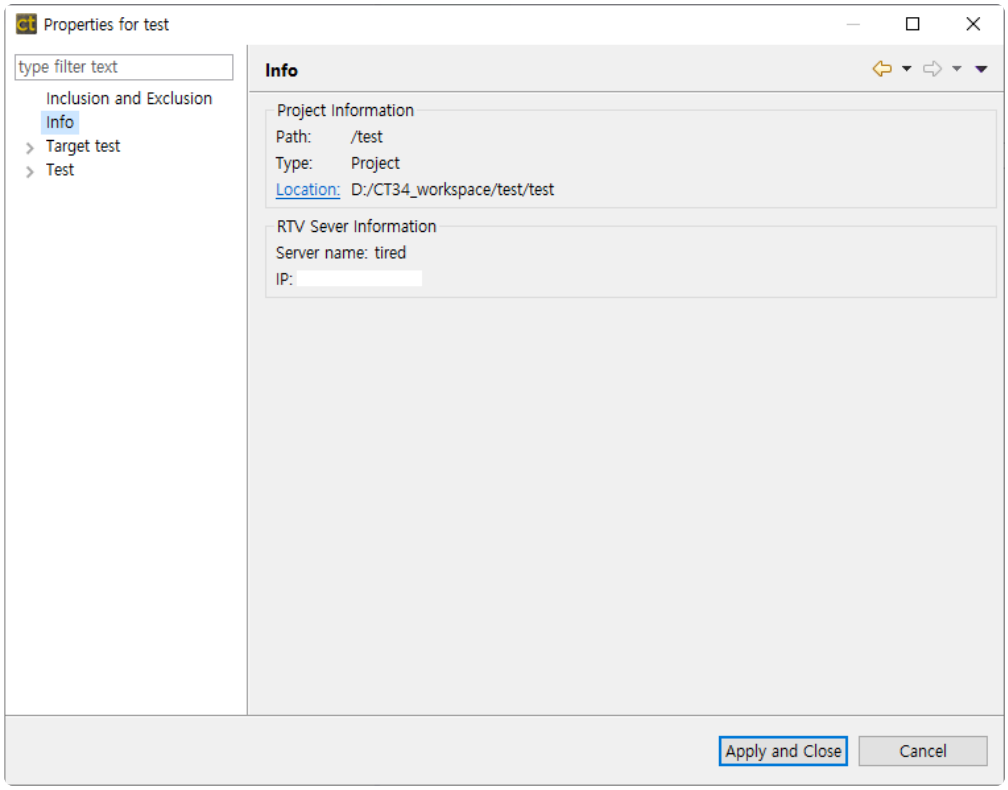


- Source files in the RTV project can not be modified(the source file editor of the RTV project is read-only).

Project properties

Info

The RTV target project displays the RTV server and project information.



Target test

The RTV target project can set about target test run and RTV environment. For target test run settings, refer to Target Test Run Settings and for RTV environment settings, refer to Target Environment Settings.

Module/file properties

For properties about module or file, refer to RTV Project Information.

27.7. Run RTV Target Test

Test run

The binary file built in Linux environment can be tested in target environment using RTV Target Plug-in. For running the RTV target test, refer to Run RTV Test.

28. DISCOVERY Plug-in

DISCOVERY automatically generates test cases to increase coverage.

The process of executing DISCOVERY is as follows.

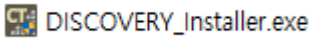
- DISCOVERY Install
- DISCOVERY Execution

