Coverity Platform 2020.12 User and Administrator Guide

Table of Contents

1. Coverity Platform Overview 1
    1.1. Installation, Product Licenses, and Supported Platforms 3
    1.2. Login 4
    1.3. Shared Navigation and Main menu tools 5
    1.4. Help Menu 8
        1.4.1. Keyboard Shortcuts 8
        1.4.2. Product downloads 8
        1.4.3. Customer feedback for Coverity 9
        1.4.4. Coverity Connect platform details 9
        1.4.5. Coverity Connect documentation set 9
2. Coverity Connect Usage 10
    2.1. Getting Started with Coverity Connect 11
    2.2. Managing issues 12
        2.2.1. Understanding the primary Coverity Connect workflows 12
        2.2.2. Finding issues 13
        2.2.3. Managing Issues 14
        2.2.4. Triaging issues 17
    2.3. Monitoring issues 24
        2.3.1. Dashboards 24
        2.3.2. View-level Charts 27
    2.4. User Reference 29
        2.4.1. Views 29
        2.4.2. View Management 34
        2.4.3. Snapshot comparison 50
        2.4.4. View-specific email notifications 54
        2.4.5. Triage attributes 57
        2.4.6. Preferences 60
3. Coverity Connect Administration 61
    3.1. Performing system configurations 62
        3.1.1. Configuring and managing the Coverity Connect server 62
        3.1.2. Configuring and managing the embedded database 129
        3.1.3. Client-side SSL Certificate Management 135
    3.2. Managing users, groups, and roles 143
        3.2.1. Managing users 143
        3.2.2. Managing groups 152
        3.2.3. Roles and role based access control 157
    3.3. Managing data on software issues 178
        3.3.1. Working with projects and streams 178
        3.3.2. Configuring triage attributes 192
        3.3.3. Using components 196
        3.3.4. Managing triage stores 213
        3.3.5. Configuring legacy issues 223
    3.4. Configuring automatic owner assignment 226
        3.4.1. Overview of the automatic owner assignment process 226
        3.4.2. Setting global SCM rules and the SCM user map 227
        3.4.3. Setting stream-level rules 233

3.5. Configuring Coverity Connect enterprise clusters 235
3.5.1. Synchronizing multiple Coverity Connect instances 235
3.5.2. Managing multiple database instances 243
3.6. Configuring a Server to Support Code Sight 245
3.6.1. Server administration for the Synopsys Code Sight Plug-in 245
3.6.2. Configuring a Coverity Connect server to support Code Sight clients 245
3.7. Server administration for Desktop Analysis 246
3.7.1. Setting up streams 246
3.7.2. Creating analysis summaries 247
3.7.3. Summary expiration 247
3.7.4. Desktop Analysis usage tracking 248
3.7.5. Analysis license service 248
3.8. Coverity Connect URL construction 249
3.9. Exported defect output 251
3.9.1. Exported defect URL variables 251
3.9.2. Exported defect XML elements 251
3.9.3. Exported defect XML file 253
3.10. Suppressing built-in views for new users 259
3.11. Changing the date format display in Coverity Connect 261
3.12. Configuring Coverity Desktop and Architecture Analysis through Coverity Connect 264
3.12.1. Configuring Coverity Desktop and shared files through the Downloads page 264
3.12.2. Downloading a CVA file for Architecture Analysis 267
4. Coverity Policy Manager Usage 268
4.1. Coverity Policy Manager Overview 269
4.1.1. Hierarchies 269
4.1.2. Metrics 270
4.1.3. Filters 270
4.1.4. Reports 271
4.1.5. Dashboards 274
4.1.6. Heatmaps 274
4.1.7. Policies 278
4.1.8. Summary Metrics 278
4.1.9. Views 281
4.2. Using Coverity Policy Manager Hierarchies 283
4.2.1. Navigating through Charts 283
4.2.2. Performing Common Coverity Policy Manager Actions 283
4.2.3. Setting up Coverity Policy Manager Dashboards 286
4.2.4. Setting up Coverity Policy Manager Heatmaps 288
4.2.5. Setting up Coverity Policy Manager Status Reports 291
4.2.6. Setting up Coverity Policy Manager Trend Reports 297
4.3. Coverity Policy Manager Metrics 301
4.3.1. Metrics: User Activity 301
4.3.2. Metrics: Analysis 302
4.3.3. Metrics: Code 304
4.3.4. Summary Metrics: Technical Debt 305
4.4. Coverity Policy Manager Filter and Segmentation Properties 307
5. Coverity Policy Manager Administration 309
5.1. Managing a Coverity Policy Manager Hierarchy 310

5.1.1. Planning a Hierarchy 310
5.1.2. Configuring a Hierarchy 320
5.2. Importing/Exporting a Coverity Policy Manager Hierarchy 327
5.3. Scheduling the Extract Transform Load (ETL) Process 330
5.4. Creating Summary Metrics 332
5.5. Coverity Policy Manager Roles and Permissions 334
6. Coverity Report Generators 335
6.1. Configuring Report Generators 336
6.1.1. Configuration overview 336
6.1.2. General considerations 336
6.1.3. Unified configuration file 349
7. Coverity Integrity Report Generation 351
7.1. Overview 352
7.1.1. Defect ratings 352
7.1.2. Recommended analysis configuration 353
7.1.3. Requirements 353
7.2. Generating a Coverity Integrity Report 354
7.3. Updating the Coverity Integrity Report configuration file 356
8. Coverity Security Report 359
8.1. Security Report Overview 360
8.1.1. Basic Workflow 360
8.1.2. Interpreting the Security Report 361
8.2. Working with the Security Report Generator 365
8.2.1. Installing the Report Generator 365
8.2.2. Connecting to Coverity Connect 366
8.2.3. Configuring the Security Report 367
8.2.4. Generating a Report 372
8.2.5. Customizing Security reports 373
9. MISRA Report 376
9.1. MISRA Report Overview 377
9.1.1. MISRA Compliance 377
9.2. Installing the MISRA Report Generator 379
9.2.1. Connecting to Coverity Connect 380
9.2.2. Configuring a MISRA Report 381
9.2.3. Using system environment variables 383
9.2.4. Generating a previously configured report 384
9.2.5. Generating a report from the command line 385
9.2.6. Localizing reports 386
10. Synopsys Software Integrity Report 387
10.1. Introduction 388
10.2. Installation 389
10.3. Setup and configuration 390
10.3.1. Setting up the Synopsys Software Integrity Report Generator 390
10.3.2. Configuring the Synopsys Software Integrity Report 391
10.3.3. Updating the report configuration file 391
10.3.4. Using system environment variables 394
10.3.5. Generating a report through the GUI 394
10.3.6. Generating a report via command line 394

10.4. PDF report outline 396
10.5. Detailed results 397
10.5.1. Coverity Results 397
11. Coverity CVSS Report 398
11.1. Introduction 399
11.1.1. Scorecard elements 399
11.1.2. Triage attributes 400
11.1.3. CVSS Report profile 401
11.2. Installing the Coverity CVSS Report Report Generator 403
11.2.1. Working with the Coverity CVSS Report Report Generator 403
11.2.2. Downloading the Coverity Reports Installer 404
11.2.3. Configuring a CVSS Report 405
11.2.4. Generating a CVSS Report 407
11.2.5. Using system environment variables 409
11.2.6. CVSS Report Input Files 409
12. Coverity CERT Report Guide 411
12.1. Overview 412
12.2. Installing the CERT Report Generator 413
12.2.1. Connecting to Coverity Connect 413
12.2.2. Working with the CERT Report 414
12.2.3. Configuring the Coverity CERT Report 415
12.2.4. Using system environment variables 417
12.2.5. Generating a previously configured report 418
12.2.6. Generating the a report from the command line 418
12.2.7. Interpreting the CERT Report 418
13. Coverity OWASP Web Top 10 420
13.1. Overview 421
13.1.1. Working with the OWASP Web Top 10 Report Generator 422
13.1.2. Interpreting the OWASP Web Top 10 Report 422
13.2. Working with the OWASP Web Top 10 Report Generator 423
13.2.1. Installing the Report Generator 423
13.2.2. Configuring an OWASP Web Top 10 Report 423
13.2.3. Generating an OWASP Top 10 Report 425
14. Coverity Mobile OWASP Top 10 428
14.1. Overview 429
14.1.1. User workflow 429
14.1.2. Interpreting the Mobile OWASP Top 10 Report 429
14.2. Working with the Mobile OWASP Top 10 Report Generator 431
14.2.1. Installing the Report Generator 431
14.2.2. Configuring a Mobile OWASP Top 10 Report 431
14.2.3. Generating a Mobile OWASP Report 433
15. Coverity PCI DSS 435
15.1. Overview 436
15.1.1. Working with the PCI DSS Report Generator 436
15.1.2. Interpreting the PCI DSS Report 436
15.2. Working with the PCI DSS Report Generator 438
15.2.1. Installing the Report Generator 438
15.2.2. Configuring a PCI DSS Report 438

15.2.3. Generating a PCI DSS Report 440

Part 1. Coverity Platform Overview

Coverity Platform supports the following components:

Coverity Connect

Developers, managers, and leads of software development teams use Coverity Connect to understand, manage, and fix issues found by Coverity Analysis and third-party tools.

Administrators set up and manage Coverity Connect for end users through Configuration screens that are accessible from the *Configuration* menu, located in the toolbar.

For details, see Part 2, “Coverity Connect Usage” and Part 3, “Coverity Connect Administration ”.

Coverity Policy Manager

Leaders of software development teams use Coverity Policy Manager charts to monitor issues found by Coverity Analysis and third-party tools.

Coverity Policy Manager administrators set up and maintain Hierarchies that support charts created by Coverity Policy Manager end users.

For details, see Part 4, “Coverity Policy Manager Usage ” and Part 5, “Coverity Policy Manager Administration”.

Coverity Integrity Report

Leaders of software development teams use Coverity Integrity Report PDF-based charts to see the current impact of Coverity Connect issues. Administrators set up and run Coverity Integrity Report to produce these reports.

For details, see Part 7, “Coverity Integrity Report Generation”.

MISRA Report

Leaders of software development teams use MISRA Report to create a formatted report on issues related to the MISRA standards in Coverity projects.

For details, see Part 9, “ MISRA Report”.

Security Report

Leaders of software development teams use Security Report to create a formatted report on security-related issues in Coverity projects.

For details, see Part 8, “Coverity Security Report”.

Synopsys Software Integrity Report

Leaders of software development teams use Synopsys Software Integrity Report to create a formatted report on security-related issues in Coverity projects.

For details, see Part 10, “Synopsys Software Integrity Report”.

Coverity CVSS Report

Leaders of software development teams use Coverity CVSS Report to create a formatted report on issues related to the CVSS standard in Coverity projects.

For details, see Part 11, “Coverity CVSS Report”.

Coverity CERT Report Guide

Leaders of software development teams use Coverity CERT Report Guide to create a formatted report on issues related to the SEI CERT standard in Coverity projects.

For details, see Part 12, “Coverity CERT Report Guide”.

OWASP Web Top 10

Leaders of software development teams use Mobile OWASP Top 10 to create a formatted report on issues related to the OWASP Web Top 10 standard in Coverity projects.

For details, see Part 13, “Coverity OWASP Web Top 10”.

Mobile OWASP Top 10

Leaders of software development teams use Mobile OWASP Top 10 to create a formatted report on issues related to the OWASP Mobile Top 10 standard in Coverity projects.

For details, see Part 14, “Coverity Mobile OWASP Top 10”.

PCI DSS

Leaders of software development teams use PCI DSS to create a formatted report on issues related to the Payment Card Industry Data Security Standard (PCI DSS) in Coverity projects.

For details, see Part 15, “Coverity PCI DSS”.

To see how Coverity products work together, see Coverity Development Testing.

Chapter 1.1. Installation, Product Licenses, and Supported Platforms

Installation instructions and supported platform details reside in the *Coverity 2020.12 Installation and Deployment Guide*:

- Installation and Licensing instructions can be found in Installation.

  Coverity Platform component accessibility varies by license.

- A list of officially supported platforms, utilities, and compilers can be found in Supported Platforms/

Chapter 1.2. Login

Coverity Connect (along with Coverity Policy Manager, its fully integrated, companion product) is a Web application that requires login credentials.

To gain access to Coverity Connect (and Coverity Policy Manager):

1. Enter the URL for the Coverity Connect server into your Web browser.

   When your user account is created, you should receive from the Coverity Connect administrator an email notification that contains your login URL and account credentials.

   Login syntax

   `<protocol>://<server_name>.<domain_name>:<port>/`

   For example, if the server is using the default port on a machine called `machine1` in a domain called `eng.company.com`, the URL is:

   ```
   http://machine1.eng.company.com:8080/
   ```

   Note

   Coverity Connect supports the `http://` and `https://` protocols.

2. When the sign-in page opens, log into your account using your user name and password.

   If you have difficulty logging in, contact your Coverity Connect administrator. This difficulty could be to due to incorrect user login credentials or that you do not have the expected Role-based Access Control (Section 3.2.3, “Roles and role based access control”) permissions assigned to you.

For browser support and software requirements, see *Coverity 2020.12 Installation and Deployment Guide*.

