

# FORTRA

beSOURCE  
5.2

**Enterprise Developer  
Guide**

## **Copyright Terms and Conditions**

---

Copyright © Fortra, LLC and its group of companies. All trademarks and registered trademarks are the property of their respective owners.

The content in this document is protected by the Copyright Laws of the United States of America and other countries worldwide. The unauthorized use and/or duplication of this material without express and written permission from Fortra is strictly prohibited. Excerpts and links may be used, provided that full and clear credit is given to Fortra with appropriate and specific direction to the original content.

202307181032

# Table of Contents

<b>Overview</b>	<b>1</b>
What is beSOURCE?	1
Supported technologies	2
Executing beSOURCE Developer	5
<b>Menus and Start Page</b>	<b>10</b>
Menus	10
Toolbar	19
Start Page	20
<b>Creating and Analyzing a Project</b>	<b>22</b>
Creating a project	22
Analyzing a project	30
Analyzing a portion of the source files	31
Setting the referencing header for C/C++	32
Analysis engine options (CodeMRI Options)	34
<b>User Interface</b>	<b>36</b>
Project window	36
Category View window	37
List of Rule Violations window	38
Summary of Violations	39
Source Viewer	40
Rule Description window	41

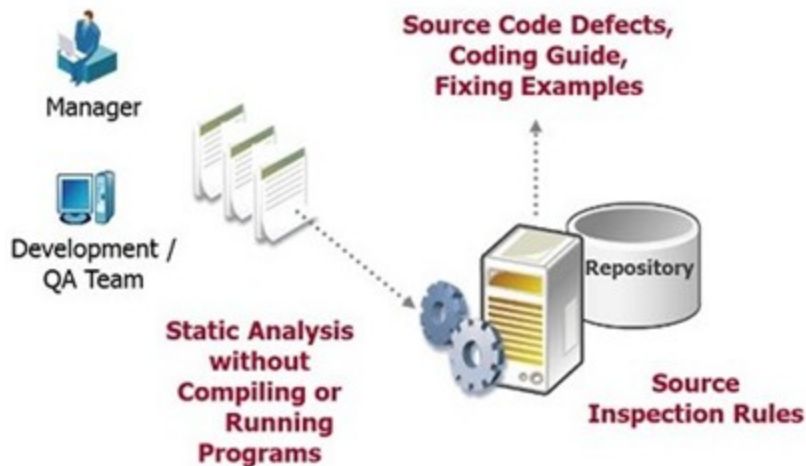
Flow Trace window .....	41
Violation Status .....	42
Violation History .....	43
Comparing History .....	43
Advanced search for Rule Violations .....	44
<b>Reports .....</b>	<b>46</b>
Creating reports for inspection results .....	46
Report types .....	47
<b>Quick Start: Analyzing C/C++ Source Files .....</b>	<b>52</b>
Overview .....	52
What you need to know .....	52
How to analyze C source files .....	52
How to analyze C++ source files .....	56
<b>Quick Start: Analyzing Java Source Files .....</b>	<b>57</b>
Overview .....	57
Preparation .....	57
How to analyze Java source files .....	57
<b>Command-line Interface .....</b>	<b>62</b>
Execution environment .....	62
Command format .....	62
Job options .....	63
Creating a project file .....	64

# Overview

## What is beSOURCE?

beSOURCE is a source code inspection tool. Through its static analysis technology, beSOURCE automatically detects software security defects in a program but without executing it. The static analysis tool takes program source files and libraries as input and then uses pattern matching and path flow analysis technologies to identify defects or vulnerabilities in source code.

Defects are defined by rules in the tool; a rule violation is reported as a defect (or vulnerability). The violation report includes code line numbers, descriptions of the rules, and detailed examples that show how to fix the violations.



Through use of the static analysis tool, maintenance efficiency is increased, costs are reduced, and business disruptions caused by source code defects are prevented.

beSOURCE provides the following benefits:

- **Security improvement** - By detecting potential security defects in the early stages of development, you can prevent system breaches through improved application security.
- **Productivity** - Maximize development productivity through automatic analysis instead of manual code review.
- **Cost reduction** - Reduce the cost of error-fixing and additional maintenance through the detection of security defects early in development.
- **Compliance** - Comply with code regulations and standards for secure code.

# Supported technologies

beSOURCE provides predefined rules that are based on industry standards. The tool detects security defects based on these predefined rules. The supported programming languages are:

- Java 1.8
- JavaWeb 1.8 (scan all Java, JS, XML and HTML files)
- PHP 7
- Perl 5
- Python 2.7
- Python 3
  - C99
- C++
  - C# 6
- VB.NET 10
- ASP.NET: aspx and C# files
- ASP.NET: aspx and VB files
- Objective C 6.3
- Visual Basic 6
- Swift 5

**NOTE:** Most previous versions of some languages are supported by default.

beSOURCE supports the following secure coding standards (rule sets):

## For ASP.NET

Ruleset
ASP.NET(C#) CWE Top 25 2019
ASP.NET(C#) CWE Top 25 2020
ASP.NET(C#) OWASP API Security Top 10 2019
ASP.NET(C#) OWASP Top 10 2017
ASP.NET(C#) Quality Ruleset
ASP.NET(C#) Ruleset
ASP.NET(C#) Security Ruleset
ASP.NET(VB) CWE Top 25 2019
ASP.NET(VB) CWE Top 25 2020
ASP.NET(VB) OWASP API Security Top 10 2019
ASP.NET(VB) OWASP Top 10 2017
ASP.NET(VB) Quality Ruleset
ASP.NET(VB) Ruleset
ASP.NET(VB) Security Ruleset

## For C#

Ruleset
C# CWE Top 25 2019
C# CWE Top 25 2020
C# OWASP API Security Top 10 2019
C# OWASP Top 10 2017
C# Quality Ruleset
C# Ruleset
C# Security Ruleset

## For C++

Ruleset
BSSC-C++
C++ CWE Top 25 2019
C++ CWE Top 25 2020
C++ OWASP API Security Top 10 2019
C++ OWASP Top 10 2017
C++ Ruleset
C++ Security Ruleset
CERT C++
HICPP
MISRA-C++-2008

## For C99

Ruleset
BSSC-C
C CWE Top 25 2011 (CWE/SANS)
C CWE Top 25 2019
C CWE Top 25 2020
C OWASP API Security Top 10 2019
C OWASP Top 10 2010
C OWASP Top 10 2017
C Ruleset
C Security Ruleset
CERT C
HIS
JPL
MISRA-C-2004

## For Java

Ruleset
CERT Java
Java CWE Top 25 2011 (CWE/SANS)
Java OWASP API Security Top 10 2019
Java OWASP Top 10 2013
Java OWASP Top 10 2017
Java Quality Ruleset
Java Ruleset
Java Ruleset_(Copy)
Java Security Ruleset

## For Javaweb

Ruleset
Android Security Ruleset
CERT Java
Java Web CWE Top 25 2019
Java Web CWE Top 25 2020
Java Web OWASP API Security Top 10 2019
Java Web OWASP Top 10 2013
Java Web OWASP Top 10 2017
Java Web Quality Ruleset
Java Web Ruleset
Java Web Security Ruleset
Java Web Security Ruleset_(Copy)

## For Objective C

Ruleset
Objective C CWE Top 25 2019
Objective C CWE Top 25 2020
Objective-C OWASP API Security Top 10 2019
Objective-C OWASP Top 10 2017
Objective-C Quality Ruleset
Objective-C Ruleset
Objective-C Security Ruleset
Objective-C iOS Security Ruleset

beSOURCE contains rules that reflect the available coding guidance of the above standards. For its static analysis technology, the tool analyzes source code based on the rules.

**NOTE:** There are two categories of software defects: quality and security. Security defects are analyzed according to the secure coding standards, detecting application's vulnerabilities. The quality-related rules are based on well-known industry coding standards and international standards. Your purchased license determines the supported languages and predefined rules. In beSOURCE, quality-related rules are represented as Quality Rule, and security rules are represented as Security Rule.

Descriptions of standards are shown below:

### Standard description

#### CERT C secure coding guide

CERT C Programming Language Secure Coding Standard (10 09 200 7). Carnegie Mellon University Software Engineering Institute's Computer Emergency Response Team defined the security related development guidelines.

#### CWE C secure coding guide

CWE means Common Weakness Enumeration. You can see vulnerabilities for several languages, see <https://cwe.mitre.org>

#### CERT C++ secure coding guide



CERT C++ Secure Coding Standard (28 01 2010).

CERT Java secure coding guide

CERT Java Secure Coding Standard (08 09 2011).

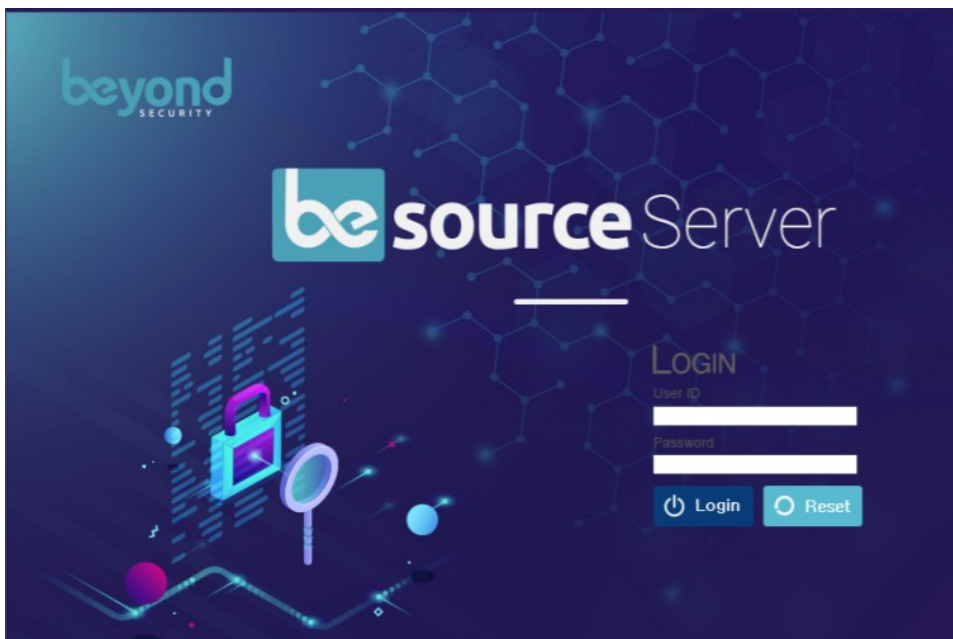
OWASP Top Ten

The OWASP Top Ten is a list of the 10 most dangerous current Web application security flaws, along with effective methods of dealing with those flaws. OWASP (Open Web Application Security Project) is an organization that provides unbiased and practical, cost-effective information about computer and Internet applications.

## Executing beSOURCE Developer

You can run the beSOURCE Developer from a web browser (Internet Explorer, Firefox, Safari etc.) The procedures are as follows:

1. Verify that the Analysis Server and View Server are running.
2. Open your web browser with system administrator privileges and go to the following URL:
  - a. **http://View\_Server\_IP:Admin\_Server\_Port** [default port is 50102] (for example, **http://localhost:50102**).
    - i. **View\_Server\_IP** is the IP address of Server Computer where the View Server is installed.
    - ii. **View\_Server\_Port** is the port number assigned at the time of installation.
3. Enter your **User ID** and **Password**, and then select **Login**.

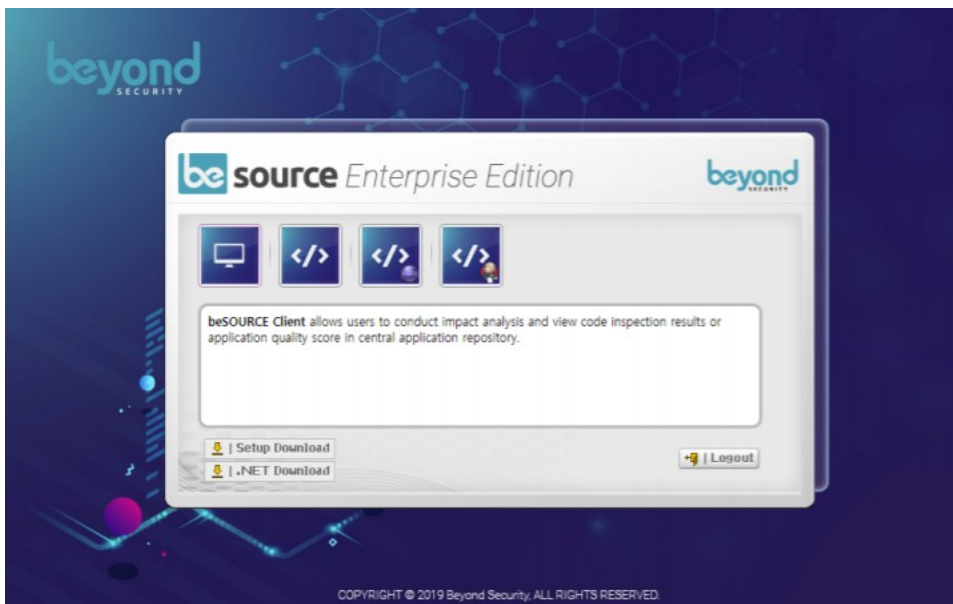


**IMPORTANT:** Only users with sufficient rights provided by the administrator can access the beSOURCE Developer. After five incorrect password attempts, beSOURCE Developer will block and disable the user account.

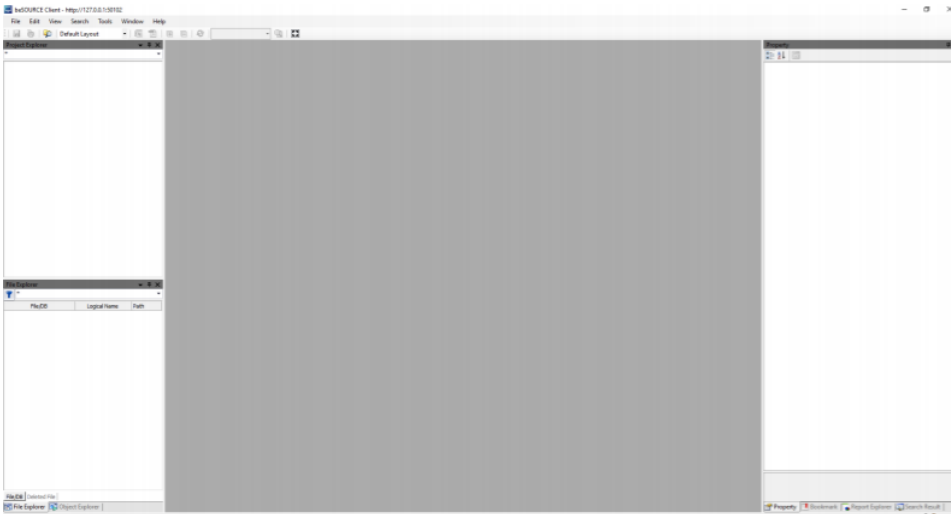
4. After logging in, you will see the icons (hover your pointer each one to display a brief description).



**NOTE:** The Admin Console icon only appears for users with administrator privileges.



5. Select the **beSOURCE Developer** icon. It will run automatically.

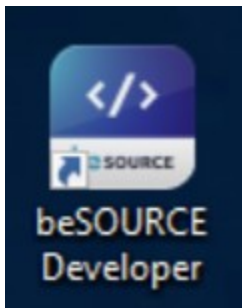


**NOTE:** If a user tries to log in from another computer after successfully logging in to the server, the first connection will automatically expire to prevent duplicate connections of the same user.

## Running beSOURCE Developer

After you download beSOURCE Developer with a web browser, you can launch it with a shortcut icon on your desktop. To launch beSOURCE Developer with a shortcut icon in your desktop, follow the below steps.

1. To open beSOURCE Developer, double-click the **beSOURCE Developer** shortcut icon on your desktop.

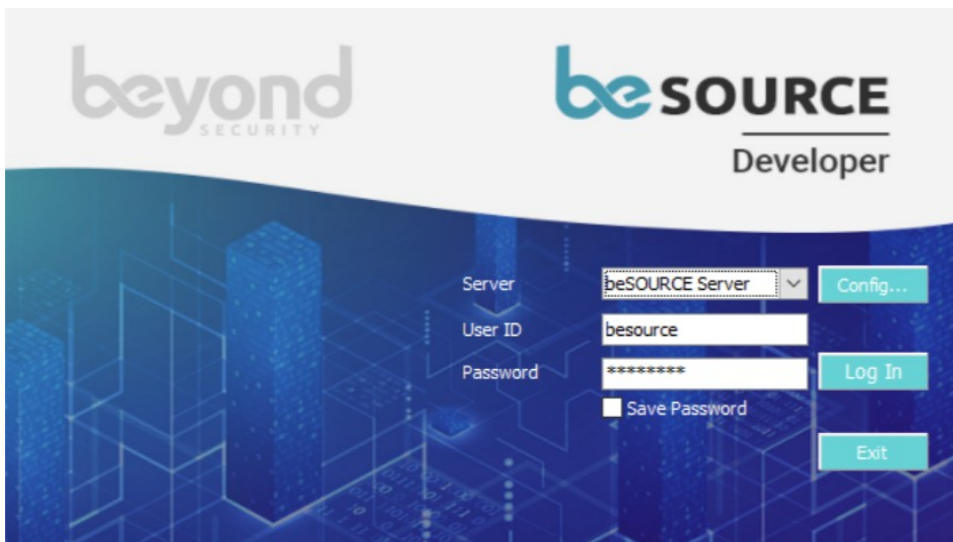


2. In the **Server** box, select your server. If you need to add a server select **Config** and do the following:
  - a. Select **Add**.
  - b. In the **Server**, **IP Address**, and **Port** boxes, enter the beSOURCE View Server's information.
  - c. Select **Save** or **Save & Close**.