28.1. DISCOVERY Install

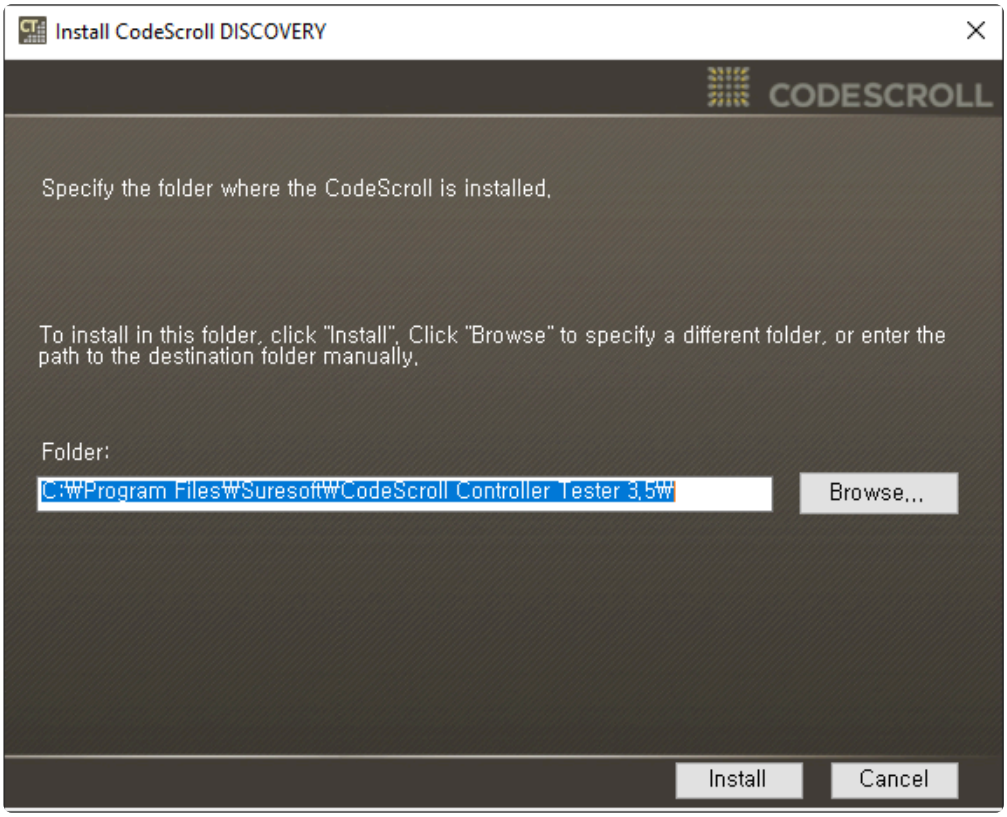
DISCOVERY Install

By using the install package, you can install DISCOVERY as follows:

1. Execute DISCOVERY_Installer.exe file.

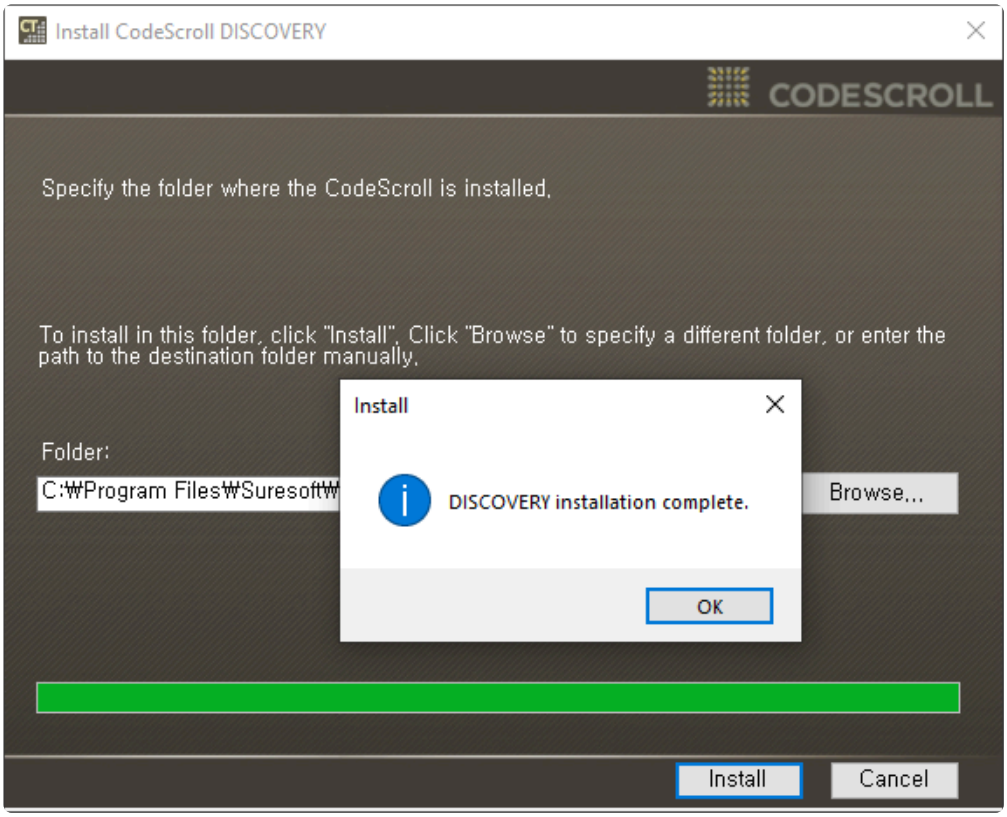
   

2. Set the path where Controller Tester is installed and click [Install].

> ! If the installation path of Controller Tester is wrong, it will not be installed. If the version of Controller Tester installed is lower than the version of Controller Tester required to install DISCOVERY, it will not be installed.

3. After installation is complete, click [OK] to automatically run Controller Tester.


Install CodeScroll DISCOVERY
CODESCROLL
Specify the folder where the CodeScroll is installed.
To install in this folder, click "Install". Click "Browse" to specify a different folder, or enter the path to the destination folder manually.
Folder:
C:₩Program Files₩Suresoft₩
Browse...
Install
DISCOVERY installation complete.
OK
Install
Cancel