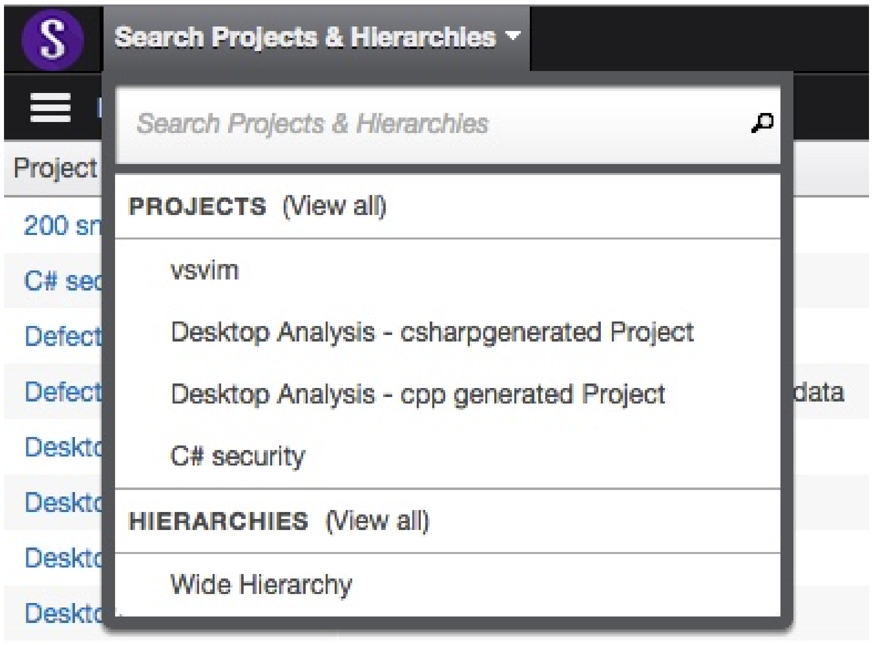
Chapter 1.3. Shared Navigation and Main menu tools

Coverity Connect Projects and Coverity Policy Manager Hierarchies (which can be enabled though your license) share a common high-level navigation UI. For example, Figure 1.3.1, “Example: Navigation to All Hierarchies and All Projects from the Main Menu” shows a list of hierarchies that a user has opened by navigating from *Projects & Hierarchies* in the Main Menu to *All Hierarchies* (in the left-side navigation pane).

Note

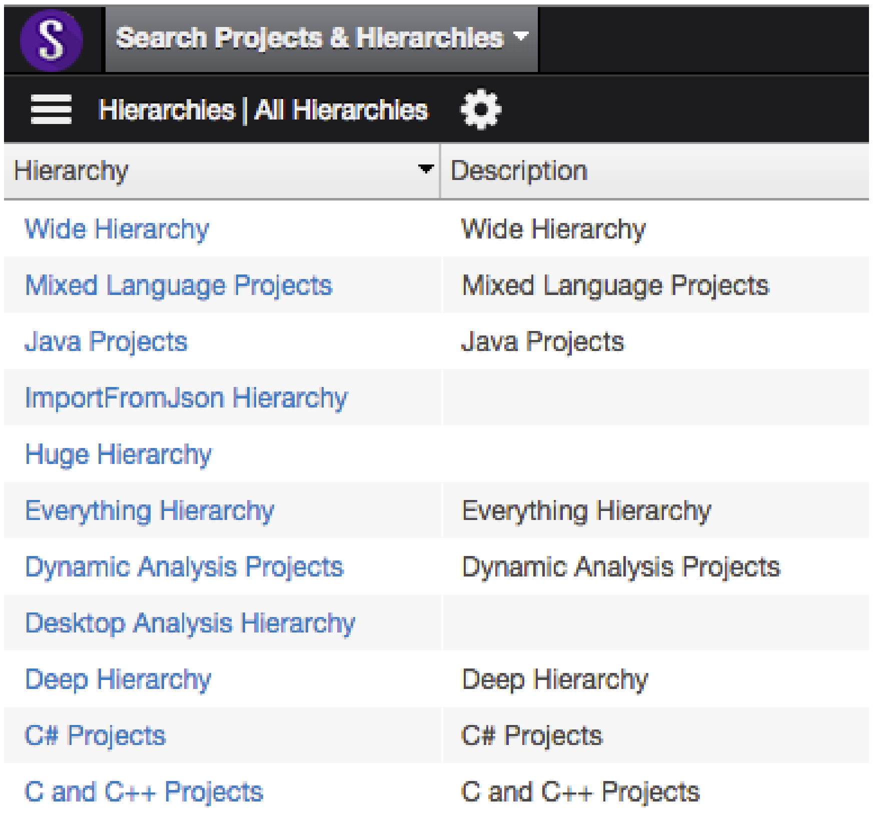
If you see *Projects* alone (not *Projects & Hierarchies*) in the Main Menu, you either need to get permission to view hierarchies from your Coverity Connect administrator (for more details, see Chapter 5.5, *Coverity Policy Manager Roles and Permissions*), or the license is not enabled for Coverity Policy Manager.

Figure 1.3.1. Example: Navigation to All Hierarchies and All Projects from the Main Menu


Search Projects & Hierarchies
Search Projects & Hierarchies
PROJECTS (View all)
vsvim
Desktop Analysis - csharpgenerated Project
Desktop Analysis - cpp generated Project
C# security
HIERARCHIES (View all)
Wide Hierarchy

The *Projects (View All)* and *Hierarchies (View All)* links list projects and/or hierarchies in Coverity Connect.

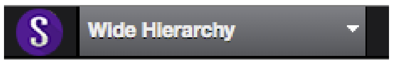
Figure 1.3.2. Example: All Hierarchies



Hierarchy	Description
Wide Hierarchy	Wide Hierarchy
Mixed Language Projects	Mixed Language Projects
Java Projects	Java Projects
ImportFromJson Hierarchy	
Huge Hierarchy	
Everything Hierarchy	Everything Hierarchy
Dynamic Analysis Projects	Dynamic Analysis Projects
Desktop Analysis Hierarchy	
Deep Hierarchy	Deep Hierarchy
C# Projects	C# Projects
C and C++ Projects	C and C++ Projects

As shown next in Figure 1.3.3, “Example: Main Menu: Hierarchy”, the top of the Main menu displays the name of the project or hierarchy that is currently open. If you open a hierarchy, you will see that name in red.

Figure 1.3.3. Example: Main Menu: Hierarchy



The Main menu has the following tools to help you interact with the Coverity Connect/Coverity Policy Manager UI:

Table 1.3.1. Main menu tools

Icon	Description
☰	The *View icon* toggles the View panel (sometimes called the "Report panel" for Coverity Policy Manager). The View panel allows you to navigate and edit your views. For more information, see Section 2.4.1, “Views”.
ⓘ	The *Information icon* displays the View Information dialog that lists information about the currently selected view or hierarchy.

Icon	Description
⚙	The *Gear icon* launches the Settings dialog. For more information, see Section 2.4.2.1, “Edit Settings”.

Note

If you are using Coverity Connect on Mozilla Firefox, you should select the *Allow pages to choose their own fonts, instead of my selections above* option. Otherwise, the icons listed above might not display correctly. This setting is typically found in Firefox under Preferences → Content → Fonts & Colors.

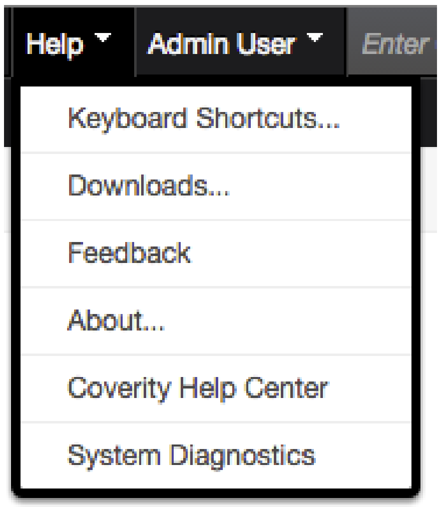
Chapter 1.4. Help Menu

Table of Contents

1.4.1. Keyboard Shortcuts 8
1.4.2. Product downloads 8
1.4.3. Customer feedback for Coverity 9
1.4.4. Coverity Connect platform details 9
1.4.5. Coverity Connect documentation set 9

The *Help* menu in the tool bar provides the options shown in Figure 1.4.1, “Coverity Connect Help menu items”.

Figure 1.4.1. Coverity Connect Help menu items



The following subsections describe the content that is available from each *Help* menu item.

1.4.1. Keyboard Shortcuts

The *Keyboard Shortcuts* link opens a pop-up window that lists useful shortcuts for navigating through Coverity Connect issues and events.

1.4.2. Product downloads

The *Downloads* link (see Figure 1.4.1, “Coverity Connect Help menu items”) provides access to a central location for downloading and/or installing the following:

- Download the Coverity Desktop product packages for Eclipse, QNX Momentics, Wind River Workbench, IBM RTC, Visual Studio, IntelliJ, and Android Studio.
- Eclipse Update site

- Coverity Desktop Analysis Tools
- Gradle Plug-In
- Coverity Desktop Reports, which include the Coverity MISRA Report, Security Report, and Coverity Integrity Report
- Software Development Kits
- Other content, such as scripts or additional documentation. For information about making this content available, see Section 3.12.1, “Configuring Coverity Desktop and shared files through the Downloads page”.

1.4.3. Customer feedback for Coverity

The *Feedback* link (see Figure 1.4.1, “Coverity Connect Help menu items”) opens to a feedback form into which Coverity invites you to send anonymous product feedback.

1.4.4. Coverity Connect platform details

The *About Coverity Connect* link (see Figure 1.4.1, “Coverity Connect Help menu items”) opens to information about the Coverity Connect version, installation directories, database, ports, and so on.

1.4.5. Coverity Connect documentation set

The Coverity Connect Help Center link (see Figure 1.4.1, “Coverity Connect Help menu items”) provides access to the Coverity Connect documentation set.

Part 2. Coverity Connect Usage

Developers, managers, and leads of software development teams use Coverity Connect to understand, manage, and fix issues found by Coverity Analysis and third-party tools.

Administrators set up and manage Coverity Connect for these end users. For guidance, see Part 5, “Coverity Policy Manager Administration”.

Chapter 2.1. Getting Started with Coverity Connect

Coverity Connect allows you to examine and manage issues that Coverity Analysis and advisor products find in your software. Coverity Connect helps you organize, understand, and prioritize these issues across related code bases (for example, multiple development branches for different target platforms) to prevent software failures, security breaches, and untested source code from reaching your customers.

After logging into Coverity Connect (see Chapter 1.2, *Login*), you can perform two major types of tasks:

- Examining and triaging security risks, quality issues, and test violations: See Chapter 2.2, *Managing issues*.
- Monitoring current metrics and trends: See Chapter 2.3, *Monitoring issues*.

For Coverity Connect configuration details, see Part 3, “Coverity Connect Administration ”. For documentation on topics that are mentioned but not thoroughly addressed in this guide (such as the analysis and advisor products, supported platforms, and so on), see Coverity Connect Help Center ↗, which is available through the *Help* menu in the Coverity Connect tool bar.

Navigation
: For guidance with navigation to projects, see Chapter 1.3, *Shared Navigation and Main menu tools*.

Chapter 2.2. Managing issues

Table of Contents

2.2.1. Understanding the primary Coverity Connect workflows 12
2.2.2. Finding issues 13
2.2.3. Managing Issues 14
2.2.4. Triaging issues 17

The Coverity Connect interface allows you to find, examine, and triage many types of issues that can occur in your source code.

Your Coverity Connect administrator uses Coverity Connect projects and streams to organize issues found by Coverity Analysis in your code base. In Coverity Connect, each project contains one or more streams into which the results of an analysis are pushed (committed). Each time a new set of analysis results is committed to a Coverity Connect stream, a new snapshot is created for it in Coverity Connect. So, if an issue that appeared in past snapshots in a given stream has been fixed, that issue will not appear in the latest snapshot.

The following sections describe the various areas of the Coverity Connect window, and provide the general workflow for using Coverity Connect.

2.2.1. Understanding the primary Coverity Connect workflows

This section provides a high-level description of two of the primary workflows that Coverity Connect addresses.

2.2.1.1. Development workflow

This workflow applies to software developers who are primarily interested in understanding the issues that currently exist in the code base. The focus here is on the details of individual issues found in a given analysis (typically the most recent snapshot):

1. Find and organize issues and issue-related data (see Section 2.2.2, “Finding issues”). For the developer workflow, the view type most likely to be used for viewing issues is *Issues: By Snapshot*.
2. Examine the source code in which an issue is found (see Section 2.2.3, “Managing Issues”).
3. Triage issues to specify their importance, assign them an owner, and add other information that you want track with the issue (see Section 2.2.4, “Triaging issues”).

2.2.1.2. Management workflow

This workflow applies to users within a given organization that are primarily interested in viewing all issues within a given stream or group of streams within a given project or component, and not necessarily within a specific snapshot. This is typically used by project leads, team managers, directors, and so forth, to get a gauge on the status (or the history) of a given software project:

1. Find, organize, and view issues within a stream, project, or component (see Section 2.2.2, “Finding issues”). For the management workflow, a helpful view type is *Issues: Project Scope*.

2. Examine the triage states of a group of defects (see Section 2.2.4, “Triaging issues”).

3. View and organize Coverity Connect dashboards and charts (see Chapter 2.3, *Monitoring issues*).

2.2.2. Finding issues

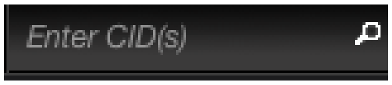
You can search for an issue by its CID or use Coverity Connect views to group issues and issue-related data in ways that make them easy to locate whenever you want to see them.

2.2.2.1. Searching for issues by CID

If you know the CID (or CIDs) that you want to examine, you can use the following feature to display it in Coverity Connect:

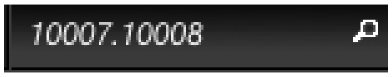
- The *Enter CID(s)* field.

  Figure 2.2.1. Enter CIDs field

  As Figure 2.2.2, “Example: Search for CIDs” shows, you need to use commas to separate multiple CIDs.

