

# CodeScroll Controller Tester

3.2 — Last update: May 21, 2020

Suresofttech



# Table of Contents

<b>1. Before starting.....</b>	<b>3</b>
<b>2. Overview .....</b>	<b>4</b>
<b>3. Install CONTROLLER TESTER .....</b>	<b>6</b>
<b>4. Uninstall CONTROLLER TESTER .....</b>	<b>9</b>
<b>5. Run CONTROLLER TESTER .....</b>	<b>12</b>
<b>6. Set Controller Tester License .....</b>	<b>14</b>
<b>7. Set a Toolchain (analyzer) .....</b>	<b>16</b>
7.1. Auto-registration of toolchain.....	18
7.2. Add a Toolchain .....	19
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
<b>8. Create a Controller Tester Project.....</b>	<b>27</b>
8.1. C/C++ Project with Source Files.....	29
8.2. C/C++ Project from Visual Studio Project .....	32
8.3. C/C++ Project from Embedded.....	34
8.4. C/C++ Project form Existing CodeScroll Project .....	35
8.5. Create a C/C++ project with CPI File.....	36
8.6. Creates a C/C++ project with Build Information .....	37
<b>9. Create a test.....</b>	<b>39</b>
<b>10. Test Editor .....</b>	<b>42</b>
10.1. Test Info.....	43
10.2. Test Case.....	45
10.3. Test Code .....	47
10.4. Configuration.....	48
10.5. Test Macro .....	49
<b>11. Test Run.....</b>	<b>51</b>
<b>12. Properties .....</b>	<b>53</b>
12.1. Project Properties.....	54
12.2. Module Properties .....	63
12.3. Source File Properties.....	65



<b>13. Preferences</b>	<b>67</b>
13.1. Analysis	68
13.2. Exclusion	74
13.3. Performance	75
13.4. Source File Types	77
13.5. Language	78
13.6. Theme	80
13.7. Test	81
13.8. Toochain	87
13.9. Persperctive	88
13.10. Editor	89
<b>14. Test Perspective</b>	<b>90</b>
14.1. Dashboard	92
14.2. Test Navigator View	96
14.2.1. Merge Project Coverages	98
14.3. Source Code Editor section	101
14.4. Unit Test View	102
14.5. Integration Test View	120
14.6. Coverage View	127
14.7. MC/DC View	130
14.8. Stub View	132
14.9. Class Factory View	143
14.10. Control Flow Graph View	144
14.11. Call Graph View	147
14.12. Function Call Hierarchy View	151
14.13. Error View	153
14.14. Fault Injection View	155
14.15. Input/output Data Graph View	158
14.16. Console View	160
<b>15. Analysis Perspective</b>	<b>162</b>
15.1. Metrics View	163
15.2. Metrics Chart View	166
15.3. Metrics Top Chart View	168
15.4. Metrics Bar Chart View	169
15.5. Metrics Diagnosis Chart View	171
15.6. Unused Function View	173
15.7. Source-Header Relation View	174
15.8. Global Variable Relation View	176
<b>16. File Menu</b>	<b>177</b>
16.1. Create a module	178



16.2. Switch Workspace .....	179
16.3. Export .....	180
16.4. Import.....	193
<b>17. Edit Menu .....</b>	<b>208</b>
<b>18. Search Menu .....</b>	<b>209</b>
<b>19. Project Menu .....</b>	<b>211</b>
<b>20. Window Menu .....</b>	<b>213</b>
<b>21. Help Menu .....</b>	<b>215</b>
<b>22. Troubleshooting .....</b>	<b>219</b>
<b>23. CONTROLLER TESTER Target Plug-in.....</b>	<b>222</b>
23.1. Target Environment Settings .....	223
23.2. C/C++ Target Test Project.....	228
23.3. Target Test Run Settings.....	234
23.4. Target Log Collector .....	235
23.5. Target Test Results .....	237
23.6. Preferences.....	238
<b>24. CODESCROLL RTV(Remote Target Verifier) .....</b>	<b>240</b>
24.1. RTV Server Settings.....	241
24.2. RTV Toolchain Settings.....	245
24.3. Create a RTV Project .....	249
24.4. Refresh/Add RTV Source Files.....	255
24.5. RTV Project Information .....	259
24.6. Run RTV Test .....	261
24.7. RTV Test Results .....	263
<b>25. EULA(End-User License Agreement) .....</b>	<b>264</b>



# 1. Before starting

---

## Purpose of Document

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



## 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, refer to “Controller\_Tester\_Supported\_Toolchains” document.

### 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



## 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)

## Overall screen of Controller Tester

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

**Source Code Editor**

```

885  /* Flush the bits in the bit buffer to pending output (leaves at
886  */
887  void ZLIB_INTERNAL _tr_flush_bits(s)
888      deflate_state *s;
889  {
890      bi_flush(s);
891  }
892
893  /*
894  * Send one empty static block to give enough lookahead for inflate.
895  * This takes 10 bits, of which 7 may remain in the bit buffer.
896  */
897  void ZLIB_INTERNAL _tr_align(s)
898      deflate_state *s;
899  {
900      send_bits(s, STATIC_TREES<<1, 3);
901      send_code(s, END_BLOCK, static_ltree);
902      #ifdef DEBUG
903      s->compressed_len += 10L; /* 3 for block type, 7 for EOB */
904      #endif
905      #endif
906      s->compressed_len += 10L; /* 3 for block type, 7 for EOB */
907
908  /*
909  * Determine the number of bytes in the current block: dynamic trees
910  * trees or store, and output the encoded block to the zip file.
911  */
912  void ZLIB_INTERNAL _tr_flush_block(s, buf, stored_len, last)
913      deflate_state *s;
914      charf *buf; /* input block, or NULL if too old */
915      ulg stored_len; /* length of input block */
916      int last; /* one if this is the last block for a file */
917  {
918      ulg opt_lenb, static_lenb; /* opt len and static len in bytes */
919      int max_bindex = 0; /* index of last bit length code of new
920
921  /* Build the Huffman trees unless a stored block is forced */
922  if (s->level > 0) {
923      /* Check if the file is binary or text */
924      if (s->strm->data_type == Z_UNKNOWN)
925          s->strm->data_type = detect_data_type(s);
926      }
927  }
928
929

```

**Dashboard section**

Unit Test Integration Test

Run

(809 / 0 / 286) 1095 **39.1%**  
(1751/4467)

Statement Coverage

Name	Result	Coverage
> _tr_align(struct internal_state *)	(1 / 0 / 7) 8	71.4% (15/21)
> _tr_flush_bits(struct internal_state *)	(0 / 0 / 0) 0	100.0% (1/1)
> _tr_flush_block(struct internal_state *, char *, ulg, int)	(0 / 0 / 12) 1	47.6% (20/42)
> _tr_init(struct internal_state *)	(1 / 0 / 0) 1	100.0% (10/10)
> _tr_stored_block(struct internal_state *, char *, ulg, int)	(0 / 0 / 8) 8	100.0% (11/11)
> _tr_tally(struct internal_state *, unsigned int, unsigned int)	(0 / 0 / 3) 3	22.2% (2/9)
> [H/T] adler32(unsigned long, const unsigned char *, ulg)	(9 / 0 / 4) 13	98.0% (101/103)
> adler32_combine(unsigned long, unsigned long, ulg, ulg)	(5 / 0 / 0) 5	100.0% (1/1)
> adler32_combine64(unsigned long, unsigned long, ulg, ulg)	(5 / 0 / 0) 5	100.0% (1/1)
> adler32_combine_64(unsigned long, unsigned long, ulg, ulg)	(11 / 0 / 0) 11	100.0% (21/21)
> bi_flush(struct internal_state *)	(5 / 0 / 6) 11	100.0% (9/9)

**Test view section**

Test Info (zlib/\_tr\_flush\_bits\_test0)

Test Structure

Test structure using a tree view and edit the information in the test.

Name	In	Out
Test global code		
User code		
Global Variable		
Test target function		
User code		
Stub		

**Test Editor section**

Test Info Test Case Test Code Configuration

108M of 248M



## 3. Install CONTROLLER TESTER

### Installation requirements

<b>OS</b>	Microsoft Windows 7/10 (32bit/64bit)
<b>RAM</b>	512MB or more
<b>HDD</b>	Free space about 500MB (GCC Compiler not installed) or 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 setup.exe file.

Name	Date modified	Type	Size
CodeScroll Controller Tester_3.0.msi	4/26/2018 1:16 AM	Windows Installer ...	648,994 KB
setup.exe	4/26/2018 1:16 AM	Application	1,349 KB



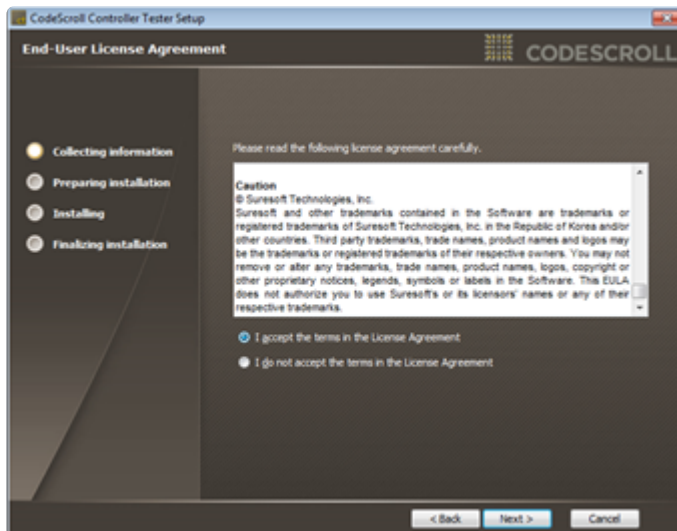
It does not work properly when installed with 'CodeScroll Controller Tester\_3.x.msi'.

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

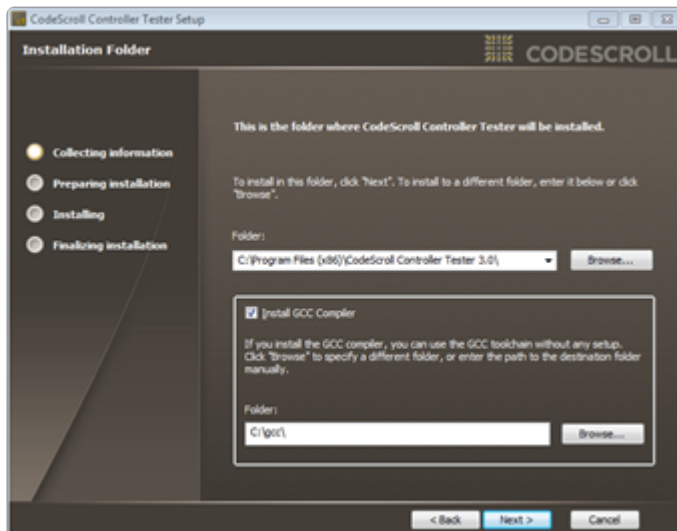


3. Accept the end-user license and click [Next].





4. Set the path to install Controller Tester and whether or not to install the GCC compiler built in Controller Tester, and click [Next].

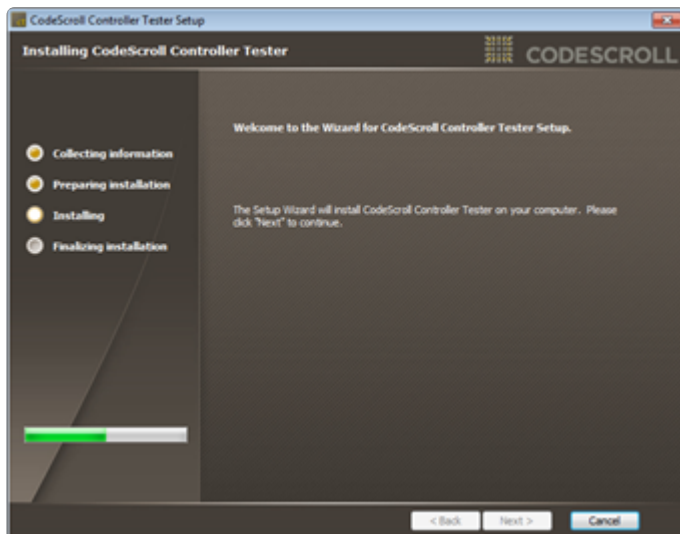


5. All required information for the installation has been collected. Click [Install].

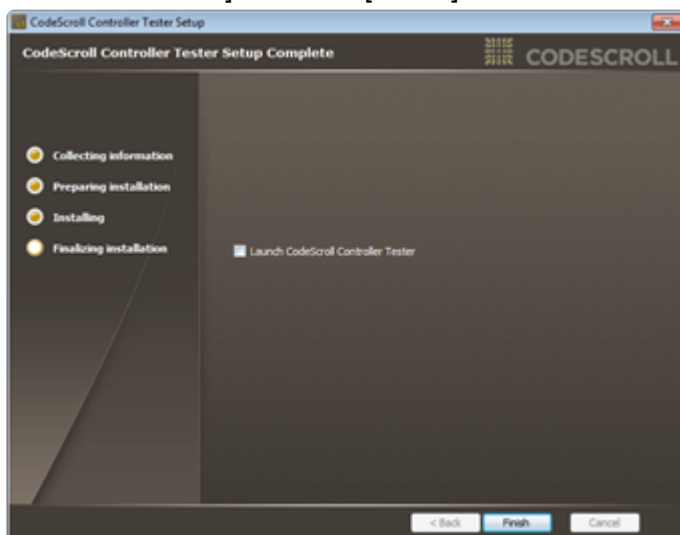




## 6. Install Controller Tester.



## 7. When running Controller Tester immediately after installation is complete, check [Start {CodeScroll Controller Tester}] and click [Finish].



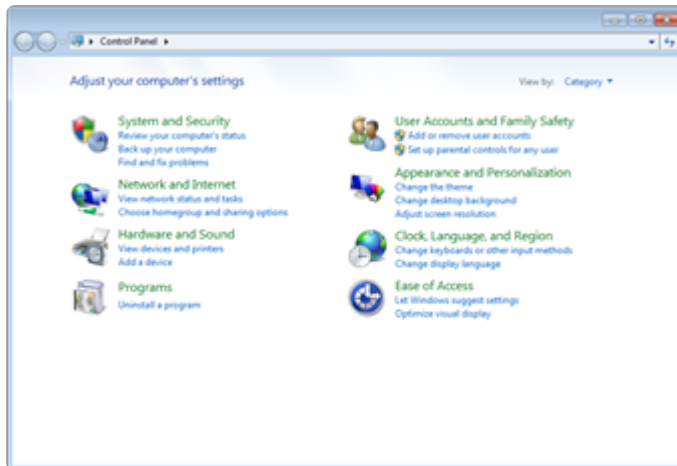


## 4. Uninstall CONTROLLER TESTER

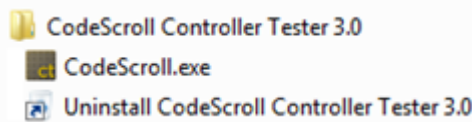
### Uninstall Controller Tester for Windows

There are three ways to uninstall Controller Tester.

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



2. Uninstall using [Start] -> [All programs] -> [CodeScroll Controller Tester 3.x] -> [Uninstall CodeScroll Controller Tester 3.x]



3. Uninstall using [Path to install] -> [Uninstall CodeScroll Controller Tester 3.x]

configuration	6/3/2018 5:15 PM	File folder	
features	6/3/2018 5:09 PM	File folder	
jre	6/3/2018 5:09 PM	File folder	
p2	6/3/2018 5:09 PM	File folder	
plugins	6/3/2018 5:09 PM	File folder	
artifacts.xml	4/26/2018 4:57 PM	XML Document	119 KB
CodeScroll.exe	4/26/2018 4:57 PM	Application	312 KB
CodeScroll.exe.manifest	10/16/2017 1:43 PM	MANIFEST File	2 KB
CodeScroll.ini	4/26/2018 4:57 PM	Configuration sett...	1 KB
eclipse.exe	4/26/2018 4:56 PM	Application	24 KB
epl-v10.html	10/16/2017 2:05 PM	HTML Document	17 KB
notice.html	10/16/2017 2:05 PM	HTML Document	9 KB
Uninstall CodeScroll Controller Tester 3.0	6/3/2018 5:09 PM	Shortcut	2 KB



“Uninstall Controller Tester” deletes the files necessary to operate the Controller Tester. The workspaces set by users are not deleted.

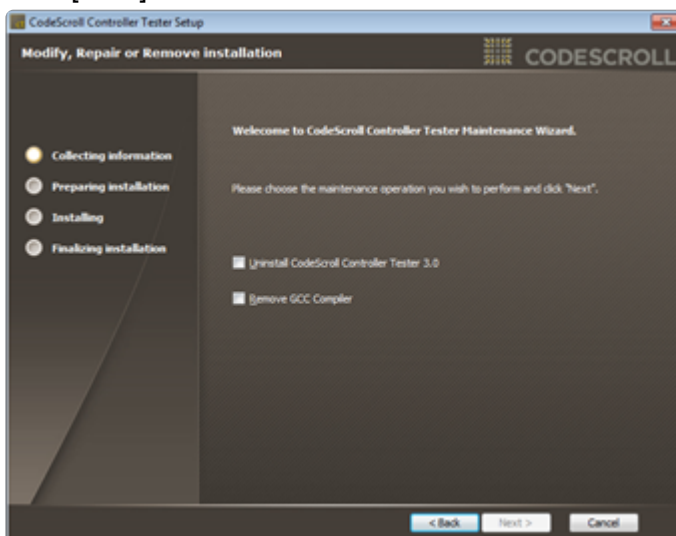


## The procedures for [Uninstall CodeScroll Controller Tester 3.x] are as follows:

1. Run [Uninstall CodeScroll Controller Tester 3.x] to collect the information needed to uninstall the program. Click [Next] to proceed.



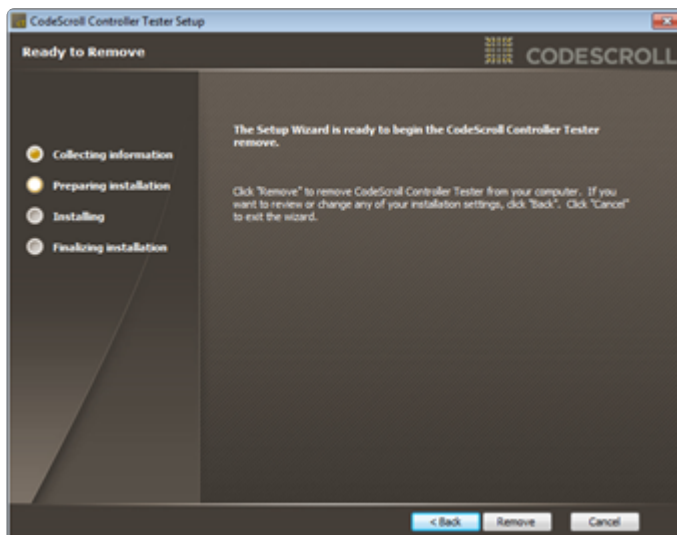
2. Select the checkbox [Uninstall CodeScroll Controller Tester 3.x] and [Uninstall GCC Compiler] and click [Next].



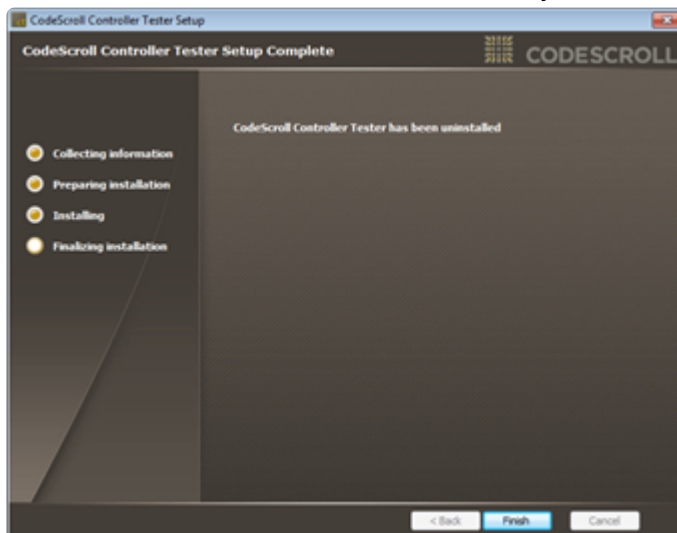
\* [Remove GCC Compiler] can be checked only when [Remove CodeScroll Controller Tester 3.x] is checked.  
If the GCC compiler has not deleted, the GCC compiler cannot be installed when re-installing the Controller Tester.

3. All the necessary information for uninstalling has been collected. Click [Uninstall].





4. When the Controller Tester is successfully uninstalled, click [Finish].

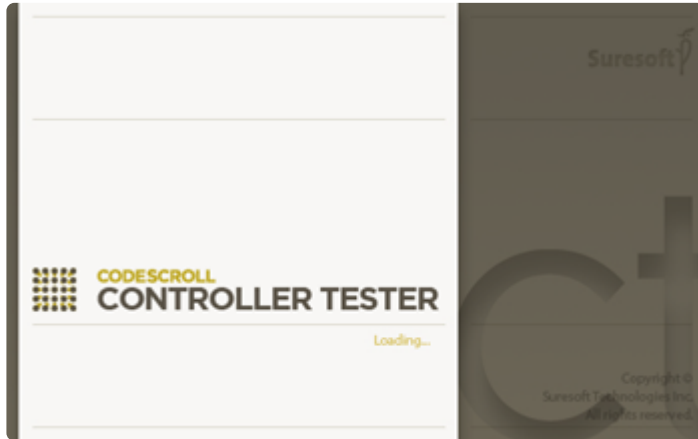




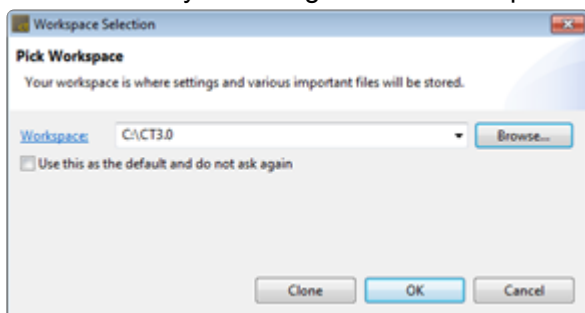
## 5. Run CONTROLLER TESTER

---

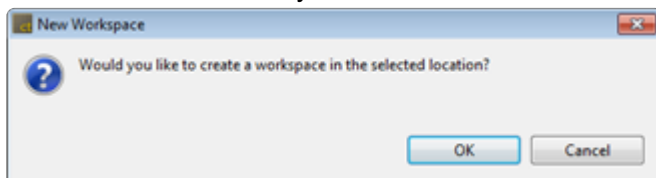
Select the shortcut icon or [Start] -> [All programs] -> [CodeScroll Controller Tester 3.x] -> [CodeScroll Controller Tester 3.x].



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.

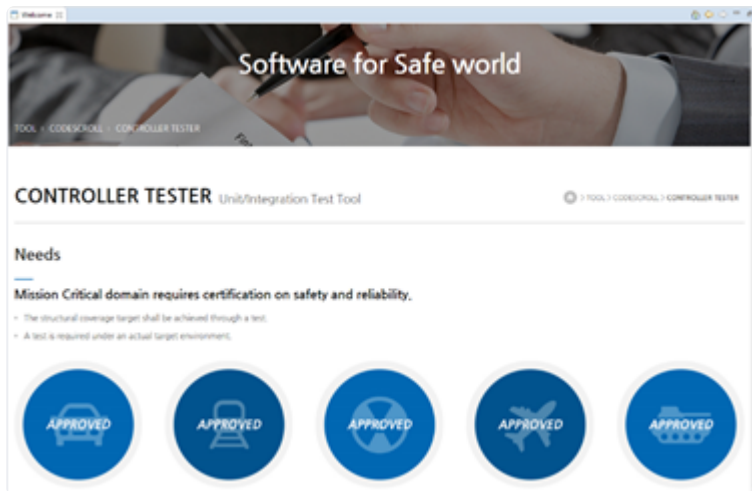


If the selected directory does not exist, confirm the user whether to create the directory.



Create a directory in the workspace you have set and the welcome page of the tool 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.

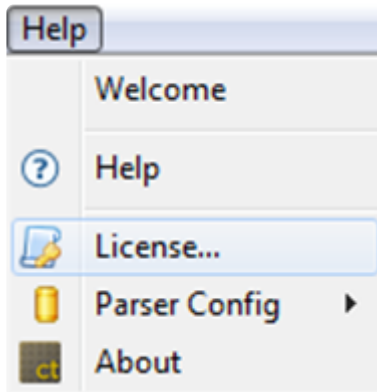


## 6. Set Controller Tester License

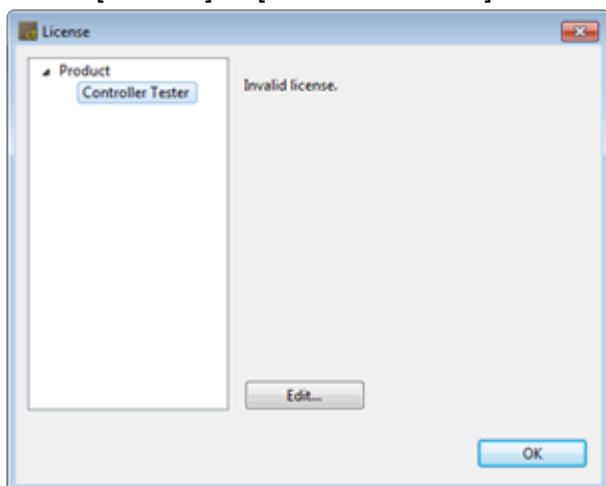
---

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.

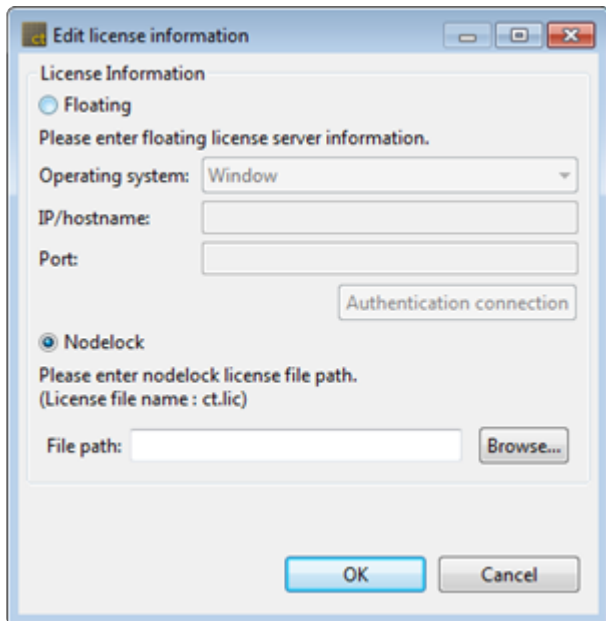


2. Select [Product] -> [Controller Tester] and click [Edit].



3. Enter the license in the manner provided 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 Test] button.
- Node-lock: Register the license file provided.

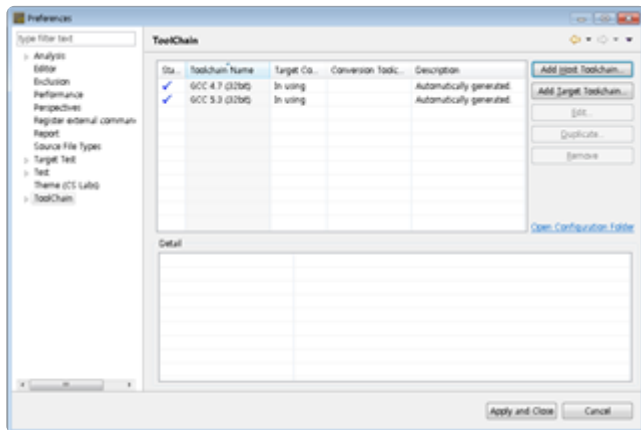


## 7. Set a Toolchain (analyzer)

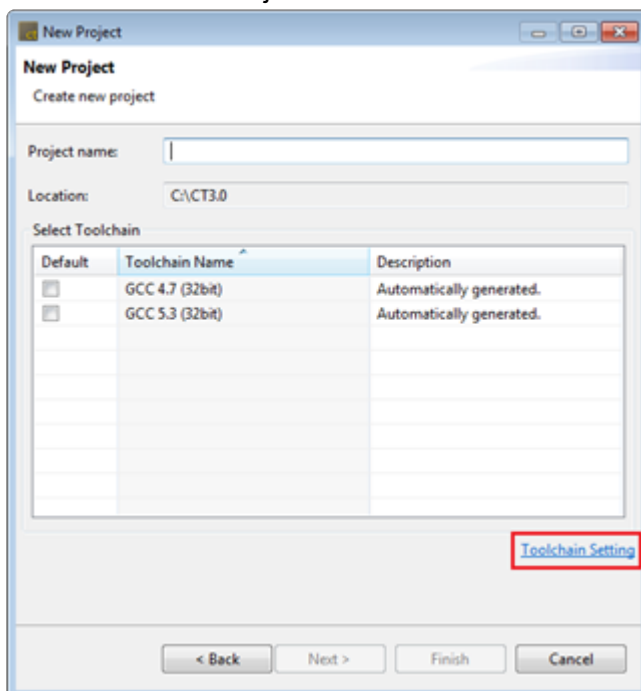
To perform the test using Controller Tester, the toolchain setting is required.

You must have a toolchain (compiler information) for the source under test to create a project.

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.

- [Add a Toolchain](#)
- [Edit a toolchain](#)
- [Duplicate a toolchain](#)

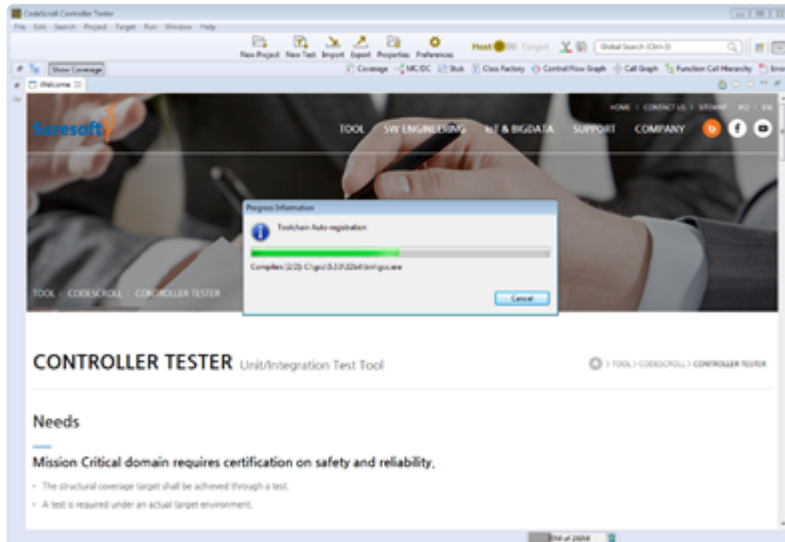


- [Remove a toolchain](#)
- [Export a toolchain](#)
- [Import a Toolchain](#)

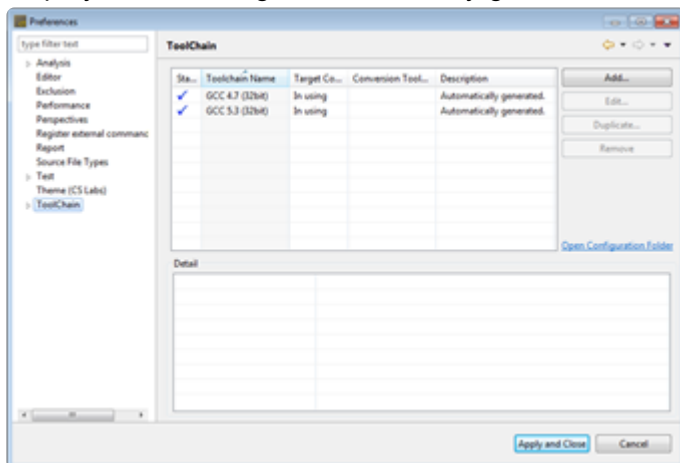


## 7.1. Auto-registration of toolchain

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.).



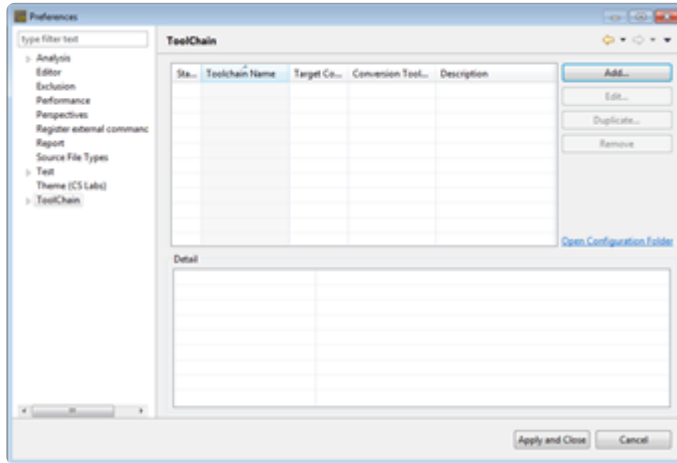
The auto-registered toolchain can be checked in [Window] -> [Preferences] -> [Toolchain] window, and it displays the message “Automatically generated.” in the description.



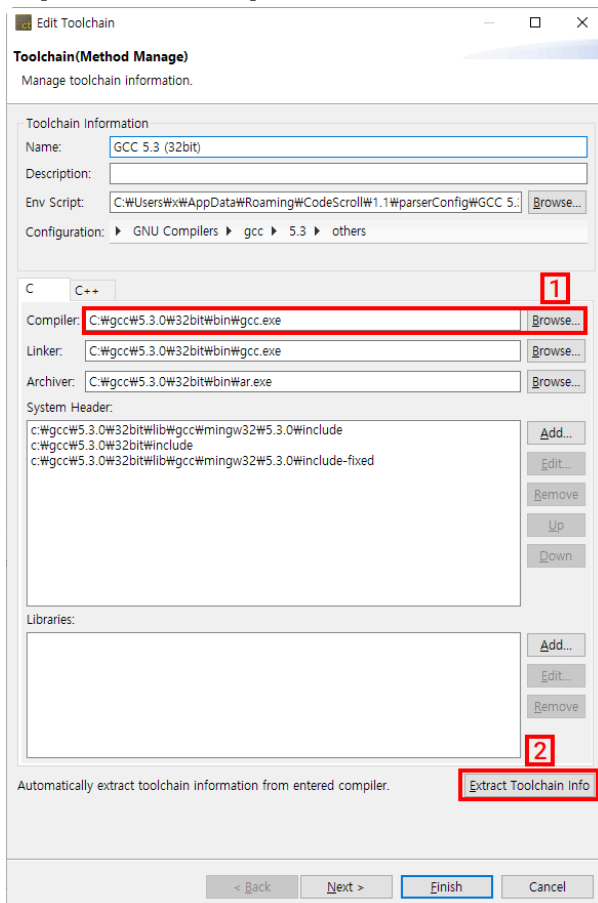


## 7.2. Add a Toolchain

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



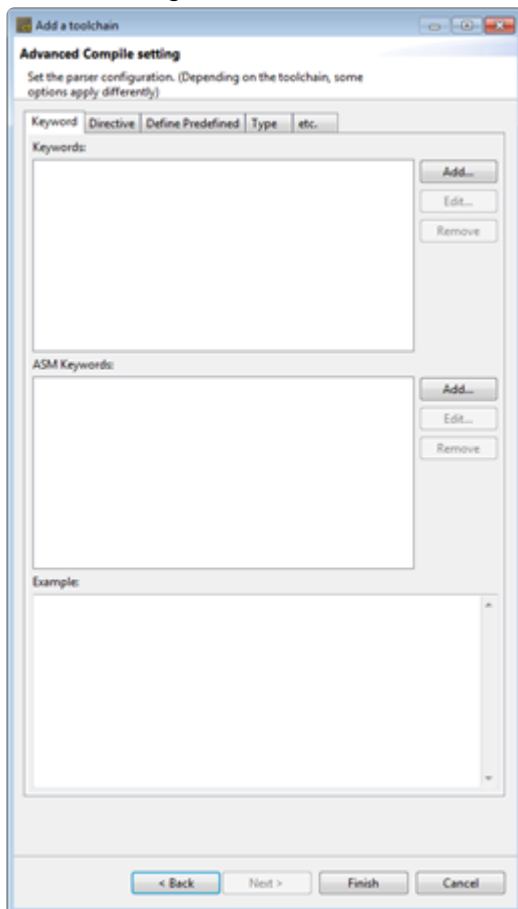
2. In [Add a toolchain] window, enter the toolchain information.



- When you click [Extract toolchain Info] <sup>2</sup> button after [Compiler] <sup>1</sup>, the toolchain information is extracted automatically from the compiler entered.
- In [Compiler] <sup>1</sup>, 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 Freescale 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], [Target Kind] 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 advance via [Example].

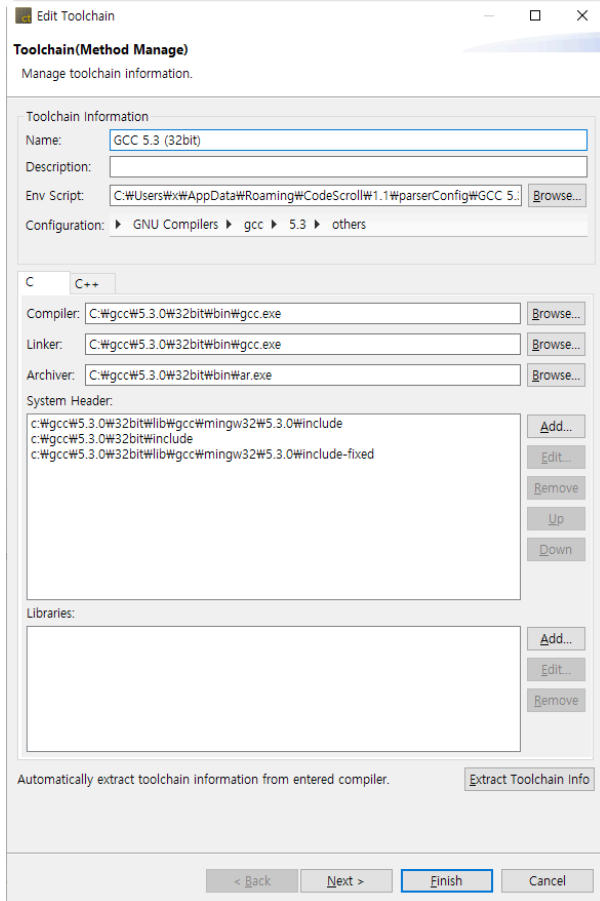


For the detailed setting procedure for the other toolchain detail, please contact technical support on [Troubleshooting](#).



## 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.



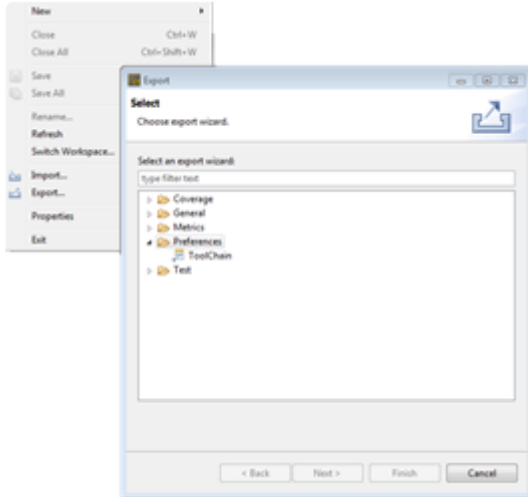
If there is a project using the toolchain you want to remove, the project will not work properly.



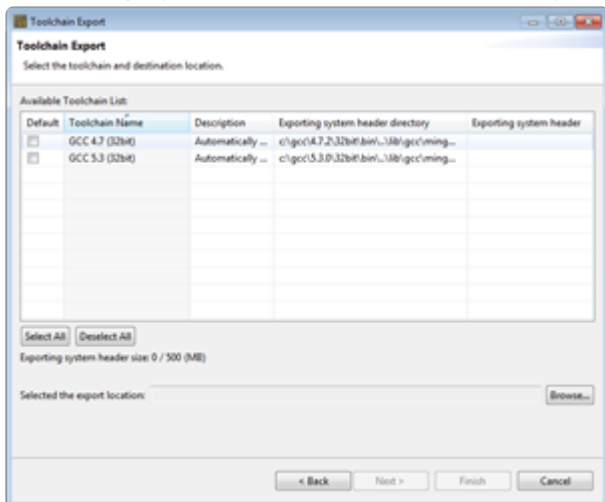
## 7.6. Export a toolchain

The toolchain information can be exported via 'Export' function.

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

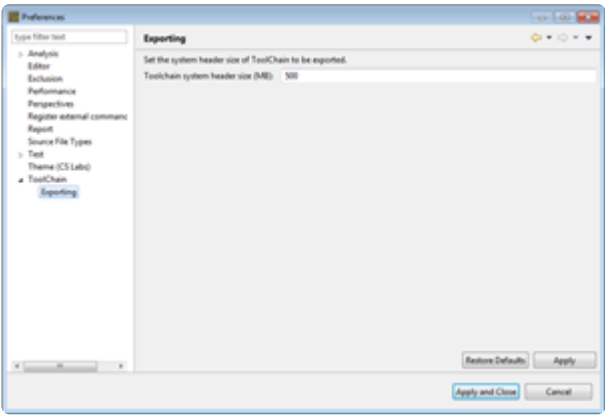


2. Select the toolchain to be exported and click [Browse] button to specify the path to be exported. If you set the system header file in the toolchain settings, a checkbox will be displayed in the [Exporting system header] column, where you can choose whether to export.



3. Click [Window] -> [Preferences] -> [Toolchain] -> [Exporting] in menu to set the system header size to be exported. The system header larger than the size you had set cannot be exported.



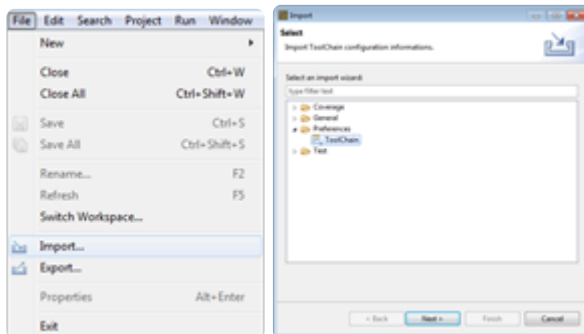




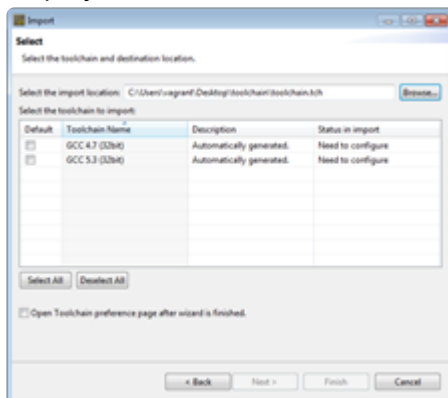
## 7.7. Import a Toolchain

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 fo the file is displayed.



If the name of the toolchain to import is duplicated with the existing toolchain name, the status of export becomes "Need to configure". The name of the toolchain to import can be modified in [Select how to resolve] window shown when clicking the [Next] button.



## 8. Create a Controller Tester 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.

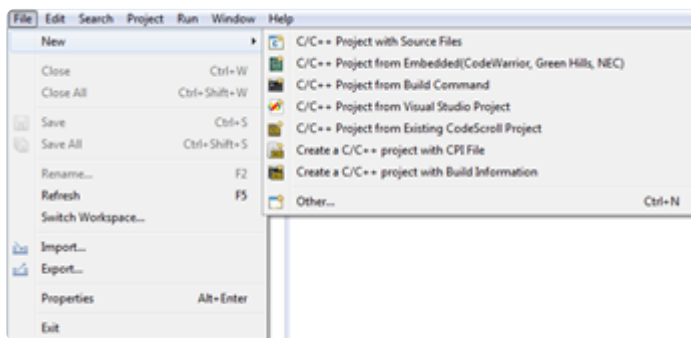
Before creating and testing a project, you need to set up a 'toolchain' that takes into account the development environment of the target software. Please refer to [Set a toolchain](#) for details.

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.

### 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 Visual Studio Project](#)  
Creates a project by using Visual Studio dsw, sln, vcxproj and vcproj files.
- [C/C++ Project from Embedded](#)  
Creates a project by using an embedded project file.
- [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.

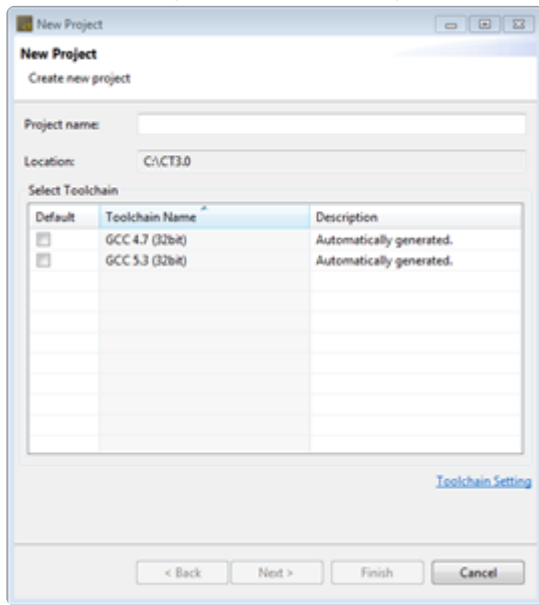


## 8.1. C/C++ Project with Source Files

---

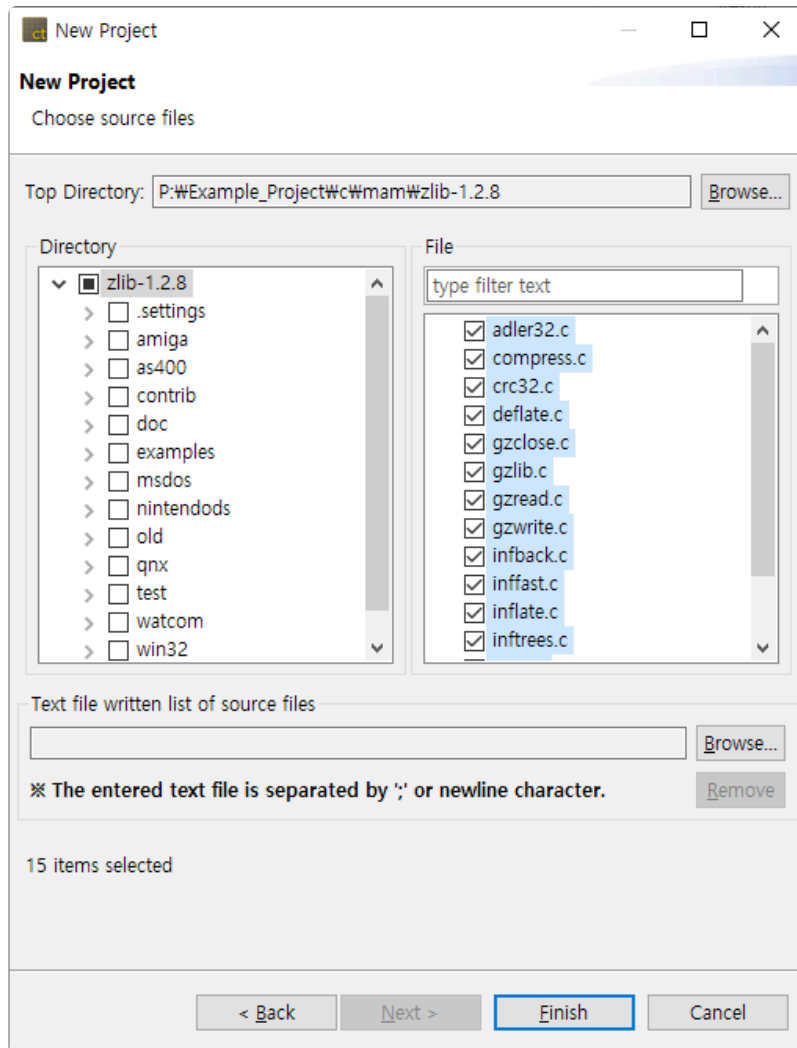
Create a project with C/C ++ source code files. The user needs to enter the necessary information to build the source code.

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.

- 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.
- On the [Directory] screen shown on the left, select the directory containing the source files to be used for project creation.
- 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.

- 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).
- Clicking the [Remove] button deselects the selected text file and the source file selected through the file.



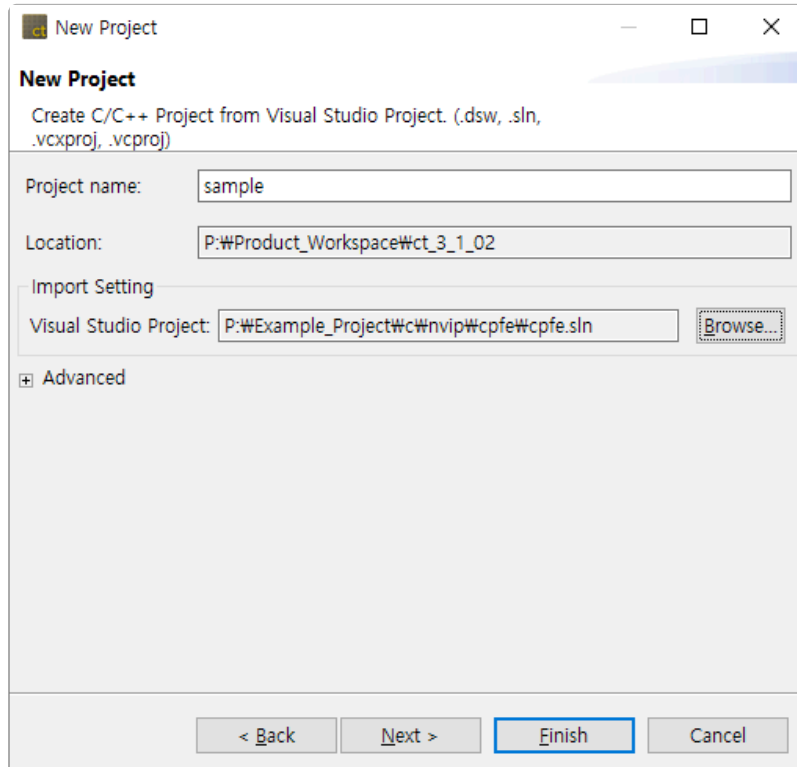
6. After all settings are completed, click the [Finish] button to create the project.



## 8.2. C/C++ Project from Visual Studio Project

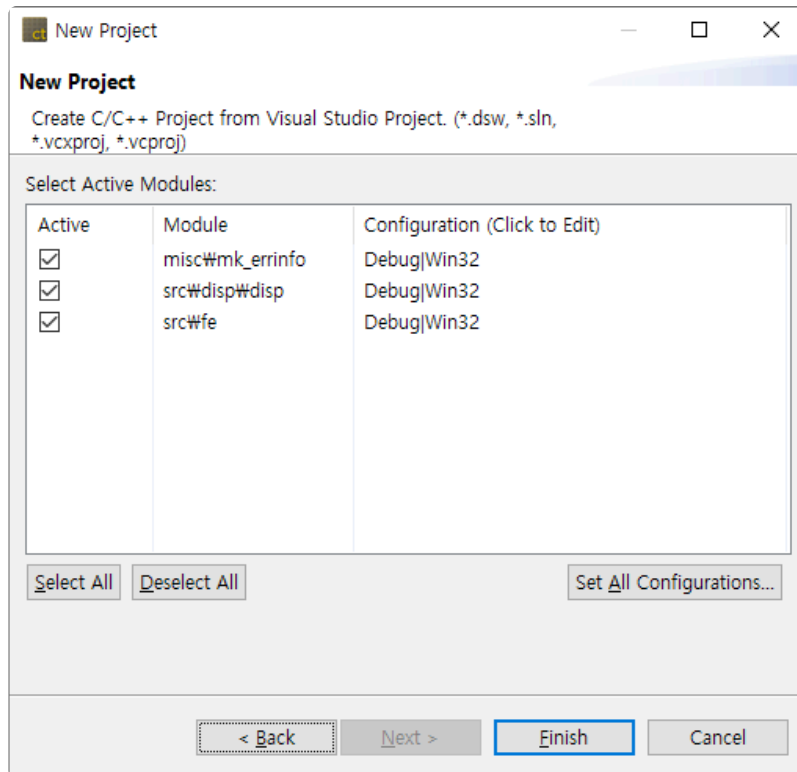
---

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.



## 8.3. 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.

**New Project**

Create new project from CodeWarrior project file(\*.xml), Green Hills MULTI project file(\*.gpj)

Project name:

Location:

Select Toolchain

Default	Toolchain Name ^	Description
<input type="checkbox"/>	CPP_TI_TMS320_6000_SoC_Unit	
<input type="checkbox"/>	GCC 4.7 (32bit)	
<input type="checkbox"/>	GCC 5.3 (32bit)	
<input type="checkbox"/>	Microsoft Visual Studio 2010 (32...	
<input type="checkbox"/>	Microsoft Visual Studio 2010 (32...	
<input type="checkbox"/>	Microsoft Visual Studio 2015 (32...	
<input type="checkbox"/>	Microsoft Visual Studio 2015 (32...	
<input type="checkbox"/>	gcc5	
<input checked="" type="checkbox"/>	iar	

[Toolchain Setting](#)

Import Setting

Embedded Project File:  [Browse...](#)

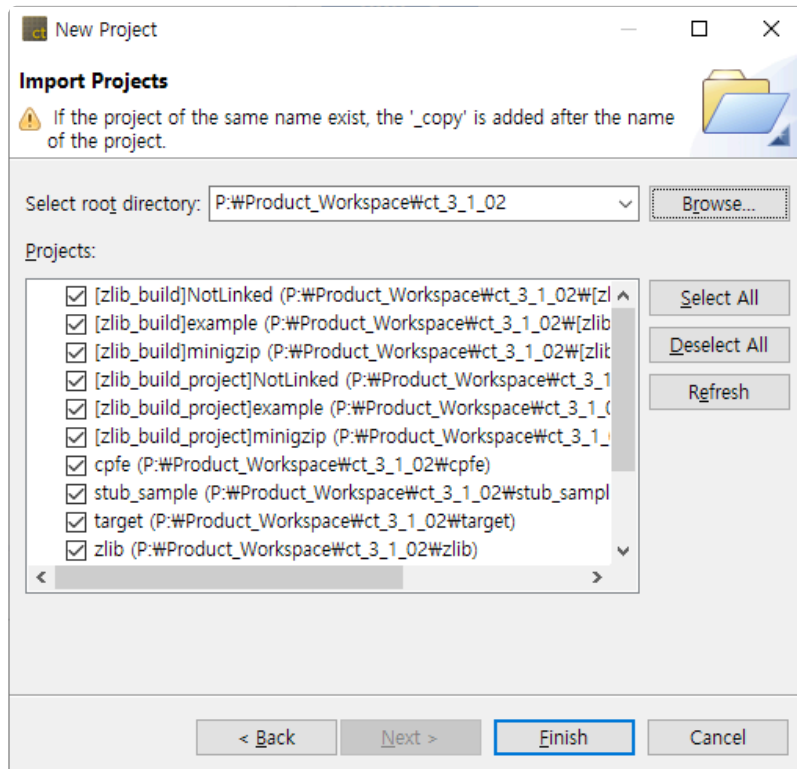
< Back   Next >   Finish   Cancel

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.



## 8.4. 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.



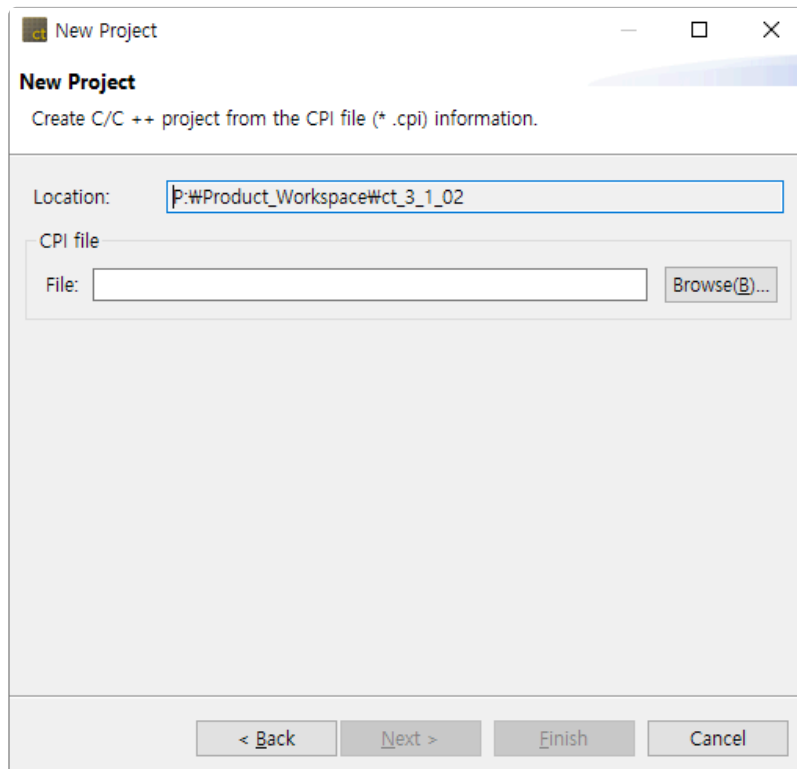
1. Use the [Browse ...] button to select the CodeScroll workspace or CodeScroll project path.
2. Projects existing in the selected path are displayed in the [Projects] list.
  - When the CodeScroll workspace is selected, all the projects under it are displayed.
3. Click [Finish] to create the project.



## 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. Creates 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

**New Project**  
Create C/C++ Project from Build Command. (e.g. makefile or build script)

Project name:

Location:

Select Toolchain

Default	Toolchain Name	Description
<input type="checkbox"/>	CPP_TI_TMS320_6000_SoC_Unit	
<input type="checkbox"/>	GCC 4.7 (32bit)	
<input checked="" type="checkbox"/>	GCC 5.3 (32bit)	
<input type="checkbox"/>	Microsoft Visual Studio 2010 (32bit)	
<input type="checkbox"/>	Microsoft Visual Studio 2010 (32bit_64bit)	
<input type="checkbox"/>	Microsoft Visual Studio 2015 (32bit)	
<input type="checkbox"/>	Microsoft Visual Studio 2015 (32bit_64bit)	
<input type="checkbox"/>	gcc5	
<input type="checkbox"/>	iar	

[Toolchain Setting](#)

Import Setting

Build Command:

Build Directory:  [Browse...](#)

**Note:** The final project name is determined by combining the entered project name with the module name extracted from the build information. (e.g. [project name] extracted module name)

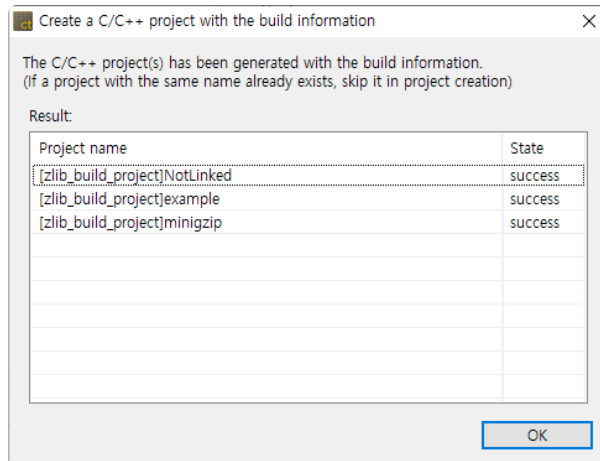
☐ Advanced

< Back   Next >   **Finish**   Cancel

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

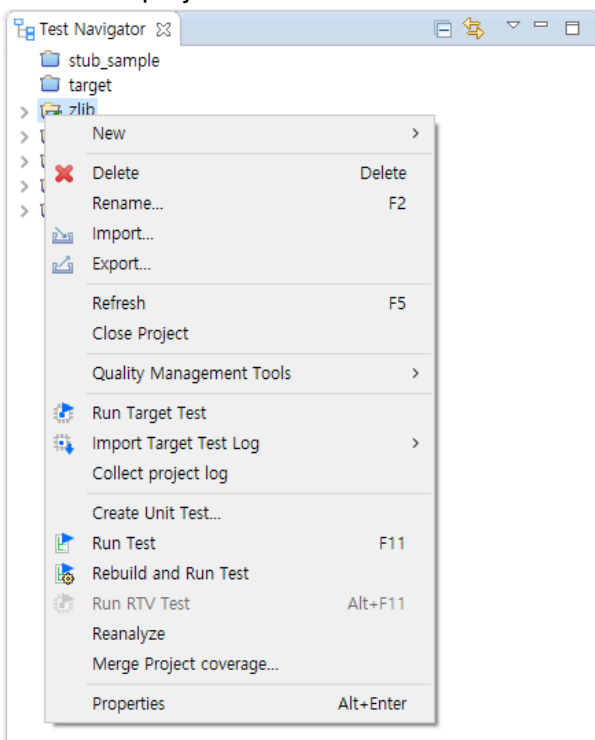
---

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

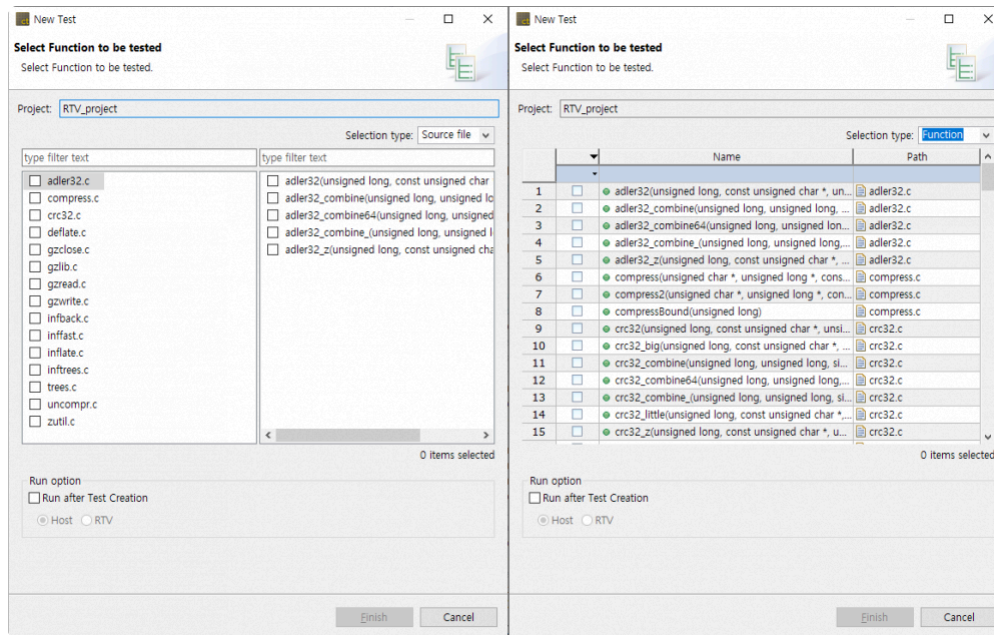
Generate test data and test code for each function under test.

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.