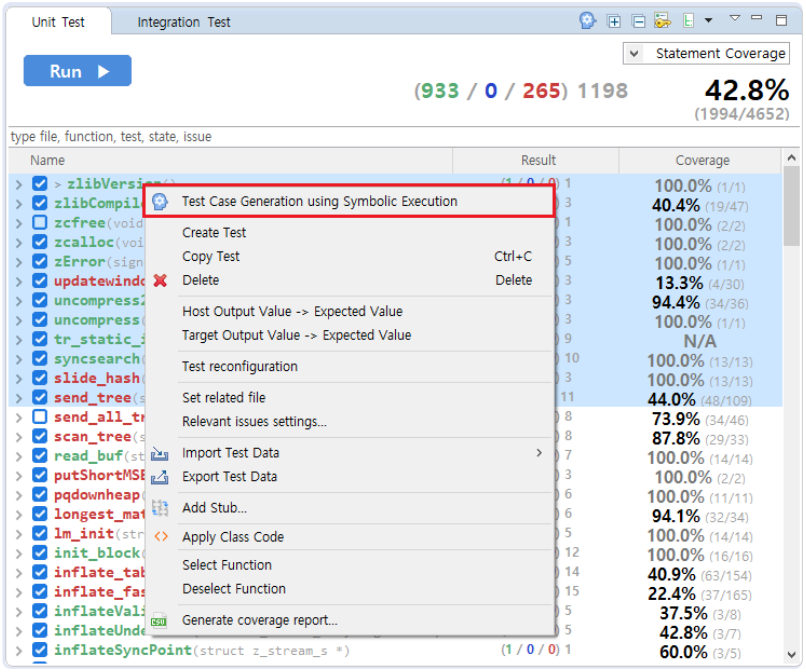
28.2. DISCOVERY Execution

If you use the [Test Case Generation using Symbolic Execution] feature, test cases that can increase coverage are automatically generated.

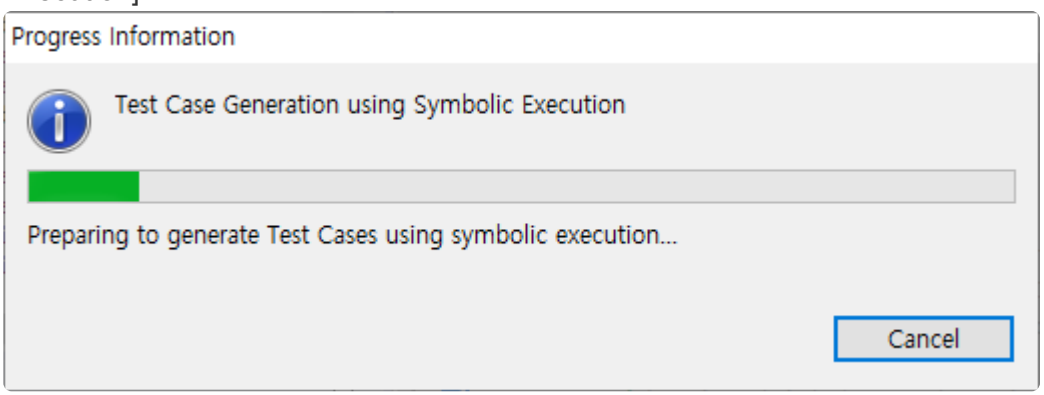
Test Case Generation using Symbolic Execution

1. Generate test cases based on symbolic execution for the selected tests.
   - Toolbar in Unit Test View
     If you click [Test Case Generation using Symbolic Execution] on the toolbar of Unit Test View, test cases are generated and executed based on symbol execution for the function/test checked in Unit Test View.

   - Context menu in Unit Test View
     Right-clicking a function/test and selecting the [Test Case Generation using Symbolic Execution] menu generates and executes a test case based on the symbolic execution.