  Figure 2.2.2. Example: Search for CIDs

  Note

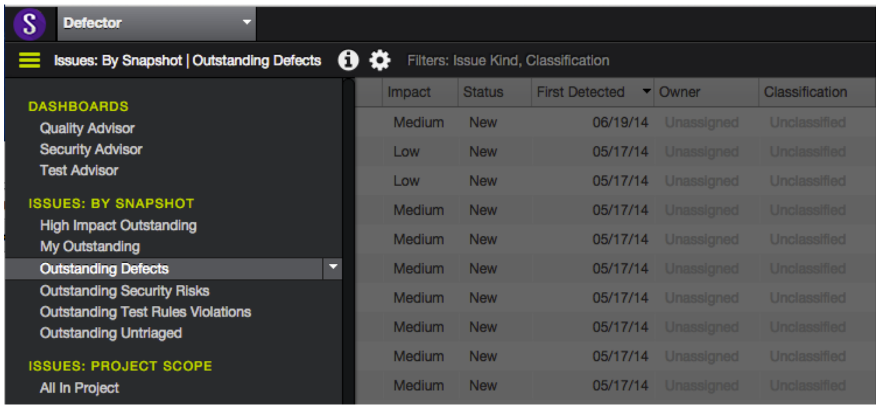
  If the Coverity Connect View pane does not display a CID that you found through the *Enter CID(s)* field, you might need to refresh your browser to see it.

2.2.2.2. Using views to find issues

If you do not know the CID of the issue (or issues) that you want Coverity Connect to display, you can use views to find the issues and issue-related data (for example, file and component data) that you need to display. The example in Figure 2.2.3, “Example: Selecting a View from a View menu” shows some issues that appear in *Defector* after selecting the *High Impact Outstanding* [p. 30] view.

To display the view types and views that are available to you, click on the View icon (☰).

Figure 2.2.3. Example: Selecting a View from a View menu



Impact	Status	First Detected	Owner	Classification
Medium	New	06/19/14	Unassigned	Unclassified
Low	New	05/17/14	Unassigned	Unclassified
Low	New	05/17/14	Unassigned	Unclassified
Medium	New	05/17/14	Unassigned	Unclassified
Medium	New	05/17/14	Unassigned	Unclassified
Medium	New	05/17/14	Unassigned	Unclassified
Medium	New	05/17/14	Unassigned	Unclassified
Medium	New	05/17/14	Unassigned	Unclassified
Medium	New	05/17/14	Unassigned	Unclassified
Medium	New	05/17/14	Unassigned	Unclassified

Note

In the 7.0 release, some views have been removed by default. If you had views (such as All Newly Detected) in a previous version of Coverity Connect and have upgraded to version 7.0, most views will be preserved in the upgraded version.

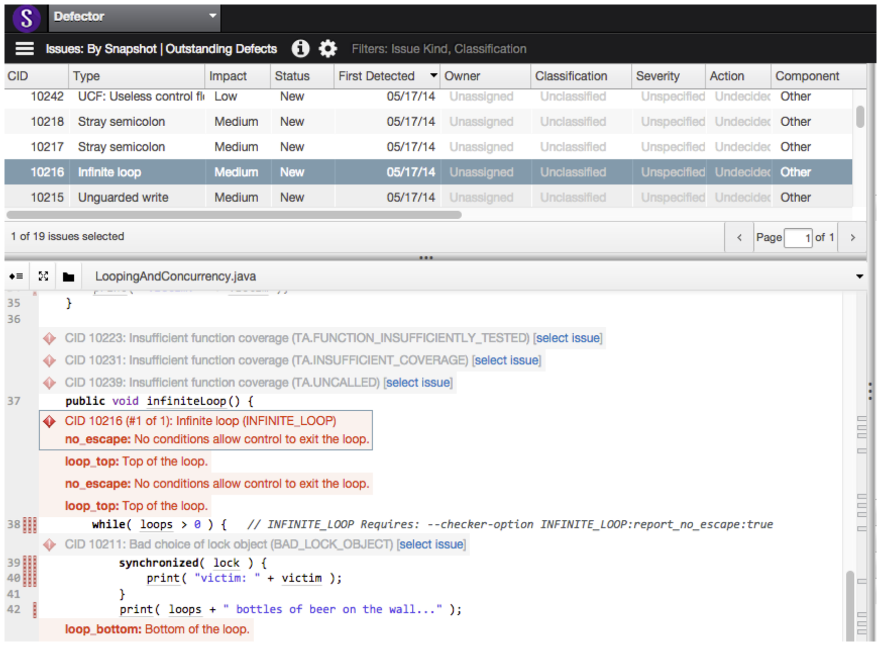
For details about using, creating, modifying, exporting, and performing other tasks with views, see Section 2.4.2, “View Management”. Note that you can rearrange the columns in the View pane and select the columns to display (see Section 2.4.2.1.4, “Columns”).

See also, Chapter 3.8, *Coverity Connect URL construction*.

2.2.3. Managing Issues

Once you find and select a CID that is important to you (see Section 2.2.2, “Finding issues”), the Coverity Connect Source pane shows you where the issue occurs in the source code and adds inline details to help you understand and address it. The example in Figure 2.2.4, “List of CIDs (upper pane) and Source pane (lower pane)” shows CID 10212 in both Coverity Connect panes.

Figure 2.2.4. List of CIDs (upper pane) and Source pane (lower pane)


Defector
Issues: By Snapshot | Outstanding Defects
Filters: Issue Kind, Classification

CID	Type	Impact	Status	First Detected	Owner	Classification	Severity	Action	Component
10242	UCF: Useless control fl	Low	New	05/17/14	Unassigned	Unclassified	Unspecified	Undecidec	Other
10218	Stray semicolon	Medium	New	05/17/14	Unassigned	Unclassified	Unspecified	Undecidec	Other
10217	Stray semicolon	Medium	New	05/17/14	Unassigned	Unclassified	Unspecified	Undecidec	Other
10216	Infinite loop	Medium	New	05/17/14	Unassigned	Unclassified	Unspecified	Undecidec	Other
10215	Unguarded write	Medium	New	05/17/14	Unassigned	Unclassified	Unspecified	Undecidec	Other

1 of 19 issues selected

Page 1 of 1

LoopingAndConcurrency.java

```
    }

CID 10223: Insufficient function coverage (TA.FUNCTION_INSUFFICIENTLY_TESTED) [select issue]
CID 10231: Insufficient function coverage (TA.INSUFFICIENT_COVERAGE) [select issue]
CID 10239: Insufficient function coverage (TA.UNCALLED) [select issue]
    public void infiniteLoop() {
CID 10216 (#1 of 1): Infinite loop (INFINITE_LOOP)
no_escape: No conditions allow control to exit the loop.
loop_top: Top of the loop.
no_escape: No conditions allow control to exit the loop.
loop_top: Top of the loop.
        while( loops > 0 ) {   // INFINITE_LOOP Requires: --checker-option INFINITE_LOOP:report_no_escape:true
CID 10211: Bad choice of lock object (BAD_LOCK_OBJECT) [select issue]
            synchronized( lock ) {
                print( "victim: " + victim );
            }
            print( loops + " bottles of beer on the wall..." );
loop_bottom: Bottom of the loop.
```

In addition to inline commentary, the source code can include the icons described in Table 2.2.1, “Issue markers”.

Table 2.2.1. Issue markers

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

For some issues, the Source pane provides a *More info* link to detailed remediation advice and reference material. For example, all issues pertaining to web application security provide this documentation (which is also available from the Security Reference). To supplement the inline comments and remediation advice, Coverity Connect also provides closely related reference material, including CWE and checker documentation, to help you understand and address the issue. Links to this information are provided in the top of the triage pane (right pane).

Note

You can use the red or green bars located to the right of the vertical scroll bar to quickly jump to the issue-related events in the source code. Coverity Connect allows you to adjust the size of the panes in your browser. You can also show and hide the Triage pane and the pane that contains the View types and dashboards.

You can use the icons located between the View pane and the Source pane to control the display of information and line numbers in the source code and to view the directory structure in which the selected source file resides. To obtain descriptions of the icons, you can mouse over them in Coverity Connect.

Data in the View pane can link to the issues to which the data pertains. For example, see the links in Figure 2.3.7, “Example: View-level column chart”.

2.2.3.1. Show Source Gutter Menu

The *Show Source Gutter Menu* is activated by clicking on the Show gutter control icon - icon. The *Show Source Gutter Menu* allows you to control the display of the following information:

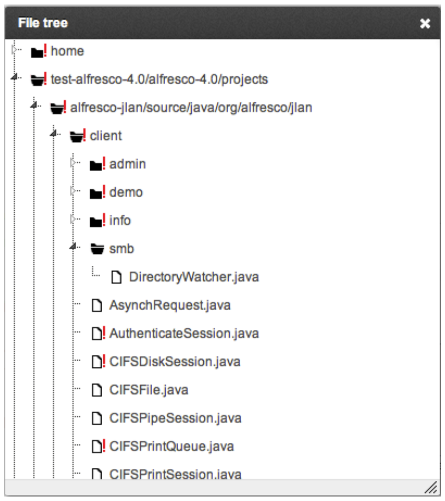
- SCM Author - The username of the user who checked the code in.
- SCM Modification Date - The date that the changed code was checked into an SCM system.
- SCM Revision - The revision number corresponding to the check-in of the changed code. Revision values depend on the SCM system.
- Line Numbers - The line numbers for each line of code.
- Issue Events - The events that lead to the issue.
- Coverage - The lines of code that are covered by analyzed developer tests (used for Test Advisor only).
- Coverage Exclusions - Allows you show, hide, or change on the fly, coverage rules (used for Test Advisor only). For usage details, see the Test Advisor 2020.12 User and Administrator Guide .

In order to see SCM data in Coverity Connect, it is necessary to run the `cov-import-scm` command prior to running `cov-analyze`. For more information, see `cov-import-scm`.

2.2.3.2. File tree

The Coverity Connect Source pane displays the location of the current source file, and provides a file tree for browsing connected source code files and directories. This gives you the ability to browse defects as they relate to individual files and directories, as opposed to using Coverity Connect views. To access the file tree, click on the file icon in the top-right corner of the source pane - .

Figure 2.2.5. Example: File Tree



```
File tree
  ! home
  ! test-alfresco-4.0/alfresco-4.0/projects
    ! alfresco-jlan/source/java/org/alfresco/jlan
      ! client
        ! admin
        ! demo
        ! info
          smb
            DirectoryWatcher.java
          AsynchRequest.java
        ! AuthenticateSession.java
        ! CIFSDiskSession.java
          CIFSFile.java
          CIFSPipeSession.java
        ! CIFSPrintQueue.java
          CIFSPrintSession.java
```

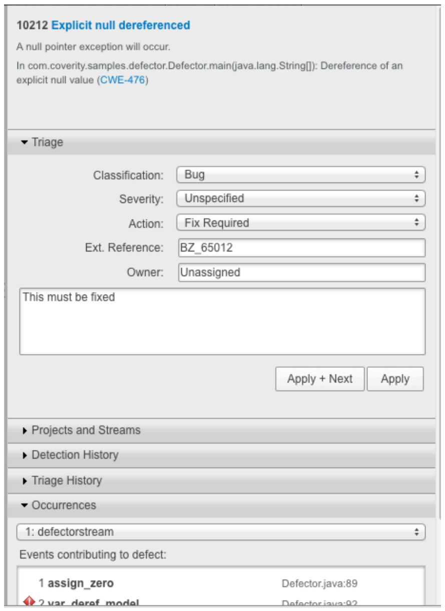
When selected, the file tree will display the various directories and source code files that make up your project. Red exclamation points signify software issues within the given file or directory.

You can display any source file by double-clicking it in the file tree.

2.2.4. Triaging issues

After viewing an issue in the Source pane (see Section 2.2.3, “Managing Issues”), you can triage it by using the Triage pane to modify one or more of its attributes.

Figure 2.2.6. Example: Triage pane



10212 Explicit null dereferenced

A null pointer exception will occur.

In com.coverity.samples.defector.Defector.main(java.lang.String[]): Dereference of an explicit null value (CWE-476)

Triage

Classification:	Bug
Severity:	Unspecified
Action:	Fix Required
Ext. Reference:	BZ_65012
Owner:	Unassigned

This must be fixed

Events contributing to defect:

```
1 assign_zero          Defector.java:89
2 var_deref_model      Defector.java:92
```

Figure 2.2.6, “Example: Triage pane” shows the default attributes and a built-in attribute (*Legacy*, which is not displayed by default):

- Classification (default)
- Severity (default)
- Action (default)
- Legacy Target (built-in attribute)
- External (Ext.) Reference (default)
- Owner (default)
- Comment (default)
- **Deprecated Value** - This selection box allows you display attributes that have been marked as deprecated, or change the attribute value to a non-deprecated value. Administrators can designate an attribute as deprecated in the Configuration - Attributes page.

  If you select this option, you can see the deprecated attribute value, but cannot change the attribute value. If this option is not selected, you can change the attribute to any value (that has not been designated as deprecated).

Note

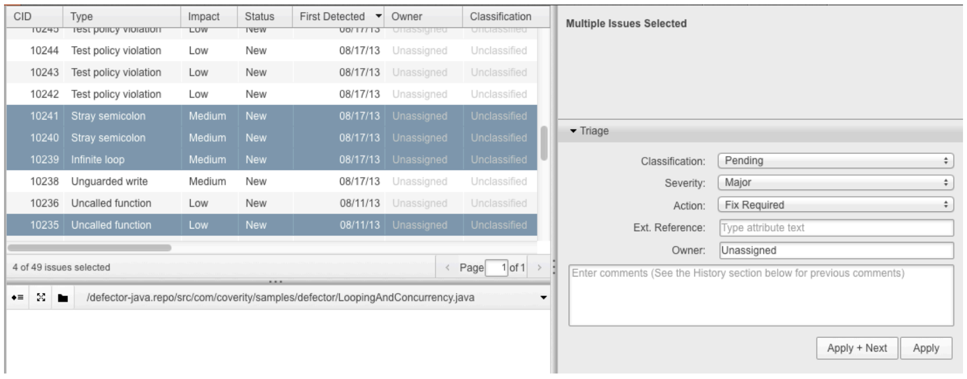
The attributes that you see in Figure 2.2.6, “Example: Triage pane” might not match the attributes you see in your Coverity Connect instance because Coverity Connect administrators can create and modify these attributes.