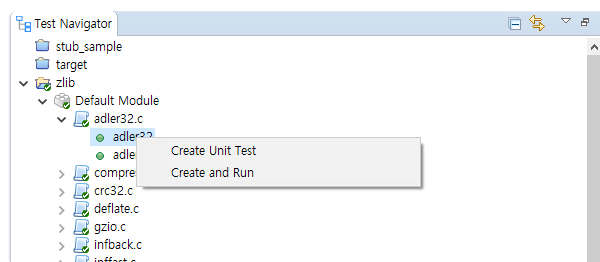


- a. You can filter the function under test using [type filter text].
  - b. You can use the [Selection type] combo box to select the function to be tested based on the source file or function.
  - c. You can easily select and deselect all functions under test with the right-click menu.
  - d. When [Run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

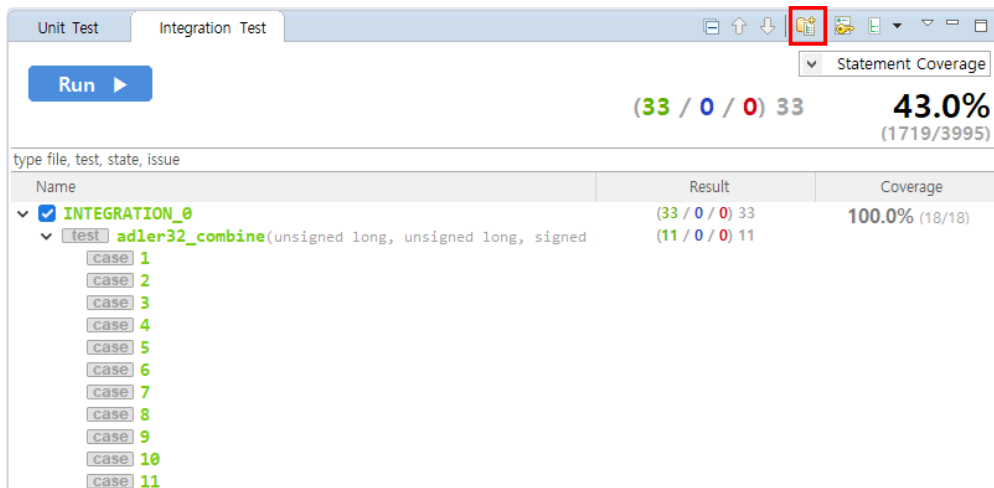


2. If you select the [Create and Run] menu, it creates a test for the target function and then runs the test.

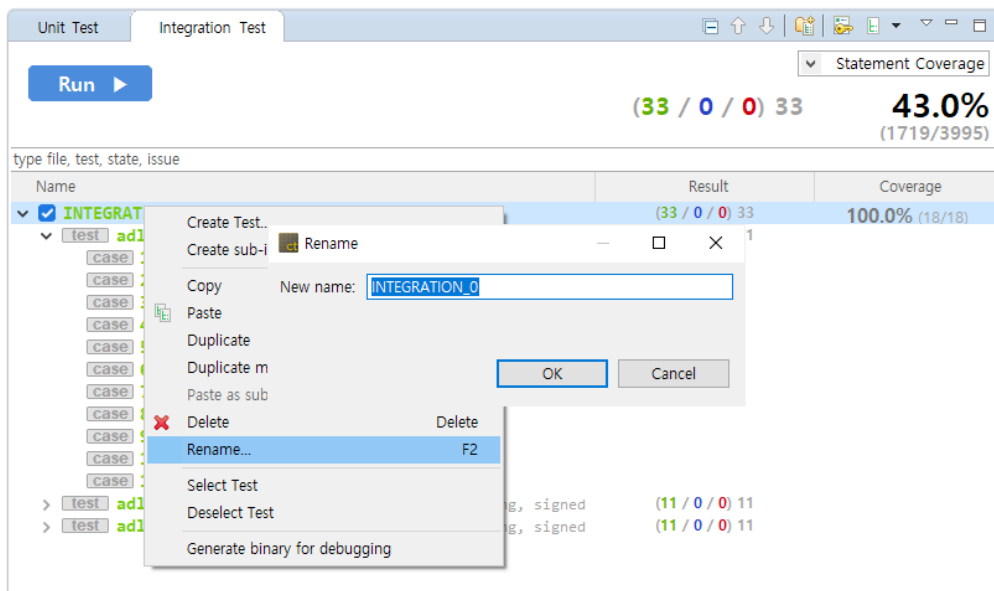
## Create an integration test

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





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





# 10. Test Editor

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

The screenshot shows the CodeScroll Controller Tester interface. The top panel displays test results for 'adler32' and 'adler32\_combine' with a statement coverage of 2.9%. The bottom panel, outlined in red, shows the 'Test Info' window for 'adler32' with a tree view of test structure and configuration options.

**Test Info (zlib/adler32\_test0)**

**Test Structure**

Test structure using a tree view and edit the information in the test.

Name	In	Out
Test global code		
User code		
Global Variable		
Test target function		
adler32(unsigned long, const unsigned long, const unsigned char *, unsigned long)		
Local Static Variable		
Parameter/return		
adler : uLong		
adler : unsigned long	<input checked="" type="checkbox"/>	<input type="checkbox"/>
buf : const Bytef *		
[0] : Bytef		
[0] : Byte		
[0] : unsigned char	<input checked="" type="checkbox"/>	<input type="checkbox"/>

**Test Info Edit**

Change the notation edit the partition

Set Numeral System

☒ 10 Numeral System ☐ 16 Numeral System

Variable Partition

Min ~ Max Add

Single Value Add

Partition List

"0" Delete

"1" Default

"2~15"

"16"

"17~5551"

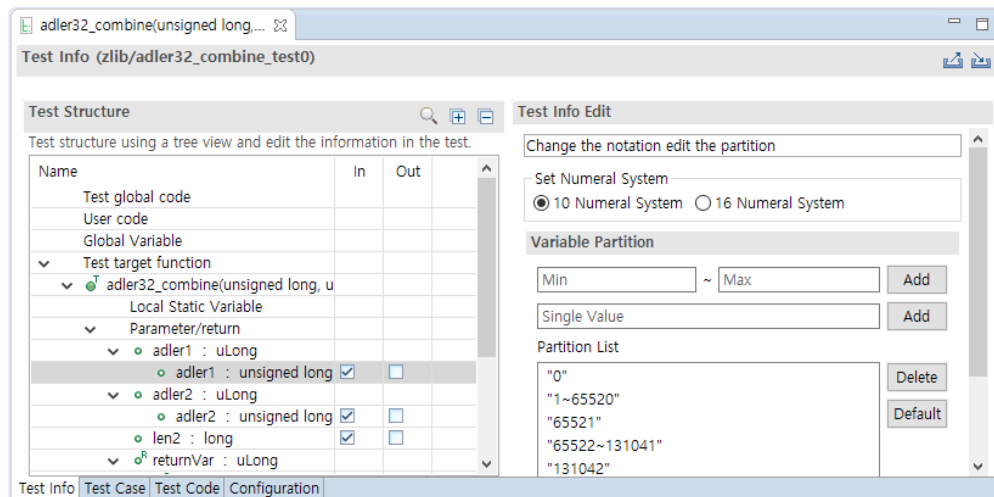


You can move the editor and open it in a separate window.



## 10.1. Test Info

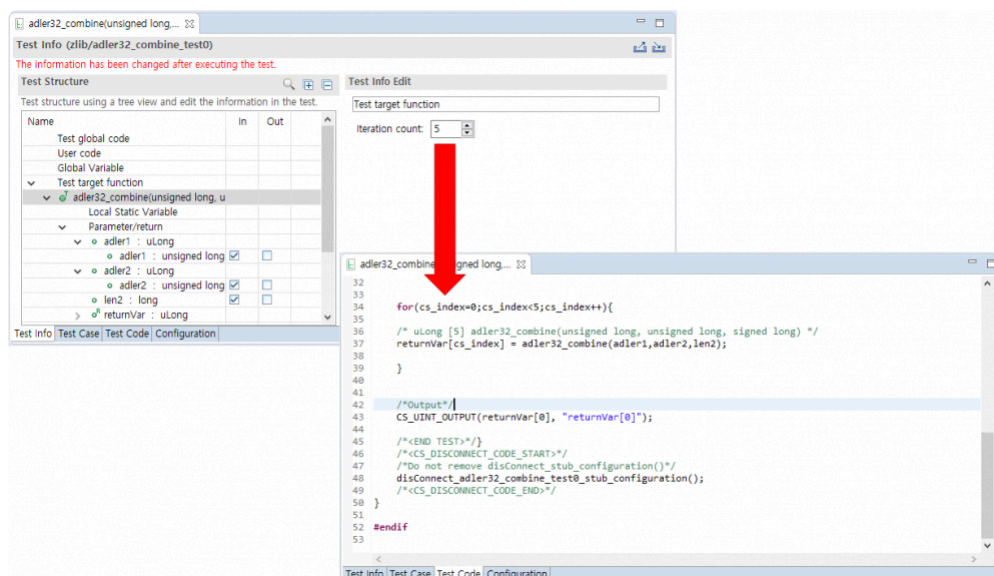
The Test Information tab is a feature that presents the test code as a tree of test structures, making it easy for users to modify the test code.



- 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.

## Repeat function calls

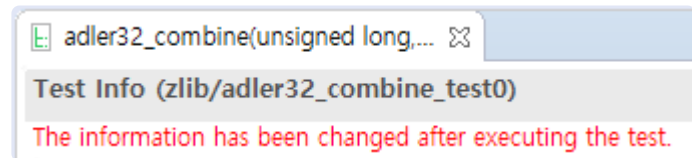
You can have the function under test be called multiple times.





## Change detection

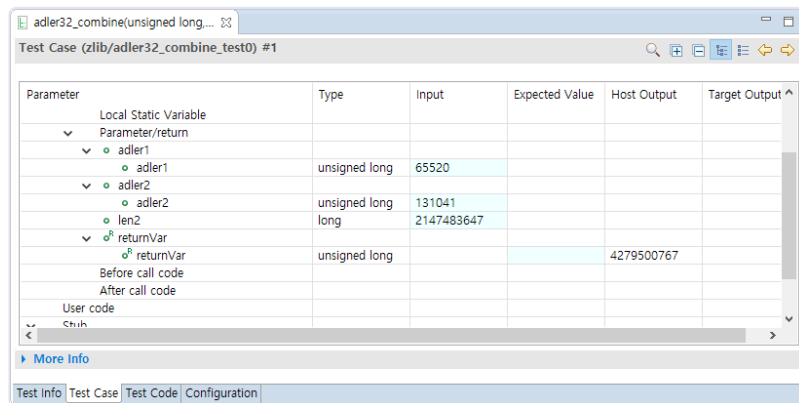
If you edit and save the information of a test that has already been executed, “The information has been changed after executing the test.” is displayed at the top of the test editor.





## 10.2. Test Case

Test cases 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.
- You can use ~ (range) and logical operators! (Not), & (and), and | (or) in expected values.
- The following shortcuts are available.
  - Move row: arrow keys (↑, ↓)
  - Editable cell focus: Enter
  - Editable cell focus out: Esc
  - Move column (editable cell): Tab (right), Shift + Tab (left)

## 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.



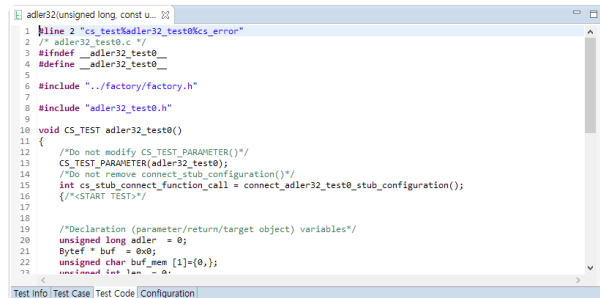
 `adler32_combine(unsigned long,...` **Test Case (zlib/adler32\_combine\_test0) #1**

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.



```
adler32(unsigned long const u... 33
1 #line 2 "cs_test\adler32_test0\cs_error"
2 /* adler32_test0.c */
3 #ifndef __adler32_test0__
4 #define __adler32_test0__
5
6 #include "../factory/factory.h"
7
8 #include "adler32_test0.h"
9
10 void CS_TEST adler32_test0()
11 {
12     /*Do not modify CS_TEST_PARAMETER()*/
13     CS_TEST_PARAMETER(adler32_test0);
14     /*Do not remove connect_stub_configuration()*/
15     int cs_stub_connect_function_call = connect_adler32_test0_stub_configuration();
16     /*<START TEST>*/
17
18
19     /*Declaration (parameter/return/target object) variables*/
20     unsigned long adler = 0;
21     Bytef * buf = 0x0;
22     unsigned char buf_mem [1]={0,};
23     /*end of test case*/
24 }
```

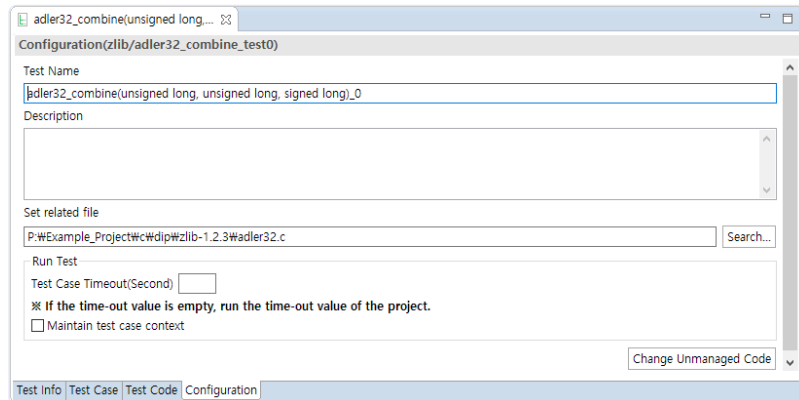
- If the user edits and saves the test structure tree, the changes are automatically reflected in the test code.
- If the return type of the function under test is a primitive type, a macro that outputs the return value is automatically generated.
- If you want to modify the test code yourself, you must 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 “[Ctrl] + [SPACEBAR]” shortcut in the editor view.

### ASSERT macro

Examine the conditional expression and print the success/failure in the test case view.

Macro	Parameter
CS_ASSERT(_b)	_B: conditional expression
CS_ASSERT_MSG(_B, _msg)	_b: conditional expression _mgs: message to print when the conditional expression is false

### Output macro

Print the values of specific variables in the test case view.

Macro	Parameter
CS_INT_OUTPUT(_v, _s)	_v: Integer variable _s: test data name
CS_UINT_OUTPUT(_v, _s)	_v: unsigned integer variable _s: test data name
CS_FLT_OUTPUT(_v, _s)	_v: real variable _s: test data name
CS_STR_OUTPUT(_v, _s)	_v: char* or char[] variable _s: test data name

### Input macro

Pass test data to the function under test.

Macro	Parameter
CS_INT_INPUT(_t, _s)	_t: integer type name _s: test data name



CS_UINT_INPUT(_t, _s)	_t: unsigned integer type name _s: test data name
CS_FLT_INPUT(_t, _s)	_t: real type name _s: test data name
CS_STR_INPUT(_t, _s)	_t: char* or char[] type name _s: test data name



Input macros cannot be used in stubs for target testing.

## 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 virtual memory area from 0xFFE40000U to 100.
CS_VIRTUAL_ADDR(0xFFE40000U,0xFFE40000U+100);

// 10 allocation of type int to 0xFFE40000U.
CS_ADDR_SET(int,0xFFE40000U,10);
CS_ADDR_ASSIGN(int,0xFFE40000U,10);