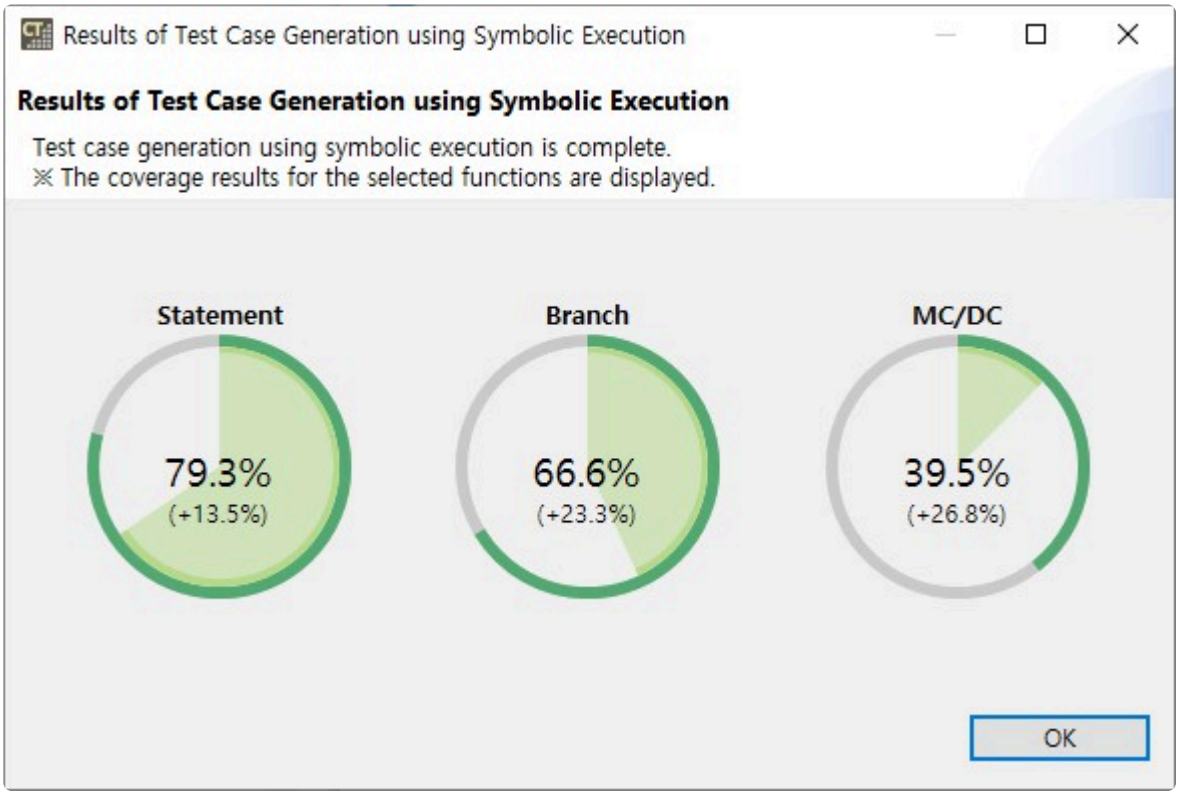


2. You can check the progress information after clicking [Test Case Generation using Symbolic Execution].



> ! When generating a test case based on symbolic execution, it may take a long time to generate test cases.

3. When the test case generation is completed, you can check the coverage through the [Test Case Generation using Symbolic Execution] dialog.



29. CONTROLLER TESTER CLI(Command Line Interface)

You can run Controller Tester in CLI environment.

The process of executing Controller Tester in CLI environment is as follows.

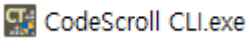
- Install CLI
- Project Import/Export
- CLI Settings
- CLI Execution

29.1. CLI Install

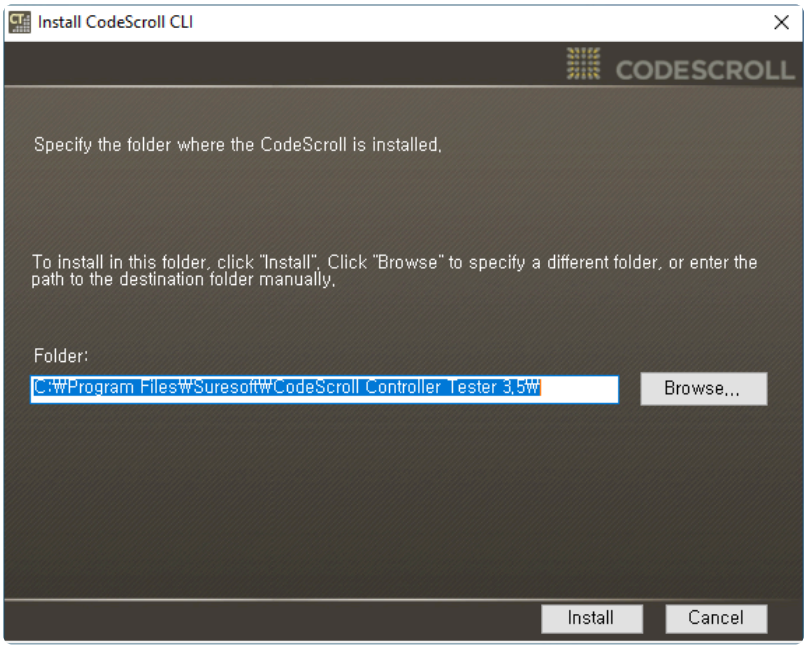
Install CLI

By using the install package, you can install CLI as follows:

1. Run the `CodeScroll CLI.exe` file.

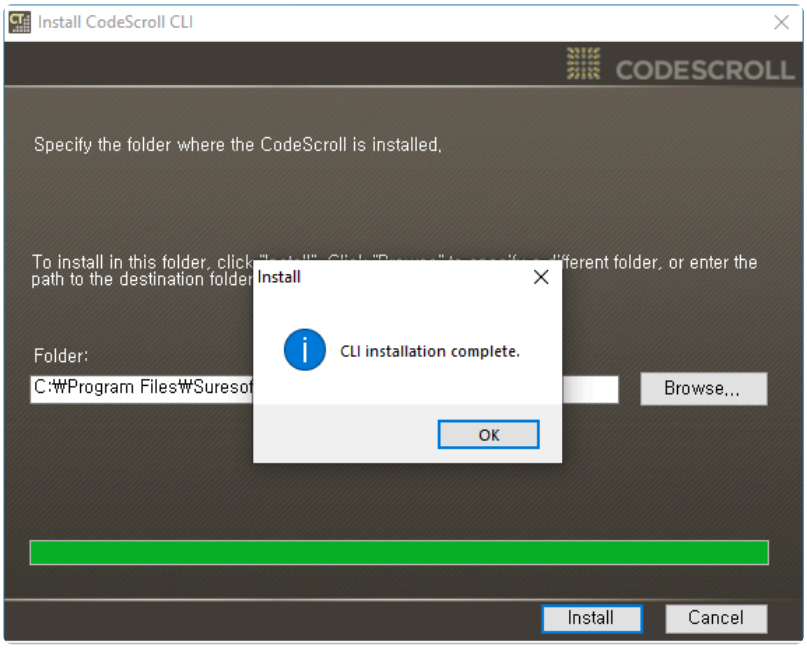
   

2. Enter the path where Controller Tester is installed and click [Install].

> ! If the installation path of Controller Tester is incorrect, it will not be installed.