A screenshot of the "Connect Config" dialog box. It contains a table with server information and buttons for "Add", "Delete", "Save", "Save & Close", and "Close".

Server	IP Address	Port	Encoding	User ID	Password	SSL
<input type="checkbox"/> beSOURCE Ent...	13.125.16.149	50102	ISO(iso-8859...			<input type="checkbox"/>
<input type="checkbox"/> Local Host	127.0.0.1	50102	ISO(iso-8859...			<input type="checkbox"/>

3. Enter your **User ID** and **Password**.
4. Select **Log In**.



# Menus and Start Page

This section describes the menus and UI layout of the beSOURCE Developer in the Enterprise edition. When you open beSOURCE Developer, it displays the main menus, toolbar, and start page.

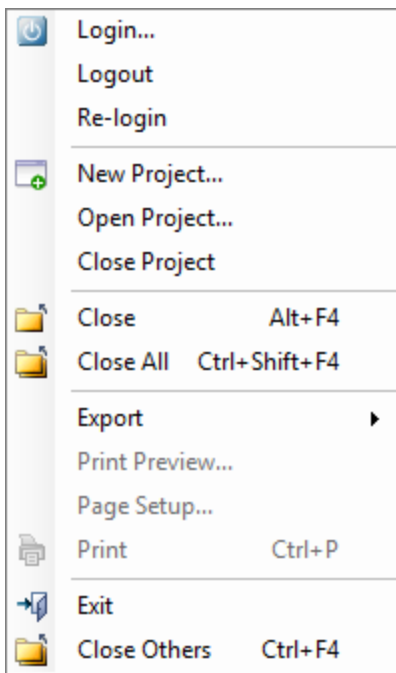
## Menus

beSOURCE Developer provides the following main menus.

- **File** - Provides basic sub-menus to create or close a project and open or close windows.
- **View** - Provides gives access to sub-menus for accessing additional windows and functions in beSOURCE Developer.
- **Analysis** - Allows you to start a full or partial code analysis job or stop an analysis.
- **Report** - Provides a sub-menus for creating reports.
- **Tools** - Provides sub-menus for editing rule sets and configuring the tool's environment.
- **Window** - Provides sub-menus to returning to the start page or selecting the next window or preceding window.
- **Help** - Provides access to on-line help, the current software version, and trademarks.

## File menu

The File menu provides the following sub-menus:

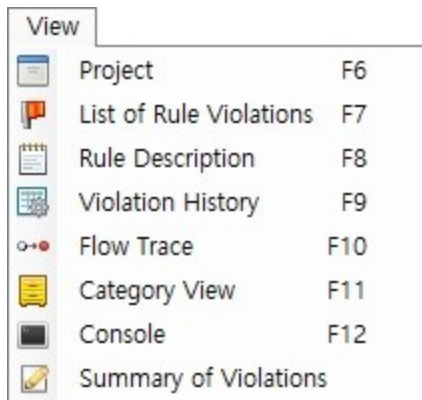


- **Login** - Opens a beSOURCE Server login window.
- **Logout** - Terminates the current user session.
- **Re-login** - Renews the current user session.
- **New Project** - Opens a wizard to create a new project.
- **Open Project** - Opens an existing project.
- **Close Project** - Closes the current, active project.
- **Close** - Closes the current window.
- **Close All** - Closes all active windows.
- **Export** - Provides sub-menus to export the selected window's data to a file. Notice that the file exporting might not be supported for the selected window type.
- **Print Preview** - Opens a preview of the print output.
- **Page Setup** - Opens the printer setup page you to specify print parameters.
- **Print** - Initiates the print operation.
- **Exit** - Terminates beSOURCE Developer.

**NOTE:** The beSOURCE Enterprise edition also supports a server environment. The administrator centrally manages base configurations on the beSOURCE server (rule sets, user accounts, privileges, and so on) and analyzes source files with the server instance. A developer can download the required data from the server and analyze source files on a local PC.

## View menu

The View menu provides the following sub-menus:

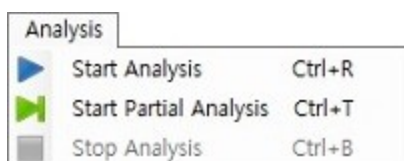


- **Project** - Opens a panel with a tree of all source files in the project.
- **List of Rule Violations** - Opens a window for listing the rule violations in the current project; it provides multiple information options.
- **Rule Description** - Opens a window that shows a description of the selected rule, examples of good and bad code related to the rule, and applicable standards.
- **Violation History** - Opens a window for showing a history of rule violations.
- **Flow Trace** - Opens a window for showing control and data flow related to detected source code lines.
- **Category View** - Opens a window with rule violations for a selected category.
- **Console** - Opens a monitoring window for the source code analysis process.
- **Summary of Violations** - Opens a window displaying the source code that violated specific rules.

**NOTE:** beSOURCE Developer supports floating window user interface. You can personalize the user interface layout. The windows from View menu cannot be placed in the central document area but can be placed around the central area.

## Analysis menu

The Analysis menu provides the following sub-menus:

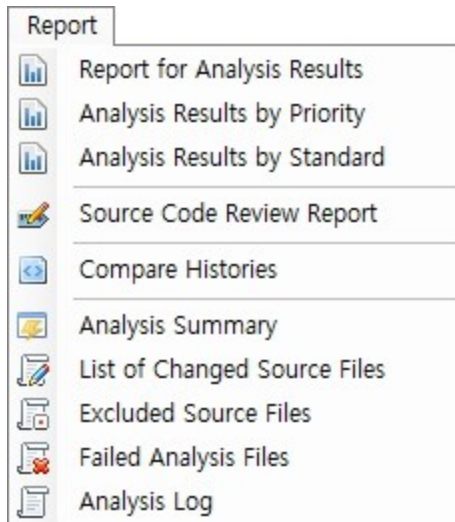




- **Start Analysis** - Initiates analysis of all source files in the current project.
- **Start Partial Analysis** - Initiates analysis of selected source files or the folder for the current project.
- **Stop Analysis** - Stops the current analysis.

## Report menu

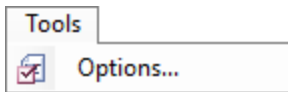
The Report menu provides the following sub-menus:



- **Report for Analysis Results** - Generates a total inspection report for the current active project.
- **Analysis Results by Priority** - Generates an analysis report for only the selected priorities.
- **Analysis Results by Standard** - Generates an inspection report based on the standard view you selected.
- **Source Code Review Report** - Generates an Excel file that includes rule violations and review form.
- **Compare History** - Shows the differences in the rule violation history of the current and preceding analysis.
- **Analysis Summary** - Displays a summary of the source file inspection.
- **List of Changed Source Files** - Shows a list of changed files in comparison to files in the preceding analysis.
- **Excluded Source Files** - Displays a list of files that were excluded from analysis.
- **Failed Analysis** - Shows the analysis' failed files with error messages.
- **Analysis Log** - Shows the log for the analysis job itself.

## Tools menu

The Tools menu provides the following sub-menus:

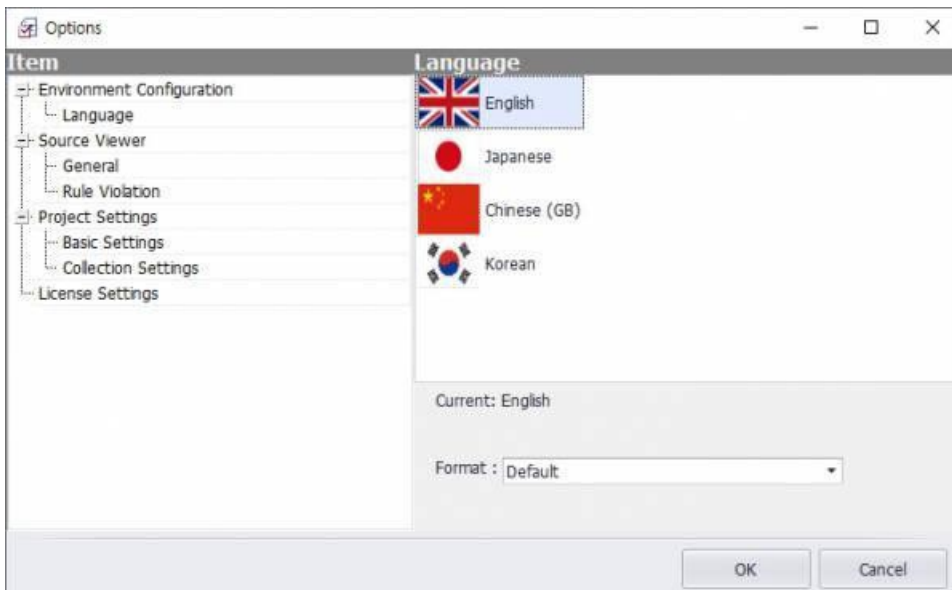


- **Options** - Opens the Options window, providing access to the environment's settings.

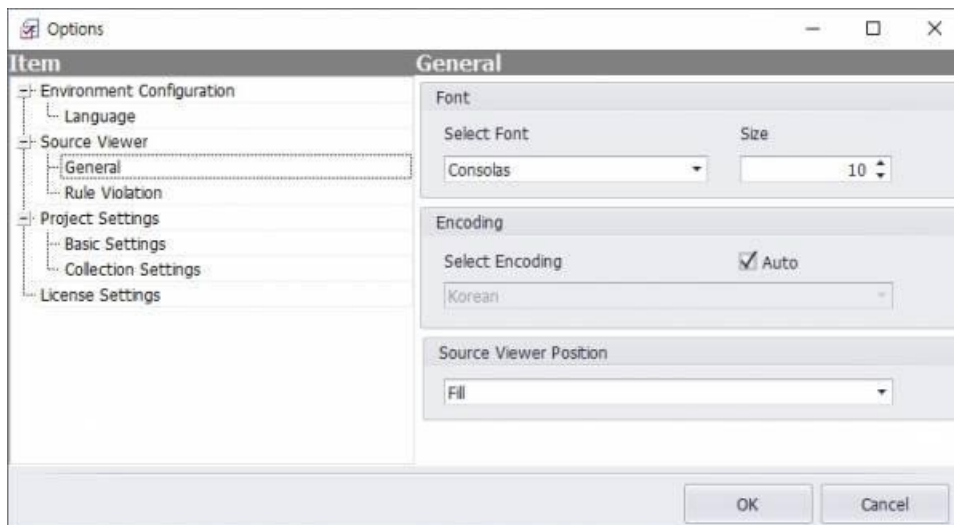
## Options settings

In the Options window, categories are displayed on the left and specific options on the right.

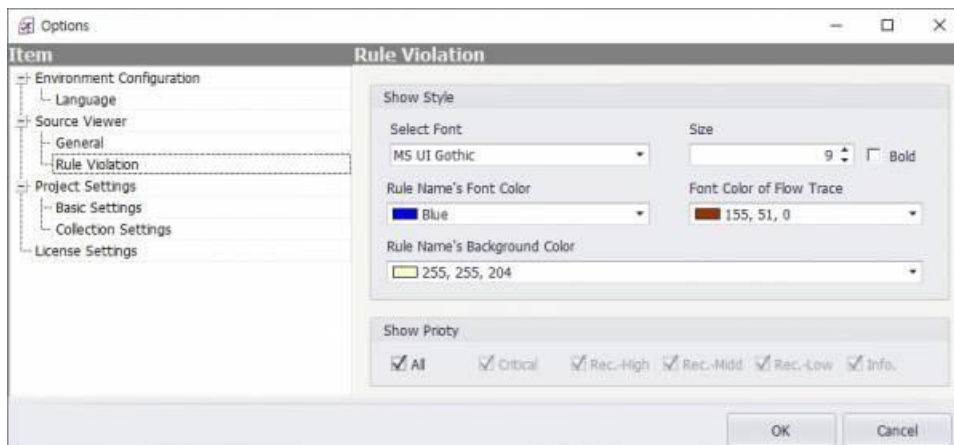
- **Environment Configuration > Language** - Specifies the default language for the user interface. You can select **English**, **Japanese**, **Chinese**, or **Korean**.



- **Source Viewer > General** - Specifies the font type and size, a default encoding (or automatic), and the position of the source viewer in the window.

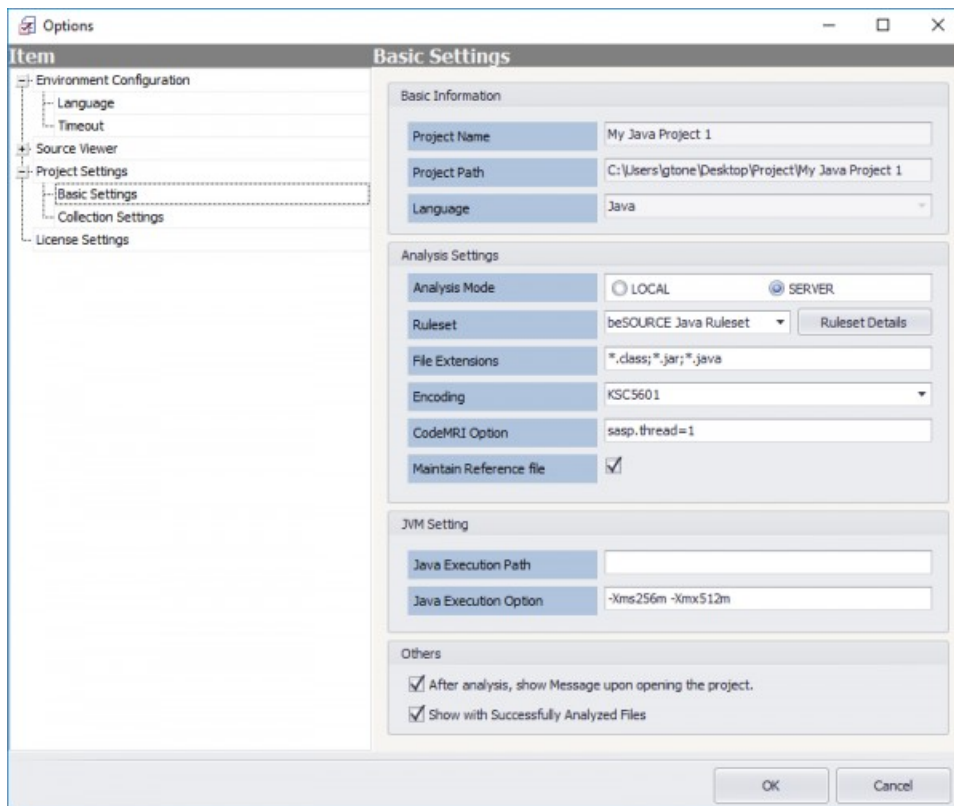


- **Font** - Specifies the font and character size.
- **Encoding** - Specifies the default encoding for the source viewer window. Select **Auto** to allow the source viewer to automatically select the encoding.
- **Source Viewer Position** - Specifies a position for the source viewer in the screen.
- **Source Viewer > Rule Violation** - Specifies the style and priority of the display of rule violations.



- **Show Style** - Specifies a font, font size, font color, and (optional) bold face of rule names and trace information. This area also lets you select the background color for rule names.
- **Show Priority** - Specifies the priorities of the rules to show in the source viewer.
- **Project Setting > Basic Setting** - Specifies detailed options for source code analysis.

**NOTE:** The project must exist and be opened before you can specify these basic settings.



- **Project Name** - Describes the name of the project.
- **Project Path**- The location where the project is saved.
- **Language** - The project's programming language type.
- **Analysis Mode** - LOCAL is for stand-alone analysis on a developer PC. SERVER is for analysis through server access.
- **Ruleset** - Allows you to select the rule set for the current project. You can only use one rule set at a time to analyze a project; this box allows you to select another rule set for analyzing the project.
- **Ruleset Details** - Opens a dialog that displays details about the rules. For the rule you select in the dialog's left pane, the right pane shows: Rule name, its priority, a description, examples of good and bad code, and information about related standards. If the rule has any options, the Option tab displays (if present): regular expressions, an include or exclude table, and a specific target type. The popup defaults to the entire set of rules (all standards) for the current project but also lets you filter by one of the project's standards (as determined by your programming language and license). The number of rules in the project appears in the upper-right corner and varies with filtering (for all standards/all rules or just for one standard).

- **File Extension** - Identifies the file extensions in the project (which is the actual target of analysis). To add multiple file extensions, use a semicolon (;) as the delimiter.
- **Encoding** - Specifies the encoding to use in analyzing a target's source files.
- **CodeMRI Option** - An internal option for the analysis engine. Refer to Analysis Engine Options (CodeMRI Options).
- **Maintain Reference File** - Specifies whether to reuse reference information generated by the analysis engine or generate the reference information whenever an analysis job runs.
- **Java Execution Path** - Specifies the JDK path used by the internal analysis engine. You can specify it as an absolute path or a relative path. For the relative path, you need to specify it in relation to the beSOURCE installation directory.
- **Java Execution Option** - Specifies JDK execution options that the internal analysis engine uses.
- **Show Message When Opening the Project after Finishing Analysis** - Selecting this option prompts a notification dialog when the project reopens after an analysis finishes:

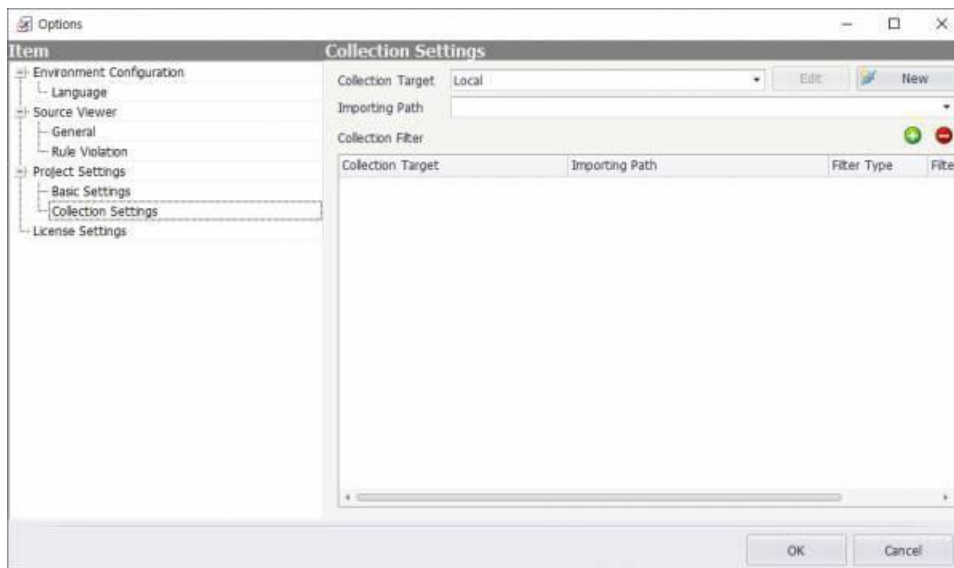


- **Show Summary after Analysis** - Selecting this option prompts the summary window to open after an analysis finishes.