// The value of 0xFFE40000U is assigned to the variable a.
a = CS_ADDR_GET(int,0xFFE40000U);
```

## Other macros

Macro	Parameter	Description
CS_LOG(_msg)	_msg: log message	Outputs user log
CS_TESTCASENO()		Returns the number of the test currently running



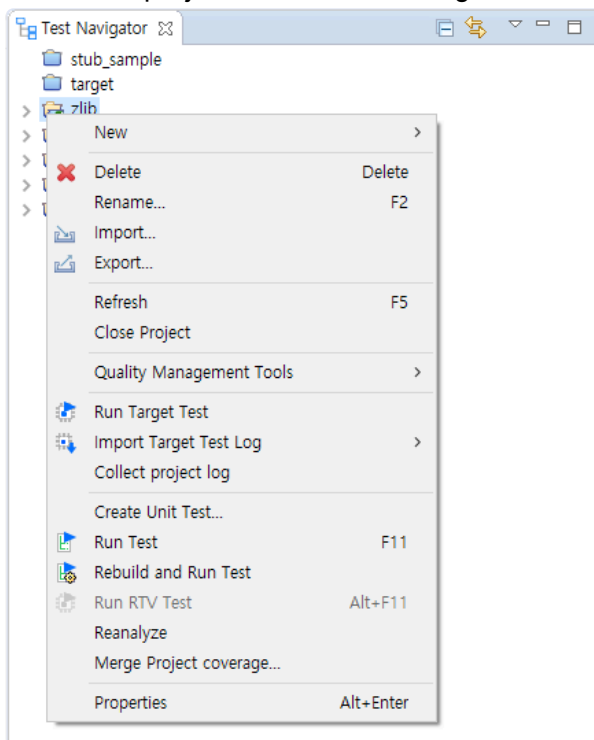
# 11. Test Run

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 and select [Run test]

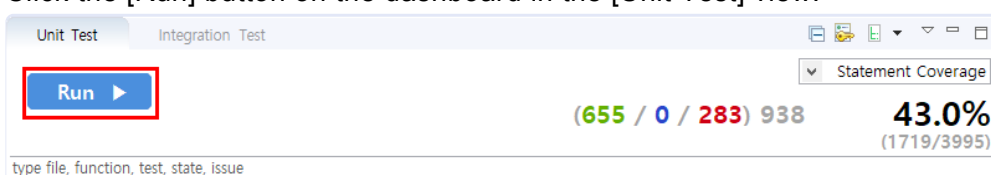


2. A progress monitor of the test opens. When the test is completed, you can check the results through each test view.

## 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, right-click on the test case 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. Properties

---

Controller Tester provides a page to set the properties of the project, module, and source files (translation unit).

The property page types are:

- [Project Properties](#)
- [Module Properties](#)
- [Source File\(Translation Unit\) Properties](#)



## 12.1. Project Properties

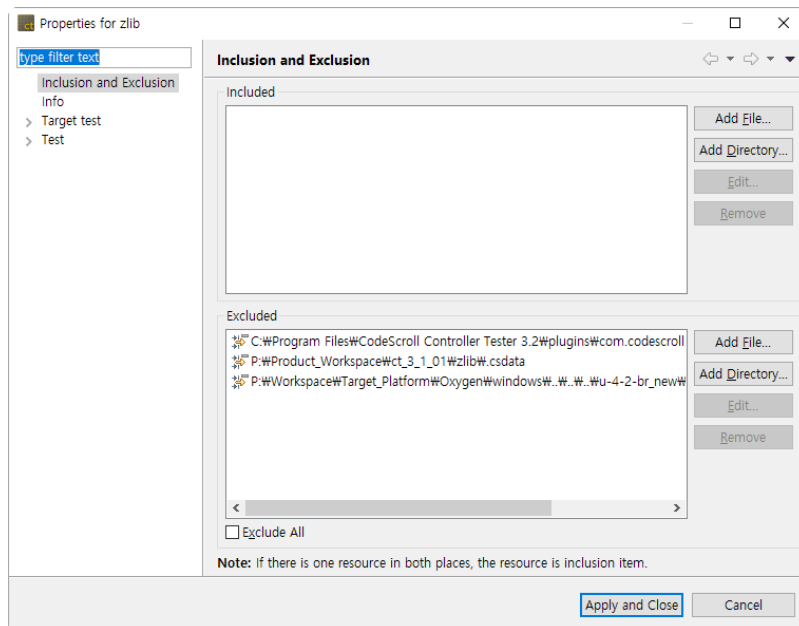
You can edit the properties of the created project.

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].

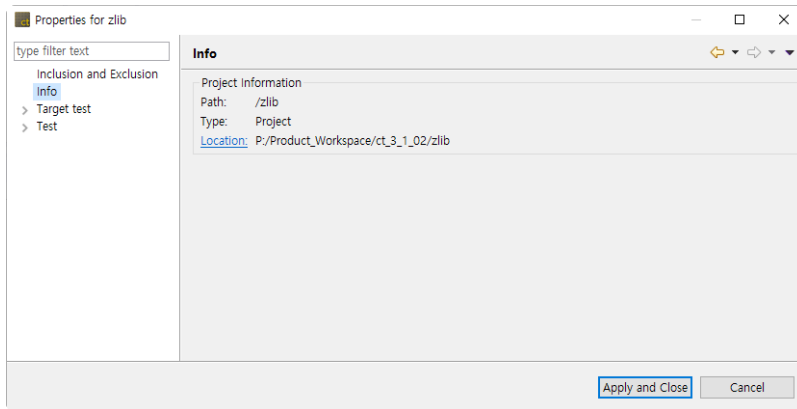


- Add targets to include in the analysis.
  1. Click the [Add File] or [Add Directory] button.
  2. Select files or a directory to analyze.
  3. Click the [Open] or [OK] button.
- Add targets to exclude from the analysis.
  1. Click the [Add File] or [Add Directory] button.
  2. Select files or a directory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



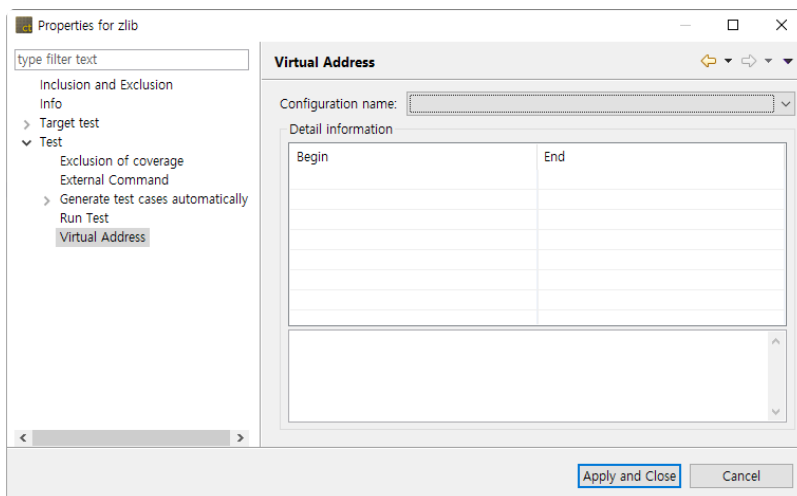
## Info

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



## Virtual Address

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



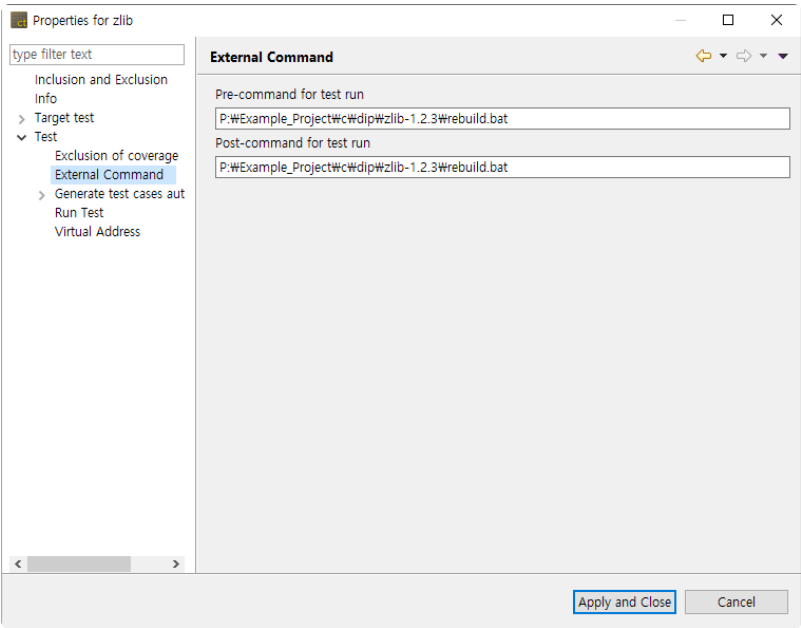
## External Command

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.

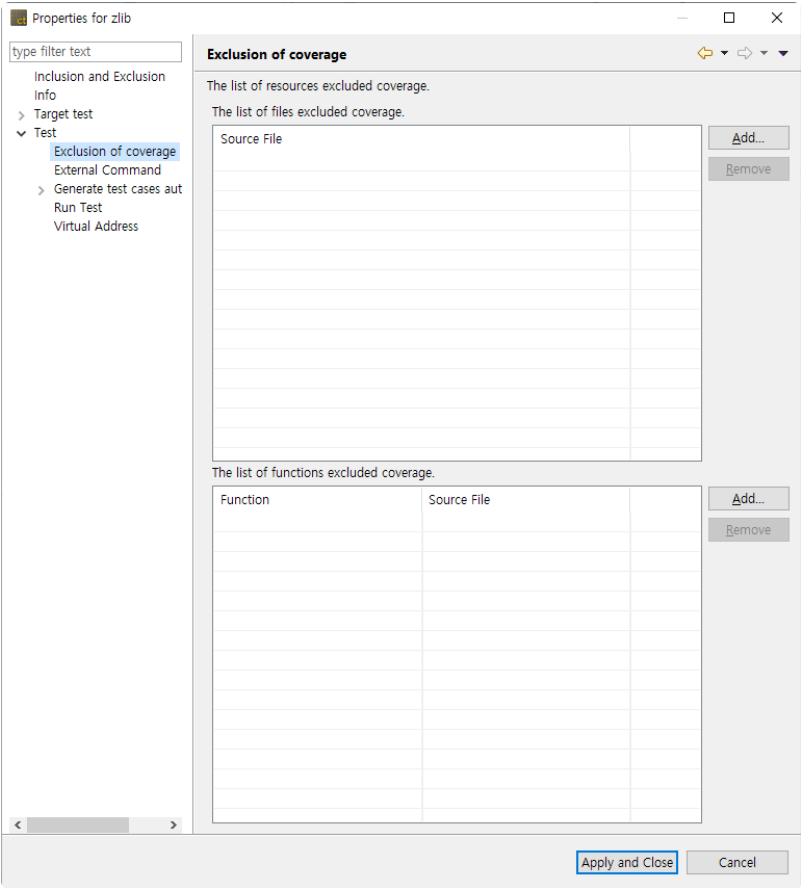
In [Post-command for test run], you can enter a command or batch file to execute an external command after running the test.





# 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.



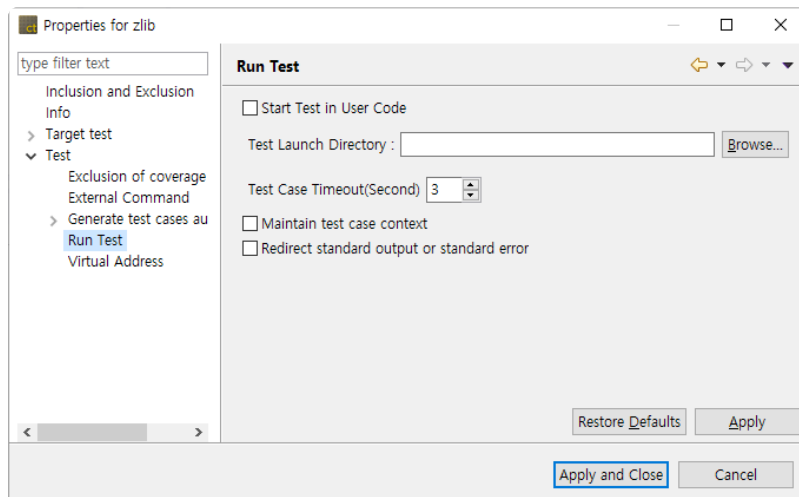


## Run Test

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.

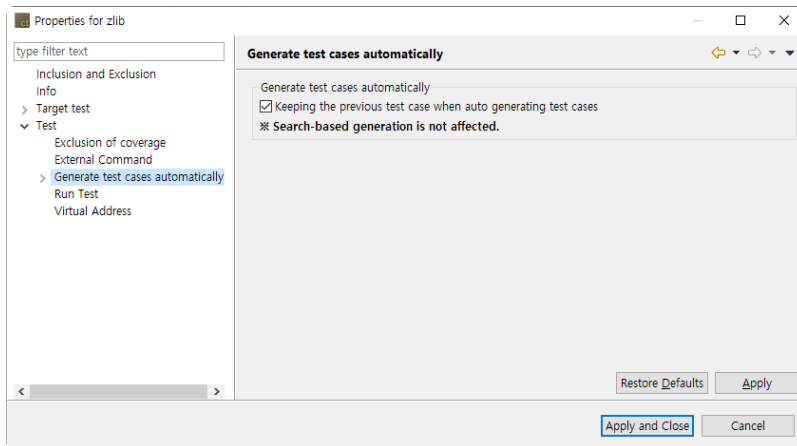
[Redirect standard output or standard error] is a feature to record standard output and standard error in the log file for each test case.

## Generate test cases automatically

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

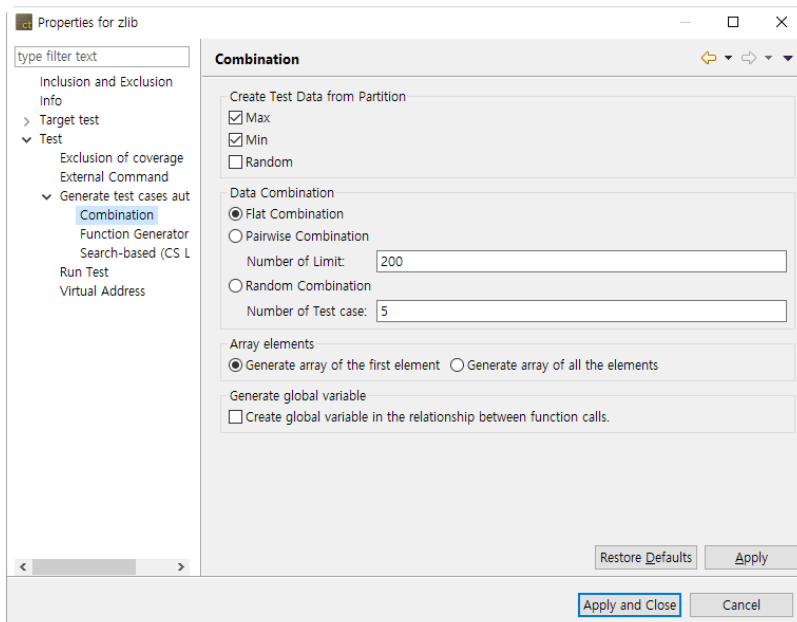
**!** The search-based generation is not affected.





## 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.

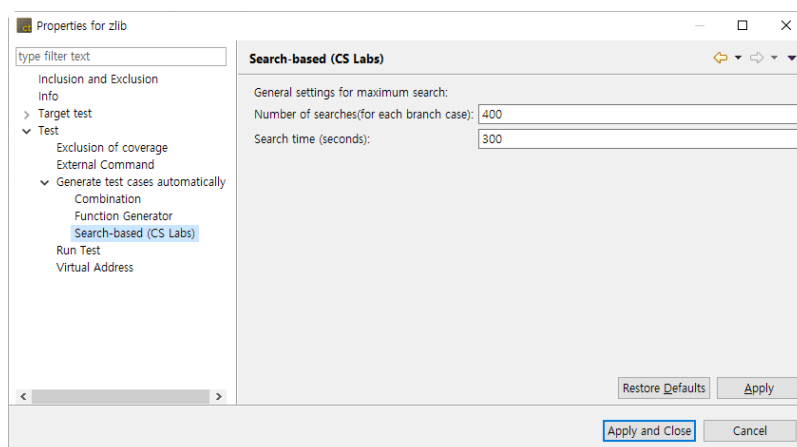
When creating a test, global variable creation is an option to create all global variables used by all functions called from the function under test.

### 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.

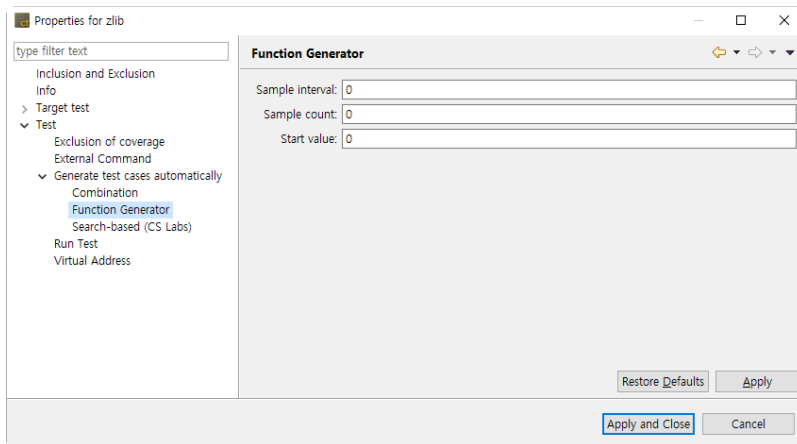




Option	Description
Number of searches(for each branch)	Searches for the number of times entered for each branch combination.
Search time(seconds)	Searches for the time entered.

## Function Generator

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

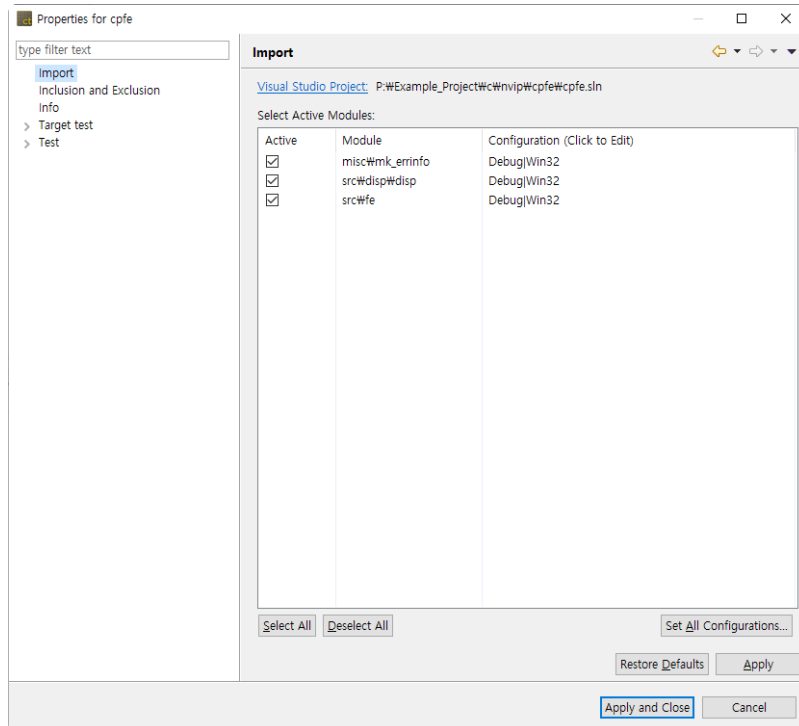


Option	Description
Sample interval	Interval of samples when sampling from function
Sample count	Number of samples when sampling from function (Number of test cases)
Start value	Default value at which the function starts, and generates a value based on the value



# Import settings

## Project created by Visual Studio project



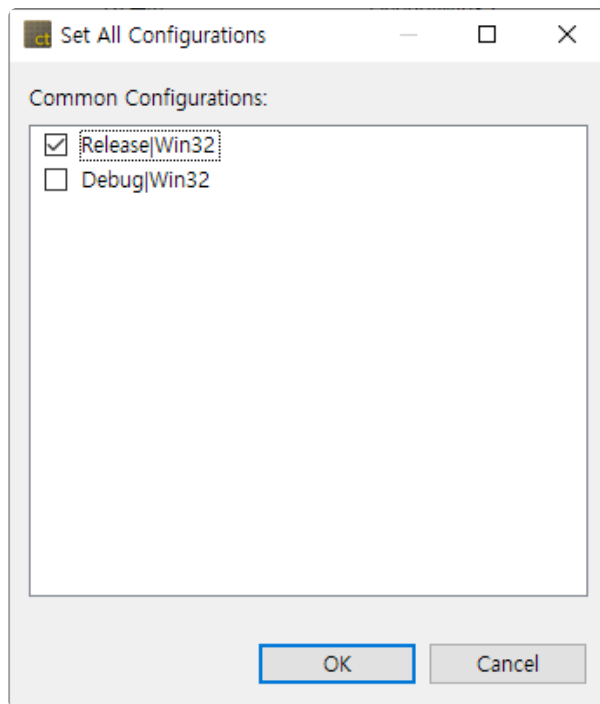
You can see the path of the Visual Studio project used for creating the project and the modules active in the selected project.

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.





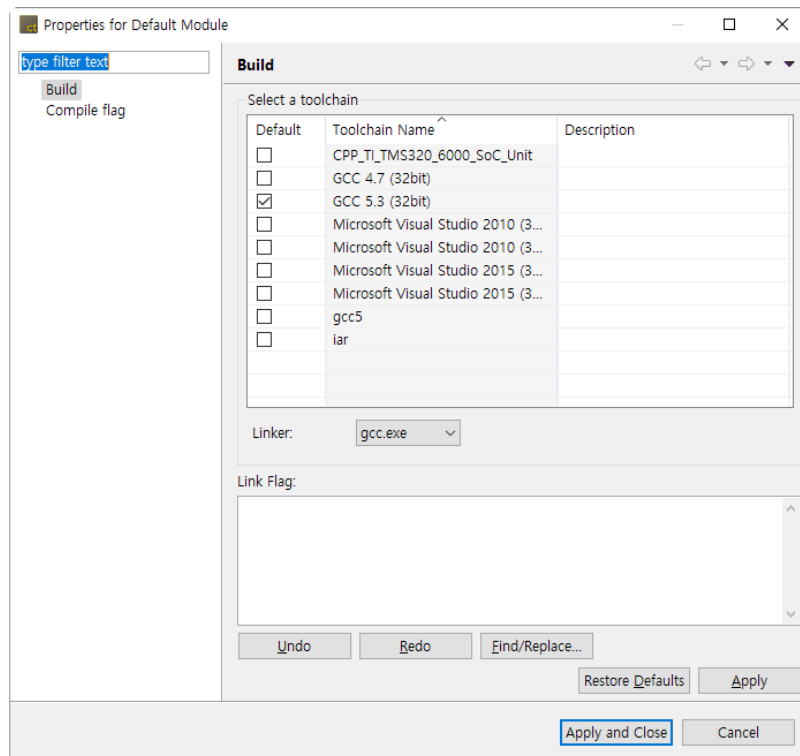
2. Select the configuration to be applied to all modules in common and click the [OK] button.



## 12.2. Module Properties

### Build

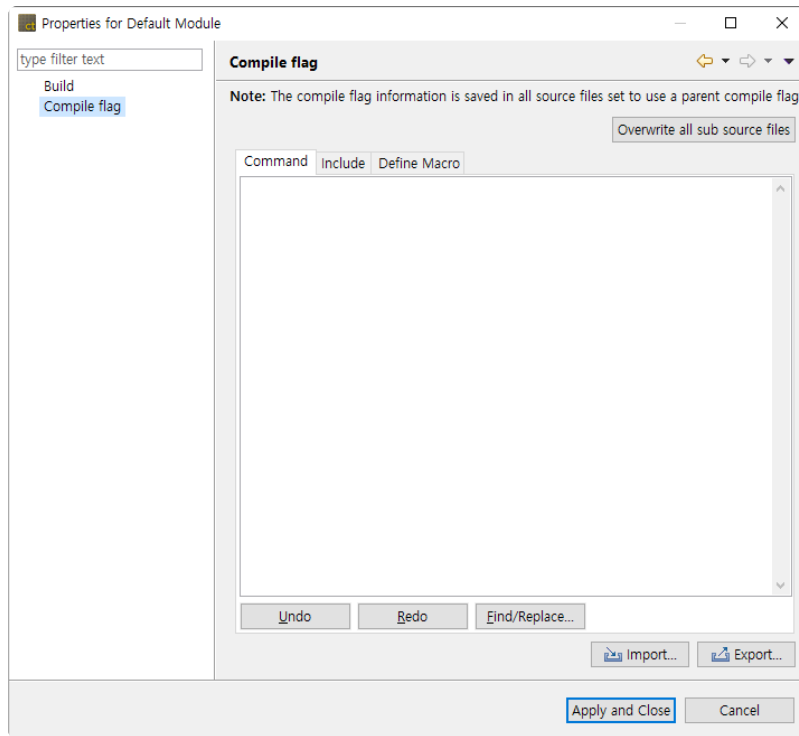
You can set toolchain related information (toolchain, linker) and link flags for the module. Changing the toolchain also changes the toolchain set in all source files(translation units) under the module.



### Compile flag

You can set compile flag information for that module.





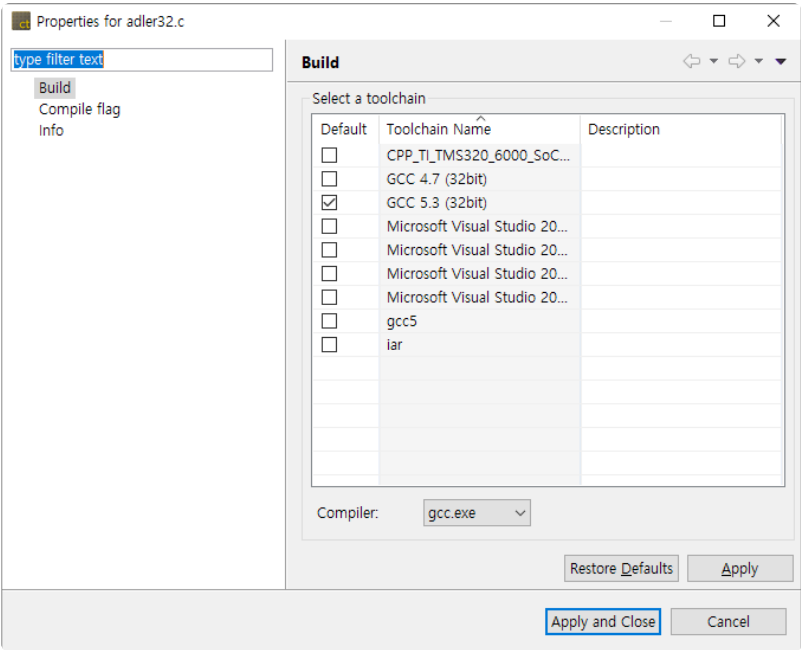
- [Overwrite all sub source files] overwrites the compile flag entered to all source files (translation units) under the module.
- [Import] imports an external compile flag. [Export] saves the current compile flag externally.
  - Import
    1. Click the [Import] button.
    2. Select the file (.cf file) 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.



# 12.3. Source File Properties

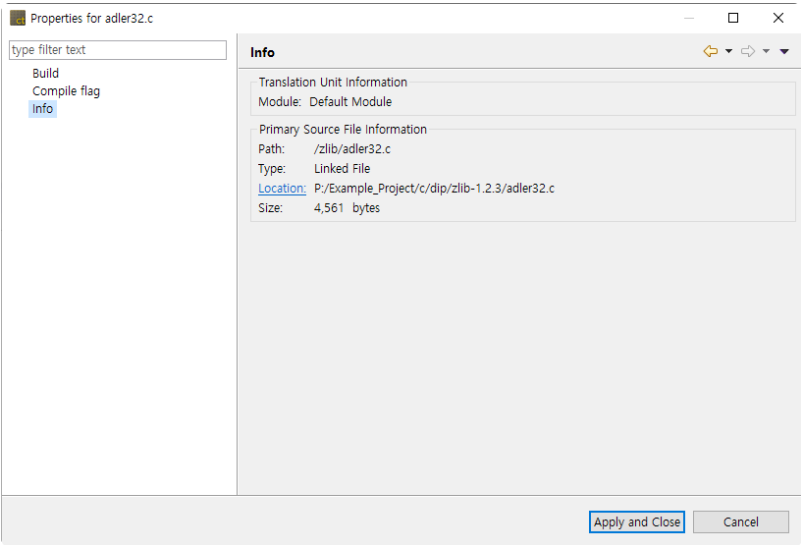
## Build

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



## Info

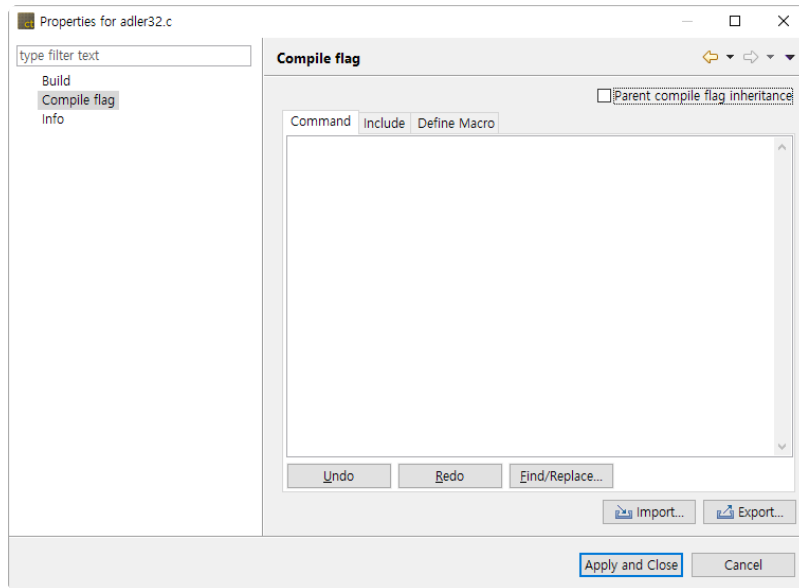
Shows the information of the source file (translation unit) and the module containing the source file.





## Compile flag

You can set compile flags for source files(translation units).



- If you select the [Parent compile flag inheritance] check button, the compile flag of the module containing the source file is applied. Conversely, deselecting can change the current compilation flag.
- [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.



# 13. 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](#)
- [Editor](#)
- [Exclusion](#)
- [Performance](#)
- [Perspective](#)
- [Report](#)
- [Source File Types](#)
- [Test](#)
- [Theme](#)
- [Toolchain](#)



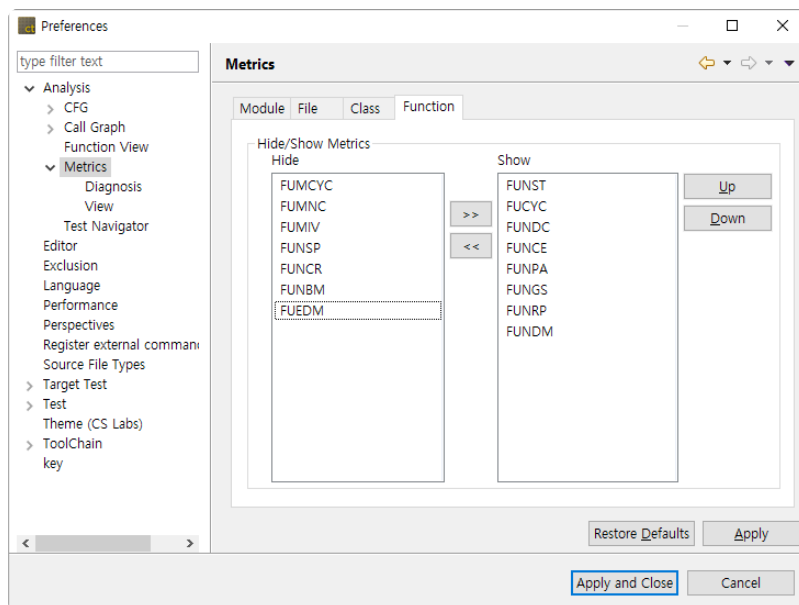
# 13.1. Analysis

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.



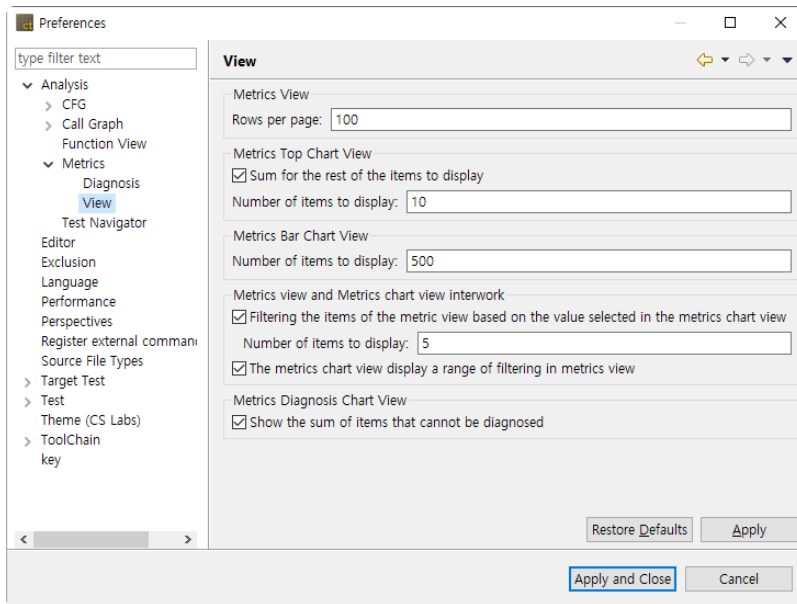
After selecting a metric, you can move to the 'Hide' list or the 'Show' list with the [>>] or [<<] button.

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.

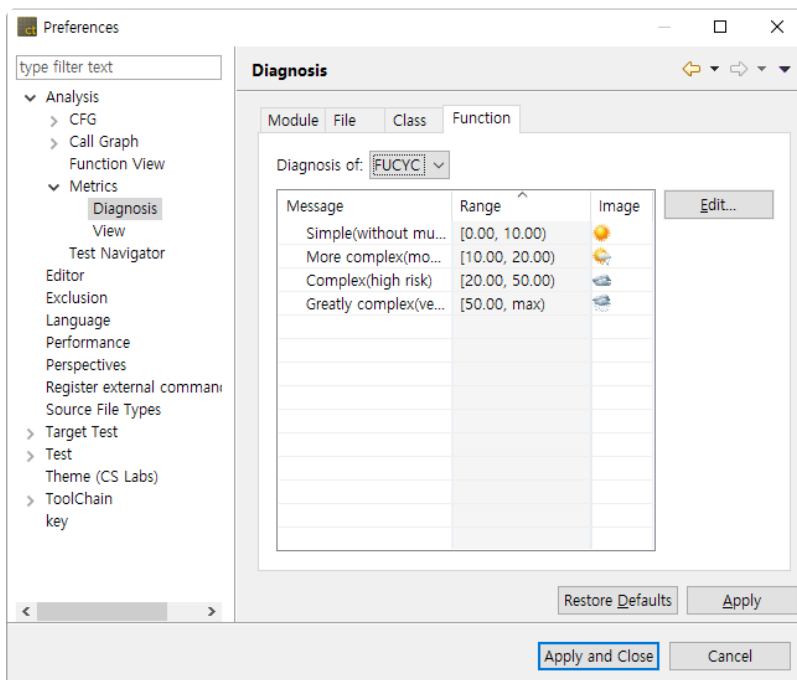




## Diagnosis

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.



Edit Diagnosis of 'FUCYC'

Diagnosis Level: 4

Set of Images: Weather4 [Change Sort Order](#)

Level1

Message: Simple(without much risk)

Range: 0.00 ~ 10.00 Image: ☀

Level2

Message: More complex(moderate risk)

Range: 10.00 ~ 20.00 Image: ☁

Level3

Message: Complex(high risk)

Range: 20.00 ~ 50.00 Image: ☁

Level4

Message: Greatly complex(very high risk)

Range: 50.00 ~ max Image: ☁

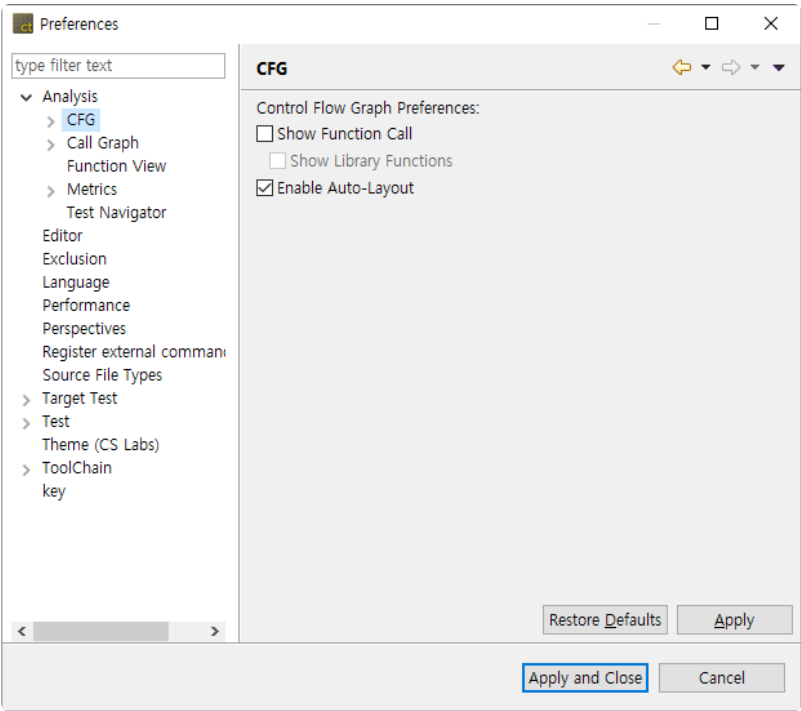
OK Cancel

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

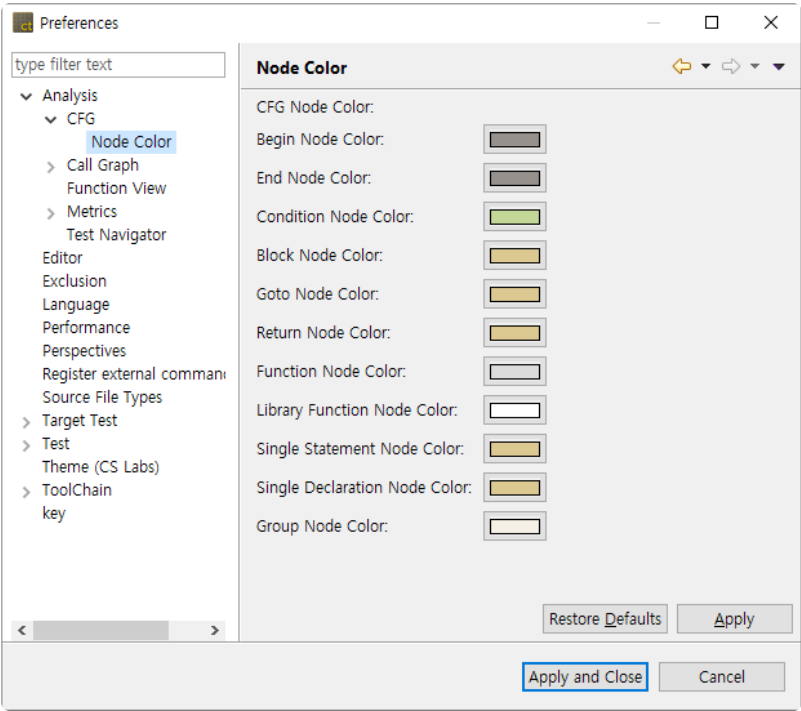
You can change the settings for the control flow graph displayed in the control flow graph view.





Node Color

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

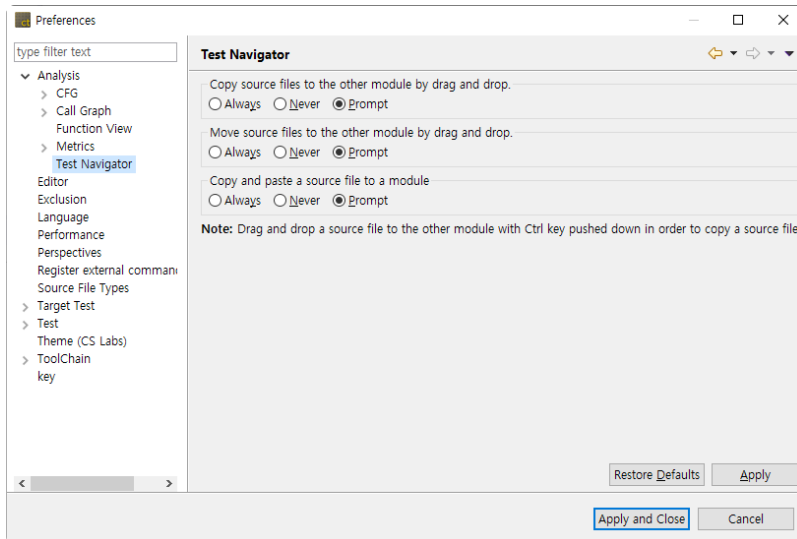


Test Navigator

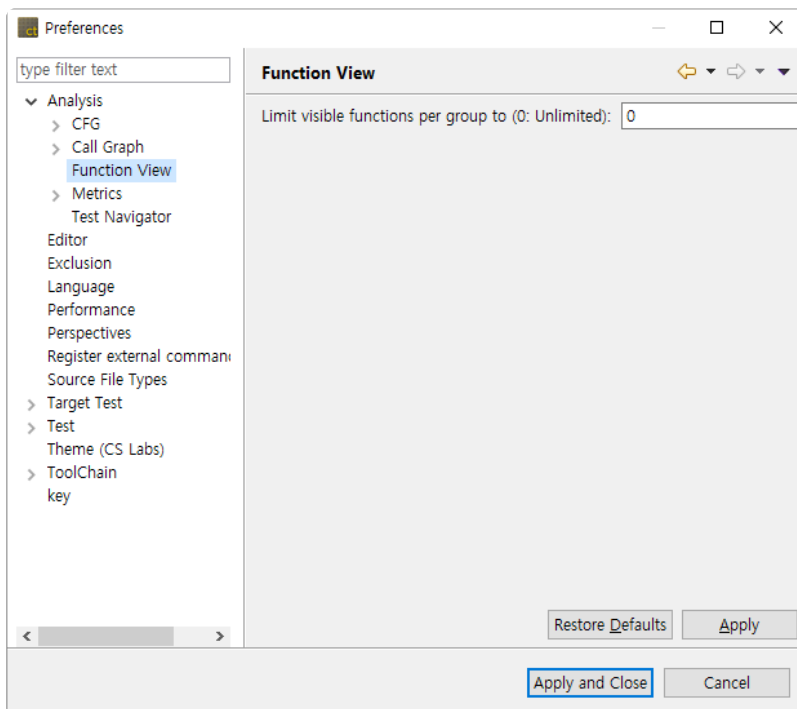
For each operation that copies or moves source files in the Test Navigator view, you can set options to



ask whether or not to continue.



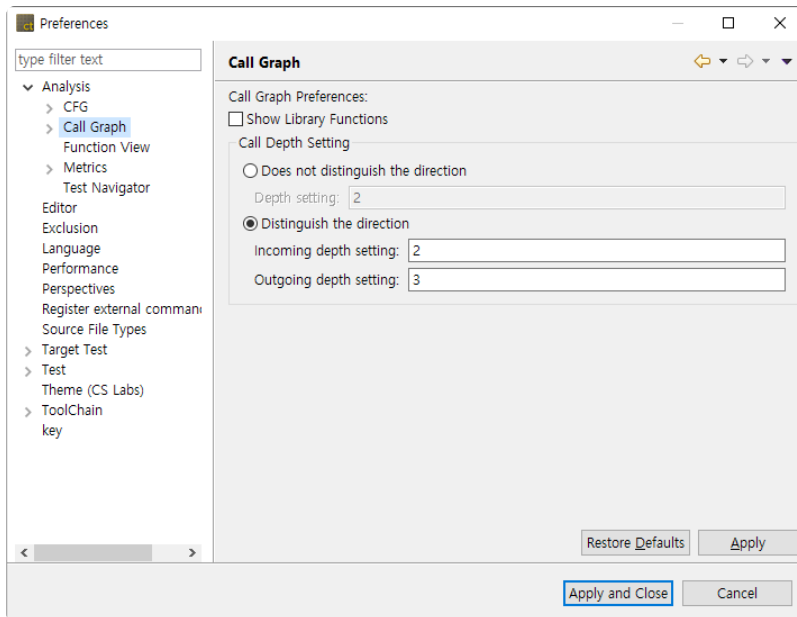
## Function View



## Call Graph

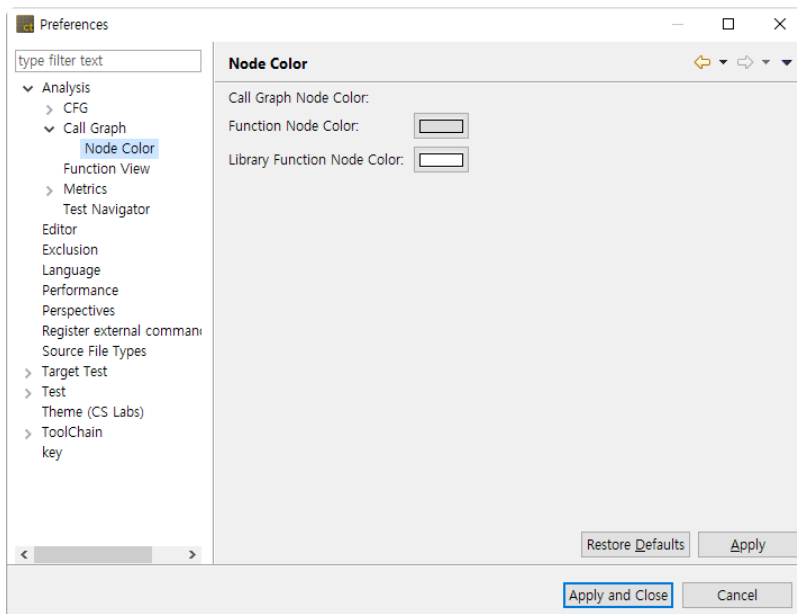
You can change the settings of the function call graph displayed in the call graph view.





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).

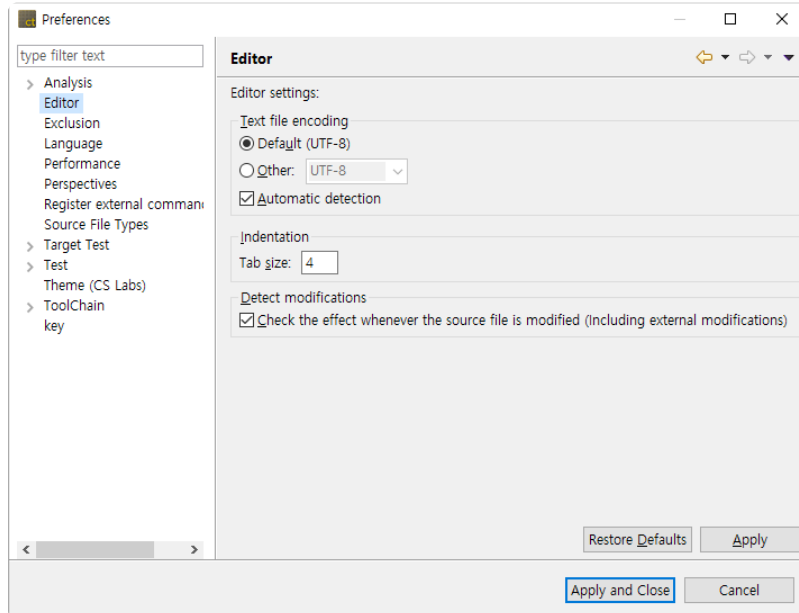
If you want to set the caller and callee separately, you can specify the value of each step by selecting the sub-option. You can change the color of nodes visible in the function call graph.





## 13.2. Exclusion

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

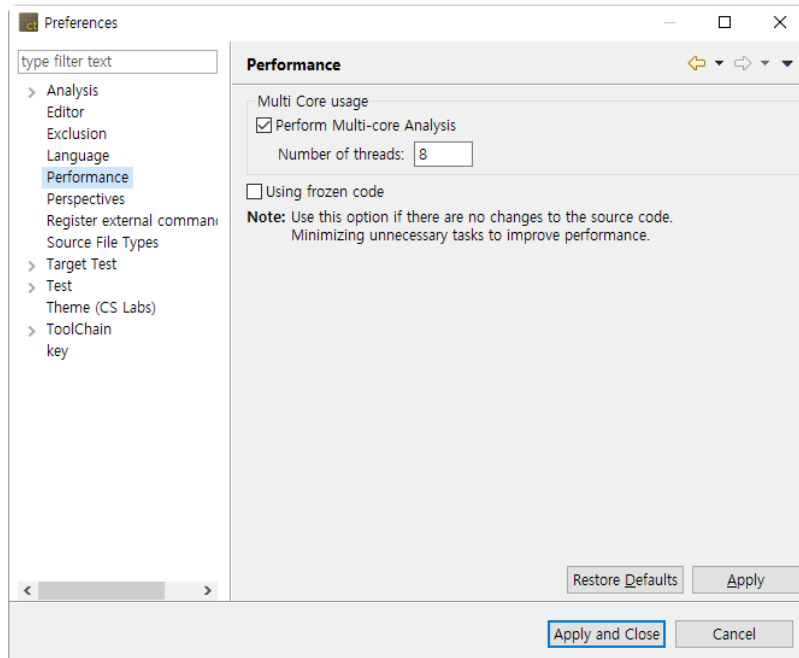


- Add the analysis exclusion target.
  1. Click [Add].
  2. Select the directories or files to be excluded.
  3. Click [OK] or [Open].
- Delete the analysis exclusion target.
  1. Select the directories or files to be deleted.
  2. Click [Remove].



## 13.3. 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.)



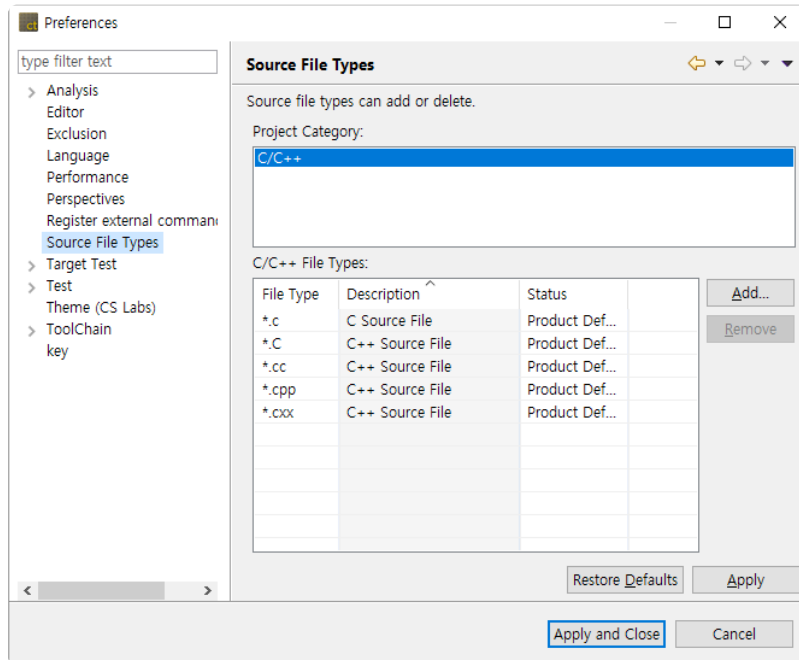
**Difference with Editor – [Detect modifications] option**

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.

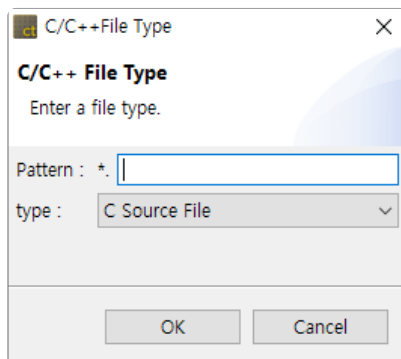


## 13.4. Source File Types

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



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



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

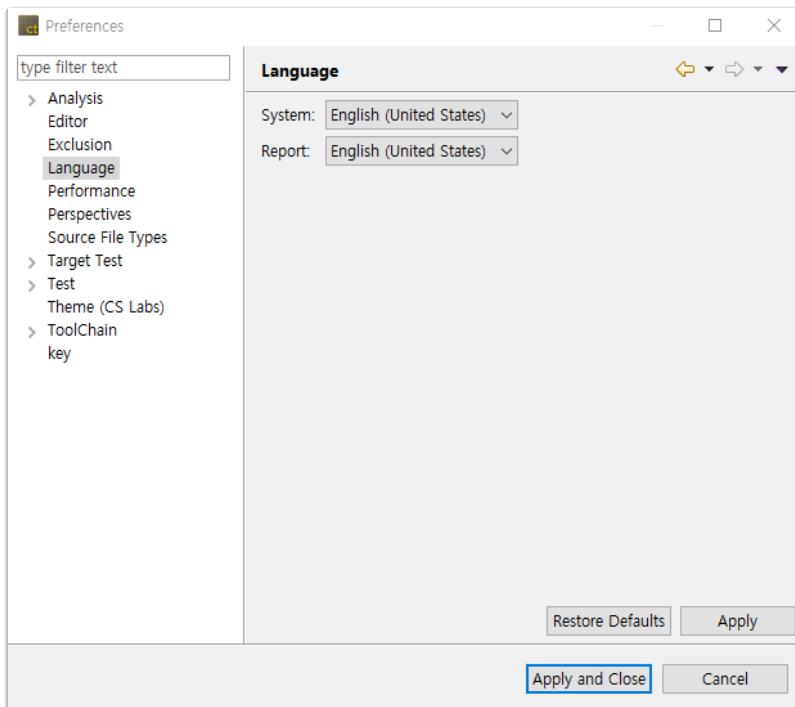


## 13.5. Language

---

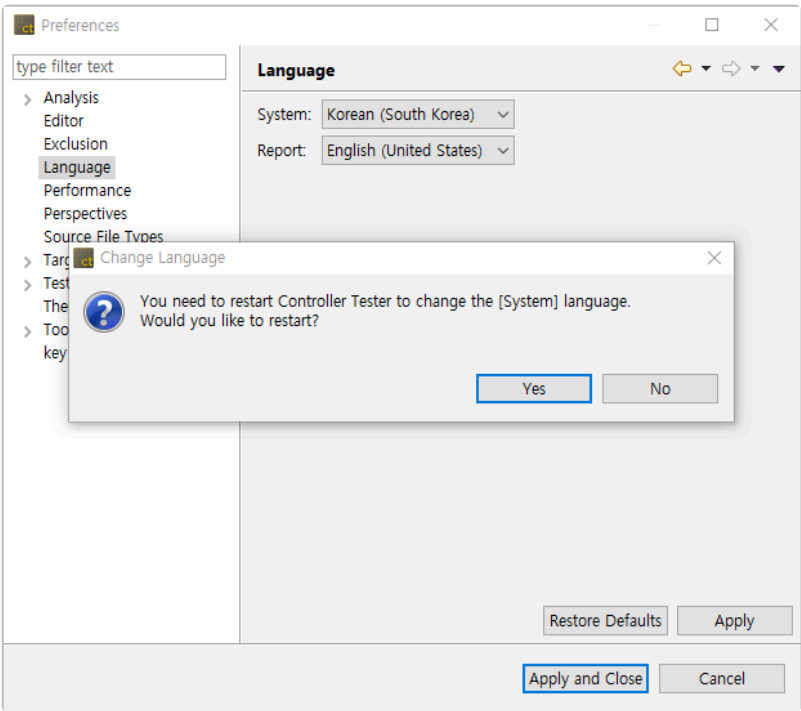
Sets the language to be displayed in tools and reports.

You can select a language from the drop-down menu for [Tool Language] and [Report Language].



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



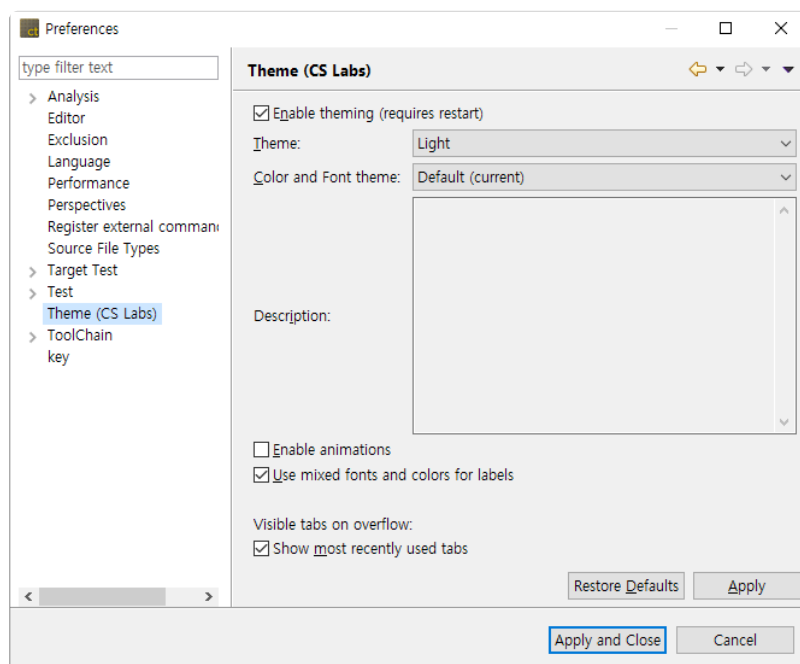




## 13.6. Theme

It provides the settings for changing the theme of the Controller Tester.

✿ 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.





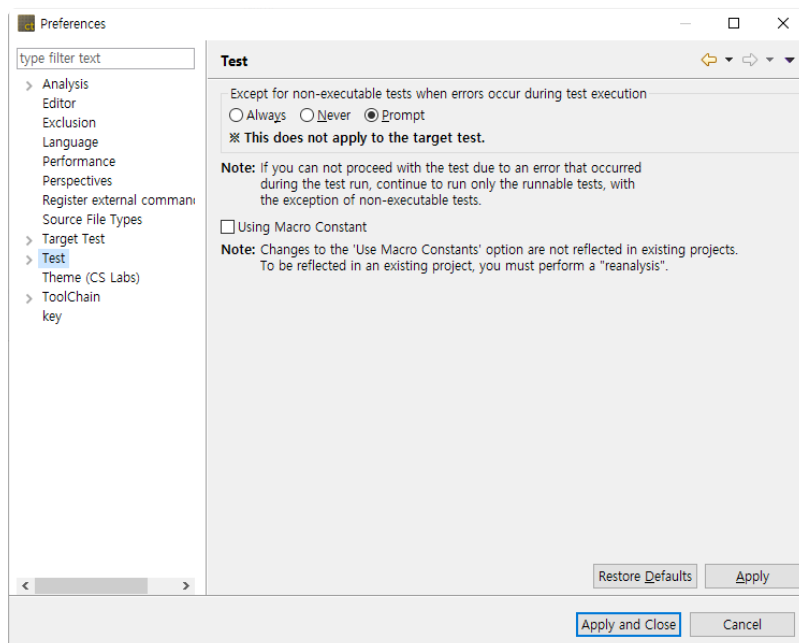
## 13.7. Test

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).

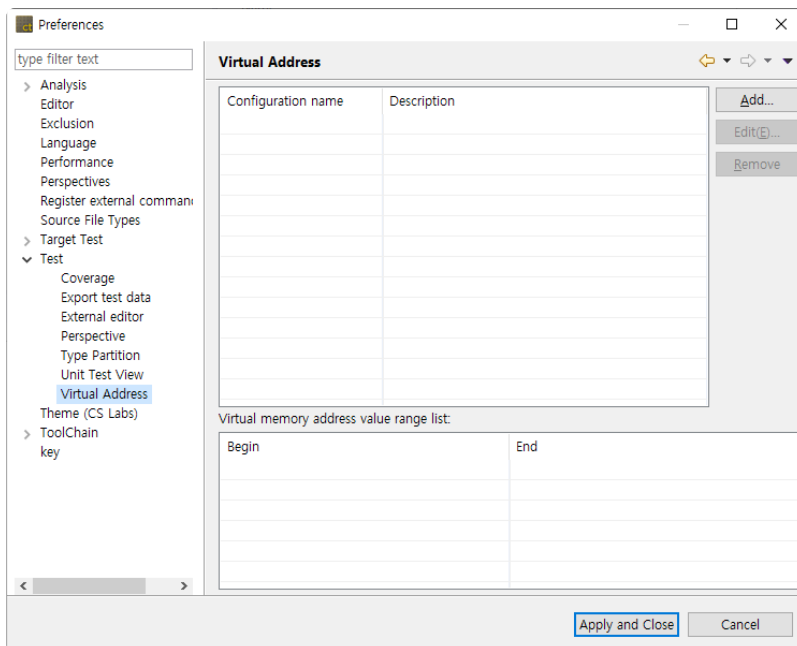
**!** This setting only applies to converted projects.



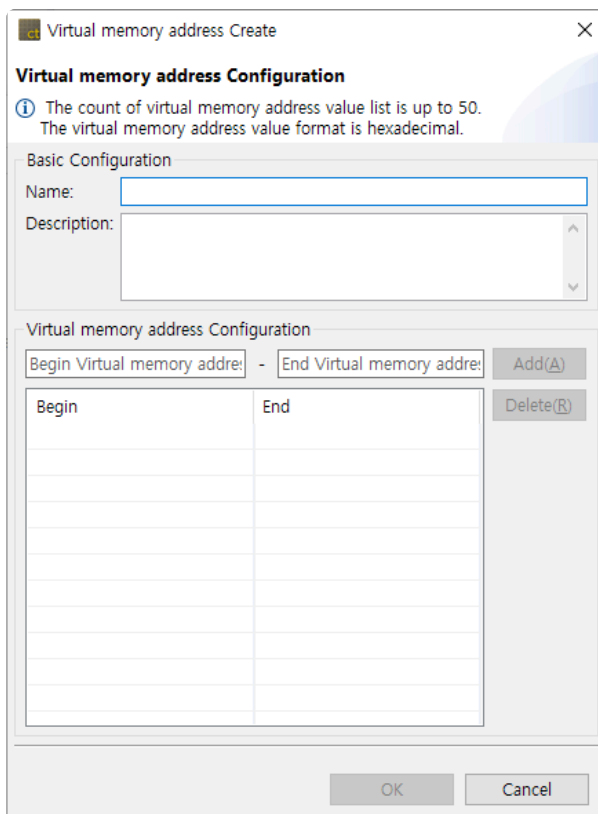
## Virtual Address

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.

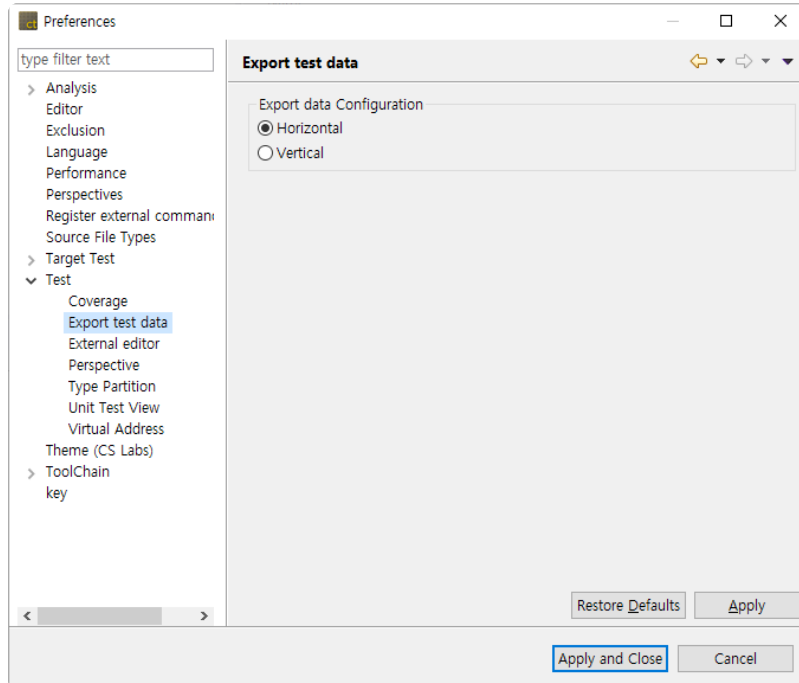


The virtual memory address areas can be specified up to 50 areas and it can be entered in hexadecimal.



## Export test data

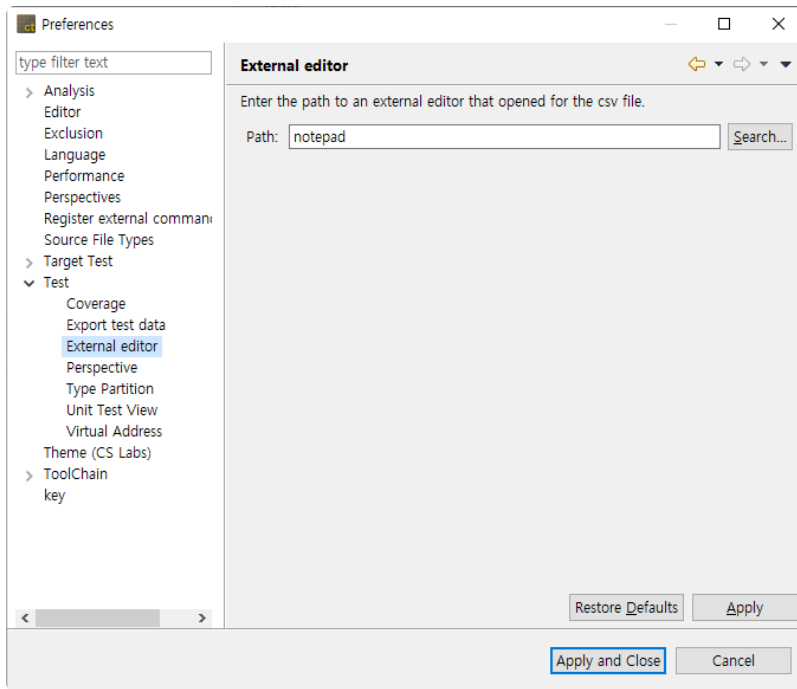
When exporting data, you can set the output direction of variables to either horizontal or vertical. Select the output direction of the export variable and click [Apply].



## External Editor

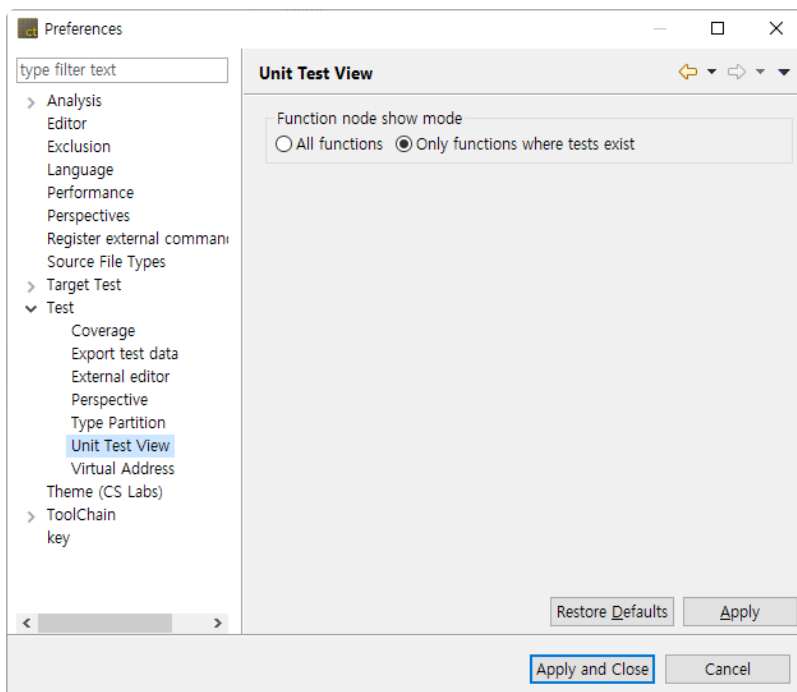
You can set the external editor to open CSV file. Enter the path of the external editor and click [Apply].





## Unit Test View

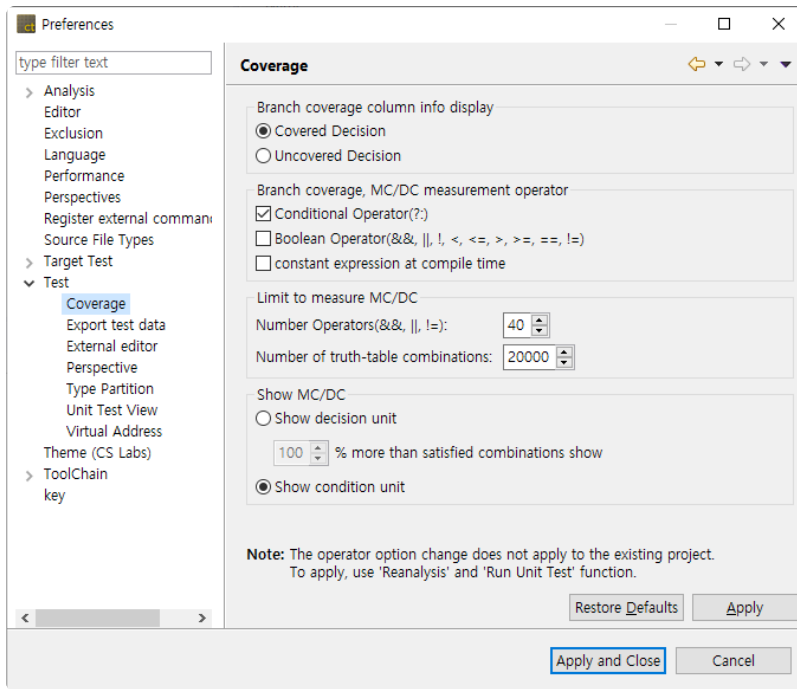
You can set the function node display in the Unit Test view included in the test perspective that UI/UX is improved.



## 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

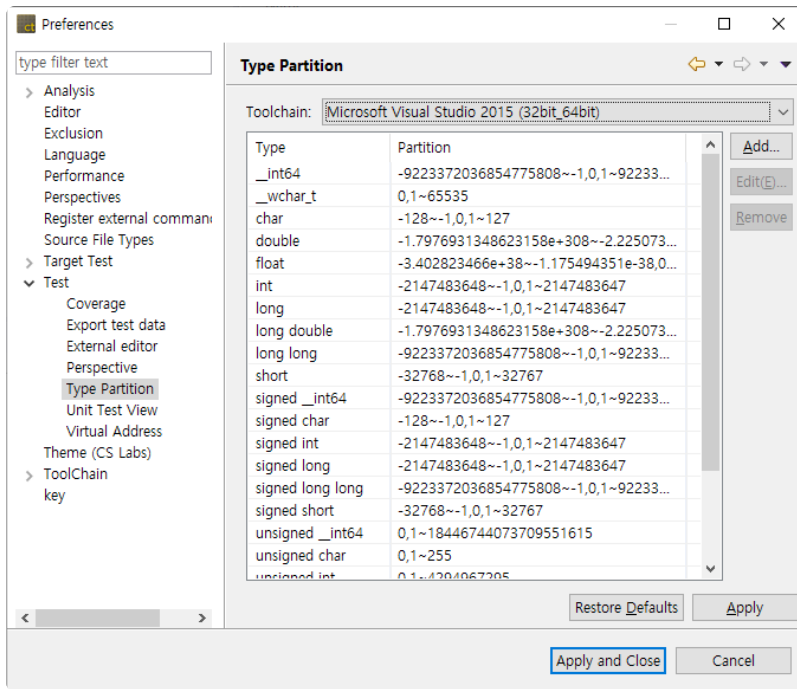
The checkbox [Branch coverage, MC/DC measurement operator] sets the branch coverage measuring target. If it is re-analyzed and executed, the settings are applied.

- 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.

## Type Partition

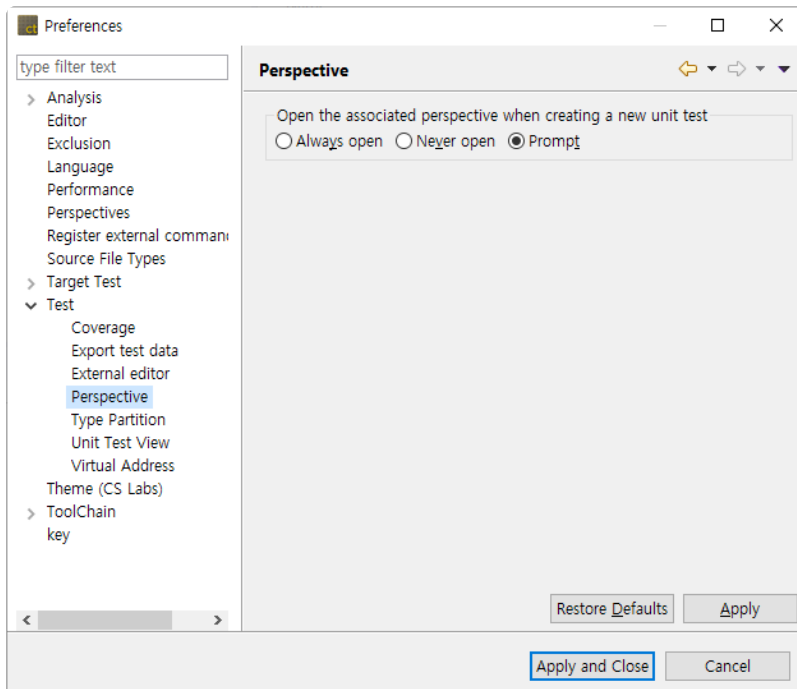
You can edit the basic type partition for each toolchain and restore it into the default value provided. In Toolchain combo box, select the toolchain to be modified, edit the partition and click [Apply].





## 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].





## 13.8. Toochain

---

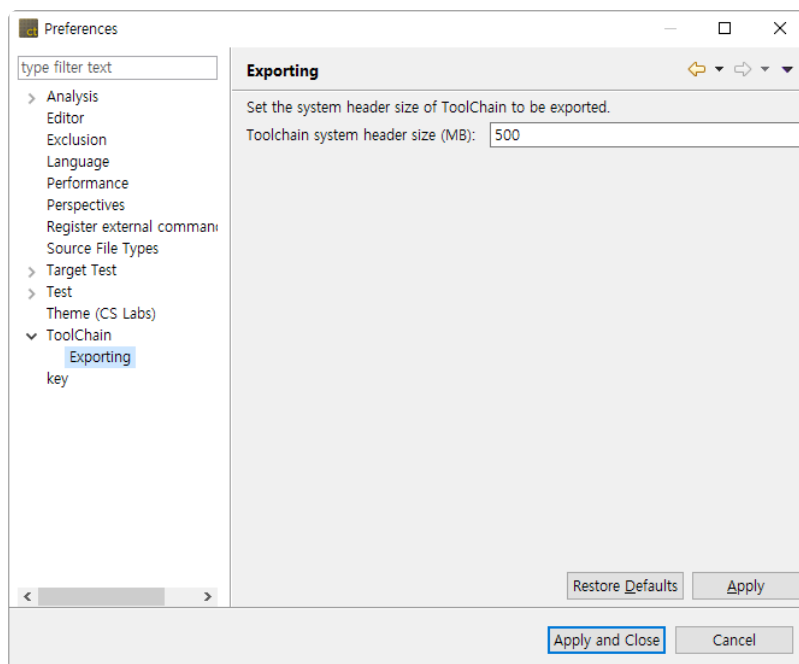
You can set the information for the toolchain to be used in the tool.

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](#) Toolchain (analyzer) settings”.

### Export Toolchain

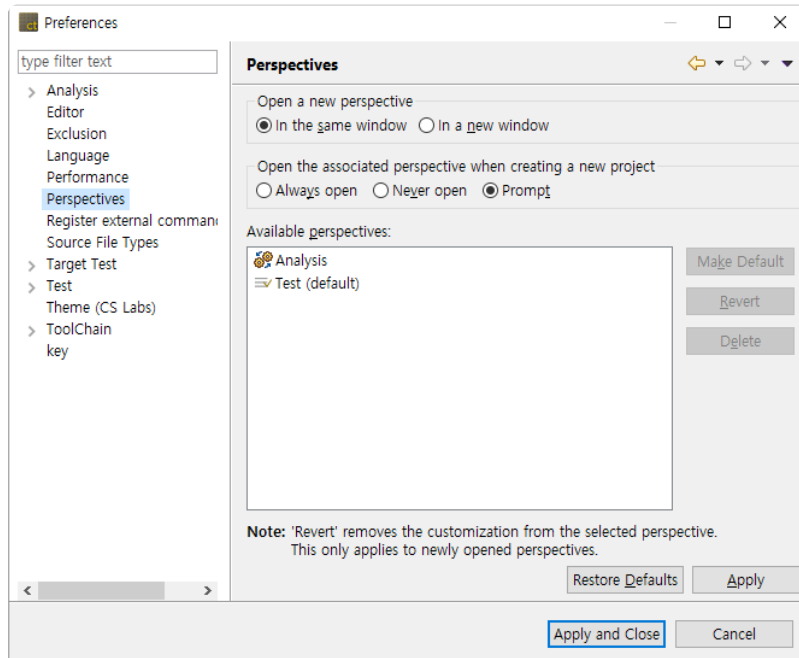
It sets the system header size of the toolchain to be exported. The system headers larger than the size you set cannot be exported.





## 13.9. Persperctive

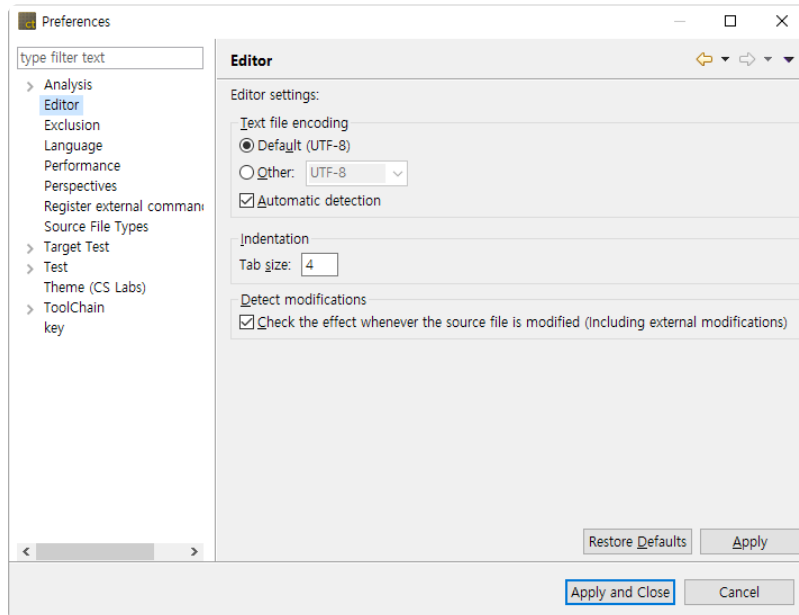
You can change the settings related to perspective.





## 13.10. 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.



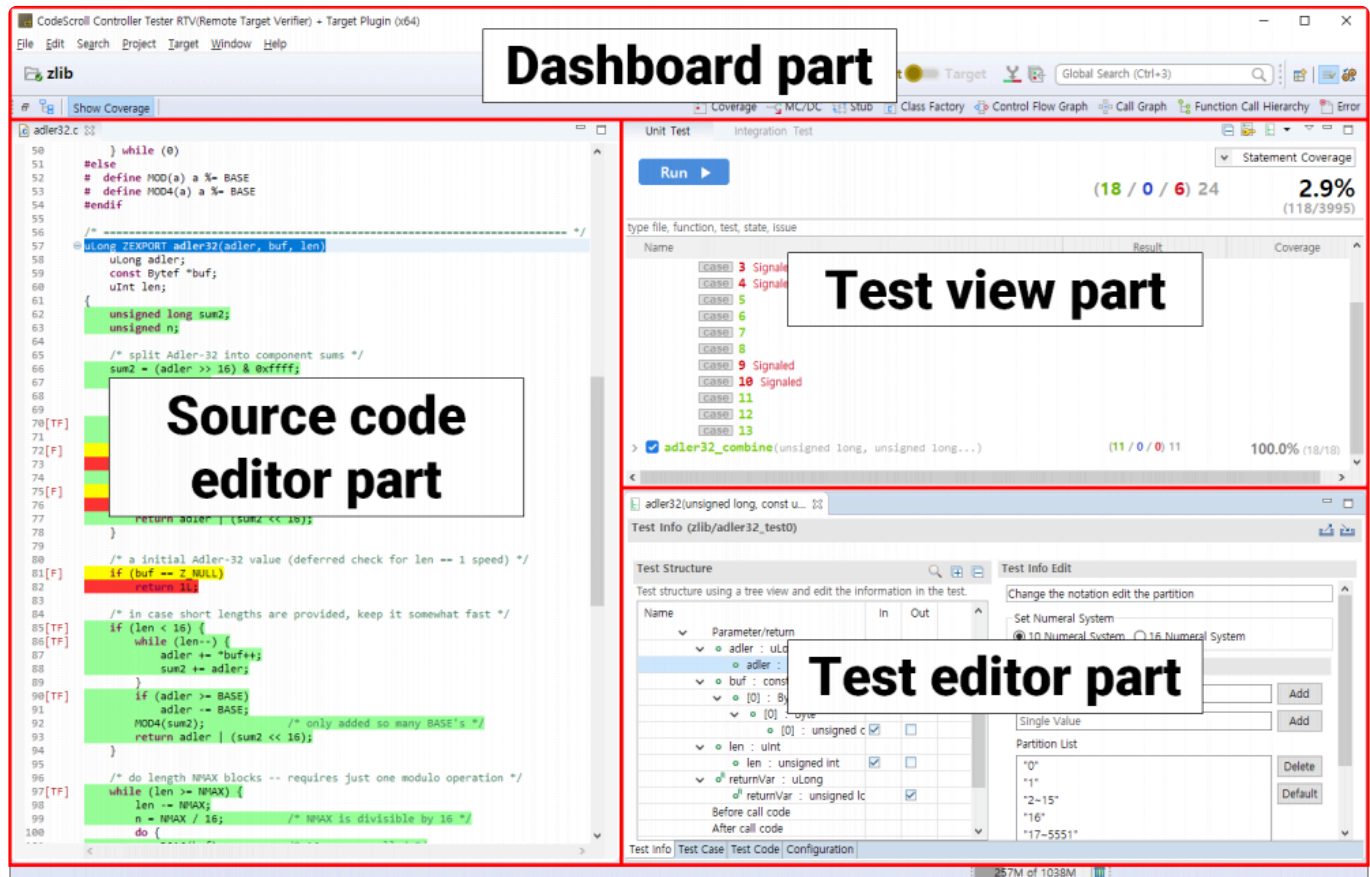
If the disc speed is slow, deselect this checkbox to improve the performance.



# 14. 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](#)
- [Source Code Editor](#)
- [Unit Test View](#)
- [Integration Test View](#)
- [Coverage View](#)
- [MC/DC View](#)
- [Stub View](#)
- [Class Factory View](#)
- [Control Flow Graph View](#)

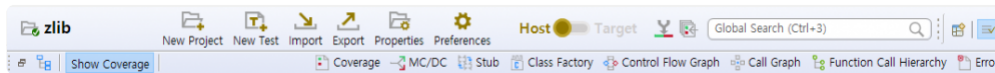


- [Call Graph View](#)
- [Function Call Hierarchy](#)
- [Error View](#)
- [Fault Injection View](#)
- [Input/Output Data Graph View](#)
- [Console View](#)



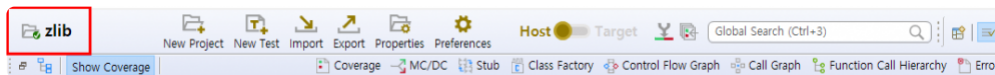
## 14.1. Dashboard

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



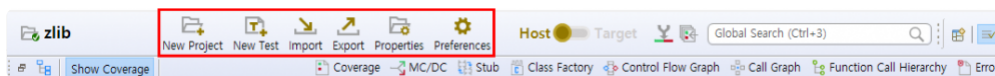
### Project name

It shows the name of the project selected in the [Test Navigator] view.



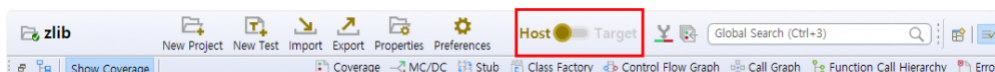
### Bookmark menu

You can use the Create Project, Create Test, Import, Export, Properties and Preferences functions.



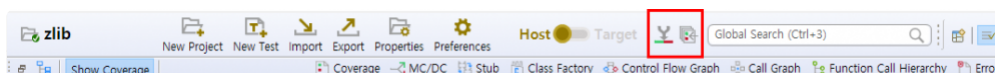
### Host/Target setting menu

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



### Coverage-related menu

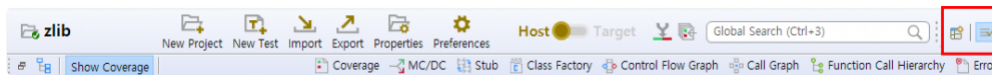
You can use the Show the Merger coverage between host and target and the Show full coverage (Include External Coverage) functions.



### Open Perspective menu

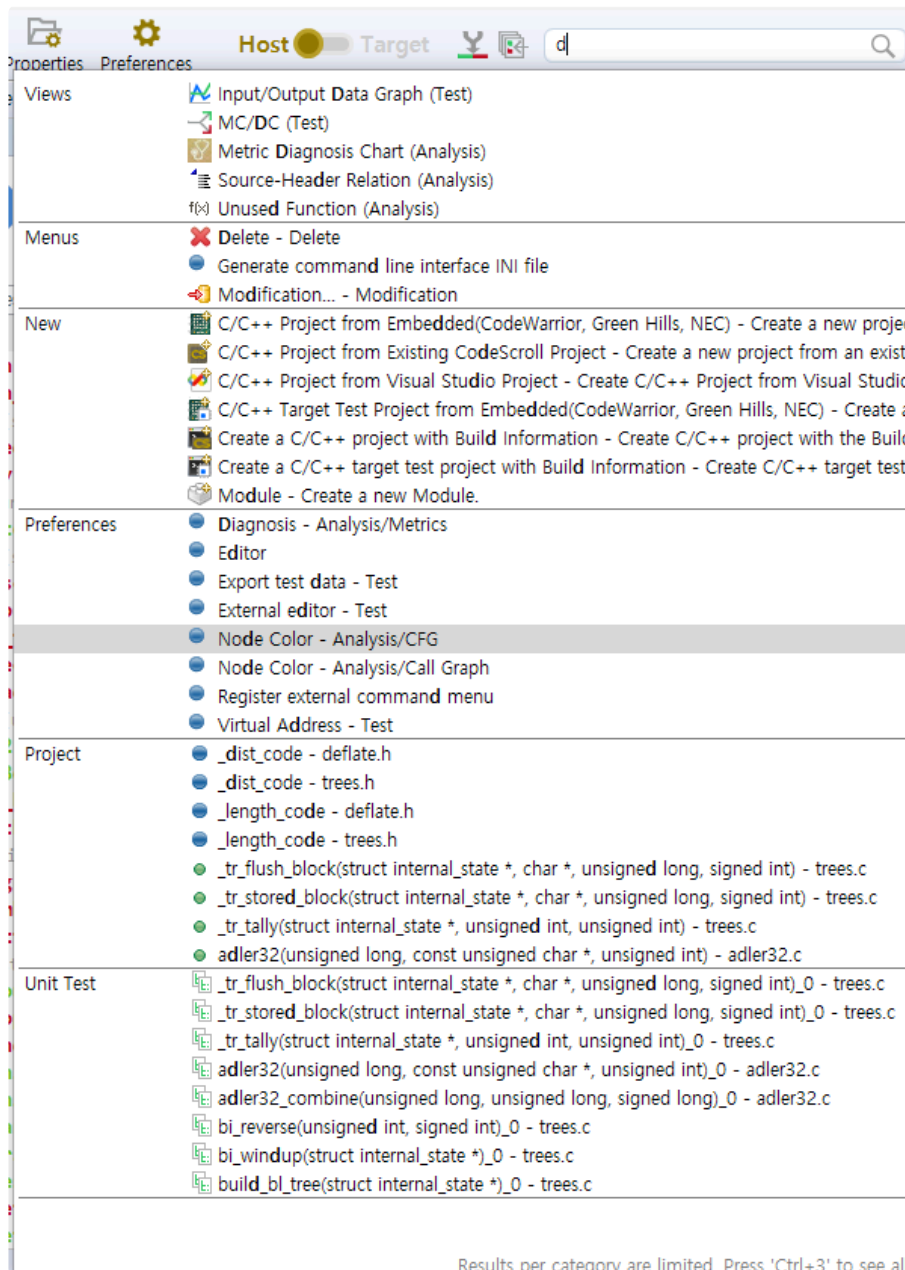
You can select and open the perspective located in Controller Tester





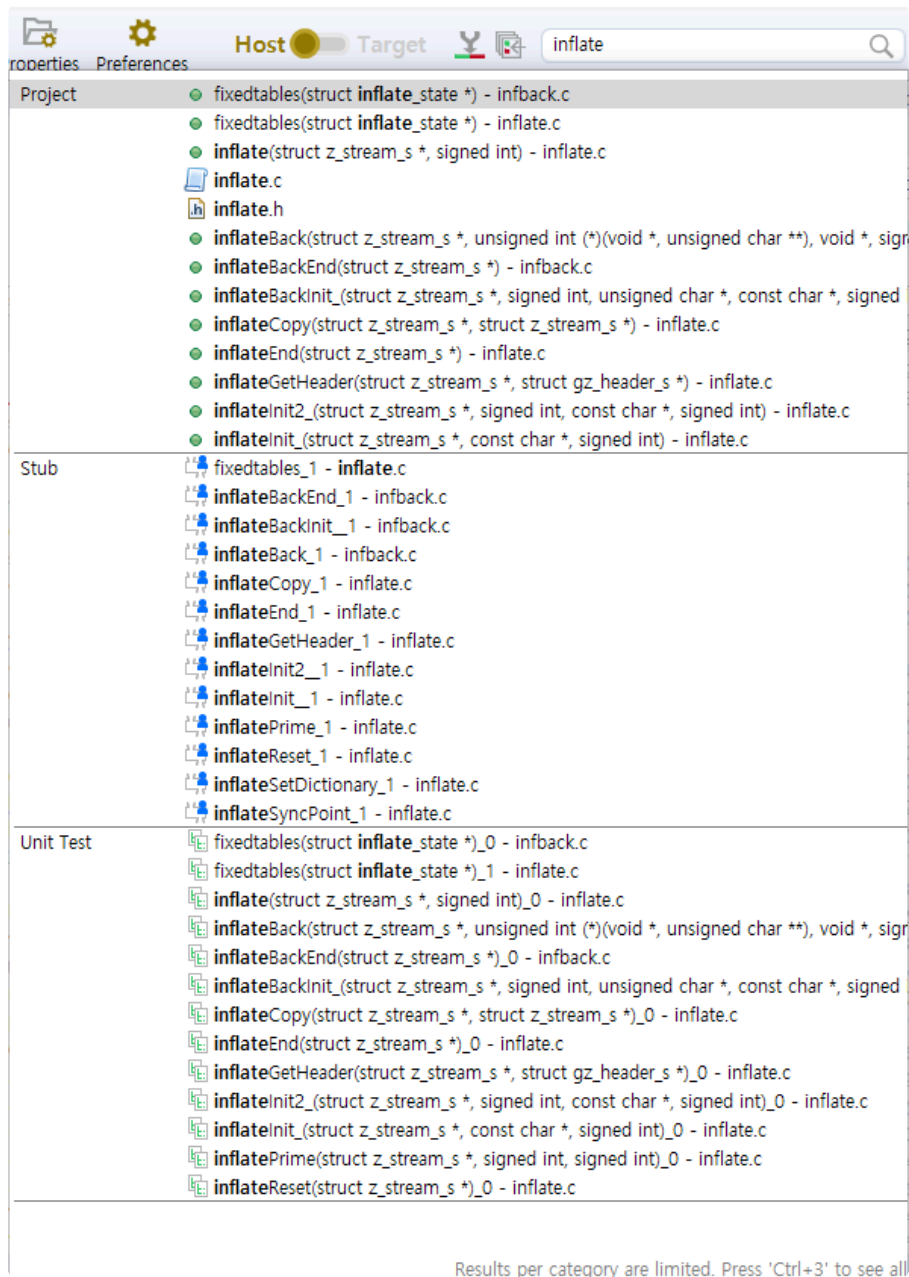
## Global Search

The Global search allows you to search New, View, Editor, Menu, 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 function.



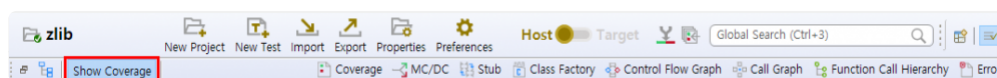
The project resource (source file, function, etc.), test, stub and class factory can be searched by selecting the analyzed project.





## Show Coverage

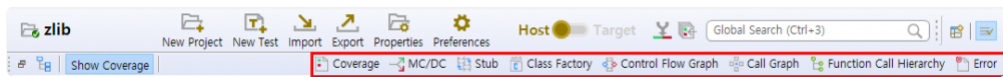
It allows you to set the covered area in the source code editor is marked or not.



## Open/Close the test-related view menu

You can open and close the [Coverage] view, [MC/DC] view, [Stub] view, [Class Factory] view, [Control Flow Graph] view, [Call Graph] view, [Function Call Hierarchy] view, and [Error] view.



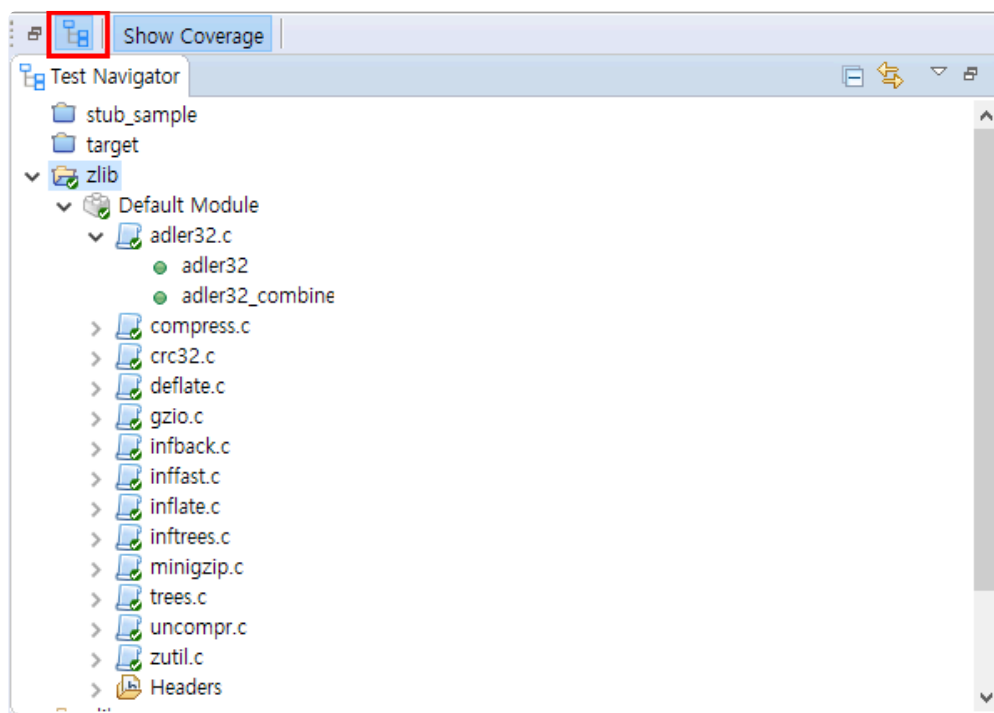




## 14.2. Test Navigator View

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 area.





### Icon

Icon	Description(*: created after analysis)
	Open project
	Closed project
	Module
	Source file (translation unit)
	*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.

## Drag and Drop from Windows Explorer

- **Creates a project by Visual Studio project**  
From Windows Explorer, drag and drop the Visual Studio file (.dsw, .sln, .vcxproj, .vcproj) into Test Navigator view.
- **Imports the source file from the file system**  
From Windows Explorer, drag and drop the source file into the target project or module in Test Navigator view.

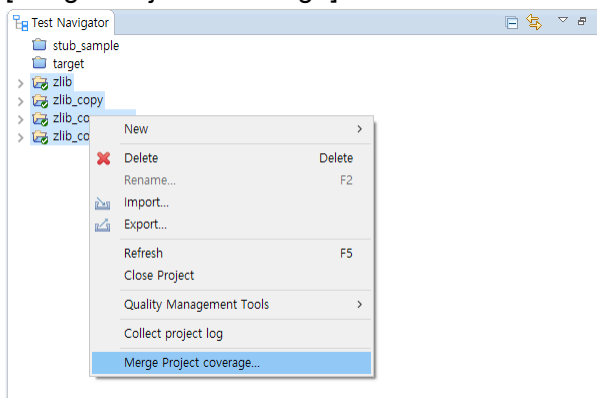


## 14.2.1. Merge Project Coverages

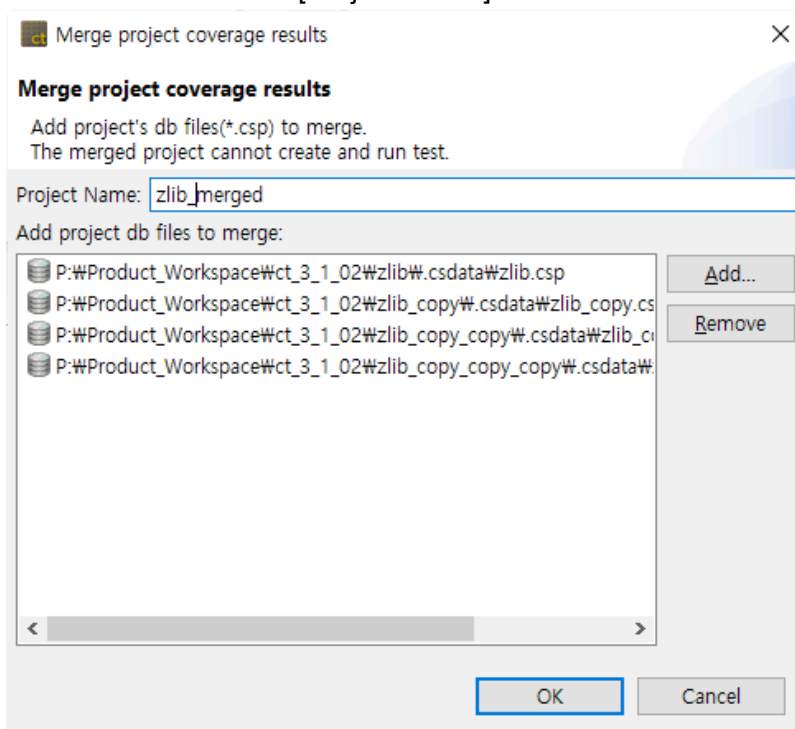
If a project size is too large to run the test at once, you can run the test by dividing it into multiple projects. [Merge Project Coverage] function runs the test by dividing one project into multiple projects and then shows the overall coverage results by merging the result of the test for these projects.

### 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].

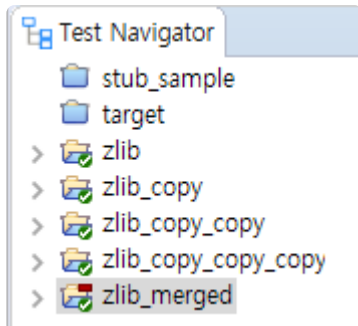


- For the projects not shown in the [Test Navigator] view, you can carry out the function for



merging the project results by using the Add the function of the external project.

- 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.



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

Target Function	Statement	Branch	MC/DC	Function Call	Function
1 _tr_align(struct internal_state *)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
2 _tr_flush_block(struct internal_state *, ...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
3 _tr_init(struct internal_state *)	0.00% (0...)	N/A	N/A	0.00% (0...)	N
4 _tr_stored_block(struct internal_state ...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
5 _tr_tally(struct internal_state *, unsig...	0.00% (0...)	0.00% (0...)	0.00% (0...)	N/A	N
6 adler32(unsigned long, const unsig...	97.08% (...)	83.33% (...)	66.66% (...)	N/A	Y
7 adler32_combine(unsigned long, un...	100.00% (...)	100.00% (...)	100.00% (...)	N/A	Y
8 bi_flush(struct internal_state *)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N/A	N
9 bi_reverse(unsigned int, signed int)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N/A	N
10 bi_windup(struct internal_state *)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N/A	N
11 build_bi_tree(struct internal_state *)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
12 build_tree(struct internal_state *, stru...	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
<b>Total</b>	<b>2.95% (1...)</b>	<b>1.32% (2...)</b>	<b>0.95% (1...)</b>	<b>0.00% (0...)</b>	<b>1.75...</b>

Target Function	Statement	Branch	MC/DC	Function Call	Function
1 _tr_align(struct internal_state *)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
2 _tr_flush_block(struct internal_state *, ...)	52.38% (...)	43.75% (...)	0.00% (0...)	70.00% (...)	Y
3 _tr_init(struct internal_state *)	100.00% (...)	N/A	N/A	100.00% (...)	Y
4 _tr_stored_block(struct internal_state ...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
5 _tr_tally(struct internal_state *, unsig...	0.00% (0...)	0.00% (0...)	0.00% (0...)	N/A	N
6 adler32(unsigned long, const unsig...	11.65% (...)	20.83% (...)	8.33% (1...)	N/A	Y
7 adler32_combine(unsigned long, un...	0.00% (0...)	0.00% (0...)	0.00% (0...)	N/A	N
8 bi_flush(struct internal_state *)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N/A	N
9 bi_reverse(unsigned int, signed int)	100.00% (...)	50.00% (...)	0.00% (0...)	N/A	Y
10 bi_windup(struct internal_state *)	71.42% (...)	50.00% (...)	0.00% (0...)	N/A	Y
11 build_bi_tree(struct internal_state *)	100.00% (...)	75.00% (...)	50.00% (...)	100.00% (...)	Y
12 build_tree(struct internal_state *, stru...	100.00% (...)	75.00% (...)	50.00% (...)	100.00% (...)	Y
<b>Total</b>	<b>11.98% (...)</b>	<b>8.22% (1...)</b>	<b>3.19% (4...)</b>	<b>15.24% (...)</b>	<b>27.19...</b>

Target Function	Statement	Branch	MC/DC	Function Call	Function
13 check_header(struct gz_stream *)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
14 compress(unsigned char *, unsigned...	0.00% (0...)	N/A	N/A	0.00% (0...)	N
15 compress2(unsigned char *, unsigne...	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
16 compressBound(unsigned long)	0.00% (0...)	N/A	N/A	N/A	N
17 compress_block(struct internal_state ...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N/A	N
18 copy_block(struct internal_state *, ch...	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
19 crc32(unsigned long, const unsigne...	26.08% (...)	25.00% (...)	0.00% (0...)	50.00% (...)	Y
20 crc32_big(unsigned long, const unsi...	100.00% (...)	100.00% (...)	100.00% (...)	N/A	Y
21 crc32_combine(unsigned long, unsi...	100.00% (...)	100.00% (...)	100.00% (...)	100.00% (...)	Y
22 crc32_combine(unsigned long, unsi...	100.00% (...)	100.00% (...)	100.00% (...)	N/A	Y
23 deflate(struct z_stream_s *, signed int)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
24 deflateBound(struct z_stream_s *, un...	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
<b>Total</b>	<b>3.05% (1...)</b>	<b>1.93% (4...)</b>	<b>1.67% (2...)</b>	<b>2.13% (8...)</b>	<b>6.14...</b>

Target Function	Statement	Branch	MC/DC	Function Call	Function
70 gzread(void *, void *, unsigned int)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
71 gzrewind(void *)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