You can use filters to find issues based on attributes in the Triage pane. For example, you might create a view that finds all issues that are classified as *Bug*. For details about the filter criteria that you can use, see Section 2.4.2.1.2, “Filters”.

Triaging multiple issues

It is possible to triage multiple CIDs at the same time after selecting all of them at once from the View pane, as shown in Figure 2.2.7, “Example: Triaging multiple issues”.

Figure 2.2.7. Example: Triaging multiple issues



CID	Type	Impact	Status	First Detected	Owner	Classification
10245	Test policy violation	Low	New	08/17/13	Unassigned	Unclassified
10244	Test policy violation	Low	New	08/17/13	Unassigned	Unclassified
10243	Test policy violation	Low	New	08/17/13	Unassigned	Unclassified
10242	Test policy violation	Low	New	08/17/13	Unassigned	Unclassified
10241	Stray semicolon	Medium	New	08/17/13	Unassigned	Unclassified
10240	Stray semicolon	Medium	New	08/17/13	Unassigned	Unclassified
10239	Infinite loop	Medium	New	08/17/13	Unassigned	Unclassified
10238	Unguarded write	Medium	New	08/17/13	Unassigned	Unclassified
10236	Uncalled function	Low	New	08/11/13	Unassigned	Unclassified
10235	Uncalled function	Low	New	08/11/13	Unassigned	Unclassified

4 of 49 issues selected

Page 1 of 1

/defector-java.repo/src/com/coverity/samples/defector/LoopingAndConcurrency.java

Multiple Issues Selected

Triage

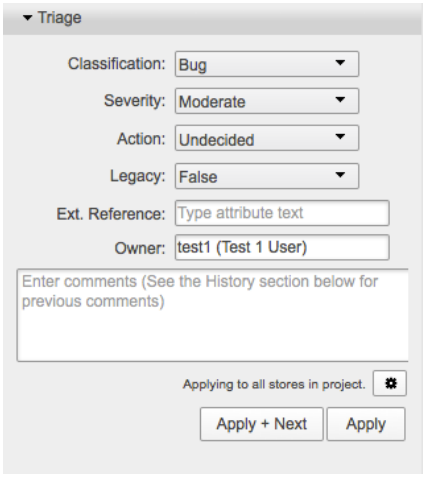
Classification: Pending
Severity: Major
Action: Fix Required
Ext. Reference: Type attribute text
Owner: Unassigned

Enter comments (See the History section below for previous comments)

Selecting triage stores to update

If you can perform advanced triage on a CID, the **Applying to all stores in the project** button will appear below the Comment field in the Triage pane for that CID.

Figure 2.2.8. Select Triage Stores



Triage

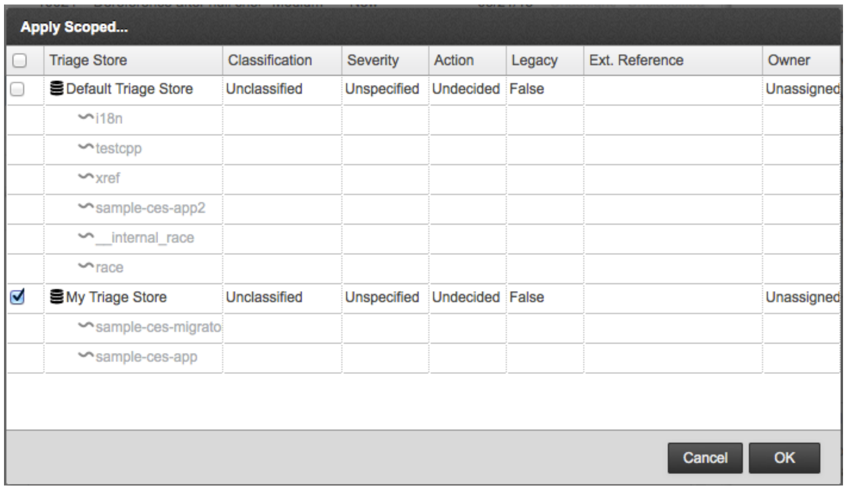
Classification: Bug
Severity: Moderate
Action: Undecided
Legacy: False
Ext. Reference: Type attribute text
Owner: test1 (Test 1 User)

Enter comments (See the History section below for previous comments)

Applying to all stores in project.

By default, Coverity Connect selects all triage stores that contain the CID. The button shown below the Comment field in Figure 2.2.8, “Select Triage Stores” opens to the advanced triage window (*Apply Scoped...* shown below), where you can select or deselect any triage stores that it lists. (Note that if all the streams in the project are associated with the same triage store, or if you only have permission to triage issues in one of the triage stores, the button will not appear because there is only one triage store that you can update.)

Figure 2.2.9. Apply Scoped Triage



Apply Scoped...

☐	Triage Store	Classification	Severity	Action	Legacy	Ext. Reference	Owner
☐	Default Triage Store	Unclassified	Unspecified	Undecided	False		Unassigned
	i18n						
	testcpp						
	xref						
	sample-ces-app2						
	__internal_race						
	race						
✓	My Triage Store	Unclassified	Unspecified	Undecided	False		Unassigned
	sample-ces-migrato						
	sample-ces-app						

When triaging a single issue, you can also examine the following:

CWE documentation

- Links to the Common Weakness Enumeration documentation of the issue. For example, Figure 2.2.6, “Example: Triage pane” includes the CWE-459 link to information on incomplete cleanup.

Projects & Streams section

- Identifies all streams in which the CID occurs in the current project. This section also links to other projects containing the CID.

Detection History

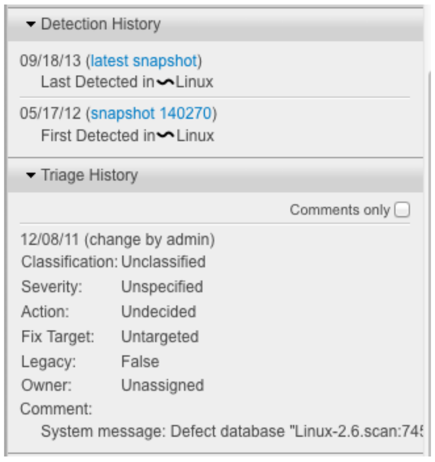
- Provides the history of when and in which snapshot the CID was first ever detected by Coverity analysis tools and last detected by Coverity. It also shows in which stream(s) the CID exists.

Triage History

- Provides information about changes to the triage states of the CID, such as classification, owner, comment, and so forth.

  You can select the *Show only commented entries* option to display only the history of the comments for the given CID.

Figure 2.2.10. History section



▾ Detection History

09/18/13 (latest snapshot)
Last Detected in Linux

05/17/12 (snapshot 140270)
First Detected in Linux

▾ Triage History

Comments only ☐

12/08/11 (change by admin)

Classification:	Unclassified
Severity:	Unspecified
Action:	Undecided
Fix Target:	Untargeted
Legacy:	False
Owner:	Unassigned
Comment:	

System message: Defect database "Linux-2.6.scan:74

Occurrences

- Provides information about each instance of the CID that occurs in the current project. The pull-down menu at the top of this section contains instances in the other streams to which you have access in the project, allowing you to view the occurrence data for each.

  Each CID can have multiple occurrences, and each occurrence (also known as an instance) has a set of events that lead to the issue. See the *Occurrences* section in Figure 2.2.6, “Example: Triage pane”. The example shows one occurrence of CID 10006, with three events. To view event information in the Source pane, click the individual event in this section. If necessary, the source will be reloaded.

  Coverity Connect keeps track of the total number of occurrences, recorded and unrecorded. When committing, Coverity Connect can store up to 15 occurrences of an issue. For example, if a function has a buffer that is overflowing with 20 separate instances, then only 15 are recorded.

  You can use the property `allowAllDefectInstences=true` defined in `cim.properties` to disable the 15 occurrences-per-issue limit, and all occurrences will be recorded. However, be aware that this can lead to a drastic increase in database size. For this reason, the property is false by default.

  Coverity Connect also tracks (internally) the number of merged defects, which are sets of nearly identical instances in a data structure.

  Note

  The number of stored occurrences does not affect the numbers of issues displayed by Coverity Connect. It also does not affect the number that is reported by the report generators, as they include merged defects into their count.

Standard Attributes

- The *Standard Attributes* panel displays standard-attribute value-description pairs associated with the selected CID. Value-description pairs are displayed only for standard-attribute columns that both have a value (other than *None*) and are enabled.

  If no standard-attribute column is enabled, the panel displays *None* for the selected CID (even if the selected CID has standard attributes whose values are not *None*).

  If all enabled standard-attribute columns have a value of *None*, the panel displays *None*.

2.2.4.1. Event-driven triage notifications

The administrator can use event-driven triage notifications to send notifications to issue owners about all triage event changes as they happen. These notifications are filterable and configurable. Even driven notifications do not replace existing notification systems.

This feature is not enabled by default. Once enabled, notifications are sent to Apache Kafka servers provided and configured by the user to consume these events. The user can then subscribe to these notifications using technology provided by Kafka. Users can decide what to do with the notifications they consume: for example, they can use them to send email or instant messenger notifications, or process them in any other way.

Triage events can originate from the following operations:

- Updating triage in the Coverity Connect UI.
- Updating triage using Web Services requests.
- Triage updates at commit time.

Triage events do not originate from importing or exporting a triage store, or from associating triage stores with streams or deassociating triage stores from streams.

For each triaged issue, the triage event is sent to the consumer in the following JSON format:

```
{
    version: 1,
    cid: 10001,
    owner: "Mike",
    email: "mike@example.com",
    urls: ["issue-url-1", "issue-url-2"],
    timestamp: "2019-10-10T17:01:28.111+0000",
    triage-attribute-changes: [
    {
        "triage-attribute-name": "classification",
        "triage-attribute-value": "Pending"
    },
    ...]
}
```

The fields are described in the following table.

Field	**Meaning**
cid	CID of the triaged issue
owner	Owner of the issue
email	Email of the issue owner
urls	List of links with all stream-defects affected by the triage event. The same CID may be present in different streams in different projects; the links will point to all occurrences of such a CID.
timestamp	The time when triage happened
triage-attribute changes	A list of the changed triage attributes and their new values.

The basic workflow for setting up event-driven triage notifications is the following:

1. Set up an Apache Kafka server and set up a `triage-events` topic to receive triage event notifications.

   If this topic is missing triage event notifications will fail.

2. Enable triage event notifications using the `eventstriage.enabled` property in the `cim.properties` file:

   ```
   events.triage.enabled=true
   ```

3. Specify the Kafka servers that have been configured to receive triage events. Use the following syntax:

   ```
   events.triage.kafka.bootstrapServers=<server1>, <server2>, ...
   ```

   Notifications can be lost if the consumer is unavailable to Coverity Connect. Events are delivered on an *at least once* basis. This means that duplicate events might sometimes be delivered by Kafka. The user is responsible for filtering or tolerating duplicate messages.

Note

In a Coverity Connect cluster, only the nodes with properly configured triage events notifications will send the notifications to Kafka. The notification is sent by the node where the triage event happens.

Chapter 2.3. Monitoring issues

Table of Contents

2.3.1. Dashboards 24
2.3.2. View-level Charts 27

The following charts provide insight into the current and evolving state of issues in your code base:

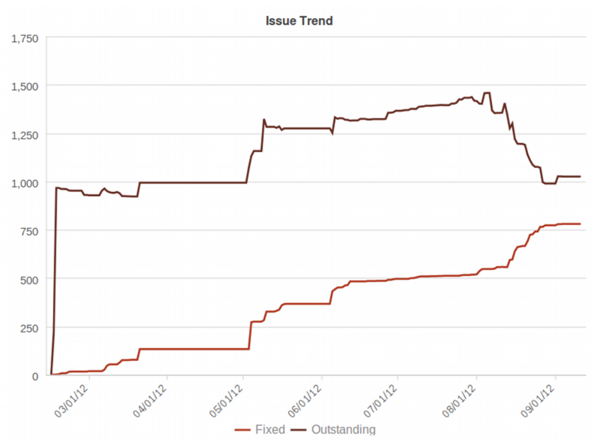
- Dashboards: The *Quality*, *Security*, *Test Advisor* dashboard links open to charts with high-level information about issues in a project. The links are located in the top-left area of the Coverity Connect window (see Figure 2.2.3, "Example: Selecting a View from a View menu").
- View-level Charts: Some Coverity Connect views contain charts that provide more granular information about issues in a given view.

2.3.1. Dashboards

This section provides examples of the Quality and Security charts that appear in Coverity Connect dashboards. The Quality charts graph data on issues found with quality-related checkers, while the Security charts graph data that is related to security-related checkers. (For information about the Test Advisor dashboards, see the *Test Advisor 2020.12 User and Administrator Guide*, available through the Coverity Connect Help Center.)

Issue Trend chart
: Shows the number of fixed and outstanding issues (CIDs) over time, in successive snapshots.

