



Coverity 2020.12 for Eclipse, Wind River Workbench, and QNX Momentics: User Guide

Copyright 2020 Synopsys, Inc. All rights reserved worldwide.

Table of Contents

1. Coverity Desktop overview	1
1.1. About this guide	1
1.2. Requirements	2
1.3. Coverity Desktop concepts and use cases	3
2. Getting Started	9
2.1. Retrieving remote issues	9
2.2. Running a local analysis	9
3. Analysis options and views	12
3.1. Toolbar shortcuts	12
3.2. Coverity menu	12
3.3. Issues view	15
3.4. Details view	24
3.5. Test Advisor annotations	28
3.6. Collect diagnostics	29
4. Analysis Configurations Dialog	31
4.1. General configuration	31
4.2. Coverity Connect	33
4.3. Streams	35
4.4. SCM	36
4.5. Advanced configuration	37
4.6. Dynamic variables	45
5. Troubleshooting Coverity Desktop for Eclipse	47
6. Finding more information about issues	53

Chapter 1. Coverity Desktop overview

Table of Contents

1.1. About this guide	1
1.2. Requirements	2
1.3. Coverity Desktop concepts and use cases	3

Coverity Desktop is a supplemental Coverity Analysis tool that allows you to identify issues in your source code within the Eclipse, Wind River Workbench, or QNX Momentics IDEs. Analysis results from your code base are displayed in the Coverity Desktop plug-in to help you find and report hard-to-spot software defects, potential security vulnerabilities, and test policy violations. Note that Dynamic Analysis defects are not displayed in Coverity Desktop.

Coverity Desktop provides two analysis modes:

- Central analysis
- Local analysis

After Coverity Desktop displays your issue data, you can view the issues for your project marked directly in the IDE's code editor view. Coverity Desktop provides two views: Issues and Details. Use these views to navigate through your issues, to determine the impact of the issues on your code, and to manage and update (*triage*) the state of each issue.

Installation instructions and supported platform details are located in the *Coverity 2020.12 Installation and Deployment Guide* [🔗](#).

1.1. About this guide

This guide provides the following information for Coverity Desktop for Eclipse, WorkBench, and QNX:

- Key concepts and use cases that describe typical end-to-end workflows.
- Coverity Desktop Getting Started guide to walk you through the configuration wizard and running your first analysis.
- Coverity Desktop user interface reference that details configuration settings, Issues view, and Details view.
- Coverity checker reference that describes all the Java (Eclipse) and C/C++ (Eclipse, QNX, and WorkBench) checkers and their options.

It is assumed that you have a functional knowledge of Eclipse, WorkBench IDE, or QNX Momentics, as well as Coverity Analysis, Coverity Connect, and Test Advisor concepts.

1.1.1. More information

If you need more information, consult the following Coverity product documentation:

For more information about Coverity Connect, see the *Coverity Platform 2020.12 User and Administrator Guide*.

For more information about Coverity Analysis, see the *Coverity Analysis 2020.12 User and Administrator Guide*.

For more information about Test Advisor, see the *Test Advisor 2020.12 User and Administrator Guide*.

1.2. Requirements



Note

Refer to the "Coverity Desktop" chapter of the *Coverity 2020.12 Installation and Deployment Guide* for supported IDE and Java version numbers.

Coverity Desktop for Eclipse has the following requirements:

- Coverity Connect 2020.12
- Coverity Analysis 2020.12 (required for local analysis. Typically installed at the same time as the plugin)
- Java Runtime Environment
- Eclipse or ARM Development Studio 5 (DS-5)
- Eclipse C/C++ Development Tooling (CDT) for the C/C++ analysis only (<http://www.eclipse.org/cdt/>)
- Eclipse Java Development Tools (JDT), only for the Java analysis
- Memory requirements: the minimum is 3 GB of RAM and more if you use parallel analysis. Refer to the *Coverity 2020.12 Installation and Deployment Guide* [🔗](#) for more information.

Coverity Desktop for Wind River Workbench has the following requirements:

- Coverity Connect 2020.12
- Coverity Analysis 2020.12 (required for local analysis. Typically installed at the same time as the plugin)
- Java Runtime Environment
- Wind River WorkBench

Coverity Desktop for QNX Momentics IDE has the following requirements:

- Coverity Connect 2020.12
- Coverity Analysis 2020.12 (required for local analysis. Typically installed at the same time as the plugin)

- Java Runtime Environment
- QNX Momentics

Before getting started with Coverity Desktop, make sure that you also have the following Coverity Connect access information:

- Host name
- Port number and type (HTTP or HTTPS)
- Authentication key file, or your user name and password for creating a new authentication key.
- Stream name

 **Note**

This information is dependent on your Coverity Connect administrator configuring the server and running an initial analysis and commit. See the *Coverity Platform 2020.12 User and Administrator Guide* for information on configuring Coverity Connect for use with Desktop Analysis.

1.3. Coverity Desktop concepts and use cases

This section provides conceptual definitions and use cases for Coverity Desktop. The point of this chapter is to help you decide how you would like to configure and run the Coverity Desktop tools to accomplish your tasks. After you decide on an analysis mode, see the subsequent chapters in this guide for information about server and analysis configuration, running a local analysis, and examining and triaging issues within the IDE.

1.3.1. Concepts

Central Analysis. A central analysis occurs when the central source code repository is analyzed by Coverity Analysis outside of the IDE. Central analyses are typically run on a build server and are triggered by an automated process. In most installations, a central analysis will be run as part of a nightly build. Alternatively, a central analysis might be initiated whenever code is checked in to the source code repository.

The issues discovered by the central analysis are committed to a Coverity Connect database that Coverity Desktop uses to retrieve and update issue information that you view and triage within the IDE. With this workflow, you are limited to the enabled checkers and the checker settings that are defined for the central Coverity Analysis configuration. If you want to use different checkers, you must run your own local analysis and then select the checkers you want to use.

Local Analysis. With local analysis, you analyze your source code for issues within your IDE. By running a local analysis, you can examine issues, fix issues, and verify that the issues are fixed before checking your code back into the central code repository. You have the option of analyzing a single source file, a set of source files of your choosing, or your entire project.

Local analysis options are set using one or more Analysis Configurations. Upon initial set up Coverity Desktop, you will have a default analysis configuration with your choice of analysis options. You

may create additional analysis configurations, with different sets of options, and choose the correct configuration for each individual analysis run. This allows you to easily change the type of analysis you want to run, depending on the project or file(s) you are currently working on.

Before you begin using Coverity Desktop, there are a few important points to keep in mind. Local analyses will run on the files within your workspace. Whether you have run a local analysis or you are viewing remote issues from a central analysis, the plug-in will present a list of issues in the Issues view, and will also show issue markers alongside the defect occurrences in the source code editor. To help the analysis run smoothly and report accurate information:

- Check the console view for output while running an analysis. Some non-critical warnings (such as number of missing classes) appear only in the console.
- There may be files added to your workspace since your last build that are not available for analysis. In these cases, when you select an analysis option to run, Coverity Desktop will prompt you with several options in the *Uncaptured Source Files* dialog. Select *Capture Build and Analyze* to complete the analysis.

You may also preemptively capture a full build of your project, prior to attempting analysis, by using the *Capture Build of Workspace* command in the Coverity menu.



Note

While Coverity Desktop displays issues reported by Test Advisor through Central Analysis, the plug-in does not support local Test Analysis. This means that you will be able to view and triage test policy violations retrieved from the Coverity Connect server, but not analyze local changes for newly introduced test issues.

Also note that central analysis results may contain issues for any Coverity supported language, but Coverity Desktop for Eclipse only supports local analysis for Java and C/C++.

Analysis Configurations. Analysis Configurations are entities that define the analysis scope and set the options for local and central analyses. You must have at least one analysis configuration in order to use the plug-in, and you may also set up additional configurations for using different options and analysis scopes.

Each Analysis Configuration can be associated with a reference stream and snapshot from a central analysis on the Coverity Connect server. It is possible, and recommended, to use the analysis settings that are associated with the reference stream. However, you may also choose to pass additional analysis options, or exclude certain files from analysis through your analysis configuration.

Web application security analysis. Coverity Desktop can perform local security analysis on Java Enterprise Edition (Java EE), servlet-based web applications. This analysis runs the Java security checkers, including XSS and SQLI.

Java Web Application Security analysis requires running a full analysis on the entire scope. The selected *Analysis Configuration* must have the *whole program* option enabled on the Advanced → Analysis configuration page. This *whole program* option is added when selecting the following checkbox on the *Analysis* configuration page: *When running Analyze Entire Scope, run all enabled whole program checkers*.

1.3.2. Use cases

The use cases described in this section provide a high-level workflow of how you can use Coverity Desktop within your organization. The use cases are:

- Central analysis
- Local analysis
- Central and local analysis

All of the use cases below use the following user roles/personas to demonstrate a particular workflow:

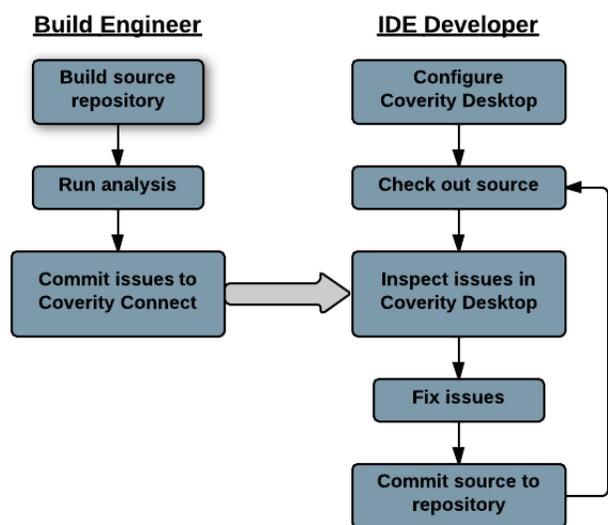
- `Build Engineer` is responsible for the central code repository, including using Coverity tools for Analysis Configuration, analysis runs, and committing the results to a central Coverity Connect server.
- `IDE Developer` is a member of a software development team that uses an Eclipse based IDE for development and Coverity Desktop to view and triage central analysis and/or local analysis issue results.

1.3.2.1. Use case - Central analysis

Goal. IDE Developer wants to view and triage analysis results from the central source code repository.

The following diagram shows a high-level view of the process:

Figure 1.1. Central analysis model



1. Build Engineer builds and runs Coverity Analysis on the central code base. (This process is typically automated on some regular interval, such as nightly).
2. Build Engineer (or automated scripts) commits the issue results to Coverity Connect. (Projects and streams are already appropriately configured to receive issue data).

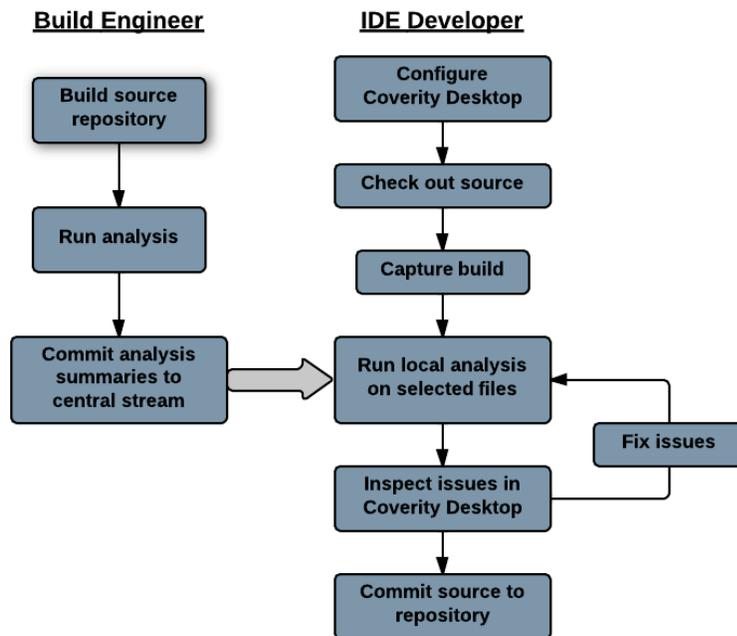
3. IDE Developer configures Coverity Desktop to connect to the central Coverity Connect server.
4. IDE Developer checks out a section of the code for which he/she is responsible.
5. IDE Developer examines the impact of the remote issues in the code and triages them within the IDE.
6. IDE Developer fixes issues and checks the code into the central repository.
7. The nightly analysis runs.

1.3.2.2. Use case - Local analysis

Goal. IDE Developer wants to analyze and view defect results on his/her local code.