3. CLI is installed.


Install CodeScroll CLI
CODESCROLL
Specify the folder where the CodeScroll is installed.
To install in this folder, click "Install". Click "Browse" to select a different folder, or enter the path to the destination folder.
Folder:
C:₩Program Files₩Suresof
Browse...
Install
Cancel
Install
CLI installation complete.
OK

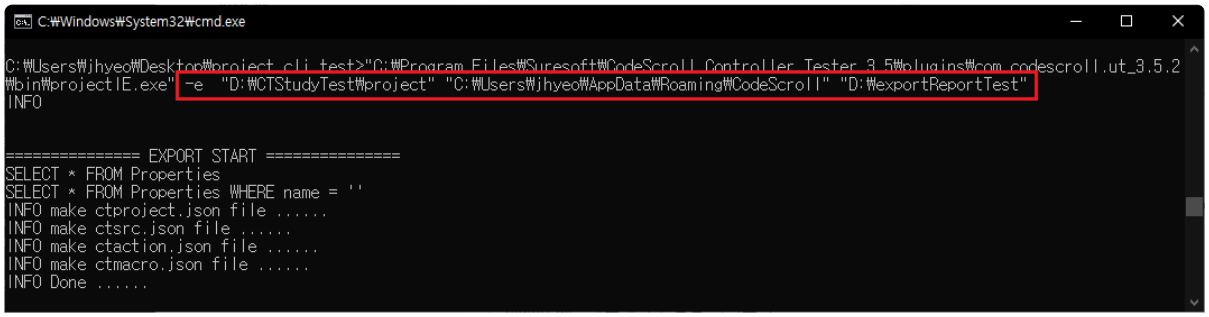
29.2. Project Import/Export

You can import and export projects in the CLI environment.

Export command

You can export a project using the `-e, --export <project top dir> <global top dir> <export dir>` command.

- projectIE.exe path: Controller Tester installation path\plugins\com.codescroll.ut_ut_3.x.x\bin.
- project top dir: export target project directory.
- global top dir: tool global directory.
- export dir: Export output directory.



Export output

- **io**: Folder containing the interface implementation file used for target export. Appears on the global path and does not create it if it does not exist.
- **target_server**: Folder containing RTV target server information. Appears on the global path and does not create if it does not exist.
- **factory**: Folder used for class factory function. Appears in the project path and does not create if it does not exist.
- **pa.ini, ut.ini**: Project analysis and test configuration files.
- **build.mk**: RTV project compilation flag file.
- **link.mk**: RTV project link flag file.
- **ctproject.json**: Project setting information and exclusion information.
  - infromation: Project information.
  - exclusions: [Project] – [Properties] – [Inclusion and Exclusion] – [Excluded].
  - Inclusions: [Project] – [Properties] – [Inclusion and Exclusion] – [Included].
  - globalExclusions: [Window] – [Preferences] – [Exclusion].
  - utCoverageExclusions: [Project] – [Properties] – [Test] – [Exclusion of coverage].
  - utUserCodeInsertNodes: Fault Injection view.
  - utClassFactories: Class Factory view.
  - properties: Project setting information.
  - globalProperties: Controller Tester global setting information.
- **ctsrc.json**: Test target source file information.
  - name: Module name.

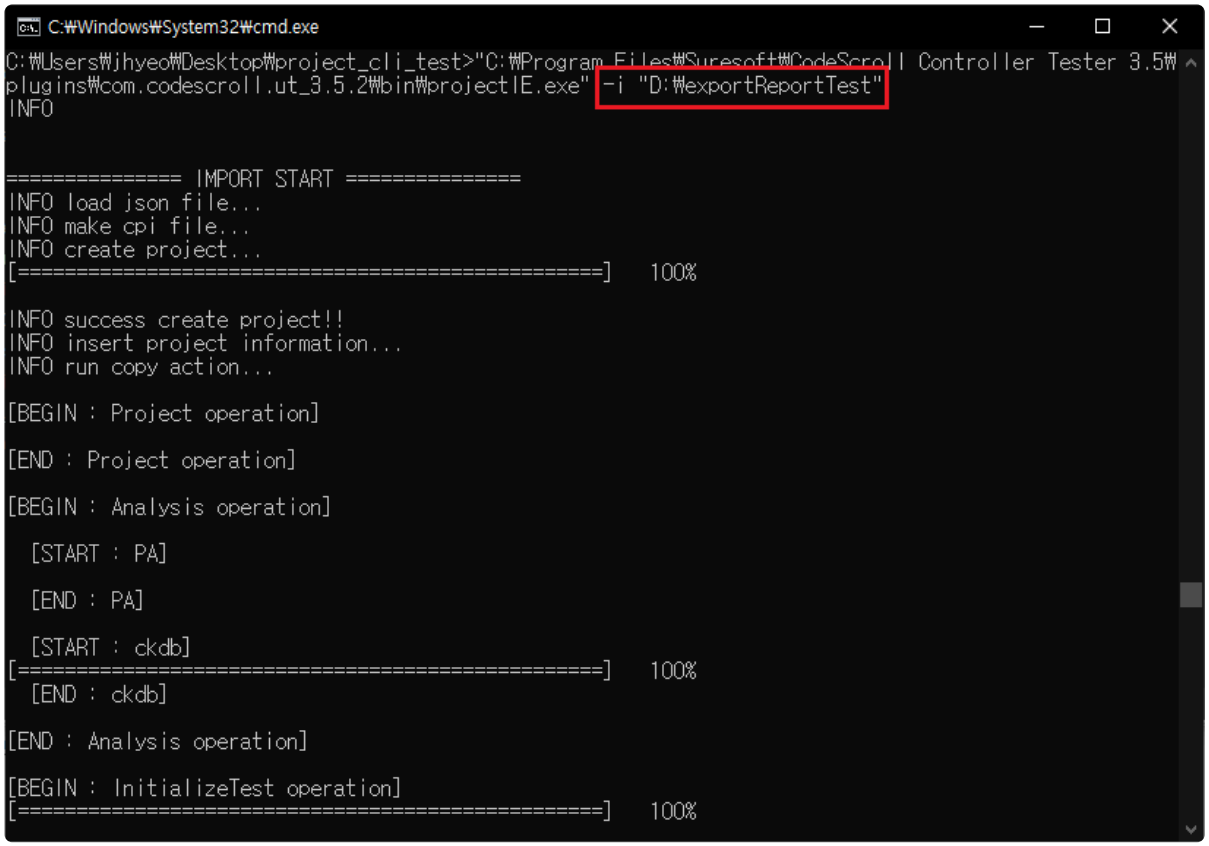
    - language: Target source language.
    - linkFlag: Link flag.
    - cmpileFlag: Compilation flag.
    - toolchainName: Toolchain name.
    - sourceTopDir: Source common parent path.
    - src.path: Source file path.
    - src.compileFlag: Compilation flag.
- **ctaction.json**: A file that defines actions to be executed for each step.
    - copy: Fields left by the existing copy action migration.
    - beforeCreateProject: Actions executed before project creation.
    - afterCreateProject: Action to be executed after project creation.
    - afterImport: Action to be executed after import is complete.
    - copy action: Description.
        - actionName: Action name.
        - from: Copy destination.
        - to: Path to be copied.
        - isDir: Whether a directory.
- **ctmacro.json**: The macro information inserted in each json file is replaced with the characters in the ctmacro.json file.
    - If you put it in the json file in the form of ${macro name}, it is replaced with the corresponding string when imported.
    - When exporting, the WORKSPACE and GLOBAL paths are created as blanks in the ctmacro.json file.
        - WORKSPACE: The workspace path of the project to be created when exporting.
        - GLOBAL: Global path of PC to be imported.