**NOTE:**

- The Server Analysis Mode is a unique feature of beSOURCE. If A project is set on the beSOURCE Server and you want to analyze the same A project in your local PC with beSOURCE Developer, you can use this mode. The analysis configuration such as rule set, required libraries and filter information on the beSOURCE Server will be automatically synchronized with beSOURCE Developer.
- beSOURCE Server administrator can filter out some false positive rule violations from the beSOURCE Server's analysis results. With the Server Analysis Mode, this filtering information will be automatically applied to beSOURCE Developer. For example, the administrator filtered out a SQL injection rule violation in project A, then beSOURCE Developer automatically filters out the same SQL injection rule violation in Project A in Local PC.

- **Project Setting > Collection Setting** -Allows modification of a collection filter.

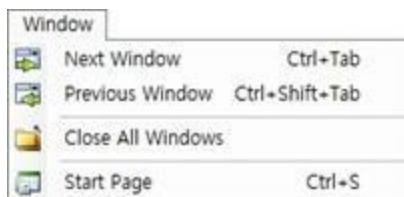


- **Collection Target** - The system that collects or imports source files.
- **Edit** - Modifies current access information for collecting or importing source files.
- **New** - Configures a new collection or importing target.
- **Importing Path** - Specifies the path to import or collect source files.
- **Collection Filter** - Provides options to add, modify, or delete filters for importing source files.

**NOTE:** **License Settings** displays the applied license. The license determines the available languages and rule sets. **Quality Rule** refers to the quality-related rules, and **Security Rule** refers to the secure coding-related rules. The license is applied to the beSOURCE Server.

## Window menu

The Window menu provides the following sub-menus:



- **Next Window** - Activates the next open window in the document area.
- **Previous Window** - Activates the previous open window in the document area.
- **Close All Windows** - Closes all windows in the document area.
- **Start Page** - Opens the Start Page.

## Help menu

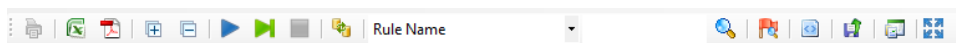
The Help menu provides the following sub-menus:







- **Help** - Opens help information for the active window.
- **Table of Contents** - Lists the contents of online help as hotlinks
- **About** - Opens a dialog with beSOURCE version and trademark information.

## Toolbar

The following toolbar buttons are available:

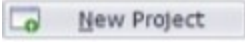






- - Opens a print preview window and shows various options for printing the file.
- - Exports the current active information to an Excel or PDF file.
- - When these icons are active and the “Category View” bar is highlighted in blue, clicking the plus (+) expands the tree of rule names, and clicking the minus (-) collapses the tree. Note that the tree shows only the rules for the standard shown in the Relevant Standard box.
- - Starts or stops an analysis in progress.
- - Manually initiates the collecting of changed source files for import.
- - This combination of dropdown list and text box (to its right) lets you search for instances that match the selected condition in the dropdown and the text in the box. For example, while ‘Rule Name’ is selected, if you enter a keyword from a rule name and click the magnifier icon (‘Search Violation’), you get rule violations corresponding to the condition—if such violations are found. Alternatively, you can select the name of a standard for the current project in the dropdown, type a value, and click the magnifier.
- - Opens the Advanced Search Violations window. This window lets you specify conditions for an advanced search on rule violations.

-  - Opens the Compare Defect History window. Use it to compare rule violations from one analysis to another between specific dates.
-  - Opens a window to upload the results of analysis to the server.
-  - Opens the Start Page.
-  - Toggles the document area between full screen mode or regular mode.

## Start Page

Use Start Page to create a new project or view a summary of an existing project's inspection. The functions and their icons are as follows:

-  - Launches the wizard for creating a new project.
- Fit to Screen (  ) toggles the width of all the columns such that all columns fit in the Start Page window or reverts to a subset of columns (which can be viewed by moving the slide at the bottom of the Start Page window). Also, by closing panes on the left or right side of the Start Page, you can change the column widths.
- Auto Fit (  ) automatically fits the width of all columns.
- Report (  ) generates a report.
- Delete (  ) removes the selected project(s) or analysis history.

The Start Page also provides a summary of the analysis history. If you click the ( ) icon next to the project name, Start Page displays all the analysis history for the project.

Start Page													
New Project...													
Project Name	Path	Language	Ruleset	Elapsed Time	# of Analysis Target Files	TLOC/SLOC/CLOC	Violation Files	Violation Count	New Viola...	Resolved	Remaining Violations	Critical	Rec-H...
My_Project	C:\Users...	C99	MISRA-C-2004	00:00:02	140	4,865/1,949/1,201	139	1,568	0	0	1568	243	49
Analysis DateTime													
2017-06-01...				00:00:02	140	4,865/1,949/1,201	139	1,560	0	0	1560	243	49
2017-05-31...				00:00:02	140	4,865/1,949/1,201	139	1,560	0	0	1560	243	49
2017-05-31...				00:00:19	140	4,865/1,949/1,201	139	1,560	1560	0	0	243	49
My_Project2													
	C:\Users...	C99	CERT C	00:00:01	6	317/68/176	4	32	0	0	32	5	0
Analysis DateTime													
2017-06-01...				00:00:01	6	317/68/176	4	32	0	0	32	5	0
2017-06-01...				00:00:03	6	317/68/176	4	32	32	0	0	5	0
My_Project3													
	C:\Users...	JavaWeb	Java/Web O...	00:00:04	13	1,084/587/188	1	1	1	0	0	1	0
Analysis DateTime													
2017-06-01...				00:00:04	13	1,084/587/188	1	1	1	0	0	1	0

- **Project Name** - The name of the project.
- **Path** - The saved path of the project.
- **Language** - The programming language type.
- **Ruleset** - The applied rule set for this project.
- **Analysis DateTime** - The date and time of the analysis.



- **Elapsed Time** - The total time that elapsed for each analysis of the project.
- **# of Analysis Target Files** - The number of analyzed source files.
- **TLOC/SLOC/CLOC** - The total lines of code/source lines of code/comment lines of code.
- **Violation Files** - The number of source files that have one or more rule violations.
- **Violation Count** - The number of defects or rule violations found in the selected analysis of the project.
- **New Violation** - The number of new rule violations, compared to the preceding analysis.
- **Resolved** - The number of resolved rule violations, compared to the preceding analysis.
- **Remaining Violations** - The number of identical rule violations, compared to the preceding analysis.
- **Priority** - Displays a priority-distribution of rule violations. beSOURCE has five priorities: critical, recommended-high, recommended-middle, recommended-low, and information.

Selecting a project name activates the project.


Selecting the number in a specific project's NEW/Cleared/Same box opens the **Compare Defect History** window.

Selecting the time in the project's **Analysis Time** column activates the project and inspection results for the date and time displayed. You cannot analyze the project with this action, but you can view the result.

# Creating and Analyzing a Project

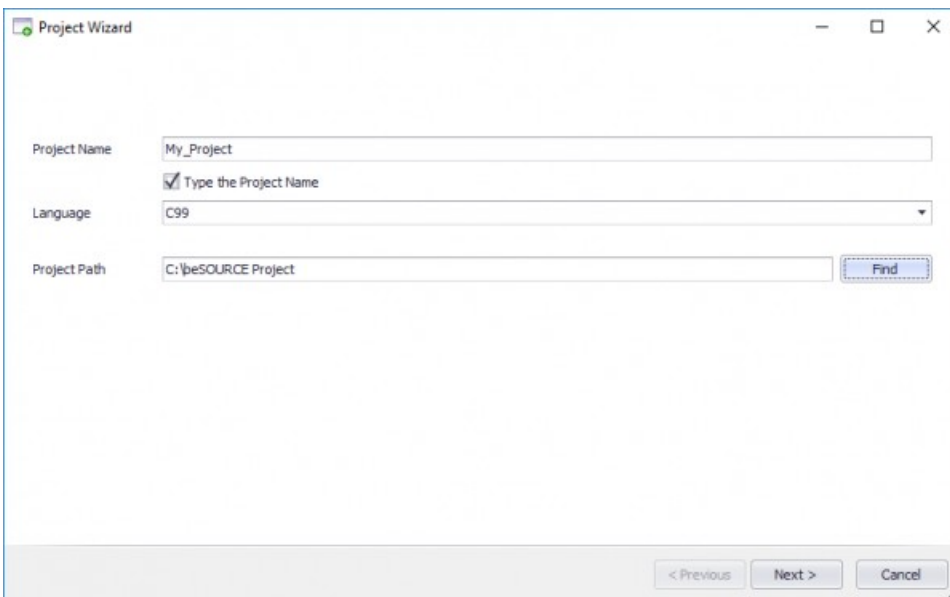
## Creating a project

To create a new project, do the following:

1. On the **Start Page**, select **New Project** (  ). The **Project Wizard** opens.
2. In the **Project Name** box, enter a name for the project. To transfer the analysis results to beSOURCE Server, clear the **Type the Project Name** checkbox and select a predefined project name on the server.

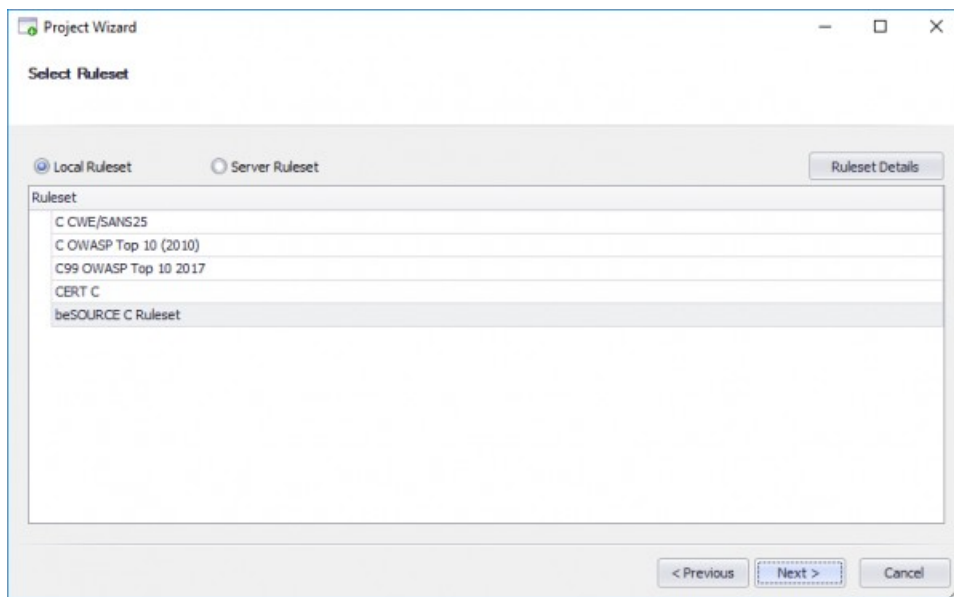
**NOTE:** The Enterprise edition of beSOURCE Developer provides additional options that let you select the project name as it is defined on the server. If the beSOURCE Server administrator registered the project name, you can select a project name from a list.

3. In the **Language** box, select the desired programming language type.
4. In the **Project Path**, use the default path or select Find and choose a different path.
5. Select **Next**.

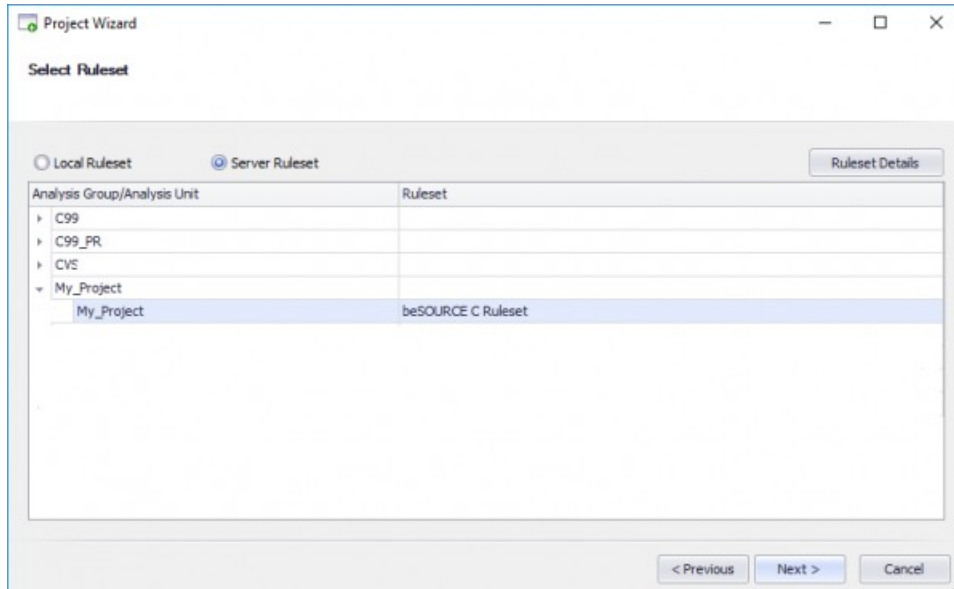


6. Select a one of the following Ruleset options:

- a. **Local Ruleset** - Applies a rule set to the project during analysis and uses the rule set inside the tool. The local ruleset is downloaded from the server. Project analysis can use only one rule set at a time. To select a different rule set for analyzing the project, select **Options > Basic Setting > Ruleset**.

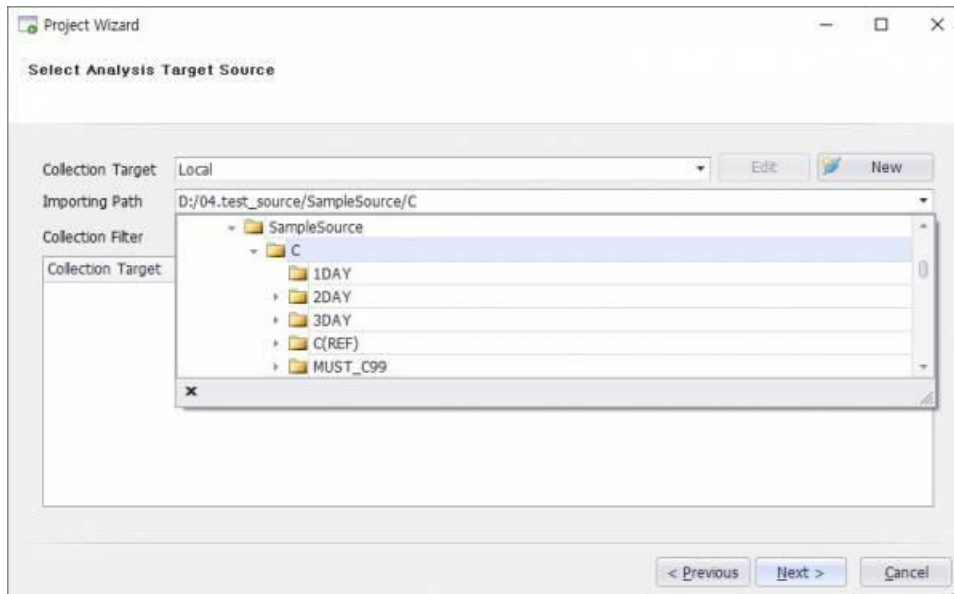


- b. **Server Ruleset** - Uses the predefined analysis configuration in the server. Server-side analysis configurations are displayed if the beSOURCE Server administrator applied the setting.

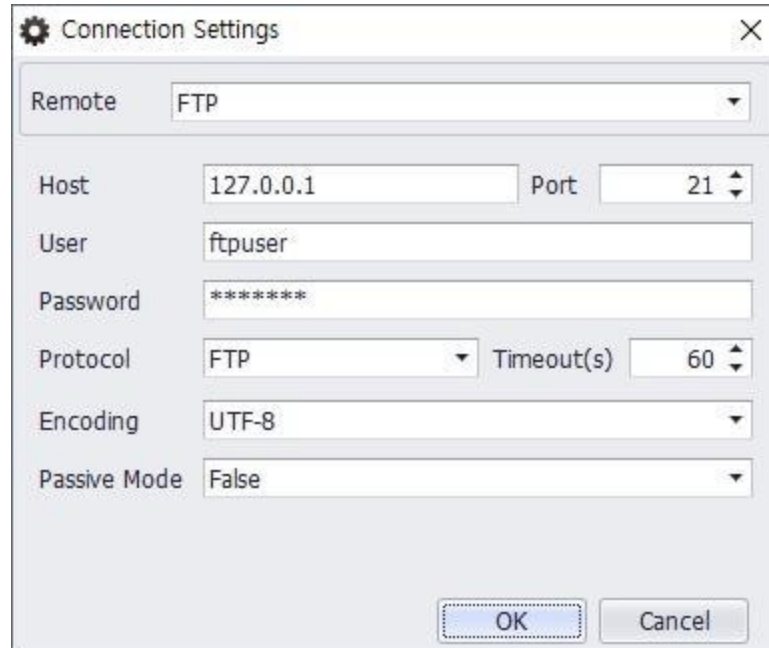


7. Select **Next**.

8. For **Collection Target**, select the location of your source files:
- To import source files from your computer, leave the **Collection Target** box set to **Local**, and then select the location of your files in the **Importing Path** box.