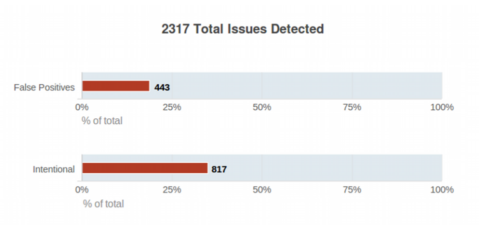
Figure 2.3.1. Example: Issue Trend chart



Total Issues Detected charts
: Identify the total number of issues detected in the project and the percentage of the detected issues that meet the following criteria:

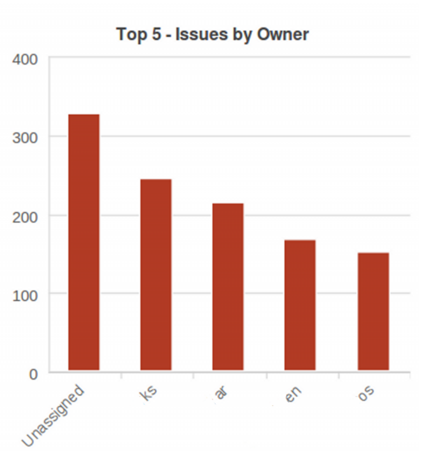
- False Positives: Percentage of the issues that have been marked by developers as *False Positive* in the Triage pane.
- Intentional: Percentage of the issues that have been marked by developers as *Intentional* in the Triage pane.

Figure 2.3.2. Example: Total Issues Detected chart



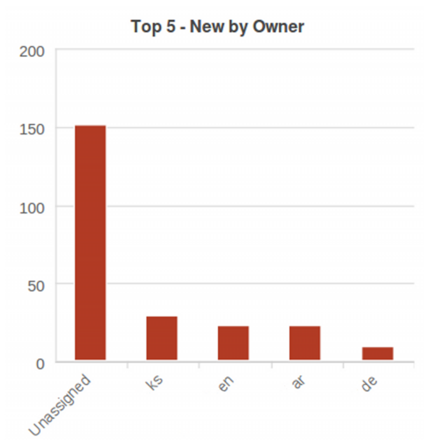
Top 5 - Issues by Owner
: Lists the total number of issues by owner. Only displays the top five.

Figure 2.3.3. Example: Issues by Owner chart



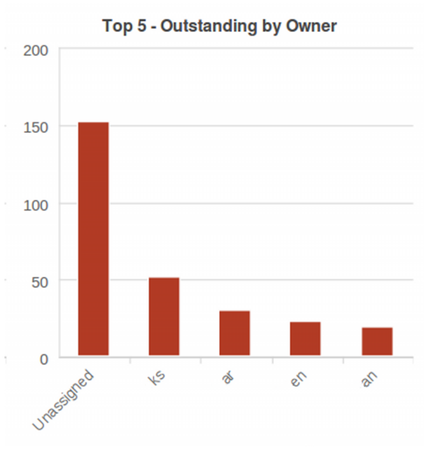
Top 5 - New by Owner
: Lists the number of new (uninspected) issues by owner. Only displays the top five.

Figure 2.3.4. Example: New Issues by Owner chart



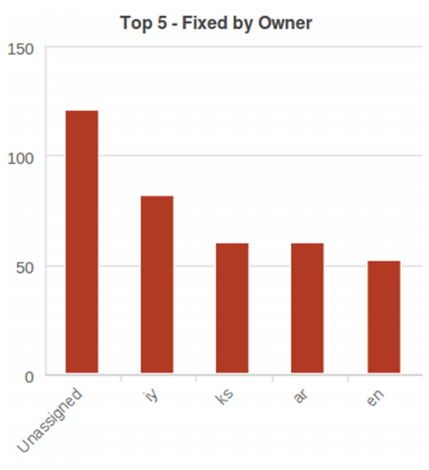
Top 5 - Outstanding by Owner
: Lists the number of unresolved (outstanding) issues by owner. Only displays the top five.

Figure 2.3.5. Example: Outstanding Issues by Owner chart



Top 5 - Fixed by Owner
: Lists the number of fixed issues by owner. Only displays the top five. The chart does not include resolved issues.

Figure 2.3.6. Example: Fixed by Owner chart

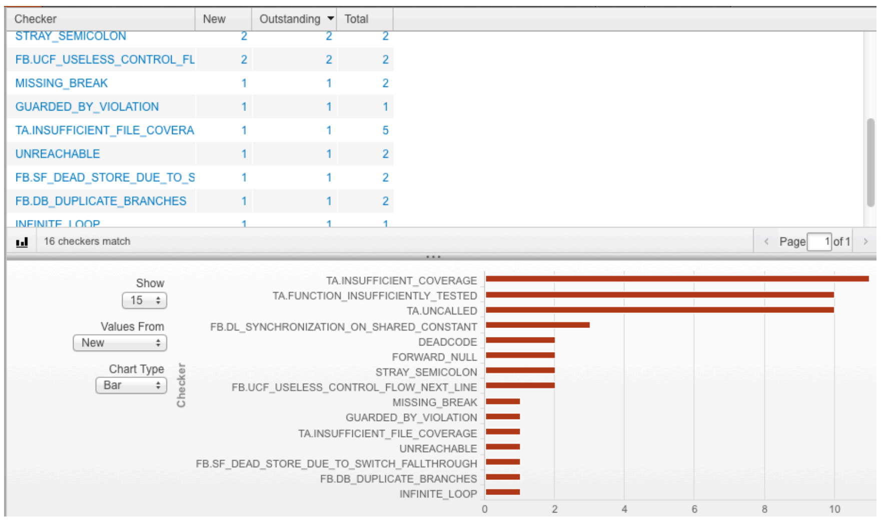


2.3.2. View-level Charts

Views belonging to the Component, Checkers, Owners, and Snapshots view types can display charts that graph view data. For a list of View types, see Section 2.4.1.1, “View Types”.

Figure 2.3.7, “Example: View-level column chart” shows a chart for the *All In Project* view of the *Checkers* View type.

Figure 2.3.7. Example: View-level column chart



Checker	New	Outstanding	Total
STRAY_SEMICOLON	2	2	2
FB.UCF_USELESS_CONTROL_FL	2	2	2
MISSING_BREAK	1	1	2
GUARDED_BY_VIOLATION	1	1	1
TA.INSUFFICIENT_FILE_COVERA	1	1	5
UNREACHABLE	1	1	2
FB.SF_DEAD_STORE_DUE_TO_S	1	1	2
FB.DB_DUPLICATE_BRANCHES	1	1	2
INFINITE_LOOP	1	1	1

16 checkers match

Show: 15
Values From: New
Chart Type: Bar

Page 1 of 1
Checker
TA.INSUFFICIENT_COVERAGE
TA.FUNCTION_INSUFFICIENTLY_TESTED
TA.UNCALLED
FB.DL_SYNCHRONIZATION_ON_SHARED_CONSTANT
DEADCODE
FORWARD_NULL
STRAY_SEMICOLON
FB.UCF_USELESS_CONTROL_FLOW_NEXT_LINE
MISSING_BREAK
GUARDED_BY_VIOLATION
TA.INSUFFICIENT_FILE_COVERAGE
UNREACHABLE
FB.SF_DEAD_STORE_DUE_TO_SWITCH_FALLTHROUGH
FB.DB_DUPLICATE_BRANCHES
INFINITE_LOOP
0
2
4
6
8
10

These fields allow you to filter view data:

- Show: For selecting the number of items to display in the chart. Because there are a total of 16 matching checkers listed in the View pane, Figure 2.3.7, “Example: View-level column chart” selects *15* to display (instead of using other options in the menu, such as `10` or `20`.).

- *Values From* menu: For filtering by the type of view data that you want to display in the chart. For example, Figure 2.3.7, “Example: View-level column chart” displays only the *New* issues in the chart (instead of filtering by other options in this menu: *Outstanding* issues, *Resolved* issues, the *Total* number of issues for a each function, or the *Line Count* of each function).

  Values From menus include countable values that pertain to the view filters. For example, the chart for the *High Issue Density (> 1)* view of the *Components* View type includes an *Issue Density* value in its *Values From* menu because the view uses the *Issue Density* filter to set a density > *1*, which is a countable value.

  Note that some items, such as a given function name, can occur in numerous source code locations, and more than one instance of the function can manifest a separate issue (CID). For this reason, it is possible to discover, for example, when there is more than one new (untriaged) issue for a given function.

- *Chart Type* menu: For displaying chart data in vertical columns (see Figure 2.3.7, “Example: View-level column chart”) or horizontal bars.

Chapter 2.4. User Reference

Table of Contents

2.4.1. Views 29
2.4.2. View Management 34
2.4.3. Snapshot comparison 50
2.4.4. View-specific email notifications 54
2.4.5. Triage attributes 57
2.4.6. Preferences 60

Reference material in the following sections supports documentation in the main sections of this guide.

2.4.1. Views

Coverity Connect allows you to organize your CIDs and other data by using saved search criteria (a view) for data in a Coverity Connect project. You can create a view and use *Edit Settings..* options to edit, copy, delete, and share views. You can also use the View menu to obtain the URL of views and to export the current content of a given view to CSV or XML.

To display the view types and views that are available to you, click on the View icon - ☰.

2.4.1.1. View Types

Coverity Connect groups CIDs and other data into views that belong to the following View types: *Issues:By Snapshot*, *Issues: Project*, *Scope*, *Files*, *Functions*, *Components*, *Checkers*, *Owners*, *Snapshots*, *Trends*, *Tests*. Different view types offer different sets of display columns and filters. For a listing of the display columns and filters available to each type of view, see Table 2.4.1, "Filters." The subsections that follow describe the default views for each View type and the default filter settings for each view.

For information about managing and customizing views, see Section 2.4.2, “View Management”.

Note

In the 7.0 release, several default views were removed. If you had views in a previous version of Coverity Connect and have upgraded to version 2020.12, your views, and their settings, will be preserved in the upgraded version.

2.4.1.1.1. Issues: By Snapshot

Use *Issues: By Snapshot* views to review issues that need to be triaged and addressed in current versions of your source code. Use the snapshot scope to refine the time frame of the issues shown.

- This view type allows you to see CIDs in one (or more) snapshot. It also allows you to view all occurrences of an issue. This view type is designed for users whose primary tasks fall under the Coverity Connect "developer workflow". By default, the views in this view type display the filtered CIDs that occurred in the most recent snapshot -- and typically, developers are interested in the state of CIDs in the most recent snapshot.

Note

It is possible to change the scope to show and/or compare a series of snapshots, snapshots that occur in different streams, and so forth. For more information, see Section 2.4.3, “Snapshot comparison”.

Coverity Connect provides the following *Issues: By Snapshot* views:

- *High Impact Outstanding*: Classification is set to *Unclassified*, *Pending*, *Bug*, *untested* AND the *High Impact Outstanding*: *Impact* filter is set to *High*.
- *My Outstanding*: *Status* filter is set to *New* and *Triaged*, and the *Owner* is set to your user name.
- *Outstanding Defects*: *Issue Kind* filter is set to *Quality*, and the *Status* filter is set to *New* and *Triaged*.
- *Outstanding Defects Count*: Show the total of security and non-security defects.
- *Outstanding Non-security Defects Count*: Show the total of non-security defects.
- *Outstanding Security Defects Count*: Show the total of security defects.
- *Outstanding Security Risks*: *Issue Kind* filter is set to *Security*, and the *Status* filter is set to *New* and *Triaged*.
- *Outstanding Test Rule Violations*: *Issue Kind* filter is set to *Test Violation*, and the *Status* filter is set to *New* and *Triaged*.
- *Outstanding Untriaged*: *Status* filter is set to *New*.
- *Unsaved view*: While this view is not a default view "out of the box", it is the view that is displayed by default from a view type (other than Issues: By Snapshot) from which you derive a list of issues. For example, when you select a component from a *Components* view, the *Unsaved view* displays the issues in that component. To create a new view from an unsaved view, select the **Save as Copy** option and rename, apply filters, and save the copied view.

You can also view issue information in occurrences mode. An *occurrence* is an instance of an issue, and the issue itself represents a set of occurences of the same defect. Multiple occurrences might happen when there are multiple findings of an issue in the same file, or when a project that uses multiple streams/triage stores has an occurrence of a merged defect for each stream despite it being the same issue. In the latter case, it is possible for any of the triage columns' cells to contain a variety of values. In this case, the string "Various" will be displayed in issues mode, but will display the correct line number in occurrences mode.

There are times when you need to look at a specific occurrence. An occurrence is associated with a file, a line number, and a triage comment. Note that column values are exactly the same in both Issues mode and Occurrences mode for both a merged defect and its occurrences for every column except line number.

To view snaphot issues in **Occurences** mode:

1. Click on the settings (gear) icon.
2. In the **Filters**, **Columns**, or **Snapshot Scope** tab of the **Settings** dialog box, click the box **Show Occurrences**, and click **OK**.

With the occurences view enabled, the **Issues: By Snapshot** display will show all occurrences of a defect. As a result the issues grid will display one row for each occurrence; occurrences belonging to the same merged defect/issue will have the same CID. Because triage cannot be changed for occurences, the Triage panel will be read-only in this mode.

- **Line number**, to display the line number of the occurrence in the source file.
- **Last triage comment**, to display the last triage comment for the issue that aggregates this occurrence.

2.4.1.1.2. Issues: Project Scope

Use *Issues: Project Scope* views to get a historical overview of all the issues that have ever been detected in the selected project, including those that have been fixed. Double clicking on an issue will lead to an *Issues: By Snapshot* view so that the source may be viewed.

- This view type is designed for users whose primary tasks fall under the Coverity Connect "management workflow". The scope of the *All In Project* view is set so that all CIDs in the current project are returned. You cannot edit the scope in views of this view type, however you can edit all of the other settings or create a new view with different filter settings.