The following diagram shows a high-level view of the process:

Figure 1.2. Local analysis model



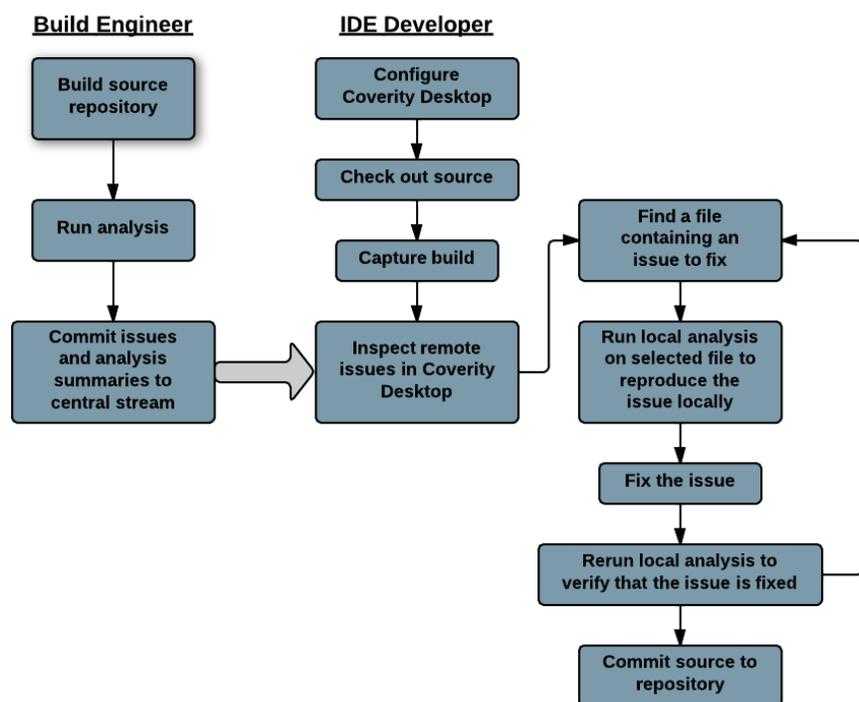
1. Build Engineer builds and runs Coverity Analysis on the central code base. (This process is typically automated on some regular interval, such as nightly).
2. Build Engineer (or automated scripts) commits the issue results, along with function summary data, to Coverity Connect. (Projects and streams are already appropriately configured to receive issue data).
3. IDE Developer configures Coverity Desktop to connect to the central Coverity Connect server and configures local analysis settings.
4. IDE Developer checks out a section of the code for which he/she is responsible.

5. IDE Developer runs local analysis on selected file(s).
6. IDE Developer examines and triages the issues that were found by the analysis.
7. IDE Developer fixes a list of issues.
8. IDE Developer runs local analysis again to ensure that the issues were fixed.
9. IDE Developer checks the code into the central repository.

1.3.2.3. Use Case - Using Coverity Desktop with central and local analysis

Goal. IDE Developer wants to see remote issues and also wants to use local analysis before committing code to the source repository. This ensures that the IDE Developer doesn't introduce any new issues into the source repository as a result of their changes.

Figure 1.3. Central and local analysis model



1. Build Engineer builds and runs Coverity Analysis on the central code base. (This process is typically automated on some regular interval, such as nightly).
2. Build Engineer (or automated scripts) commits the issue results, along with function summary data, to Coverity Connect. (Projects and streams are already appropriately configured to receive issue data).
3. IDE Developer configures Coverity Desktop to connect to the central Coverity Connect server and configures local analysis settings.

4. IDE Developer checks out a section of the code for which he/she is responsible.
5. IDE Developer retrieves remote issues from the Coverity Connect server, and finds the file containing the defect to be fixed.
6. IDE Developer runs local analysis on the file to reproduce the defect locally.
7. IDE Developer fixes the defect in question.
8. IDE Developer runs local analysis again to verify that the defect is fixed.
9. IDE developer continues working on issues in individual files, or checks the completed code into the central repository.

Chapter 2. Getting Started

Table of Contents

2.1. Retrieving remote issues	9
2.2. Running a local analysis	9

This chapter provides you with the basic steps for retrieving remote issues and running local analyses. If you have not set up an initial analysis configuration, you will be prompted for pertinent configuration information as you perform these tasks for the first time. You can find more detailed information on the various Coverity Desktop views and features in their respective sections of Chapter 3, *Analysis options and views*.

Installation instructions for Coverity Desktop are located in the *Coverity 2020.12 Installation and Deployment Guide* [↗](#).

2.1. Retrieving remote issues

Coverity Desktop allows you to view remote issues from central analyses on your Coverity Connect server. This section will guide you through the process of viewing remote issues in the Coverity Desktop plug-in. This section assumes that you have already set up an analysis configuration with a Coverity Connect connection and associated stream. See Chapter 4, *Analysis Configurations Dialog* for details.

Note

You will only be able to view issues from source code files to which you have access. These files should also be present in your IDE's workspace. Otherwise, the file name associated with the issue will display in red in the *Issues* view, and the issue's source code will not be available for display in the editor.

1. Navigate to Coverity → View Issues from Coverity Server.

This will open the *Issues* view in *Remote Issues* mode.

2. Double-click on any of the issues to display the issue details and open the *Details* view.
3. From here, you can triage and fix the issue directly within the IDE. See Section 3.3, “Issues view” for details on the Issues view filters and options. See Section 3.4, “Details view” for information about the tabs and fields in the Details view.

2.2. Running a local analysis

Local analysis allows you to locate new and existing issues for code that you have changed before checking it into your central code repository. This short tutorial will guide you through the necessary steps.

In order for local analysis to return the most accurate results, you must connect to a relevant stream on the Coverity Connect server. This will provide necessary interprocedural analysis summary information

for fast, accurate analysis results. This will also allow issue triage information to be shared between Coverity Desktop and Coverity Connect. See Section 4.3, “Streams” for additional information.

 **Note**

This tutorial assumes that you have a Java or C/C++ project open within the IDE.

1. Ensure that the source file you want to analyze is open in the main *Editor* pane.
2. Select *Analyze Current Editor File* from the Coverity menu, or simply click the shortcut icon () on the toolbar. This will save the current editor file if it has any unsaved changes.

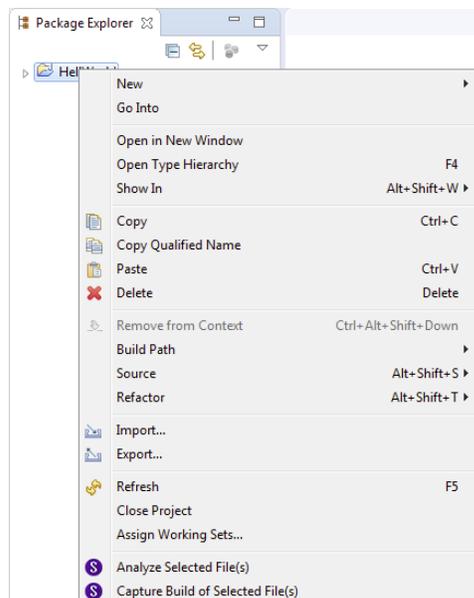
 **Note**

You can also analyze multiple files in the same local analysis. To do so, use the *Analyze Modified Files (SCM)* option, via the Coverity menu or the toolbar shortcut. If you have configured your SCM for use with Coverity Desktop, this will analyze all source files that are new or modified since your latest checkout.

If you have not configured your SCM, you can simply select multiple files manually:

1. Highlight each of the files, packages, and/or projects you wish to analyze in the package explorer.
2. Right-click the selection to open the context menu.
3. Click *Analyze Selected File(s)*.

Figure 2.1. Context menu



Note that any source files that have not been previously captured in a build by Coverity Desktop will not be available for analysis. When you attempt to run local analysis on one of these files, you will be prompted with several options in the *Uncaptured Source Files* dialog. To proceed with analysis, select *Capture Build and Analyze*. This will capture a build of any projects impacted by these files, and then run the local analysis as requested.

3. Once the analysis is complete, the *Issues* view will be updated to reflect any issues in the code you've selected. Double-click on an issue to view and triage the issue from within the IDE.

See Section 3.2, "Coverity menu" for additional analysis options.

Chapter 3. Analysis options and views

Table of Contents

3.1. Toolbar shortcuts	12
3.2. Coverity menu	12
3.3. Issues view	15
3.4. Details view	24
3.5. Test Advisor annotations	28
3.6. Collect diagnostics	29

Coverity Desktop allows you to perform a local analysis on your source code from your IDE, so you can quickly pinpoint and fix issues before checking your code into a central code base. This chapter provides details on the various analysis options and views available in the plug-in.

3.1. Toolbar shortcuts

Coverity Desktop adds several shortcuts to the Eclipse toolbar, allowing you to quickly select and/or edit analysis configurations, or run a local analysis. The shortcuts are:

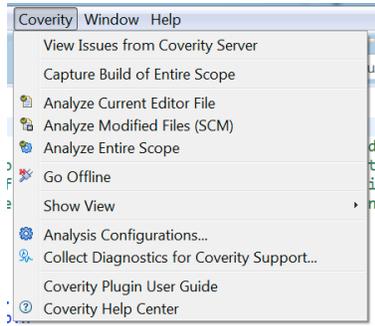
Table 3.1. Toolbar shortcuts

Shortcut	Action	Description
	Analyze current editor file	Analyze the current editor file, using the settings specified by the active analysis configuration.
	Analyze modified files	Query the configured SCM to see which files have been modified locally, then run local analysis on those files, using the settings specified by the active analysis configuration.
	Analyze entire scope	Run local analysis on all the files specified in the active analysis configuration's analysis scope, using the settings specified by the analysis configuration.
Drop-down menu	Active analysis configuration	Drop-down menu that allows you to select the active analysis configuration. You can also create a new configuration, or edit an existing configuration, from this menu.

3.2. Coverity menu

You can configure analysis options, retrieve remote issues, and run local analyses from the Coverity menu.

Figure 3.1. Coverity menu



This menu contains the following menu items:

- View Issues from Coverity Server
- Capture Build of Entire Scope
- Analyze [Current Editor File | Modified Files | Entire Scope]
- Go Offline
- Show View
- Analysis Configurations...
- Collect Diagnostics for Coverity Support
- Coverity Plugin User Guide
- Coverity Help Center

3.2.1. View Issues from Coverity Server

This item will switch the *Issues* view to *Remote Issues* mode, allowing the user to view and triage issues from Coverity Connect.

See Section 3.3, “Issues view” for additional information on *Remote Issues* mode.

3.2.2. Capture Build of Entire Scope

This option will attempt to build all of the source files in your current Analysis Configuration's scope, and will capture the build under Coverity monitoring. Build settings can be configured in the Analysis Configurations dialog.

Note that it is not necessary for you to capture the build up-front, as you will be prompted to capture any unbuilt files on demand when running local analyses.

3.2.3. Analysis scope options

There are three ways that Coverity Desktop chooses which files to analyze with Desktop Analysis:

Analyze Current Editor File

This option will run Desktop Analysis on the currently displayed source file. If there are any unsaved changes, the file will be saved when this option is selected.

Analyze Modified Files (<SCM>)

This option will run Desktop Analysis on all of the source files that are new or modified, relative to the code version most recently checked out from the specified <SCM>.

This option is not available if you have not configured an SCM system. See Figure 4.5, “SCM tab”.

Analyze Entire Scope

This option will run Desktop Analysis on all of the source files in your active Analysis Configuration's scope.

3.2.4. Offline Mode

This option will run Coverity Desktop in a completely unconnected state. When in Offline mode, Coverity Desktop will not be able to download summary or triage data from Coverity Connect, and instead relies on any cached data from previous local analyses. As a result, analysis summaries may be out of date or nonexistent, and the results of local analyses could be less accurate.



Note

Because local analysis requires summary data from Coverity Connect, Offline mode requires that the Coverity Desktop plug-in was previously connected to a Coverity Connect server, with a reference stream configured.

Be sure that you are still able to connect with Coverity Connect when selecting *Go Offline* to allow the summaries to be downloaded in full. This will give the best results when doing offline analysis.

3.2.5. Show View

This sub-menu allows you to open or close the following Coverity Desktop views:

- Issues
- Details

3.2.6. Analysis Configurations...

This option takes you to the Analysis Configurations Dialog, which contains important options and information for central and local analysis.

3.2.7. Coverity Desktop Plug-in User Guide and Help Center

This option open the Coverity Desktop user guide.