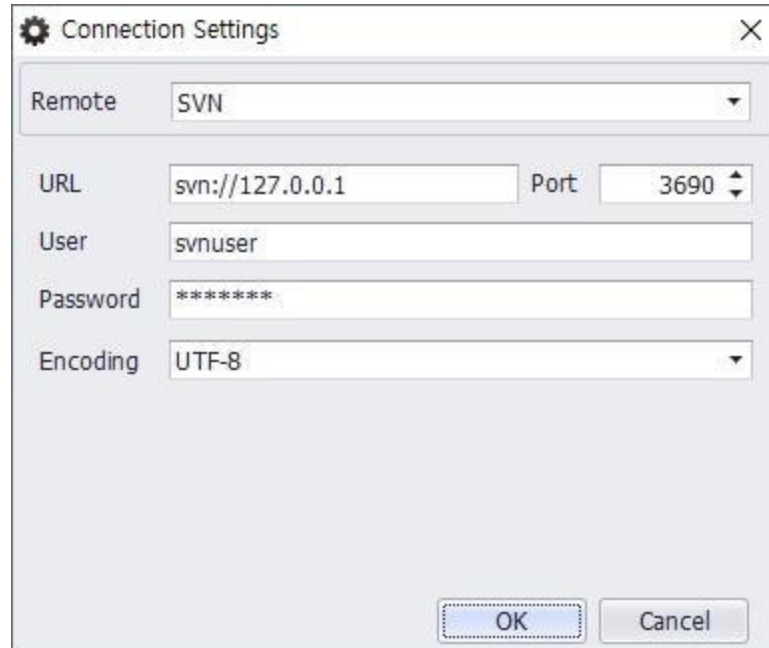


- To import your source files by way of FTP, SVN, CVS, GIT, or TFS, select **New**. The **Connection Settings** dialog will open. In the **Remote** box, select the desired method, and then refer to the following settings to configure it:

i. **FTP**

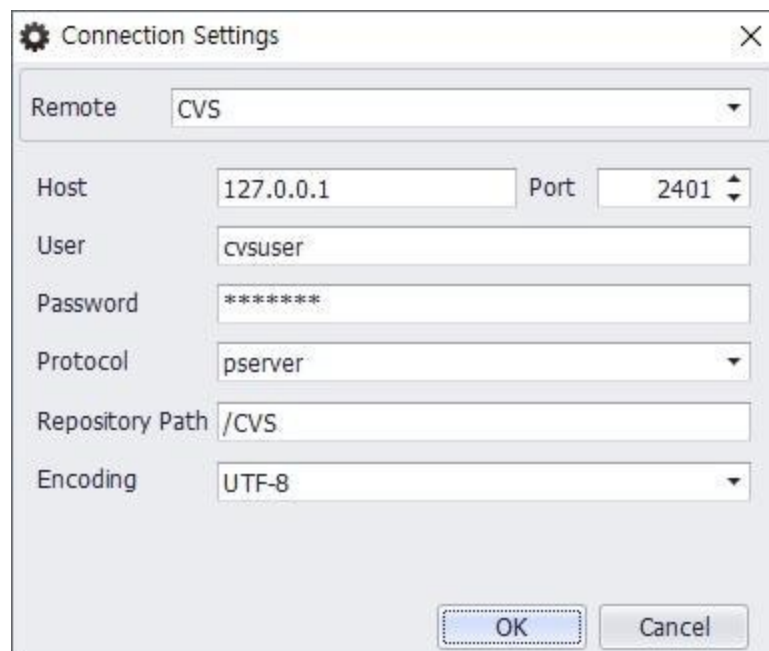
The screenshot shows a 'Connection Settings' dialog box with a gear icon and a close button (X). The 'Remote' dropdown is set to 'FTP'. The 'Host' field contains '127.0.0.1' and the 'Port' spinner is set to '21'. The 'User' field contains 'ftpuser' and the 'Password' field contains '\*\*\*\*\*'. The 'Protocol' dropdown is set to 'FTP' and the 'Timeout(s)' spinner is set to '60'. The 'Encoding' dropdown is set to 'UTF-8' and the 'Passive Mode' dropdown is set to 'False'. At the bottom are 'OK' and 'Cancel' buttons.

1. **Host** - Specifies the FTP server's IP address.
2. **Port** - Specifies the port number (default is 21).
3. **User** - Specifies the user ID to authenticate with the FTP server.
4. **Password** - Specifies the password that corresponds with the user ID.
5. **Protocol** - Specifies the protocol (FTP, SFTP, or FTPS).
6. **Timeout(s)** - Specifies the number of seconds to wait before timing out the connection.
7. **Encoding** - Specifies the encoding for the FTP connection.
8. **Passive Mode** - FTP uses two channels between the client and server—the command channel and the data channel (each channel is a TCP connection). The command channel is for commands and responses and the data channel is for transferring the files. In passive mode, the client establishes both channels, and the server tells the client the port number to use for the data channel.

ii. **SVN**

The 'Connection Settings' dialog for SVN shows the following fields: Remote (SVN), URL (svn://127.0.0.1), Port (3690), User (svnuser), Password (masked with asterisks), and Encoding (UTF-8). The OK button is highlighted with a dashed border.

1. **URL** - Specifies the SVN access address.
2. **Port** - Specifies the port number (default is 3690).
3. **User** - Specifies the user ID to authenticate with the SVN server.
4. **Password** - Specifies the password that corresponds with the user ID.
5. **Encoding** - Specifies the encoding for the SVN connection.

iii. **CVS**

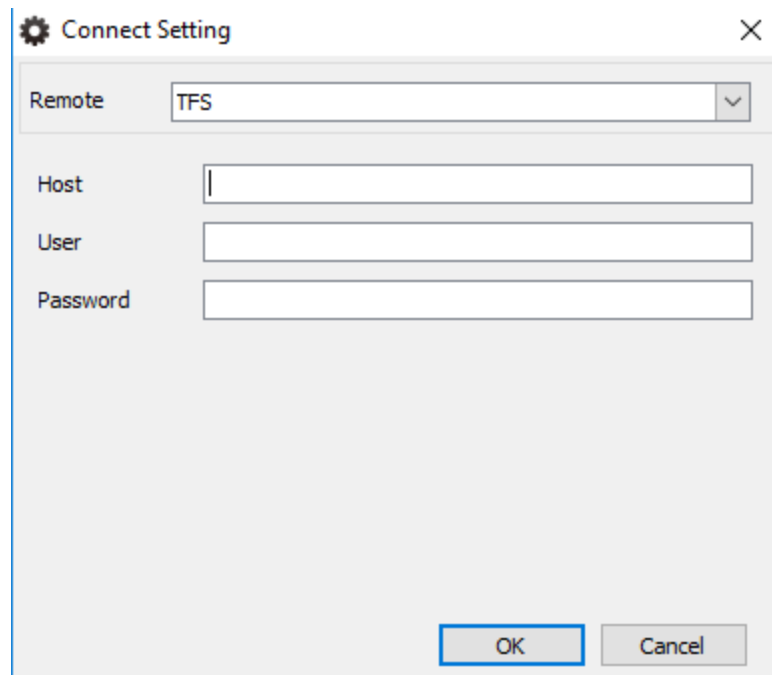
The 'Connection Settings' dialog for CVS shows the following fields: Remote (CVS), Host (127.0.0.1), Port (2401), User (cvsuser), Password (masked with asterisks), Protocol (pserver), Repository Path (/CVS), and Encoding (UTF-8). The OK button is highlighted with a dashed border.

1. **URL** - Specifies the CVS server's IP address.
2. **Port** - Specifies the port number (default is 2401).
3. **User** - Specifies the user ID to authenticate with the CVS server.
4. **Password** - Specifies the password that corresponds with the user ID.
5. **Protocol** - Specifies the protocol (pserver, ext, extssh, or pserverssh2).
6. **Repository Path** - Specifies the path to the source repository.
7. **Encoding** - Specifies the encoding for the CVS connection.

iv. **GIT**

1. **URL** - Specifies the GIT access address.
2. **User** - Specifies the user ID to authenticate with the GIT server.
3. **Password** - Specifies the password that corresponds with the user ID.
4. **Encoding** - Specifies the encoding for the GIT connection.
5. **Use SSH** - Enables SSH for the connection (checkbox is cleared by default). If selected, you must enter the following:
  - a. **SSH-password** - Specifies the SSH password.
  - b. **SSH-private-key** - Specifies the SSH private key.

v. **TFS** (Team Foundation Server)

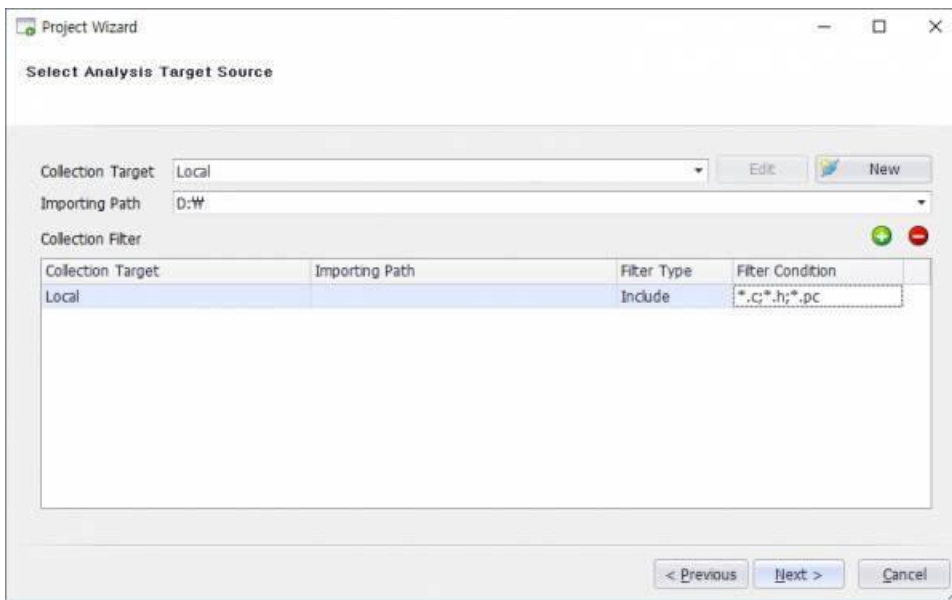
A screenshot of a 'Connect Setting' dialog box. The dialog has a title bar with a gear icon and a close button. Inside, there is a 'Remote' dropdown menu currently set to 'TFS'. Below this are three text input fields labeled 'Host', 'User', and 'Password'. At the bottom right, there are 'OK' and 'Cancel' buttons. The 'OK' button is highlighted with a blue border.

1. **Host** - Specifies the Team Foundation Server IP address.
  2. **User** - Specifies the user to authenticate with the GIT server.
  1. **Password** - Specifies the password that corresponds with the user.
9. For **Collection Filter**, select **Add (+)** to add a new filter condition as needed.

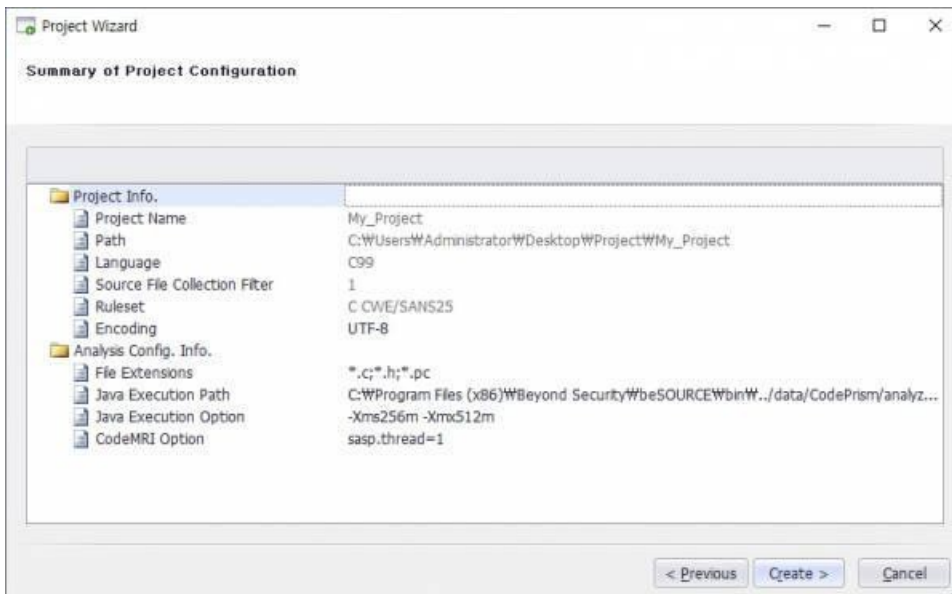
**NOTE:**

- You can specify whether the filtering target is included or excluded while the source files are imported.
- Filter Condition is automatically set according to the selected language, but you can manually modify it. It displays the analysis target's file extensions.

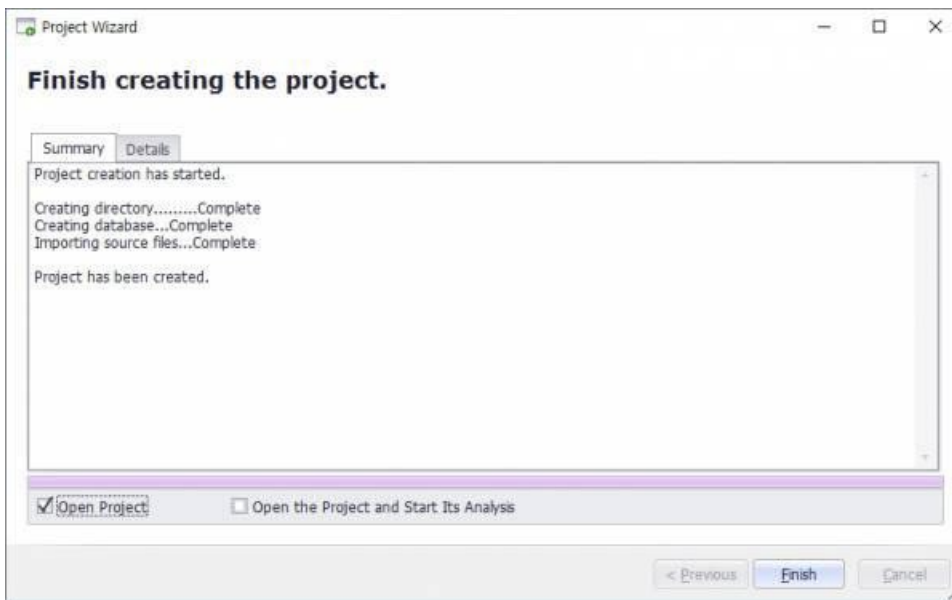




10. Select **Next**.
11. Review the summary of project settings. You can change options, such as encoding or the analysis configuration.




12. If the project settings are correct, select **Create**.
13. Check the results of project creation. To automatically start the analysis after completing the wizard, select the **Open the Project and Start the Analysis** checkbox.



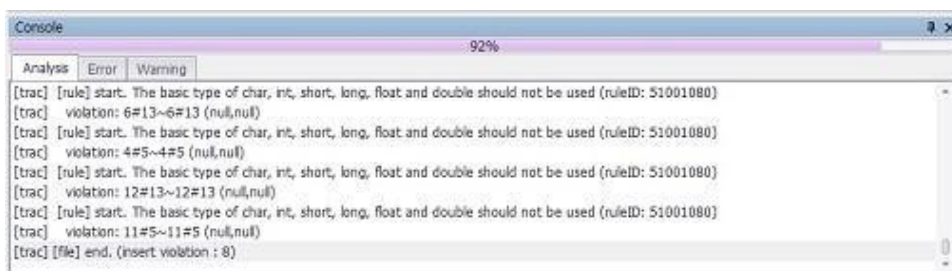
14. Select **Finish** if the project creation appears complete.

## Analyzing a project

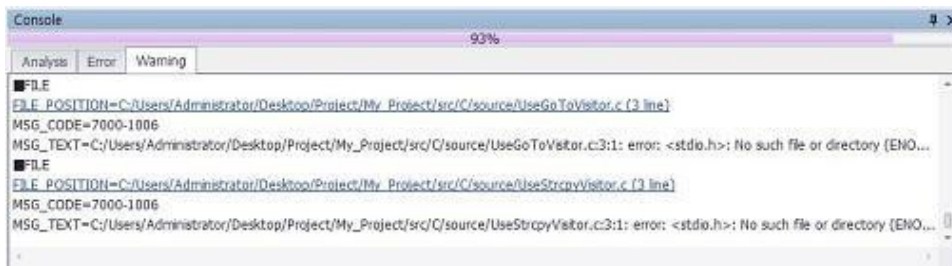
You can analyze source files in the active project. To start analyzing a project, take either of the following two approaches:

- Select **Analysis > Start Analysis**.
- Select the **Start Analysis** (  ) icon from the main toolbar.

While the tool is analyzing the project, you can monitor its progress in the lower part of the Console window.




Error and Warning areas show analysis errors and logs, respectively.




Selecting the hyperlinks in the Error or Warning tab lets you see the actual source code in the Source Viewer.

You can see detailed inspection results in the Rule Violations window after the analysis finishes. You can also see inspection results in the Project window and Category View window.



If you want to stop an analysis while it is in progress, select **Analysis > Stop Analysis** or select the **Stop Analysis** (  ) icon from the main toolbar.