2.4.1.1.3. Files

Use *File* views to look at the files in the selected project. Click through different values on the top right of the *Triage* panel to view specific issues associated with a file.

- By default, Coverity Connect provides the following *Files* views:
  - *In Latest Snapshot*: This view is not filtered.
  - *Uncovered by Tests*: *Raw Coverage* filter is set to `>0`, and the *Raw Covered Lines* filter is set to `>0`.

  When you select a file from the list of files, the file opens in the Source browser. The right pane displays the number of (and a link to) the outstanding issues in that file, as well as the metrics for the file.

2.4.1.1.4. Functions

Use *Function* views to look at the functions and methods in the selected project. Click through different values on the top right of the *Triage* panel to view specific issues associated with a function.

- By default, Coverity Connect provides the following *Functions* views:
  - *High CCM (>15)*: *CCM* filter is set to `>15`.
  - *In Latest Snapshot*: This view is not filtered.

- *Uncovered by Tests*: *Raw Coverage* filter is set to `>0`, and the *Raw Covered Lines* filter is set to `>0`.

  When you select a function from the list of functions, the function opens in the Source browser. The right pane displays the number of (and a link to) the outstanding issues in that function, as well as the metrics for the file.

2.4.1.1.5. Components

Use *Component* views to look at the components in the selected project. Click through different column values to view specific issues associated with a component.

- By default, Coverity Connect provides the following *Components* views:
  - *All in Project*: This view is not filtered.
  - *High Issue Density (>1)*: *Issue Density* filter is set to `>1`.
  - *With Outstanding Issues*: *Outstanding* filter is set to `>0`.
  - *With Untriaged Issues*: *New* filter is set to `>0`.

2.4.1.1.6. Checkers

Grouping an *Issues: By Snapshot* view by Checker is the encouraged workflow. *Checker* views can be used to look at the number of issues found by specific checkers in the selected project.

- By default, Coverity Connect provides the following *Checkers* view:
  - *All in Project*: This view is not filtered.

2.4.1.1.7. Owners

Grouping an *Issues: By Snapshot* view by Owner is the encouraged workflow. *Owner* views can be used to look at the number of issues owned by specific owners in the selected project.

- By default, Coverity Connect provides the following *Owners* view:
  - *All in Project*: This view is not filtered.

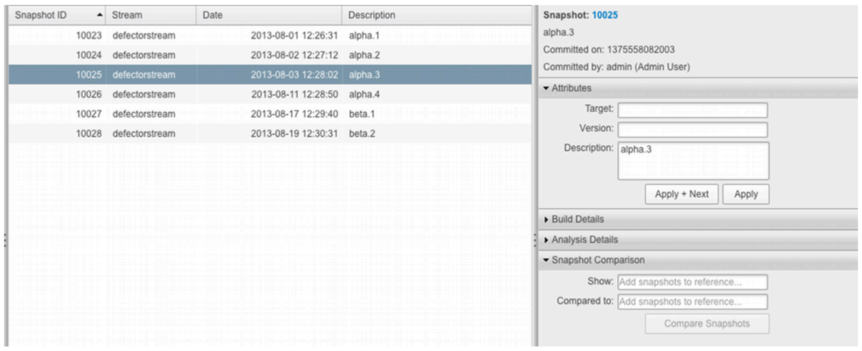
2.4.1.1.8. Snapshots

Use *Snapshot* views to look at the snapshots that belong to the selected project. Click through different values on the top right of the *Triage* panel to view specific issues associated with a snapshot.

- By default, Coverity Connect provides the following *Snapshots* view:
  - *All in Project*: This view is not filtered.

  When a snapshot is selected, users with appropriate permissions can access and use the following tools in the right pane:

Figure 2.4.1. Snapshot view panel



Snapshot ID	Stream	Date	Description
10023	defectorstream	2013-08-01 12:26:31	alpha.1
10024	defectorstream	2013-08-02 12:27:12	alpha.2
10025	defectorstream	2013-08-03 12:28:02	alpha.3
10026	defectorstream	2013-08-11 12:28:50	alpha.4
10027	defectorstream	2013-08-17 12:29:40	beta.1
10028	defectorstream	2013-08-19 12:30:31	beta.2

Snapshot: 10025
alpha.3
Committed on: 1375558082003
Committed by: admin (Admin User)

Attributes
Target:
Version:
Description: alpha.3

Show: Add snapshots to reference...
Compared to: Add snapshots to reference...

Snapshot information
: The snapshot ID links to a view of the issues in that snapshot in an *Unsaved view* [p. 30] in *Issues: By Snapshot*. The commit date and the username of the committer is also displayed.

Snapshot comparison
: Filters snapshots based on show and comparison scope. See Section 2.4.3, “Snapshot comparison”.

Attributes
: Defines attribute values for *Target*, *Version*, and *Description*, for which you can construct filters.

Build Details
: Provides the build information for the snapshot. See Table 3.3.2, “Build Details”.

Analysis Details
: Provides the analysis information for the snapshot. See Table 3.3.3, “Analysis Details”.

2.4.1.1.9. Trends

Use *Trend* views to look at the trends over time associated with the selected project. Adding columns will add data to the graph. Click through different column values to view specific issues.

- By default, Coverity Connect provides the following *Trends* view:
  - *Project Lifetime*: This view is not filtered.

2.4.1.1.10. Tests

Use *Test* views to see a list of tests that were run under Test Advisor instrumentation. If test separation is not enabled, only a single test called "default" will be shown. If the test source code for a given test was captured, it will be shown when the test is selected.

- By default, Coverity Connect provides the following *Tests* views:
  - *All Tests*: This view is not filtered.

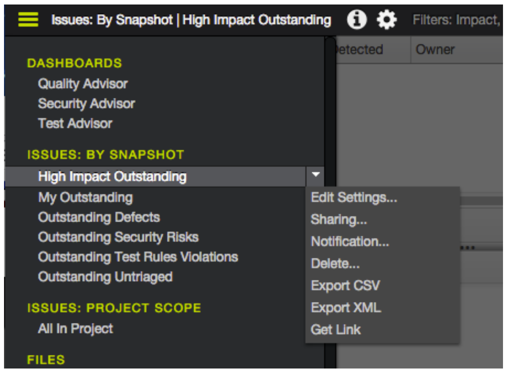
- *Currently Failing*: *State* filter is set to *Failing*.

2.4.2. View Management

Each view provides a menu that allows you to edit, copy, delete, share, receive issue notifications (specifically, notifications from an Issues: By Snapshot view only), and obtain the URL of the view. You can also export the content of the view to CSV or XML. You gain access to this menu by mousing over the view and clicking the menu selector or by clicking the gear icon in the main menu. Figure 2.4.2, “View Menu Options” shows all the options that are available.

To display a history of your saved or edited views, use the Back button on your browser and select the view to which you want to display. Deleted views are not displayed in the history.

Figure 2.4.2. View Menu Options

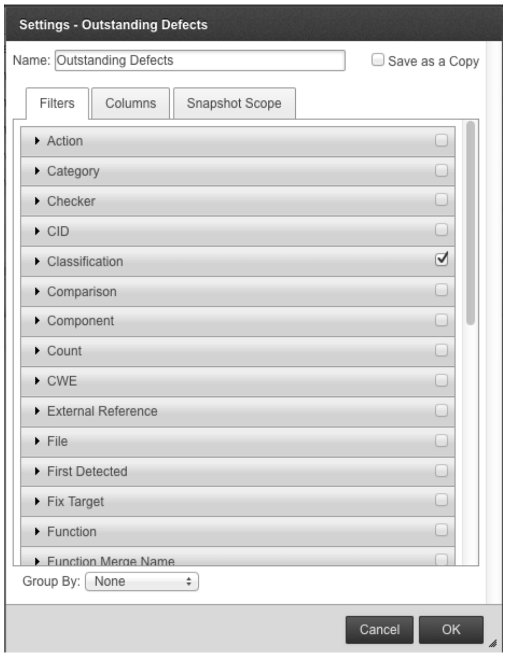


The options are available to all views. For additional view options that administrators can configure for new users, see Chapter 3.10, *Suppressing built-in views for new users*.

2.4.2.1. Edit Settings

The *Edit Settings* menu option allows you to edit and/or or duplicate a complete view within one window. This includes applying filters, selecting display columns, creating a snapshot scope, and so forth. It is displayed by selecting the Gear icon (⚙) in the Main menu, or by selecting the menu name in the pulldown next to the view's name in the View panel.

Figure 2.4.3. Edit Settings window



2.4.2.1.1. Save as a Copy

This option allows you to create a copy of a view. This feature helps save time when you want to create a new view that is similar to the view that you are copying. By renaming and making a few modifications to the filters, you can create the view with minimal effort. See also, Creating a view.

2.4.2.1.2. Filters

You use filters to specify the scope of the issues and issue-related data that you want to display. Filters vary according to the View type (for example, *Issues: By Snapshot*, *Functions*, or *Snapshots*) that you are using.

By default, Coverity Connect displays some filter data in columns of the same name. For example, many View types display a *Total* column in their views, by default. You can select other columns to display them in the view (see Section 2.4.2.1.4, “Columns”).

Table 2.4.1. Filters

Filter	Description	View Types
Action	Recommended action. See Section 2.4.5.3, “Action”.	Issues: By Snapshot, Issues: Project Scope
Acyclic Path Count	Number of execution paths through the function.	Functions

Filter	Description	View Types
Analysis Time	Time that it took to analyze the code that underlies the snapshot. Example: `01:16:46`	Snapshots
Backedge Count	Number of back edges in the control flow graph.	Functions
Blank Lines	Number of blank lines in the source code.	Files, Components, Snapshots, Trends
Block Count	Number of blocks in the control flow graph.	Functions
Build Time	Time that it took to build the source code that underlies the snapshot.	Snapshots
Category	Issue category.	Issues: By Snapshot, Issues: Project Scope, Checkers
CCM	Cyclomatic complexity metric.	Functions
Checker	Name of the checker that discovered the issue. You can quickly select all filters by enabling the **Select All** field.	Issues: By Snapshot, Issues: Project Scope, Checkers
CID	The CID of the issue.	Issues: By Snapshot, Issues: Project Scope
Classification	Classification of the issue. See Section 2.4.5.1, “Classification”.	Issues: By Snapshot, Issues: Project Scope
Code Lines (LOC)	Number of lines of code in the project.[a]	Files, Components, Snapshots, Trends
Comment Lines	Number of lines in the code that contain comments. Uncommented code or code that has too few comments can lead to issues when other developers attempt to modify it.	Files, Components, Snapshots, Trends
Comparison	Comparison indicates whether an issue is present in the snapshot(s) specified in the view’s field in the comparison scope. You can choose from one of the following options: • Present - The CID exists in at least one of the comparison snapshots.	Issues: By Snapshot, Issues: Project Scope

Filter	**Description**	**View Types**
	• Absent - The CID does not exist in any of the comparison snapshots.	
Component	Component name. For example, the component in which the issue, file, or function is found.	Issues: By Snapshot, Issues: Project Scope, Files, Functions, Components
Count	Number of issue occurrences. For example, see the *Occurrences* tab in Figure 2.2.6, "Example: Triage pane".	Issues: By Snapshot
CWE	Common Weakness Enumeration (CWE) identifier for software weaknesses, which include issues such as resource leaks, cross-site scripting vulnerabilities (XSS), null pointer dereferences, and so on.	Issues: By Snapshot, Issues: Project Scope
Date	Date when the snapshot was committed to Coverity Connect. Date of the trend record. Example: `2012-09-09 20:09:19.136`	Snapshots, Trends
Description	Snapshot description.	Snapshots
Dismissed	Number of dismissed issues.	Files, Functions, Components, Checkers, Owners, Trends
Duration(ms)	Duration of test in milliseconds. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See "Viewing test status" in the *Test Advisor 2020.12 User and Administrator Guide*.	Tests
External Reference	An internal identifier used by your company to track the issue. See Section 2.4.5.7, "Ext. Reference".	Issues: By Snapshot, Issues: Project Scope
File	Name of the file in which the issue, file, or function is found.	Issues: By Snapshot, Issues: Project Scope, Files, Functions
File Count	Number of files in the snapshot.	Snapshots
First Detected	Date when the issue was first detected.	Issues: By Snapshot, Issues: Project Scope
First Detected By	Value that helps identify the process by which an issue was initially reported to Coverity Connect: Snapshot (for issues initially reported	Issues: Project Scope

Filter	Description	View Types
	through a commit process that yields a snapshot), Preview (for issues initially reported through a preview process, which does not produce a snapshot, for example, when Coverity Desktop invokes `cov-run-desktop`), API (for issues initially reported through a special, rarely used process). In each case, a CID for the issue is created. **Note** Preview issues that a developer fixes before pushing code changes to the source code repository will never have (or need) a snapshot. Preview issues left unfixed before they are pushed to the repository will typically undergo the server-based analysis and commit process. Therefore, these issues will receive a snapshot in Coverity Connect *after* they were initially reported, and it will be possible to triage the associated CIDs and see events related to them in the source code browser (from an Issues: By Snapshot view that lists the CID). Whether fixed or left unfixed prior to the push to the source code repository, issues will be be identified as Preview issues if they were initially reported through a preview process.	
First Snapshot (column only - not a filter)	ID of the snapshot in which the issue was first detected.	Issues: By Snapshot, Issues: Project Scope
First Snapshot Date	Specified range of dates in which one or more issues was first committed.	Issues: By Snapshot, Issues: Project Scope
First Snapshot Description (column only - not a filter)	Description of the snapshot in which the issue was first detected.	Issues: By Snapshot, Issues: Project Scope
First Snapshot Stream (column only - not a filter)	Stream in which the issue was first detected.	Issues: By Snapshot, Issues: Project Scope
First Snapshot Target (column only - not a filter)	Target platform of the snapshot in which the issue was first detected.	Issues: By Snapshot, Issues: Project Scope
First Snapshot Version	Version number of the snapshot in which the issue was first detected.	Issues: By Snapshot,