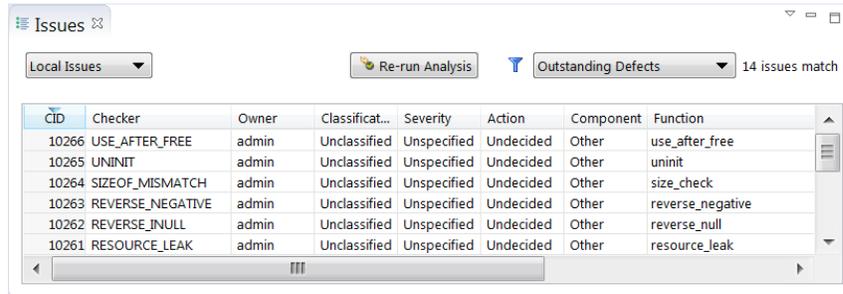
3.2.8. Coverity Desktop Plug-in User Guide and Help Center

This option open the full Coverity documentation set, which provides additional information about related Coverity components.

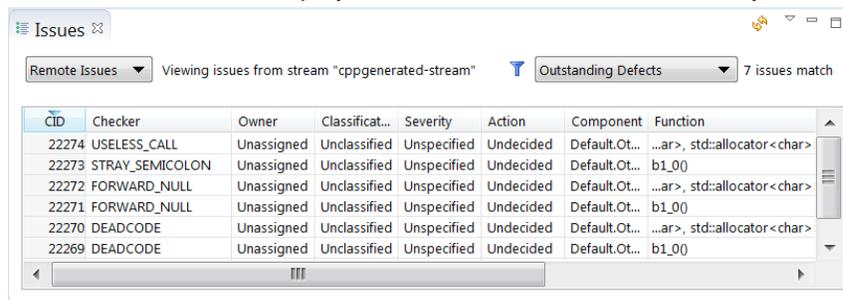
3.3. Issues view

The *Issues* view displays in one of two modes:

- *Local Issues* mode displays the results of a local analysis.



- *Remote Issues* mode displays issues retrieved from the Coverity Connect server.

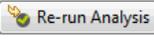


If the *Issues* view is not open, select **Coverity** → **Show View** → **Issues**.

After running a local analysis or retrieving remote issues, the *Issues* view displays a list of the returned issues. It is possible that the list may be too large for you to effectively manage, or you might want to organize your issues in a manner that makes sense for your particular task. For example, you might want to list only the critical and open issues that are assigned to you. You can use filtering and sorting in the *Issues* view to focus the scope of your issues.

Each of the *Issues* view controls is described in Table 3.2, “Issues view buttons”.

Table 3.2. Issues view buttons

Button	Name	Action
	Filter	Opens the Issue Filters screen. The <i>edit filters</i> link, located within the <i>Filter</i> drop-down menu, also opens the Issue Filters screen. See Section 3.3.2, “Issue Filters”.
	View Menu	Allows you to sort and configure the display of the issue listing. See Section 3.3.1, “Configuring the Issues view display”.
	Re-run Analysis	Runs a local analysis using the same scope and options as the most recently completed analysis.

Button	Name	Action
		You can use this to verify that your changes have fixed existing issues, and have not introduced any new issues.

After you select a manageable number of issues to view, double-click on an issue to open it in the editor. When you open an issue, the source code file that contains the issue opens in the source editor to the location where the issue occurs, placing markers that correspond to relevant lines. Additionally, opening an issue loads the current triage information for the issue in the Details view.

To close the issue and remove the markers, click **Close** in the *Details* view.

3.3.1. Configuring the Issues view display

You can customize the organization of the issue reports by using the View menu () to sort your list of issues and customize what columns are displayed.

 **Note**

The attributes described in this section represent the fields that are available to you by default. It is possible that there are additional attributes and attribute values if custom attributes were created in Coverity Connect.

3.3.1.1. Sorting issues

Use the *Sort by* menu to sort the list of issues based on the following default categories:

CID

A numeric identifier for the essential characteristics of a defect that are unlikely to change from snapshot to snapshot. When two occurrences have the same CID, it is likely that both would be fixed by the same source code change. Triage is associated with the CID rather than any particular occurrence.

MergeKey

Internal signature used to merge separate occurrences of the same software issue and identify them all by the same CID.

Checker

The checker that reported the issue.

Owner

The user assigned to resolve the issue. You can change the owner that is assigned to the issue in the Details view.

Classification

Indicates the state of an issue. The available classifications are Unclassified, Pending, False Positive, Intentional, Bug, Untested, No Test Needed, and Tested Elsewhere. You can change the level assignment in the Details view.

Severity

Indicates an issue's magnitude of potential risk. The default severity levels are Unspecified, Major, Moderate, and Minor. You can change the severity level in the Details view.

Action

Indicates how an issue is to be handled. The default categories are Undecided, Fix Required, Fix Submitted, Modeling Required, and Ignore. You can change the action in the Details view.

Fix Target

The targeted time frame in which the issue should be fixed.

Ext. Reference

An identifier (such as an issue number in a different database) specified by your company.

Legacy

Displays *True* if the CID is a Legacy Issue, otherwise *False*.

Component

The name of the component in which the issue was discovered.

Function

The name of the function that contains the issue.

File

The file path that contains the issue. If the file is not located, the path displays in red text. If red text is displayed, go to the *File Path Mapping* screen, and make sure that you have the proper definitions for stripping the remote path prefix and for adding the proper local path to search (if needed). The displayed path is the path known to Coverity Connect, the tooltip will display the local file path if the file is found locally.

Occurrences

The number of defect occurrences that all have the same CID.

First Detected

The date of the analysis in which the issue was first detected.

Last Detected

The date of the analysis in which the issue was last detected.

Sortable in *Remote Issues* mode only.

Last Triaged

The date of the last analysis in which a change was made to the issue's triage data.

Sortable in *Remote Issues* mode only.

Impact

Issue impact as determined by Coverity Connect: High, Medium, Low, or Audit

Category

Short description of the nature of the software issue.

Type

Issue type. For example, *Resource leak*, *Out-of-bounds write*.

Present in Reference

Indicates whether a CID is present in your reference stream.

Sortable in *Local Issues* mode only.

Language

Programming language associated with the issue.

Custom attribute (text)

Appears if a custom attribute that accepts a text field exists in Coverity Connect. Sorting is done alphabetically.

Custom attribute (picklist)

Appears if a custom attribute with an ordered set of pick-list values exists in Coverity Connect. Sorting is according to the index of the pick-list element. That is, it is only ordered alphabetically if the list is ordered alphabetically in the Coverity Connect configuration.

3.3.2. Issue Filters

To focus the scope of your issues in order to produce a more manageable work list, you can construct and apply a filter to one of your analyzed projects.

To apply a filter, select one of the following from the *Filter* drop-down menu:

- A pre-defined filter.
- A customized filter that you have created and saved.

After you have selected a filter, the issue list is updated with the matching results.

3.3.2.1. Pre-defined filters

Pre-defined filters represent common search criteria to quickly focus your list of issues. You can add filter expressions, or edit the attributes and values in these filters to customize them. For information about editing pre-defined filters, see custom filters.

Coverity Desktop provides the following pre-defined filters:

Table 3.3. Built-in filter fields

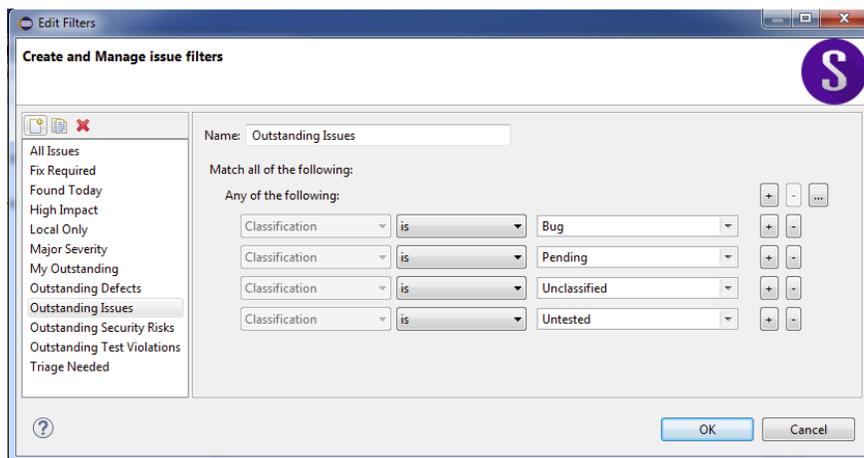
Column	Description
All Issues	Displays all issues present in the current analysis.
All Outstanding Issues	Displays all outstanding issues present in the current analysis.
Fix Required	Displays all issues that have action - fix required.
Found Today	Displays all issues that were found by the analysis for the current day.
High Impact	Displays all issues that have an impact rating of High.
Local Only	Displays only local issues that are not present in the reference stream.
Major Severity	Displays all issues with a severity of Major.
Triage Needed	Displays all issues that have are unclassified.

Column	Description
My Outstanding*	Displays all outstanding issues that are assigned to the current user.
Outstanding Defects	Displays all outstanding quality defects.
Outstanding Issues	Displays all outstanding quality defects, security vulnerabilities, and test violations.
Outstanding Security Risks	Displays all security vulnerabilities found by Security Analysis.
Outstanding Test Violations	Displays all Test Advisor policy violations.

*. To modify the owner of the My Outstanding issues filter, manually set the Owner field to the user name of the person you want to own the issues listed as My Outstanding.

3.3.2.2. Custom filters

Customized filters allow you to configure one or more filters to restrict the number of issues you wish to be displayed in the Issues view. To construct your filter settings, click the **Filter** button (). This launches the Edit Filters window:



To create a new filter, click the **Add** button () and enter the name of the filter in the *Name* field. This is the name that will display in the *Filter* list in the Issues view. If there is a pre-defined or an existing saved filter that you want to be the basis for a new filter, highlight it and select the **Copy** button () to edit the filter's configuration.

Filters are made up of one or more filter expressions. A Filter expression consists of the following:

Attribute

Describes an aspect of the analysis results for the issue. For example, you can specify user-defined triage states, a checker or issue type, and so forth. The attributes that you can define for a filter are described in Table 3.5, "Filter attributes and values".

 **Note**

The attribute and value table does not include any custom attributes that might exist if they were created in Coverity Connect.

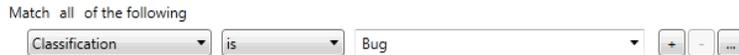
Qualifier

Defines the criteria for how the value, related to its attribute, appears in the search. Each attribute will have a unique list of possible qualifiers from which you can choose, but each qualifier is described in Table 3.6, "Filter qualifiers".

Value

Represents a specific aspect of an attribute. Attribute values are described in Table 3.5, "Filter attributes and values".

For example, the following filter expression, when saved and applied as a filter, will return only the issues that have a Classification of Bug:



Coverity Desktop allows you to construct more complex filters. The filter expression buttons allow you to add multiple filter expressions and nested filter expressions. The controls are:

Table 3.4. Filter expression buttons

Button	Action	Description
	Add	Adds a filter expression at the current level. There are two levels at which you can add a filter, the top level and the nested level. New filter expressions created at the top level are AND operators, so results are returned if they match all of filter expressions that you add at that level. Top-level filter expressions have to contain a unique filter value. The Add button also creates new nested expressions at the same level as the previous nested expression (if one exists).
	Remove	Removes the current filter expression. If you remove a top level expression that contains nested expressions, all of those expressions are deleted.
	Add nested	Adds a new nested expression one level under the current top level expression. Nested expressions are OR operators, so results are returned for the current top level expression if the results match the top-level and any of the nested levels. All nested level expressions that belong to the same top level expression must filter on the same attribute. You can change the qualifiers and values.

For example, the following filter expressions, when saved and applied as a filter, return any issues that match the following:

- Any Classification value that is Unclassified

AND

- Impact rating of High Impact

AND

- Any of the following Action values:

- Undecided

OR

- Fix Required

OR

- Modeling Required

Match all of the following

Classification	is	Unclassified	+	-	...
Impact	is	High	+	-	...

Any of the following

Action	is	Undecided	+	-	...
Action	is	Fix Required	+	-	...
Action	is	Modeling Required	+	-	...

The following table lists the default attributes and possible values that you can use in a given filter expression. Note that this table does not include custom attributes and values that can be created in Coverity Connect:

Table 3.5. Filter attributes and values

Attribute	Description	Values
Action	The action to be taken on the issue.	Undecided, Fix Required, Fix Submitted, Modeling Required, or Ignore. For definitions of the action values, see Section 3.4.5.3, “Action”.
Category	Represents a description of the type of issues that one or more checkers might find during analysis.	A checker category chosen from a pick list.
Checker	The name of the checker that reported the issue.	A checker name chosen from a pick list or a full or partial name of a checker entered by the user.
CID	The CID (unique numerical representation) of the issue.	A number or range of numbers.
Classification	The classification of the issue.	Unclassified, Pending, False Positive, Intentional, or Bug. Test Advisor specific classifications include Untested, No Test Needed, and Tested Elsewhere. For definitions of the classification values, see Section 3.4.5.1, “Classification”.