## Analyzing a portion of the source files

You can analyze a portion of the active project by selecting some source files or a folder. To analyze a portion of the source files, do the following:

1. In the Project window, select a folder or the source files.
2. Select the Analysis > Start Partial Analysis or select the (  ) icon in the main toolbar. You can check the progress of the analysis in the Console window.

You can see detailed inspection results in the Rule Violations window after finishing an analysis. The Project window and Category View window also provide the inspection results.

**NOTE:** When you analyze a project, the inspection results are automatically saved (to manage your analysis history). If you want to view the history or remove some of the history, use the Start Page. To delete an analysis history, select the  icon, select an analysis history, and then select the **Delete** (  ) icon. To change the project's settings, use the **Tools > Options** menu.

Project Name	Path	Language	Ruleset	Elapsed Time	# of Analysis Target Files	TLOC/SLOC/CLOC	Violation Files	Violation Count	New Violations	Resolved	Remaining Violations	Critical	Rec-H...	Rec...
My Project	C:\Users...	C99	MISRA-C-2004	00:00:02	140	4,865/1,949/1,201	139	1,560	0	0	1,560	243	49	496
2017-06-01...				00:00:02	140	4,865/1,949/1,201	139	1,560	0	0	1,560	243	49	496
2017-05-31...				00:00:02	140	4,865/1,949/1,201	139	1,560	0	0	1,560	243	49	496
2017-05-31...				00:00:19	140	4,865/1,949/1,201	139	1,560	1,560	0	0	243	49	496

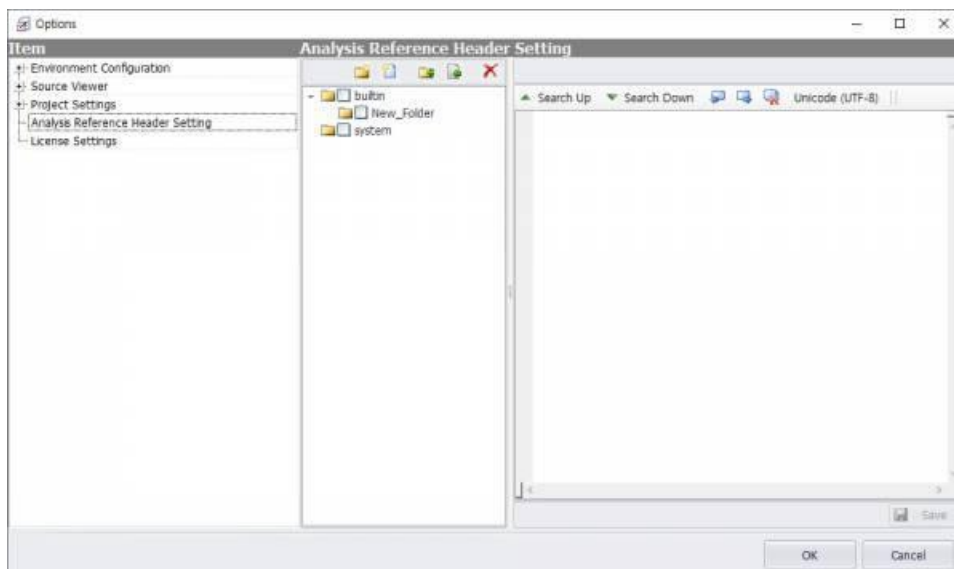
## Setting the referencing header for C/C++

For a precise analysis of C and C++ projects, all header files referenced by the source files must be provided. If any portion of the total required header files is missing for any reason, parsing errors will occur, resulting in an incorrect analysis.




beSOURCE supports a feature for referencing header setting that lets you analyze the project without parsing errors for the case mentioned above.



To set the referencing header, do the following:

1. Create or open a C or C++ project.
2. Select **Tools > Options > Analysis Reference Header Setting**. This option has two categories: builtin and system.
  - a. **builtin** - Allows adding header information about missed user defined functions and types.
  - b. **system** - Allows adding header information about system defined functions and types.





The following tool icons are provided:

- **Add Folder** (  ) - Adds a sub-folder under the selected item.
- **Add Header File** (  ) - Adds a header file under the selected item.
- **Import Folder** (  ) - Imports a folder that contains header files.

- **Import Header File** (  ) - Imports header files.
- **Delete** (  ) - Deletes the selected header file.

## Add missing user-defined header information



To add missing user-defined header information, do the following:

1. Select the **builtin** category.
2. If necessary, make a sub-category by selecting the **Add Folder** (  ) icon.
3. Select the **Add Header File** (  ) icon.
4. Change the name of an added header file within the builtin category.
5. Double-click the new header file and edit its content with additional header information.
6. Select **Save**.
7. Check the built-in header file to be included in your project.
8. Select **OK**.

You can add or edit user-defined declarations or macros in the header file that was added to the built-in category. The checked header files under the built-in category are expanded with the analysis target source files by the internal analysis engine.



## Adding missing system header information

To add missed system header information, do the following:

1. Select the **system** category.
2. If necessary, make a sub-category by selecting the **Add Folder** (  ) icon.
3. Select the **Add Header File** (  ) icon.
4. Change the name of the added header file in the system category.
5. Double-click the added header file and edit its content.
6. Select **Save**.
7. Check the system header file to be included in your project.
8. Select **OK**.

You can add system-related header information to a header file under the system category (for example, Windows win32 API or a library of UNIX and GCC can be added).

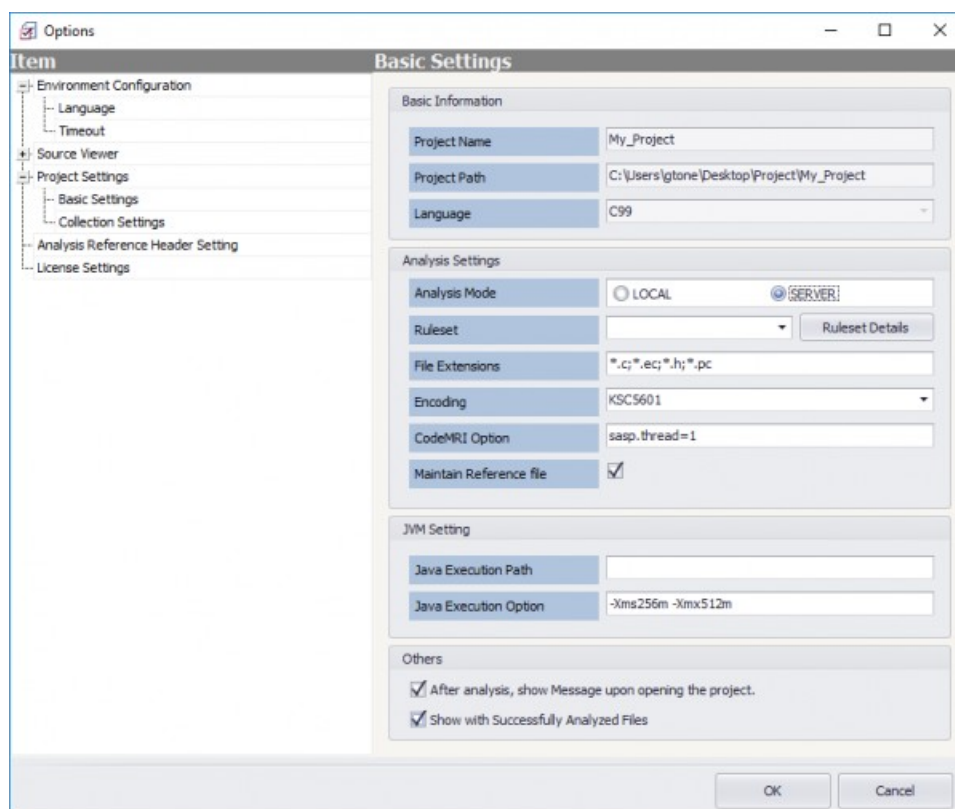
The checked header files under the system category are expanded with analysis target source files by internal analysis engine.

If you have predefined header file information, you can import it by using either Import Folder (  ) or Import Header File (  ) icons.

## Analysis engine options (CodeMRI Options)

To specify or change the analysis engine's options, select **Tools > Options > Project Setting > Basic Setting** and then edit the CodeMRI box (see possible values in the figure below.)

The following are the default option values:



- **sasp.thread=1** The thread count for parallel processing. A sasp.thread value of 0 means analyzing all targets at once.
  - A 1 means analyzing one target at a time after all targets are grouped. A value of 2 or more specifies parallel analysis after all targets are grouped.
  - The thread count should be set to the number of physical CPU cores or 'physical CPU core count -1'. If the thread count is specified by the number of logical CPU cores, the consequence is bad performance for a large project.

- **sasp.clear=true** - Removes all temporary files and reference information generated by the last analysis and then analyzes source files. A 'sasp.clear=false' setting reuses temporary files and reference information from the last analysis.
- **sasp.indiv** - Analyzes source files one at a time. (So function calls through multiple files are ignored, but function calls within one file are handled during the analysis).
- **sasp.info** - Provides results in more detailed logs.

To enter multiple option values, separate each with a comma and no space (for example, sasp.thread=4,sasp.indiv).

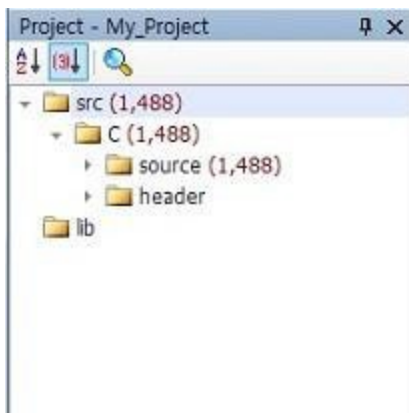
**NOTE:** CodeMRI options are mainly used for security rules or potential runtime error rules.

# User Interface

## Project window

The Project window shows the source files for an active project and lets you analyze some of them.

To open the Project window, select **View > Project**.



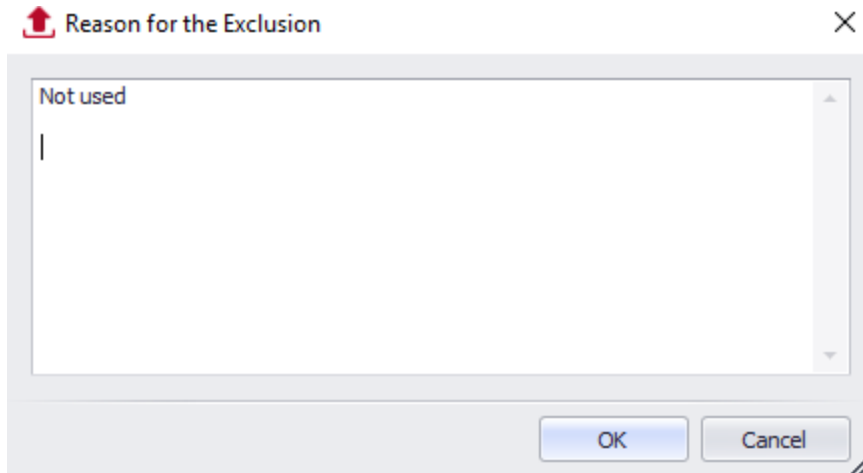
This panel shows directory structures and source files the project wizard has imported. It also shows a red number for the defects or rule violations next to each directory and source file (with red color).

Double-clicking a source file in the Project window opens the source viewer in the central document area. It provides the following context menus:

- **Start Partial Analysis** - Starts analysis for the selected source files or folder only.
- **Exclude** - Excludes some folder or files in analysis. Optionally, add a comment for



the exclusion in the dialog.






If you exclude a folder, all the source files in the folder are excluded from the analysis job. Excluded files are displayed with inactive status and any comments you added are shown in the tool tip window. You can later include the excluded folder or files. If you do so, any comment for the excluded file will be initialized.

**NOTE:** Excluded source files are imported but are not analyzed.



The Project window provides the following function icons:

- **Sort by File Name** (  ) - Sorts the source files by their name.
- **Sort by Defects Count** (  ) - Sorts the source files by the number of defects.
- **Find** (  ) - Performs a simple search.

## Category View window






The Category View window shows defects or rule violations by a predefined category.

To open the Category View window, select **View > Category View**. It lists inspection rules with the number of defects in parentheses and red color.



Selecting any category (other than All) from the box in the upper-left corner will display the rules for that category. Selecting a rule in the list synchronizes information in other corresponding windows, such as the rule description and rule violation.

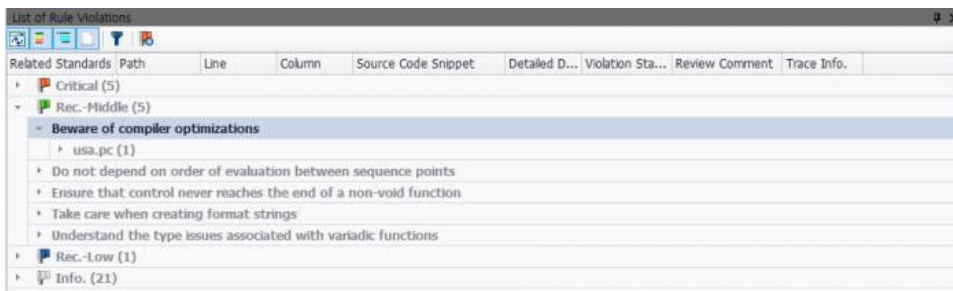
Category View provides the following tool icons:

- **Sort by Priority** (  ) - Sorts rules by their priority.
- **Sort by Rule Name** (  ) - Sorts rules by their name.
- **Sort by Defects Count** (  ) - Sorts the rules by the number of violations.
- **Sort by Standard Name** (  ) - Sorts the rules by their related international standard.
- **Find** (  ) - Provides a simple search function.







## List of Rule Violations window

The List of Rule Violations window shows all defects or rule violations along with detailed information. Because the List of Rule Violations window is also synchronized with the Project and Category View window, the corresponding information is automatically displayed when you select an item in those windows.

To open the List of Rule Violations window, select the **View > List of Rule Violations**. The List of Rule Violations window shows information about rule violations or defects by grouping with priority, rule name, and file name.



The List of Rule Violations window provides the following tool icons:

- **Synchronize** (  ) - Synchronizes the List of Rule Violations window (or pad) with the Project window and Category window.
- **Grouping by Priority** (  ) - Sorts the violation list by the rule's priority.
- **Grouping by Rule Name** (  ) - Sorts the violation list by the rule's name.
- **Grouping by file name** (  ) - Sorts the violation list by source file's name.
- **View Excluded Items** (  ) - Displays excluded defects or rule violations that were excluded from the list by your configuration of violation status.
- **Show Rule Violations for Analyzed Source Only** (  ) - Displays defects or rule violations for only the source files that were recently analyzed.

If you select a rule name, the description of it appears in the Rule Description window.

If you select a source file name, its flow trace and defect history appear in the Trace Info. window and Defect History window.

If you double-click a source file name, actual source code appears in the Source Viewer.

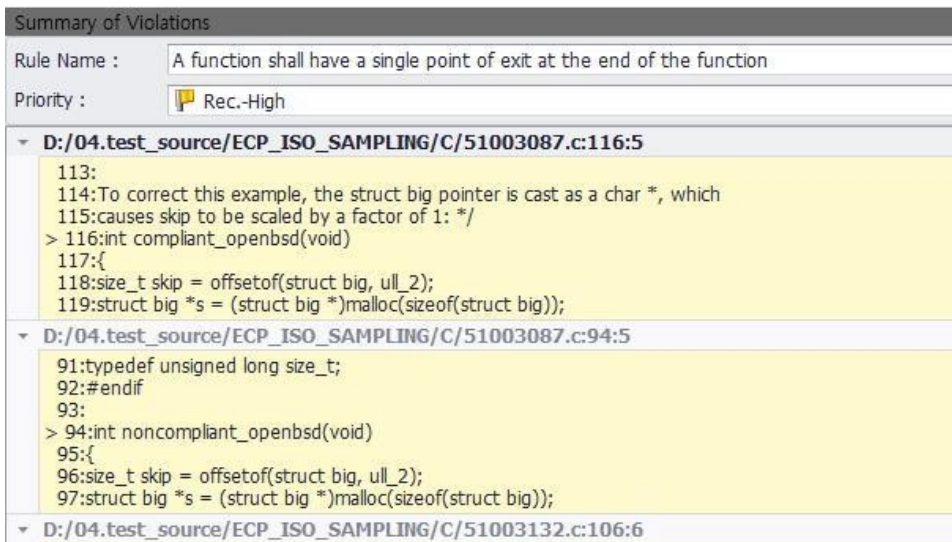
**NOTE:** For more information about setting violation status, refer to Violation Status.

## Summary of Violations

The Summary of Violations provides summaries of source code for the inspection rule you select. This information helps you understand the coding patterns that were detected as violations of the selected rule.

To open the Violation Source Summary window, do the following:

1. In the Category View, select a rule that has violations.
2. Select **View > Summary of Violations** to see the window illustrated below.