72 gzseek(void *, signed long, signed int)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
73 gzsetparams(void *, signed int, sign...	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
74 gztell(void *)	0.00% (0...)	N/A	N/A	0.00% (0...)	N
75 gzungetc(signed int, void *)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N/A	N
76 gzwrite(void *, const void *, unsigne...	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
77 inflate(struct z_stream_s *, signed int)	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
78 inflateBack(struct z_stream_s *, unsi...	2.95% (1...)	0.30% (1...)	0.00% (0...)	0.00% (0...)	Y
79 inflateBackEnd(struct z_stream_s *)	40.00% (...)	50.00% (...)	0.00% (0...)	0.00% (0...)	Y
80 inflateBackinit(struct z_stream_s *, s...	13.63% (...)	10.00% (...)	0.00% (0...)	0.00% (0...)	Y
81 inflateCopy(struct z_stream_s *, stru...	0.00% (0...)	0.00% (0...)	0.00% (0...)	0.00% (0...)	N
<b>Total</b>	<b>0.65% (2...)</b>	<b>0.14% (3...)</b>	<b>0.00% (0...)</b>	<b>0.00% (0...)</b>	<b>3.50...</b>



Coverage ⓘ

Host coverage information of 'zlib\_merged' project

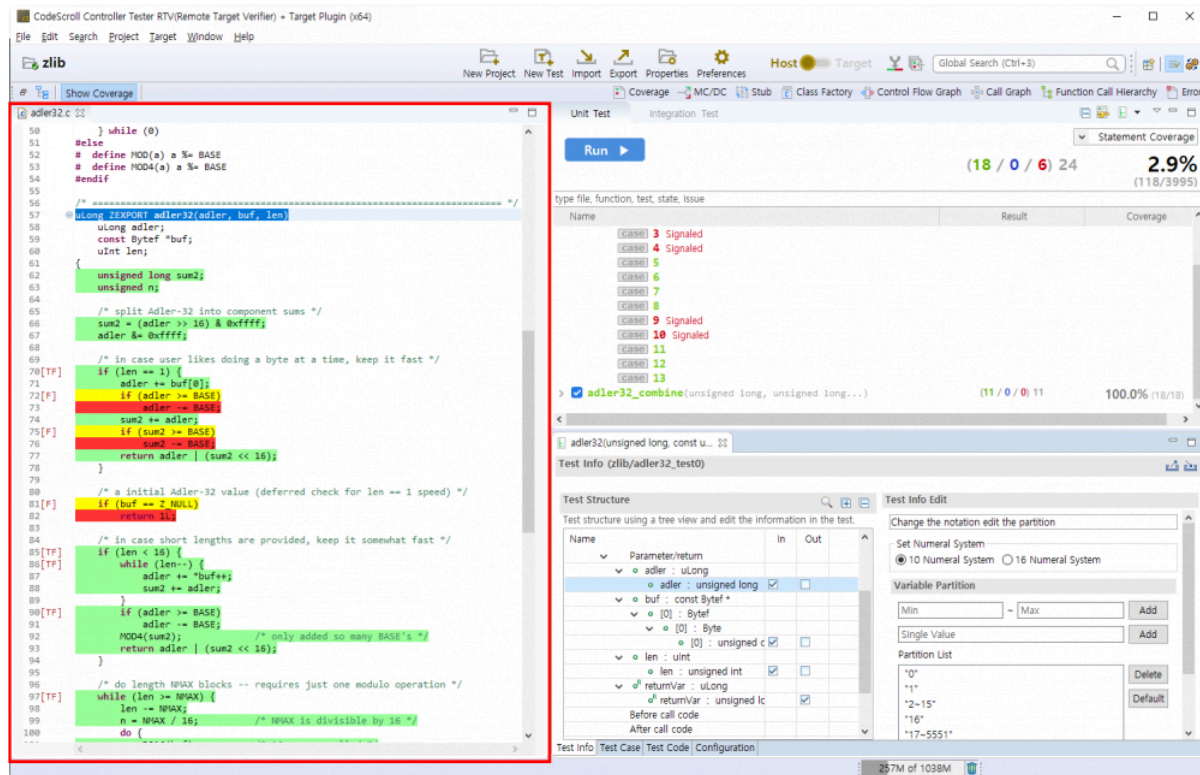
	Target Function	Statement	Branch	MC/DC	Function Call	Function
4	_tr_stored_block(struct internal_state ...	0.00% (0...	0.00% (0...	0.00% (0...	0.00% (0...	N
5	_tr_tally(struct internal_state *, unsig...	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
6	adler32(unsigned long, const unsign...	98.05% (...)	87.50% (...)	75.00% (...)	N/A	Y
7	adler32_combine(unsigned long, un...	100.00%...	100.00%...	100.00%...	N/A	Y
8	bi_flush(struct internal_state *)	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
9	bi_reverse(unsigned int, signed int)	100.00%...	50.00% (...)	0.00% (0...	N/A	Y
10	bi_windup(struct internal_state *)	71.42% (...)	50.00% (...)	0.00% (0...	N/A	Y
11	build_bl_tree(struct internal_state *)	100.00%...	75.00% (...)	50.00% (...)	100.00%...	Y
12	build_tree(struct internal_state *, stru...	100.00%...	75.00% (...)	50.00% (...)	100.00%...	Y
13	check_header(struct gz_stream *)	0.00% (0...	0.00% (0...	0.00% (0...	0.00% (0...	N
14	compress(unsigned char *, unsigned...	100.00%...	N/A	N/A	100.00%...	Y
15	compress2(unsigned char *, unsigne...	95.23% (...)	87.50% (...)	75.00% (...)	100.00%...	Y
16	compressBound(unsigned long)	100.00%...	N/A	N/A	N/A	Y
<b>Total</b>		<b>18.37% (...)</b>	<b>11.44% (...)</b>	<b>5.82% (7...</b>	<b>17.37% (...)</b>	<b>37.71...</b>

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



## 14.3. Source Code Editor section

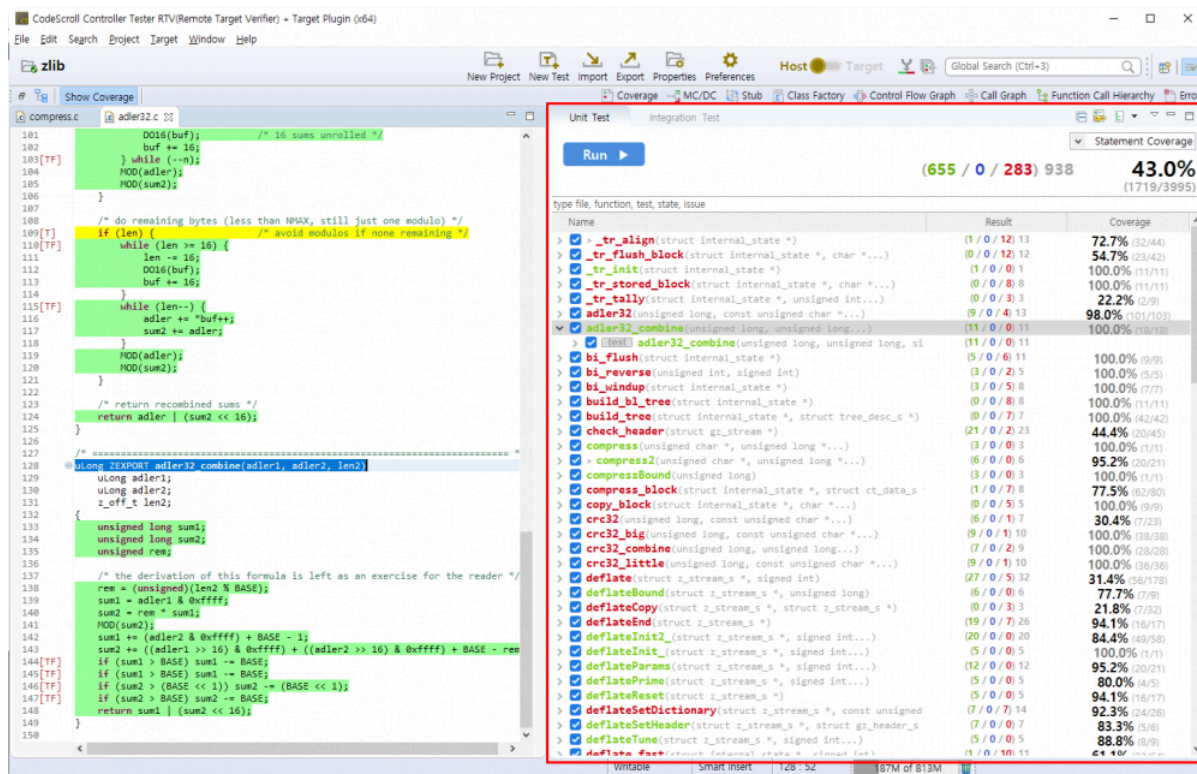
The source code editor section is a pre-reserved area, located on the left of the screen.





# 14.4. 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.



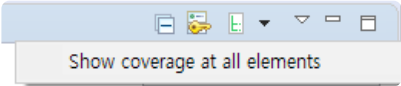
## Toolbar menu in the Unit Test view

Toolbar icon	Description
Collapse all	Hides all test view trees.
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.



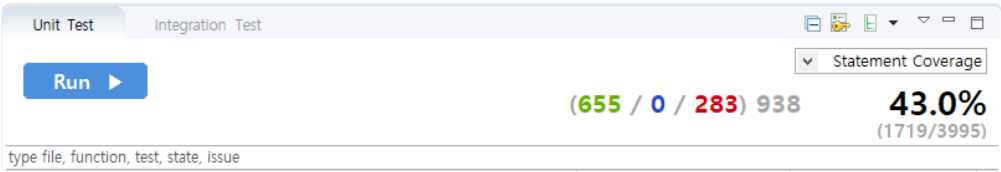
 Host/Target result different	Displays only the test that the host result and the target result are different.
--	--


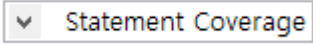
## Pull-down menu in the Unit Test view (▽)



Menu name	Description
Show coverage at all elements	Displays the coverage in all items. (function, test, test case)

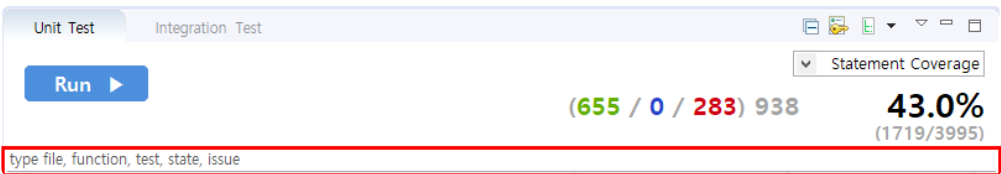
## Dashboard in Unit Test



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

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

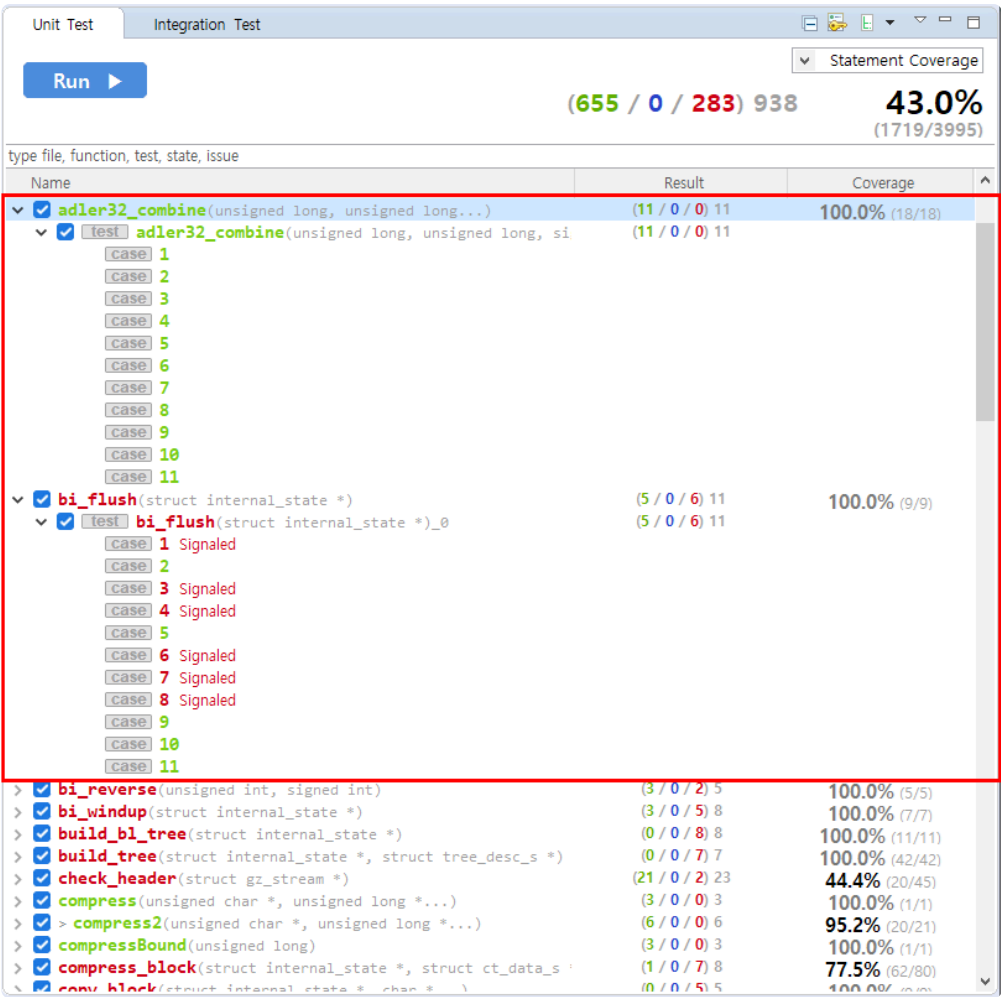


'Run not guaranteed' is a case where test code is generated, but there is no guarantee that it will run normally. This is the case when using a type that is difficult to specify as a parameter or return value.

## 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 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.



<b>Red</b>	Function/Test: If test cases with error exist under it. Test case: The result of execution is error.
<b>Orange</b>	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.
<b>Black</b>	Function/Test: All test cases under it have not been executed. Test case: Not been executed.

## Function node

The Function node provides the function name, the coverage and the test case execution results (success, failure, error and total).

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

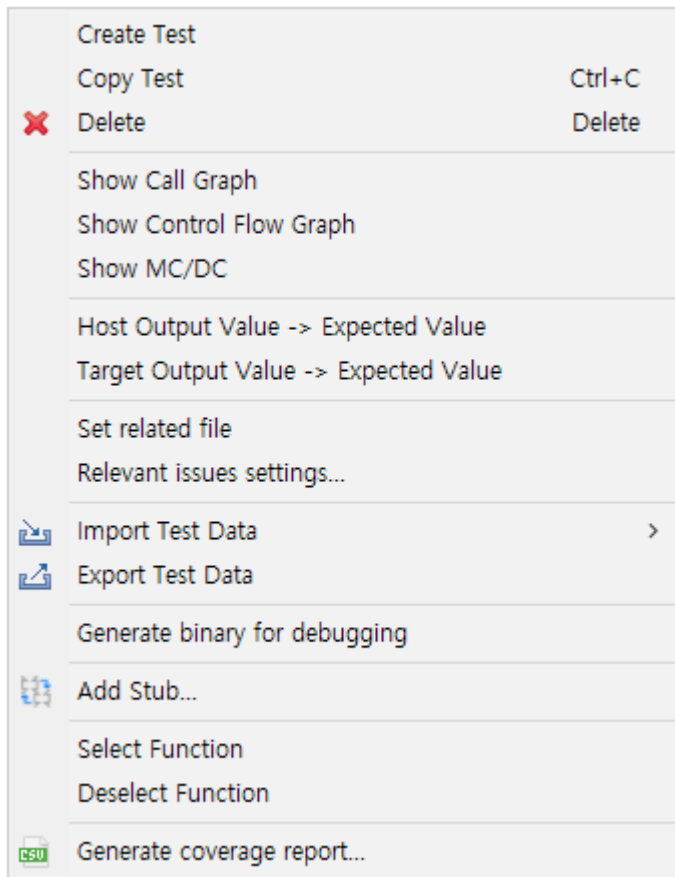
Double-click on a function node to check the location of the function in the Source Code Editor window.

The screenshot displays the 'Unit Test' tab in the CodeScroll Controller Tester. At the top, there is a 'Run' button and a summary of test results: (655 / 0 / 283) 938 with a 43.0% statement coverage (1719/3995). Below this is a table listing functions and their test case results. The 'adler32\_combine' function is highlighted with a red box, showing 100.0% coverage (18/18). Other functions like 'bi\_flush', 'bi\_reverse', 'bi\_windup', 'build\_bl\_tree', 'build\_tree', 'check\_header', 'compress', 'compress2', 'compressBound', 'compress\_block', and 'copy\_block' are also listed with their respective coverage percentages.

Name	Result	Coverage
adler32_combine(unsigned long, unsigned long...)	(11 / 0 / 0) 11	100.0% (18/18)
test   adler32_combine(unsigned long, unsigned long, si	(11 / 0 / 0) 11	
case 1		
case 2		
case 3		
case 4		
case 5		
case 6		
case 7		
case 8		
case 9		
case 10		
case 11		
bi_flush(struct internal_state *)	(5 / 0 / 6) 11	100.0% (9/9)
test   bi_flush(struct internal_state *)_0	(5 / 0 / 6) 11	
case 1	Signaled	
case 2		
case 3	Signaled	
case 4	Signaled	
case 5		
case 6	Signaled	
case 7	Signaled	
case 8	Signaled	
case 9		
case 10		
case 11		
bi_reverse(unsigned int, signed int)	(3 / 0 / 2) 5	100.0% (5/5)
bi_windup(struct internal_state *)	(3 / 0 / 5) 8	100.0% (7/7)
build_bl_tree(struct internal_state *)	(0 / 0 / 8) 8	100.0% (11/11)
build_tree(struct internal_state *, struct tree_desc_s *)	(0 / 0 / 7) 7	100.0% (42/42)
check_header(struct gz_stream *)	(21 / 0 / 2) 23	44.4% (20/45)
compress(unsigned char *, unsigned long *...)	(3 / 0 / 0) 3	100.0% (1/1)
compress2(unsigned char *, unsigned long *...)	(6 / 0 / 0) 6	95.2% (20/21)
compressBound(unsigned long)	(3 / 0 / 0) 3	100.0% (1/1)
compress_block(struct internal_state *, struct ct_data_s ...)	(1 / 0 / 7) 8	77.5% (62/80)
copy_block(struct internal_state *, char * ...)	(0 / 0 / 5) 5	100.0% (5/5)



## Context menu of the function node



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.
Target output value -> Expected value	Pastes the target output value to the expected value.
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.
Select Function	Selects the checkbox of all the functions selected by a mouse.



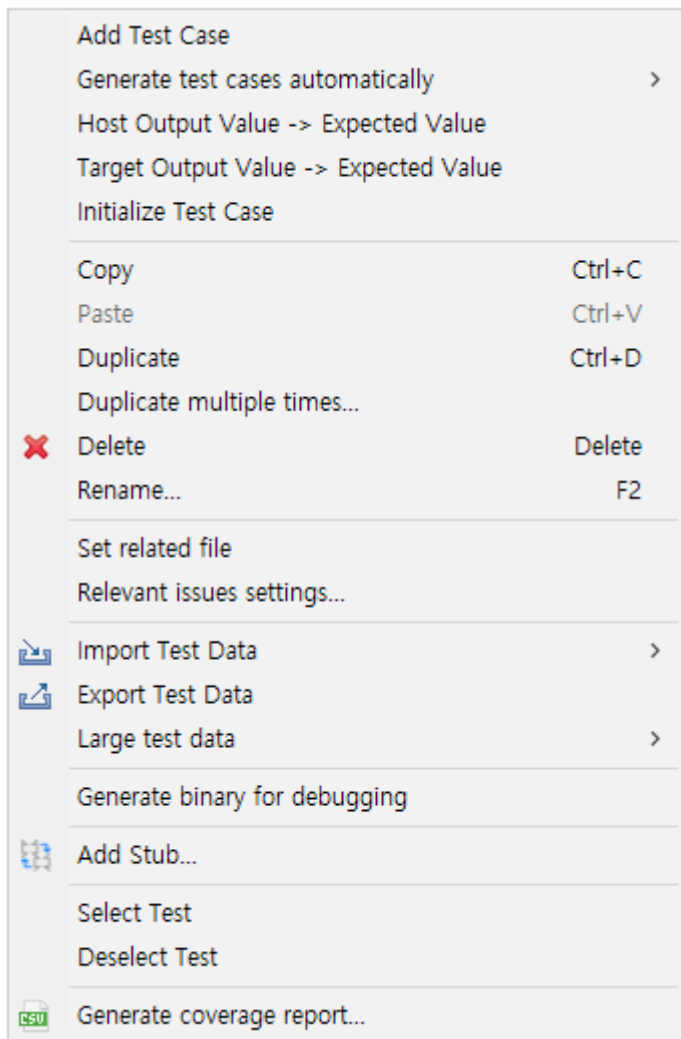
Deselect Function	Deselects the checkbox of all functions selected by a mouse.
Generate coverage report	Exports the selected test coverage to the selected path.

## Test node

The test node provides coverage, test case execution results(success, failure, error, total).

Double-click on a test node to open its [Test Editor](#).

### Context menu of the test node



Context menu	Description
Add Test Case	Adds the test case to the unit test selected.
Generate test cases automatically	Creates the test case in various ways.
Host output value -> Expected value	Pastes the host output value to the expected value.



Target output value -> Expected value	Pastes the target 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.
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.
Select tests	Checks all checkbox of the tests selected.
Deselect tests	Checks all checkbox of the tests selected.
Generate coverage report	Exports the selected test coverage to the selected path.

- Rule for exporting test data

When you try to export test data, if the test data for the same test as the test exported before exist in the export location, it creates the test data by numbering the file automatically.

The numbering convention is "test name\_#No.".

Ex) test\_1\_test0.csv, test\_1\_test0\_0.csv, test\_1\_test1.csv .....

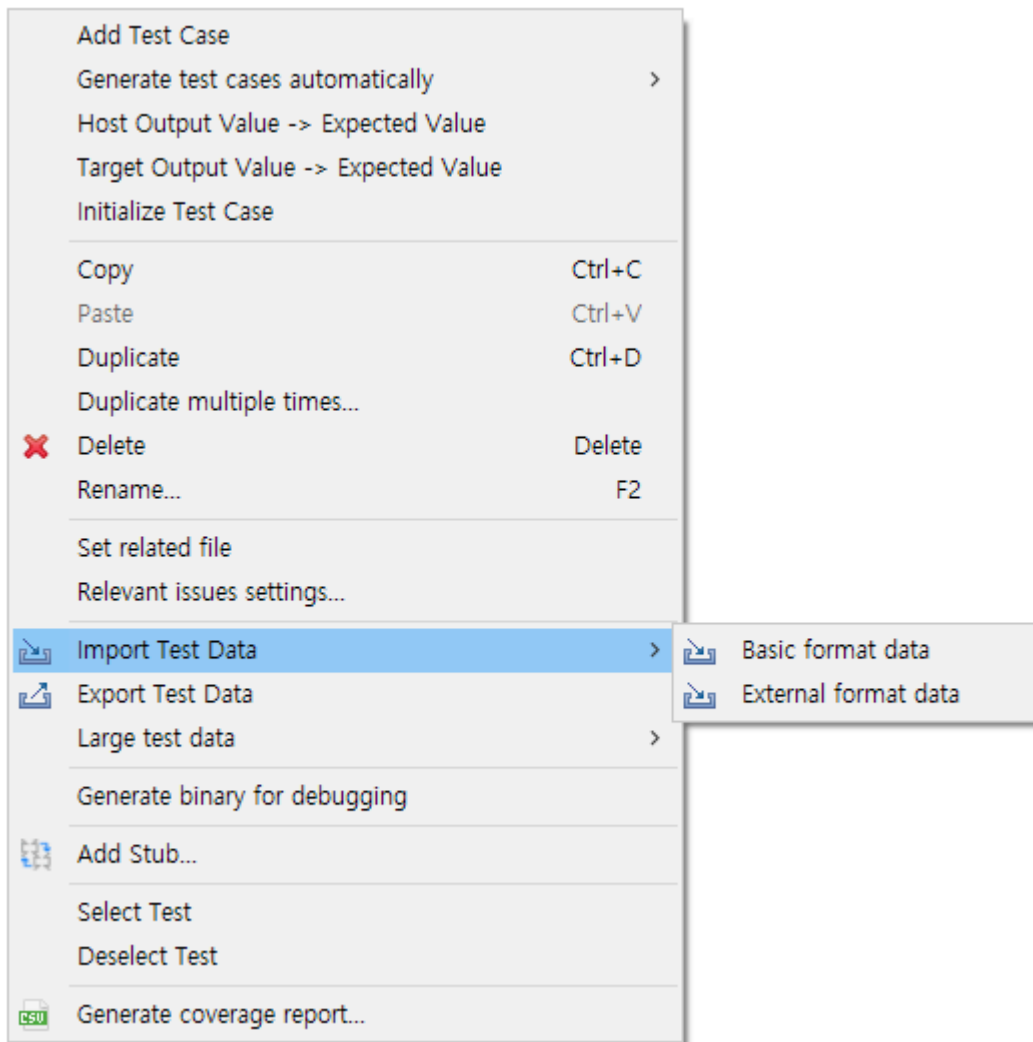
- Rule for importing test data

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

## Import test data

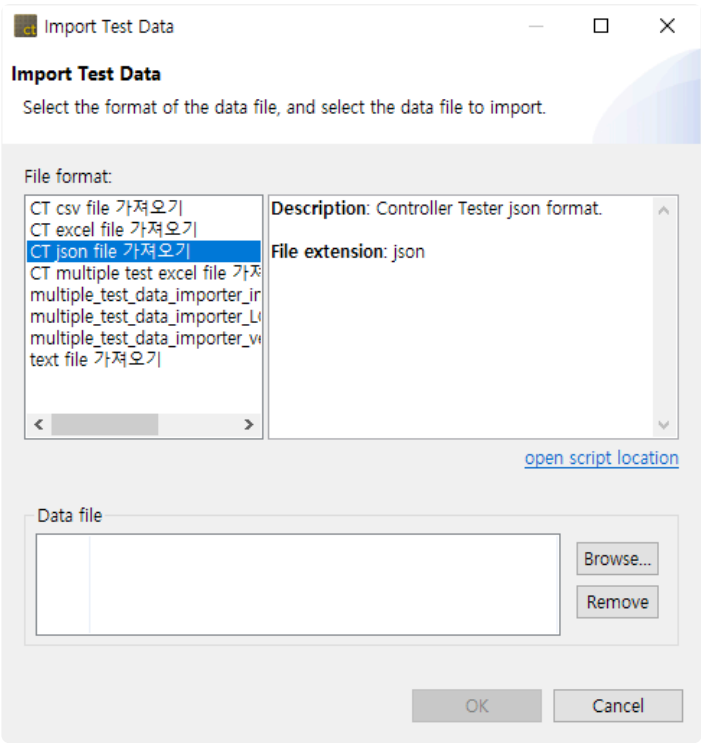
You can import the test data in various formats (csv, xlsx, txt, json).





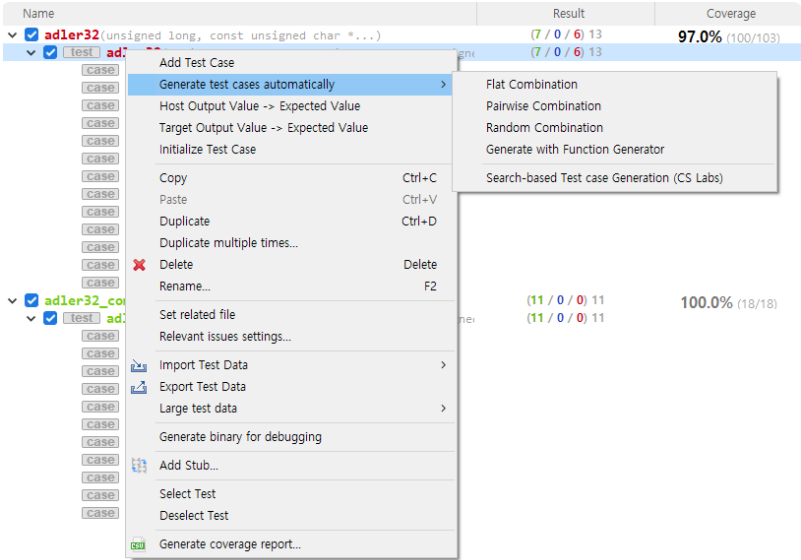
1. In the context menu of Unit Test view, click [Import Test Data] and select either [Basic format data] or [External format data].
2. If you select [Basic format data], the test data with the format exported from the Controller Tester is imported.
3. If you select [External format data], the test data 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).



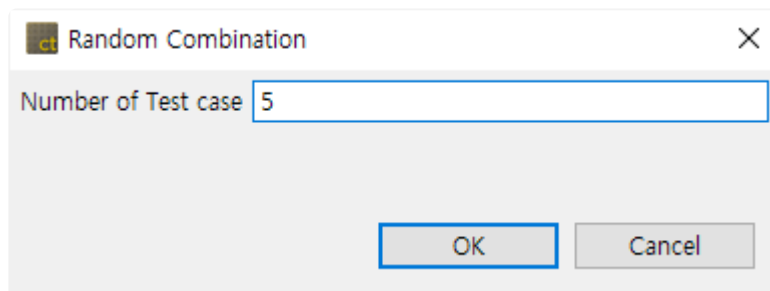
- 1. In the context menu of Unit Test or Integration test view, click [Generate test cases automatically] and select one among [Flat Combination], [Pairwise Combination], [Random Combination], [Generate with Function Generator] and [Search-based Test case Generation (CS Labs)].

Method	Description
--------	-------------



FLAT	Combines simply based on variables having the largest number of test data.
PAIR WISE	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 <u>AVM</u> , which is a local search algorithm. ※ Currently, only C language supported

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



- 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 [Project property settings], the function generator information can be set.



[illegible]

## Common settings

You can change the value in [Project] -> [Properties] with the setting value common to each function. In [Project property settings], the function generator information can be set.

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.)



## 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

## 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

## 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 Step	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.....



### 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

### 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

### SingleValue function

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

Setting	Description
Value	The value to be created

### None

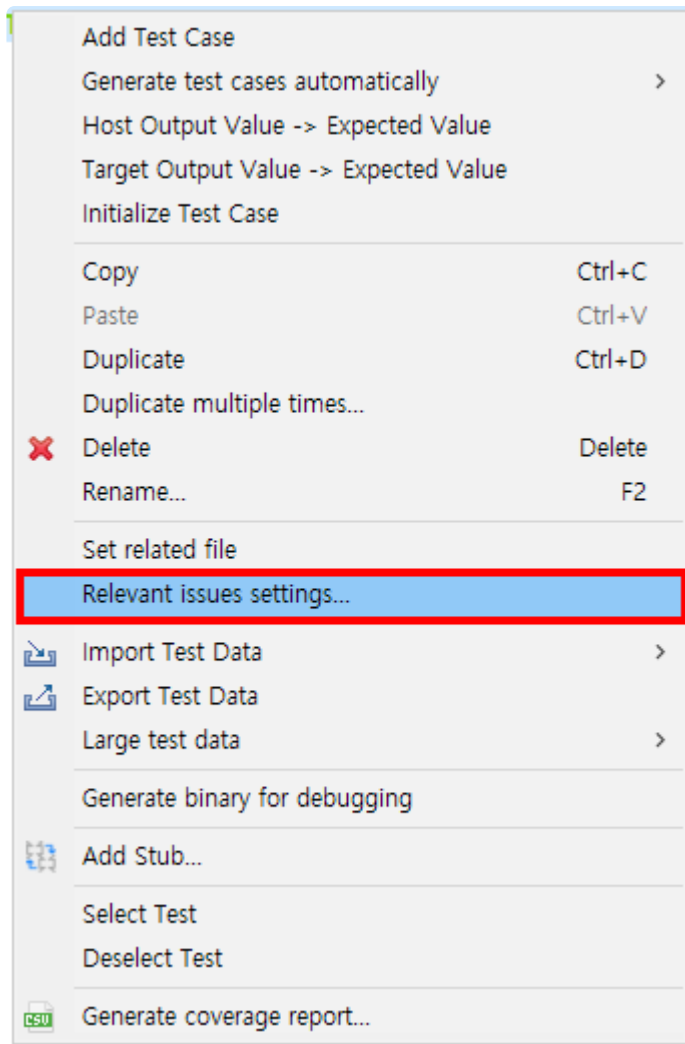
It does not create a function.

### Relevant issues settings

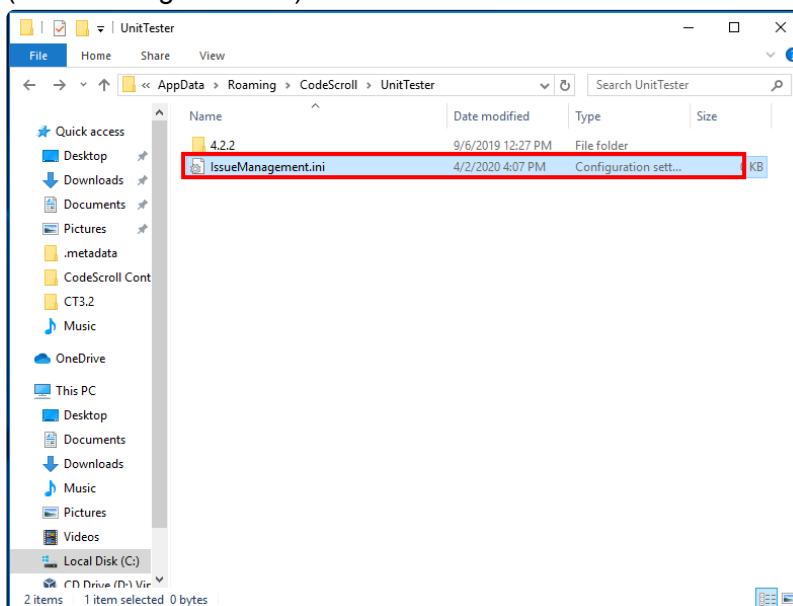
It associates the selected test with the issue in the issue management tool registered. The Controller Tester supports the following issue management tools: JIRA, Trac, Redmine, Mantis, Bugzilla.

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



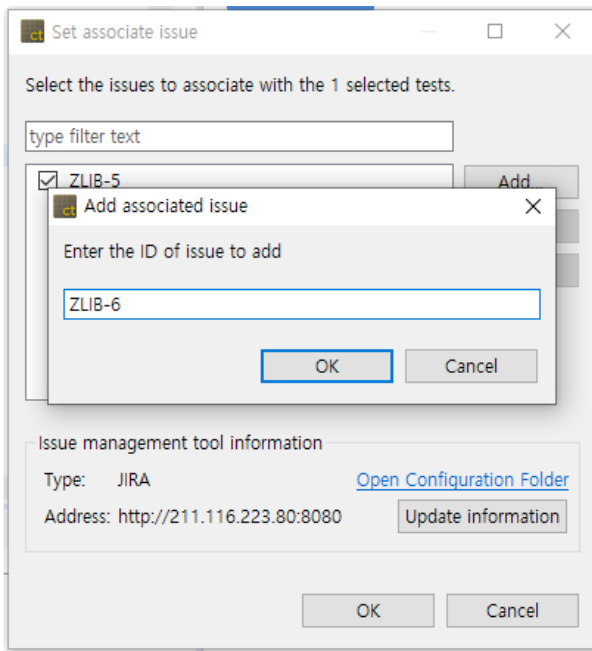


2. Enter the information of the issue management tool in the configuration file (IssueManagement.ini).

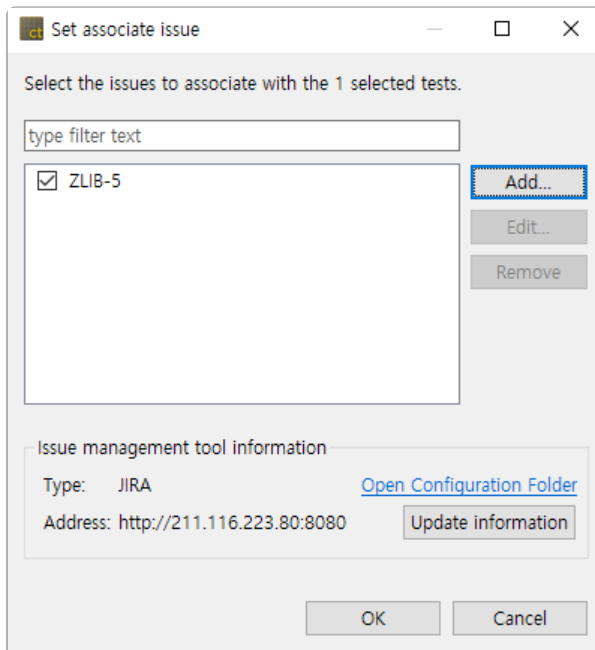


3. Add the issues to be associated with the test.





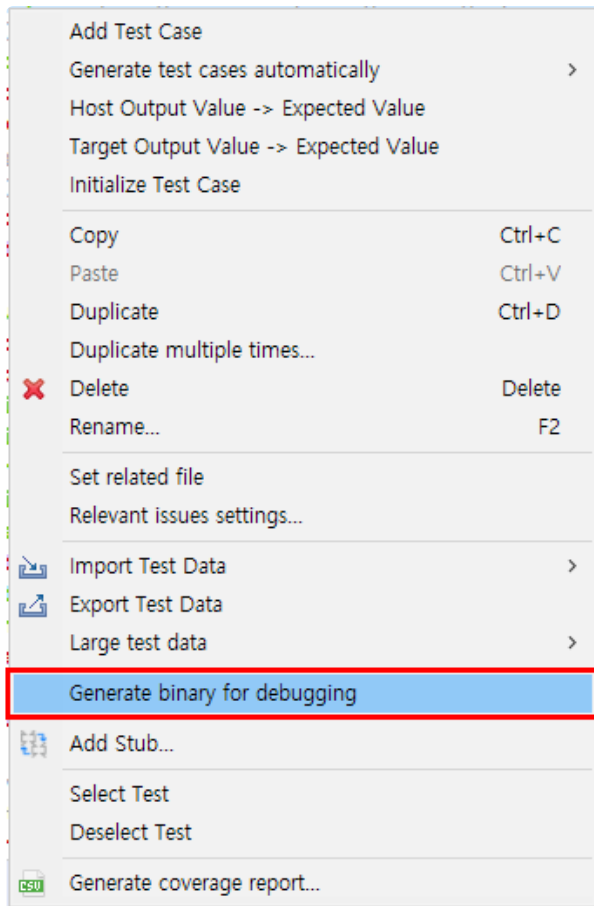
4. Check the checked issue list and click [OK].



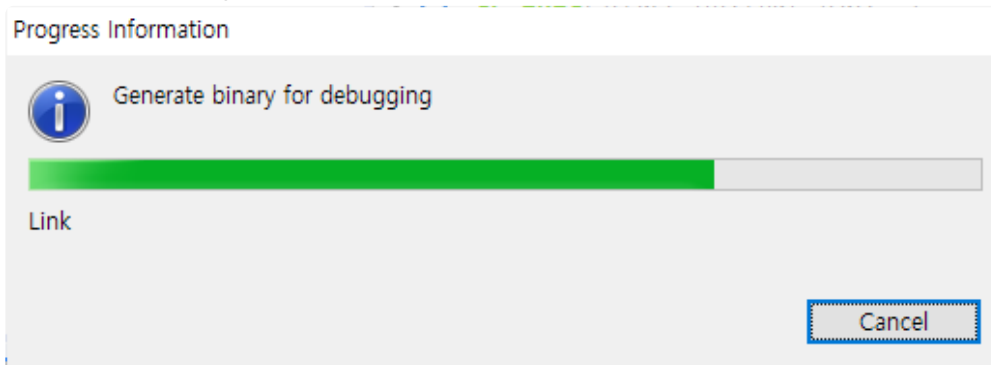
## Generate a binary for debugging

1. Right-click it in Unit Test or Integration Test view. From the context menu, select [Generate binary for debugging] menu to generate a binary for debugging.

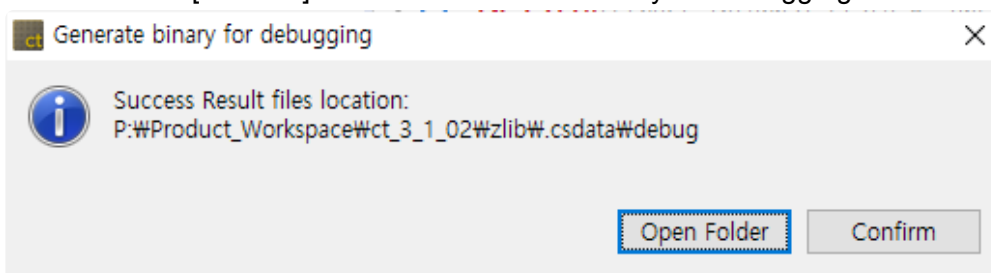




2. You can check the progress status information for the Generate binary for debugging. Also, a window indicating the location of the result file is opened.



3. Click [Open Folder] to move to the directory where the result for the Create a binary debugging is saved or click [Confirm] to exit the Generate binary for debugging.


























- If the project uses “Visual studio CL compiler”, it executes “vsjitdebugger” automatically.

### Results for the Generate binary for debugging

The list created when the user executes [Generate binary for debugging] function is as the figure below, and it creates the binary (testrun.exe) for debugging suitable to the user’s environment. The user can debug the testrun.exe file directly by using a debugging tool suitable to the toolchain of the target project.

The list of files generated when the [Generate binary for debugging] is performed is shown below.

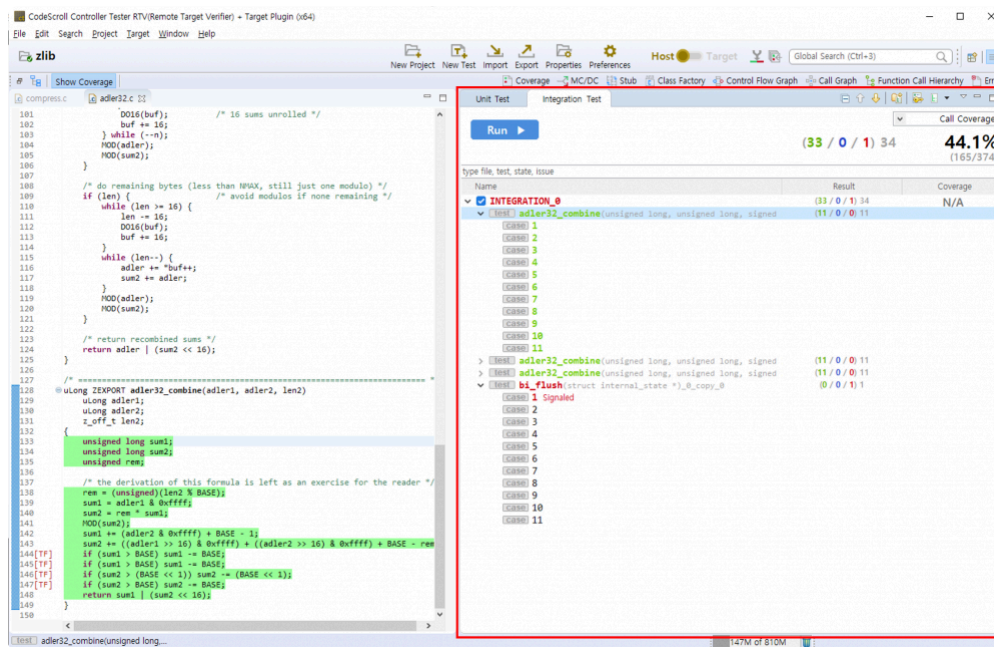
 compress_test0.csv	2020-03-31 .
 adler32_1.o	2020-03-31 .
 compress_2.o	2020-03-31 .
 crc32_3.o	2020-03-31 .
 cs_tfx.o	2020-03-31 .
 deflate_4.o	2020-03-31 .
 gzio_5.o	2020-03-31 .
 infback_6.o	2020-03-31 .
 inffast_7.o	2020-03-31 .
 inflate_8.o	2020-03-31 .
 inftrees_9.o	2020-03-31 .
 minigzip_10.o	2020-03-31 .
 trees_11.o	2020-03-31 .
 uncompr_12.o	2020-03-31 .
 zutil_13.o	2020-03-31 .
 cstrhook.lib	2020-02-18 .
 cstrlib.lib	2020-02-21 .
 testrun.pdb	2020-03-31 .
 testrun.exe	2020-03-31 .
 cstrhook.dll	2020-02-18 .
 cstrlib.dll	2020-02-21 .

- File list when using Visual Studio CL compiler
- testrun.exe: binary for debugging
- \*.csv: Input value for a test



## 14.5. 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.



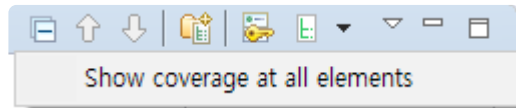
### Toolbar menu in the Integration test view

Toolbar icon	Description
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.



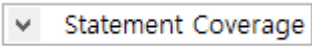
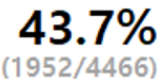
## Pull-down menu in Integration Test view (▽)



Menu	Description
Show Coverage at all elements	Displays the coverage in all items(Integration test, test, test case).

## Dashboard in Integration Test



Menu	Description
	Runs integration tests.
	Shows success, failure, error and total of test case.
	Selects the coverage type to be displayed in the [Integration Test] view. (statement, branch, MC/DC, function call)
	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

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

Structure of the Integration Test view




The Integration Test view presents the hierarchy structure of [Integration tes]-[Test]-[Test case].



Item icons in the Integration Test view

Item icon	Description
-----------	-------------



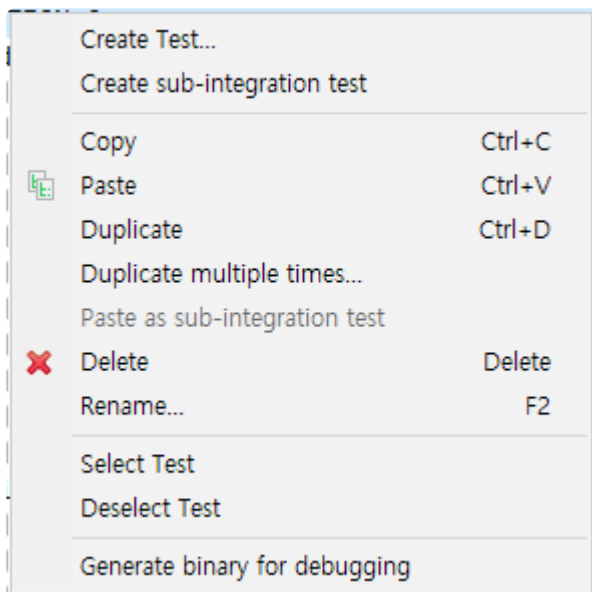
None	Integration Test
	Test
	Global variable test
	Test case, Test case group

## 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 Integration test view



Context menu	Description
--------------	-------------

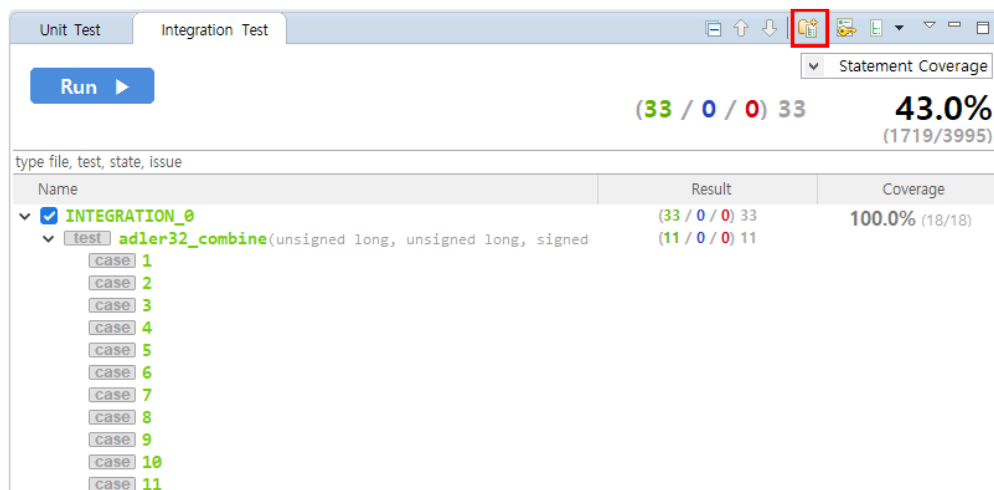


Create Test	Creates the dialog for creating a test
Create sub-integration test	Creates a sub-integration test
Copy	Copies the test and test case
Paste	Pastes the test and test case
Duplicate	Duplicates the test and test case
Duplicate multiple times	Duplicates tests and test cases as many as the number 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
Generate coverage report	Exports the selected test coverage to the selected path

## Create Integration test and Rename

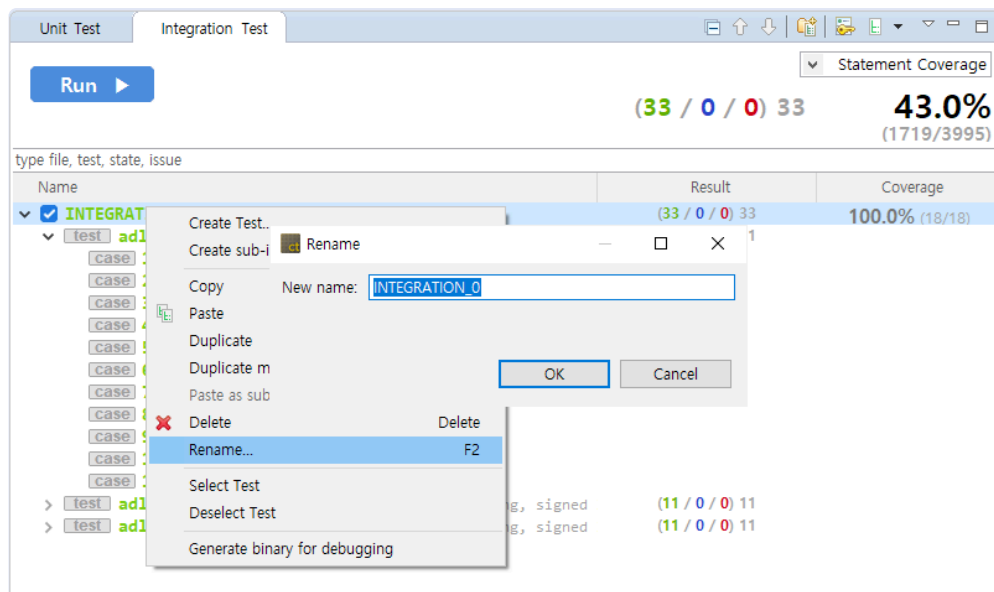
The integration test can be created in the toolbar menu. The name of the integration test is assigned automatically and can be changed by using [Rename] context menu.

- Create



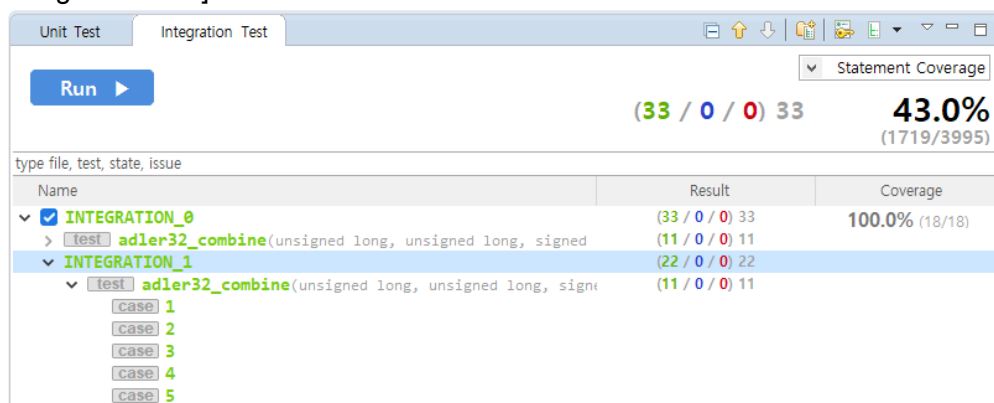
- Rename





## 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 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.

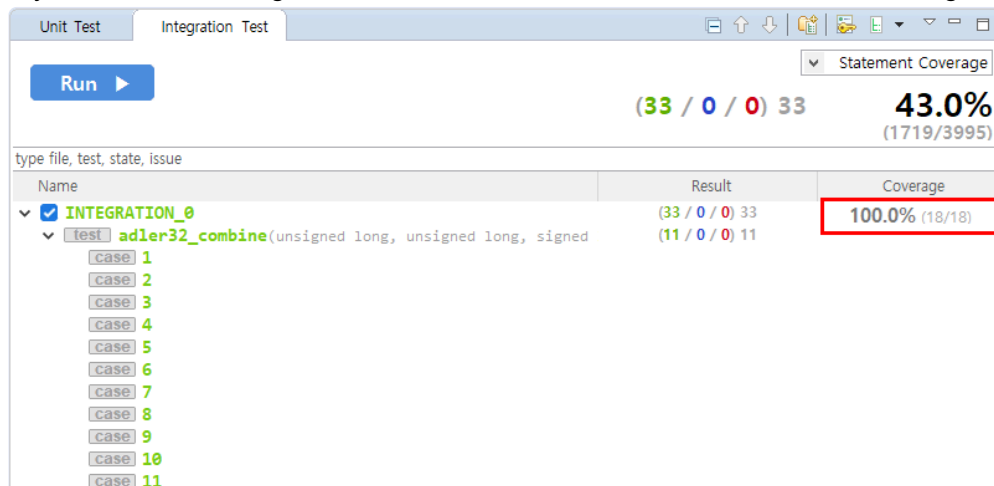


1. You can select a test or test case and change it by moving with a mouse.
2. You can select a test or test case and change it by selecting either “Move test up” or “Move test down” in the toolbar.



## The coverage result of Integration Test

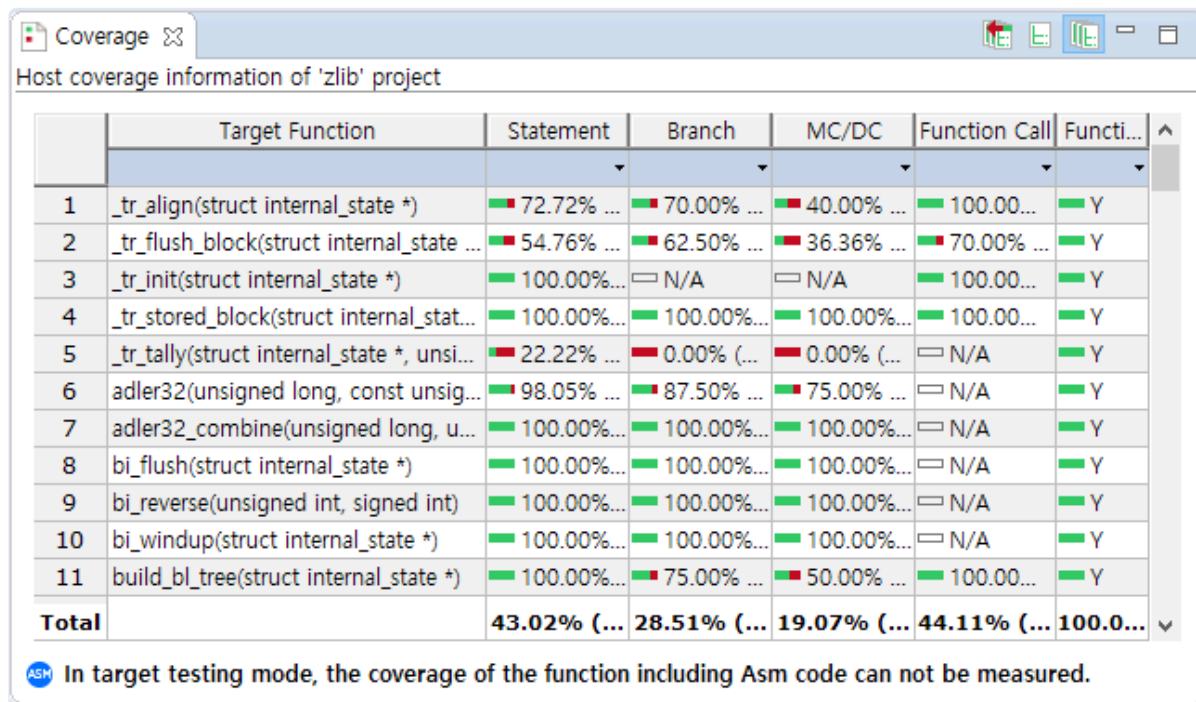
If you select the integration test, the result can be checked in the coverage view.





## 14.6. 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	Function
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
<b>Total</b>		<b>43.02% (...)</b>	<b>28.51% (...)</b>	<b>19.07% (...)</b>	<b>44.11% (...)</b>	<b>100.00%</b>

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

## 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 Coverage view

Icon	Description
	Function changed
	Function containing Asm code



## Toolbar menu in Coverage view

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

























































## Show Coverage


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.

<b>Total</b>	<b>43.02% (171...</b>	<b>28.51% (603/2115)</b>	<b>19.07% (239/1253)</b>	<b>44.11% (165/...</b>	<b>100.00% (114/...</b>
--------------	-----------------------	--------------------------	--------------------------	------------------------	-------------------------

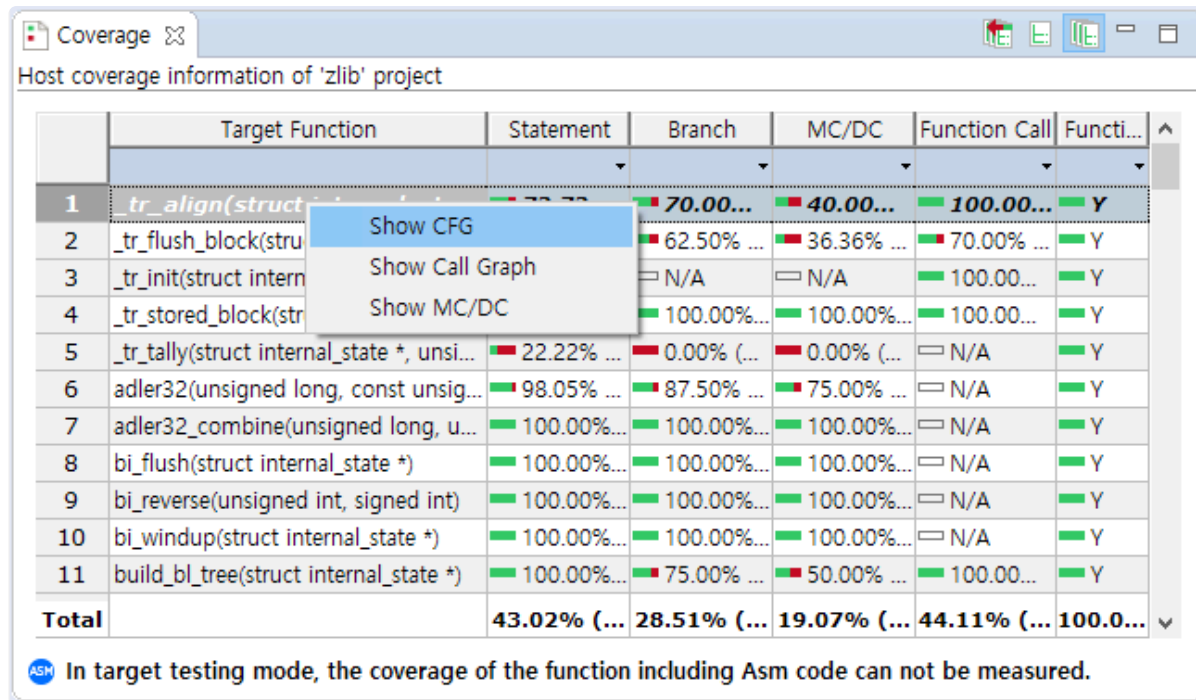
In the Coverage view, the coverage for each function is displayed for the items selected in the Unit or Integration Test view.

Coverage 						
Host coverage information of 'zlib' project						
	Target Function	Statement	Branch	MC/DC	Function Call	Funci...
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
<b>Total</b>		<b>43.02% (...</b>	<b>28.51% (...</b>	<b>19.07% (...</b>	<b>44.11% (...</b>	<b>100.0...</b>

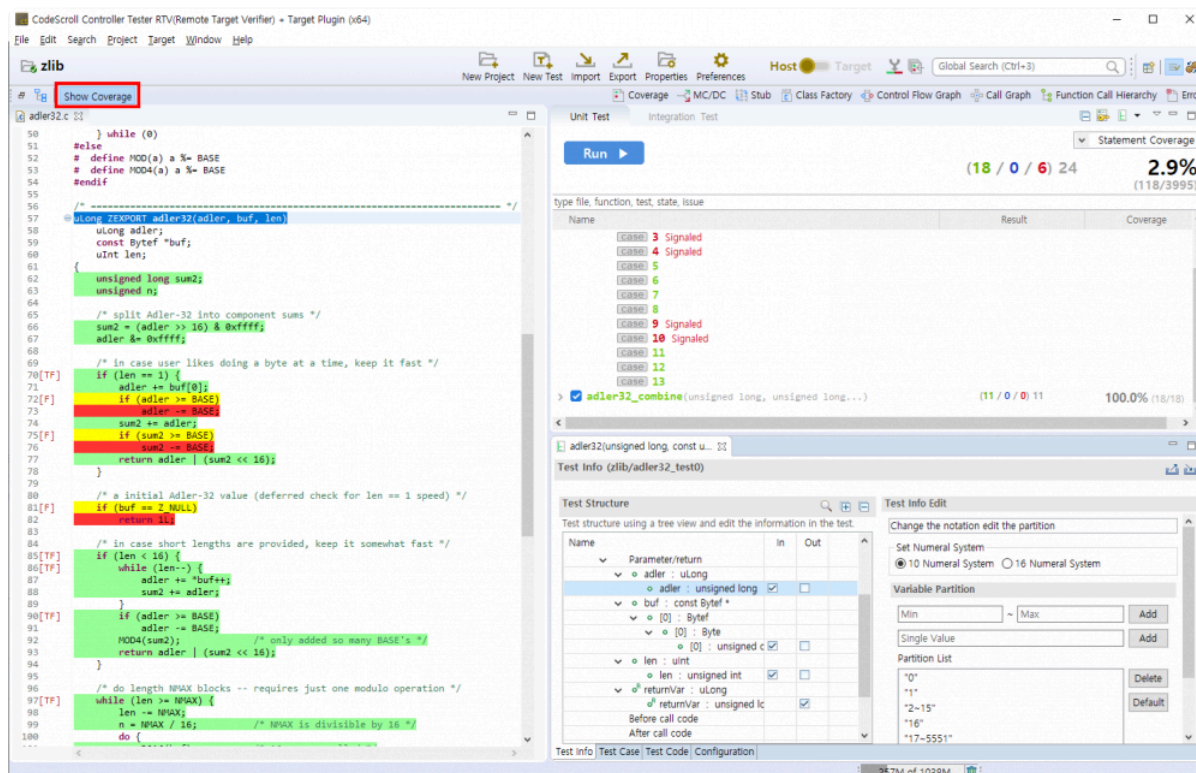
 In target testing mode, the coverage of the function including Asm code can not be measured.

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.





If you enable [Show Coverage] icon in the main toolbar and click the wanted 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 executed and all branches are covered, the red color indicates that the code is not executed, and yellow color indicates that the portion of the branch is covered only. You can check T/F of the branch statement covered in vertical column left to the editor.





## 14.7. 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

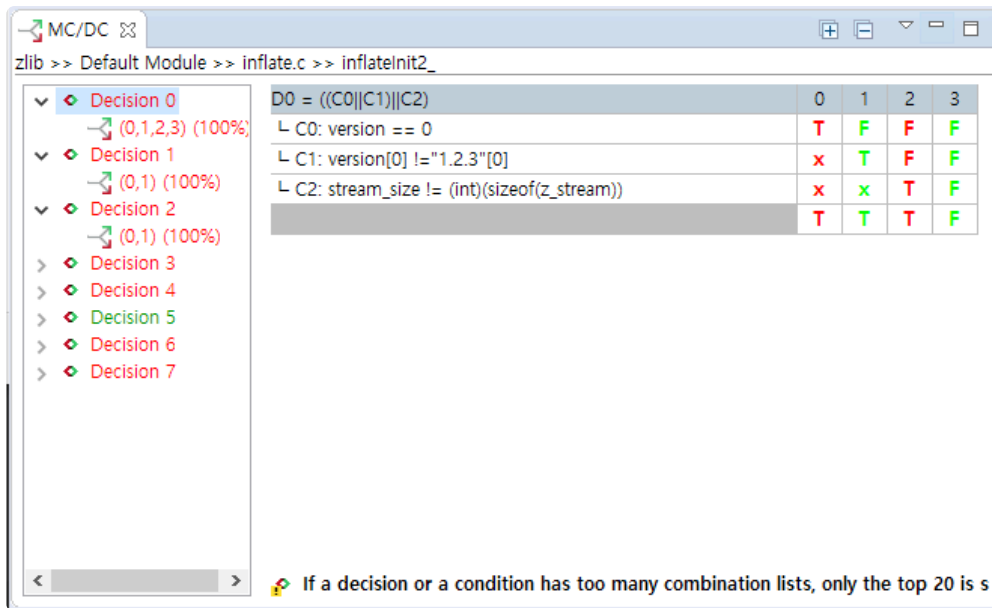
The following figure is a view that can identify easily the combinations which need to be covered to achieve the target coverage.

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.

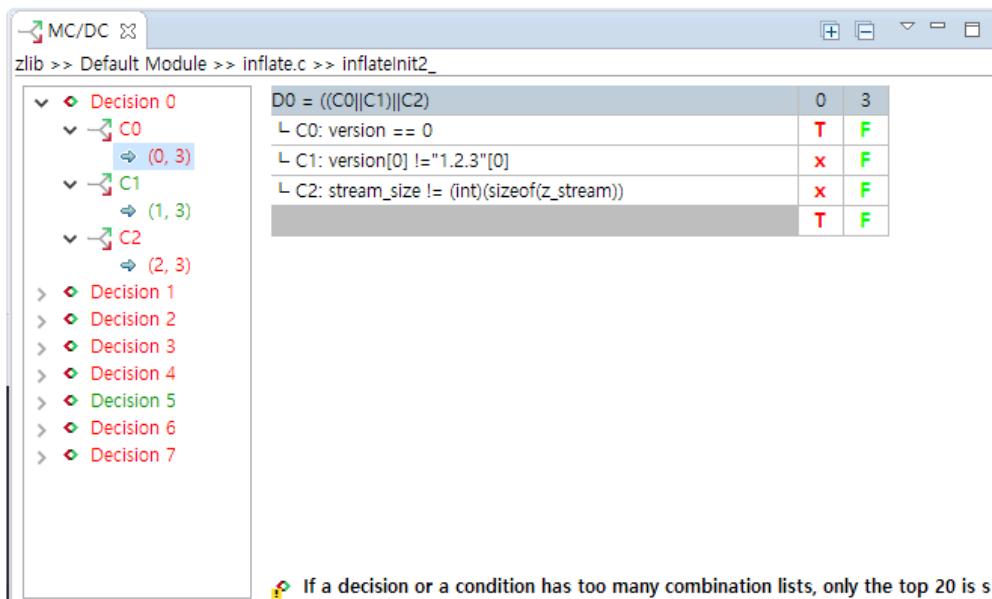
In the Preferences page, you can change the target coverage rate and the screen view mode. And you can select all for the truth table and use 'Ctrl + C' or Copy clipboard function from the context menu.





- 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.



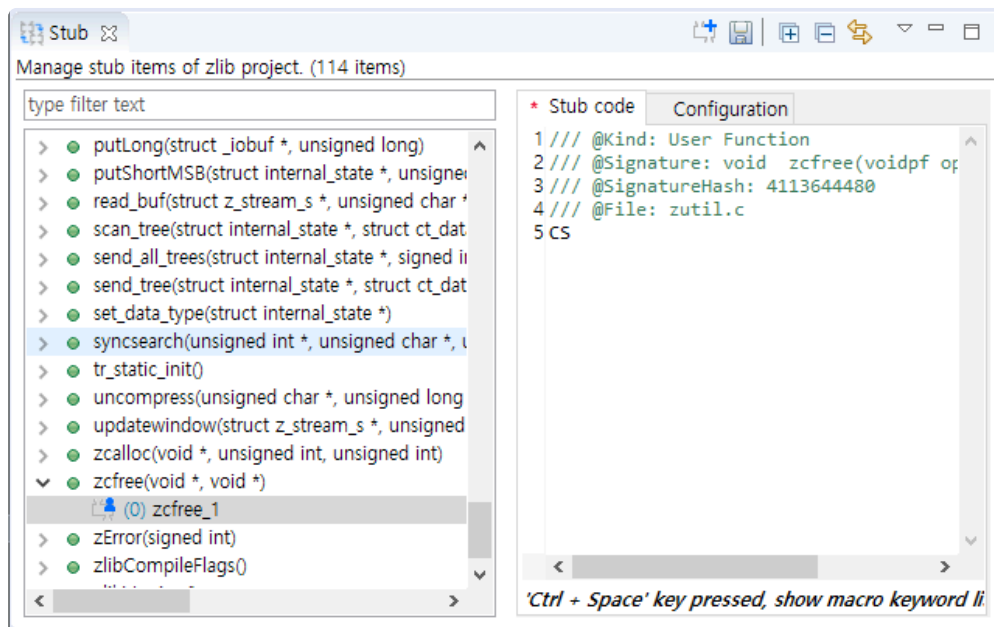


## 14.8. Stub View




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 definition
 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

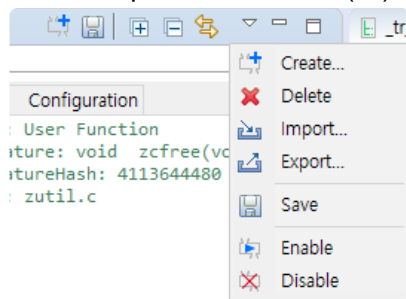
## Menu

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

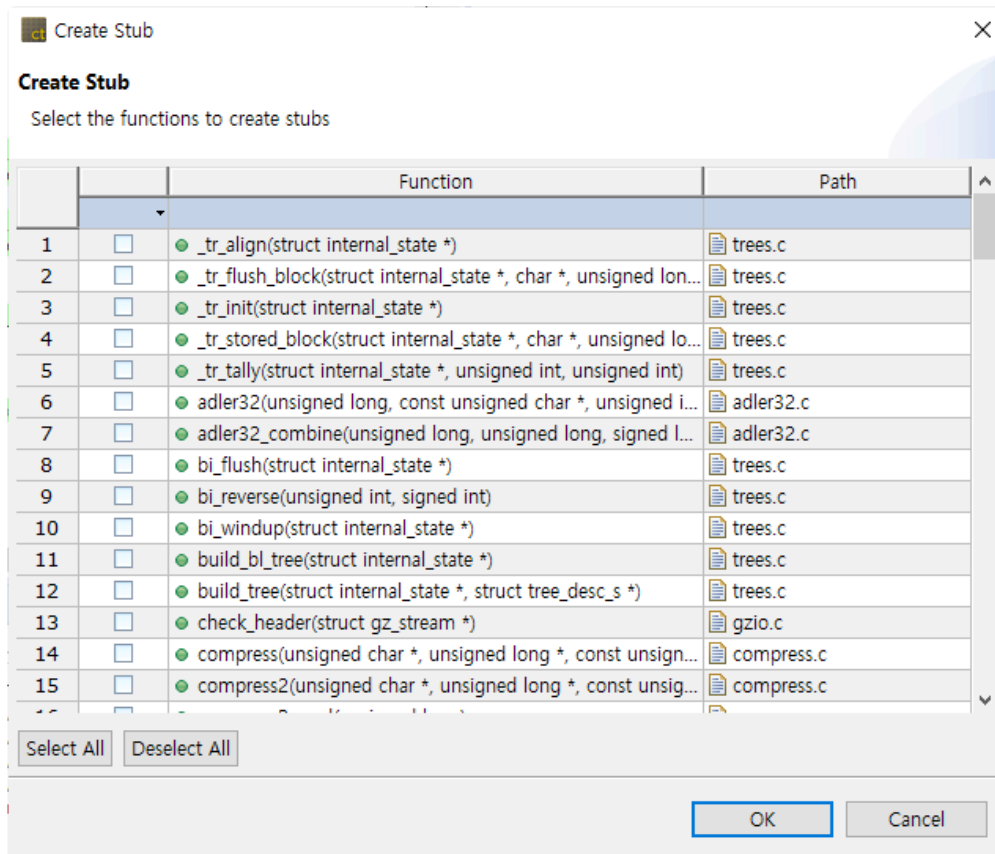
- Create stubs in Stub view

- Click the pull-down menu (▽) in Stub view and click [Create] menu.

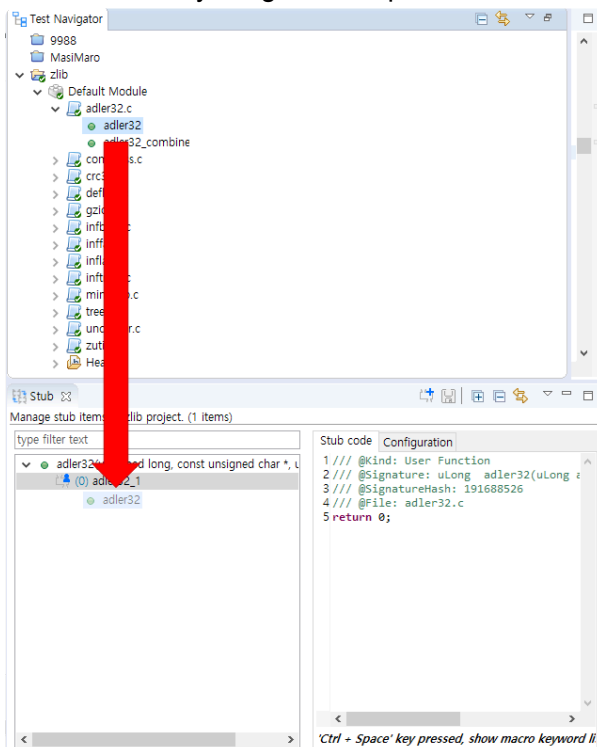


- Check the functions to be created as stubs and select [OK].





- Create stubs by drag-and-drop in the Test Navigator



- Create stubs by drag-and-drop in the Test Navigator
  1. Select the functions to be created as stubs.

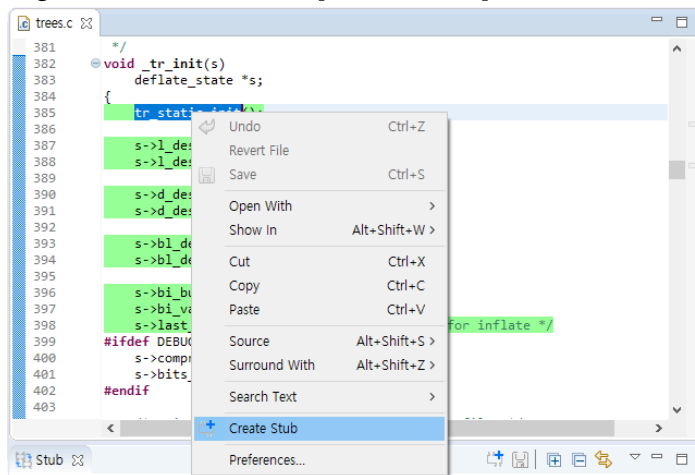