The WORKSPACE path and GLOBAL path are required.

Import command

1. You can import the project using the # `-i, --i <import resource dir>` command.
    - import resource dir: Import destination directory.
2. Import will create a .import_product folder in the destination directory. In that folder, setup files (.cpi, .ini) and .bat files used for project creation and test export import are created.

Create an exportTest folder in the import destination directory and copy the result of [Project] – [Export] – [Test] – [Export Test] to import the test when importing.

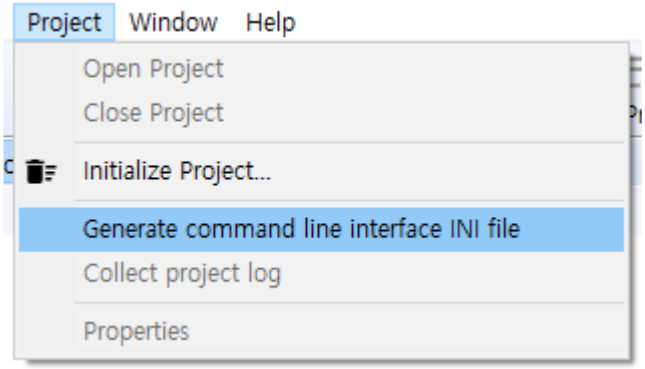

C:₩Windows₩System32₩cmd.exe
C:₩Users₩jhyeo₩Desktop₩project_cli_test>"C:₩Program Files₩Suresoft₩CodeScroll Controller Tester 3.5₩
plugins₩com.codescroll.ut_3.5.2₩bin₩projectIE.exe" -i "D:₩exportReportTest"
INFO
========== IMPORT START ==========
INFO load json file...
INFO make cpi file...
INFO create project...
[==========] 100%
INFO success create project!!
INFO insert project information...
INFO run copy action...
[BEGIN : Project operation]
[END : Project operation]
[BEGIN : Analysis operation]
[START : PA]
[END : PA]
[START : ckdb]
[==========] 100%
[END : ckdb]
[END : Analysis operation]
[BEGIN : InitializeTest operation]
[==========] 100%

29.3. CLI Settings

Creates a configuration file to run Controller Tester in CLI environment.

Creating CLI configuration file

- If you select [Project] – [Generate command line interface INI file], the projectName_CLI.ini file and path.bat file are created in the path selected by the user.



If there is no selected project, CLI.ini file and path.bat file that does not contain project information are created.

[Information]

Item	Description
workspace	Workspace path.
ct_version	Controller Tester version.

[Information] automatically fills in the values when creating the CLI configuration file.

[Project]

Item	Description
create	Whether to create a project.
cpi_file	If `create=true`, the path to the `.cpi` file for project creation.
name	Project name.
toolchain	Toolchain name (create in case of RTV project).
target_project	Target project name (create in case of RTV project).

> ! The target_project option must be entered in the same way as the target project name in [Project]-[Properties]-[Information]-[RTV Server Information].

[Analysis]

Item	Description
run	Whether to perform analysis.

[InitializeTest]

Item	Description
run	If true, delete all existing tests.