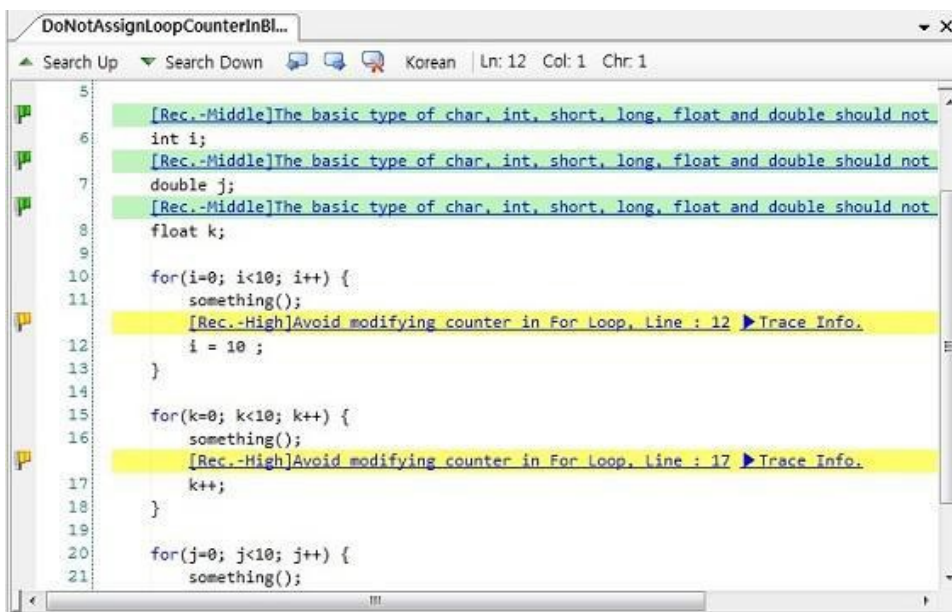


## Source Viewer

The Source Viewer shows source code with defects or rule violation information.

To open the Source Viewer, take the appropriate action from the following:

- Double-click a source file in the Project window.
- Double-click a source file in the Defect Info. window.



The Source Viewer displays followings:

- On the left-side, the color-coded flag indicates the detected rule's priority. This line is immediately above the detected code line (same priority color as the flag). The summary shows the rule's name, line number, and more.
- Selecting a rule name's link opens the Rule Description window.
- If the defect has flow trace information, the line has a ► **Trace Info** link. Select the link to open the Trace Info. window.

**NOTE:**

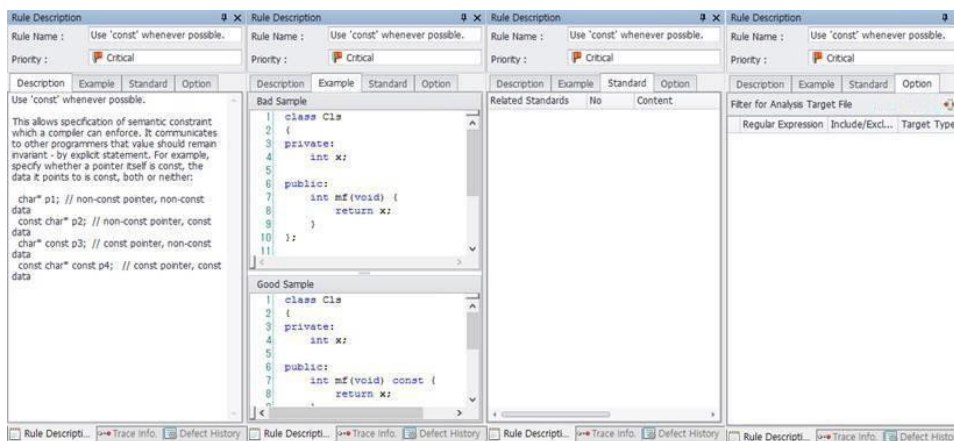
- You cannot edit the source code in the Source Viewer window.
- The Source Viewer provides a 'Go to Explorer' context menu. If you select this menu, the corresponding nodes in Project Explorer will have focus.

## Rule Description window

The Rule Description window provides a description, example code, related standards, and option for the rule you selected.

To open the Rule Description window, select **View > Rule Description**. The window opens and displays the following tabs:

- **Description** - Provides detailed information on a rule selected in the Category View or Defect Info window.
- **Example** - Provides good example and bad example source code about a rule selected in the Category View or Defect Info window.
- **Standard** - Provides international standard information related to a rule selected in Category View or Defect Info window. If your PC has Internet connection, you can open a web page of the standard by clicking hyperlink.
- **Option** - Provides option of rule selected in Category View or Defect Info window.

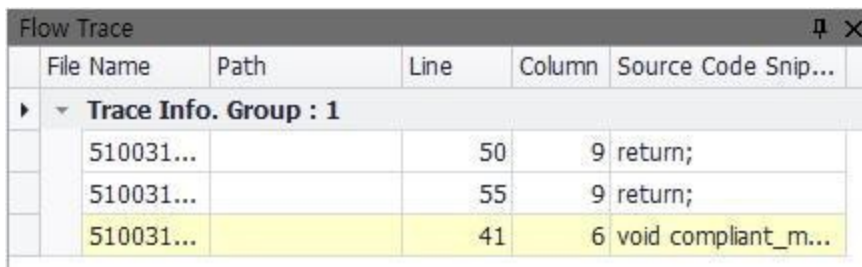


## Flow Trace window

The Flow Trace window provides flow trace information on a specific defect or rule violation. For example, if the tool detects a defect in the source file's line number 15, Trace Info. displays source code flows related to that line, allowing you to trace back to other source code lines that affect the detected rule violation.

To open the Trace Info. window, select **View > Flow Trace**. If you double-click on a row in the Trace Info window, the Source Viewer opens and puts the cursor at the corresponding line.


**NOTE:** Not every rule can provide flow trace information, but if you select a rule that does support flow trace, the information appears in the window.

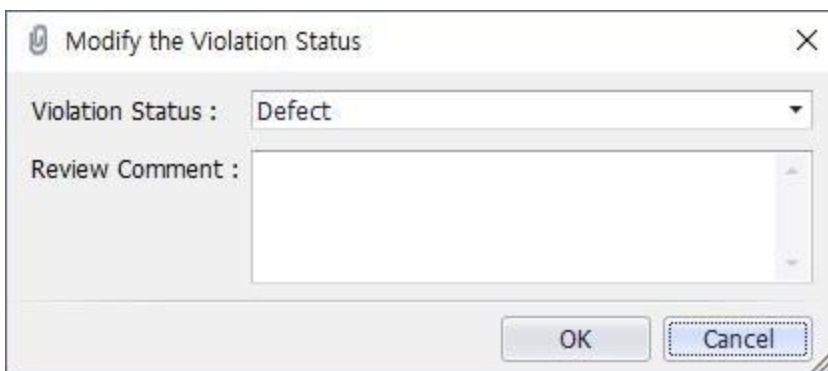


File Name	Path	Line	Column	Source Code Snip...
<b>Trace Info. Group : 1</b>				
510031...		50	9	return;
510031...		55	9	return;
510031...		41	6	void compliant_m...

## Violation Status

You can manage the status of one or more rule violations (defects), as follows:

1. Select a source file or rule violation in the List of Rule Violations window.
2. Select **Modify the Violation Status** (  **Modify the Violation Status** ) to open its dialog.



**Modify the Violation Status**

Violation Status : Defect

Review Comment :



OK Cancel

3. In the Violation Status, select one of the following statuses:
  - a. **Defect** - Confirmed as a defect or rule violation.
  - b. **Not Defect** - Confirmed as a false alarm or the probability of a false alarm is high.
  - c. **Ignorable** - Confirmed as ignorable defect by user review.

- d. **Cleared** - Confirmed the defect is resolved.
- e. **Not Reviewed** - Review has not been completed.

If you specify a status for the defect, the List of Rule Violations window shows it.

If you set a defect or rule violation to **Not Defect**, **Ignorable**, or **Cleared**, the default list in List of Rule Violations window and generated report will exclude it.

Selecting **View Excluded Items** (  ) icon displays the excluded items by setting a violation status for them. To revert the status of defects, use the Modify the Violation Status context menu again. Selecting the **View Excluded Items** (  ) icon again displays the original defects list.

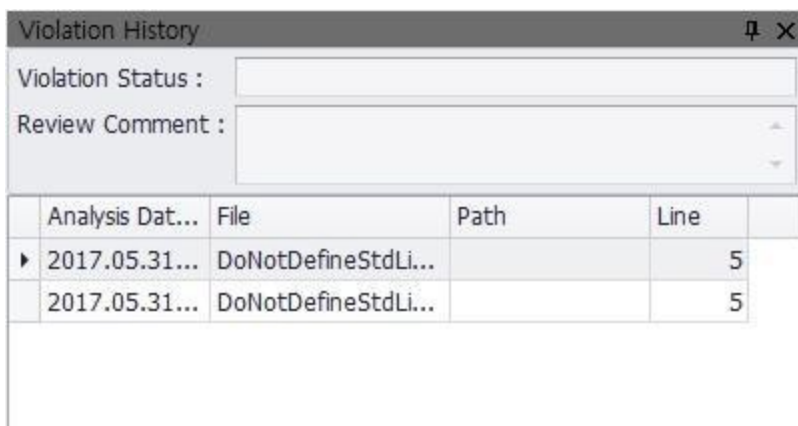
**NOTE:** In Enterprise edition, you can transfer the inspection result to the server. A manager can review the result and communicate about it with others.

## Violation History

The Violation History window displays the history of a specific defect or rule violation selected from the Defect Info window.

To view a defect or rule violation's history, do the following:

1. Select **View > Violation History**.
2. In the Defect Info window, select a defect. You can check the history of the same rule violation based on its time of analysis.



Analysis Dat...	File	Path	Line	
2017.05.31...	DoNotDefineStdLi...		5	<input type="checkbox"/>
2017.05.31...	DoNotDefineStdLi...		5	<input type="checkbox"/>

## Comparing History

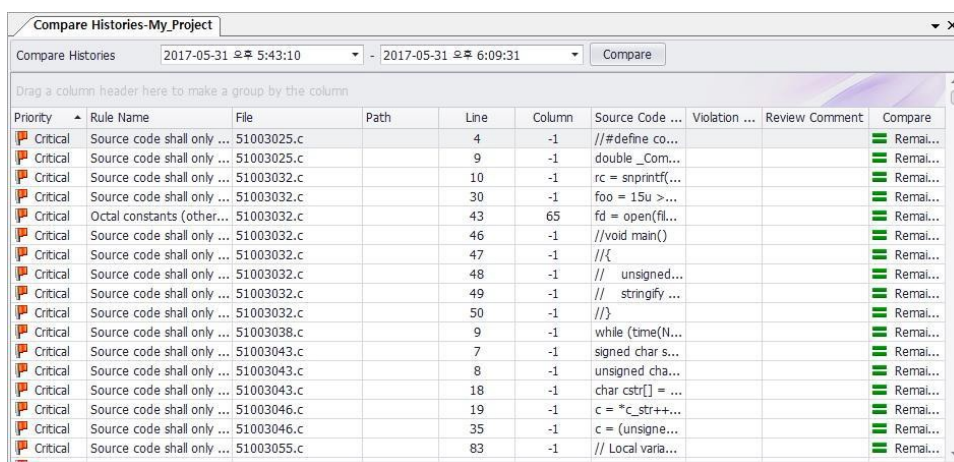


The beSOURCE tool provides useful functionality for intuitively comparing the current inspection result and the recent defects history for a project.

**NOTE:** You must have more than two analysis results for a project to make a comparison.

To compare defect histories, do the following:

1. Open a project.
2. Select **Report > Compare History** in the main menu or click the **Compare Defect History** icon. A window opens that will display the comparison results.
3. Alternatively, select two analysis dates/times.
4. Select the **Compare** button.



Priority	Rule Name	File	Path	Line	Column	Source Code ...	Violation ...	Review Comment	Compare
Critical	Source code shall only ...	51003025.c		4	-1	//#define co...			Remai...
Critical	Source code shall only ...	51003025.c		9	-1	double _Com...			Remai...
Critical	Source code shall only ...	51003032.c		10	-1	rc = snprintf(...			Remai...
Critical	Source code shall only ...	51003032.c		30	-1	foo = 15u >...			Remai...
Critical	Octal constants (other...	51003032.c		43	65	fd = open(fil...			Remai...
Critical	Source code shall only ...	51003032.c		46	-1	//void main()			Remai...
Critical	Source code shall only ...	51003032.c		47	-1	//{			Remai...
Critical	Source code shall only ...	51003032.c		48	-1	// unsigned...			Remai...
Critical	Source code shall only ...	51003032.c		49	-1	// stringify ...			Remai...
Critical	Source code shall only ...	51003032.c		50	-1	//}			Remai...
Critical	Source code shall only ...	51003038.c		9	-1	while (time(N...			Remai...
Critical	Source code shall only ...	51003043.c		7	-1	signed char s...			Remai...
Critical	Source code shall only ...	51003043.c		8	-1	unsigned cha...			Remai...
Critical	Source code shall only ...	51003043.c		18	-1	char cstr[] = ...			Remai...
Critical	Source code shall only ...	51003046.c		19	-1	c = *c_str++...			Remai...
Critical	Source code shall only ...	51003046.c		35	-1	c = (unsigne...			Remai...
Critical	Source code shall only ...	51003055.c		83	-1	// Local varia...			Remai...

If you double-click an item in the list, the source viewer opens with the corresponding code line.


## Advanced search for Rule Violations

Advanced Search can use one of the following bases when searching for rule violations:

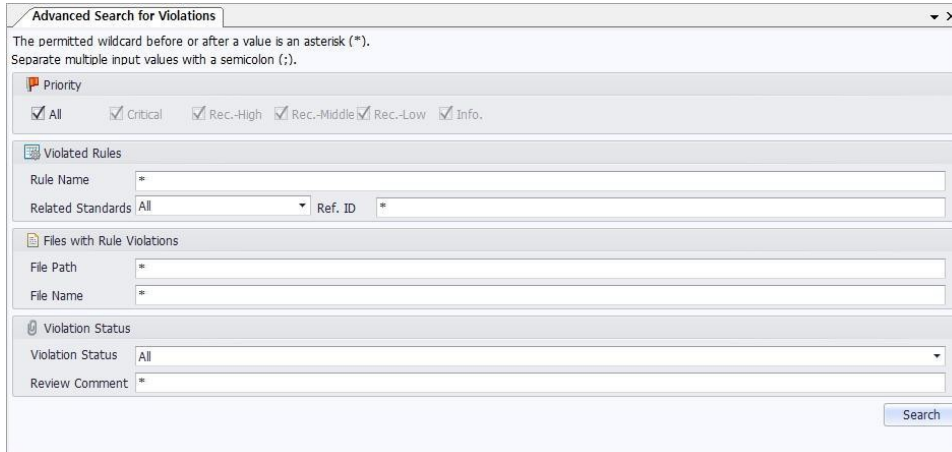
- Rule Priority
- Rule name and related standard
- File path and name
- Defect status and your review comment

To perform an advanced search, do the following:



1. Open the **Advanced Search** window by doing one of the following:
  - a. Open a project. The Advanced Search window automatically opens.
  - b. Select the (  ) icon in the main toolbar.
2. Specify the search conditions, as illustrated in the figure below.
3. Select **Search**. The search results appear in the Defect Infoz window.

In the search conditions, you can use a (\*) as a wild card. Use a semicolon (;) as a delimiter between multiple search conditions.



**Advanced Search for Violations** [Close]

The permitted wildcard before or after a value is an asterisk (\*).  
Separate multiple input values with a semicolon (;).

**Priority**

☒ All ☒ Critical ☒ Rec.-High ☒ Rec.-Middle ☒ Rec.-Low ☒ Info.

**Violated Rules**

Rule Name \*

Related Standards: All Ref. ID \*

**Files with Rule Violations**

File Path \*

File Name \*

**Violation Status**

Violation Status: All

Review Comment \*

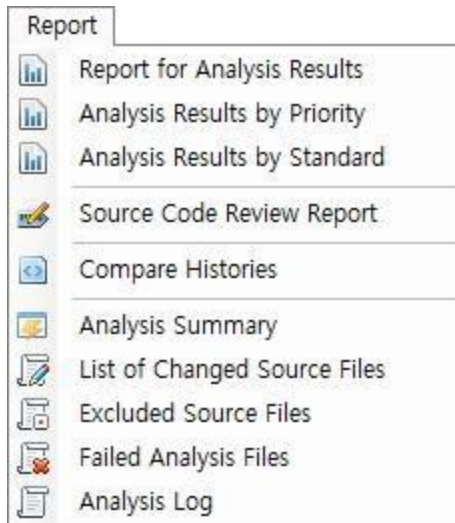
**Search**

# Reports

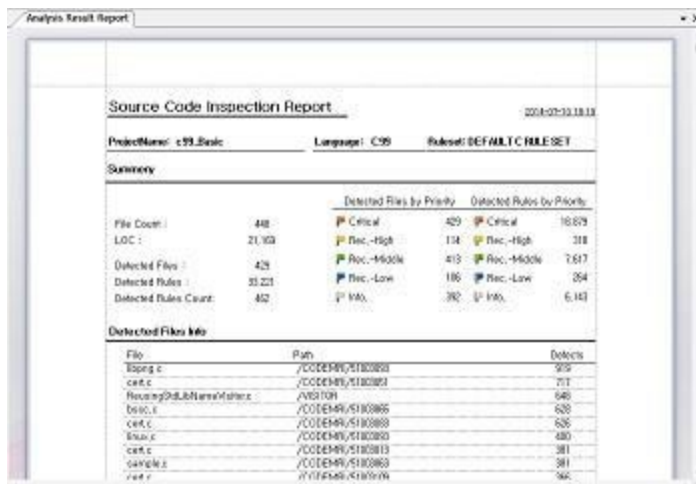
## Creating reports for inspection results

To generate a report for beSOURCE inspection results:

1. Open a project.
2. Select the **Report** menu.



3. Select a sub-menu for the specific report type.
4. The report is shown in the central document area.



5. Select the **Excel** or **PDF** button in the main toolbar. Alternatively, select **File > Export** menu.
6. Specify a name for the report file and save it.

# Report types

The following report types are available by default.