Analysis options and views

Attribute	Description	Values
Component	<p>Displays issues that are contained in one or more components to which you have access.</p> <p>Component filtering is CID-based, so an issue is included even if only one of its occurrences happens to be in the included component. Components that are invisible to users (through Access Control or exclusion) do not appear in the filter. For more information about components, see the <i>Coverity Platform 2020.12 User and Administrator Guide</i>.</p>	A component name chosen from a pick list (if there is more than one) or a full or partial name of a component entered by the user.
External Reference	An identifier (such as an issue number in a different database) specified by your company.	An external reference value chosen from a pick list or a full or partial external reference value entered by the user.
File	The name of the file that contains the issue.	A full or partial file name entered by the user.
First Detected	The date of the analysis in which the issue was first detected.	A date or a number of hours, days, weeks, or years.
Fix Target	The targeted time frame in which the issue should be fixed.	Untargeted, or any custom value set by an administrator.
Function	The name of the function that contains the issue.	A full or partial function name entered by the user.
Impact	Impact is a rating of how the issue will affect your code or program. Some issue types have a greater impact on software stability than others. Filtering by impact allows you to view high impact issues first.	High Impact, Medium Impact, Low, or Audit Impact.
Issue Kind	Specifies which Coverity product (Coverity, Security Advisor, Test Advisor) found the issue.	Quality, Security, Test Violation.

Analysis options and views

Attribute	Description	Values
Language	Programming language associated with the issue.	A Coverity supported programming language (C++, Java, etc.)
Legacy	Specifies whether the CID is a <i>Legacy</i> issue.	True or False.
MISRA Category	The classification for MISRA issues	Mandatory, Required, or Advisory (If this choice is left blank, it is set to None.)
Occurrences	The number of issues that have this CID.	A number or range of numbers.
Owner	The owner of the issue. Coverity Desktop gives you a drop-down list of all available users on the Coverity Connect instance to which the plug-in is connected. By default, the owner is the current user.	A Coverity Connect username chosen from a pick list or a full or partial username entered by the user.
Present in Reference	Indicates whether a CID is present in your reference stream.	True or False.
Severity	The severity assigned to the issue.	Unknown, Major, Moderate, or Minor. For definitions of the severity categories, see Section 3.4.5.2, "Severity".
Type	Issue type. For example, <i>Resource leak</i> , <i>Out-of-bounds write</i> .	An issue type chosen from a pick list.

The following table lists the qualifiers that are available to you when you construct your filter expressions. This table is a comprehensive list of the qualifiers. Attributes will only contain a certain subset of these qualifiers (based on the type of attribute):

Table 3.6. Filter qualifiers

Qualifier	Action
is	Includes the value for the attribute in the filter.
is not	Includes all other values (excluding the value that is specified) for the attribute in the filter.
contains	Include the value for the attribute if the value contains the characters entered in the text field.
matches glob	Include the value if it matches any part of the entered glob pattern. For example, if you choose the Checker attribute and add <code>FB.*NW*</code> , the filter will return all available checkers that contain "FB." and the characters "NW", such as <code>FB.UWF_UNWRITTEN_FIELD</code> or <code>FB.NP_UNWRITTEN_FIELD</code> .
starts with	Include the value for the attribute if the value starts with the exact characters entered in the text field.

Qualifier	Action
ends with	Include the value for the attribute if the value ends with the exact characters entered in the text field.
is in the range	Select a range of two dates or numbers.
is greater than	A range that is greater than the number you enter.
is less than	A range that is less than the number you enter.
is after	Select a single date. Returns values that occur after the specified date.
if before	Select a single date. Returns values that occur before the specified date.
is today	Returns the values that match the current date.
in the last	Returns the values that match the specified number of hours, days, weeks, or months.
not in the last	Returns the values that do not match the specified number of hours, days, weeks, or months.

3.3.3. Context Menu

When you right-click on an issue in the *Issues* view, the context menu is displayed. From this menu, you can use the following options:

Show Checker Help

Opens the documentation for the checker that found the issue.

Go to

Opens the issue in the *Details* view.

View in Coverity Connect

Displays the issue in Coverity Connect via your default browser. This option is not available when the issue is only present locally, or if multiple issues are selected.

Copy

Copies the currently selected row so you can paste it as a text string into an email message, bug report, and so forth.

Set <attribute>

Defines an attribute value for Classification, Severity, Action, and so forth. This is a quick way to triage your issue without having to use the Details view. You can select multiple issues and set the triage value for all of the selected issues at the same time. If a value is greyed out, that means it is an invalid value for the Issue Kind(s) of the selected issues.

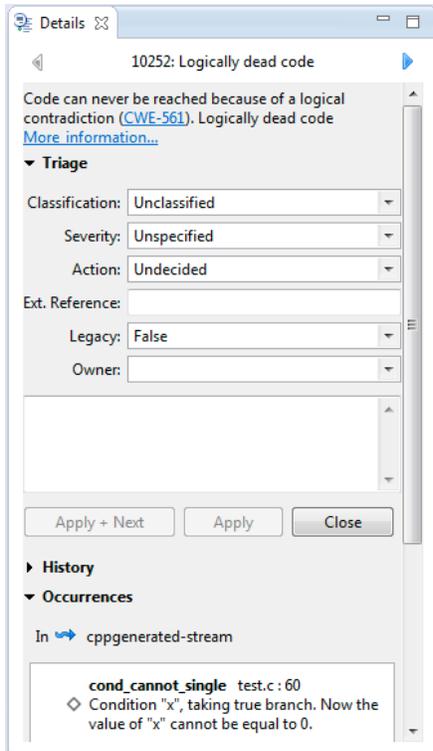
3.4. Details view

The *Details* view allows you to view and triage issue details. For example, you can assign an issue's classification, severity level, owner, or add a comment. The Details view is not populated until you double click an issue in the *Issues* view.

**Note**

The attributes described in this section represent the fields that are available to you by default. It is possible that there are additional attributes and values created as custom attributes in Coverity Connect.

Use the next/previous issue buttons at the top of the Details view to display the next or previous CID in your list.

**3.4.1. Issue information**

The top of the *Details* view describes the category of the issue discovered by the Coverity Analysis checker and provides a description of the issue. It also provides a link to the relevant SpotBugs documentation (if applicable) and cross reference to the industry standard Common Weakness Enumeration [CWE](#) (CWE). CWE is extremely useful for further research on the impact of your issue. It provides detailed descriptions and examples of the issue type.

3.4.2. Triaging an issue

To triage an individual issue, update its various attributes by completing the following steps:

1. Assign the issue an owner by selecting his/her username from the *Owner* drop-down menu.
2. Set the issue's classification, severity, and action to be taken by selecting the appropriate values from their respective drop-down menus.

**Note**

For detailed information on the issue attributes you can triage, see Section 3.4.5, “Issue attributes”.

- After you have triaged the issue attributes, click **Apply**. Your applied triage state is updated in Coverity Connect. You can also click **Apply + Next** to apply your changes and move on to the next issue in the list.

The **Close Issue** button closes the issue in the Details view and deselects the open issue in the Issues view. If you have unsubmitted triage changes, Coverity Desktop will prompt you to either apply the changes, close without applying the changes, or cancel the closing process.

3.4.3. Occurrences tab

The *Occurrences* tab shows the stream in which the occurrence of the issue is located, lists the name of the event(s) that lead to the issue for each category, and the filename and line number where this event was detected. Click the event name to go to the event in the source editor.

You can also browse through the next/previous occurrence of the CID (if more than one occurrence exists) by using the arrow controls labeled *Occurrence 1 of <n>*.

Coverity Desktop provides tags that highlight the line where the event appears in the code. Tags that are associated with red markers indicate events and messages that are directly related to the issue. Tags that are associated with green markers indicate conditional branches and the programmatic decisions necessary for this issue to occur. When you double click a marker, the source editor focuses on the section of code in which the event occurs. Coverity Desktop provides the following markers:

Table 3.7. Issue markers

Marker	Description
	Issue main event.
	Issue event.
	Multiple issue events that occur on the same line.
	Path event.

3.4.4. History tab

Each time an attribute is changed or a comment is added, the user responsible for that change is stored in the database, and the listing on the bottom of the issue summary is updated to reflect the modification history. Click the History tab to see the complete history of the modifications.

3.4.5. Issue attributes

You can set the classification, severity, action, and owner for each issue. These issue attributes are intended to tell only part of the story; if an issue is marked as a `BUG`, for example, you might also want to

document the reason in the designated text area for comments. The issue attributes are useful to filter the issue list you are viewing.

3.4.5.1. Classification

Table 3.8, “Classifications” lists descriptions for each of the possible issue classifications, along with the Issue Kind(s) each classification can be attributed to.

Table 3.8. Classifications

Classification	Description	Issue Kind(s)
Unclassified	This attribute is the default when a new result is inserted. It is intended for results that have yet to be viewed by a developer.	Quality, Security, Test Violation
Pending	An issue that has been examined, but has yet to be conclusively classified.	Quality, Security, Test Violation
False Positive	Results that are not real issues in the code. If these appear to reflect shortcomings or flaws in the analysis engine, report the issue to software-integrity-support@synopsys.com .	Quality, Security
Intentional	Designates that analysis has accurately diagnosed behavior that is usually unintentional, but in this scenario is actually the intended behavior.	Quality, Security
Bug	Reflects a determination that the issue found by Coverity analysis is an issue in the code, and is not a False Positive or Intentional.	Quality, Security
Untested	Reflects a determination that a section of the code analyzed by Test Advisor requires that a test be added to the code.	Test Violation
No Test Needed	Reflects a determination that even though the Test Advisor analysis found a section of code that is not covered by a test, you are aware of the violation and that there is an accepted reason that the code is not covered.	Test Violation
Tested Elsewhere	Indicates that a section of code is tested outside of the set of tests that are specified in the Test Advisor analysis process.	Test Violation

3.4.5.2. Severity

Severities for issues describe how critical the issue is, that is, how much damage the issue will potentially cause to your program. Coverity Connect allows you to add, delete, and rename severity issue attribute values. Coverity Desktop displays these customized attributes if they exist in the most current snapshot. For more information, see the *Coverity Connect User Guide*. The default severity attributes are:

Unspecified

This attribute is the default when a new issue is inserted. It is intended for those results that have not been viewed by a developer.

Major

Issues that have been confirmed as major reflect a decision that the bug is of the highest level of urgency. It is probable that major severity bugs should be fixed as soon as possible.

Moderate

Issues that have been confirmed as moderate severity bugs that should be fixed in the near term.

Minor

Issues that have been confirmed as minor severity bugs are perhaps of lesser priority.

3.4.5.3. Action

Actions describe what should be done about the issue in question. Coverity Connect allows you to add, delete, and rename action issue attribute values. Coverity Desktop displays these customized attributes if they exist in the most current snapshot. For more information, see the *Coverity Connect User Guide*. The default action attributes are:

Undecided

This attribute is the default when a new issue is inserted. It reflects that no decision about fixing or ignoring has been made.

Fix Required

The issue is outstanding and requires a fix; such an issue will continue to appear in future commits.

Fix Submitted

The issue is fixed in the source code, but the fix has not been identified as Fixed through the build, analysis, and commit processes.

Modeling Required

An investigation is required of each method in the application that is used for interprocedural analysis, created as each function is analyzed. For example, the model shows which arguments are dereferenced, and whether the function returns a null value. This can be a form of false positive in which after the modeling is corrected, the analysis will no longer report this issue.

Ignore

The issue can be ignored. This might be an appropriate action for a bug of minor severity.

3.4.5.4. Ext. Reference

You can provide an external reference (such as an issue number in a different database).

3.4.5.5. Owner

To provide or change the owner of the issue, enter the user's name in the text field.

3.4.5.6. Comment

You can provide a comment for the issue in the designated text area.

3.5. Test Advisor annotations

Coverity Desktop integrates with Test Advisor by allowing you to view test policy violations along with your other Coverity issues. To view and triage Test Advisor issues, set the Issues View to *Remote Issues* mode, and *Filter* by Outstanding Test Violations.