Filter	Description	View Types
(column only - not a filter)		Issues: Project Scope
Fix Target	Target milestone for fixing an issue, such as a release or version. See Section 2.4.5.5, “Fix Target”.	Issues: By Snapshot, Issues: Project Scope
Fixed	Number of fixed issues.	Files, Components, Checkers, Owners, Trends
Forwardedge Count	Number of forward edges in the control flow graph.	Functions
Function	The name of the function that contains the issue.	Issues:By Snapshot, Functions
Function Count	Number of functions in the source code that underlies the snapshot.	Snapshots
Function Merge Name[b]	Internal function name used as one of the criteria for merging separate occurrences of the same software issue, with the result that they are identified by the same CID.	Issues: By Snapshot
Halstead Effort	Estimated effort required to modify a program based on empirical findings by Halstead. In general, the less effort required, the easier it should be to modify the program.	Functions
Halstead Errors	Estimate that relates program errors to the number of operands used. In general, the more operands used, the more prone to error the program is. Estimate is based on empirical findings by Halstead.	Functions
Has analysis summaries	Indicates if analysis summaries are included in the snapshot.	Snapshots
ID	Snapshot ID.	Snapshots
Impact	Issue impact as determined by Coverity Connect: High, Medium, Low, Audit.	Issues: By Snapshot, Issues: Project Scope, Checkers
Inspected	Number of inspected issues.	Trends
Issue Density	Number of unresolved (outstanding) issues per 1000 lines of code.	Files, Components, Trends
Issue Kind	Quality, Security, Test, or Various issue.	Issues: By Snapshot,

Filter	**Description**	**View Types**
		Issues: Project Scope
Language	Programming language associated with the defect.	Issues: By Snapshot, Issues: Project Scope, Files, Functions
Last Failure[c]	Date when test last failed. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Tests
Last Impacted[c]	Date of the most recent snapshot in which the function was either changed directly by a developer (resulting in a modification of its syntactic structure) or affected by a direct change elsewhere in the code base that affects the behavior of the function. Examples include changes to the items called by the function or to the global variables used by the function. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Functions
Last Modified[c]	Date of the most recent snapshot in which the function was modified. Last modification date is computed from SCM data and is taken to be the most recent date on which any of the source code lines belonging to the function were modified. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Functions
Last Run[c]	Date when test was last run. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Tests
Last Snapshot (column only - not a filter)	ID of the snapshot in which the issue was last detected.	Issues: By Snapshot, Issues: Project Scope
Last Snapshot Date [c]	Specified range of dates in which one or more issues was last detected..	Issues: By Snapshot, Issues: Project Scope
Last Snapshot Description (column only - not a filter)	Description of the snapshot in which the issue was last detected.	Issues: By Snapshot, Issues: Project Scope
Last Snapshot Stream (column only - not a filter)	Stream in which the issue was last detected.	Issues: By Snapshot, Issues: Project Scope

Filter	**Description**	**View Types**
Last Snapshot Target (column only - not a filter)	Target platform of the snapshot in which the issue was last detected.	Issues: By Snapshot, Issues: Project Scope
Last Snapshot Version (column only - not a filter)	Version number of the snapshot in which the issue was last detected.	Issues: By Snapshot, Issues: Project Scope
Last Success[c]	Date when test last passed. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Tests
Last Triaged[c]	Date when the issue was most recently triaged.	Issues: By Snapshot
Legacy	Searches for issues by the Legacy attribute. Options are: • False • True • Various	Issues: By Snapshot, Issues: Project Scope
Line Count	Number of lines of code.	Functions
Merge Extra[b]	Internal property used as one of the criteria for merging occurrences of an issue.	Issues: By Snapshot
Merge Key[b]	Internal signature used to merge separate occurrences of the same software issue and identify them all by the same CID.	Issues: By Snapshot
MISRA Category	Number of MISRA issues that fall into these selected categories: • Mandatory • Required • Advisory • None	Issues: By Snapshot, Issues: Project Scope
Name	Test name. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*..	Tests
New	Number of new issues.	Files, Functions, Components, Checkers, Owners, Trends

Filter	Description	View Types
Newly Detected	Issues that were not in the previous snapshot.	Components, Checkers, Snapshots
Newly Eliminated	Issues from the previous snapshot that are no longer present.	Components, Checkers, Snapshots
Operand Count	Number of operands in the program.	Functions
Outstanding	Number of unresolved (outstanding) issues.	Files, Functions, Components, Checkers, Owners, Trends
Owner	User or group name of the issue owner.[d]	Issues: By Snapshot, Issues: Project Scope, Owners
Owner Name	Name of the issue owner.	Issues: By Snapshot, Issues: Project Scope, Owners
Policy Coverage	Policy Coverage. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Files, Functions, Components
Policy Covered Lines	Number of lines covered by tests according to your Test Advisor policy. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Files, Functions, Components
Policy Uncovered Lines	Number of lines not covered by tests according to your Test Advisor policy. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Files, Functions, Components
Raw Coverage	Raw coverage. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Files, Functions, Components
Raw Covered Lines	Number of lines covered by tests as reported by the coverage tool. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Files, Functions, Components

Filter	Description	View Types
Raw Uncovered Lines	Number of lines not covered by tests as reported by the coverage tool. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Files, Functions, Components
Resolved	Number of dismissed and fixed issues. See resolved issues.	Trends
Severity	Severity of the issue. See Section 2.4.5.2, “Severity”.	Issues: By Snapshot, Issues: Project Scope
Standard: AUTOSAR C++14	Identifies the AUTOSAR C++14 coding-standard rule violated by the issue.	Issues: By Snapshot, Issues: Project Scope
Standard: CERT C	Identifies the SEI CERT C coding-standard rule or recommendation violated by the issue.	Issues: By Snapshot, Issues: Project Scope
Standard: CERT C++	Identifies the SEI CERT C++ coding-standard rule or recommendation violated by the issue.	Issues: By Snapshot, Issues: Project Scope
Standard: DISA-STIG V4R10	Identifies the DISA-STIG V4R10 coding-standard rule violated by the issue.	Issues: By Snapshot, Issues: Project Scope
Standard: DISA-STIG V4R10 Severity	Identifies the DISA-STIG V4R10 Severity coding-standard rule violated by the issue.	Issues: By Snapshot, Issues: Project Scope
Standard: DISA-STIG V4R3	Identifies the DISA-STIG V4R3 coding-standard rule violated by the issue.	Issues: By Snapshot, Issues: Project Scope
Standard: DISA-STIG V4R3 Severity	Identifies the DISA-STIG V4R3 Severity coding-standard rule violated by the issue.	Issues: By Snapshot, Issues: Project Scope
Standard: ISO TS17961 2016	Identifies the ISO/IEC TS 17961:2013/Cor1:2016 coding-standard rule violated by the issue.	Issues: By Snapshot, Issues: Project Scope

Filter	**Description**	**View Types**
Standard: OWASP Mobile Top Ten 2016	Identifies the OWASP Mobile Top Ten 2016 software vulnerability category violated by the issue.	Issues: By Snapshot, Issues: Project Scope
Standard: OWASP Web Top Ten 2017	Identifies the OWASP Web Top Ten 2017 software vulnerability category violated by the issue.	Issues: By Snapshot, Issues: Project Scope
Standard: PCI DSS 2018	Identifies the PCI DSS 2018 coding-standard requirement violated by the issue.	Issues: By Snapshot, Issues: Project Scope
State	Test state. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*.	Tests
Status	Issue status.[e]	Issues: By Snapshot
Stream	Stream name. You can choose to include or exclude matching names.	Snapshots
Streams	The Streams filter is useful if you need to perform a search for issues in a subset of streams in your project. Enter a full or partial stream name in the menu to select streams that you want to examine. You can select multiple streams, one at a time. Specify the scope of the software issues that you want to include in the search results: • *Any*: CIDs that have ever occurred in any of the selected streams. • *All*: Only those CIDs that have ever occurred in all of the selected streams at one time or another.	Issues: By Snapshot
Suite	Name of the test suite. This filter is only applicable to Coverity Connect projects that include Test Advisor data. See in the *Test Advisor 2020.12 User and Administrator Guide*	Tests
Target	Target platform as specified by: • In the Target attribute in a *Snapshots* view. • The `--target` option to the `cov-commit-defects` command when committing the snapshot to Coverity Connect. For more information about the `cov-commit-defects` command, see the *Coverity 2020.12 Command Reference*.	Snapshots

Filter	Description	View Types
Total	Total number of issues.	Files, Functions, Components, Checkers, Owners, Trends
Total Detected	Total number of issues detected in the snapshot.	Snapshots
Triaged	Number of triaged issues.	Triaged column: Files, Functions, Components, Checkers, Owners, Trends
Type	Issue type. For example, *Resource leak*, *Out-of-bounds write*.	Issues: By Snapshot, Issues: Project Scope, Checkers
Version	The version of the most recent commit to any of the streams in the project. The version is specified by the `--version` option to the `cov-commit-defects` command. For more information, see *Coverity 2020.12 Command Reference*.	Snapshots

[a]LOC is determined by the union of all the file paths in all the latest snapshots in each stream in the project. Coverity Connect counts LOC, comment lines, and blank lines for each file path. If there is more than one instance for a given filepath (when the file path occurs in more than one snapshot), Coverity Connect selects the file with the highest LOC, and take the LOC, comment lines, and blank lines count from that file.

[b]Separate instances of software issues (issue occurrences) receive the same CID if they are found by the same checker and have the same Merged Function Name and Merge Extra property. These instances share the same Merge Key.

[c]See Section 2.4.3.1.2, “Grammar for time filter usage” for information about how CIDs are returned for date filters.

[d]You can use the **<User>** token on the *Owner* filter to return only software issues assigned to whichever user is currently logged in. See Section 2.4.2.3.1, “Relative user”

[e]Status values:

- *New*: Issues that are classified as *Unclassified*.
- *Triaged*: Issues that are classified as *Pending* or *Bug*.
- *Dismissed*: Issues that are classified as *Intentional* or *False Positive*, and are present in the latest snapshot.
- *Absent Dismissed*: Issues that have been classified as *Intentional* or *False Positive* in an earlier snapshot, but are absent from the latest snapshot.
- *Fixed*: Issues that do not occur in the latest snapshot are assigned this status by Coverity Connect.

Coverity Connect automatically tracks the status of issues based on the state of an issue: For example, Coverity Connect assigns the status *New* if the analysis discovers a new issue in the latest snapshot. If you change the classification of an issue, Coverity Connect updates the status to reflect the change. For example, if you change the classification to *Bug*, the status is updated as triaged. If an identical issue exists in more than one stream, Coverity Connect unifies them. This attribute is not displayed in the Triage pane.

2.4.2.1.3. Group By

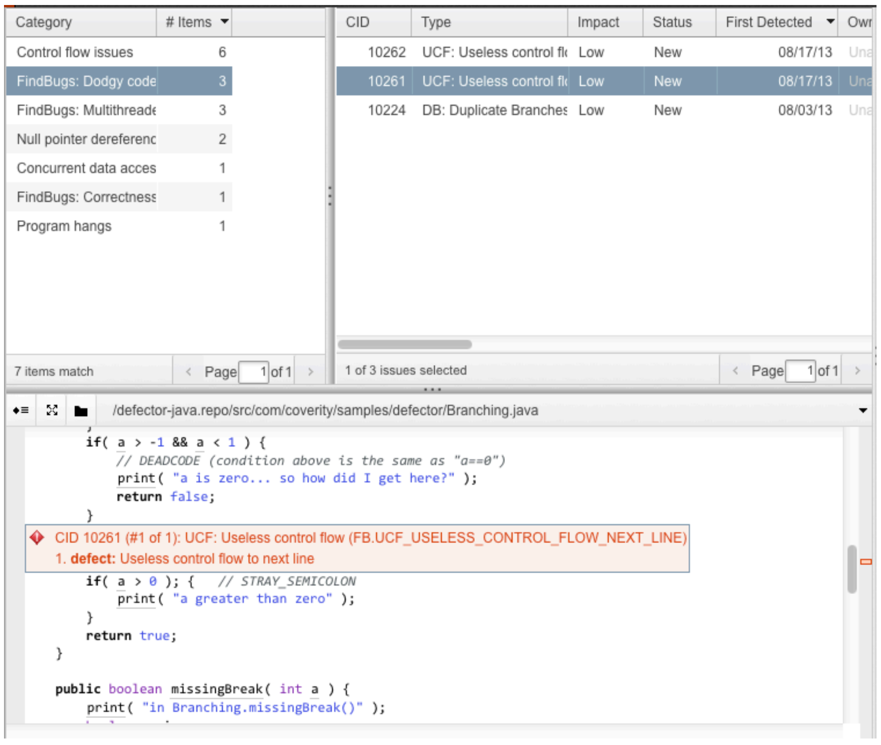
The *Group By* option allows you to organize the software issues from your current view into attribute groups. To use this feature, find the *Group By* pull-down at the bottom of the Filters menu (for the *Issues: By Snapshot* and *Issues: Project Scope* view types only), and select the attribute by which you want to group your issues. See Figure 2.4.4, “Group By Menu”

Figure 2.4.4. Group By Menu

When you choose an attribute to *Group By*, the view pane will include a short list of attribute groupings on the left, with the number of associated software issues for each. The list is sortable by attribute and number of software issues. Figure 2.4.5, “Grouped issues” shows a list of software issues grouped by category.