[CreateTest]

Item	Description
run	Whether to generate unit tests.
target_function	It is a function name for generating unit tests. Unit tests can be created for one or more functions. When entering multiple functions, they can be separated with a semicolon (;) or Enter. If there is no target_function list, create unit tests for all functions.

[ImportTestInfo]

Item	Description
run	Whether to fetch test information.
testinfo_file	Path to the testinfo.export file.

[CreateTestCase]

Item	Description
run	Whether to create a test case.
mode	Test case generation mode. (flat, pairwise, random)
random_testcase_count	In case of random, input the number of test cases generated.

[ExecuteTest]

Item	Description
run	Whether to run unit test.

remote_target	RTV run or not.
run_robust	If an error occurs while running a test, whether to exclude non-executable tests.

[Report]

Item	Description
run	Whether to generate a report.
output_dir	Report output path. The default output path is project path.
each_test	Whether to generate a report by test.
HTML, XLSX, PDF, DOCX, PPTX, XLS, DOC, PPT	Specify Report Format
URL	URL of the results report accessible from ITS

Individual import of test information

> By setting `run=false` in [ImportTestInfo], [ImportStub], [ImportTest], and [ImportIntegrationTest] can be executed individually.

[ImportStub]

Item	Description
run	Whether to perform stub import.
stub_file	Path to the stub file.
stub_dir	Path to the stub directory.

[ImportTest]

Item	Description
run	Whether to perform test import.
test_source_file	.utest file path. When entering multiple source file paths, they can be separated with a semicolon (;) or Enter.
test_source_dir	Path to the test source directory. If you enter the test source directory path, perform a full test import included in the directory path.

[ImportTestCase]

Item	Description
run	Whether to perform test import.
testcase_dir	Test case directory path. If you enter the test case directory path, import all test cases included in the directory path.

[ImportIntegrationTest]

Item	Description
run	Whether to perform integration tests and import of test cases.
test_file	.itest file path.
test_dir	Common parent path for .itest file. If the test and test case are in the same directory, perform an import.

Create CPI file

CPI file is a configuration file for creating a project in CLI environment. The project is created only when the [Project] – `create` option in the CLI.ini file is set to true.

You can set the option to create a project from the CPI template file in the path of `Controller Tester installation path\plugins\com.codescroll.gp.cli_version\cpi`.

> ! You cannot create a project in the CLI environment while using the workspace.

Item	Description
KIND	Project creation type setting. General project =1 Get Build Script = 2 Visual Studio Import =3 Import Embedded Project = 4
NAME	Project name.
LOCATION	Location of project creation.
DESCRIPTION	Project description.
LANGUAGE	Project type. Java = 0, C/C++ = 1

SourceFileProject

Item	Description

IUT_LANGUAGE	Target language type. C=0, C++=1, Java=2
DBSCHEMA	Global database schema version.
LINKFLAG	Link flag.
COMMON_COMPILEFLAG	Compilation flag to apply to all TUs.
TOOLCHAIN_NAME	Toolchain name.
BINARY_KIND	Binary type.
SOURCE_TOP_DIR	Source file top level directory.
SOURCE	Absolute path to source file.
COMPILEFLAG	Compilation flag to apply only to source files.
COMPILER_PATH	Path to the compiler to apply only to source files.

VisualStudioPorject

Item	Description
PROJECT_PATH	Visual Studio project file to import.
ENVFILE	Environment variable required for import.

BuildScriptProject

Item	Description
BUILD_COMMAND	Build script to perform the import.
WORKING_DIRECTORY	Working directory to perform import.
ENVFILE	Environment variable required for import.

EmbeddedProject

Item	Description
COMMON_COMPILEFLAG	Compilation flag to apply to all Source Files.
TOOLCHAIN_NAME	Toolchain name.
PROJECT_PATH	Visual Studio project file to import.

29.4. CLI Execution

Controller Tester can be done in CLI environment.

Performing Controller Tester with CLI

1. Execute the path.bat file created with [Generate command line interface INI file] to get the path information necessary for CLI execution.
2. Execute UtCLI.exe by passing the CLI.ini file edited by the user as an argument.
    - UtCLI.exe path: `Controller Tester installation path\plugins\com.codescroll.ut_ut_3.x.x\bin`

3. The Controller Tester function set by the user is performed.

```
[projectproject CLI start!!!]


[BEGIN : Project operation]
[END : Project operation]


[BEGIN : Analysis operation]
  [START : PA]
        Suresoft  (R) Program Analyzer (internal use only)
        Copyright (C) Suresoft Technologies Inc. All rights reserved.

        Build   : May  5 2021 20:10:42
        REV     : ENT 6.2 Windows(Release) {ce4b8c6 2021-05-04 15:42:28 +0900}
                  LIBANALYSIS 6.2 Windows(Release) {ba34ac82 2021-05-03 14:05:21 +0900}
                  LIBSST 6.2 Windows(Release) {ab41074 2021-04-28 14:53:32 +0900}


        VALUE1
        PHASE21002
        VALIF101
        VALUE100
log4j:WARN No appenders could be found for logger (com.codescroll.ut.module.cwg.generator.ClassFactoryManager).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Load JDBC Driver...OK
  [END : PA]
  [START : ckdb]
        VALUE1
        VALUE100
  [END : ckdb]
[END : Analysis operation]


[BEGIN : InitializeTest operation]
        VALUE1
        VALUE10
        VALUE30 : Delete Test From DB
        VALUE70 : Delete Test From Dir
        VALUE100
[END : InitializeTest operation]
```

30. EULA(End-User License Agreement)

Software End User License Agreement

Important: This End User License Agreement (“EULA”) is a legal agreement between the user (individual or organization) and Suresoft Technologies, Inc. (“SureSoft Tech”) which is the manufacturer the software products (“Software product” or “Software”) identified by the product identification card or recognition label attached to this EULA. The software product includes computer software, related media, related documents, and “online” or electronic manual. When installing, copying, downloading, backing up, accessing or using the software product, the user must accept the terms of use in the EULA; if the user does not accept the terms of use in EULA, the user may not be licensed for the software product. Unless the user is licensed to use the software product, the user cannot install, copy, download, backup, access or use this software product.