```

381
382 void _tr_init(s)
383     deflate_state *s;
384 {
385     tr_static_init();
386
387     s->l_desc.dyn_tree = s->dyn_ltree;
388     s->l_desc.stat_desc = &static_l_desc;
389
390     s->d_desc.dyn_tree = s->dyn_dtree;
391     s->d_desc.stat_desc = &static_d_desc;
392
393     s->bl_desc.dyn_tree = s->bl_tree;
394     s->bl_desc.stat_desc = &static_bl_desc;
395
396     s->bi_buf = 0;
397     s->bi_valid = 0;
398     s->last_eob_len = 8; /* enough lookahead for inflate */
399
400 #ifdef DEBUG
401     s->compressed_len = 0L;
402     s->bits_sent = 0L;
403 #endif

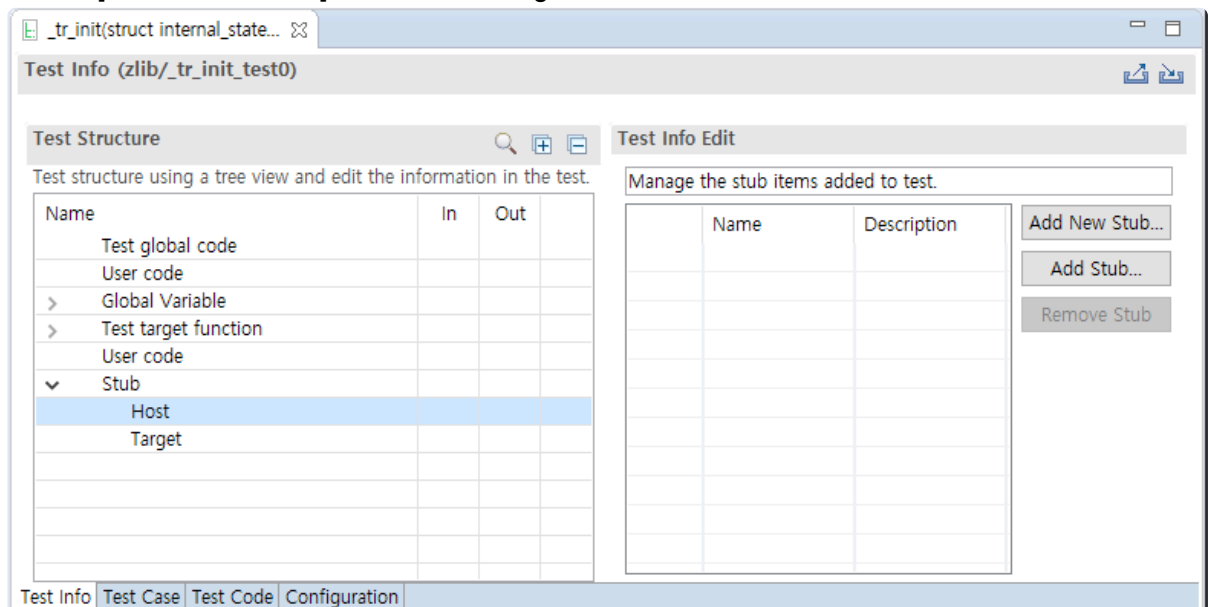
```

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



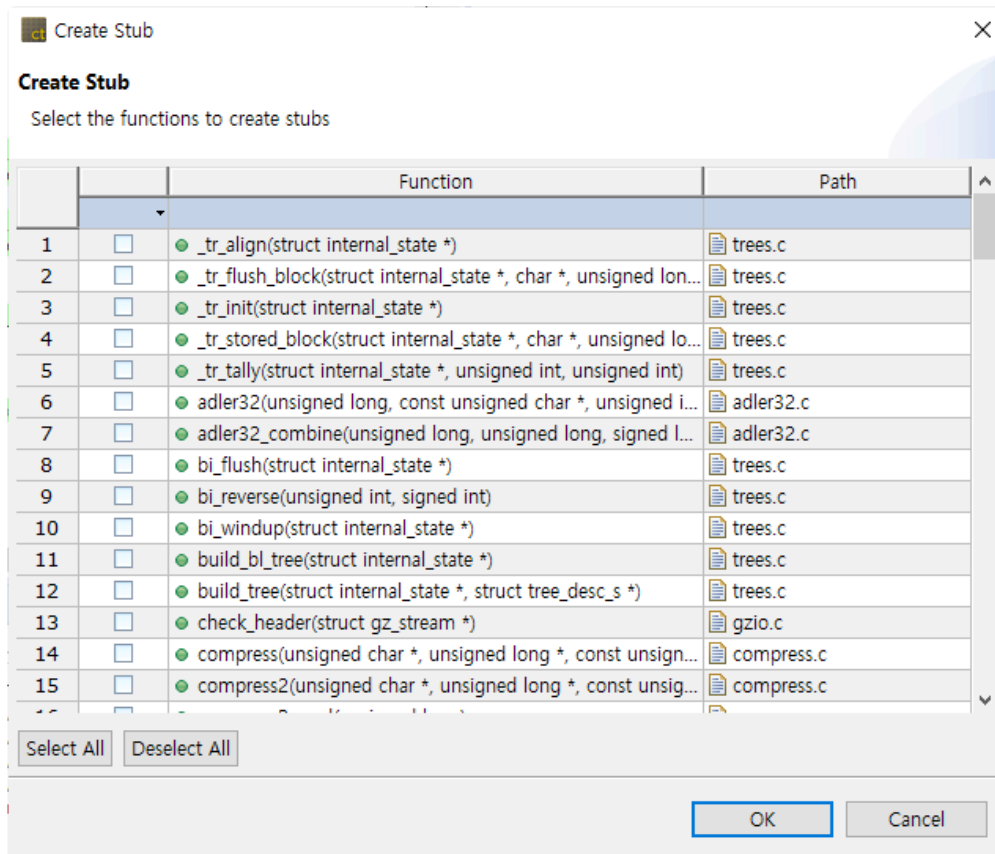
- Create and Link in Test Editor

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



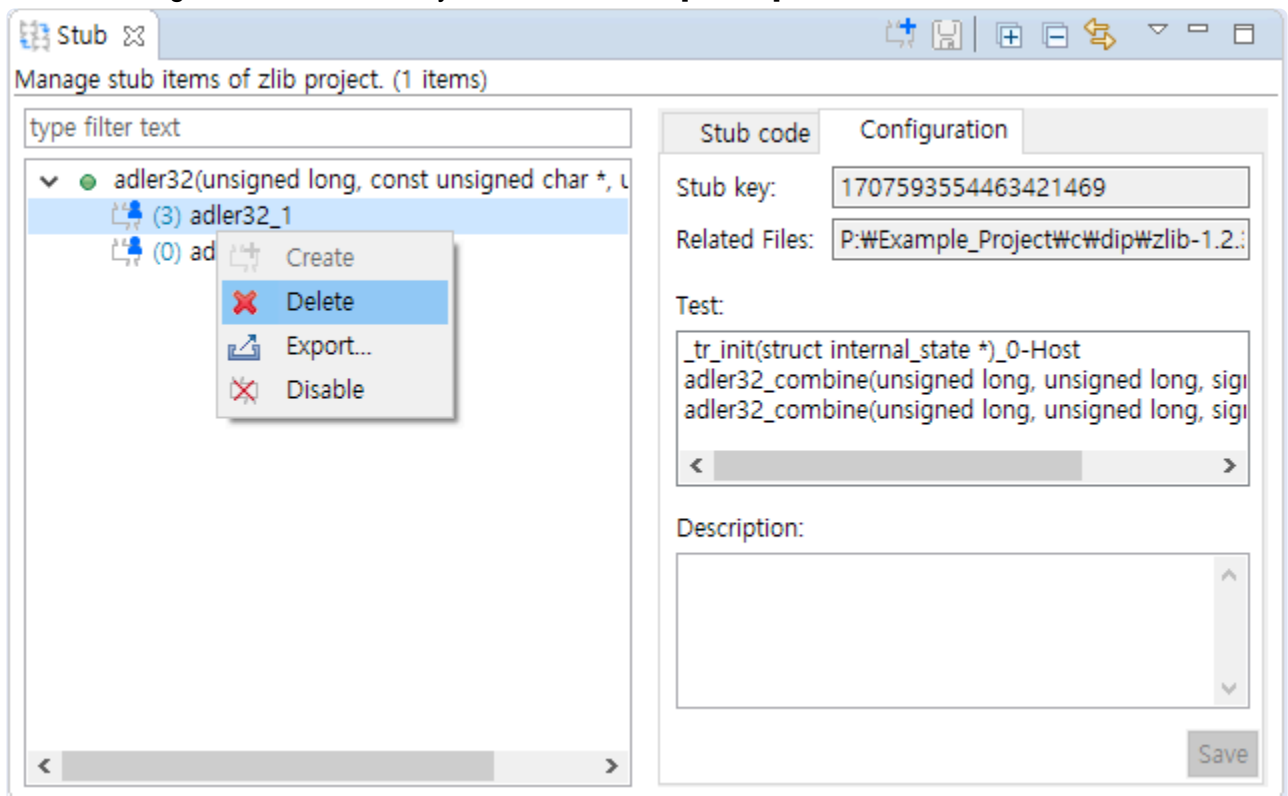
3. Check the functions to be created as stubs and select [OK].





## Delete stubs

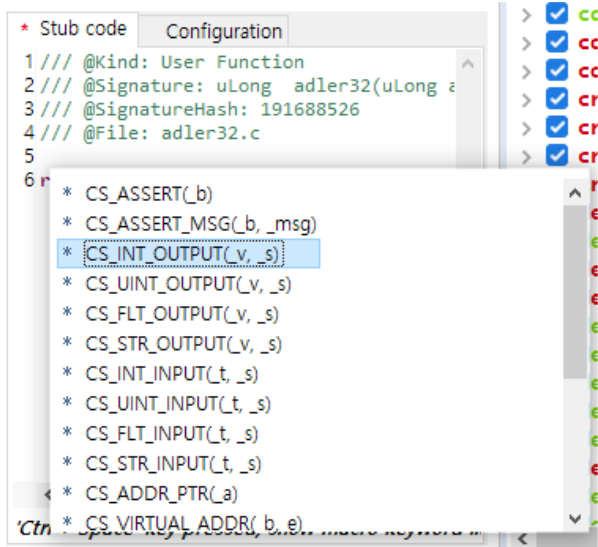
- Select and right-click unnecessary stubs and select [Delete] menu.





## 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.

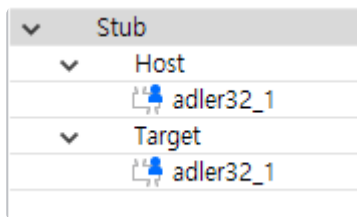


## Link between stubs and tests

Each stub can be linked to the host/target test.

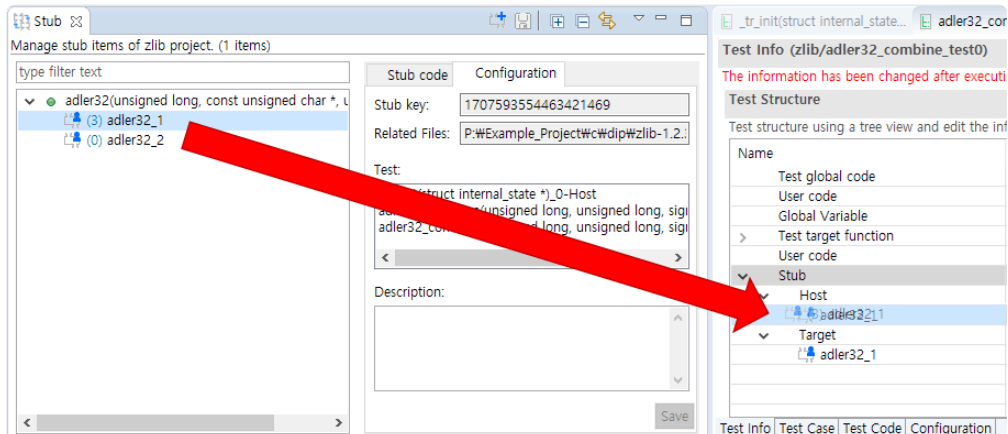
If you select "Stub" in the test structure editor, you can edit all the stubs included in the host and the target.





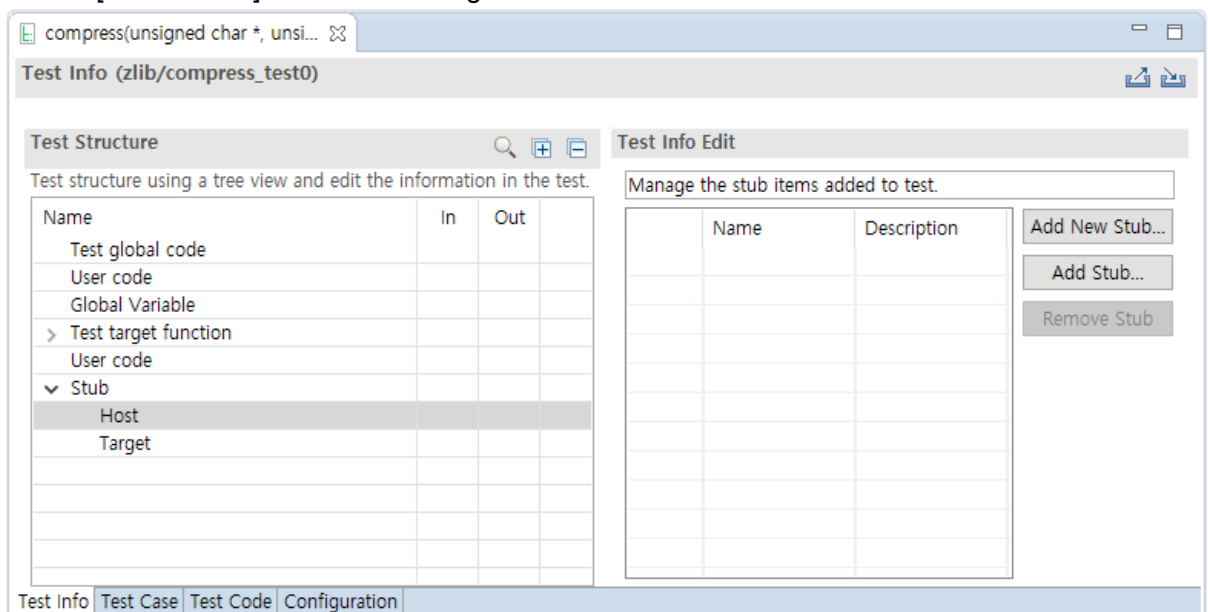
If you select “Host” or “Target”, you can only edit the stubs in the selected test environment.

- Dragging and dropping into the test editor in Stub view



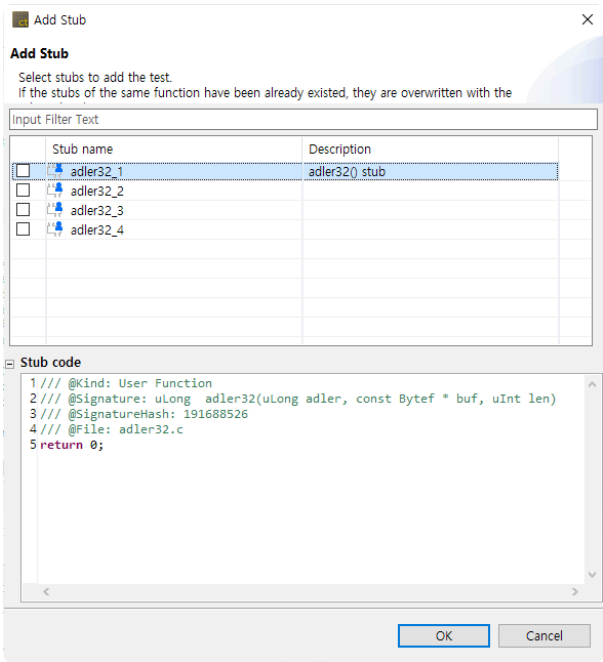
✿ If linked by drag & drop, it is linked to all the targets and all the hosts.

- 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.



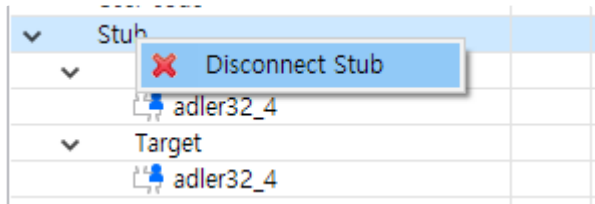
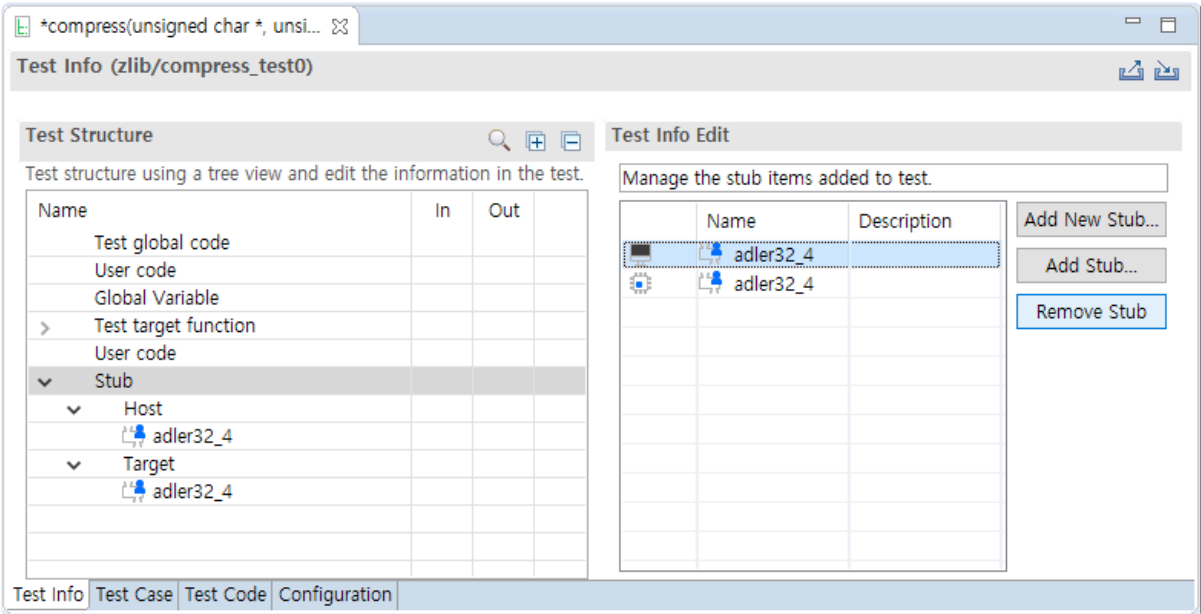
3. Among the existing stub list, check the stubs to be linked and select [OK].





# Unlink between stubs and tests

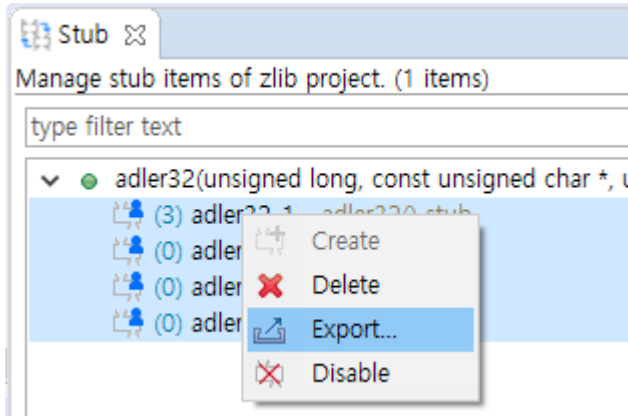
Select the stubs linked in the Test editor and select [Remove Stub] button or select the stubs in the test structure tree and right-click it to unlink them.



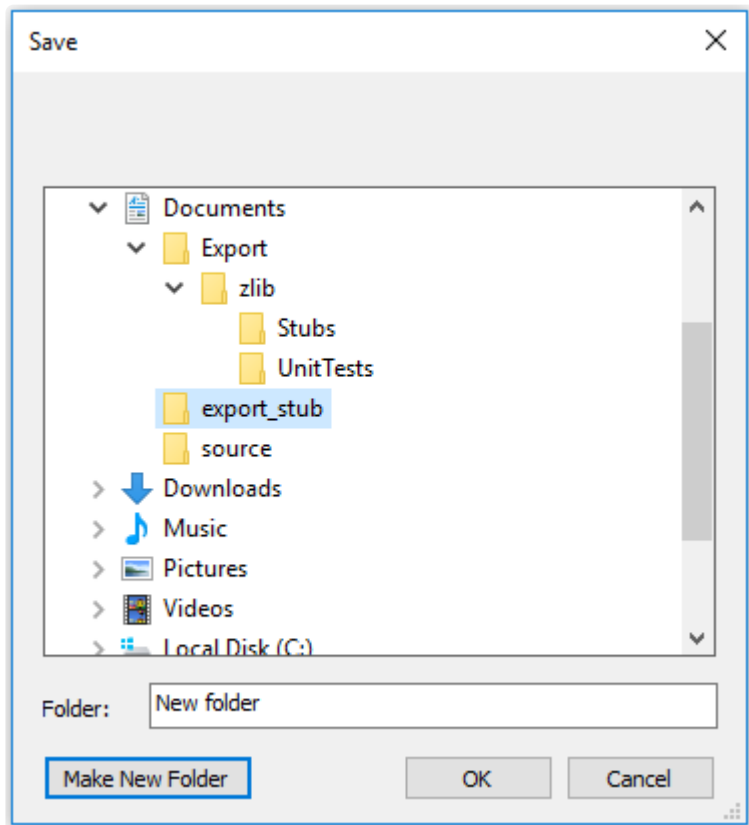


## Export stubs

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



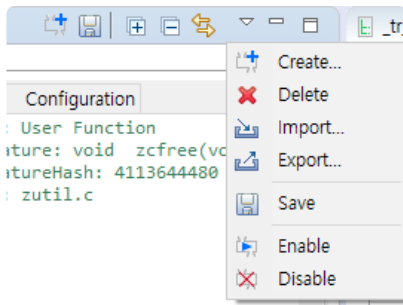
2. Specify the saving location of the stub file in the directory dialog and select [OK].



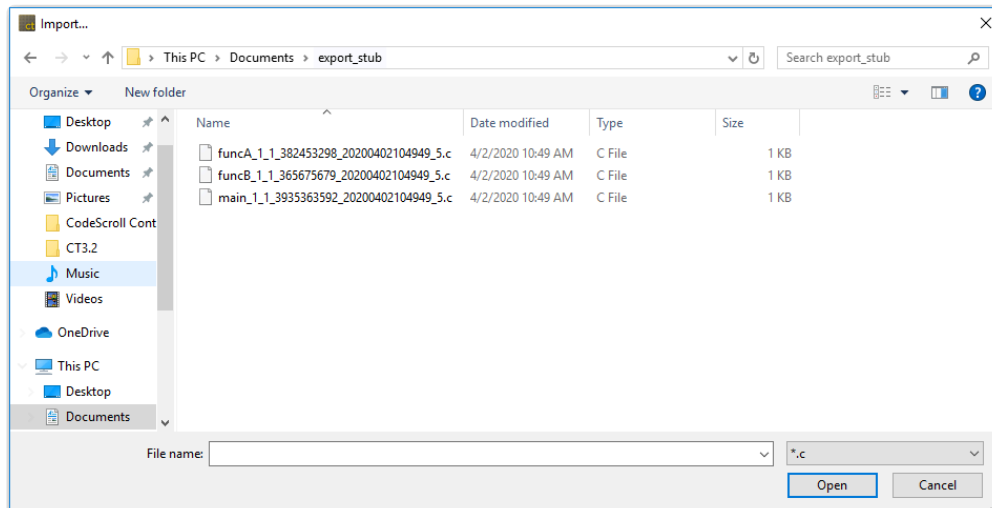
## Import stubs

1. Select [Import...] menu from the pull-down menu (▽) in Stub view.

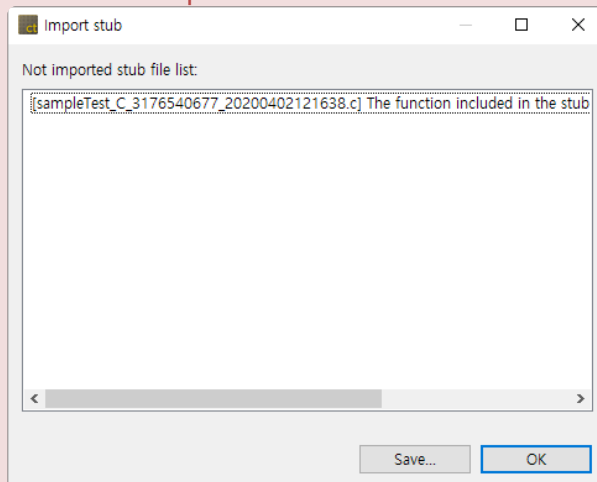




2. Select the stub file to be imported and select the [Open] button.



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

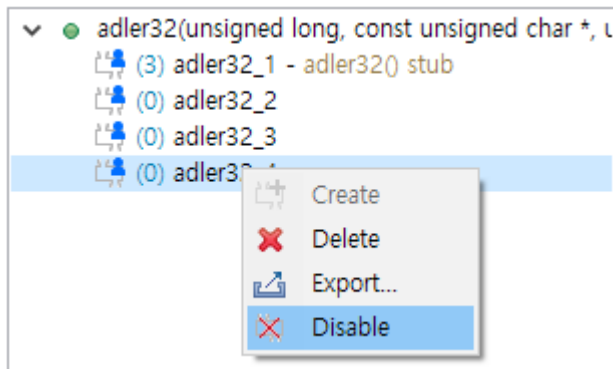


## Use/Not use stubs

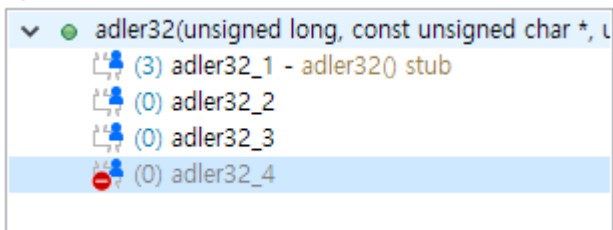
You can set whether to use the created stubs.

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.

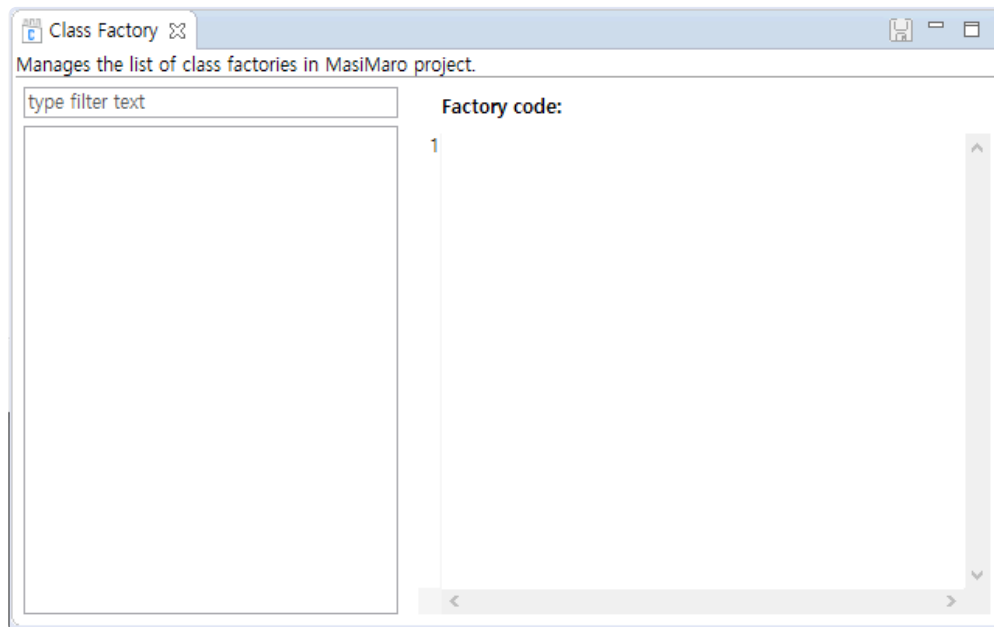




## 14.9. Class Factory View

---

The Class Factory view shows the class list belonging to the project. The auto-created test code creates an object by using the class creation function located in the Class Factory view. If you modify the object creation code in the Class Factory view, it is applied to the object creation of all the tests that use that factory.



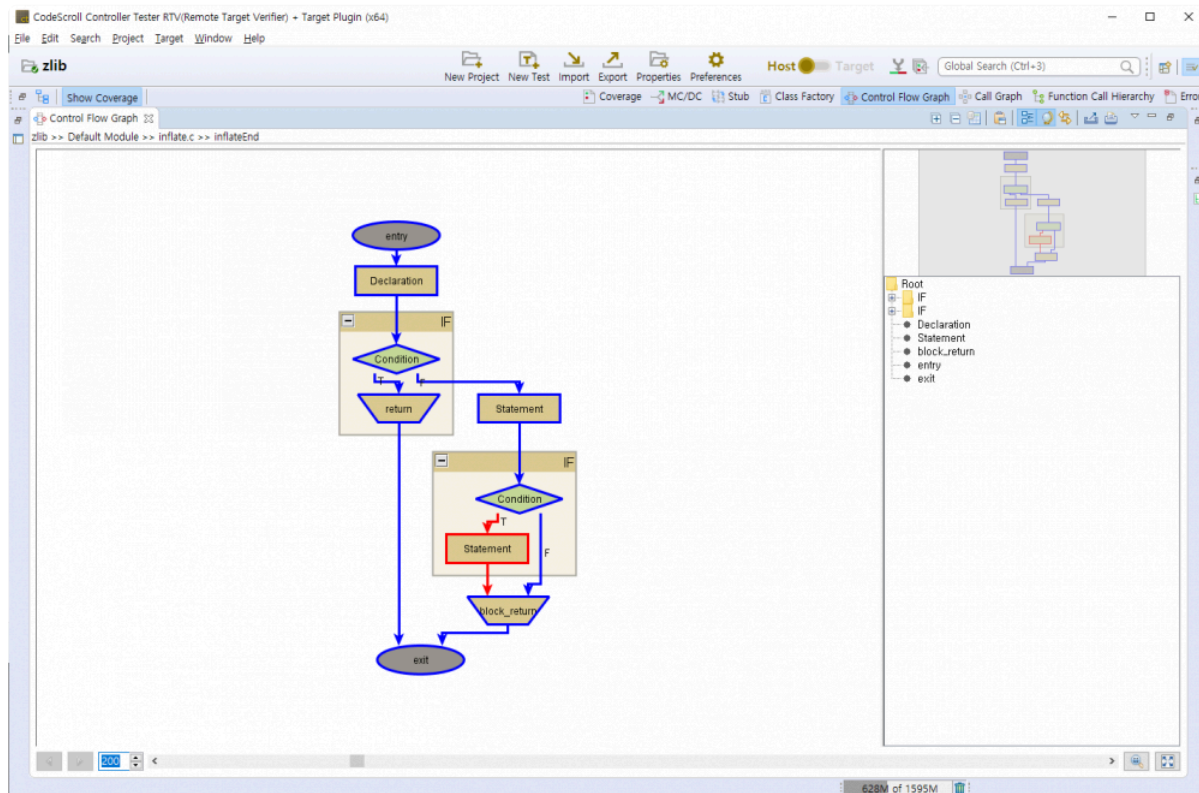
If you don't want to use the class factory, you can select and change the constructor to be used directly in the test information editor.



## 14.10. 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

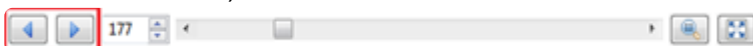
Menu	Description
 Unfold All	Shows all group nodes.
 Fold All	Hides all group nodes.
Show Call Graph	Shows the function call graph for the function selected in the current view.
 Show Legend	Shows the legend(the node type in the currently displayed graph).
 Show Outline	Displays the graphs in a tree 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.
 Export View Content	Exports the contents of view in a report.
Save as Graph Format	Creates the graph model file for the graph displayed on the current screen. Four kinds of formats supported: <ul style="list-style-type: none"> <li>• Graph Modeling Language XML (*.xgml)</li> <li>• Graph Modeling Language(*.gml)</li> <li>• yWorks Binary Graph Format(.ygf)</li> <li>• Trivial Graph Format(*.tgf)</li> </ul>
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.

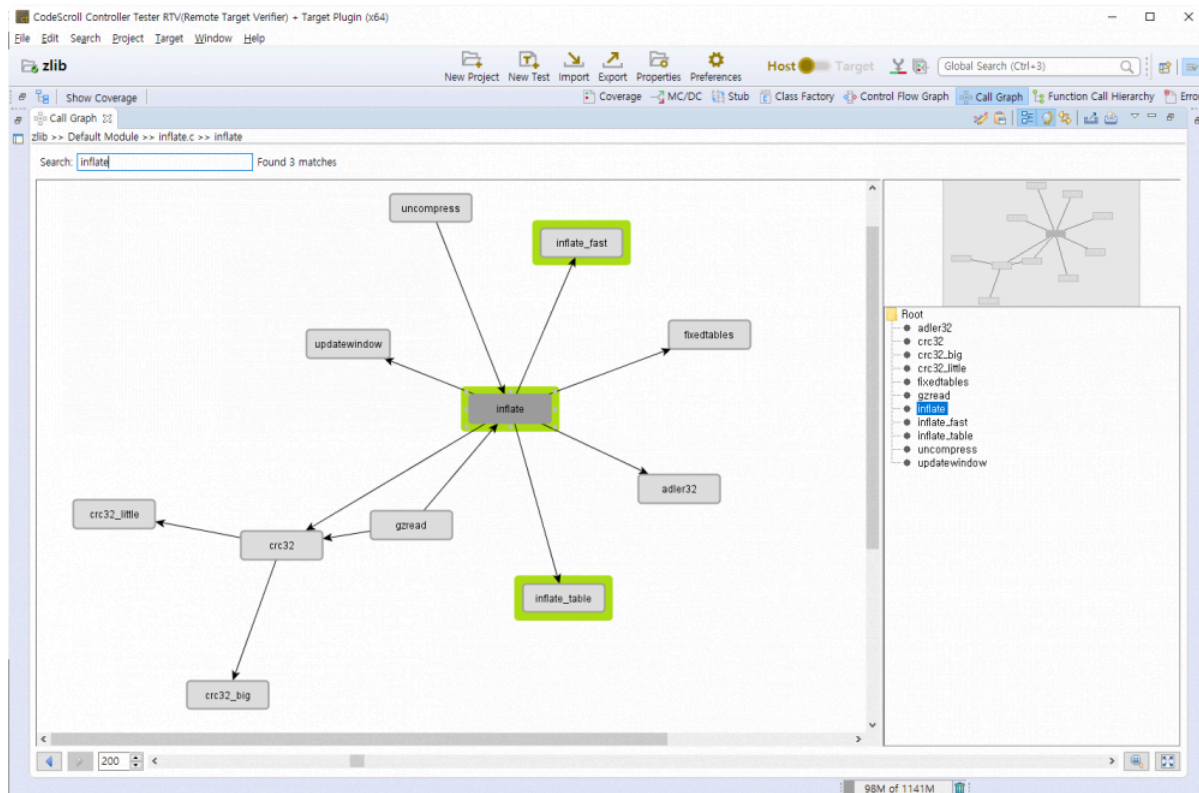


You can reset the zoom out/in ratio by using  [Initialize zoom out/in ratio] button at the lower right corner of the view, and change the zoom out/in ratio accordingly to the view size by using  [Fit to view size] button.



## 14.11. 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.



 <b>Print View Content</b>	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: <ul style="list-style-type: none"> <li>• Graph Modeling Language XML (*.xgml)</li> <li>• Graph Modeling Language(*.gml)</li> <li>• yWorks Binary Graph Format(.ygf)</li> <li>• Trivial Graph Format(*.tgf)</li> </ul>
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.

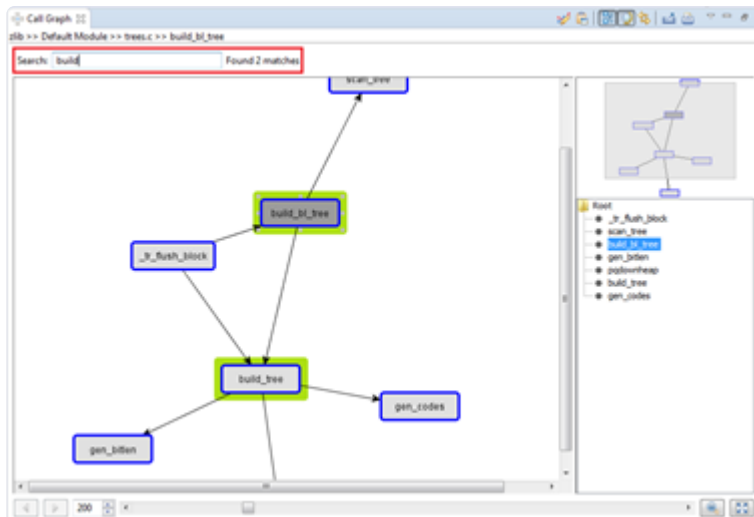
## Node Pop-up menu

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

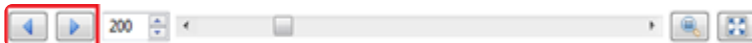
When you enter a search word in the top of the view, the functions with the name containing the entered search word are highlighted.





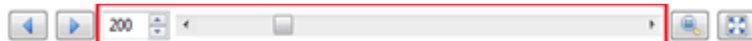
## History function



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 existed between two nodes so that you



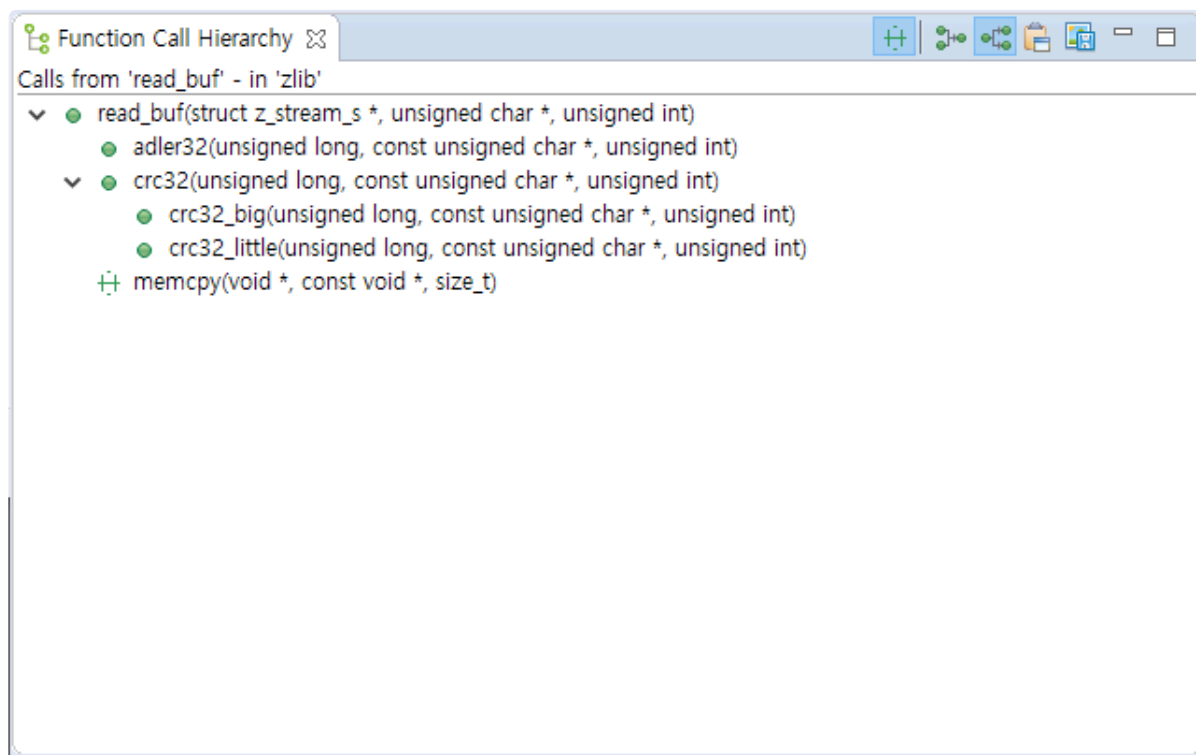
can see easily the interesting section in the complex function call relationship.



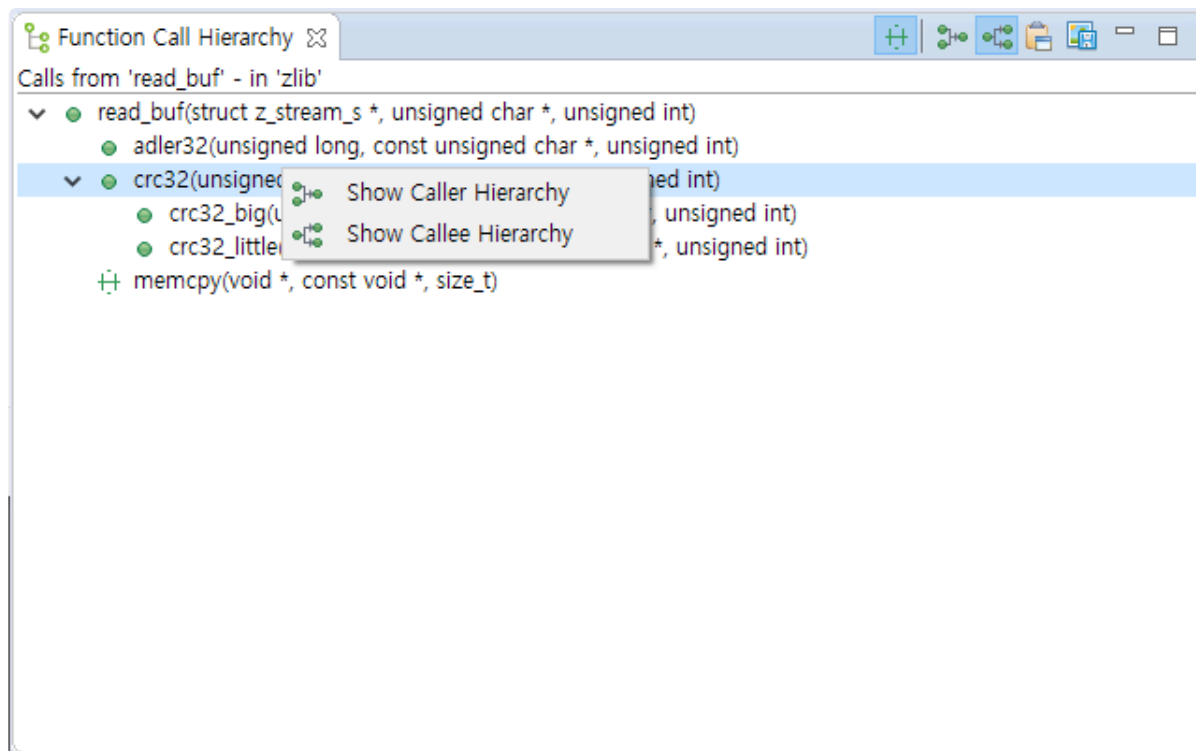
## 14.12. Function Call Hierarchy View




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)





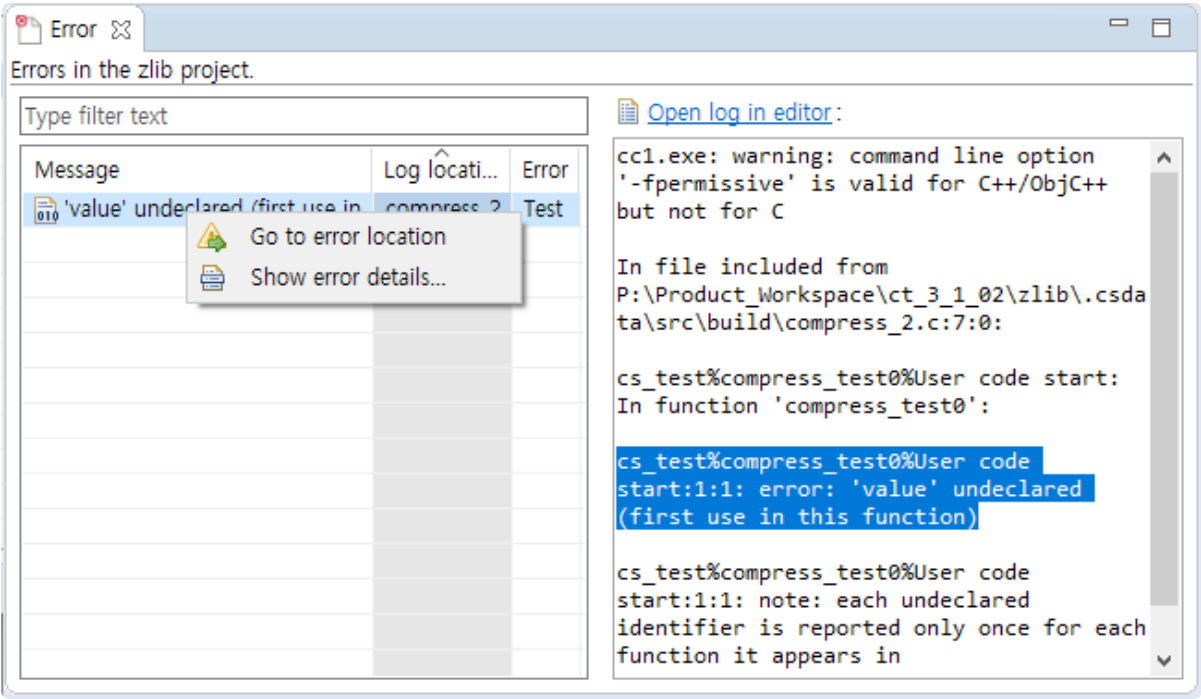


Menu	Description
 Show System Function	Sets whether to indicate the call information for the system function in a hierarchical structure.
 Show Caller Hierarchy	Shows the functions that call the selected functions in a hierarchical structure.
 Show Callee Hierarchy	Shows the functions called by the selected function in a hierarchical structure.






# 14.13. 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

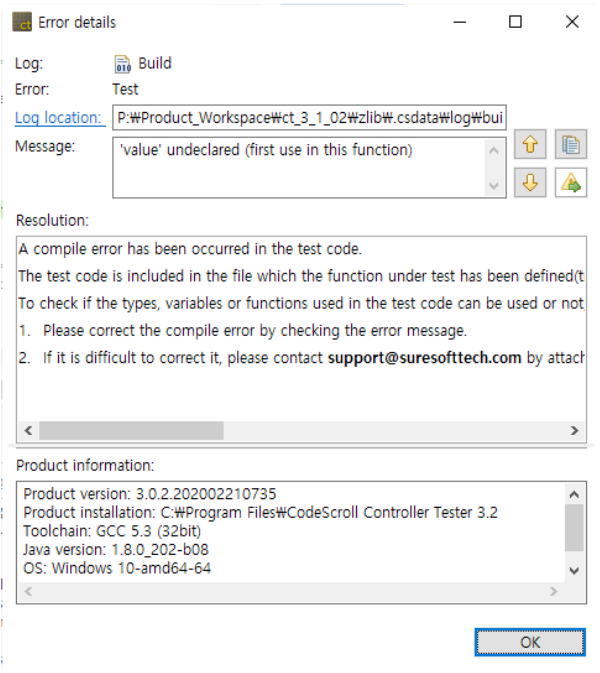


# Go to the error location

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

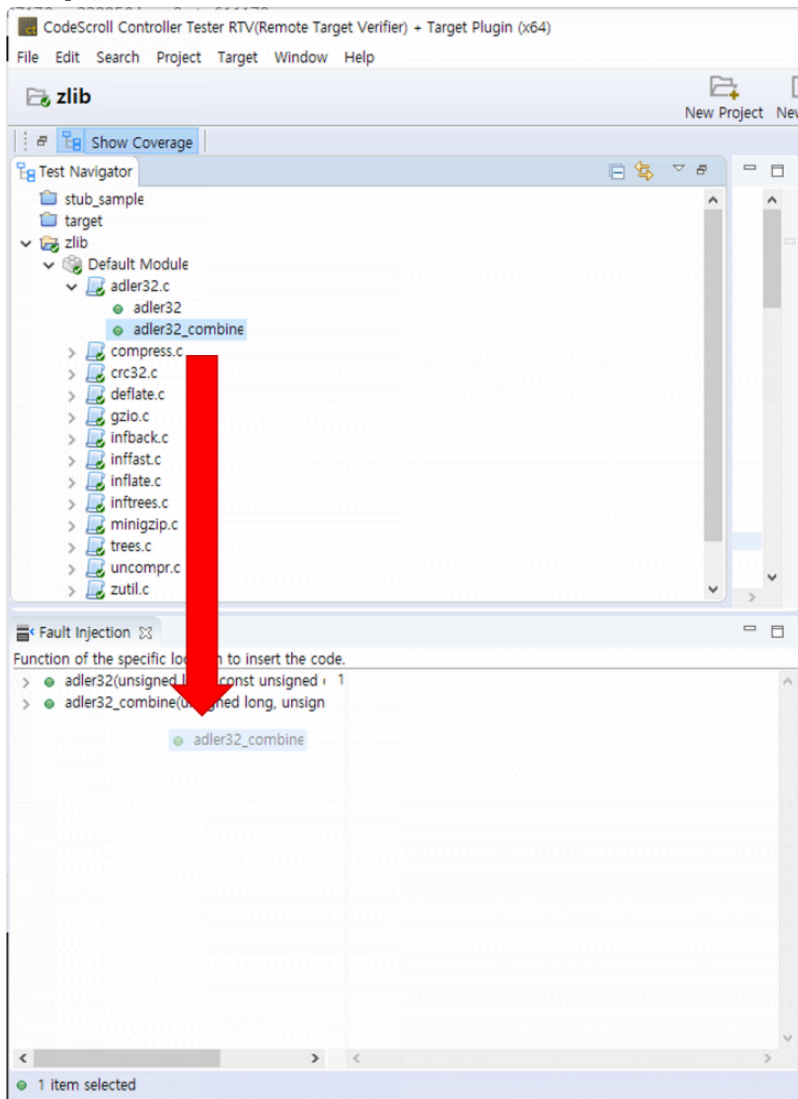
Button	Description
Previous	Shows the previous error information.
Next	Shows the next error information.
Copy	Copies the error information.
Go to Error	Points to the location of the error.



## 14.14. Fault Injection View

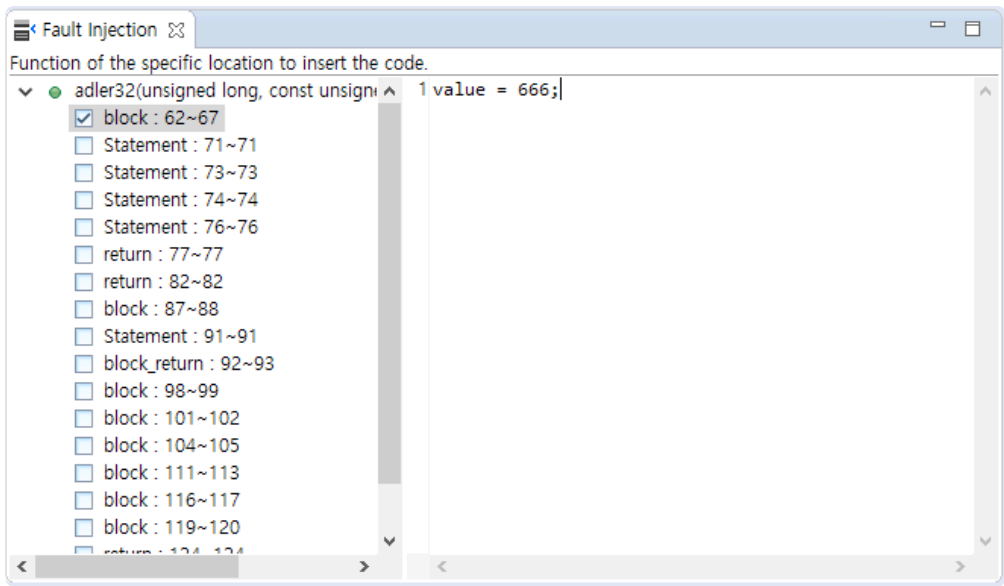
The [Fault Injection] view provides the feature for inserting the code needed to test additionally into the specific area of the function to be tested.

1. In [Test Navigator], drag and drop the function that you want to insert a fault into [Insert Code view].

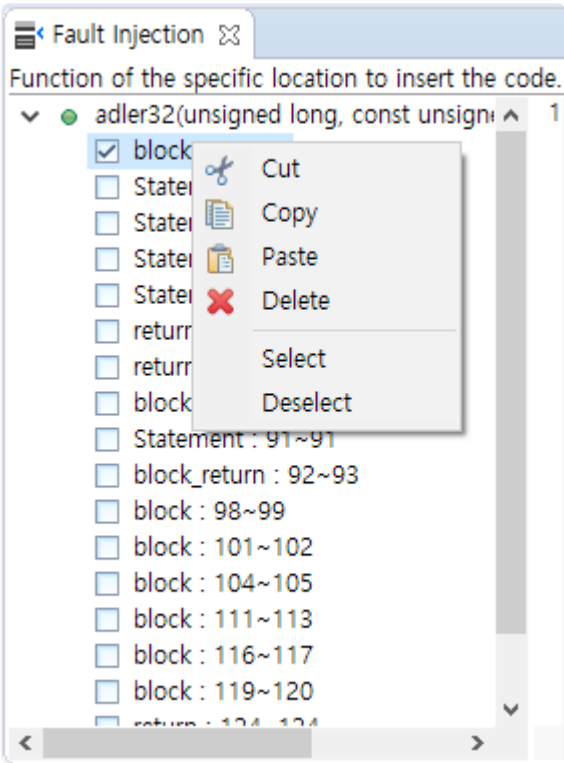






2. In the tree structure left to the view, click the checkbox for the node that you want to insert a fault and enter the user code in the right edit window.
3. Click [Run Unit Test] button to run the test that the fault inserted is applied.





Context menu in the Fault Injection view



Menu name	Description
 Cut	Cuts the code inserted by user.
 Copy	Copies the code inserted by user.
 Paste	Pastes the code inserted by user.
 Delete	Deletes the code inserted by user.

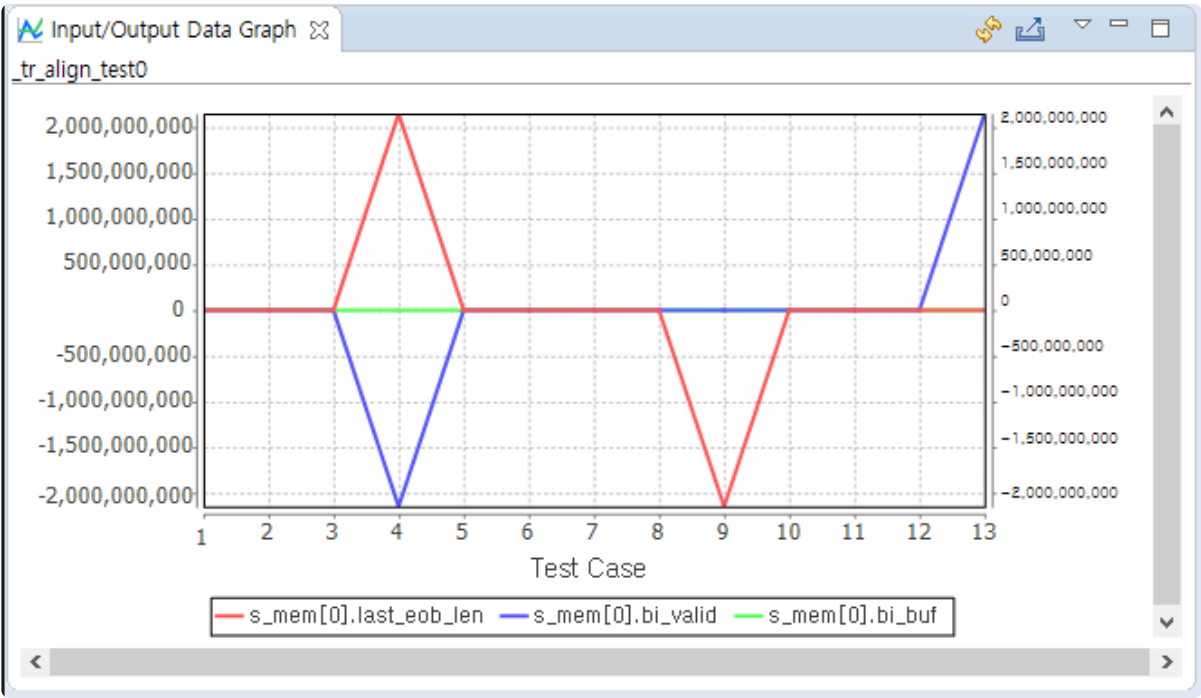


Select	Selects the checkbox for the selected node.
Deselect	Deselects the checkbox for the selected node.



# 14.15. Input/output Data Graph View

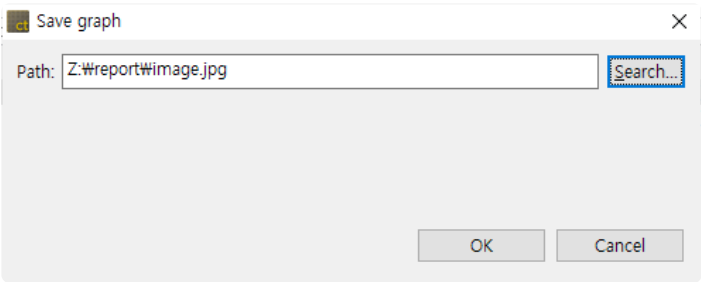
The Controller Tester provides the data for input value/expected value/output value in graph format. The horizontal axis of the graph indicates the number of the test case and the vertical axis indicates the data of the test case.



Toolbar icon in Input/output Data Graph view

Toolbar icon	Description
	Refreshes the input/output data.
	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.

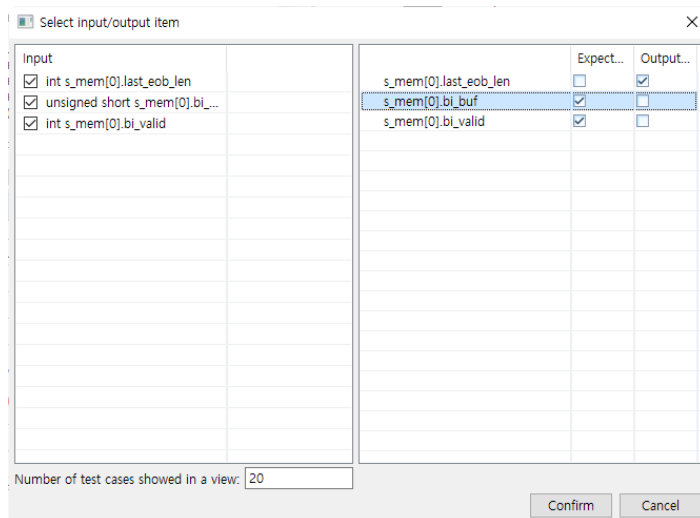




## Graph settings

Select [Select input/output item] in the pull-down menu ( $\nabla$ ) 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.








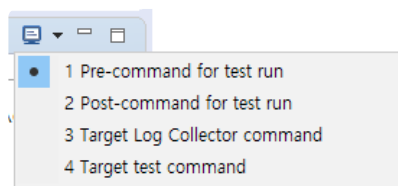
## 14.16. Console View

Enter the instruction or batch file in [Pre-command for test run] and [Post-command for test run] of [Project] -> [Properties] -> [Unit 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.

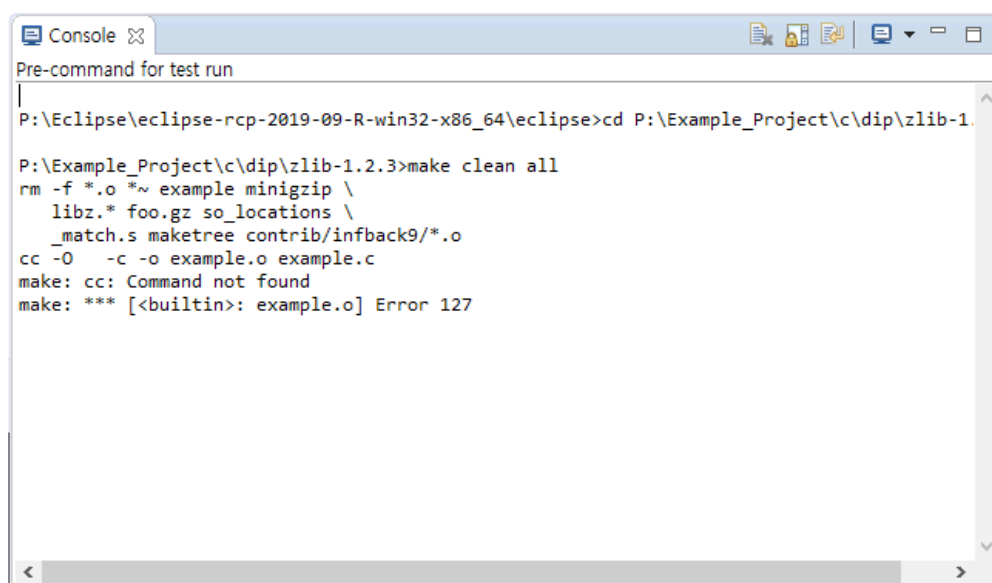
### Toolbar icon in the [Console] view

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.



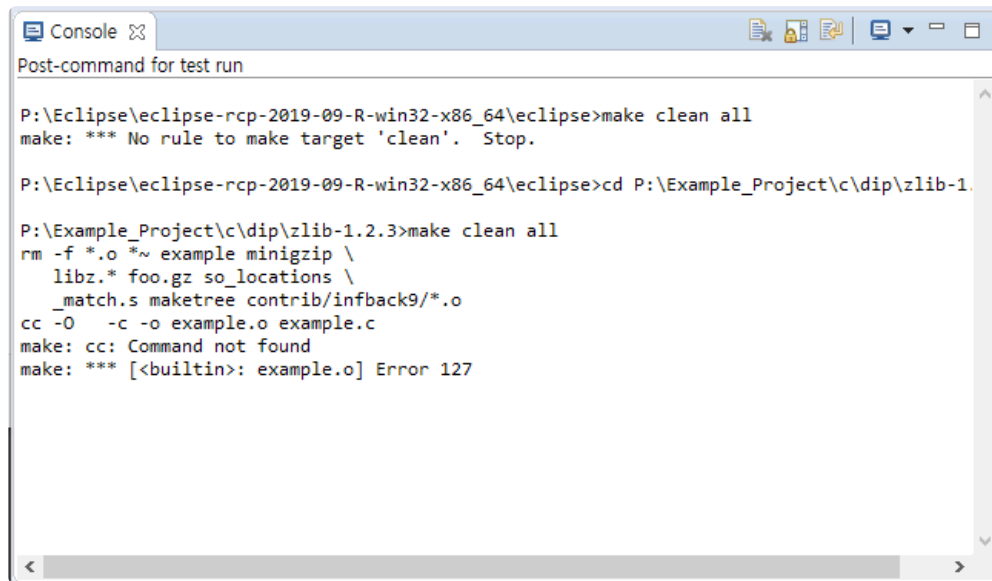
```

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/infbck9/*.o
cc -O -c -o example.o example.c
make: cc: Command not found
make: *** [<built-in>: example.o] Error 127

```



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/infbck9/*.o
cc -O -c -o example.o example.c
make: cc: Command not found
make: *** [builtin]: example.o] Error 127
```



# 15. Analysis Perspective

---

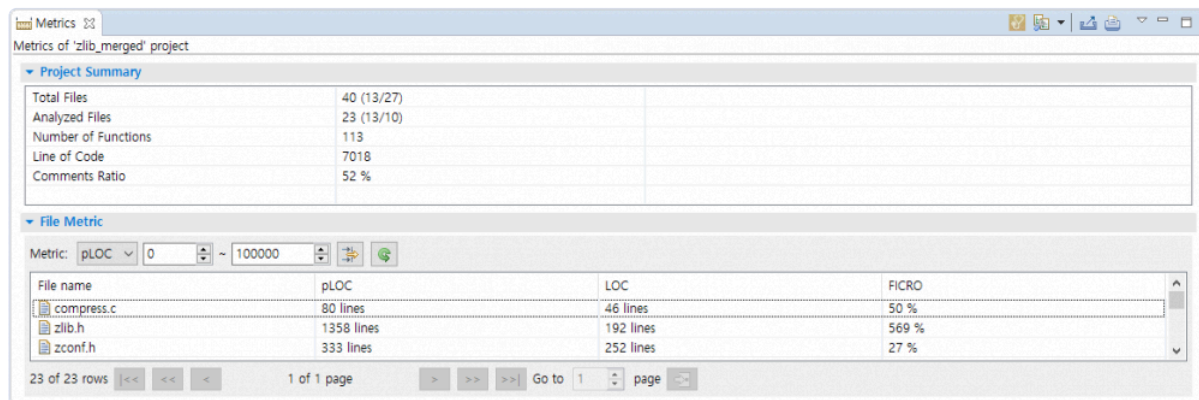
The components that make up the analysis perspective are:

- [Metrics View](#)
- [Metrics Chart View](#)
- [Metrics Top Chart View](#)
- [Metrics Bar Chart View](#)
- [Metrics Diagnosis Chart View](#)
- [Control Flow Graph View](#)
- [Call Graph View](#)
- [Unused Function View](#)
- [Source-Header Relation View](#)
- [Global Variable Relation View](#)
- [Test Navigator View](#)







## 15.1. 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.

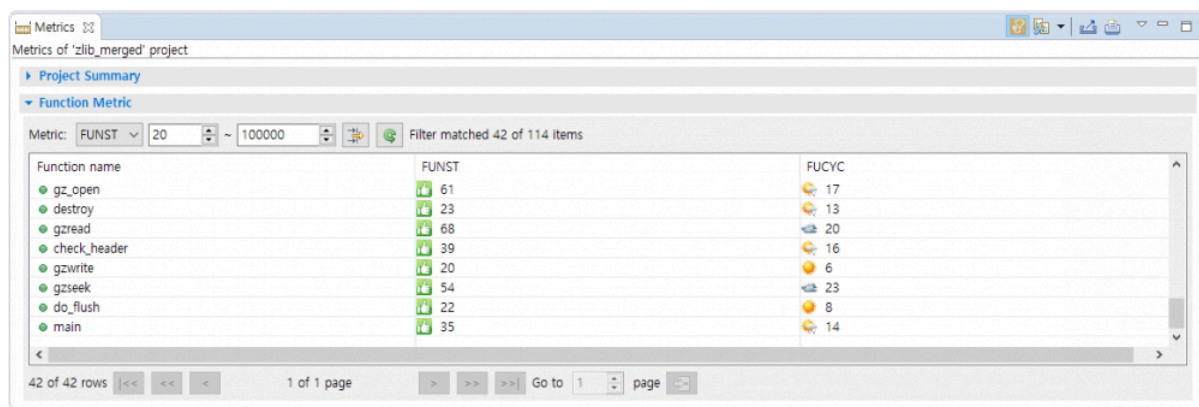
## Pull-down menu

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

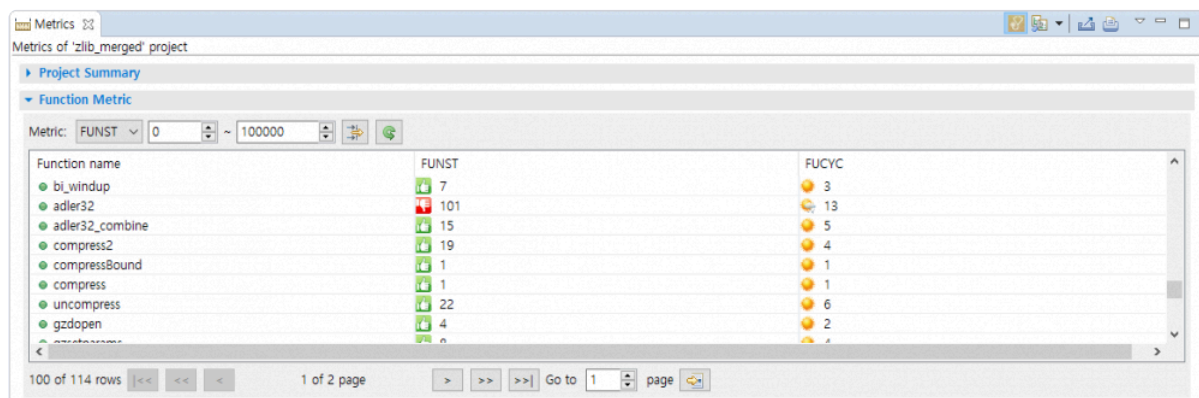


The filtering allows you to view only those items within the desired range for the specific metric.