Test Advisor issues, such as coverage, impact, and exclusion data, are represented with various annotations in the left gutter of the text editor pane. See Table 3.9, “Test Advisor annotations” for a description of each. For function level test violations, only coverage data for the relevant function is displayed. For file level test violations, coverage data for the entire file is displayed.

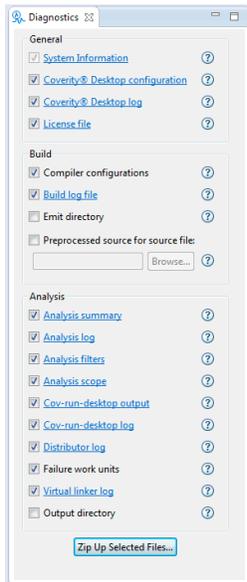
Table 3.9. Test Advisor annotations

Icon	Description
	<p>Covered [Old, Med, New] - Green bars denote that the line or function is covered by one or more tests. The number of bars indicate the code age for the line, with one line for older code, two bars for medium, and three bars for newer code.</p> <p>Hover over the green bar(s) to see a list of tests which cover the line.</p>
	<p>Uncovered [Old, Med, New] - Red bars denote that the line or function is not covered by any test. The number of bars indicate the code age for the line, with one line for older code, two bars for medium, and three bars for newer code.</p>
	<p>Coverage Exclusion - Grey bars denote that the line has been excluded from consideration by Test Advisor.</p> <p>Hover over the grey bar to see the reason for exclusion, if one was specified.</p>
<p>Month DD, YYYY</p>	<p>Impact Date - Displays the most recent impact date for the line, class, or function.</p> <p>On function and class definition lines, you can hover over the date to see additional impact information or click the link to <i>View in Coverity Connect</i>. If the information listed in the pop-up is insufficient, Coverity Connect provides additional data and a richer interface for working with coverage issues. See the <i>Test Advisor 2020.12 User and Administrator Guide</i> for more information.</p>

3.6. Collect diagnostics

Coverity Desktop provides the Diagnostics view, which helps you create a zip file of diagnostic information for your active Analysis Configuration.

Analysis options and views



To create the diagnostics zip file:

1. Navigate to Coverity → Collect Diagnostics for Coverity Support.
2. Click the check boxes to select which diagnostics items you want to include. You can view the information that will be collected for each item by clicking the link, where applicable.
3. Click the **Zip up selected files...** button.
4. Save the zip file to the desired location.

Chapter 4. Analysis Configurations Dialog

Table of Contents

4.1. General configuration	31
4.2. Coverity Connect	33
4.3. Streams	35
4.4. SCM	36
4.5. Advanced configuration	37
4.6. Dynamic variables	45

Analysis options and Coverity Connect server information are provided to Coverity Desktop through grouped settings known as analysis configurations. Upon installation of the plug-in, you will have a default analysis configuration in place, which requires some additional settings and connection information to be used for analysis. You may also choose to create additional configurations, to toggle between different options or streams from analysis to analysis.

The Analysis Configurations Dialog, accessible via the Coverity Desktop toolbar or Coverity → Analysis Configurations... menu item, allows you to create and edit analysis configurations for Coverity Desktop. To create a new configuration, select *New Configuration* from the *Active Configuration* drop-down, choose a name for the configuration, and then fill out the appropriate information in each of the dialog's subsequent tabs. Optionally, when creating a new configuration, you can copy the settings from an existing configuration (if any) and then edit them in place. The following sections contain details for each of the configuration tabs.



Note

To edit an existing configuration, select its name from the *Active Configuration* drop-down, and update the information in the configuration tabs accordingly.

Additionally, the *Edit Configurations* option (in the *Active Configuration* drop-down) allows you to rename or remove any existing analysis configurations.

Resetting individual fields/values

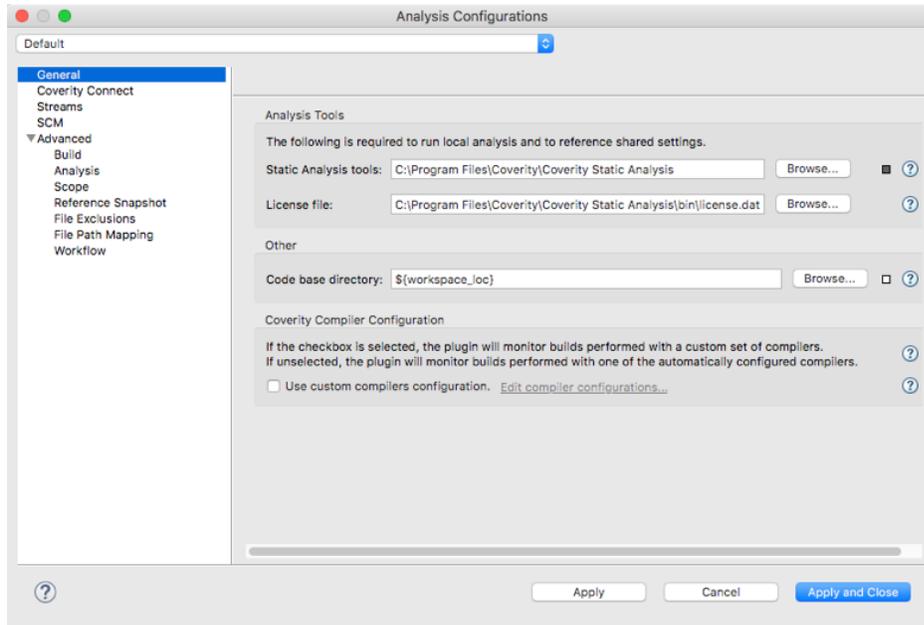
Many of the fields in the *Analysis Configurations* dialog are accompanied by a small grey (or black) "reset box". When you hover over these boxes, they will display how the value was configured (either locally configured, inherited from a `coverity.conf` configuration file, or the Coverity Desktop default value). If the box is black, this means that the value has been set locally. To reset to the original value, click on the box and select *Reset*. This will use the value specified by the `coverity.conf` file, if one exists, or reset to the Coverity Desktop default.

See the *Coverity Desktop Analysis 2020.12: User Guide* for more information on `coverity.conf`.

4.1. General configuration

The *General* tab, as shown in Figure 4.1, "General tab", specifies the disk locations of your Coverity Analysis tools, including license files, as well as your code base directory. This information is required for running local analysis and loading shared settings from a `coverity.conf` file.

Figure 4.1. General tab



Static Analysis tools

The full path to your Coverity Analysis installation directory. This is required for local analysis.

License file

The location of your `license.dat` file, which contains your existing Coverity Analysis license. The default location is `<install_dir>/bin`.

Code base directory

The plug-in will store additional Coverity configuration information in the specified directory, as well as look for a `coverity.conf` file here. The `coverity.conf` file may be created by your Coverity Connect or analysis administrator, and used to specify configuration information common to all developers. See the *Coverity Desktop Analysis 2020.12: User Guide* for additional details.

Use custom compilers configuration

This option allows you to specify a custom set of compilers for building the files in the current Analysis Configuration. If enabled, click *Edit compiler configurations* to open the *Compiler Configuration* dialog, and specify your project's compilers.

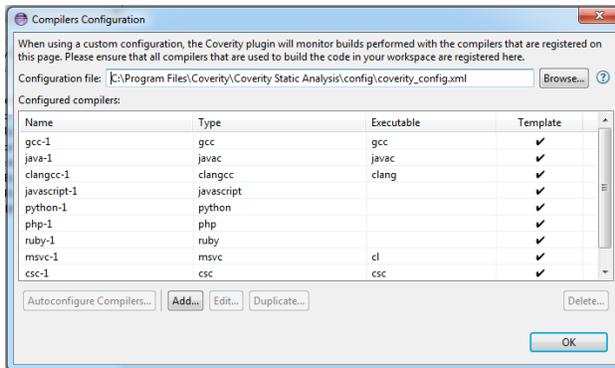
If this option is left unselected, Coverity Desktop will use the default compilers, which are configured automatically. These include:

- GNU C/C++ compiler (`gcc`)
- Microsoft Visual C/C++ compiler (`cl`) - for Microsoft Windows only
- Sun/Oracle compiler (`javac`)

- Microsoft C# compiler (csc) - for Microsoft Windows only
- Clang compiler (clangcc)

4.1.1. Compiler Configuration dialog

Figure 4.2. Compiler Configuration



To use a custom compiler configuration, add each of your required compilers by completing the following steps:

1. Click **Add...**
2. Choose a new configuration name for the compiler.
3. Choose the appropriate compiler from the *Compiler type* dropdown menu.
4. In the *Compiler executable* field, enter the command line name of the compiler.

Note

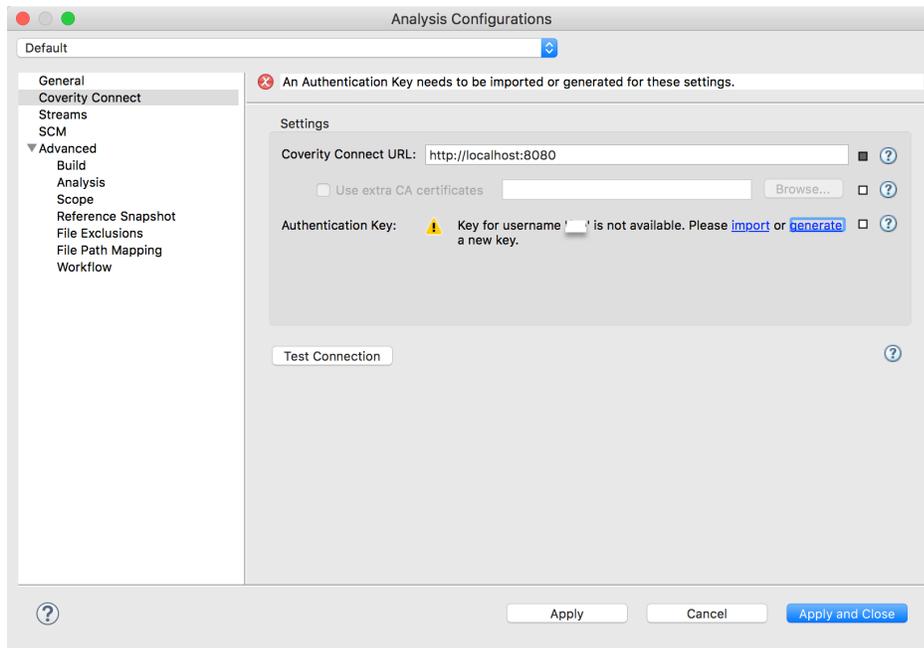
For a compiler like Wind River Diab C, which has both C and C++ variants, just enter the name of the C compiler. The C++ compiler will be configured automatically.

5. Click **OK** to add it to the list of configured compilers.

4.2. Coverity Connect

The *Coverity Connect* tab specifies server and authentication information for the Coverity Connect instance relevant to each analysis configuration. This information is required for local analysis, and for retrieving remote issues.

Figure 4.3. Coverity Connect tab



Coverity Connect URL

The fully qualified URL for your *Coverity Connect* server. This should include protocol, host name, and port number; for example, `https://connect.synopsys.com:8080`. If you are connecting to a *Coverity Connect* server over SSL, using a certificate signed by a recognized CA, the host name needs to match the name of the host on the certificate. This is not necessary when you use an unsecured connection, or when you use a self-signed certificate from *Coverity Connect*.

Use extra CA certificates

You can also turn on the check box for *Use extra CA certificates* and use the controls to point to a directory that contains additional CA certificates for communicating with *Coverity Connect*. For additional details, see "Using SSL with *Coverity Analysis*" in the *Coverity Platform 2020.12 User and Administrator Guide*.

Authentication Key

This field displays information about any existing authentication key file for the specified *Host name* and *Port*.

If authentication fails, or no authentication key exists, you will be prompted to import or generate a new key file. You can generate a new key by clicking the **generate** link and entering your username and password. You can also import an existing key by clicking the **import** link and choosing a key file that's already been created or saved.

If you attempt to import or generate a key when a key file already exists, then a **Replace key** dialog box appears, prompting you to either **Replace** the old key file or to **Save as new**.