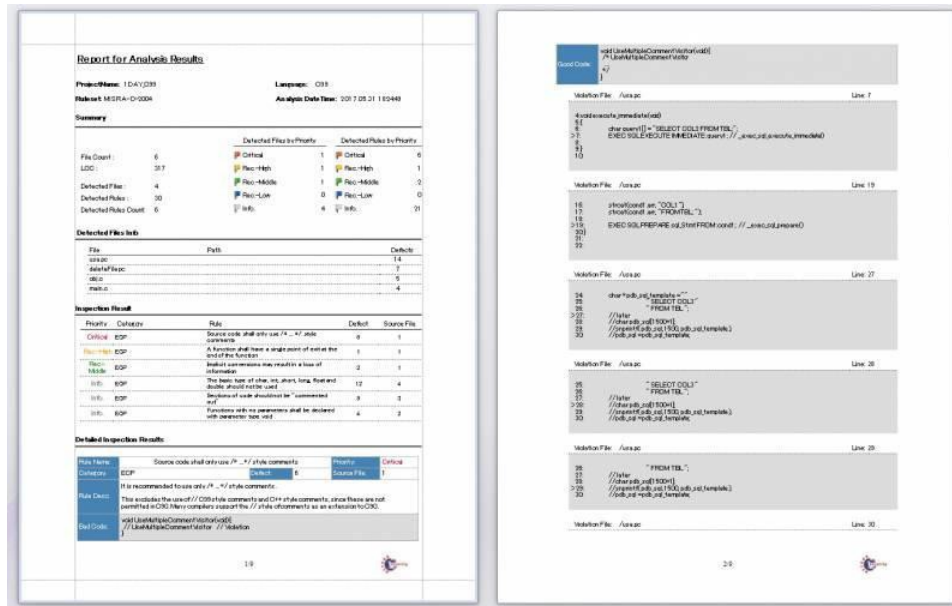
- **Report for Analysis Results**
- **Analysis Result by Priority**
- **Analysis Result by Standard**
- **Source Code Review Report**
- **Compare Histories**
- **Analysis Summary**
- **List of Changed Source Files**
- **Excluded Source Files**
- **Failed Analysis Files**
- **Analysis Log**

## Reporting Analysis Results

The Analysis Result Report includes following datasets:

- **Summary** - Displays information about the number of source files within a project, LOC, the number of files with defects, total defect count, and defect count by priority.
- **Defect files** - Displays a list of source files that have rule violations or defects.
- **Inspection result by rule** - Displays a list of detected rule and defect counts for the rules.
- **Details** - Displays rule description, sample code, and code snippet of actual defects

for each rule.



To create a report with a subset of the rule priorities, select **Report > Analysis Result Report (by Priority)** and turn off any of the priorities (Critical, Rec-High, Rec-Middle, Rec-Low, and Info.).



To select the international standard that the report will reflect, select the **Report > Analysis Result Report (by Standard)** menu. The window that pops up is illustrated below:



# Source Code Review Report

The Source Code Review Report lets you download an Excel file that includes a list of rule violations.







1	A	B	C	D	E	F	G	H	I	J	K	L	M	N									
	Priority	Rule Name	Related Standards	Collection Target	Defect File	Path	Line	Column	Violation Source Summary	Detail	Defect Status	Review Comment	Manager Comment	Developer Review Result									
2	Critical	Avoid reusing DB resources		Local	UserDAO.java	/db	394	25	201 print stmt(1, Integer.parseInt(id)); 202 stmt.executeUpdate(stmt); 203 204 stmt = 205 conn.createStatement(); 206 print stmt(1, Integer.parseInt(id)); 207 stmt.executeUpdate(stmt); 208 conn.close(); 209 public static String 210 CH_SQL_ASSOCIATION = "top"; 211 212 private static Constants singleton ()= 213 new Constants(); 214 215 public static Constants getInstance() 216 { 217 return singleton(); 218 } 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000														
3	Critical	Defensively copy private mutable class members before returning their references	[CERT JAVA0005-J]	Local	Constants.java	/base	23	34	23 private static Constants singleton ()= 24 new Constants(); 25 26 public static Constants getInstance() 27 { 28 return singleton(); 29 } 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000														
4	Critical	Empty Method in Abstract Class Should Be Abstract		Local	Abuser.java	/base	17	23	17 public abstract class Abuser { 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 														

The Analysis Summary provides information on analysis configuration, source file changes compared to preceding analysis, inspection results, and detected rule violations.

- **Changed Source Files** - Provides information on changed (added, deleted, and modified) source files counts in the project. The tool automatically imports the changed source files when you restart a project analysis. If you select a chart, you can move it to the Source File Changes List window.
- **Result of Source File Analysis** - Provides the number of excluded, failed, success, and defect files. If you select the excluded and the failed files chart, you can move to the Excluded Files List to the Failed Files List window.
- **Changes in Violations** - Provides the number of new, cleared, and remaining defects, comparing with the last analysis result.
- **Violation Count for Each Priority** - Provides the number of defects (rule violations) by priority.
- **Violation Trend for Each Priority** - Displays trends in the counts for rule violations (defects) by priority.
- **View Log** - Opens the Analysis Log window.

## List of Changed Source Files

When you analyze a project again, the tool automatically checks file changes in the project and again imports only the changed files. The report shows a detailed list of added, deleted, and modified files in the project.

List of Changed Source Files - ...		
2017.05.31 18:24:49		
Path	File Name	Reason for Change
	deleteFile.pc	 Add
	main.c	 Add
	obj.c	 Add
	obj.h	 Add
	sql.h	 Add
	usa.pc	 Add

## Excluded Source Files

If you exclude files in a project analysis for any reason, the analysis report shows the list of excluded files and written comments (if comments exist).

Excluded Source Files - 1DAY_...		
Path	File Name	Reason for the Exclusion
	obj.c	Not Use.

## Failed Analysis Files

If the project analysis generates errors for source files, the report shows a list of source files whose analysis failed and related error messages.

Failed Analysis Files - 1DAY_C...				
2017.05.31 18:30:29 <input type="checkbox"/> show with analysis success files				
Path	File Name	Succeeded   Fal...	Error	Line
	obj.c	Analysis Faile...	[CPSA-2000] Parsing error.	0

## Analysis Log

beSOURCE displays a log of the analysis of your project.

```

Analysis Log - My_Project
2014.05.26 14:53:51
1 [info] init project...
2 [trac] [RuleSync] rule sync start.
3 [trac] [RuleSync] base : C:\Program Files (x86)\GfOne\CodePrism\data\CodePrism\lib\data.db, project : C
4 [trac] [RuleSync] already has been updated...
5 [trac] [RuleSync] rule sync end. (128es)
6 [info] [SourceCollector] start collect
7 [trac] [Collector] ID: 20140526144300> file filter <INCLUDE> "*.c;*.h;*.pc" matches "E:/04.test_source/!
8 [trac] [Collector] ID: 20140526144300> file filter <INCLUDE> "*.c;*.h;*.pc" matches "E:/04.test_source/!
9 [trac] [Collector] ID: 20140526144300> file filter <INCLUDE> "*.c;*.h;*.pc" matches "E:/04.test_source/!
10 [trac] [Collector] ID: 20140526144300> file filter <INCLUDE> "*.c;*.h;*.pc" matches "E:/04.test_source/!
11 [trac] [Collector] ID: 20140526144300> file filter <INCLUDE> "*.c;*.h;*.pc" matches "E:/04.test_source/!
12 [trac] [Collector] ID: 20140526144300> file filter <INCLUDE> "*.c;*.h;*.pc" matches "E:/04.test_source/!
13 [trac] [Collector] ID: 20140526144300> file filter <INCLUDE> "*.c;*.h;*.pc" matches "E:/04.test_source/!
14 [trac] [Collector] ID: 20140526144300> file filter <INCLUDE> "*.c;*.h;*.pc" matches "E:/04.test_source/!
15 [trac] [Collector] ID: 20140526144300> file filter <INCLUDE> "*.c;*.h;*.pc" matches "E:/04.test_source/!
16 [trac] [Collector] ID: 20140526144300> file filter <INCLUDE> "*.c;*.h;*.pc" matches "E:/04.test_source/!

```

# Quick Start: Analyzing C/C++ Source Files

## Overview

This section describes the procedure for analyzing C or C++ source code.

## What you need to know

One characteristic of the C and C++ languages is that they are expanding-type languages. They merge different types of files, such as a .c file or .cpp file into one expanded file and a .h file, and then try to compile it. The pre-processing macro that merges and expands the source files is `#include`. Every time (most) C or C++ compilers meet a `#include` macro in source files during the pre-processing phase, they try to merge the original file and target 'include' file. Therefore, if any header files (which define the types used in .c or .cpp files) are missed, parsing errors occur, and the sources files cannot be correctly analyzed.

To analyze C or C++ source files, you must prepare for all the necessary header files.

C sample source files are provided. You can find them in the `beSOURCE_Server_Installation_Folder\Sample\C` folder.

C++ sample source files are provided. You can find them in the `beSOURCE_Server_Installation_Folder\Sample\CPP` folder.

Copy the sample source files to your PC.

## How to analyze C source files

To analyze source files, do the following:

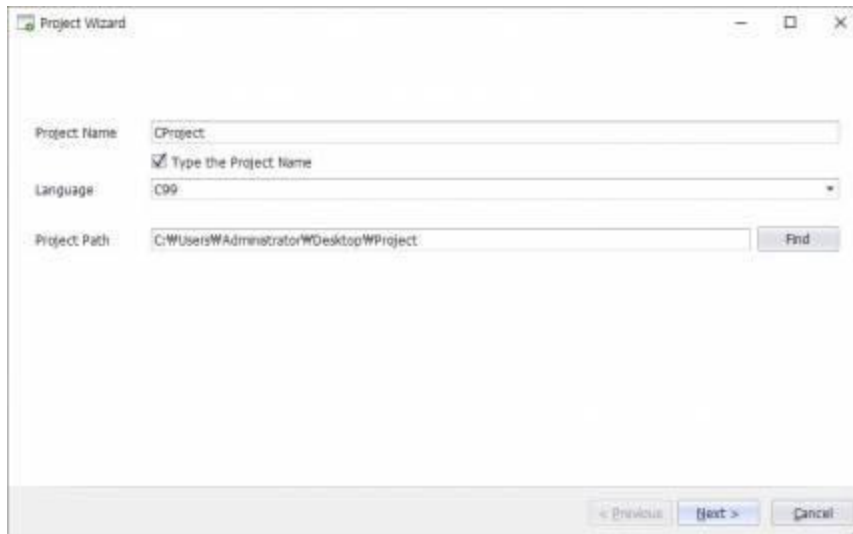
1. On the **Start Page**, select **New Project**. The **Project Wizard** opens.

**NOTE:** To analyze source files, you must create a new project.

2. On the first page of the wizard, do the following:

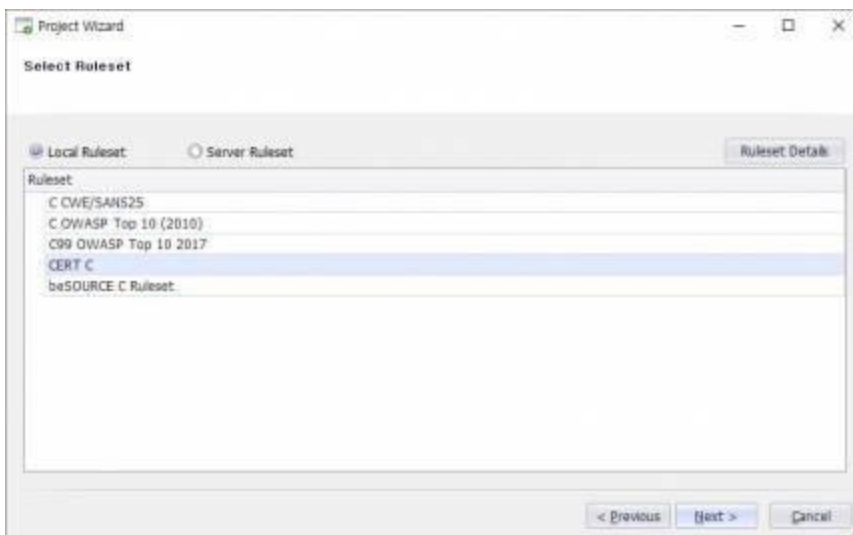


1. In the **Project Name** box, enter a name for the project.
2. In the **Language** box, select **C99**.
3. In the **Project Path** box, use the default location, or select **Find** to specify a location.




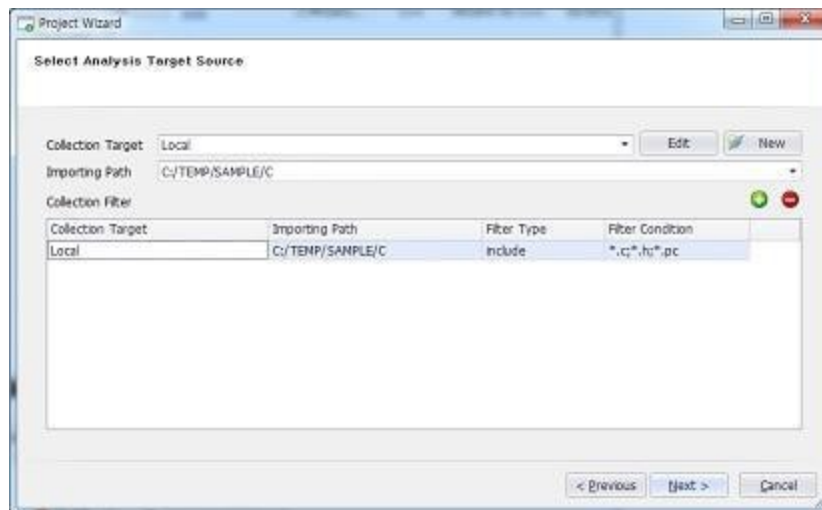
3. Select **Next**.
4. On the **Select Ruleset** page, select one of the available rule sets.

**NOTE:** A rule set is a group of rules. The available rule sets are determined by the license.



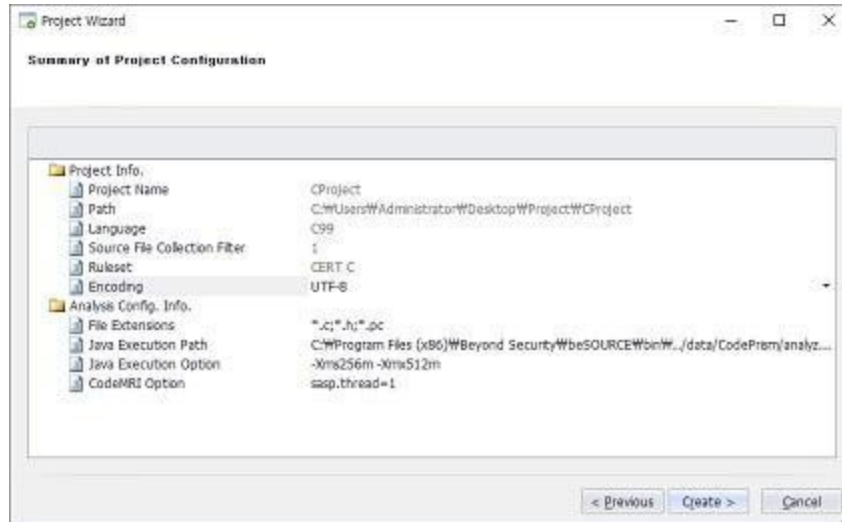
5. Select **Next**.
6. On the **Select Analysis Target Source** page, do the following:

- a. In the **Importing Path** box, select the root directory that includes all the .c and .h files.
- b. (Optional) To import source files from FTP, SVN, CVS, GIT, or TFS select **New** and specify the corresponding connection information. Then, specify the root directory that includes all the .c and .h files.
- c. Select the **Add** icon (  ) to insert a collection filter.
  - i. If the filter type is **Include**, all files that match a filter condition are imported. If the filter type is **Exclude**, all files that match a filter condition are not imported. The wizard automatically fills out the filter condition with default values, and you can add or remove some file extensions. For specifying multiple file extensions, use a semicolon (;) as the delimiter.

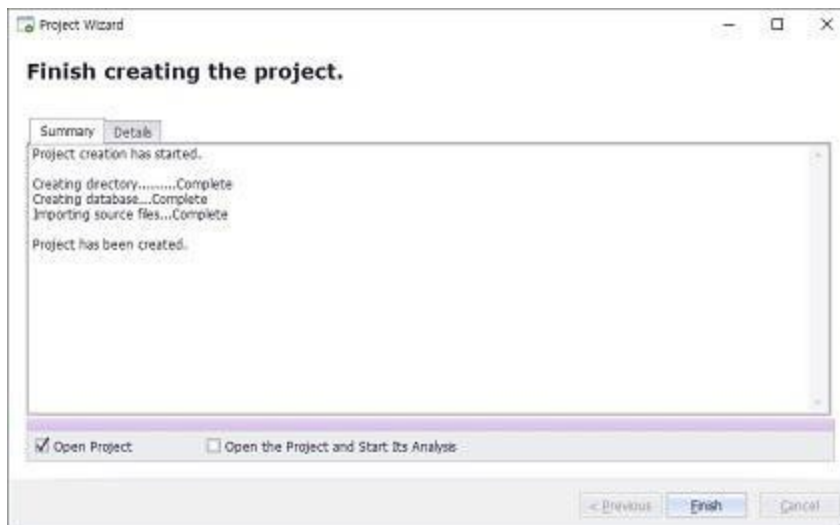


7. Select **Next**.
8. On the **Summary of Project Configuration** page, examine the default project options and, if necessary, specify these options:
  - a. **Encoding** - You must select the actual encoding value used in the source files.
  - b. **Target File Extension** - Adds or removes file extensions. The delimiter for multiple file extensions is a semicolon (;). If a file extension is not specified in this option, analysis ignores the file.
  - c. **Java Execution Option** - Specifies the JDK option for internal analysis engine. On a PC that can support it, you can increase JVM memory. Example: –

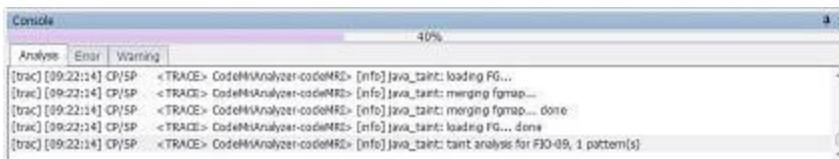
Xms256m -Xmx1024m.