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



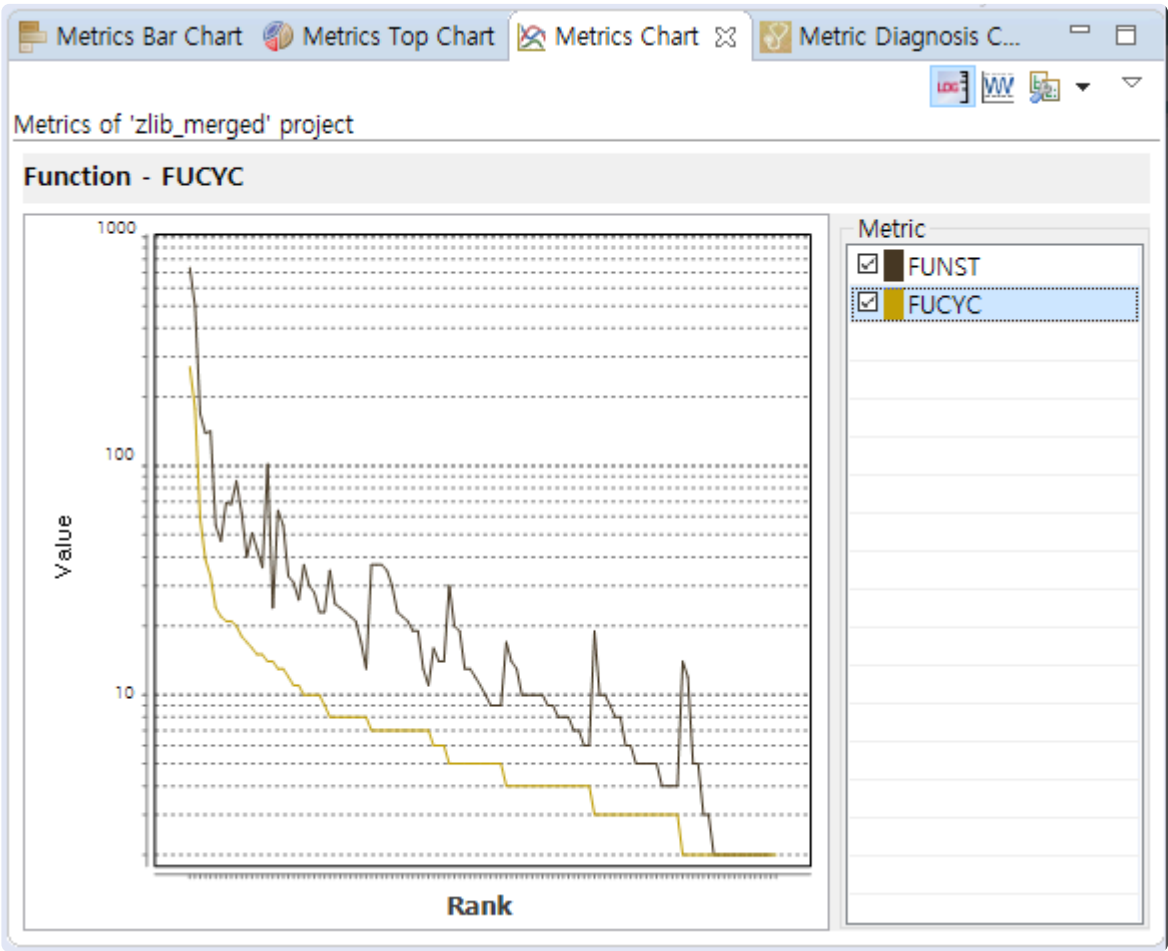


The Paging allows you to view many items conveniently.










# 15.2. 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 metric	Sets whether to hide or show the metrics.
Metrics View Options	Sets the connection function with Metric view.

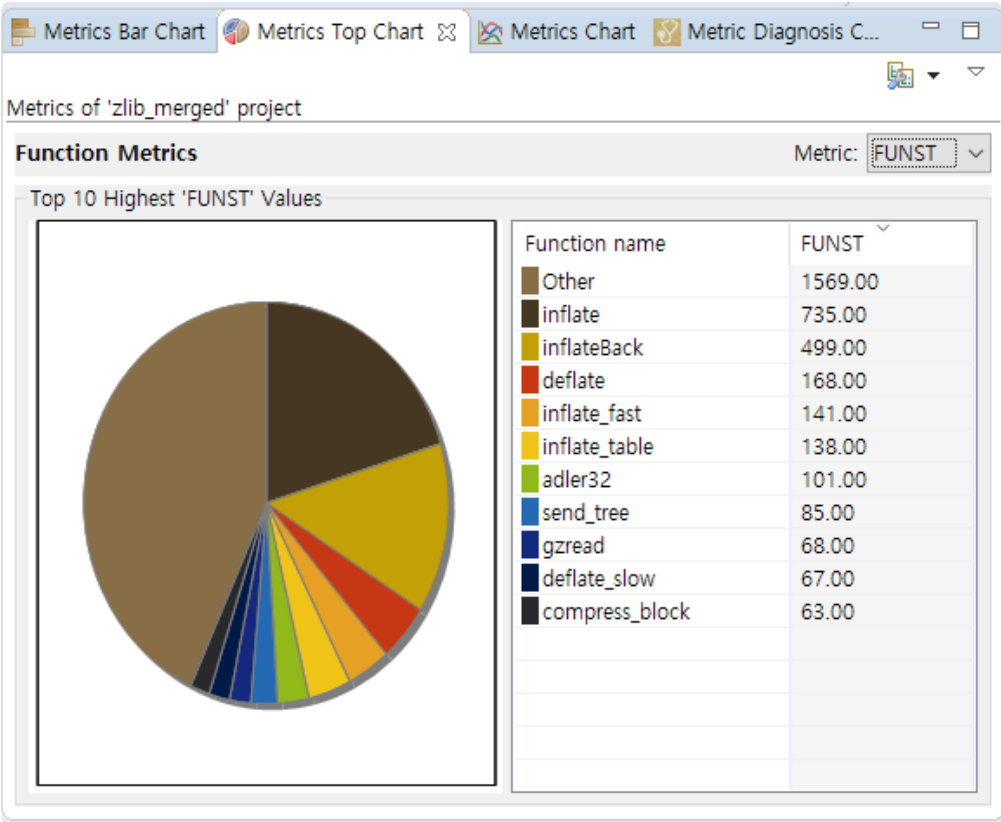
## Select Metric

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.



# 15.3. 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

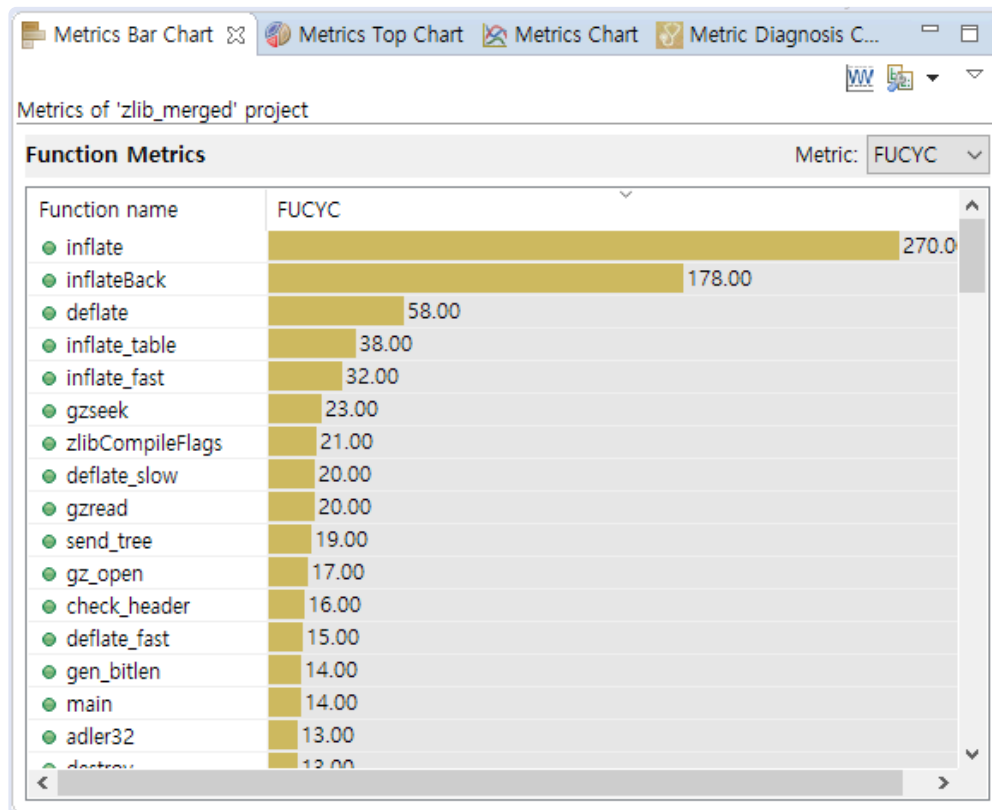
Menu	Description
Metrics Hide/Show Setting	Sets whether to hide or show the metrics.
Metrics View Options	Sets the number of top items to be displayed and sets whether to show the sum of the resting items.

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







## 15.4. 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).



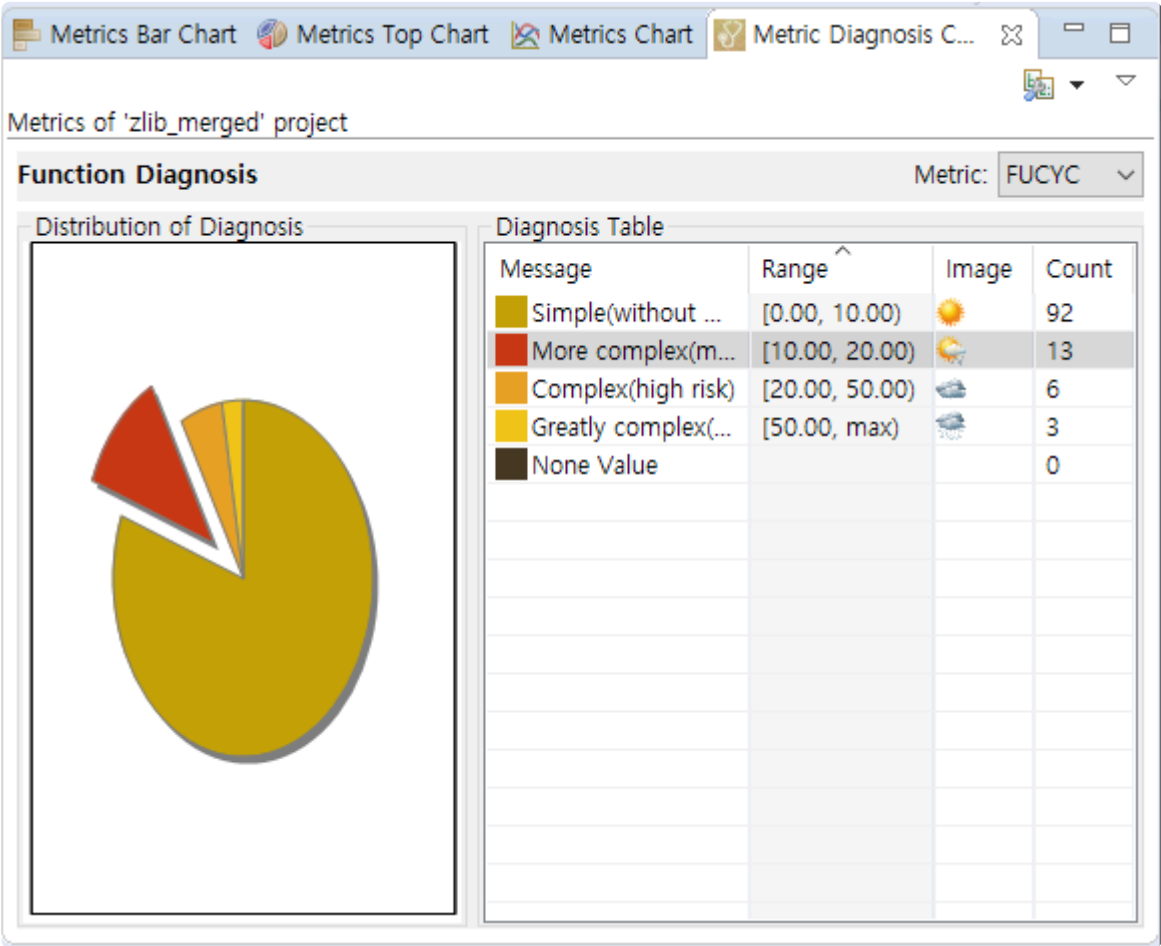
## Pull-down menu

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.



## 15.5. 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/	Sets whether to hide or show the metrics.



Show	
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.



## 15.6. 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.

Name	Module	File
Unused (42 items)		
• _tr_tally	Default Module	trees.c
• adler32_combine	Default Module	adler32.c
• compress	Default Module	compress.c
• crc32_combine	Default Module	crc32.c
• deflateBound	Default Module	deflate.c
• deflateCopy	Default Module	deflate.c
• deflatePrime	Default Module	deflate.c
• deflateSetDictionary	Default Module	deflate.c
• deflateSetHeader	Default Module	deflate.c
• deflateTune	Default Module	deflate.c
• deflate_fast	Default Module	deflate.c
• deflate_slow	Default Module	deflate.c
• deflate_stored	Default Module	deflate.c
• get_crc_table	Default Module	crc32.c
• gzclearerr	Default Module	gzio.c
• gzdirect	Default Module	gzio.c
• gzeof	Default Module	gzio.c
• gzflush	Default Module	gzio.c

### Icons

Icon	Description
	Classification criteria
	Function group
	Function

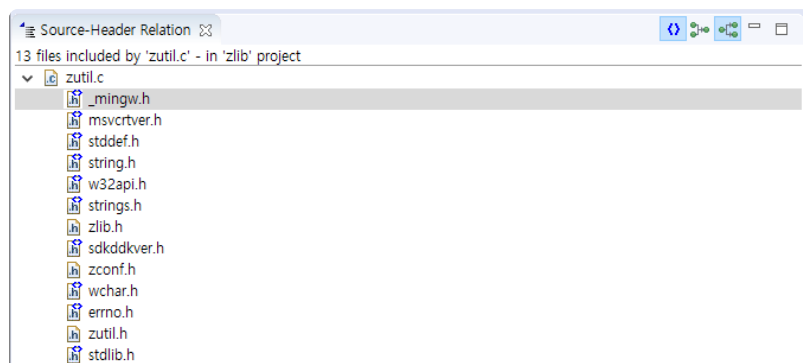
### Pull-down menu

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






## 15.7. 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

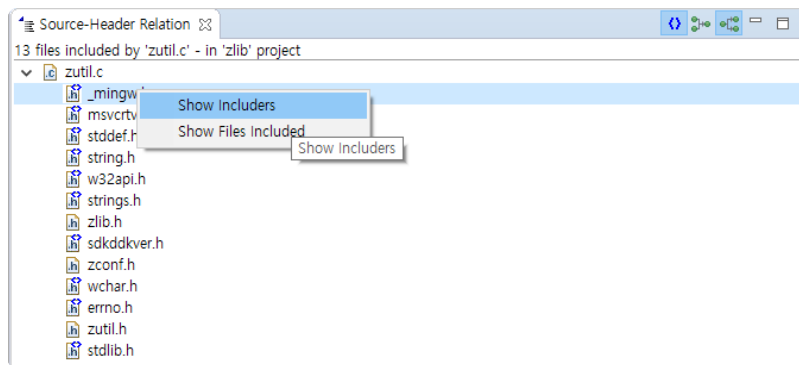
### Toolbar menu

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

Show Includes, 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.

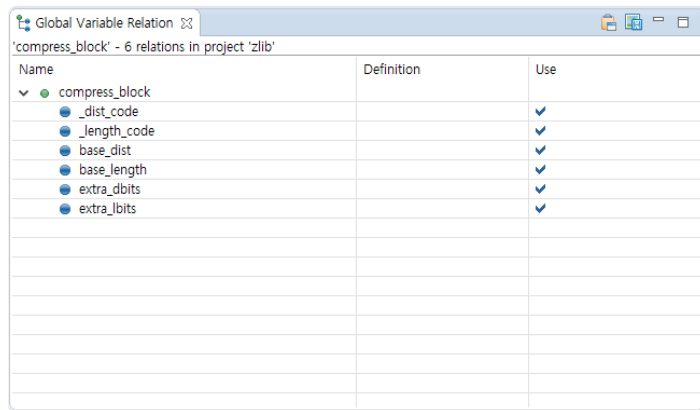








## 15.8. Global Variable Relation View

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.

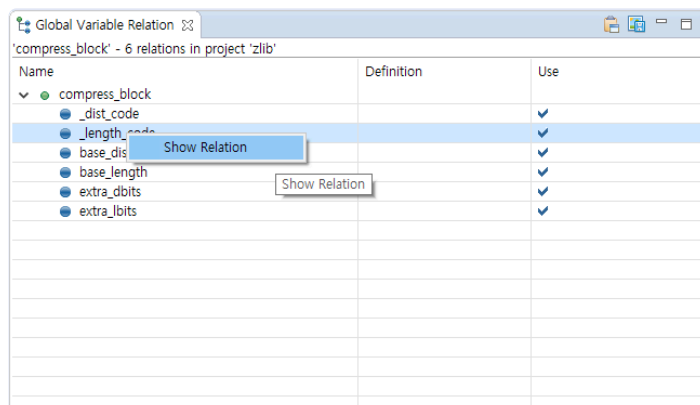


## Icons

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.

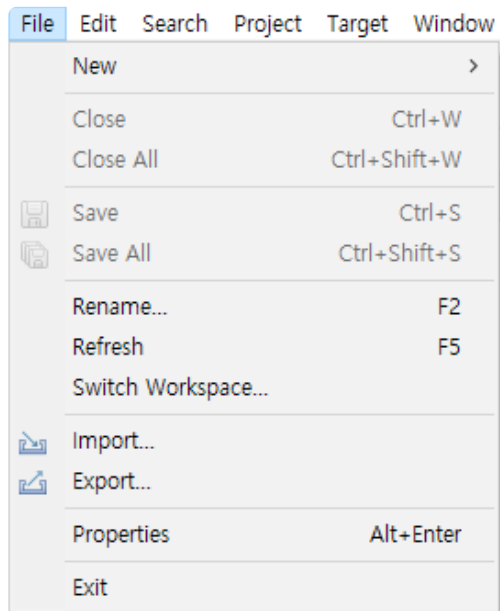




## 16. File Menu

---

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.

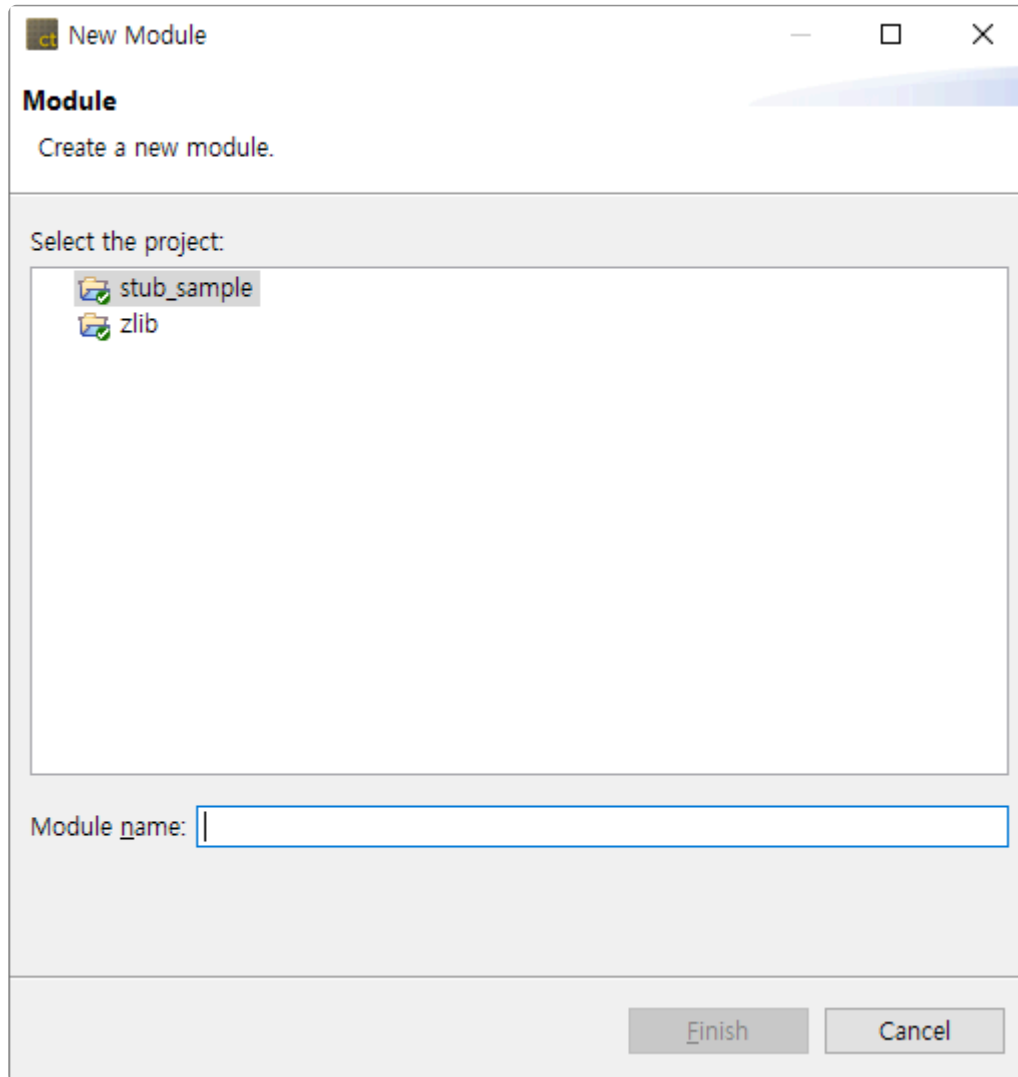


- New:
  - [Create a project](#)
  - [Create a module](#)
- Close/Close all/Save/Save all: Editor-related menu
- Rename: You can modify the name of project or module.
- [Switch workspace](#)
- [Export](#)
- [Import](#)
- Exit: Exit tool.



## 16.1. 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.



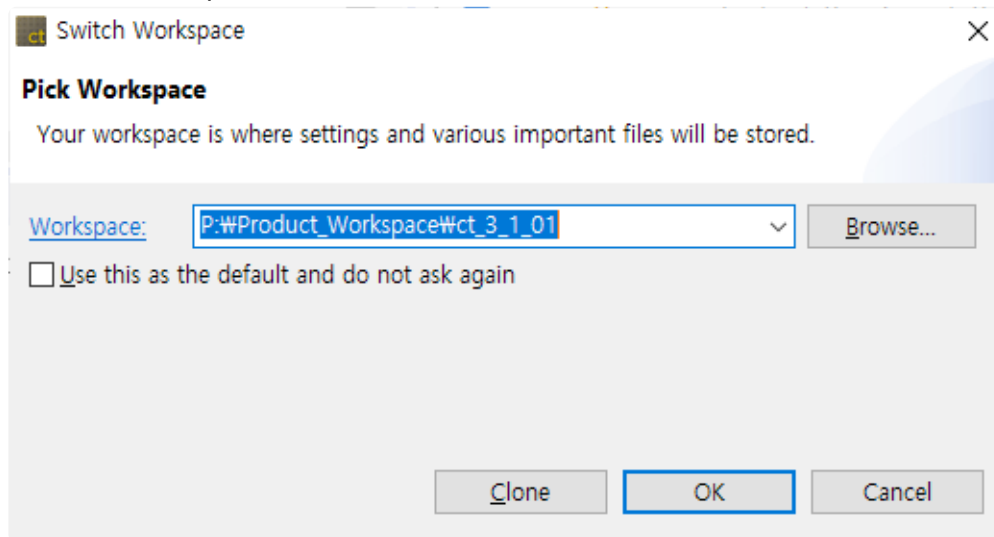
You can also create a module by using the [New] menu displayed when right-clicking a project in Test Navigator view.



## 16.2. Switch Workspace

---

You can switch the directory of workspace. When you switch 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.



## 16.3. Export

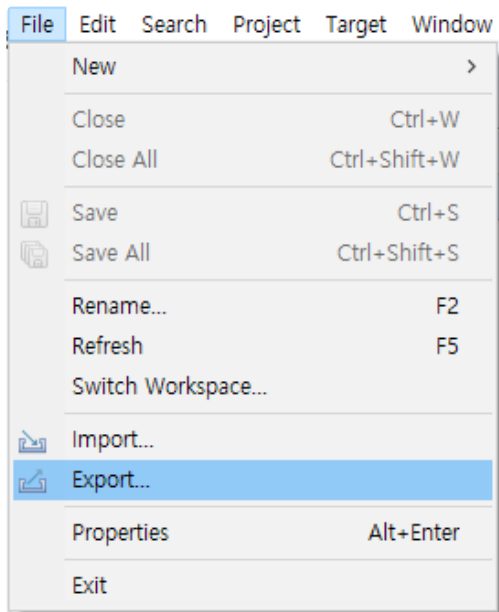
---

### Export a Project

The project can be exported including the project settings and the test. The exported project can be imported in CLI environment by using projectIE.exe.

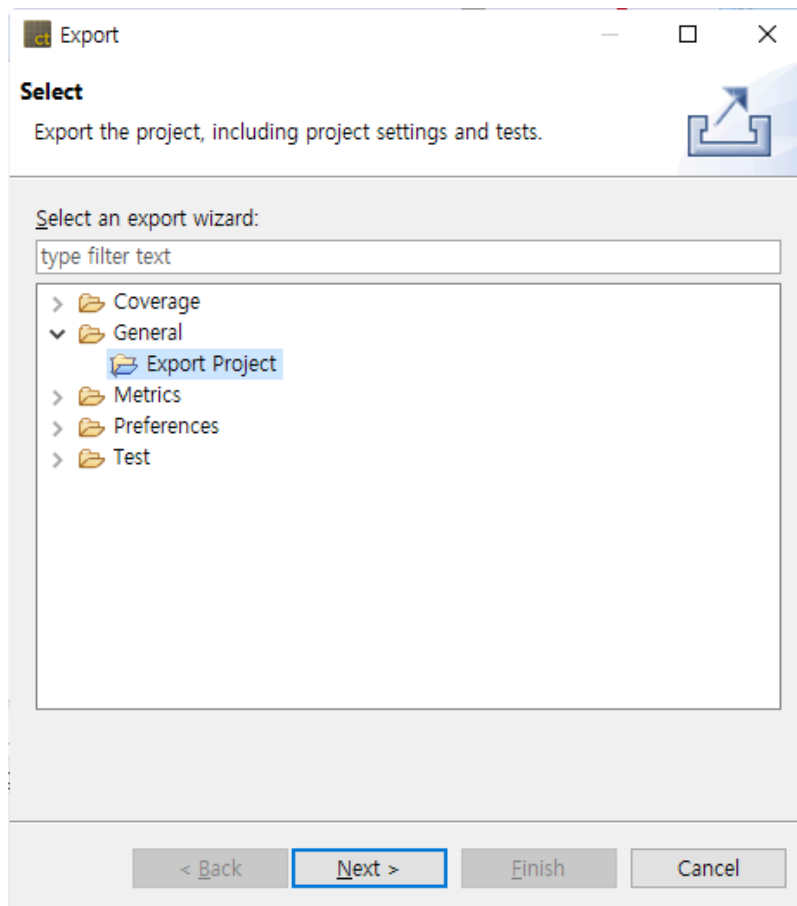
p(banner tip). For more information, see projectIE document.

1. Click [File] -> [Export] menu.



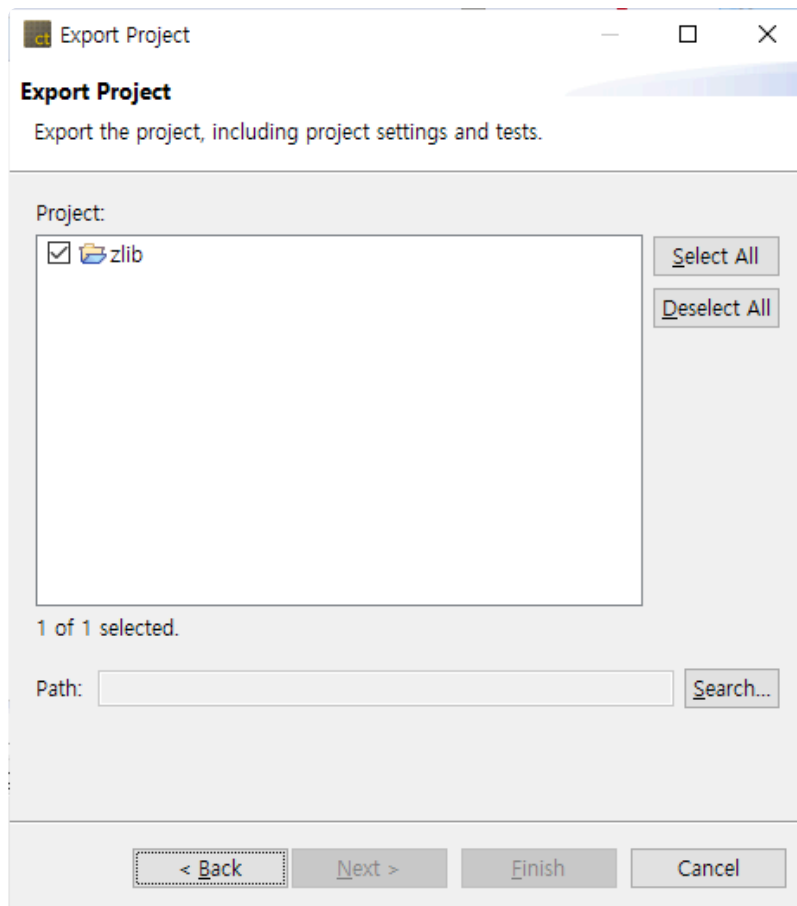
2. Click [General] -> [Export Project].





3. Select the project to be exported and the export path and click [Finish].

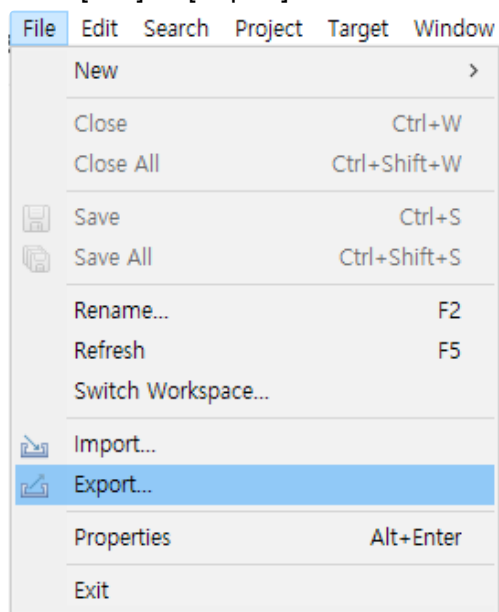




## Create a test result report

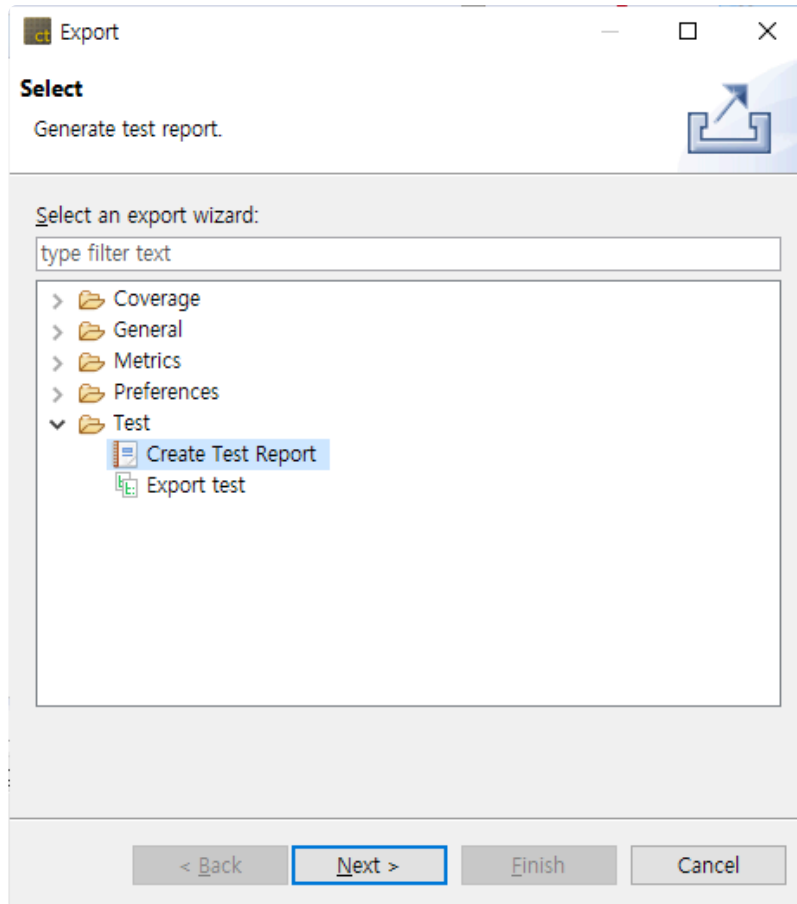
You can create the test result report in various formats by using Export.

1. Click [File] -> [Export] menu.

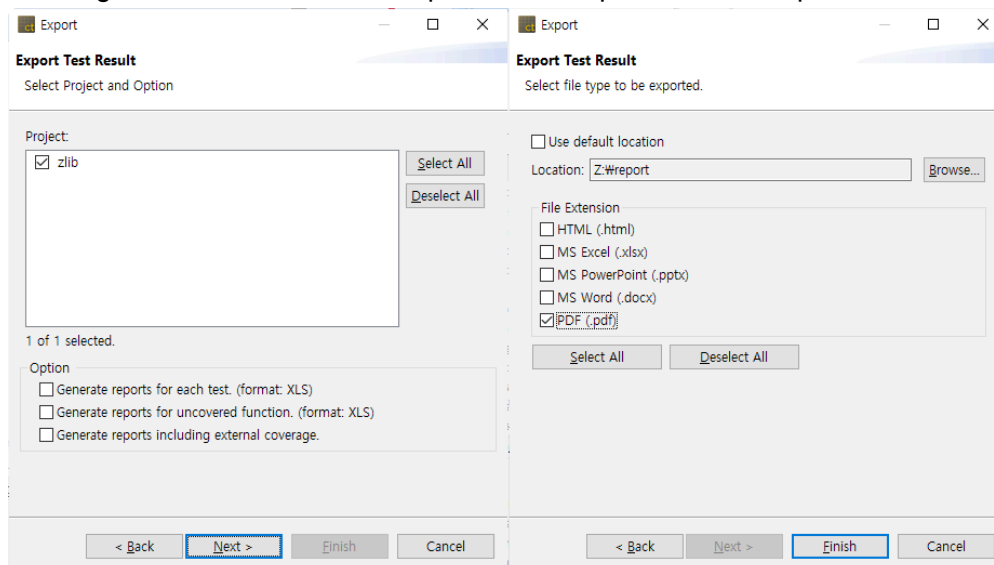




2. Click [Test] -> [Create Test Report].



3. Select the project that you want to create the report, and as an option, select Create Report for each Test, Create Report for uncovered function, and Create Report containing the external coverage, and then select the report creation path and the report extension.

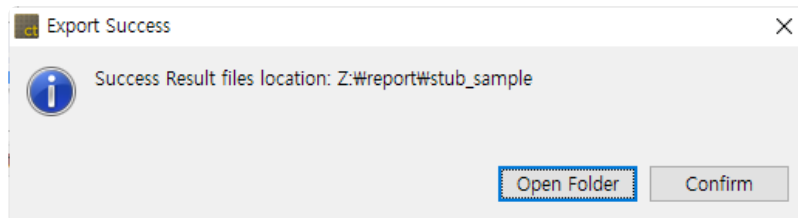


- In [Project], select the project that you want to create a report.
- In [Option] group, select the desired checkbox among [Generate reports for each test], [Generate reports for uncovered function], and [Generate reports including external



coverage].

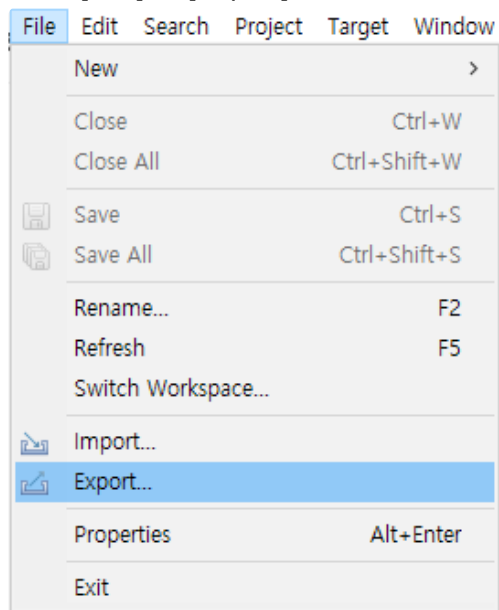
- **Generate reports for each test:** Generates the report for each test in the selected project.
  - **Generate reports for uncovered function:** Generates the report for the uncovered function in the selected project.
  - **Generate reports including external coverage:** Generates the report containing the external coverage in the selected project.
- c. Selecting the checkbox [Use default location] specified “User Account\Export” directory by default.
  - d. [File Extension] provides the report file format with html, pdf, docx, pptx and xlsx (※ if you need to create the report with the extension of doc, ppt and xls, please contact the support team). The report file extension can be selected in duplicate.
4. Click [Open Folder] button in the message “Report has been created” to check the report in the folder where it is saved, or click [Confirm] to finish the export.



## Export Test

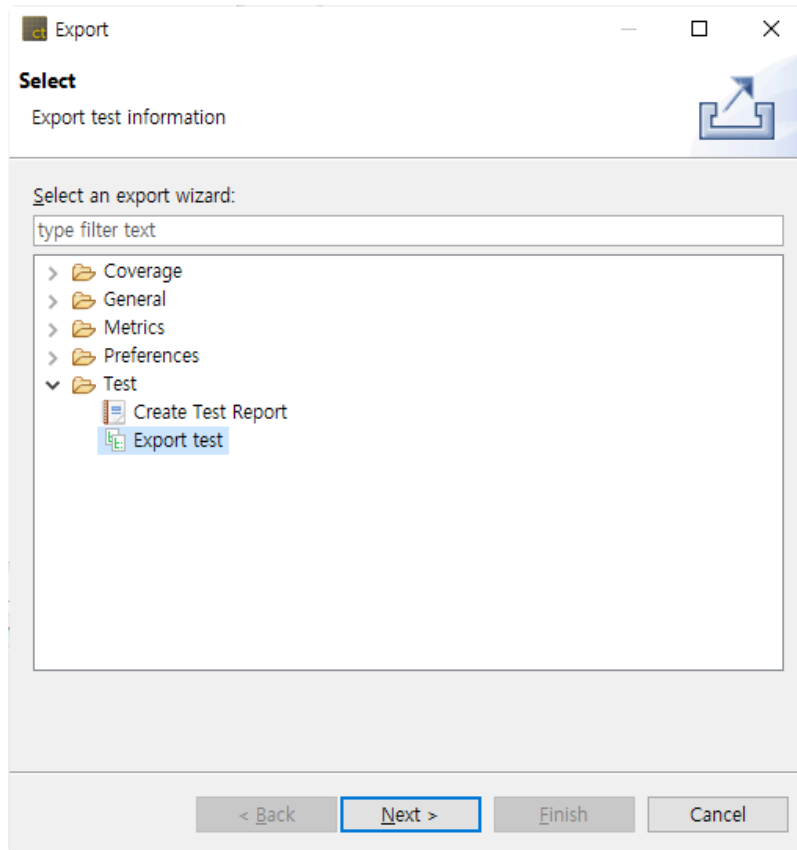
The test information created in the project can be exported at once. Before a project is not analyzed or if there is no test information (Unit/Integration Test, Stub) created, the export cannot be carried out.

1. Click [File] -> [Export].



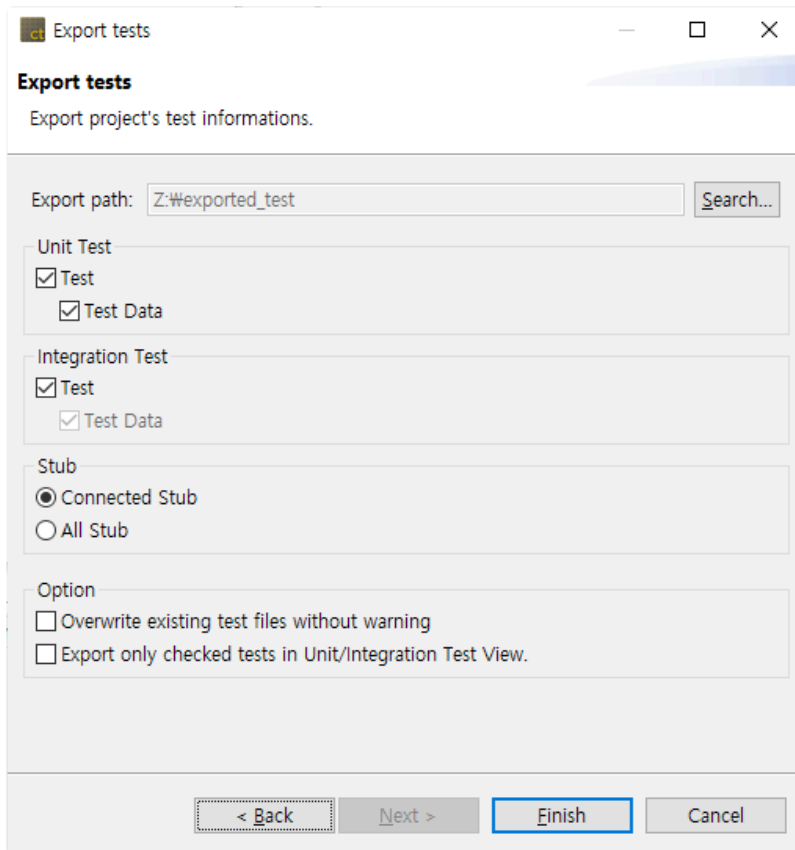


2. Click [Test] -> [Export Test].

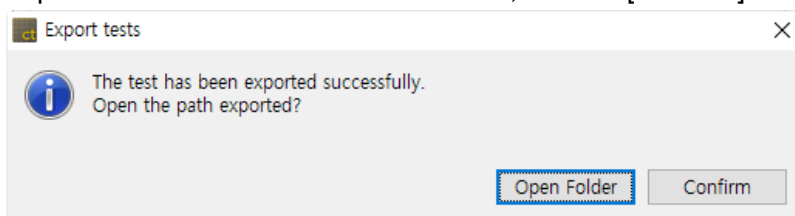


3. Select the export path, the unit test items, the integration test items, the stub items, and the options, and click [Finish].





- 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. [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 [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 2.6.14 or later version, when exporting tests, [testinfo.export] file is created in the project directory exported.

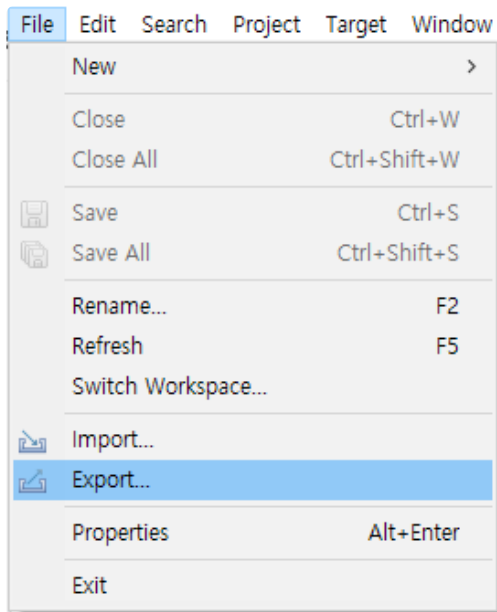


(**testinfo.export**: the file that records the exported information)

## Export Coverage

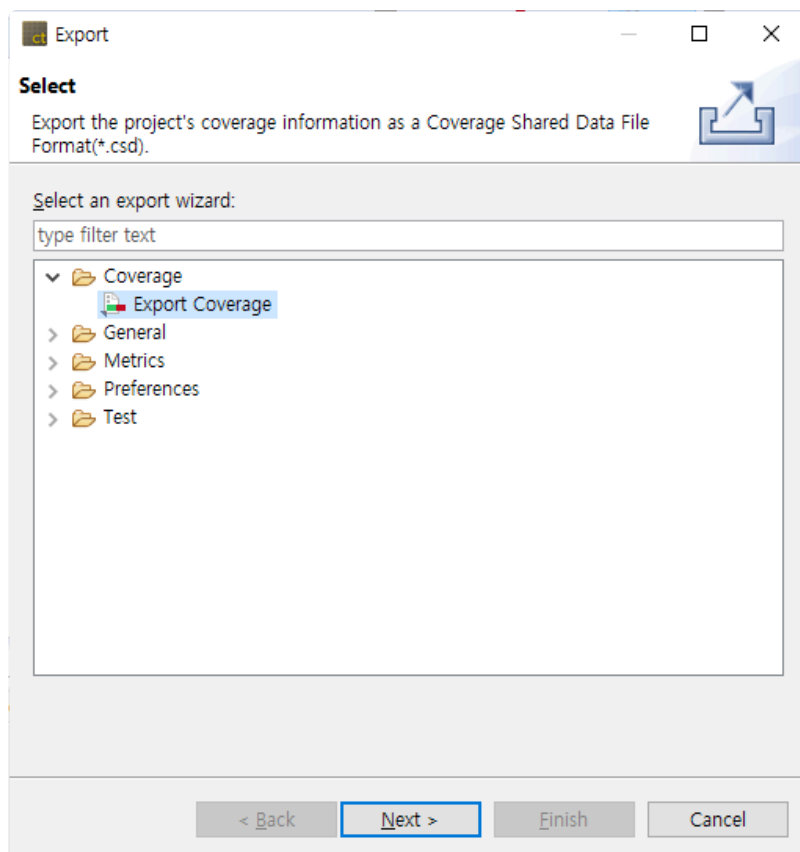
You can export the coverage information measured in the project as a file in the form of Coverage Shared Data (\*.csd) format.

1. Click [File] -> [Export].

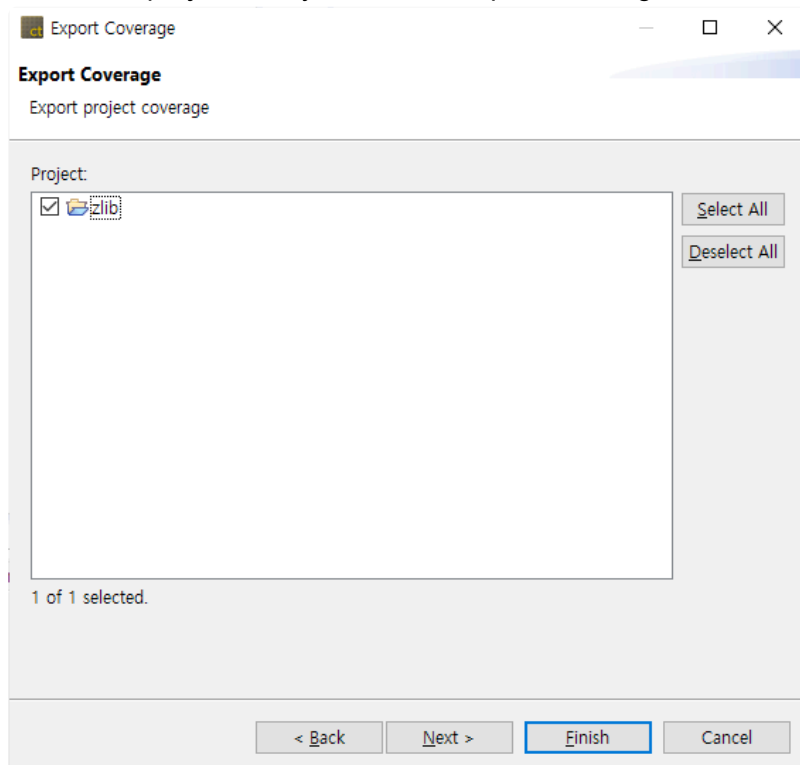


2. Select [Coverage] -> [Export Coverage] and click [Next].



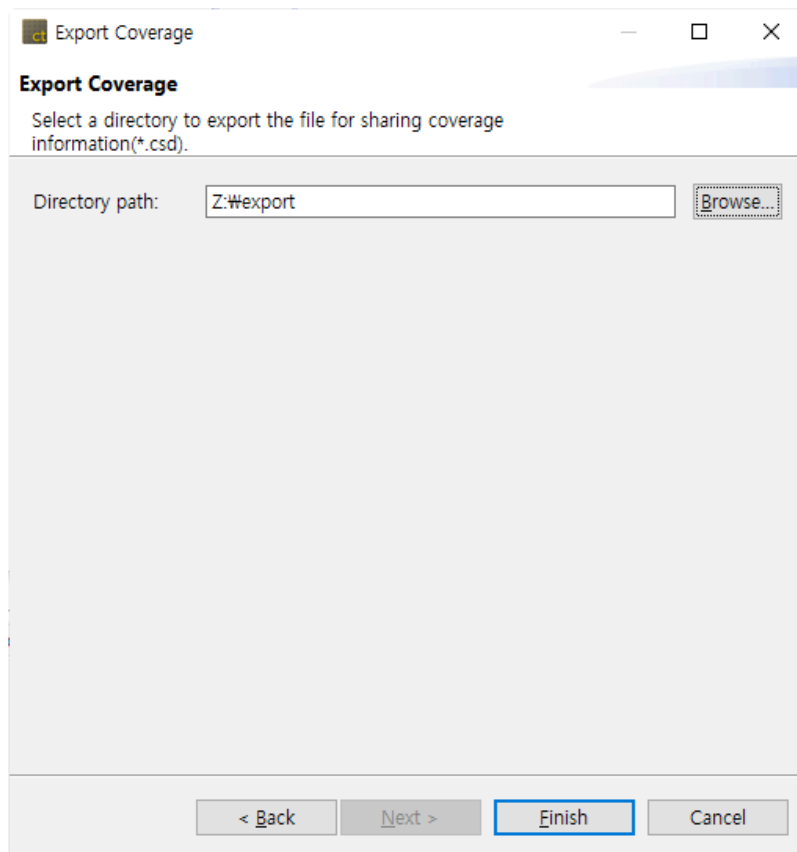


3. Select the project that you want to export coverage information and click [Next].

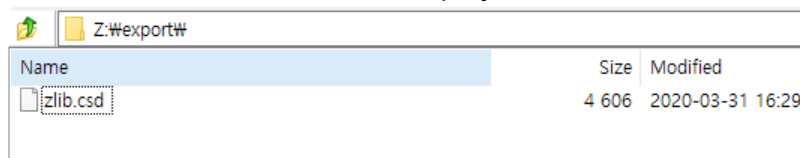


4. Enter the directory path to export and click [Finish].





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

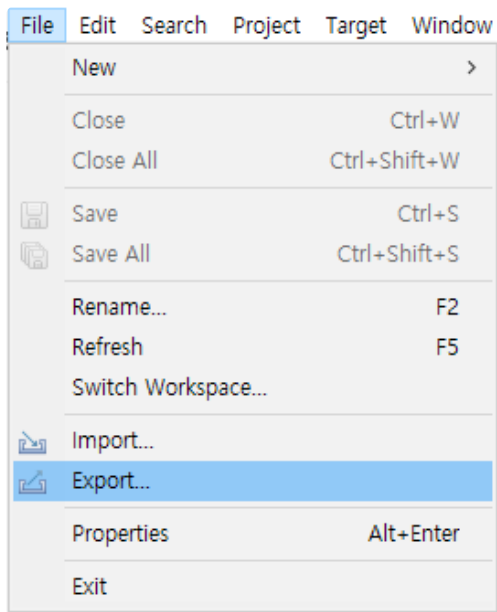


## Metric

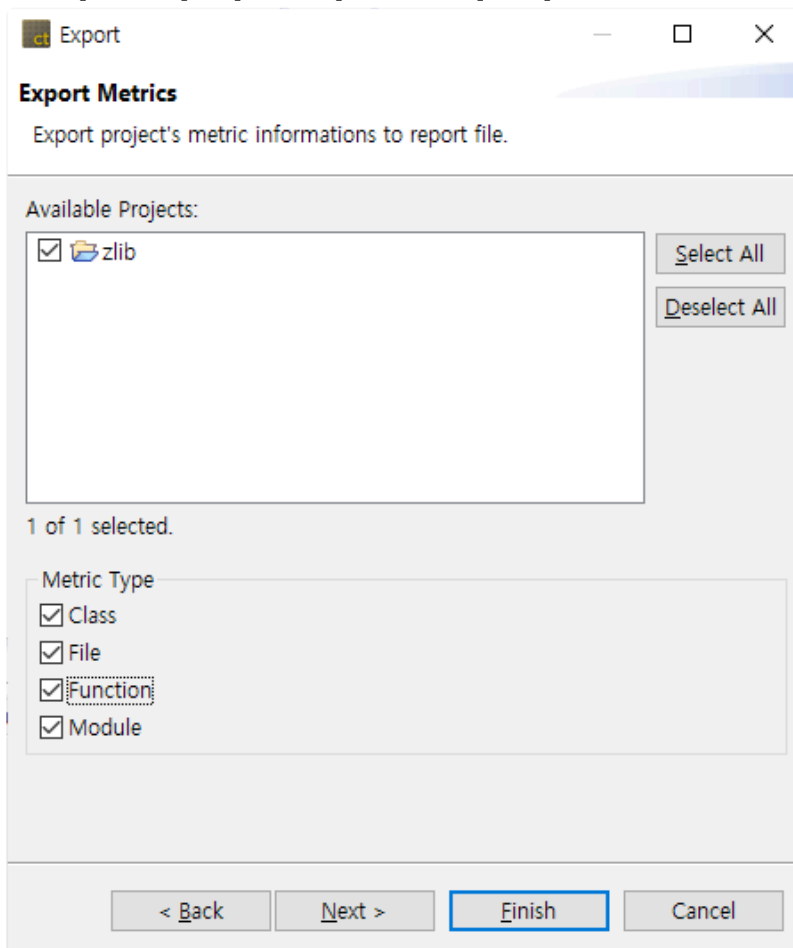
It exports the metric result of the project in a report file.

6. In the main menu, click [File] -> [Export].



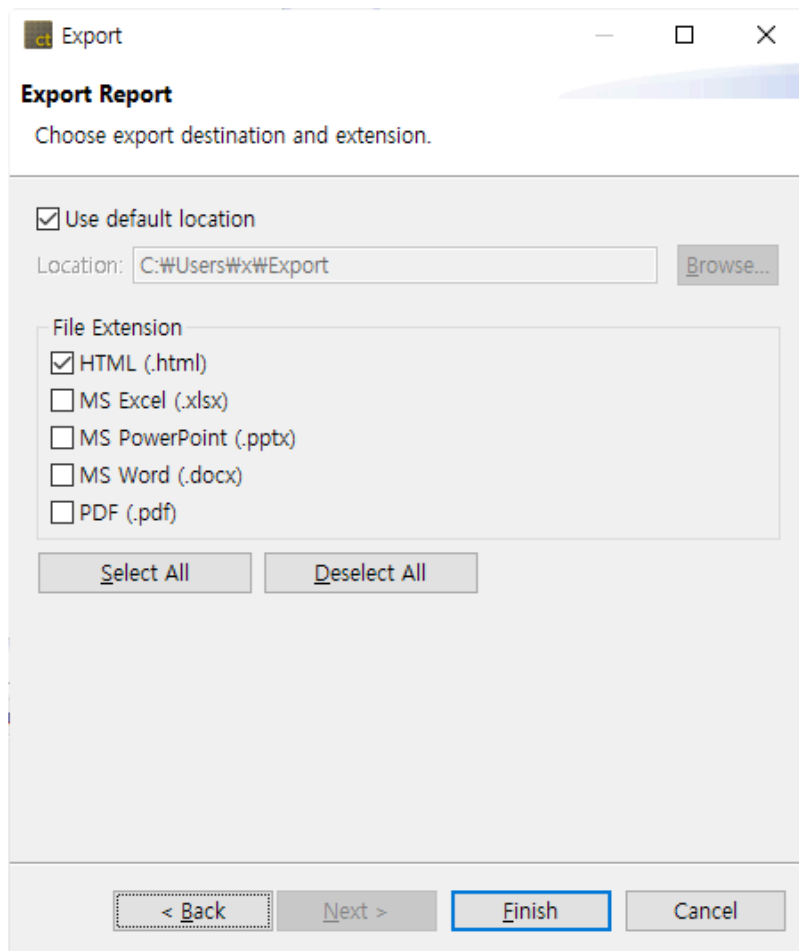


7. Click [Metrics] -> [Metrics] and click [Next].



1. Enter the path where the report is exported to.
2. Select the report format.





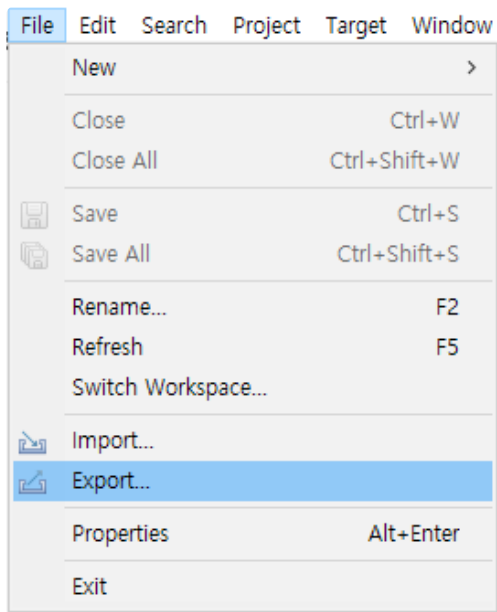
3. Click [Finish].