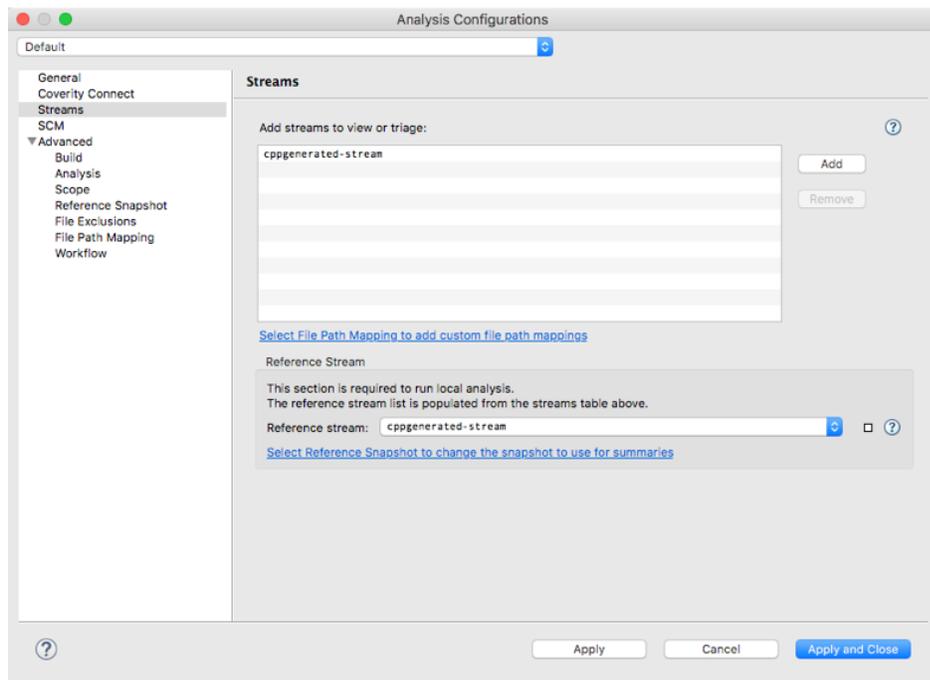
Test Connection

Click this button after entering all connection information. This will test the connection and validate your settings.

4.3. Streams

The *Streams* tab allows you to specify any relevant streams for which you want to view and triage remote issues, as well as select a reference stream local analysis.

Figure 4.4. Streams tab



Stream list

Specifies the streams that contain issues you want to view and/or triage. Click the *Add* button to open the *Select Streams* dialog, which displays a list of all available streams for you to choose from.

Note that all selected streams should use the same triage store.

Select File Path Mapping...

This will take you to the File Path Mapping tab.

Reference Stream

Select the reference stream that the analysis configuration will use for local analysis.

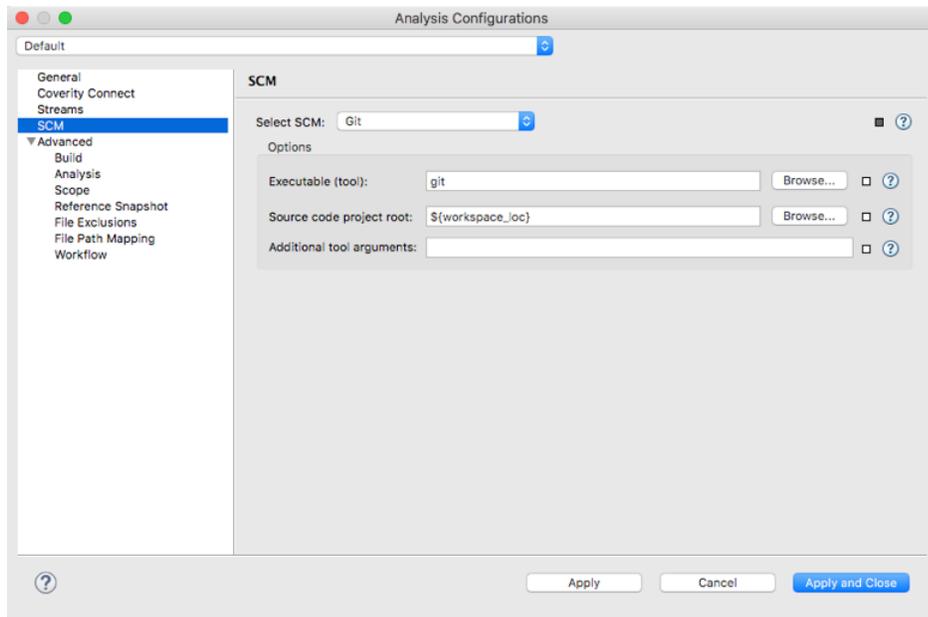
Select Reference Snapshot...

This will take you to the Reference Snapshot tab.

4.4. SCM

The *SCM* tab is an optional configuration, which specifies any SCM information for the current Analysis Configuration. This is required only if you want to use the Analyze Modified Files command for local analysis.

Figure 4.5. SCM tab



Select SCM

Select your source code management system from the drop-down menu. The following SCM systems are currently supported:

- Accurev
- Clearcase
- CVS
- Git
- Mercurial
- Microsoft ADS
- Microsoft TFS
- Perforce
- Plastic

- Plastic (distributed)
- Subversion

Executable (tool)

This is the executable for querying the specified SCM. To be used, the SCM must be installed in the command PATH.

Source code project root

This is the path to the root of the source repository.

 **Note**

For Clearcase, this should point at the target vob folder under the snapshot view location. For example: `<snapshot_view_location>/<target_VOB>`

Additional tool arguments

This is a sequence of additional command line arguments to pass after the executable ("tool") name.

P4PORT (Perforce only)

The protocol, host name, and port number of the Perforce server, in the format "`<protocol>:<host>:<port>`".

The `<protocol>` is either `tcp` or `ssl`.

P4CLIENT (Perforce only)

The name of the Perforce Client, as typically shown in the P4CLIENT environment variable. This is also known as a workspace name.

Remote Repository (Plastic (distributed) only)

The location of the central Plastic server. This should be in the format of "`<repository>@<remote-server>:<port>`".

This field is required.

Authentication Arguments (Plastic (distributed) only)

If needed, the arguments that will be passed directly to 'cm replicate'. Arguments include, but are not limited to, `--user` and `--password`.

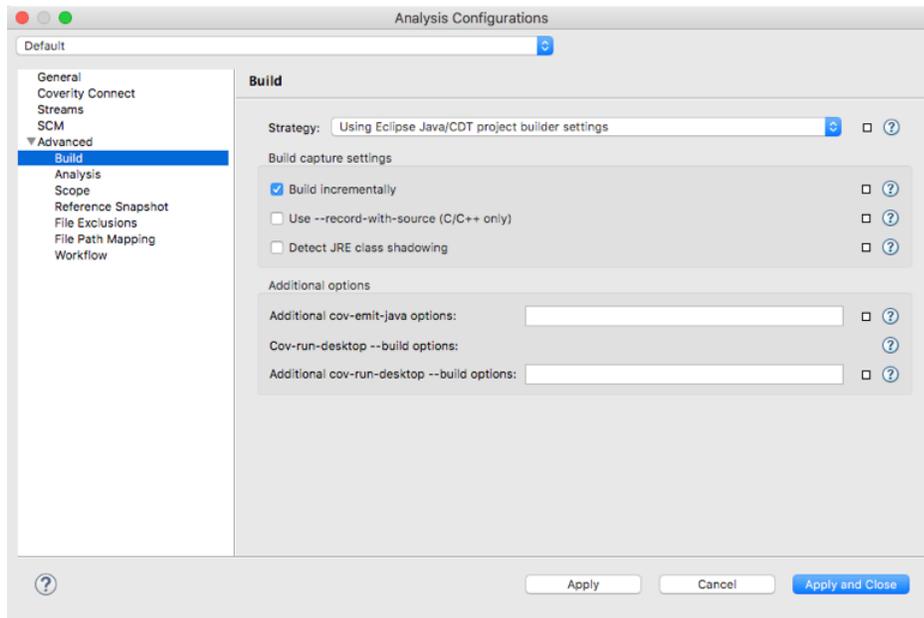
This field is optional.

4.5. Advanced configuration

The *Advanced* configuration tabs allow you to configure additional build and analysis options for more advanced use cases.

4.5.1. Advanced: Build

Figure 4.6. Advanced: Build tab



Strategy

Select whether to build your workspace using the Eclipse Java/CDT project builder, or specify your own build/clean commands. If you choose to build using custom settings, the *Command settings* section will be displayed, where you can specify your build commands and working directory. You can also enable incremental builds, which allows you to capture previously uncaptured source files for local analysis, without having to complete a full build of your entire project/workspace.

For C/C++ projects in Eclipse, the Eclipse CDT (C/C++ Development Tools) provides *Build Configurations*. *Build Configurations* specify build processes to create different variants of a project. To create new configurations or specify the default configuration, access the *Manage Configurations* dialog from the *Project Properties* of a C/C++ project. The Coverity Dynamic Analysis will use the active *Build Configurations* for each project, thus enabling the developer to build different variants of the project without setting up custom settings in *Analysis Configurations*.

Command settings

This section contains the options for configuring custom build settings. It is displayed only if you have chosen the *Strategy* option *Using custom settings*.

Configure the following settings:

- **Clean command** - Command line to clean (delete) artifacts produced by the build, so that the next build command will recompile all of the source code. For example, `make clean`.
- **Build command** - Command line to compile the source code. For example, `make` (for C/C++) or `ant` (for Java).

- **Working directory** - The clean and build command will be run from this directory.
- **Build incrementally** - When enabled, selecting the *Capture and Analyze* button in the Uncaptured Source Files dialog will execute the build command and attempt to capture the uncaptured files. The custom Clean command is not executed.

When running *Capture build of Entire Scope*, this setting is ignored and the custom Clean command is executed.

Build capture settings

- **Use --record-with-source** - When enabled, the build will also capture header file dependencies, which increases the breadth of your analysis, but may also slow down the process considerably.
- **Detect JRE class shadowing** - When enabled, the plug-in will attempt to prepend custom JRE classpaths to the value passed to `--bootclasspath` when running `cov-emit-java`. This will use custom classes instead of JRE classes of the same name. This option is available only if you have chosen the *Using Eclipse Java/CDT project builder settings* option.

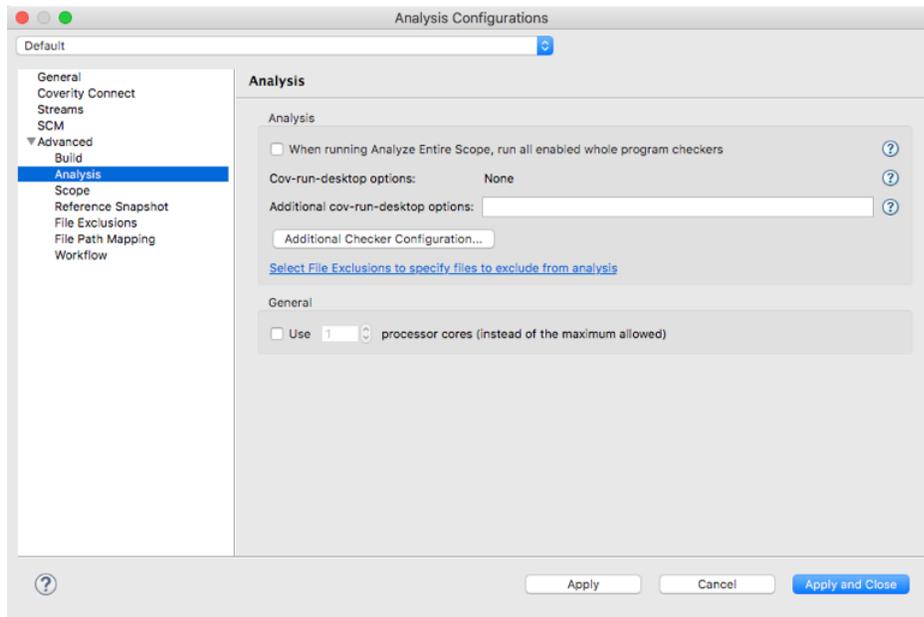
Additional options

- **Additional cov-emit-java options** - Additional options to pass to `cov-emit-java` during the emit portion of the build. This option is available only if you have chosen the *Using Eclipse Java/CDT project builder settings* option.
- **Cov-run-desktop --build options** - Options to pass to `cov-run-desktop` during the build. These are inherited from the `coverity.conf` file.
- **Additional cov-run-desktop --build options** - Additional options to pass to `cov-run-desktop` during the build. These will be appended to the list above. If there is a conflict, values defined here will be used.

See the *Coverity 2020.12 Command Reference* for information on available `cov-emit-java` and `cov-run-desktop --build` options.

4.5.2. Advanced: Analysis

Figure 4.7. Advanced: Analysis tab



When running Analyze Entire Scope, run all enabled whole program checkers

If whole program checkers are defined in the *Additional Checker Configuration* dialog, the analysis will attempt to use them. This includes coding standard checkers and web application security checkers.