Software Product License

The Software product is protected by intellectual property rights as well as copyrights, program copyrights, and international copyright regulations. The term “computer” used in this agreement means a single computer system, the range of which is described below.

1. License Grant

The EULA grants the user the following rights.

Software Installation and Use. Except for those clearly explained in the EULA, only one copy of software product can be installed, used, accessed, executed or interacted (“execution”) on the computer. To install, use or execute the software product on two or more computers, the user must obtain the legal license(s) that corresponds to the number of computers regardless of the interaction of software used or executed on each computer.
Backup Copy. When there is no software product backup copy on the computer, the user can create a single backup copy that corresponds to the computer software part of the software product. The backup copy can be used for the purpose of recording. To create a single backup copy, a backup utility can be used. Except for those clearly stated in this EULA, the user may not make copies of the software product including documents provided with the software product.

2. Other Rights and Limits

Computer or Single Computer System. The Software product’s license is applied only to a single computer system. If the single computer system has multiple CPUs, it is considered as a single computer system only when these CPUs are installed to one circuit board and the structure is the symmetric central processing structure that shares one system bus.

Choose language versions. When the software product is included in one or more language versions, it is allowed to use only one language version.
Reverse engineering limitations. De-compilation and decomposition. The user cannot decompile or decompose software the product using reverse-engineering.

Component separation. The software product is allowed as a single product. Users cannot separate the components to use in one or more computers.
Lease. The Software product cannot be leased or lent.
Right Transfer of software product. The user may permanently transfer all rights as part of the sale of hardware or right transfer according to this EULA, it is regarded that the recipient has accepted the terms of use in this EULA.
Expiration. When the user violates the terms and conditions in the EULA, without infringing other rights, Suresoft Technologies can make user's rights expire according to the terms and conditions in this EULA. In such a case, the user must discard all the duplicates and composing parts, including software product that has been already installed.
Registered Trademark. The EULA doesn't grant any rights related to the registered trademark and service trademark of the software manufacturer, the computer manufacturer, and other related suppliers.
Upgrade. If the software product is upgraded via Suresoft Technologies, the terms of use in the EULA are still valid for the upgraded software and it cannot be separated to use in one or more computers.

3. Copyright

Every right and intellectual propriety right (including all the image, figure, animation, video, audio, music, text, and applet of the software product) of the software product, the enclosed manual, and every copy of the software product are the property of software manufacturer or supplier. All rights and intellectual property rights for the contents that are not built-in the software product but are accessible through the use of the software product are the property of the Company, and this EULA does not grant any rights for such use. When software product includes the manual in a file, only one copy of that file can be printed. The documents accompanied by the software product cannot be copied. All the rights that are not clearly stated are the property of the corresponding software manufacturer and supplier.

4. Double Media Software Product

The Software product can be delivered in one or more media. Regardless of the received media's type or size, only one media can be used in the computer. Other media cannot be executed on another computer. Except for the case of transferring rights permanently (as described above), the user cannot lease or lend the other media or transfer the right to other users.

5. Guarantee

The Software product's guarantee on quality follows the attached warranty. If there is no separate warranty enclosed, Suresoft Technologies guarantees that there was no defect during the production process if it has been normally used for one year from the day of purchase. This does not mean that the software product has no error, and does not exclude the possibility that unexpected result may appear when using the product. Except for the cases where it is clearly stated in the separate warranty agreement, Suresoft Technologies is not liable not only for the inaccuracy, error or defect of the software product but also for the loss and damage produced from these. Suresoft Technologies doesn't take any responsibility for the third party's claims for damage or the user's claims for damage to the third party. The Limited warranty statement, mentioned in the EULA, is still valid even though the user notified Suresoft Technologies or its official representative in advance of the possibility mentioned above.

6. Export limitation

The right transfer of the software product can be carried out only within Korea, and it cannot be exported to other countries.

7. JAVA Support

The Software product includes the support of programs written in JAVA. JAVA technology is not designed as an online control device, or not made, used or sold in dangerous environments that need to error preventive function, such as nuclear facility, aircraft navigation, aircraft control, medicine, medical system, and weapon system that can cause death, injury, or critical physical/environmental damage.

Caution

© Suresoft Technologies, Inc. All copyrights reserved. Printed in Korea
Suresoft Technologies and CODESCROLL are trademarks of Suresoft Technologies, Inc.

The product name or trade name mentioned in this document and related document are the trademarks or registered trademarks of the Company having the copyright for the product. The software described in this document is provided according to the usage agreement and non-disclosure agreement.

The software may only be used or copied according to the terms of use in this Agreement.