9. Select **Create**.
10. On the **Finish creating the project** page, select **Open the Project and Start the Analysis**, and then select **Finish**. You can check the analysis progress in the Console window.



11. You can check the analysis progress in the Console window.

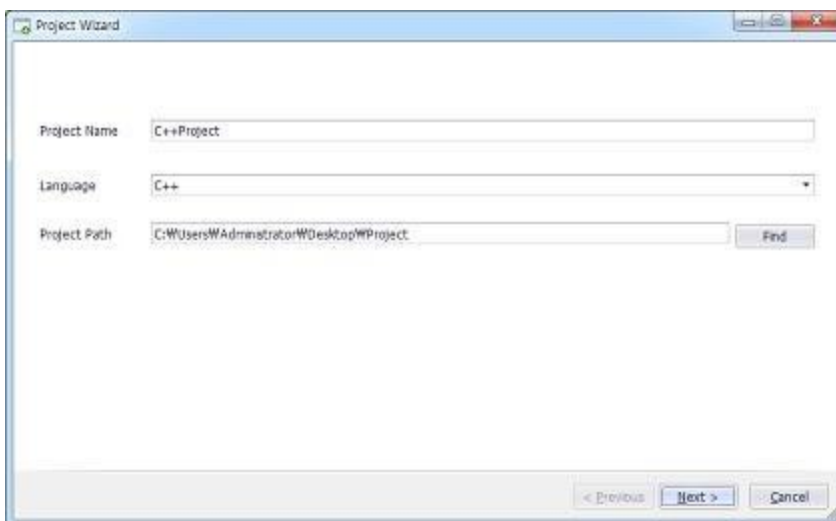


12. The List of Rule Violations window shows the inspection results after the analysis finishes.

- a. If you select a rule name, you can see the rule's description, examples of good and bad code, and international standard identifier of the rule in the Rule Description window.
- b. If you double-click a line number in the column number of the defect, the Source Viewer displays the actual source code.
- c. You can also search specific defects or rule violations by using the Advanced Search for Violations window.

## How to analyze C++ source files

The steps for analyzing C++ source files are the same as for analyzing C source files. However, there are differences in the setting of the language parser and the rule set in the Project Wizard. For example, you must specify C++ as the language, as the figure below illustrates. You must also specify a rule set that applies to C++.



For the remaining steps to analyzing C++ source code, use the steps in the preceding section, *How to Analyze C Source Files*.

**NOTE:** If you see parsing errors due to missing header files, refer to *Setting Referencing Header of C/C++*.

# Quick Start: Analyzing Java Source Files

## Overview

This section describes how to analyze Java source files.

## Preparation

beSOURCE requires all of the Java source files and libraries for accurate Java analysis. Typically, Java libraries are included in a .jar file. Therefore, you must ensure that all .jar files that are used by the targeted source files are imported through the Project Wizard.

Java sample source files are provided. You can find them in the `beSOURCE_Server_Installation_Folder\Sample\Java` folder.

Copy the source files to your PC.

## How to analyze Java source files

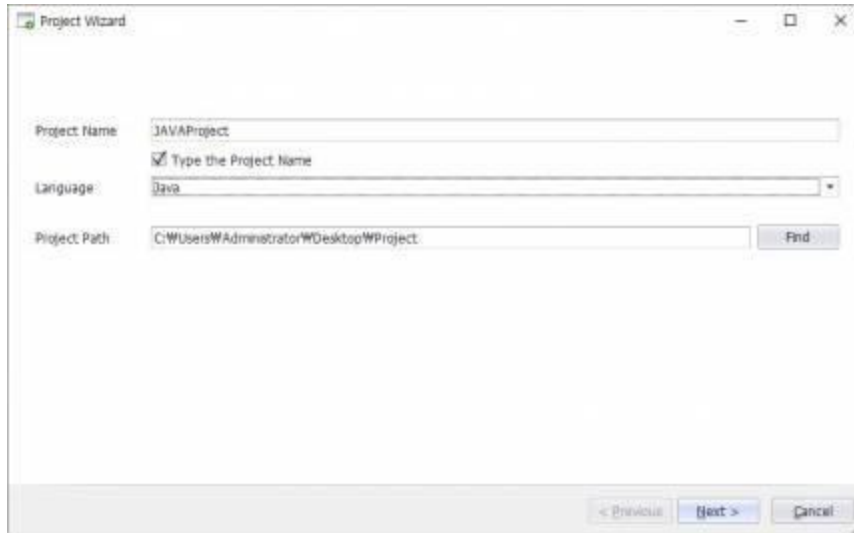
To create a new project for analyzing Java the source files and then immediately start analysis:

1. On the **Start Page**, select **New Project**. The **Project Wizard** opens.
2. On the first page of the wizard, do the following:
  1. In the **Project Name** box, enter a name for the project.
  2. In the **Language box**, select **Java**.

**NOTE:** For the Java language, you can select one of two parsers: **Java** and **Java Web**. **Java** only analyzes java and jar files. **Java Web** analyzes multiple language types at once, such as Java, JSP, JS, and XML.

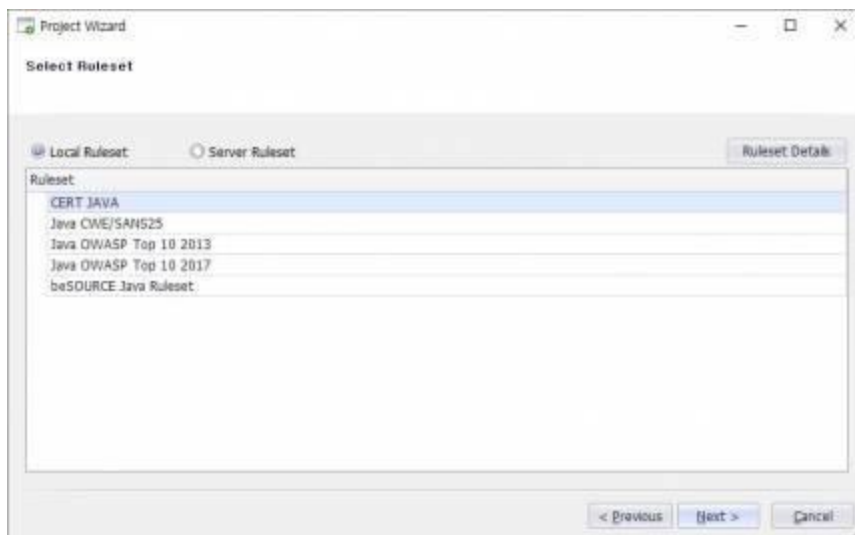
3. In the **Project Path** box, use the default location, or select **Find** to specify a

location.




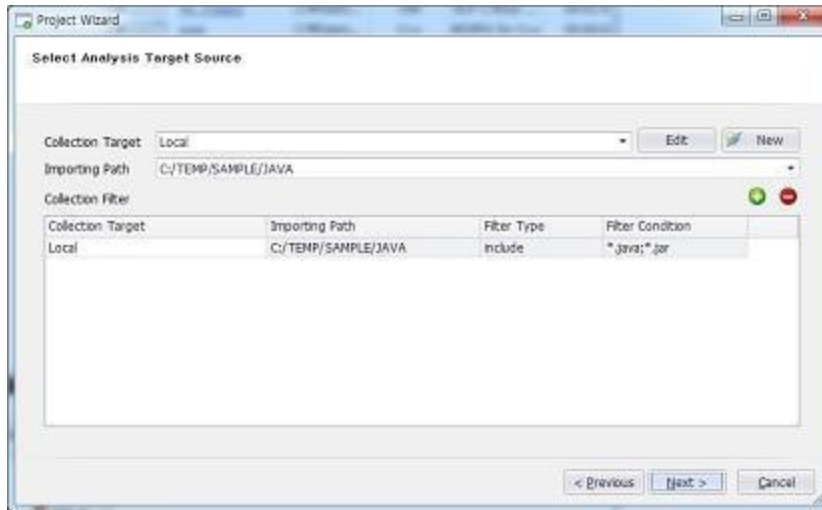
3. Select **Next**.
4. On the **Select Ruleset** page, select one of the available rule sets.

**NOTE:** A rule set is a group of rules. The available rule sets are determined by your license.



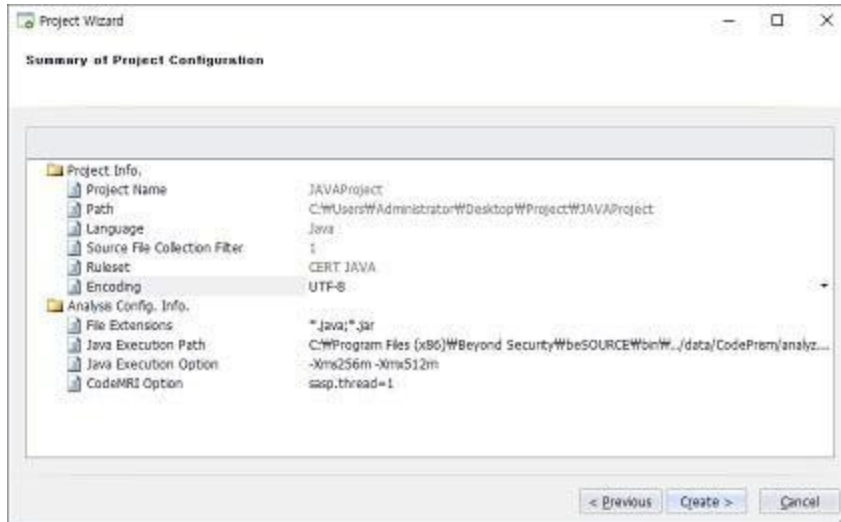
5. Select **Next**.
6. On the **Select Analysis Target Source** page, do the following:
  - a. In the **Importing Path** box, select the root directory that includes all the .c and .h files.
  - b. (Optional) To import source files from FTP, SVN, CVS, GIT, or TFS select **New** and specify the corresponding connection information. Then, specify the root directory that includes all the .c and .h files.

- c. Select the **Add** icon (  ) to insert a collection filter.
  - i. If the filter type is **Include**, all files that match a filter condition are imported. If the filter type is **Exclude**, all files that match a filter condition are not imported. The wizard automatically fills out the filter condition with default values, and you can add or remove some file extensions. For specifying multiple file extensions, use a semicolon (;) as the delimiter.

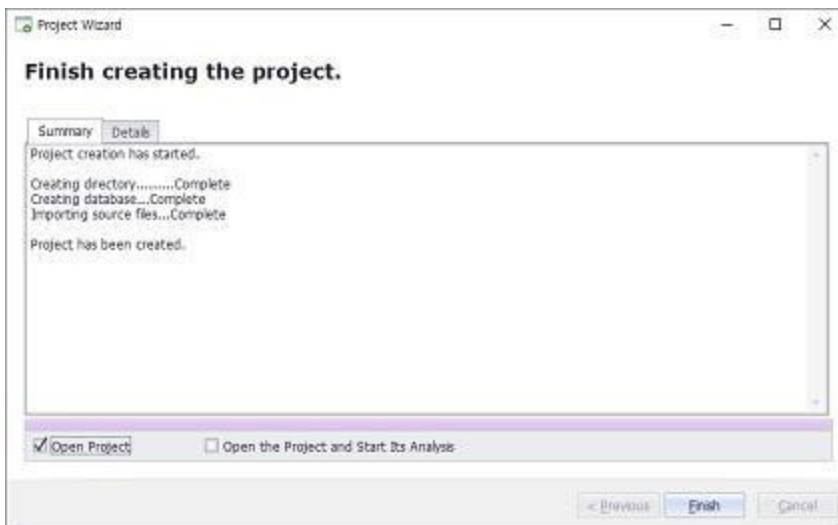


7. Select **Next**.
8. On the **Summary of Project Configuration** page, examine the default project options and, if necessary, specify these options:
  - a. **Encoding** - You must select the actual encoding value used in the source files.
  - b. **Target File Extension** - Adds or removes file extensions. The delimiter for multiple file extensions is a semicolon (;). If a file extension is not specified in this option, analysis ignores the file.
  - c. **Java Execution Option** - Specifies the JDK option for internal analysis engine. On a PC that can support it, you can increase JVM memory. Example: –

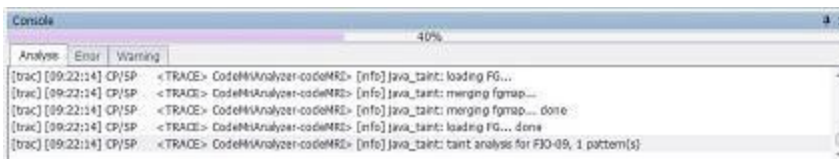
Xms256m -Xmx1024m.



9. Select **Create**.
10. On the **Finish creating the project** page, select **Open the Project and Start the Analysis**, and then select **Finish**. You can check the analysis progress in the Console window.



11. You can check the analysis progress in the Console window.



12. The List of Rule Violations window shows the inspection results after the analysis finishes.



- a. If you select a rule name, you can see the rule's description, examples of good and bad code, and international standard identifier of the rule in the Rule Description window.
- b. If you double-click a line number in the column number of the defect, the Source Viewer displays the actual source code.
- c. You can also search specific defects or rule violations by using the Advanced Search for Violations window.

# Command-line Interface

The CLI Analyzer is a utility tool to support simple source code analysis from command-line window or plug-in of editor tools such as Ultra Edit etc. It is recommended to use it for simple checking of source code on whether it has rule violations rather than for gaining full inspection information.

## Execution environment

Requires Java virtual machine installation. The Java VM must be version 1.6 or higher.

## Command format

```
java -DCLI_HOME=[cli home directory] -Xmx1024m -jar [CLI_
HOME]/lib/com.codeprism.cli.jar [COMMAND...]
```

java # java execution file

-DCLI\_HOME=[cli home directory] -Xmx1024m # Virtual machine option

-jar [CLI\_HOME]/lib/com.codeprism.cli.jar # Java program execution option

## Execution command

Sets environment variable and memory option for Java VM.

-DCLI\_HOME=[cli home directory] # CLI program's home directory path

-Xmx1024m # memory usage

## Virtual machine option

Enters actually executing program and user input commands.

```
-jar [CLI_HOME]/lib/com.codeprism.cli.jar [COMMAND...] # Input
com.codeprism.cli.jar file of CLI_HOME's lib directory.
```

# Refer to below for [COMMAND...] parameters.

# Job options

Job list available in command line mode

- cmd # run analysis
- stop # stop current analysis job
- update # get and update module and rules from server

## Main job options

Detail options list for each job

### Run analysis

- prj\_path=[project file path] # assigns project file location
- output=[result file path] # sets file location to save inspection or analysis result
- editor=[result open program] # sets to see inspection result in editor (Window only)

### Stop analysis

None

### Update rule & library

- server=[cm server url] # assigns server URL

## Option for each job

### Run analysis

```
java -DCLI_HOME=c:\cli -Xmx512m -jar  
c:\cli\lib\com.codeprism.cli.jar -cmd -prj-  
path=c:\workspace\sample\project.prj -  
outout=c:\workspace\sample\out.txt -editor=EditPlus
```

### Stop analysis

```
java -DCLI_HOME=c:\cli -Xmx512m -jar
c:\cli\lib\com.codeprism.cli.jar -stop
```

## Update

```
java -DCLI_HOME=c:\cli -Xmx512m -jar
c:\cli\lib\com.codeprism.cli.jar -update -
server=192.168.0.100:50102
```

## Creating a project file

You may configure a project before analyzing source code.

You need to create a project file to indicate project location, location of target source files, programming language, applying rules and other information.

The extension of the project file is .prj and XML format. Its encoding must be UTF-8.

## Project file

```
<?xml version="1.0" encoding="utf-8"?>
<ProjectTemplate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- Project Location -->
  <!-- Input actual project directory path -->
  <Path>[PROJECT_PATH]</Path>
  <!-- Parser Type -->
  <!-- Input parser type for programming language. -->
  <LangCode>3112</LangCode>
  <LangName>Java</LangName>
  <!-- Input encoding type of target source files. -->
  <Encoding>euc-kr</Encoding>
  <!-- Applying Rule -->
  <RuleSetID>3023112</RuleSetID>
  <RuleSetName>CWE/SANS Top 25 Ruleset</RuleSetName>
  <!-- Engine Option -->
  <CodeMRIOptions>sasp.thread=4</CodeMRIOptions>
  <!-- Internal Options -->
```

```

<!-- You may not need to modify them. -->
<LogPath>.log</LogPath>
<FilterPath>.filter</FilterPath>
<DataPath>.data</DataPath>
<DataFileName>data.db</DataFileName>
<DataBackup>true</DataBackup>
<SrcPath>src</SrcPath>
<LibPath>lib</LibPath>
<!-- C, C++ Header File Information -->
<BuiltInHeaders>
<Path />
</BuiltInHeaders>
<SystemHeaders>
<Path />
</SystemHeaders>
<!-- Importing Source Files -->
<Collectors>
<!-- Assigns it for each importing source files directory. -->
<!-- Unique ID is required for each collector. 'Type=2001' means that it will import source
files from local disk. -->
<Collector id="1" type="2001" path="[source path]">
<Filters>
<!-- You can import specific file types. 1001 = importing. 1002 = Not importing. -->
<Filter code="1001" value="*.java;*.jar" />
</Filters>
</Collector>
<Collector id="2" type="2001" path="[source or library path]">
<Filters>
<Filter code="1001" value="*.jar" />
</Filters>
</Collector>
</Collectors>
</ProjectTemplate>

```