## Toolchain

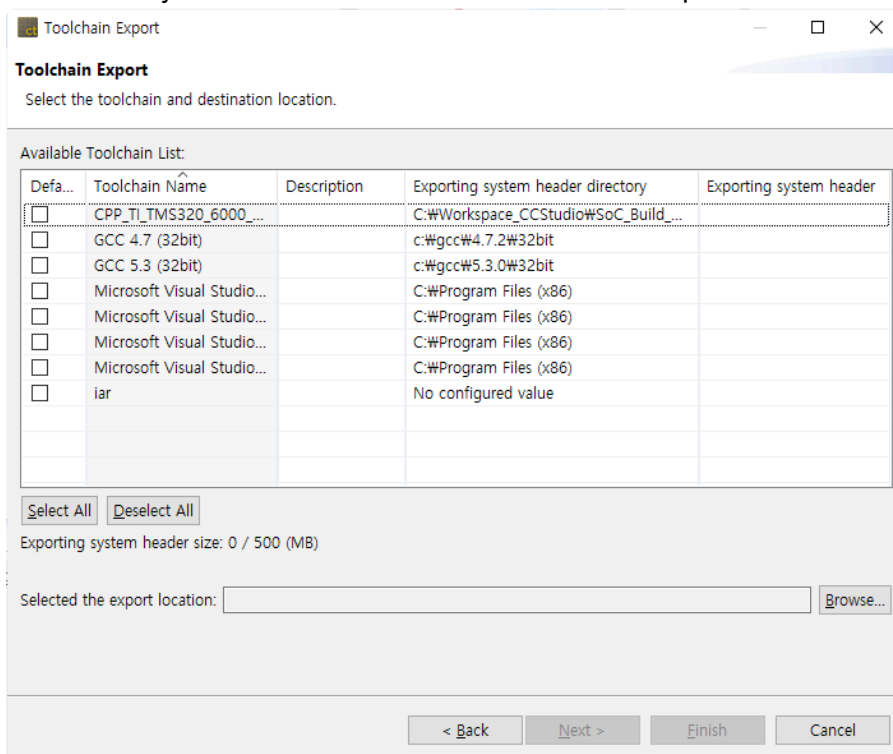
You can export the information of the toolchain created.

1. In the main menu, click [File] -> [Export]. The Export wizard is opened.





2. Click [Preferences] -> [ToolChain] and click [Next].
3. The list of toolchains registered currently is displayed.
4. Check the system header and the toolchain to be imported and enter the export path.



5. Click [Finish].



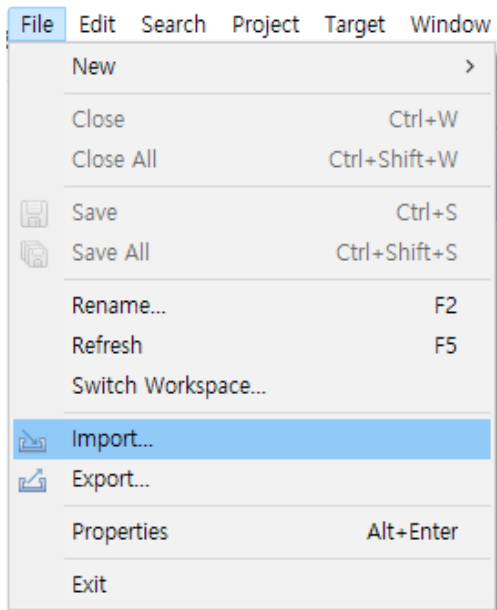
## 16.4. Import

---

By using Import the existing project to workspace function, you can import the project not existed in the workspace into the workspace.

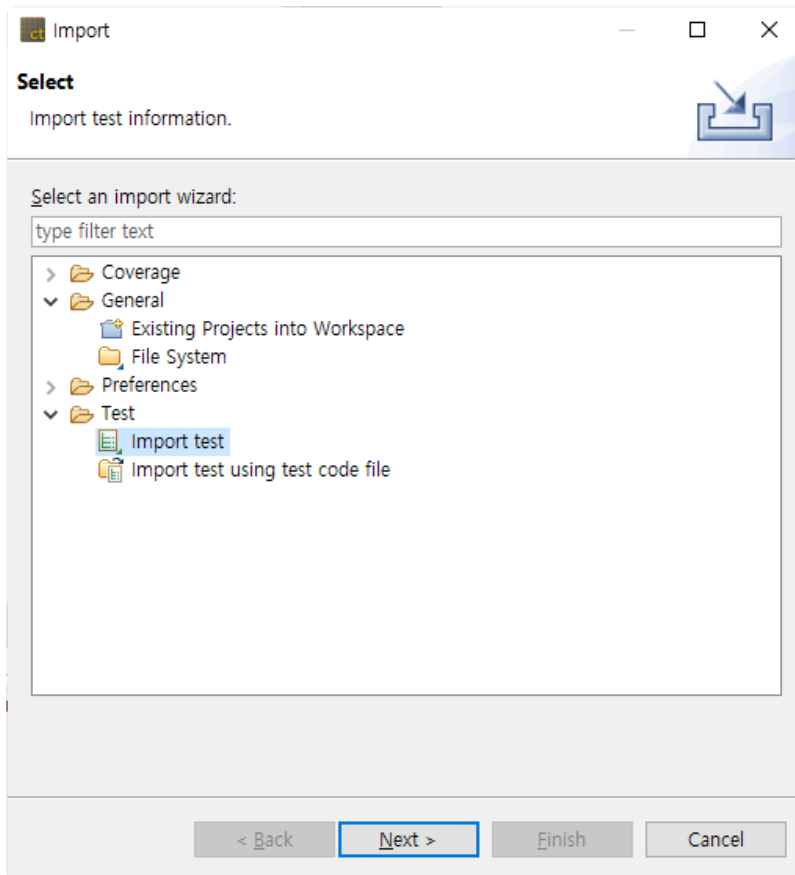
### Import Test

1. Click [File] -> [Import].

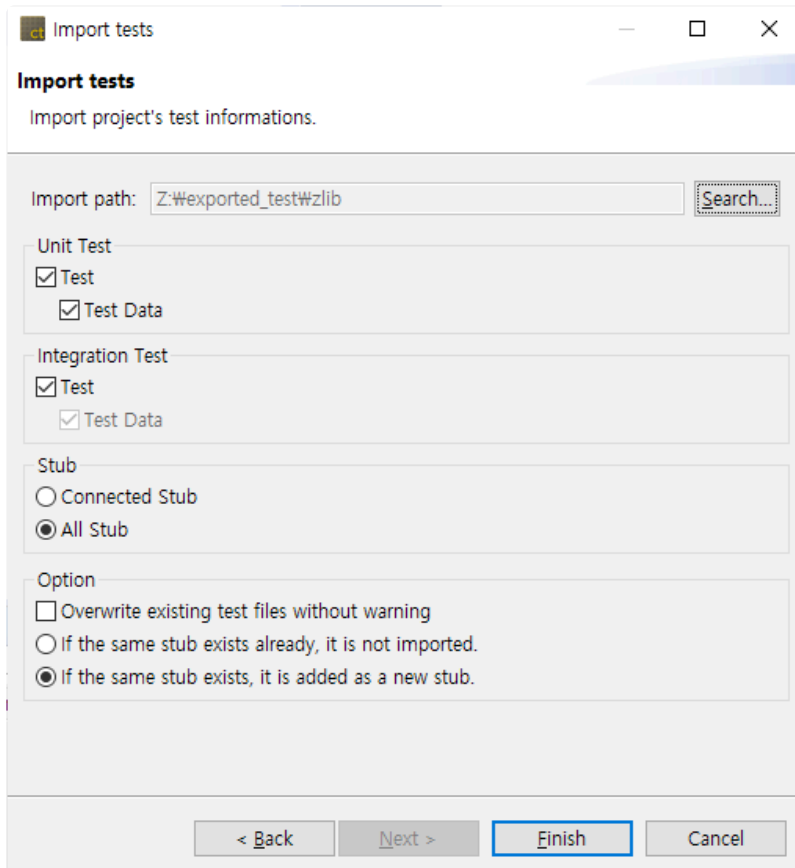


2. Select [Test] -> [Import Test] and click [Next].



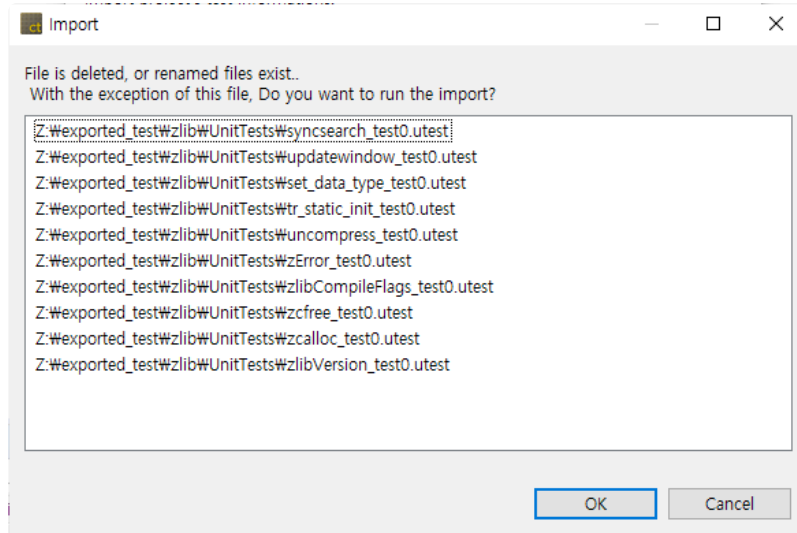


3. Enter the information for the test to be imported and click [Finish].

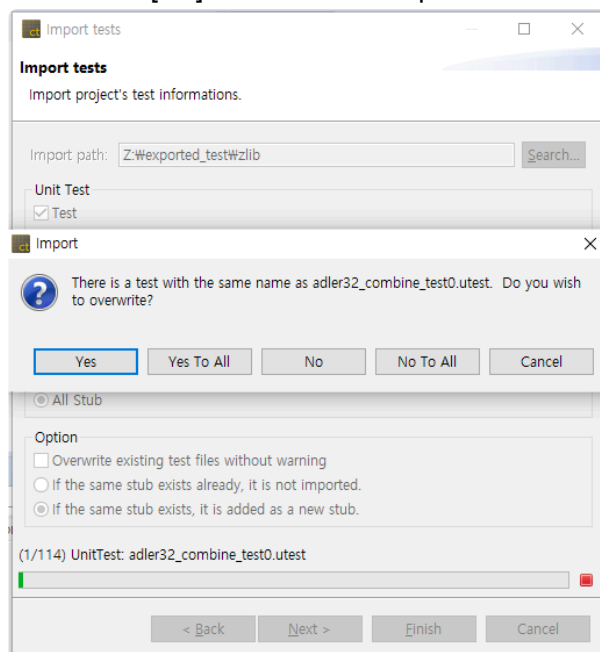




4. If there are items that the name has been changed or the file has been deleted when importing the test information, the notification windows is displayed, allowing you to select whether to import the information except for that items. This notification window is displayed only if the information was exported from 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.



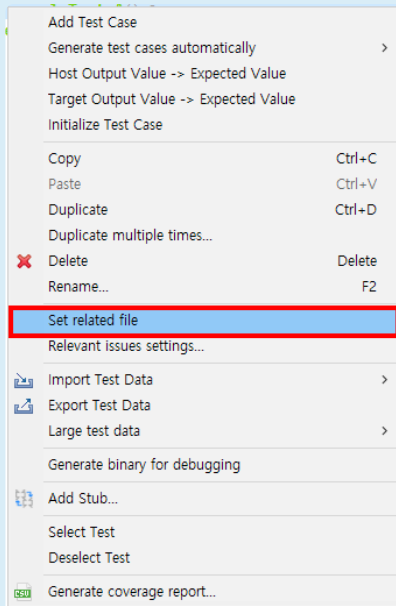
5. If the same test exists in the project, select whether to overwrite the test. Select [Yes] to overwrite and select [No] to cancel the import.



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

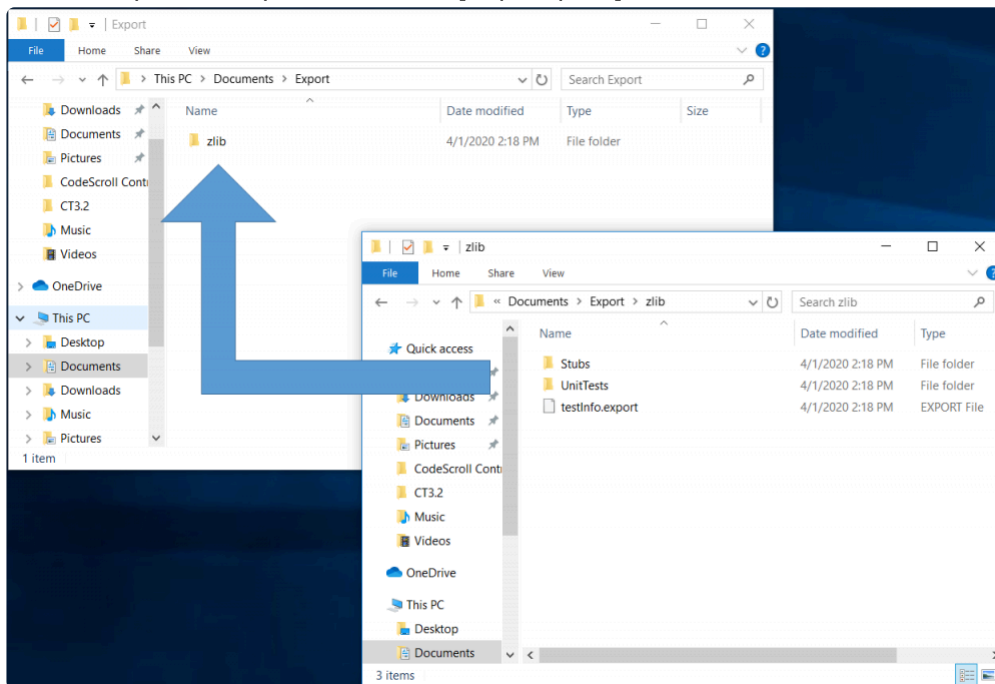
\* The status icon(?) 'the test that cannot find the function to be tested in the source file to be tested' is displayed, the associated file can be set by using the [Set related file] context menu.





## Import information exported from 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 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**

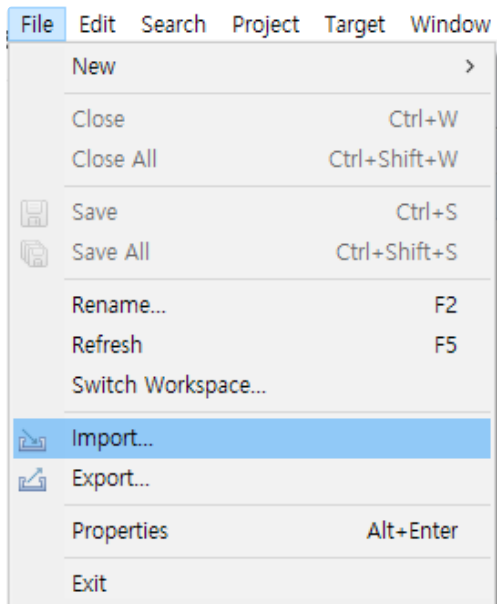


imported.

4. In [Stub] group, select the stub to be imported among [Connected Stubs] and [All stubs].
  - For the stubs created in 2.3 version, or the stubs migrated to 2.6, if the same stubs exist in the project, the import cannot be carried out.
  - For the stubs created in 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.
    - **If you import the test linked with a stub, see Appendix A containing examples.**
  - **There is no feature to link the stubs to the test in 2.3 version, so you need to link directly to the test when migrating to version 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.

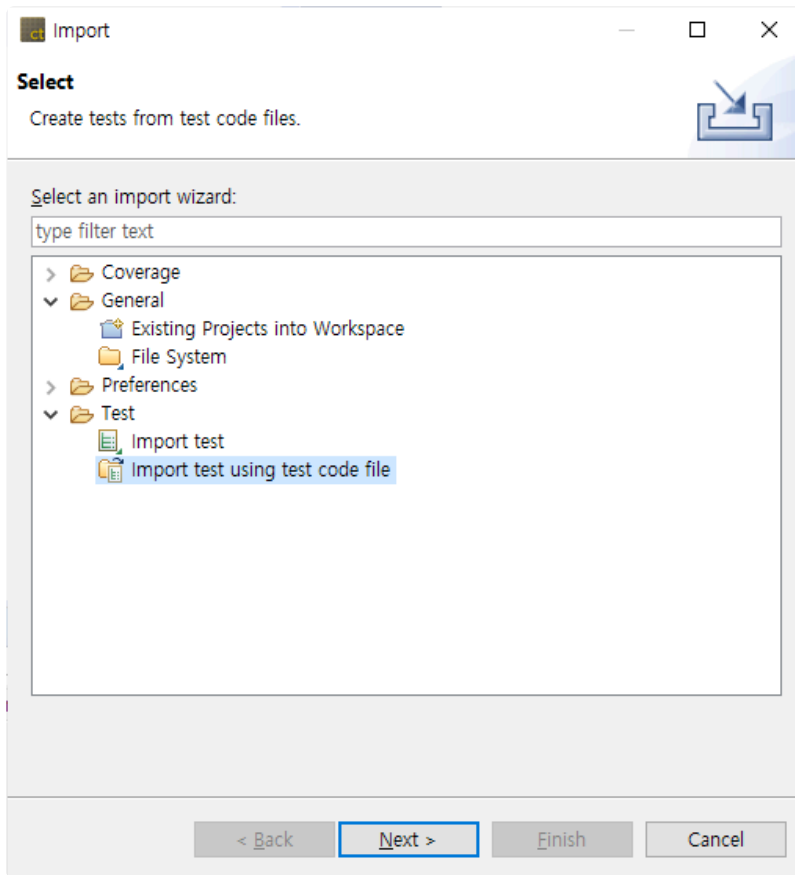
## Import Test from Test Code file

1. Click [File] -> [Import].

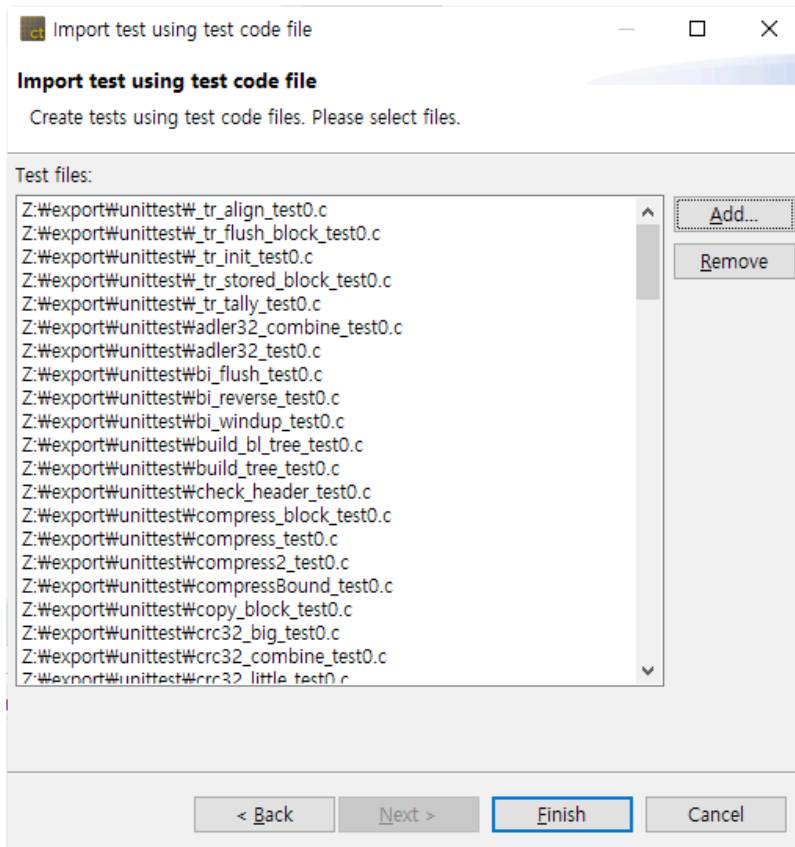


2. Select [Test] -> [Import test using test code file] and click [Next].





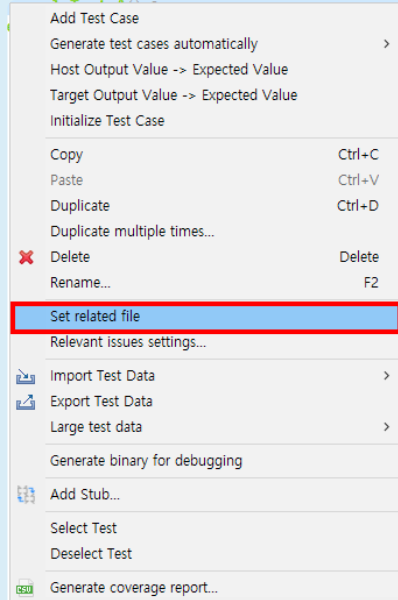
3. Click [Add...] and select the test to be imported.





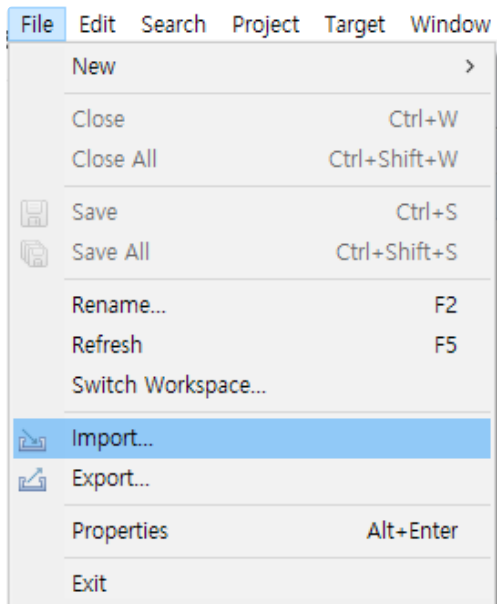
- Click [Finish] to apply the selected test to the selected project.

\* The status icon(?) 'the test that cannot find the function to be tested in the source file to be tested' is displayed, the related file can be set by using the [Set related file] context menu.



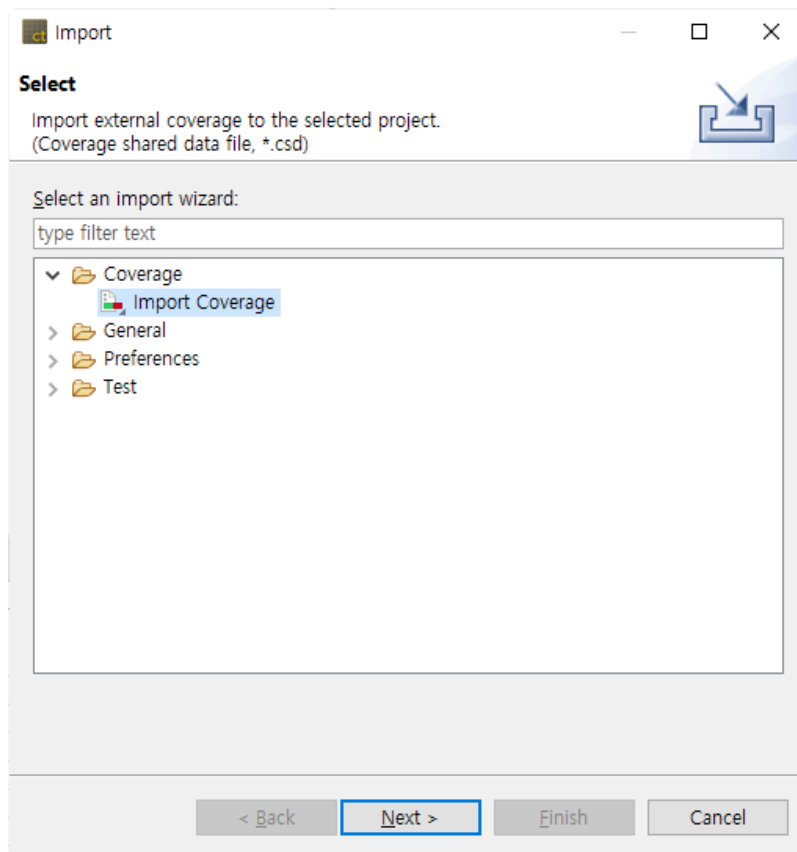
## Import Coverage

- Click [File] -> [Import].

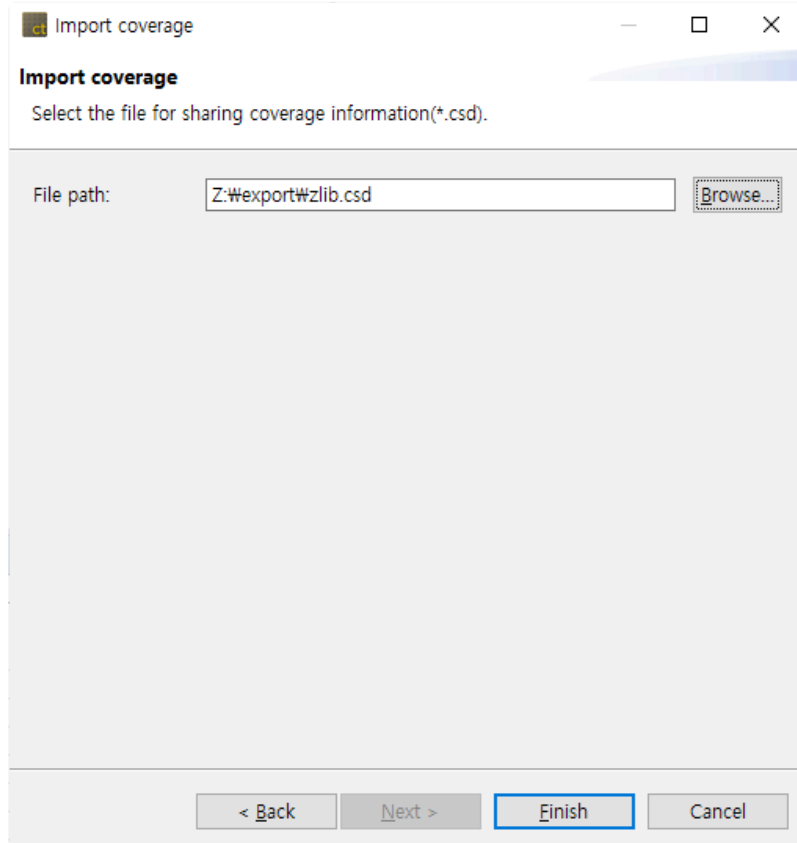


- Select [Coverage] -> [Import Coverage] and click [Next].









3. 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 Coverage view.

Coverage    

Host coverage information of 'zlib' project

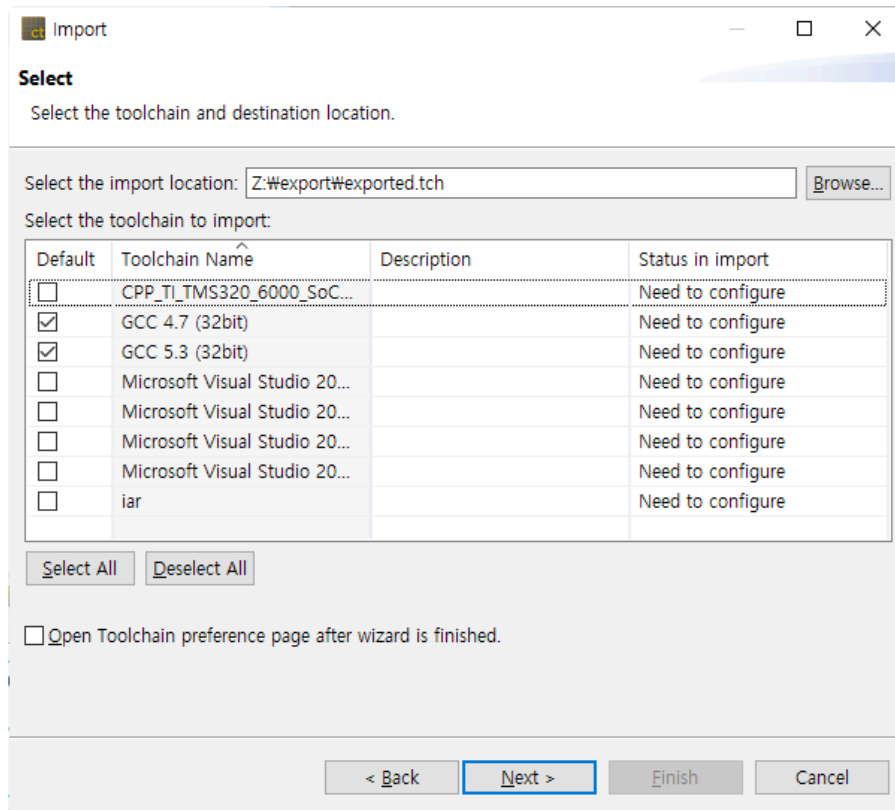
	Target Function	Statement	Branch	MC/DC	Function Call	Funci...	
1	_tr_align(struct internal_state *)	72.72% ...	70.00% ...	40.00% ...	100.00...	Y	
2	_tr_flush_block(struct internal_state ...	54.76% ...	62.50% ...	36.36% ...	70.00% ...	Y	
3	_tr_init(struct internal_state *)	100.00%...	N/A	N/A	100.00...	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	
<b>Total</b>		<b>43.02% (...</b>	<b>28.51% (...</b>	<b>19.07% (...</b>	<b>44.11% (...</b>	<b>100.0...</b>	

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

## Toolchain

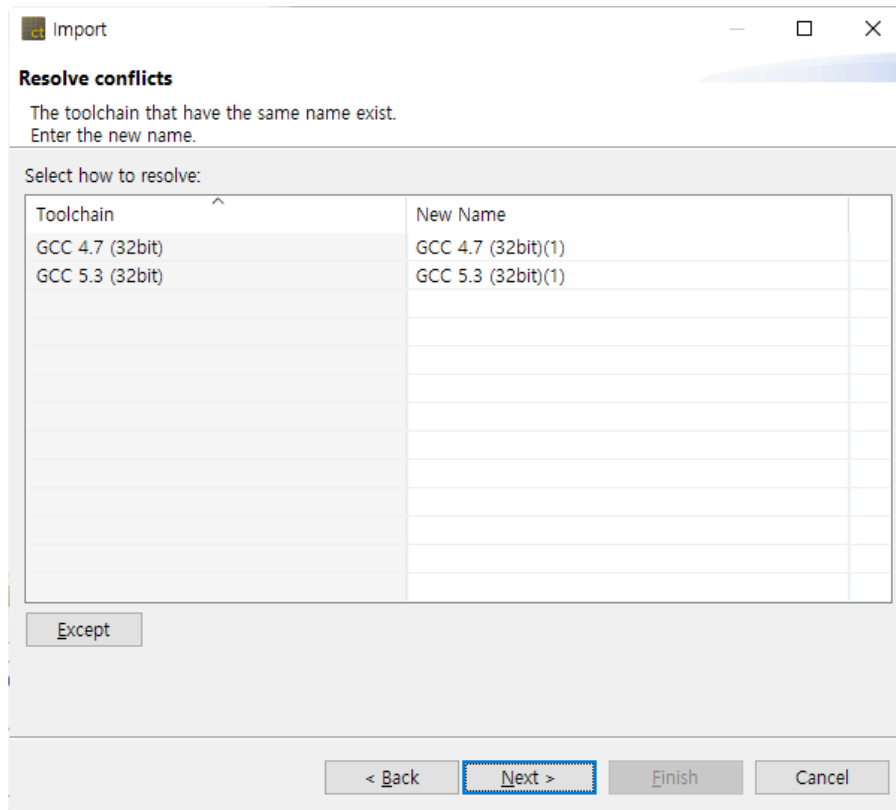
You can import the toolchain information exported by using the Import function.





1. In the main menu, click [File] -> [Import]. The Import wizard is opened.
2. Click [Preferences] -> [ToolChain] and click [Next].
3. If you select the file (\*.tch) to be imported, the list of toolchains included in the selected file are displayed.
4. Select the toolchains to be imported from the toolchain list.
5. If the toolchain to be imported uses the same compiler as the existing toolchain, or 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.



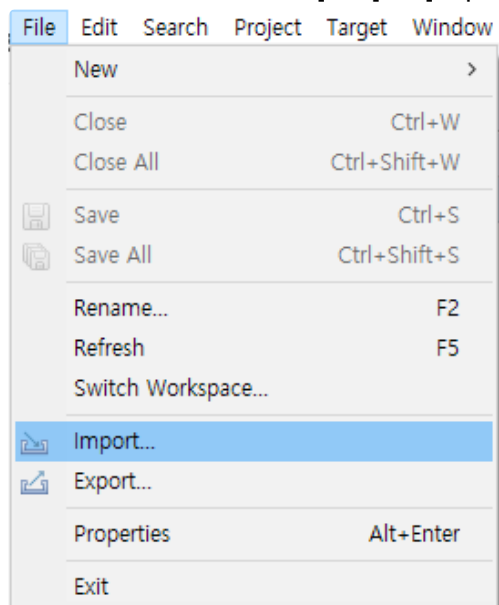


6. Click [Finish].

## Import the existing project into workspace

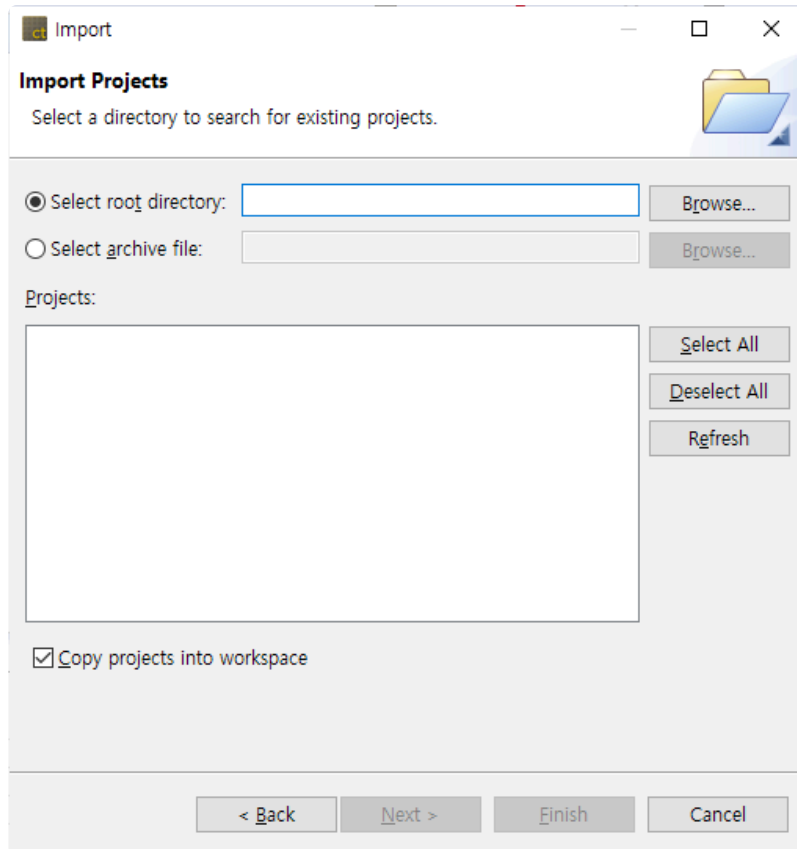
By using Import the existing project to workspace function, you can import the project not existed in the workspace into the workspace.

1. In the main menu, click [File] -> [Import]. The Import wizard is opened.





2. Click [General] -> [Existing Projects into Workspace] and click [Next].
3. Select the Root directory or the Archive file and click [Browse] to find the directory or file containing the project.
4. From the list of projects included in the selected root directory or the archived file, select the project to be imported.



5. Click [Finish].

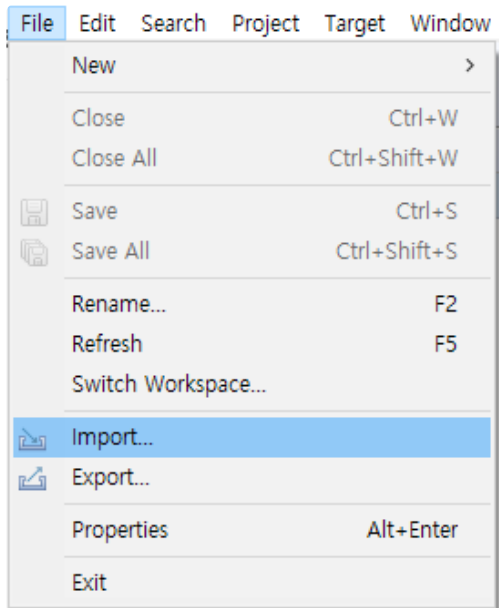
\* If you select [Copy projects into workspace], it copies the project to be imported into the workspace directory. If you don't select it, it connects only the project to be imported to the workspace.

## File System

You can add the source files to the project by using Import File System function.

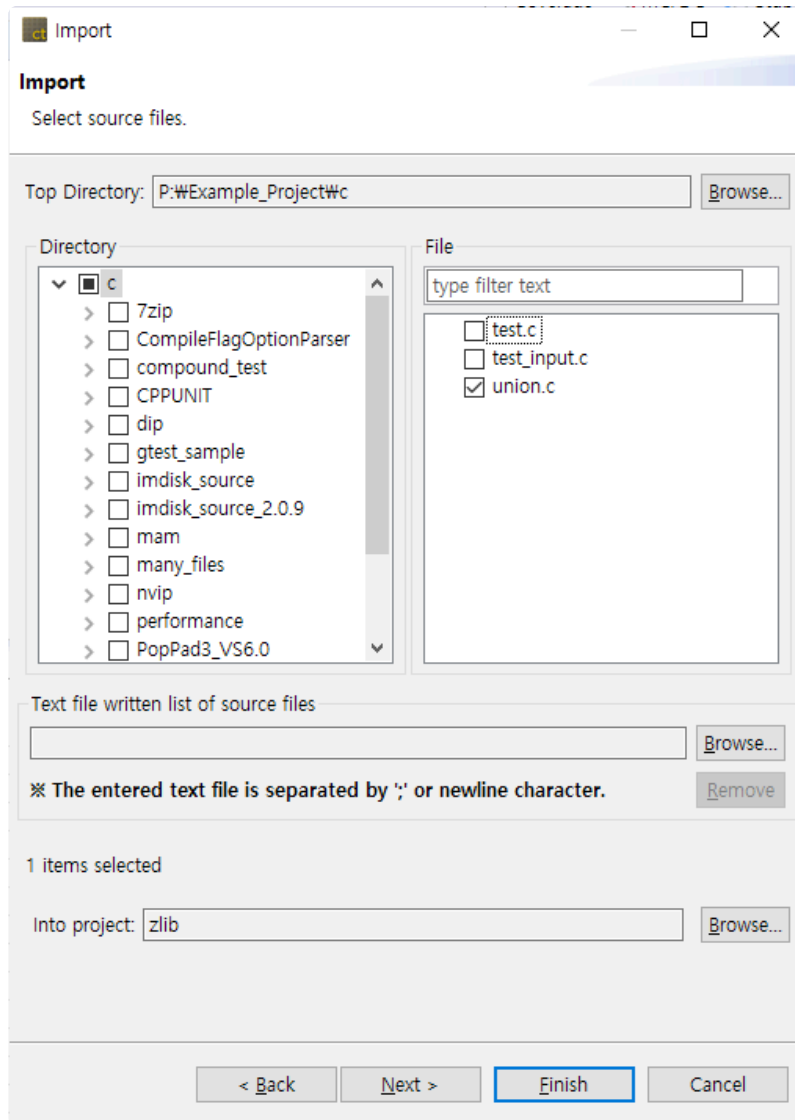
1. In the main menu, click [File] -> [Import]. The Import wizard is opened.





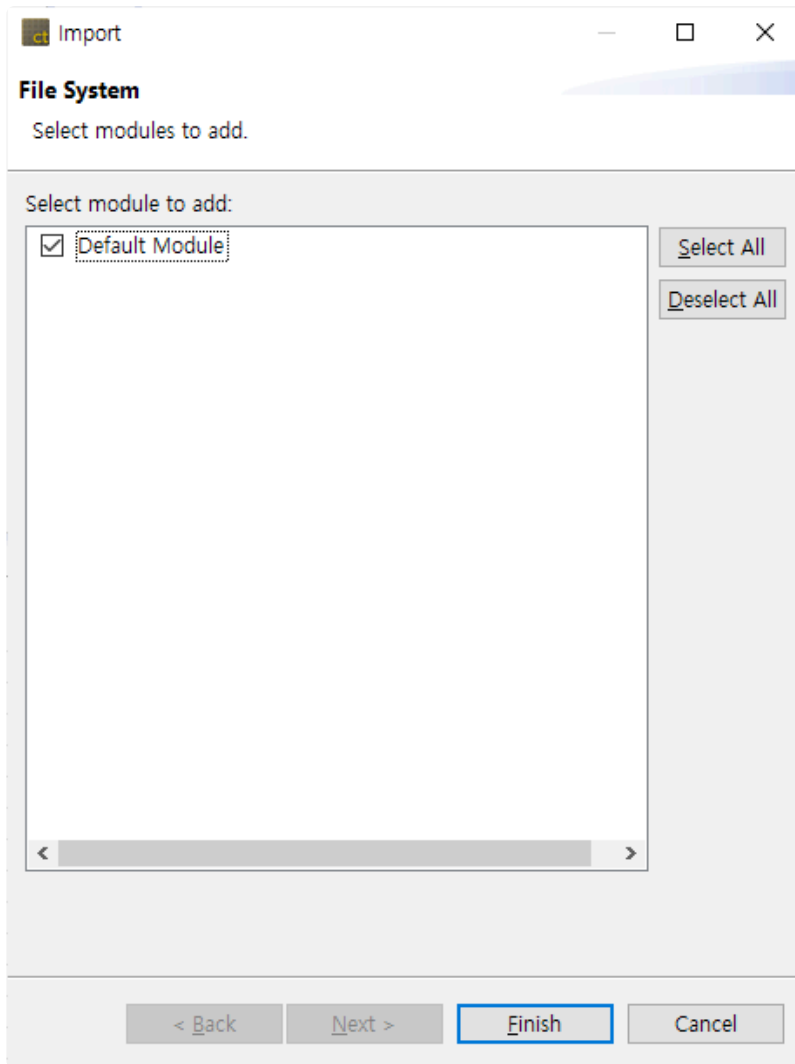
2. Click [General] -> [File System] and click [Next].
3. Select the top-level directory path containing the source file to be added by clicking [Browse...] button.
4. Select the source file to be added.
5. Select the target project that the selected source file will be added to by clicking [Browse...] button.





6. Click [Next].
7. Select the module that the source file will be added from the list of modules included in the project.





8. Click [Finish].

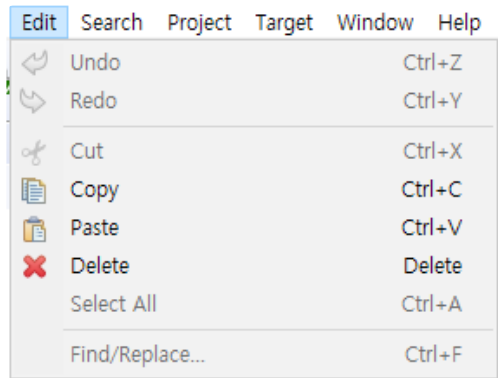
✿ You can also add the source file by, from Windows Explorer, dragging and dropping the source file into the target project or module in Test Navigator view.



## 17. 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.

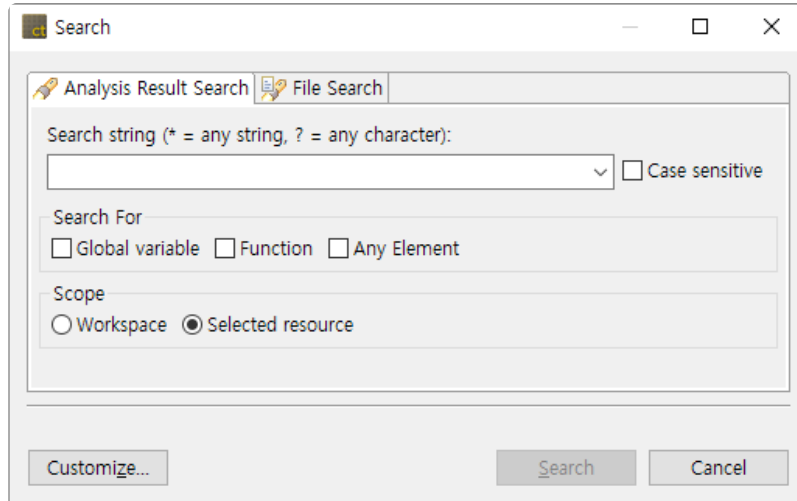


# 18. Search Menu

---

## 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.

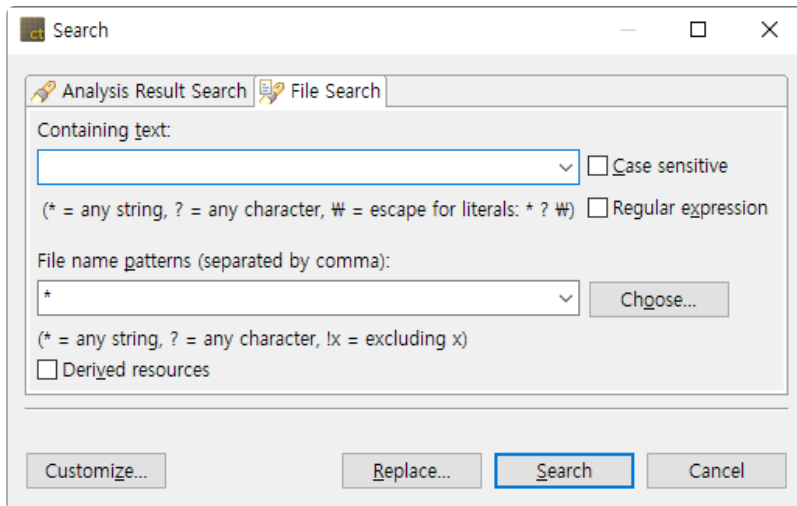
### Search range

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.

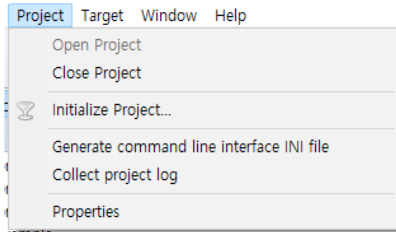
- “\*” Wildcard string: matches the set of characters containing the blank string.
- “?” Wildcard character: matches all characters.



# 19. Project Menu

---

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



## Open Project

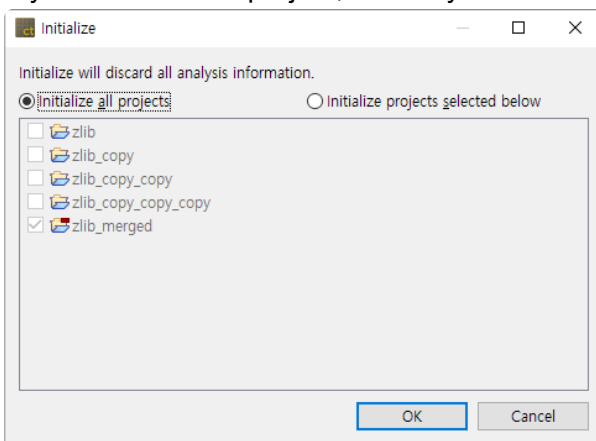
You can open a closed project selected in the Test Navigator view.

## 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

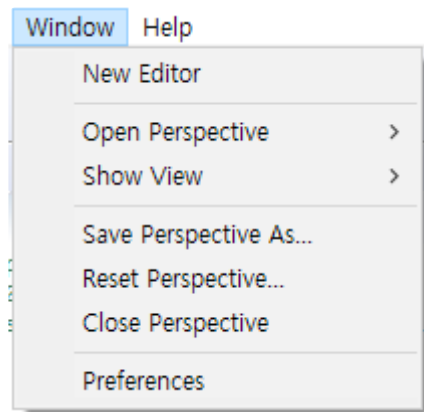
You can see the information or change the settings for the project selected in the Test Navigator view.



## 20. 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.



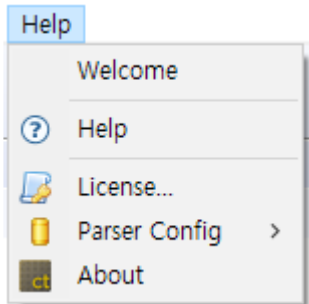
## Preferences

You can check or change the settings currently applied to the tool.  
For more information, please refer to [Preferences](#)



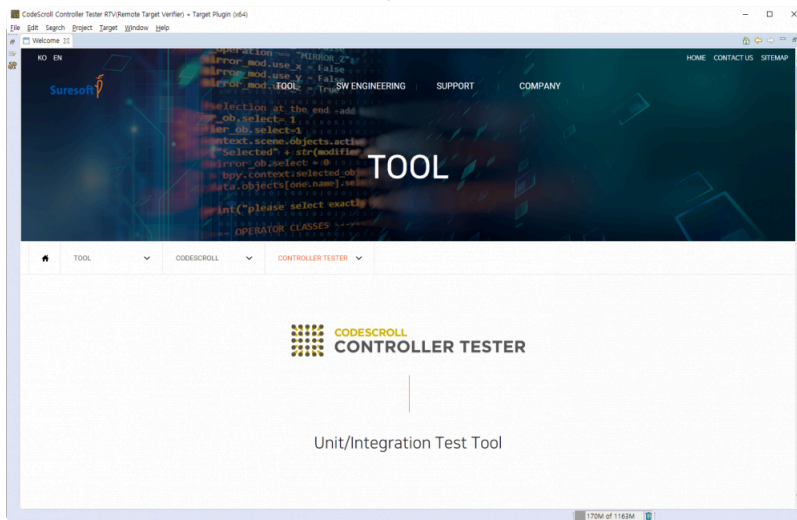
## 21. Help Menu

In the Help menu, you can set the license, and modify or revert the parser config and check the product information.



### Welcome

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



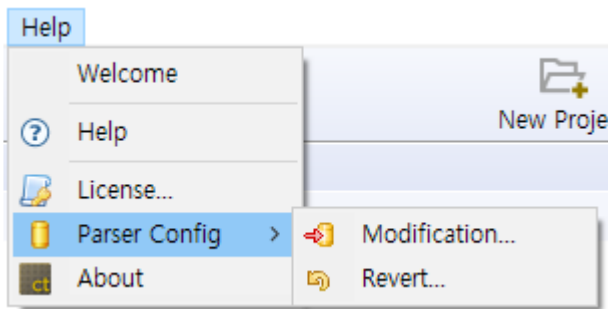
### Modify Parser Config

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

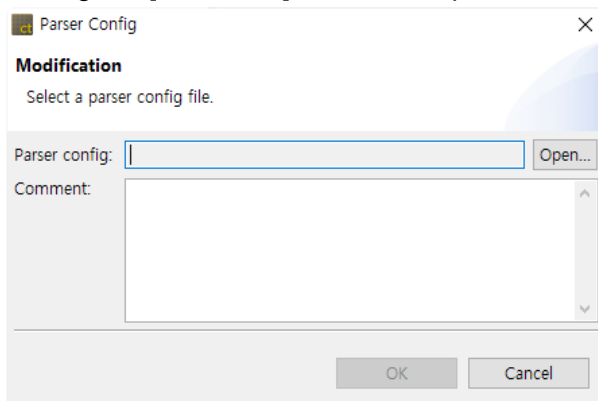
#### How to use

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

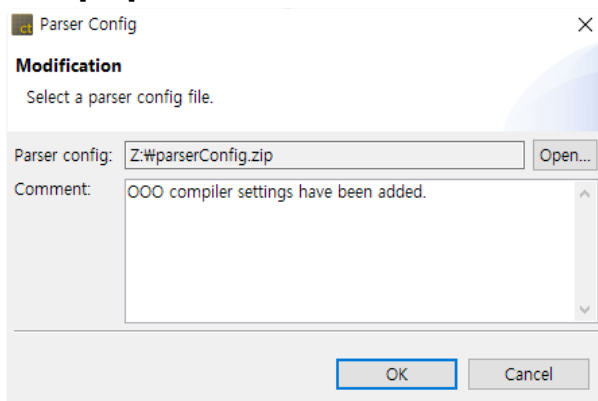




2. Click [Open...] to select the compressed analysis setting file and enter the changes of analysis settings to [Comment] text as an option.



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”.

Z:\#parserConfig.zip#		
Name	Size	Packed Size
parserConfig	241 261	40 758

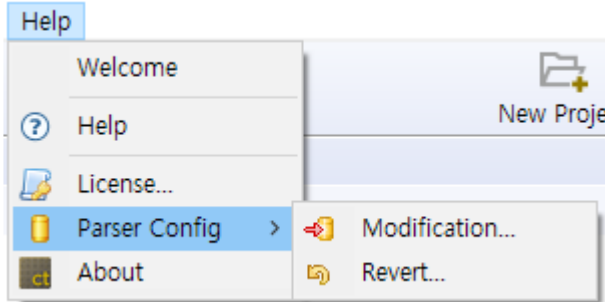


## Revert the Parser Config

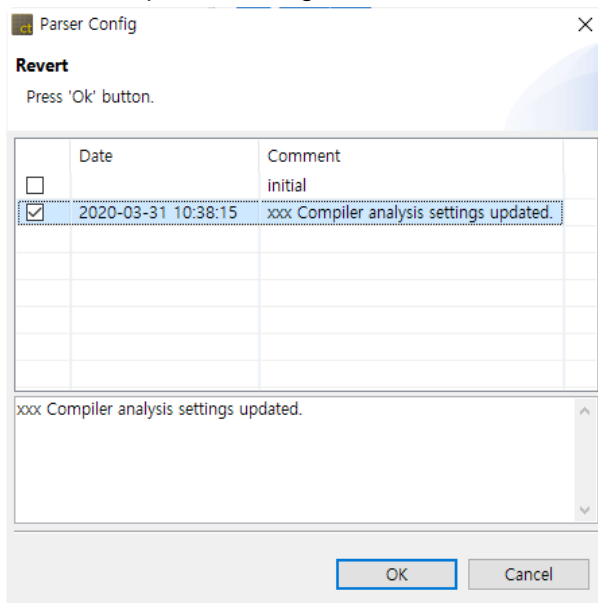
You can revert the parser config by selecting one among the previous parser configs.

### 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.



## 22. 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



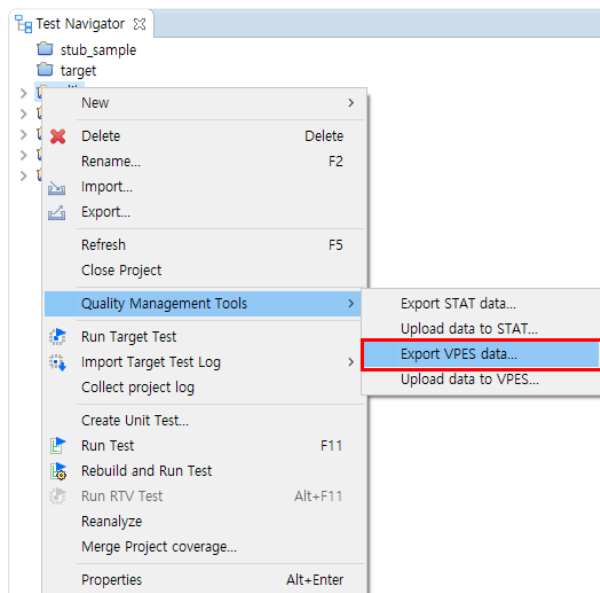
when executing the test must be added as an exception to the firewall setting to run normally.

## 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

## Upload data to VPES

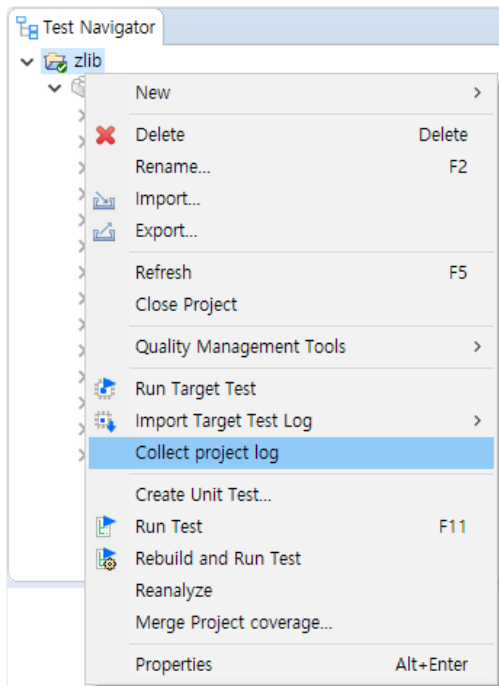


When you select and right-click a project, [Upload data to VPES...] is indicated in the menu. For this feature, please contact Technical support.

## 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.

## Technical support

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

- [support@suresofttech.com](mailto:support@suresofttech.com)
- +82-2-6472-2800



## 23. 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.



## 23.1. 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.

Name	Value
Toolchain Name	Microsoft Visual Studio 2010 (32bit)
Status	This toolchain is supported.
C Compiler	C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\bin\cl.exe
System Header(C Compiler)	C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\ATLMFC\I...
Library(C Compiler)	C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\ATLMFC\LI...
C++ Compiler	C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\bin\cl.exe
System Header(C++ Compiler)	C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\ATLMFC\I...
Library(C++ Compiler)	C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\ATLMFC\LI...

Name:

Description:

### 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.

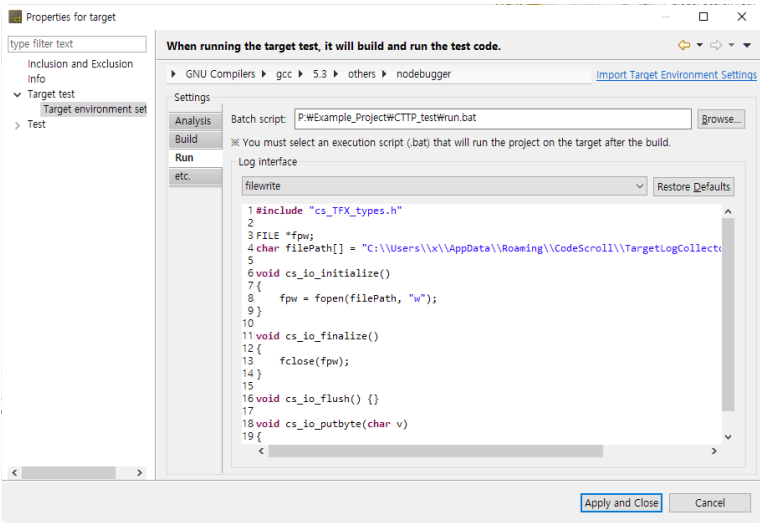
Required settings completed category	Description
None	Test run impossible
Analysis	If you click [Run], it overwrites the original source code with the test code. To perform the test, the user must manually build and run the test.
Analysis, Build	If you click [Run], it builds the test code. To perform 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
<code>void cs_io_initialize()</code>	Initial function for transfer
<code>void cs_io_finalize()</code>	Transfer end function
<code>void cs_io_flush()</code>	Remaining data transfer function
<code>void cs_io_putbyte(codescroll_byte v)</code>	1-byte data transfer function

## Import target log automatically

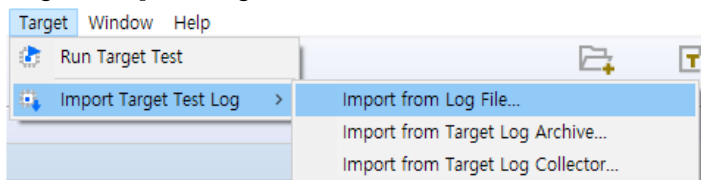
The target log, which is the result of the test execution, can be automatically imported through the target log collector. For the target log collector settings, please refer to [Target Log Collector] in [Target Log Collector](#) and [Target Test Preferences](#).

## Import target log manually

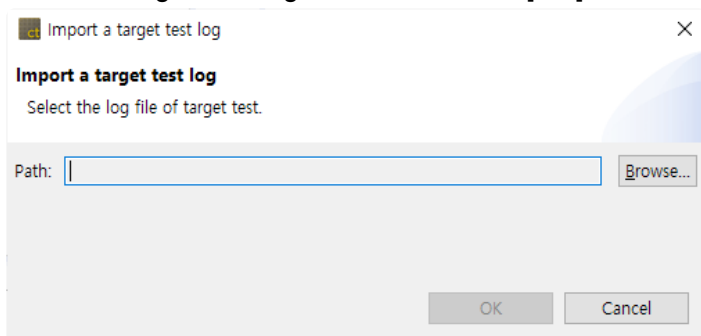
If the target log cannot be imported automatically, you can import the target log manually.

### Import from a log file

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



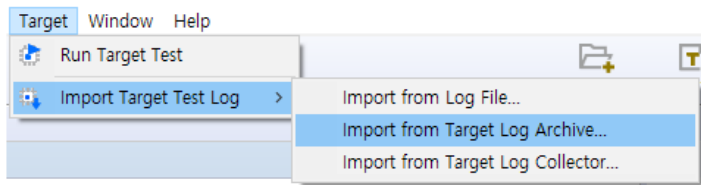
2. Select a target test log file and click the [OK] button.



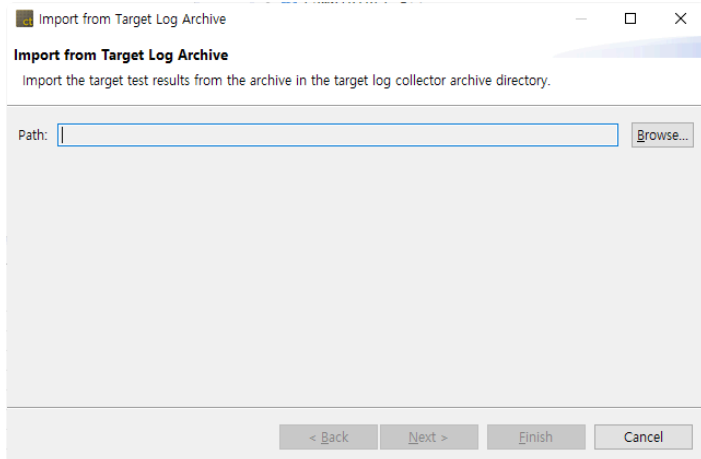
### Import from a target log archive

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

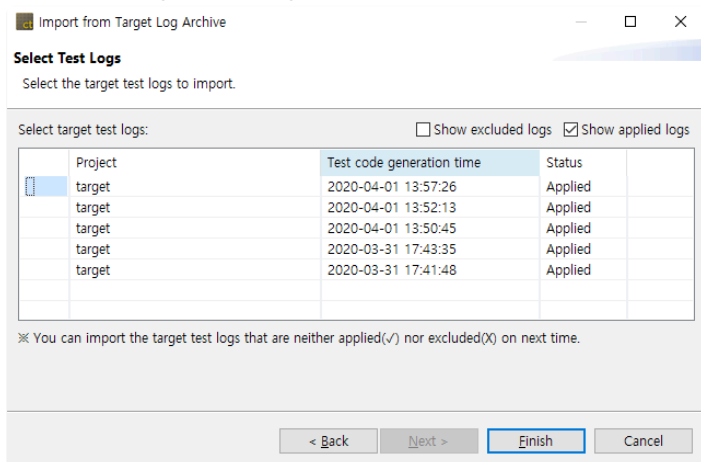




2. Select a target log archive file and click the [Next] button.

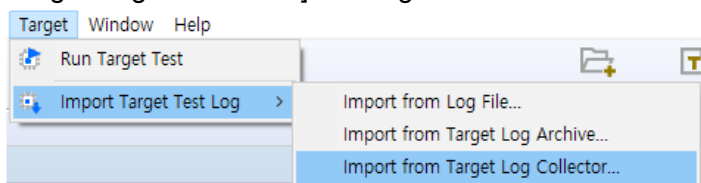


3. Check a target test log file and click the [OK] button.



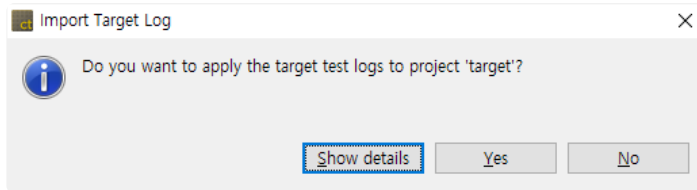
## Import from target log collector

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 Test] -> [import Target Test Log] -> [Import from Target Log Collector...] in the global menu.