Cov-run-desktop options

Options passed to `cov-run-desktop` for local analysis. These are inherited from the `coverity.conf` file. See the *Coverity Desktop Analysis 2020.12: User Guide* for more information on `coverity.conf`.

Additional cov-run-desktop options

Specifies any additional options to be passed to `cov-run-desktop` during local analysis.

These options will be added to those listed under the *cov-run-desktop options* field. If there is a conflict, the options specified here take precedent.

Select File Exclusions...

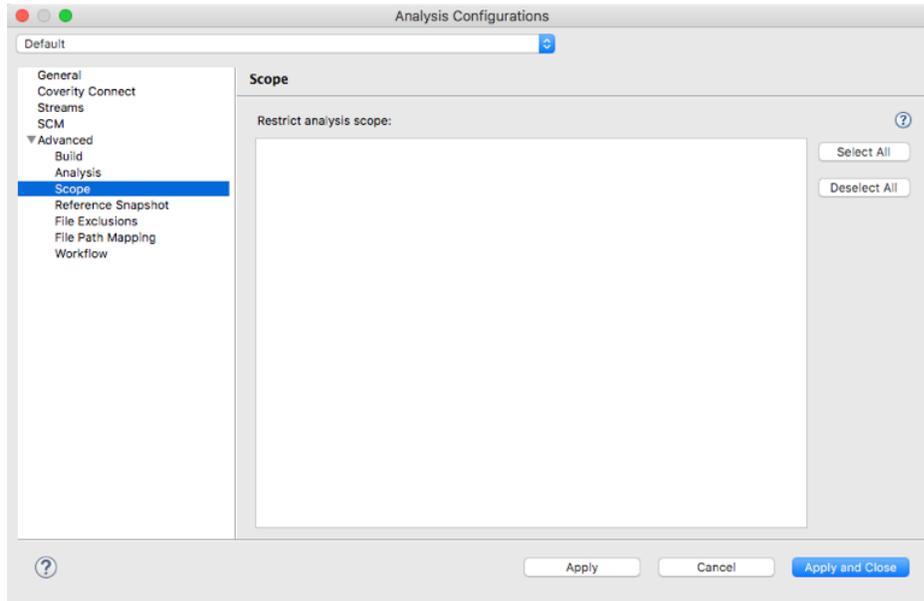
This will take you to the File Exclusions tab.

Use <N> processor cores

Specifies the number of cores to use for parallel analysis. The default is the smaller of the number of cores on the machine and the number allowed by the license file.

4.5.3. Advanced: Scope

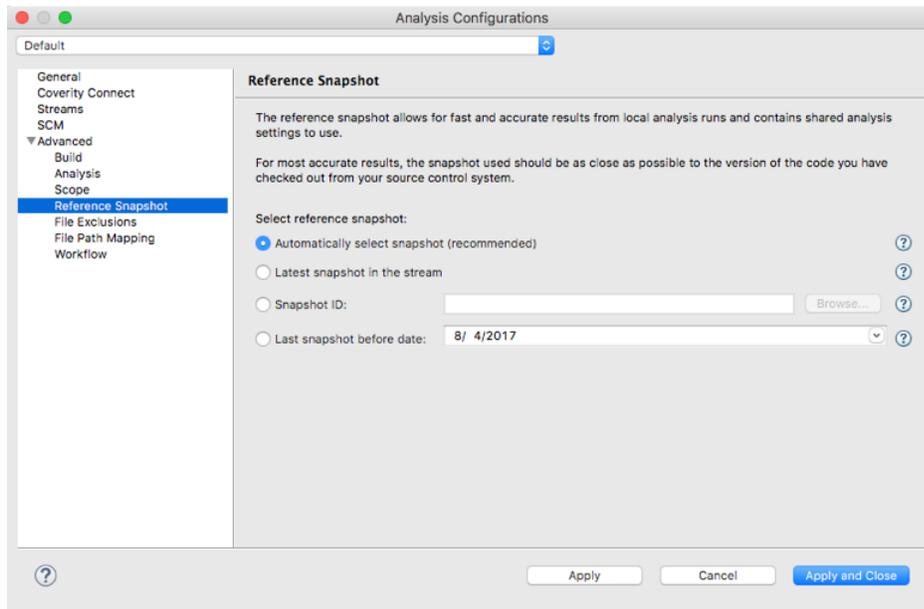
Figure 4.8. Advanced: Scope



The *Scope* tab allows you to restrict your local analyses to specific files and directories. Use the file tree to select which source files to analyze with the active analysis configuration. To analyze all of the files specified, use the Analyze Entire Scope option.

4.5.4. Advanced: Reference Snapshot

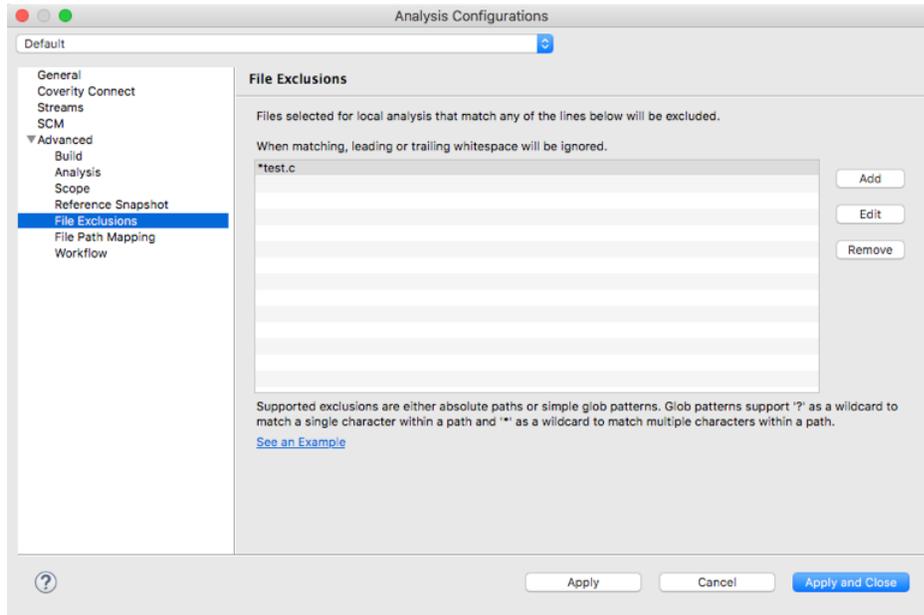
Figure 4.9. Advanced: Reference Snapshot



The *Reference Snapshot* tab lets you select which reference snapshot to use for local analysis. It is recommended that you use the *Automatically select snapshot* option. This should provide the most accurate analysis results, by using the snapshot closest to the version of the code being analyzed. If you have not configured an SCM, this option will instead use the latest snapshot in the reference stream. Alternatively, you can choose to use the last snapshot committed before a certain date, or choose a specific snapshot ID.

4.5.5. Advanced: File Exclusions

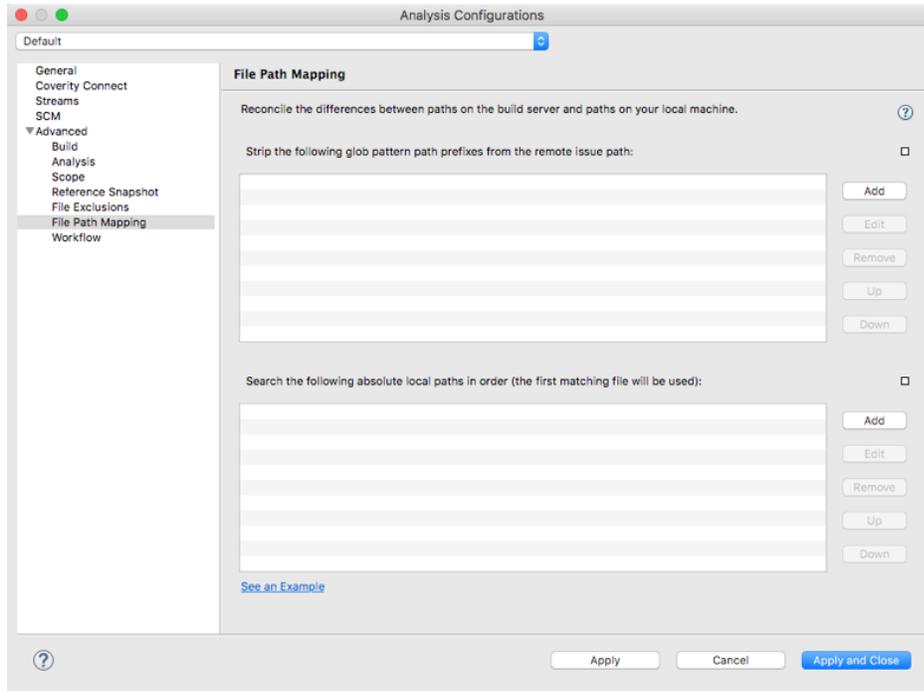
Figure 4.10. Advanced: File Exclusions



The *File Exclusions* tab lists any files or filepath patterns that should be excluded from local analysis. To add a file exclusion, click the *Add* button and enter the file path or glob pattern to exclude.

4.5.6. Advanced: File Path Mapping

Figure 4.11. Advanced: File Path Mapping

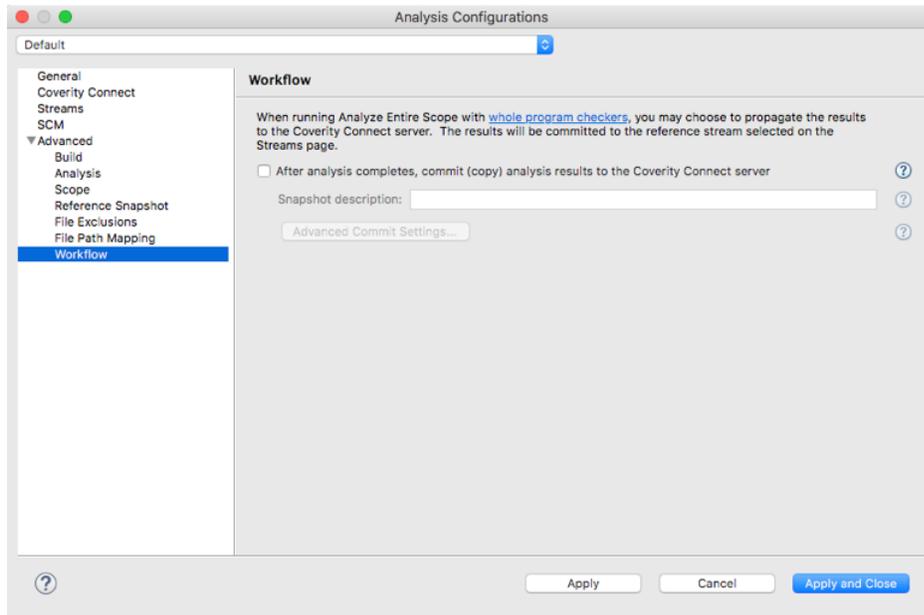


The *File Path Mapping* tab allows you to translate between remote and local file paths. With path translation, you can configure Coverity Desktop to strip a specified path prefix, and then search local paths for matching files. Either absolute paths or simple glob patterns may be used.

In the event that the plug-in cannot resolve the path mapping or locate a source code file locally, the filename will appear in red in the *Issues* view, and you will not be able to view the issue in the source file.

4.5.7. Advanced: Workflow

Figure 4.12. Advanced: Workflow tab



The *Workflow* tab presents an option to send analysis results to the Coverity Connect server:

- **After analysis completes, commit (copy) analysis results to the Coverity Connect server**

Selecting this box will commit a new snapshot, containing all issues found by local analysis, to the Coverity Connect server associated with the analysis configuration. This option will only run by selecting the *Analyze Entire Scope* command and only if the *When running Analyze Entire Scope, run all enabled whole program checkers* option is enabled on the analysis page.

Specify a *Snapshot Description* in the text field, and click the **Advanced Commit Settings** button to configure any additional commit options.

 **Note**

If this option is enabled for an Analysis Configuration, running local analysis with that Analysis Configuration will not display any CIDs or triage information in the IDE.

Likewise, when you choose this option, analysis is disconnected from the Coverity Connect snapshot. Therefore, the analysis options included in the reference snapshot are not used.

4.6. Dynamic variables

The following variables are supported in the Coverity Desktop UI:

- `${env_var:<ANY_ENVIRONMENT_VARIABLE_NAME>}` - Substitutes corresponding environment variables.