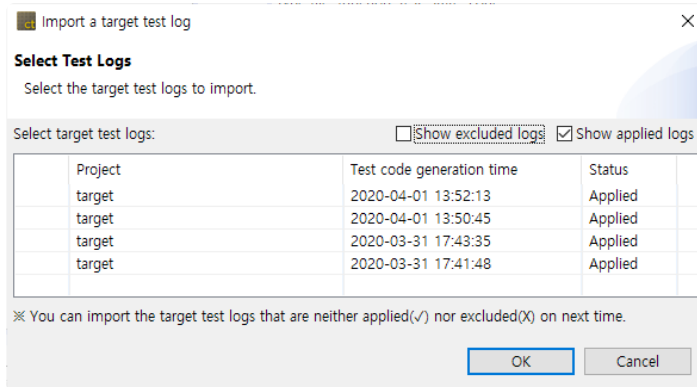




- Click the [Yes] button to import all the new target test logs or click the [Show details] button.



- 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.

Coverage						
Target coverage information of 'target' project						
	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.



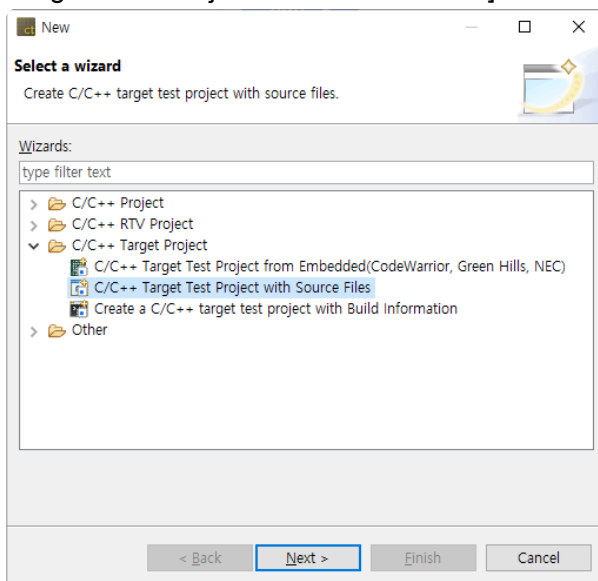
## 23.2. C/C++ Target Test Project

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

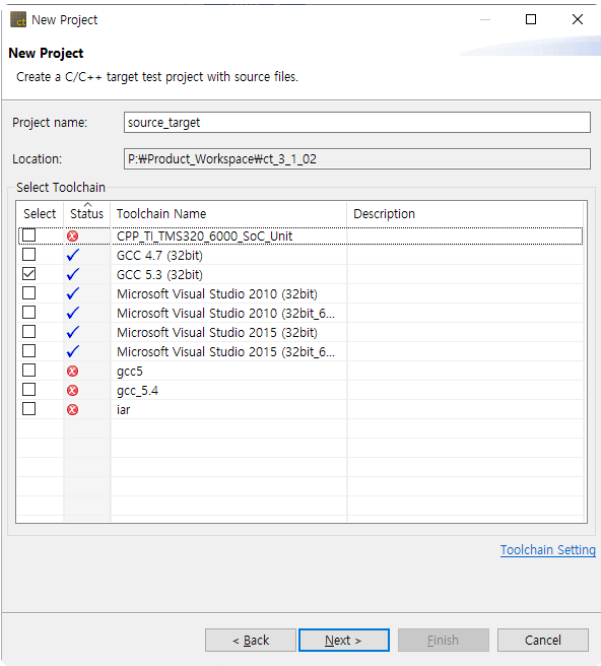
#### C/C++ Target Test Project with Source Files

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

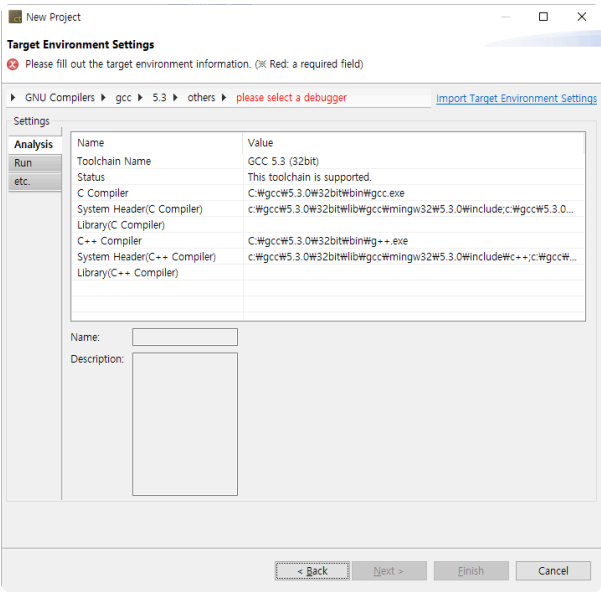


2. Enter a project name and select a toolchain to use.



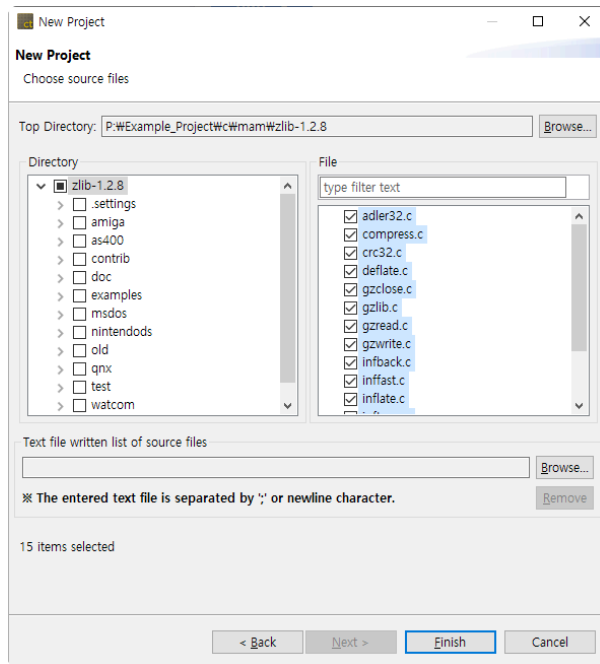


3. After selecting the basic information of the target environment, enter the detailed settings for each category and click the [Next] button.



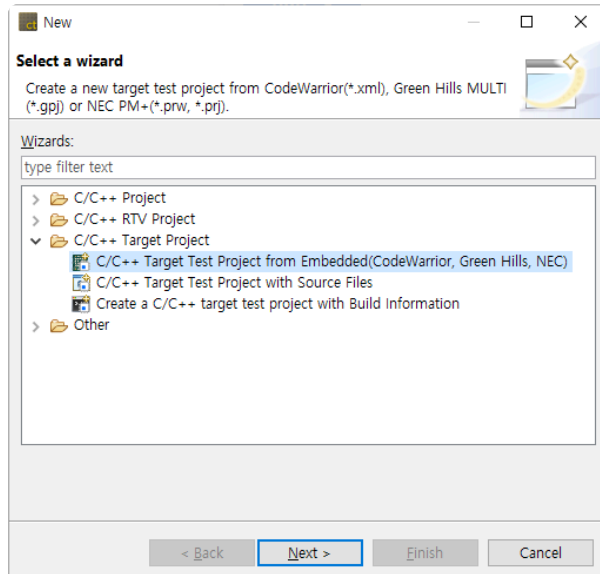
4. Select the source files under test and click the [Finish] button.





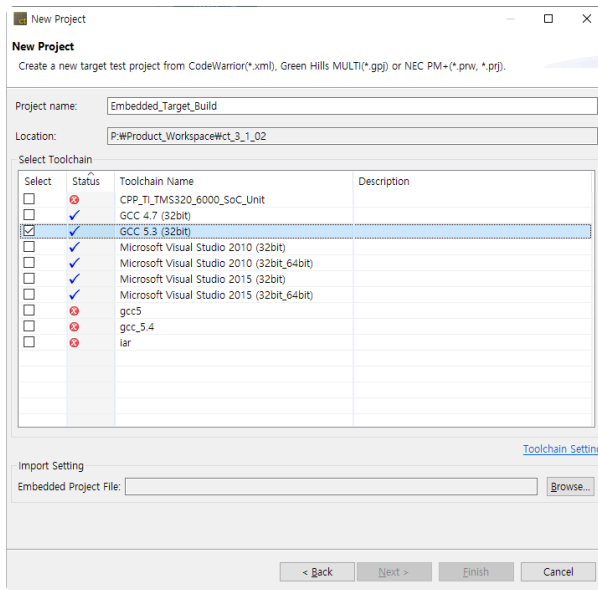
## C/C++ Target Test Project from Embedded(CodeWarrior, Green Hills, NEC)

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)].

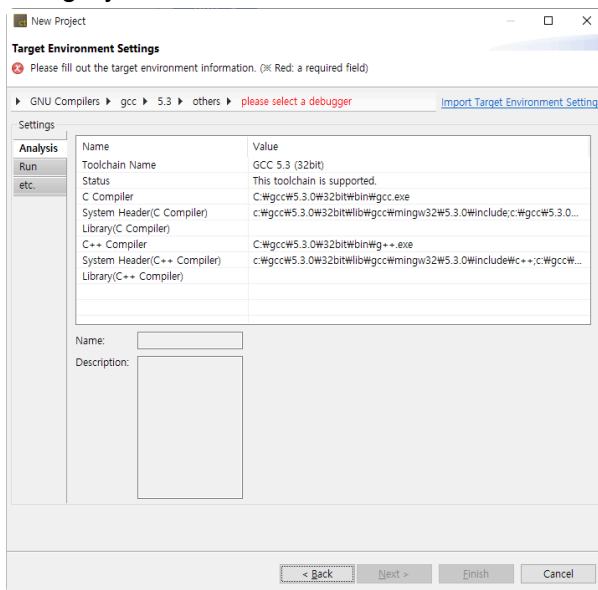


2. Enter a project name and select a toolchain to use. After selecting the toolchain, import the embedded project file to test. Currently, embedded projects supported by Controller Tester are CodeWarrior, Green Hills, and NEC.





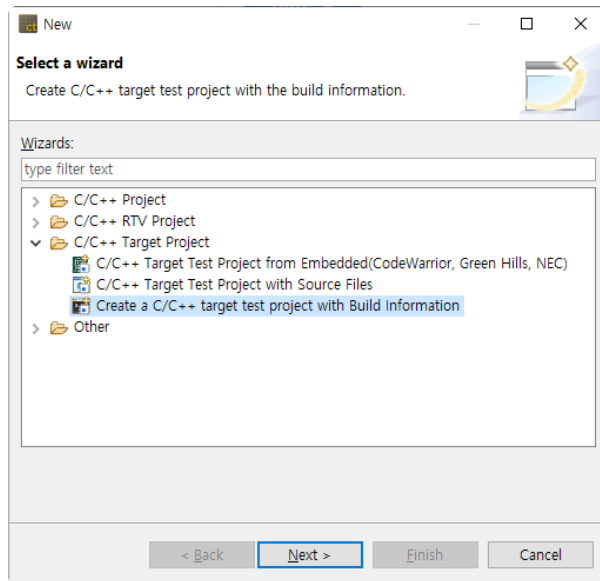
3. After selecting the basic information of the target environment, enter the detailed settings for each category.



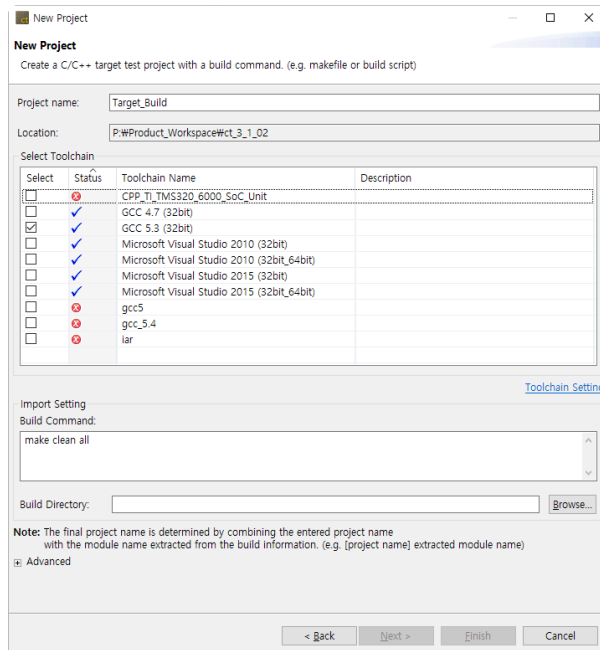
## Create a C/C++ Target Test Project with Build Information

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].



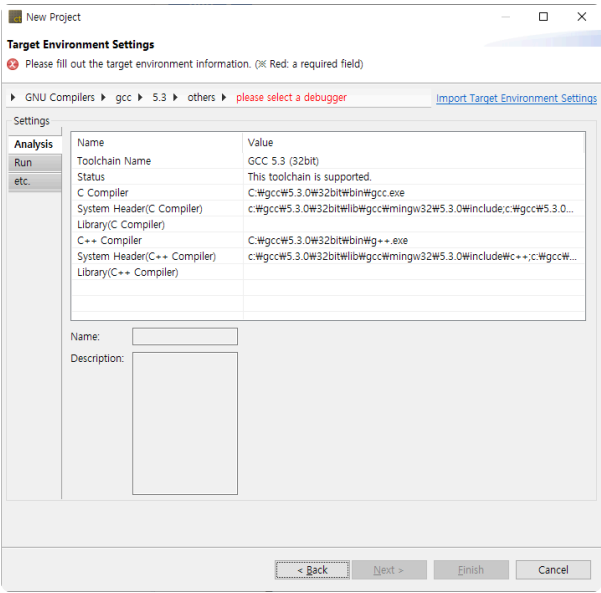


2. Enter a project name and select a toolchain to use. After selecting the toolchain, Specify the path of the created Makefile and build script as the build directory and write the build command. You can choose a script file to set the build-environment from the [Advanced] menu.



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.



## 23.3. 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(syntax, 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.

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

### Testcase run

You can choose how to run the test case.

<b>Run at once</b>	Loads and runs all test cases on target at once. Each test case is affected by the previous test case run results.
<b>Run one by one</b>	Repeats loading and running each test case on target. It takes longer than running at once but uses less memory.



## 23.4. 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.

<b>Protocol</b>	TCP, UDP, UART
<b>File format</b>	Target log(text), memory dump(Hex, Intel HEX)

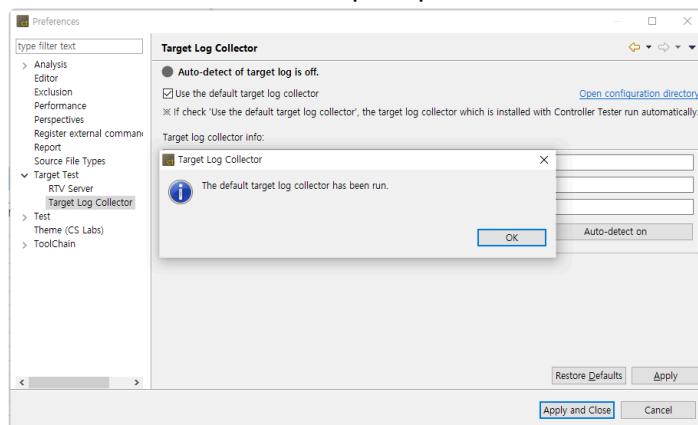
### Installation

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.

<b>[LogReceiveServer]</b>	Settings for receiving target test results
<b>port</b>	TCP, UDP communication port
<b>protocol</b>	Communication protocol for receiving target test results



<b>timeout</b>	Connection timeout
<b>lastString</b>	String representing the end of the data
<b>serialPort</b>	Serial communication port (Windows: COM#, Linux: /dev/ttyS#)
<b>baudRate</b>	Serial communication settings
<b>dataBits</b>	Serial communication settings
<b>stopBits</b>	Serial communication settings
<b>parity</b>	Serial communication settings
<b>flowControl</b>	Serial communication settings
<b>[ScanLog]</b>	Settings for file scanning
<b>dir</b>	Directory to scan
<b>fileExtension</b>	File extensions to scan(Scan all files if blank)
<b>[LogSendServer]</b>	Settings for communication with Controller Tester
<b>port</b>	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.

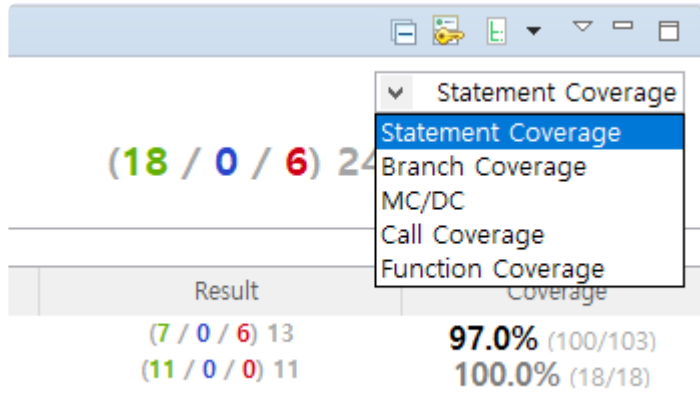


## 23.5. Target Test Results

---

### Coverage

You can change the type of coverage to be displayed in the coverage menu of the [Unit Test] view.

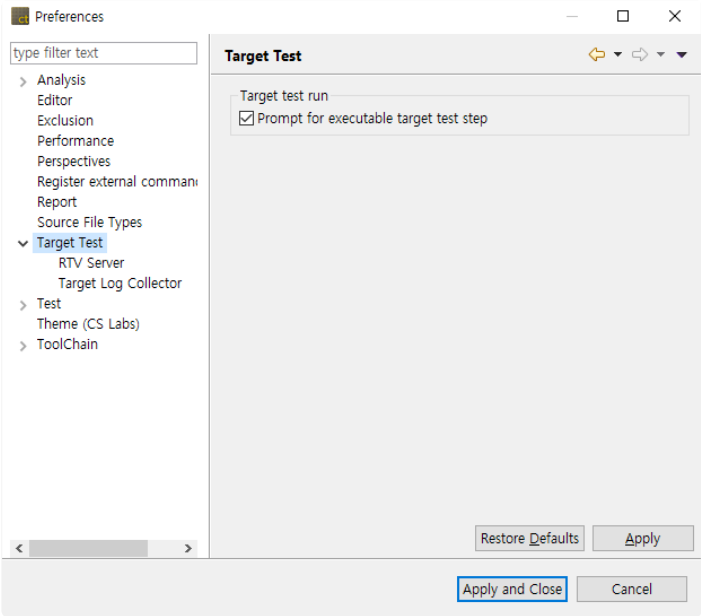




# 23.6. Preferences

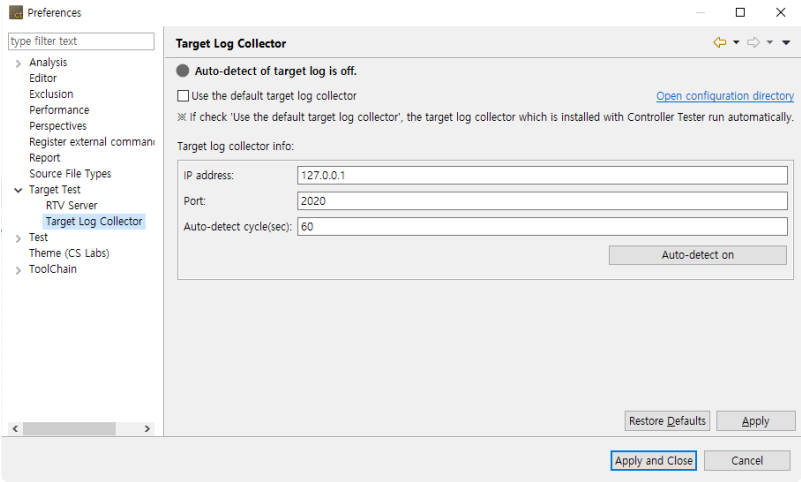
## Target Test

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 [Auto-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).



## 24. CODESCROLL RTV(Remote Target Verifier)

---

### Introduction

Controller Tester RTV helps you perform automated testing in your target environment.

After installing the Controller Tester RTV server in your target environment and completing the default configuration, you can use the Controller Tester RTV client to perform the target test as easily as you would in a hosted environment.

For information on how to install the Controller Tester RTV server in your target environment, please contact the technical support contact on page 2 of this document.

### Supported Operating Systems

Server: Linux (RHEL, Ubuntu, Debian, Fedora based)

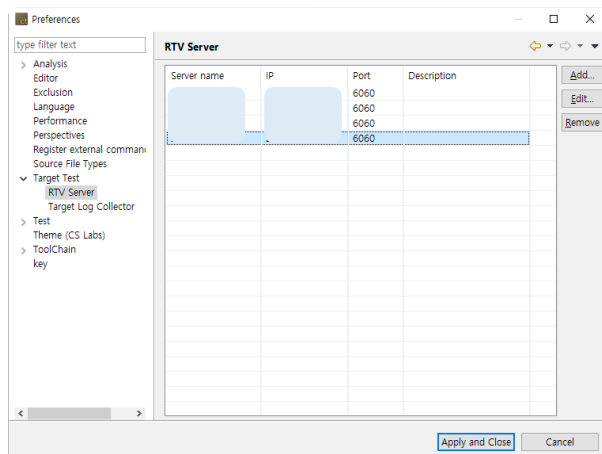
Client: Windows 7/10



## 24.1. RTV Server Settings

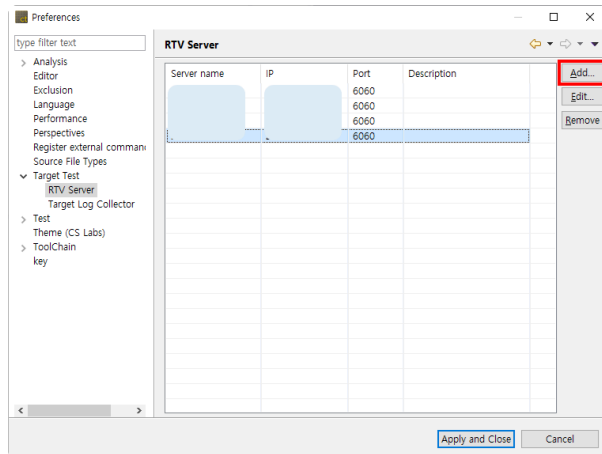
In order to use Controller Tester RTV, you need to add a target server with Controller Tester RTV server installed.

The target server can be added in [Window] -> [Preferences] -> [Target Server].



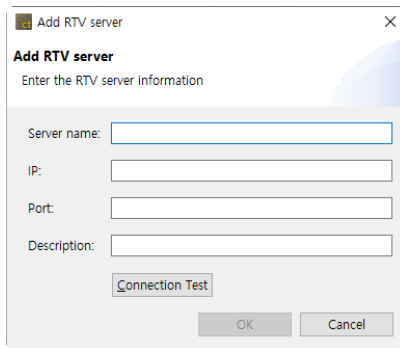
### Add a target server

1. Click [Add] in [Window] -> [Preferences] -> [Target Server].



2. In the [Add Target Server] window, enter the target server information.





**Add RTV server**

Enter the RTV server information

Server name:

IP:

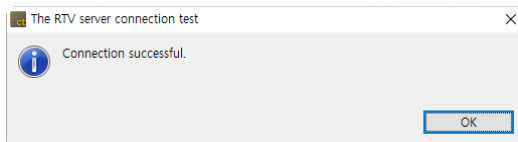
Port:

Description:



The server name only accepts characters that can be used as Windows file names and can not be modified after adding the target server.

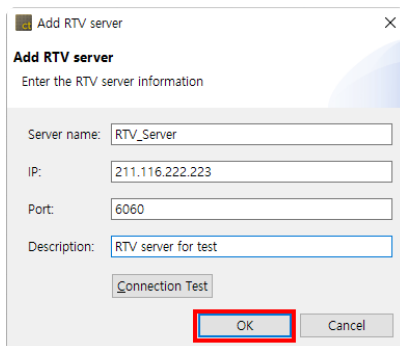
- Click the [Connection Test] button to check the connection status with the target server you entered.



**The RTV server connection test**

Connection successful.

- If the connection test with the target server is successful, click the [OK] button to add the target server.



**Add RTV server**

Enter the RTV server information

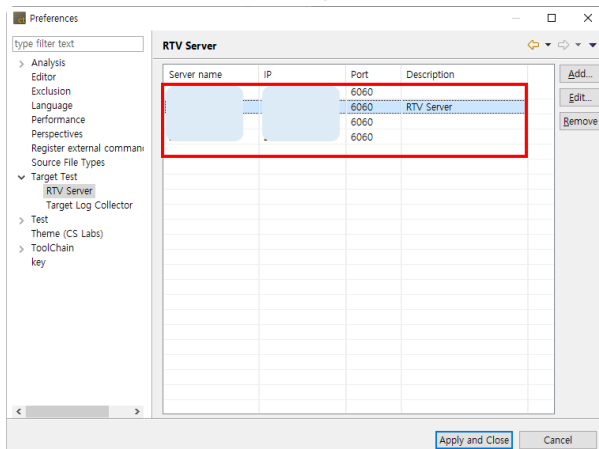
Server name:

IP:

Port:

Description:

- You can see that the target server has been added.



**Preferences**

type filter text

- Analysis
- Editor
- Exclusion
- Language
- Performance
- Perspectives
- Register external commands
- Source File Types
- Target Test
  - RTV Server**
  - Target Log Collector
- Test
- Theme (CS Labs)
- ToolChain
- key

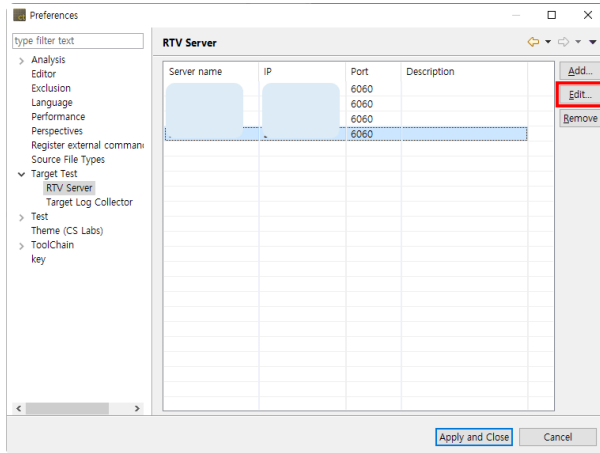
**RTV Server**

Server name	IP	Port	Description
		6060	
		6060	RTV Server
		6060	
		6060	

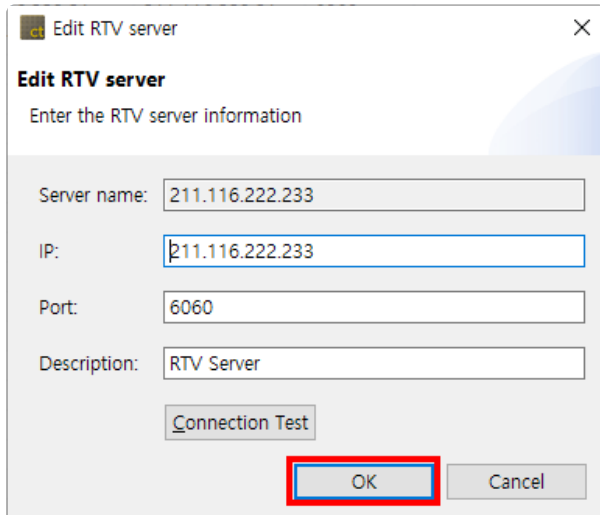


## Edit a target server

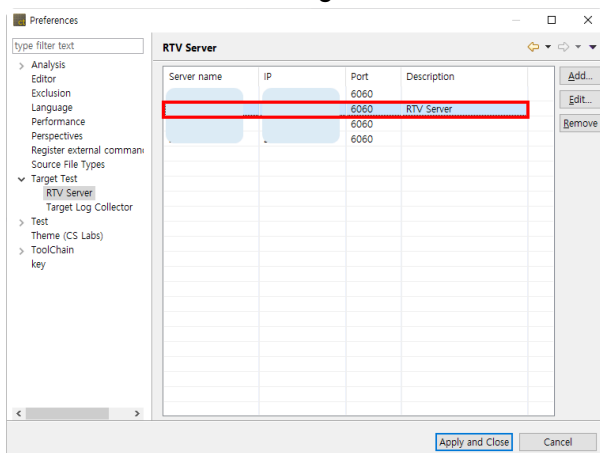
1. Double-click the server in the target server list, or select the server and click the [Edit] button.



2. On the [Edit target server] window, edit the server information and then click [OK].



3. You can see that the target server information has been edited.



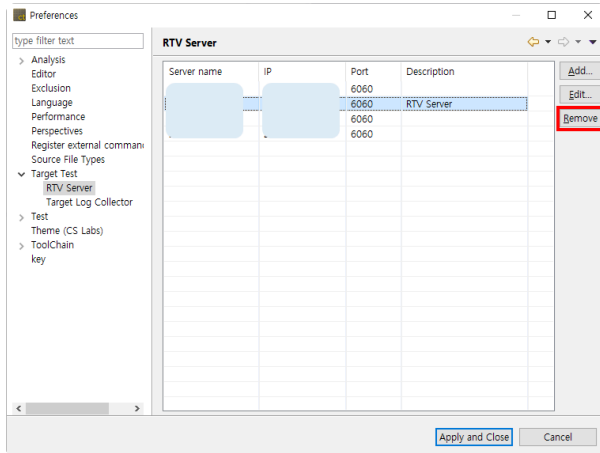


## Remove a target server

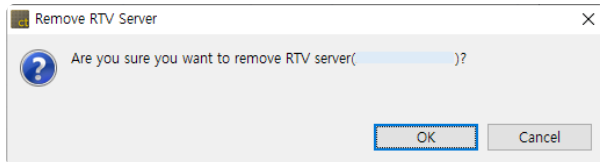
If you no longer use the target server, you can remove it.

When you remove a target server, the toolchain or project resources associated with the target server are deleted from the client environment(Can not be deleted if a target toolchain added from that target server exists).

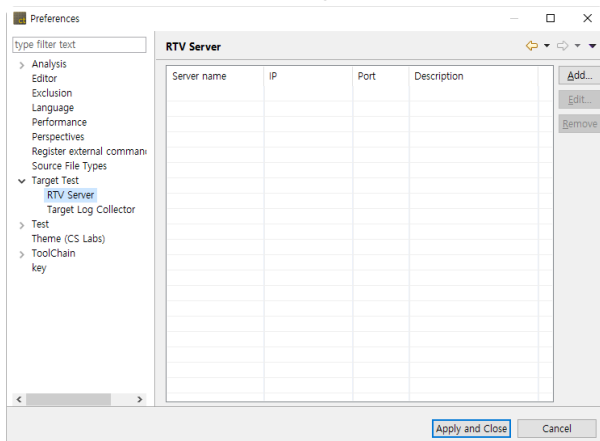
1. Select a server from the list of target servers and click the [Remove] button.



2. Click the [OK] button in the [Remove Target Server] window(Depending on the size of the target server resource, it may take a minute or two).



3. You can see that the target server has been removed.





## 24.2. RTV Toolchain Settings

### RTV toolchain

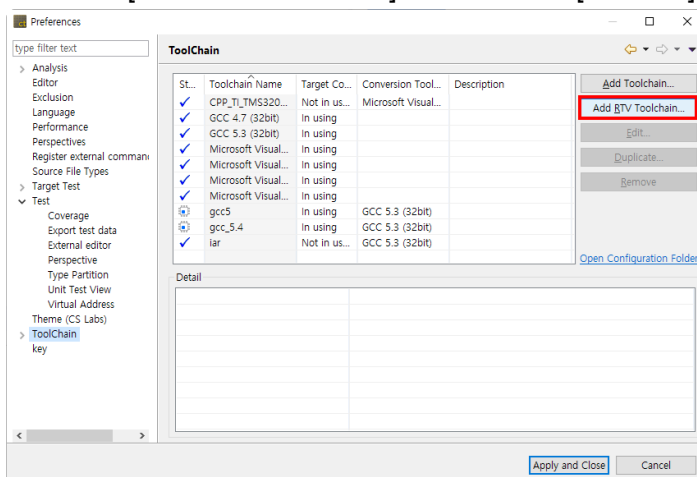
To build and run a test in a target environment with Controller Tester RTV, you need to set the toolchain (compiler) information for the source file being tested.

This chapter describes how to add, edit, and delete the target toolchain information that has previously extracted from the target-environment where the RTV Server is installed.

(For extracting toolchain information from your target environment, please contact the Technical Support Contact on [Troubleshooting](#).)

### Add an RTV toolchain

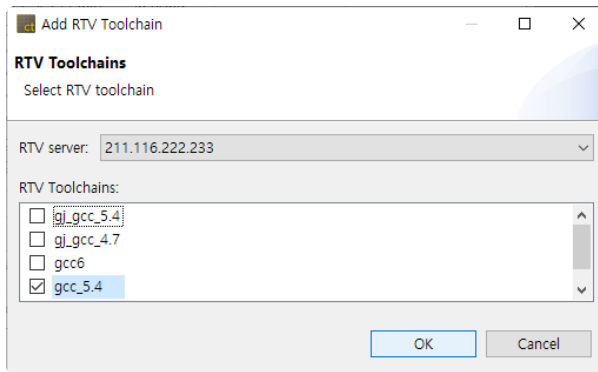
1. Click the [Add RTV Toolchain] button in the [Window] -> [Preferences] -> [Toolchain] window.



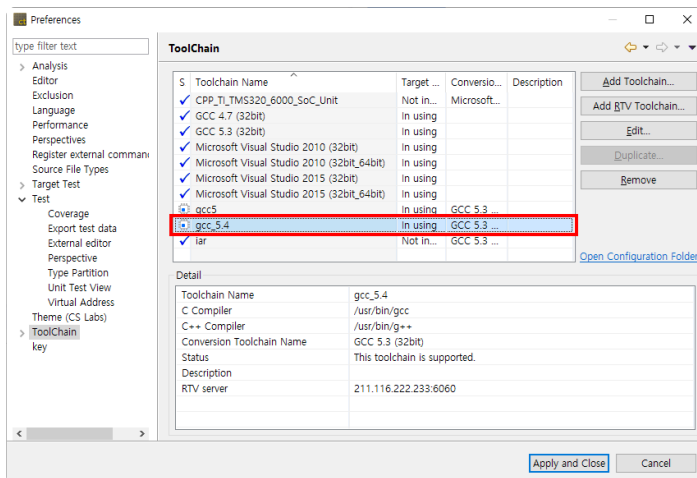
✿ If you do not have a target server, the [Add RTV Server] window will appear first.

2. If you select the target server in the [Add RTV Toolchain] window, you can see the list of target toolchain extracted from the target server. Select the toolchain you want to add and click the [OK] button.





- The toolchain information is imported from the target 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 target toolchain has been added.



- If you select the target toolchain added, you can see that the information of the target server is displayed on the [Detail].

## Edit an RTV toolchain

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



**Edit Toolchain**

**Toolchain(Method Manage)**  
Manage toolchain information.

**Toolchain Information**

Name:

Description:

Env Script:  

Configuration:

**C** **C++**

Compiler:  

Linker:  

Archiver:  

**System Header:**

**Libraries:**

Automatically extract toolchain information from entered compiler.

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 [Delete] 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 target toolchain can not be imported/exported because it can be added through the target server.



## 24.3. Create a RTV Project

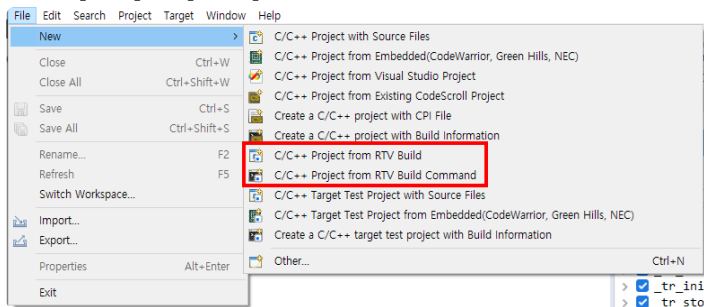
Create an RTV project to test the source files that exist in the target environment.

When building source files in the target 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 target environment, or extracting information at the same time as building the target source file.

To create an RTV project, run the [Create Project Wizard].

Click [File] -> [New] or Shortcut.

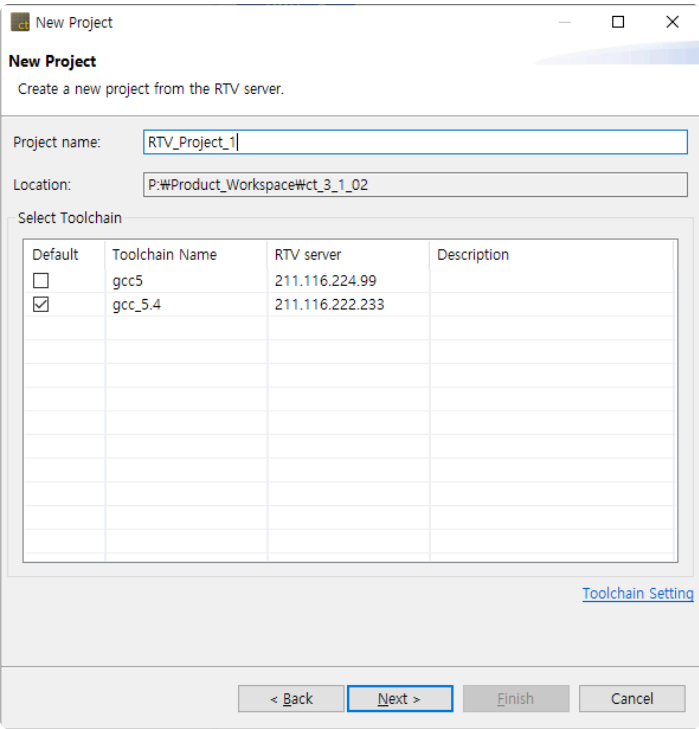


### C/C++ Project from RTV Build

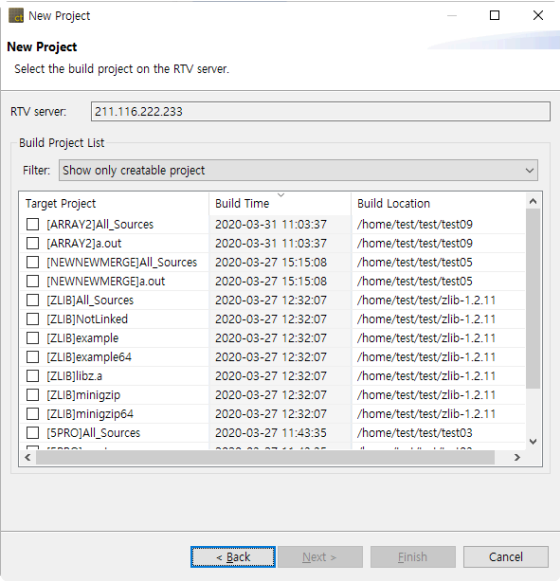
It is selected when creating an RTV project using pre-extracted source file build information (using the Controller Tester RTV Server Utility) in the target environment.

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





- 2. A list of target 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 target server.

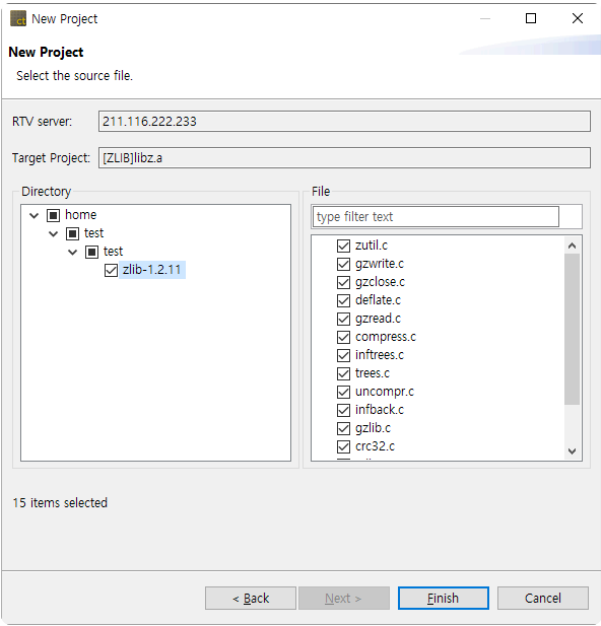


\* Among the generated projects, 'All\_Sources\_Project Name' is a project that combines all the source files of the modules captured by csbuild. Since this project is a combined project without considering whether it is built or not, it may not be able to perform the test normally.

- 4. In the [Build Project List], select the item for which you want to create the RTV project and click



the [Next] button.

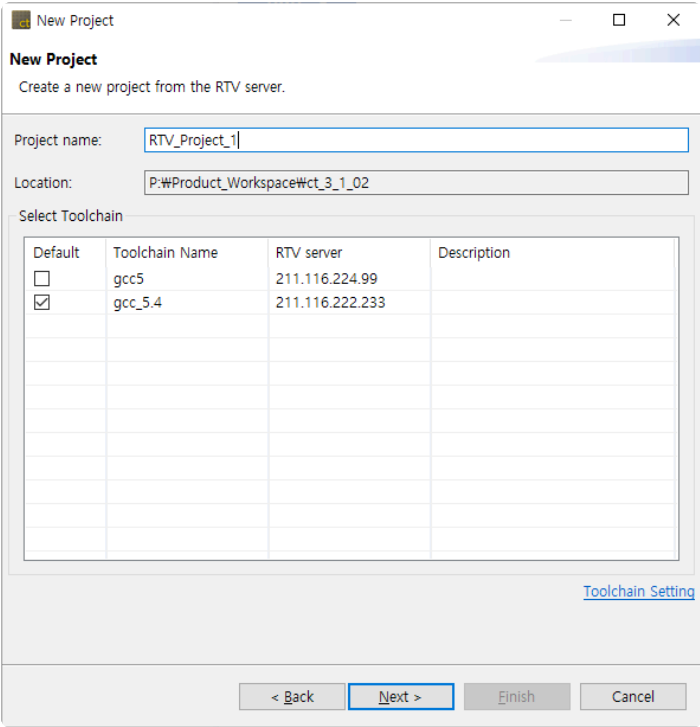


5. After selecting source files to be tested, click the [Finish] button to create the project.

## C/C++ Project from Target 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 target server using the build command and then creates the RTV project.

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





2. A list of target 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 target server from which the selected toolchain was imported.

4. Enter the path to build from the target 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 target 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.

Target Project	Build Time	Build Location
<input type="checkbox"/> [ARRAY2]All_Sources	2020-03-31 11:03:37	/home/test/test/test09
<input type="checkbox"/> [ARRAY2]a.out	2020-03-31 11:03:37	/home/test/test/test09
<input type="checkbox"/> [NEWNEWMERGE]All_Sources	2020-03-27 15:15:08	/home/test/test/test05
<input type="checkbox"/> [NEWNEWMERGE]a.out	2020-03-27 15:15:08	/home/test/test/test05
<input type="checkbox"/> [ZLIB]All_Sources	2020-03-27 12:32:07	/home/test/test/zlib-1.2.11
<input type="checkbox"/> [ZLIB]NotLinked	2020-03-27 12:32:07	/home/test/test/zlib-1.2.11
<input type="checkbox"/> [ZLIB]example	2020-03-27 12:32:07	/home/test/test/zlib-1.2.11
<input type="checkbox"/> [ZLIB]example64	2020-03-27 12:32:07	/home/test/test/zlib-1.2.11
<input type="checkbox"/> [ZLIB]libz.a	2020-03-27 12:32:07	/home/test/test/zlib-1.2.11
<input type="checkbox"/> [ZLIB]minigzip	2020-03-27 12:32:07	/home/test/test/zlib-1.2.11
<input type="checkbox"/> [ZLIB]minigzip64	2020-03-27 12:32:07	/home/test/test/zlib-1.2.11
<input type="checkbox"/> [SPRO]All_Sources	2020-03-27 11:43:35	/home/test/test/test03

At the bottom are buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

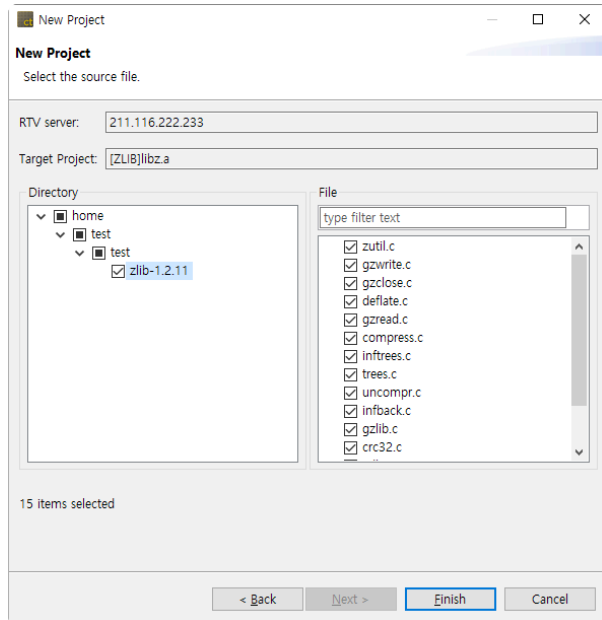


Among the generated projects, 'All\_Sources\_Project Name' is a project that combines all



the source files of the modules captured by csbuild. 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.

## Target build command

To create a project from build information, you must use the cs command on the build command.

### cs command

The cs is a utility that automatically extracts the project configuration information of the source files that are built so that the Controller Tester RTV project can be created.

### How to use cs

Prefix the existing build command with cs.

- Ex) make → csbuild capture make

### h3.cs 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) `csbuild new --toolchain=gcc5.4 --project=PRJ`

**!** The `-toolchain` option is required.

**capture:** Creates an RTV project by performing build command extraction.

- Ex) `csbuild capture make`

**convert:** Convert a STATIC project to an RTV project.

- Ex) `csbuild convert`



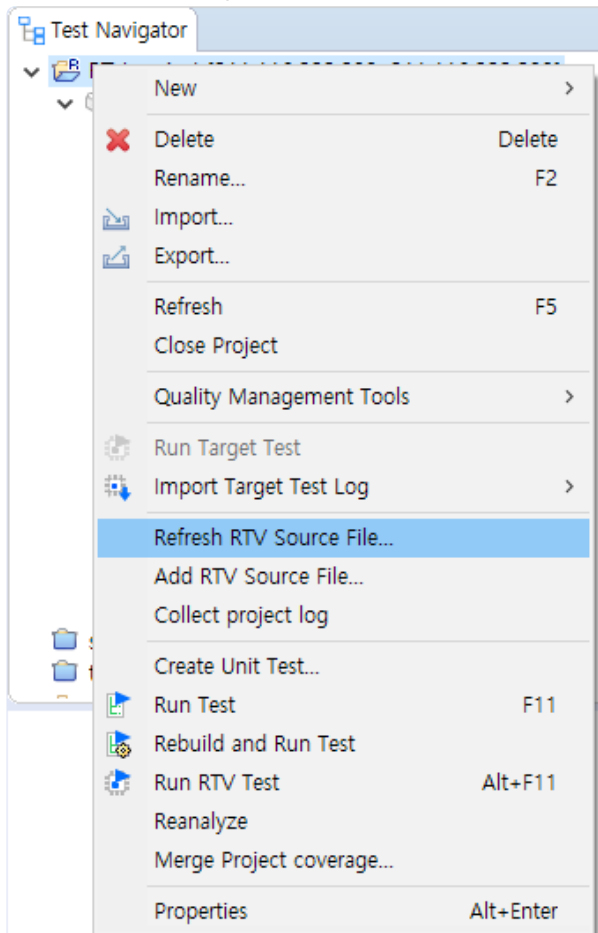
## 24.4. Refresh/Add RTV Source Files

### RTV Refresh target source file

If the source file is modified and rebuilt on the target server, the [Refresh RTV Source File] allows the revised content to be reflected in the project.

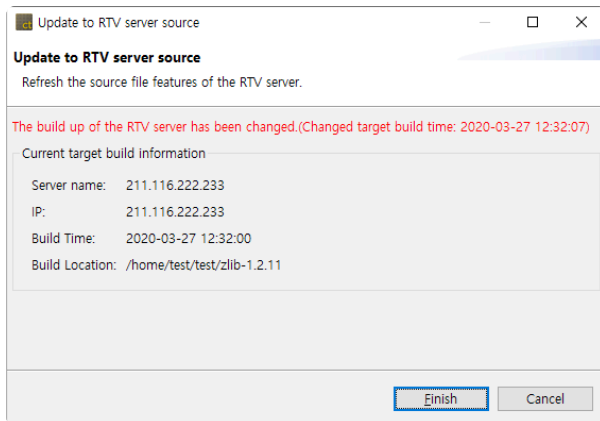
! If the source file is modified but not rebuilt, the changed contents will not be reflected. If you modify the source files on the target server and then rebuild, you must perform a full build.

1. In the [Test Navigator] view, click the [Refresh RTV Source File] context menu.

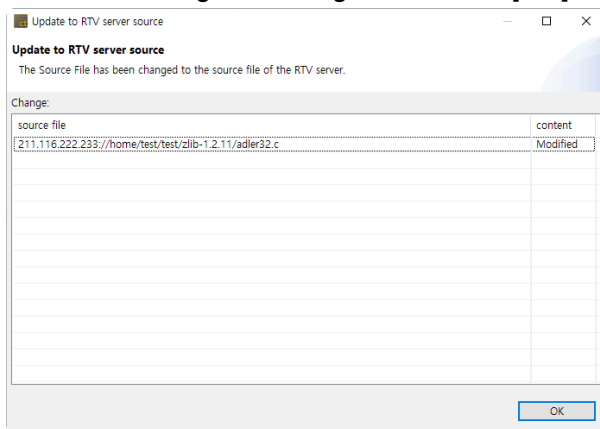


2. When you create an RTV project, you can see the latest build information for the source files you selected (display a change notification if the build has changed). Click the [Finish] button to confirm your changes.





3. After confirming the changes, click the [OK] button.



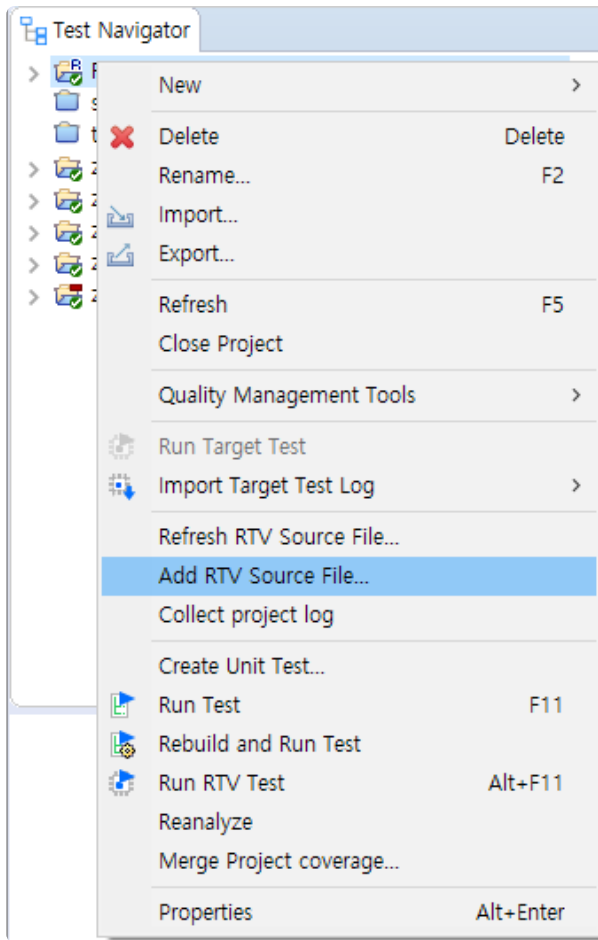
## Add target source file

You can add source files to your RTV projects via the [Add Target Source File] if new source files are added and built on the target server, or if only some source files are selected when you create the RTV project.

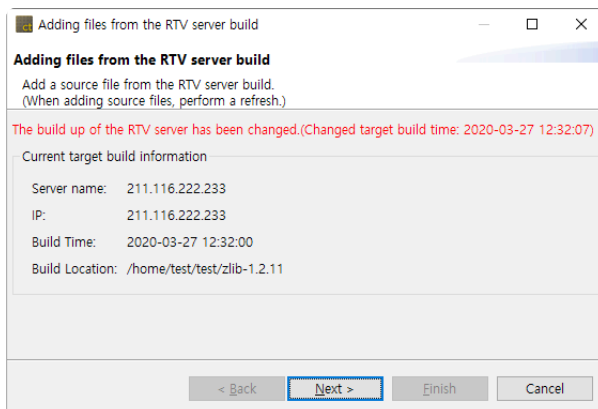
- ! If the source file is added but not rebuilt, the changed contents will not be reflected. If you build the target server after adding the source file, you must perform a full build.

1. In the [Test Navigator] view, click the [Add Target Source File] context menu.



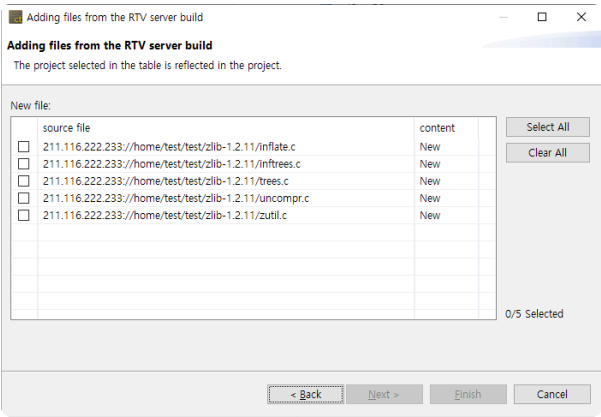


2. When you create the RTV project, you can see the latest build information for the selected source file (show change notification if the build has changed). Click the [Next] button to confirm the new file list.

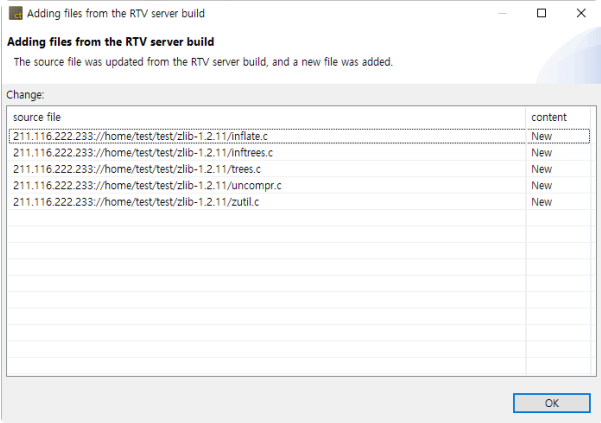


3. In the [New file], select the file you want to add and click the [Finish] button.






4. After confirming the changes, click the [OK] button.

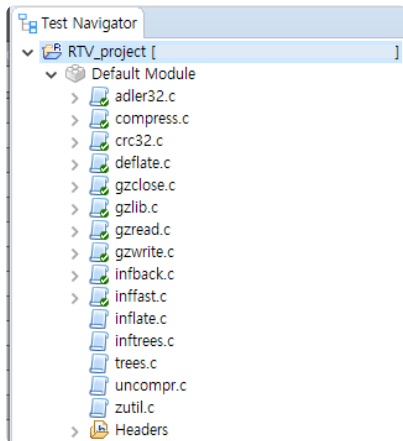




## 24.5. RTV Project Information

### Target Project Information

1. The icon for the RTV project is displayed as .
2. The target server's information ([server name: IP]) is displayed after the name of the RTV project.

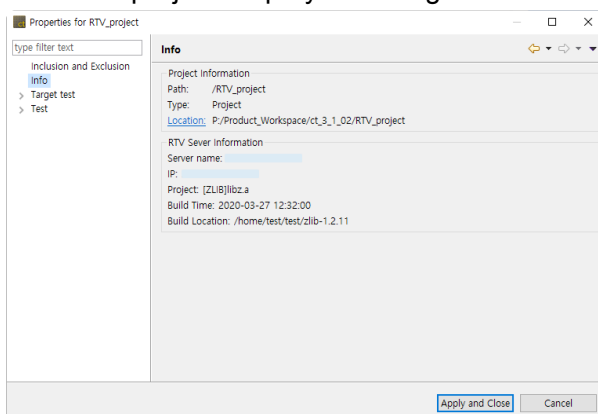


3. Source files in the RTV project can not be modified(the source file editor of the RTV project is read-only).

### Project property

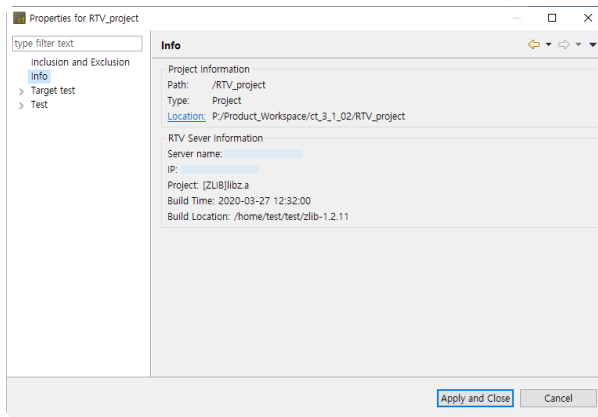
#### Info

1. The RTV project displays the target server and project information.



2. If the general project has set the toolchain as the RTV toolchain, information about the target 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 target 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).



## 24.6. Run RTV Test

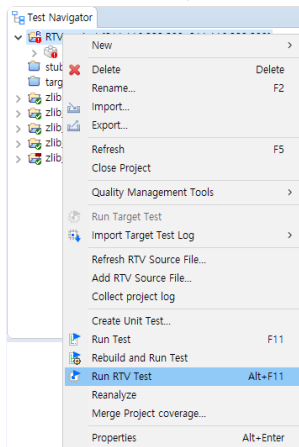
The RTV project allows you to run RTV tests without any additional setup. To test a general project in the target environment, you must set the toolchain of the project to the RTV toolchain created from the target server to be tested.

### Run all RTV tests

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

Target tests can be run in the following way.

1. Select the project to be tested, right-click it and click [Run Target Test].

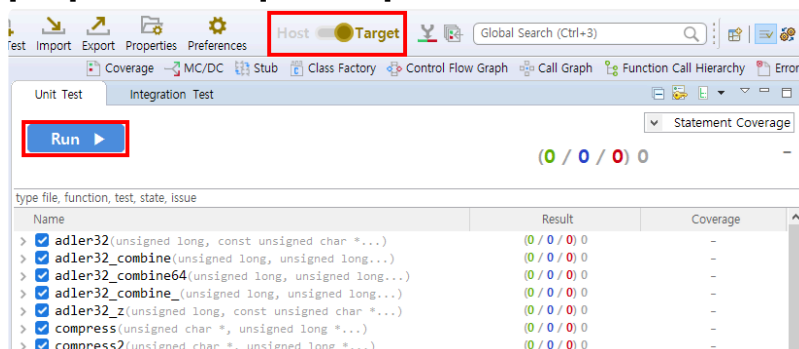


### Run RTV unit test

Runs the selected test in the [Unit Test] view.

### Run RTV test

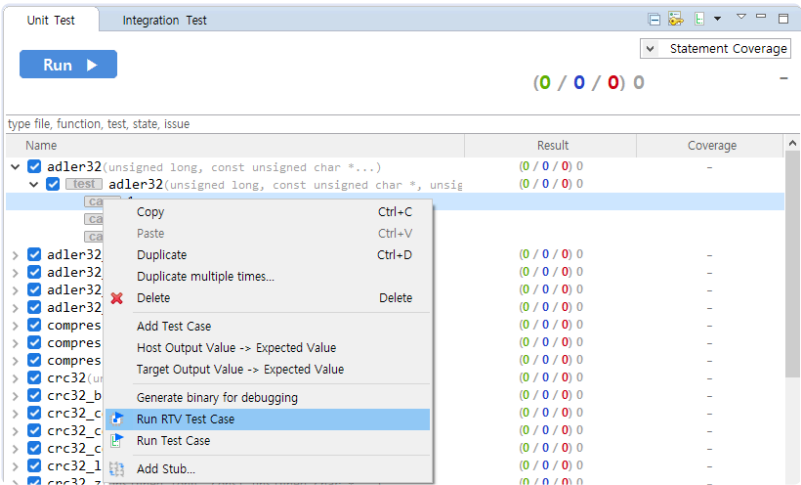
[Run] button In the [Unit Test] view.





## Run RTV test case

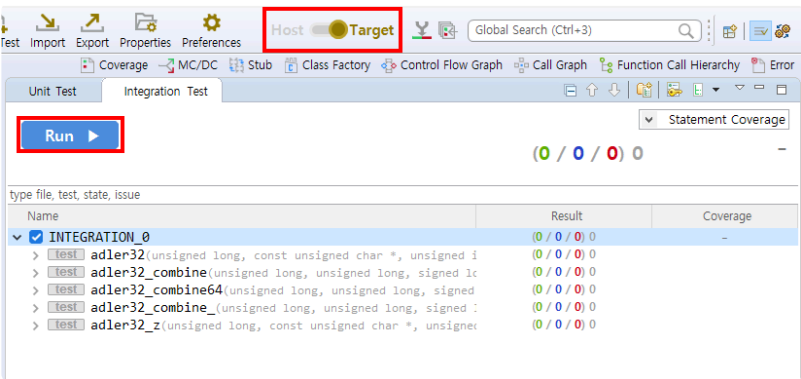
[Run Target Test Case] context menu (multi-selectable) in the [Unit Test] view.



## Run RTV integration test

Runs the selected test in the [Integration Test] view.

[Run] button In the [Integration Test] view.





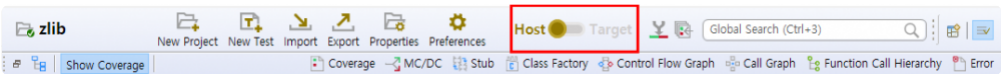
# 24.7. RTV Test Results

## 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.

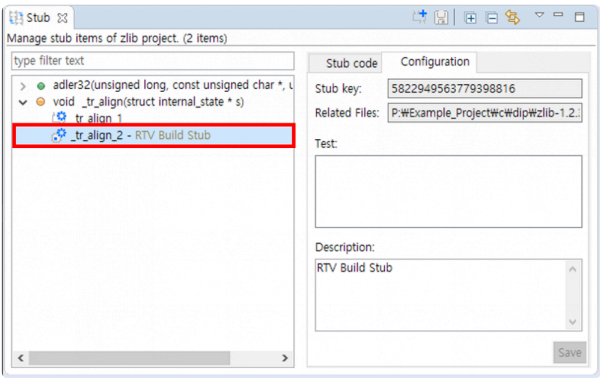
## Test results


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 target stub



# 25. 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.