- `${workspace_loc}` - Substitutes the location of the IDE workspace.
- `${cov-inter-dir}` - The full path to the intermediate directory. For example:
`/home/me/workspace/.metadata/coverity/idir`
- `${cov-im-user}` - The Coverity Connect username.
- `${cov-im-password}` - The Coverity Connect password.
- `${cov-im-host}` - The Coverity Connect host name.
- `${cov-im-port}` - The Coverity Connect port number.
- `${cov-im-auth-key}` - The full path to the location of the Coverity Connect authentication key file.
- `${cov-sa-bin}` - The full path to the Coverity Static analysis `/bin` directory. For example:
`<install_dir>/bin`

Chapter 5. Troubleshooting Coverity Desktop for Eclipse

This troubleshooting section provides instructions for fixing the following common issues with Coverity Desktop:

1. Coverity Desktop plug-in not displayed in the IDE after installation, despite appearing in the list of installed plug-ins
2. Coverity Desktop cannot connect to Coverity Connect using domain names
3. Local analysis causing PermGen memory issues
4. Eclipse crashes when opening context help
5. When inspecting and triaging issues, the Eclipse plug-in sometimes opens a file with a `coverity_external_files` prefix
6. Clicking the CWE and/or SpotBugs link in the Details view does not open the correct URL on Linux platforms
7. The `cov-emit-java` tool is not being invoked during analysis when using a non-standard builder configuration (Ant Builder, custom made builder, etc)
8. Expected Java source code files are not appearing in the list of translation units returned by the `cov-manage-emit` tool
9. Analysis returns the error message "Cannot find any compiler outputs for Java files" when using a non-standard builder configuration (Ant Builder, custom made builder, etc)
- 10LDAP users: *My Outstanding* filter not returning expected results in *Issues* view
- 11 *Analyze Entire Workspace* command attempts to analyze files that aren't part of the build, causing errors
- 12.The "*Uncaptured Source Files*" dialog appears and pressing the "**Capture Build and Analyze**" button does not resolve the problem
- 13.Unable to open remote issues and filepath displays in red text in the Issues view, despite files being present in your workspace
- 14[ERROR] No snapshot in stream "<streamName>" has analysis summaries...

Coverity Desktop plug-in not displayed in the IDE after installation, despite appearing in the list of installed plug-ins

The Coverity Desktop plug-in requires Java 1.8+ to run. (Note that you must install the 32-bit Java version if you use a 32-bit IDE. This applies to both WindRiver 3.2 and QNX 4.7 32-bit only). To fix this issue, install Java 1.8 or later, and update your IDE executable to launch with the correct Java version:

For Eclipse and QNX:

1. Navigate to the IDE's install directory.

2. Locate the executable file that is used to launch the IDE (`eclipse.exe` for example). There should be an associated `.ini` file with the same name (for example, `eclipse.ini`).

If the `.ini` file does not exist, create it.

3. Open the `.ini` file and look for the argument, `-vm`. If present, add the path to the new Java executable directly below the `-vm` line. Otherwise, add `-vm` to the file, with the path to the new Java executable on the following line.

 **Note**

Make sure that:

- The `-vm` and the path are on separate lines.
- The path points directly to the Java executable - not just the Java home directory.
- If `-vmargs` is present in the file, the `-vm` argument and path must come before it.

4. Run the IDE's executable file. You should be able to see the Coverity features now.

For WindRiver

To launch WindRiver with the appropriate Java version, launch the IDE from the command line, using the `-vm` argument:

From the command line, enter the IDE's executable file name, followed by "`-vm <path_to_java_executable>`".

Coverity Desktop cannot connect to Coverity Connect using domain names

1. Try to turn off your firewall.
2. If you use proxy settings, try to include all necessary domain names into your proxy exceptions.

Local analysis causing PermGen memory issues

 **Note**

The Permanent Generation space is removed in Java 1.8. The JVM will ignore the options `-XX:PermSize` and `-XX:MaxPermSize`.

On Eclipse, running local analysis using the default memory settings might cause PermGen memory issues. To increase the default PermGen space, add the following arguments to the command line:

```
-XX:PermSize=64M -XX:MaxPermSize=384M
```

For example:

```
% eclipse -vmargs -XX:PermSize=64M -XX:MaxPermSize=384M
```

The memory size needed depends on your system, so suggested settings to try are: 256M, 384M, or 512M. For more information about PermGen settings, see the Eclipse documentation at http://wiki.eclipse.org/FAQ_How_do_I_increase_the_permgen_size_available_to_Eclipse%3F. 

Eclipse crashes when opening context help

Upgrade to the most recent Java Runtime Environment.

For additional information, please refer to Eclipse Bug 353740. https://bugs.eclipse.org/bugs/show_bug.cgi?id=353740 

When inspecting and triaging issues, the Eclipse plug-in sometimes opens a file with a `coverity_external_files` prefix

This occurs because the Eclipse plug-in can not locate the file inside the workspace, so it creates a hidden project called `coverity_external_files` to load the file and show markers as if it were part of the workspace. Sometimes the file is not found because of links in a project that cause the source file and the project to have different path prefixes.

If the file that was opened inside the `coverity_external_files` path is located in the current workspace, then the plug-in needs additional information to locate that file. To resolve this situation, complete the following steps:

1. Navigate to Preferences → Coverity Analysis → Central Analysis.
2. Ensure that the *View and triage issues from Coverity Connect* check box is selected.
3. Click the **Issue Location...** button.
4. Add the strip path to your local source files in the *Strip remote path prefixes* dialog. After closing the preferences dialog and refreshing the Issues view, opening an issue should open the proper file that is found in the workspace as expected.

Clicking the CWE and/or SpotBugs link in the Details view does not open the correct URL on Linux platforms

If the CWE or SpotBugs link does not open to the correct location, right-click the link and copy the URL. Then paste the URL into your web browser.

Non-standard builder errors

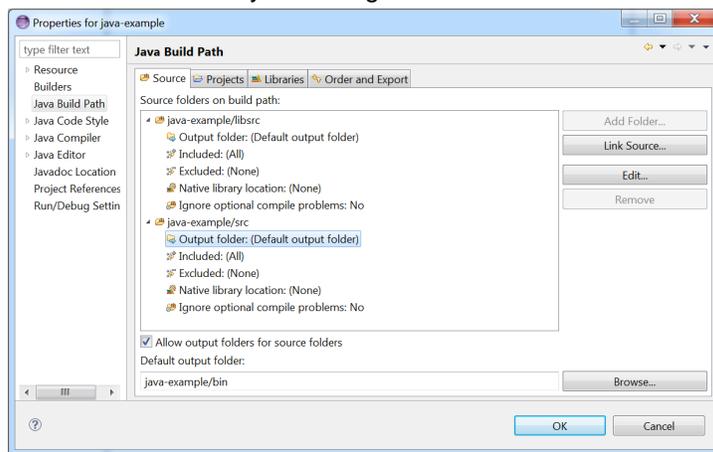
This is a common issue with non-standard builder configurations, which can manifest in one of three ways:

- *The `cov-emit-java` tool is not being invoked during analysis when using a non-standard builder configuration (Ant Builder, custom made builder, etc)*
- *Expected Java source code files are not appearing in the list of translation units returned by the `cov-manage-emit` tool*
- *Analysis returns the error message "Cannot find any compiler outputs for Java files" when using a non-standard builder configuration (Ant Builder, custom made builder, etc)*

This is likely caused by your `.class` files being written to an unexpected location that the Eclipse plug-in can not find. To fix this issue, ensure that the output folders are configured correctly for each source folder by completing the following steps:

1. Right-click on the project in the *Project Explorer* and select *Properties*.

2. Open the *Java Build Path* page.
3. Check the *Allow output folders for source folders* checkbox.
4. Expand each source folder entry in the *Source folders on build path* pane.
5. For each *Output folder*, select it and click **Edit...**
 - a. Choose the *Specific output folder* option.
 - b. Enter the path of the output folder relative to the source folder.
6. Create a new Analysis Configuration for the project (the changes may go undetected if you reuse an existing Analysis Configuration).
7. Run the new Analysis Configuration.



LDAP users: *My Outstanding* filter not returning expected results in *Issues* view

Be sure that you have logged into the Coverity Connect server using the full LDAP username format: `<user>@<domain>`.

Analyze Entire Workspace command attempts to analyze files that aren't part of the build, causing errors. For any files in your workspace that should be excluded by Desktop Analysis, a resource filter should be added:

1. Right-click on the project that contains the extra files in the *Project Explorer* tab.
2. Click on *Properties*.
3. Expand the *Resource* menu, and select *Resource Filters*.
4. Click **Add...** to create a new filter, excluding any files or directories from inclusion in the project.
5. Apply your changes to finish adding the filter.

The "*Uncaptured Source Files*" dialog appears and pressing the "**Capture Build and Analyze**" button does not resolve the problem

This situation occurs if a file specified for analysis is not among those that Coverity Desktop recognizes as having been compiled during the build capture.

For C/C++, try the following potential solutions:

Solution 1: Files not meant for analysis

In the case that the files in question are never captured because they are not actually part of the project (and so not meant for analysis), proceed with your analysis by clicking **Analyze Captured Files Only**.

It may also be useful to exclude these files from all future analyses. To do so, check the box to always ignore the files in question, or navigate to Coverity → Configuration → Coverity Analysis → Local → Analysis and click on the **File Exclusions** button. This will launch the *Local Analysis File Exclusions* dialog, where you can specify any files you want excluded from your local analyses.

Solution 2: Confirm your compiler configurations

It is possible that the file you are attempting to analyze uses a compiler that has not yet been configured in Coverity Desktop. Navigate to Coverity → Configuration → Coverity Analysis → Local → Compiler Configuration and confirm that all of the compilers used by your project are configured for use.

After adding any new compilers, be sure to recapture the build by selecting *Capture Build of Entire Workspace* from the *Coverity* menu, and try the analysis again.

Solution 3: Non-primary source file capture

If the file you are attempting to analyze is a non-PSF, such as a header file, then it is necessary to use the `--record-with-source` option for the build capture. Navigate to Coverity → Configuration → Coverity Analysis → Local → Build and select *Use --record-with-source*.

After selecting the `--record-with-source` option, be sure to recapture the build by selecting *Capture Build of Entire Workspace* from the *Coverity* menu, and try the analysis again.

This problem is usually very rare when analyzing Java source code, but if it does occur, please try the following solution:

Solution 1: Files not meant for analysis

In the case that the files in question are never captured because they are not actually part of the project (and so not meant for analysis), proceed with your analysis by clicking **Analyze Captured Files Only**.

Solution 2: When using a custom Java build command, ensure the working directory is correct
If you have:

- Imported Java source code from some original location into a project in your workspace
- Elected to copy the files into the workspace rather than establishing links in the workspace to the original location
- Supplied a custom Java build command

Then ensure that the working directory under Coverity → Configuration → Coverity Analysis → Local → Build matches the location of the project directory within the workspace. If the working directory refers to the original location rather than the location within the workspace, then although the build may succeed, the wrong set of files will be compiled and the plug-in will continue to complain that the expected files were not build-captured.

If none of these solutions resolve the issue, contact
software-integrity-support@synopsys.com.

Unable to open remote issues and filepath displays in red text in the Issues view, despite files being present in your workspace

This is likely caused by a discrepancy between remote and local file paths. To fix this, make sure that your file path mappings are configured correctly:

1. Navigate to Coverity → Analysis Configurations → Advanced → File Path Mapping.
2. In the first box, enter the incorrect path prefixes to be stripped (displayed in red in the Issues view).

 **Note**

You can click **See an Example** to see an example scenario for using File Path Mapping.

3. In the second box, enter the local paths you want to be searched in place of the stripped paths.
4. Click **OK** to save and exit.

[ERROR] No snapshot in stream "<streamName>" has analysis summaries...

If the SCM analysis option was used, and the codebase has a last updated date before the reference snapshot was committed to the stream:

1. Update your codebase and push the changes (so your SCM repository will be last updated after the snapshot was committed).
2. Re-run SCM analysis again on the newly updated codebase.

If you are attempting to analyze your program without using snapshot summaries, and your stream does not contain snapshots with analysis summaries, the build command will fail with this message. To work around this issue, please select the 'Go Offline' menu item and retry.

Please note, if you encountered this issue while running *Analyze with Configuration* and use the suggested workaround, the analysis will also be run in a disconnected state. If you do not want this, use *Capture Build* while offline, then select the 'Go Online' menu item (to start working online) before retrying the *Analyze with Configuration* command.

Chapter 6. Finding more information about issues

Issues are reported by specific checkers. Prevent Desktop provides information about the checkers and the types of issues they find. You can view this information by right-clicking the checker name or event in the Issues view and selecting **Show Checker Help**.

A help view opens that shows information about the checker. You can follow the links in the help view for more detailed information.