If a notification email is set on the view, the email will show a table in the same format as the view pane.

Figure 2.4.5. Grouped issues



Category	# Items
Control flow issues	6
FindBugs: Dodgy code	3
FindBugs: Multithreade	3
Null pointer dereferenc	2
Concurrent data acces	1
FindBugs: Correctness	1
Program hangs	1

7 items match Page 1 of 1

CID	Type	Impact	Status	First Detected	Owr
10262	UCF: Useless control fl	Low	New	08/17/13	Una
10261	UCF: Useless control fl	Low	New	08/17/13	Una
10224	DB: Duplicate Branches	Low	New	08/03/13	Una

1 of 3 issues selected Page 1 of 1

/defector-java.repo/src/com/coverity/samples/defector/Branching.java

```
        if( a > -1 && a < 1 ) {
            // DEADCODE (condition above is the same as "a==0")
            print( "a is zero... so how did I get here?" );
            return false;
        }
CID 10261 (#1 of 1): UCF: Useless control flow (FB.UCF_USELESS_CONTROL_FLOW_NEXT_LINE)
1. defect: Useless control flow to next line
        if( a > 0 ); {    // STRAY_SEMICOLON
            print( "a greater than zero" );
        }
        return true;
    }

    public boolean missingBreak( int a ) {
        print( "in Branching.missingBreak()" );
```

2.4.2.1.4. Columns

This settings window allows you to specify which columns to display in a given view, typically so that you can see the most important data by which you are filtering a view (see Section 2.4.2.1.2, “Filters”). By default, Coverity Connect only displays a subset of the columns that are available to a view. To help you determine which columns to display, Coverity Connect provides a description of each column that you can select for a given view.

2.4.2.1.5. Snapshot scope

You use the snapshot scope to specify one or more snapshots that you can then use to create a view of the issues you want to display:

Show

The scope designated in this field determines the snapshots whose issues will be listed after filters are applied.

The scope field(s) can be referenced by a combination of snapshot IDs and a relative statements constructed with a specialized snapshot selection grammar.

This field is only editable in views of the *Issues: By Snapshot* and *Snapshots* view types.

Compared to

This optional field defines the scope of snapshots that will be compared to in the *Show* scope. It determines the value of the *Comparison* column; whether an issue is present or absent from the comparison scope. For more information, see Section 2.4.3, “Snapshot comparison”.

This field is only editable in *Issues: By Snapshot*.

The scope field(s) can be referenced by a combination of snapshot IDs and a relative statements constructed with a specialized snapshot selection grammar.

Show Matches

Displays the snapshots that match the snapshot selection grammar expression that you entered (if any) per field. If the statement is not formatted correctly, Coverity Connect will alert you.

Include outdated streams

Allows Coverity Connect to process the data contained in streams that have been designated as "outdated".

A user with proper RBAC permissions at the stream level can designate a stream to be "outdated" to exclude the stream from Coverity Connect processes. This designation is performed in the Project and Streams configuration window. For more information, see Section 3.3.1.2.2, “Setting up streams”.

2.4.2.2. Creating and editing views

If none of the default views (described in Section 2.4.1.1, “View Types”) provides the data that you need to display in the Coverity Connect View pane, you might decide to create or modify a view.

Modifying a view

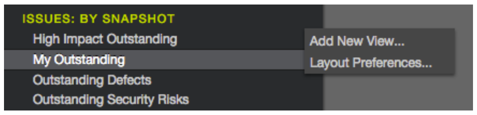
You can modify a view by renaming it and/or changing the filter criteria that it uses for its searches. For a list of available filters and the View types to which they belong, see Section 2.4.2.1.2, “Filters”.

Creating a view

You can create a view in any of the following ways:

- Using the Save as a copy option to create a copy of a view that is similar to the one you want to create and then modifying the copy.
- Modifying the filters of a view that you do not need and then renaming the view.
- Mousing over the View type to which you want to add a view, clicking the view creation menu, and saving a unique name for the view:

Figure 2.4.6. Creating a new view



Note

Each View type supports a unique set of filters, so it is important to create your view with the View type that contains the filters that you want to use. For details, see Section 2.4.2.1.2, “Filters”.

After you create the view, you need to specify any filters you want to apply to it. By default, Coverity Connect does not apply any filters to a new view (unless the view is a saved copy of an existing view).

The *Layout Preferences...* link allows you to define your user preferences.

2.4.2.3. Sharing

This View menu option allows you to share a view with a user or group. You need to supply the user name or group name. Coverity Connect will display the shared view at the end of their list of views (for the View type). This feature avoids the need for users to re-create the view. However, these users cannot delete the view. Instead, the user who shares the view can unshare it by removing any user name or group name from the *Sharing* pop-up window for the view.

Additionally, you can control the display of the views that are shared with you in the *Preferences* window. For more information, see Section 2.4.6, “Preferences”.

2.4.2.3.1. Relative user

When sharing a view with other users or groups, filtering on the relative user will limit the view's software issues to those assigned to the currently logged-in user. A single view can be used to show each user his or her own issues.

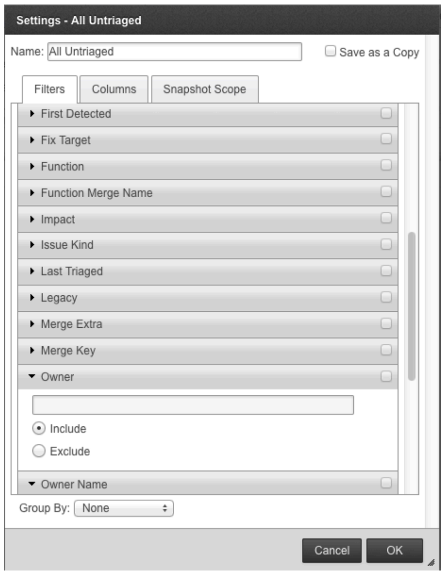
To use this feature, follow these steps (see Figure 2.4.7, “Relative user configuration”):

1. Open the *Filters* menu for the view you wish to share.
2. Click to expand the *Owner* filter.
3. Enter the <*User*> token into the text field.

4. Select the *Include* radio button to show issues assigned to the current user, or the **Exclude** radio button to show all issues not assigned to the current user.

5. Click **OK**.

Figure 2.4.7. Relative user configuration



Name: All Untriaged

[✓ Include] Exclude

Group By: None

2.4.2.4. Notification

This View menu option allows you to configure periodic email notifications to help you track software issue data for a particular view. Emails will be sent according to the specified schedule and will contain a list of issues based on the filters and columns configured for the view. See Section 2.4.4, “View-specific email notifications” for configuration information.

2.4.2.5. Delete

This View menu option allows you to delete a view. If you create a view that is not needed or do not use certain default views that Coverity Connect provides, you can delete the view. You are prompted before you delete the view. Note that if you accidentally delete one of the default views, you can re-create it by assigning the filters described in Section 2.4.1.1, “View Types”. See also, Creating a view.

2.4.2.6. Export CSV

This View menu option allows you to export the data in the View pane to a CSV file to use for internal reports.

2.4.2.7. Export XML

This View menu option allows you to export the data in the View pane to an XML file, for example, to use for internal reports.

2.4.2.8. Get Link

This View menu option allows you to copy the URL of a view, for example, so that you can bookmark the view.

2.4.3. Snapshot comparison

Snapshot comparison is a filtering mechanism that allows you to construct a scope by using the snapshot selection grammar to compare snapshots to the scope defined in the Show field. From the comparison results, you can then apply filters to create views that list the CIDs in which you are interested. For example, you can create views that list:

- Issues introduced in the latest analysis run
- Issues fixed in the latest analysis run
- Issues introduced in the last <number_of> days
- Issues introduced since the last release
- Issues that are have not been fixed since the last release

This feature is not a required operation. It is intended as an "advanced" option to determine a CID's absence or presence in a given scope of snapshot (as displayed in the *Comparison* column):

- present - The CID exists in the snapshot(s) defined in *Show* and at least in one of the comparison snapshots.
- absent - The CID exists in the snapshots defined in *Show*, but not in any of the comparison snapshots.

Note

Use the Comparison filter to list only the CIDs that are present or absent from the results of the comparison scope.

The comparison scope is available to the following view types:

- *Issues: By snapshot* - Located in the Snapshot scope filter of the *Edit Settings* window.
- *Snapshots* - When you click on a snapshot, the *Snapshot Comparison* filter is available in the right hand panel. After you apply the filter, the CIDs that match the comparison scope are opened as an *Unsaved* view in *Issues: By Snapshots*. To create a new view from this unsaved view, click the **Save as Copy** option and rename, apply filters, and save the copied view.

2.4.3.1. Snapshot comparison scope grammar

Coverity Connect provides a grammar that you can use to define the show/comparison scope. Basically, you can define the scope by one or more snapshot IDs and/or through relative expressions. The grammar is as follows:

Table 2.4.2. Snapshot selection grammar

Grammar expression	Description
Snapshot ID	You can specify scope based the number assigned as the snapshot ID. To locate a snapshot ID, go to the *Snapshots* view type with the *Snapshot ID* column enabled. Snapshot IDs should not be specified in selection expressions in views because results will not be returned in projects that do not contain the specified snapshots.
`first()`	Represents the first (earliest) snapshot in a stream or set of streams.
`last()`	Represents the latest (most recent) snapshot committed to a stream or set of streams.
`firstAfter([<snapshot ID> | <expression> | <date>])`	Represents the snapshot that occurs immediately after the specified expression.
`lastBefore([<snapshot ID> | <expression> | <date>])`	Represents the snapshot that occurs immediately before the specified expression.
`<date>`	Either an absolute or relative date: • **absolute** - in the form of `yyyy-mm-dd hh:mm:ss`, however you can exclude the time (hours, minutes, seconds) if desired. If you specify the time, it must be in 24-hour format. • **relative** - in the form of `<N> days ago.` For example, `lastBefore(5 days ago)`. For `<date>` expression behavior, see Section 2.4.3.1.2, “Grammar for time filter usage”.
`..` (dot-dot)	Denotes a range of snapshots. For example: • `20021..20025` • `first()..lastBefore(last())` • `firstAfter(2014-01-02)..firstAfter(2014-01-05)`
`,` (comma)	Denotes a set of snapshots. For example, `20021,20022,20025,30031`.

2.4.3.1.1. Snapshot selection grammar usage notes

- `firstAfter()` and `lastBefore()` must have an embedded expression (snapshot ID, expression, or date).
- You can use combinations of snapshot IDs, expressions, dates, ranges, and sets in the snapshot show/comparison statements. For example:

  - `first()..lastBefore(20023)`
  - `last()..20025`
  - `20021,30031..lastBefore(3 days ago)`
- Use the **Show matches** button to see to which snapshots the grammar maps.
- Coverity Connect will alert you if your snapshot selection grammar is incorrect.
- If there are multiple streams in the project, and you use relative expressions, Coverity Connect will gather the CIDs that exist in the specified snapshot from each stream. For example (using Figure 2.4.8), if you specified a show scope of `first()`, Coverity Connect will union the CIDs from snapshots 10011,20021, and 30031.

2.4.3.1.2. Grammar for time filter usage

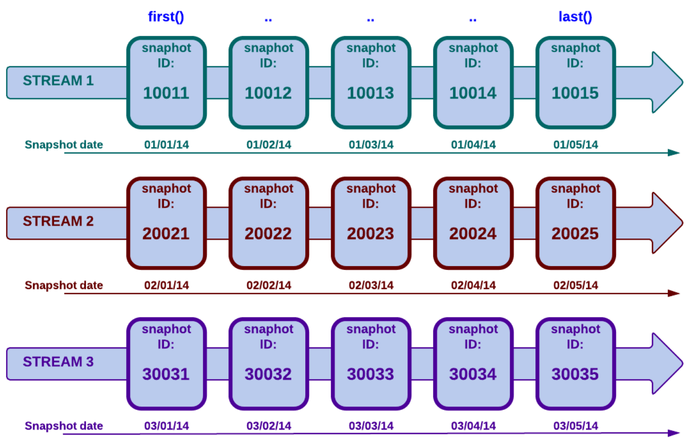
- For absolute dates, if the hours are missing from the `<date>` expression value, Coverity Connect interprets the time of day as 12:00 AM of that day; meaning the beginning of the specified day. For example, `firstAfter(2014-01-01)` returns results from the first snapshot of January 1, 2014 and NOT the results from the first snapshot of January 2, 2014.
- For relative dates, the query considers the current time of day, so `1 days ago` means "in the last 24 hours" and NOT "since 12:00 AM yesterday".

2.4.3.2. Examples

This section provides examples of show and comparison scope expressions as well as examples of creating views to filter on a desired set of defects. Some of the examples reference the illustration below.

The following figure represents a Coverity Connect project containing three streams, each of which are represented by snapshot ID and the date of commit.

Figure 2.4.8. Snapshot comparison example



Comparing the last snapshot with its predecessor to determine new issues

The following are basic examples of comparing the most recent snapshot with the snapshot that occurs just before it. When the scope is applied and the list of CIDs are displayed, a CID that is absent in the preceding (compared to) snapshot implies that the issue was just introduced.

- **Relative expressions:**

  ```
  Show: last()
  Compared to: lastBefore(last())
  ```

  This scope will compare the union of snapshots 10015, 20025, 30035 to the union of snapshots 10014, 20024, 30034. Those CIDs with a Comparison column value of absent will be the newly introduced issues.

- **Snapshot ID:**

  ```
  Show:10015
  Compared to:10014
  ```

  The preceding example compares two snapshots in Stream 1. To specify snapshots in multiple streams, use:

  ```
  Show: 10015,20025,30035
  Compared to: 10014,20024,30034
  ```

Comparing snapshots in separate streams

Snapshot comparison is not limited to snapshots that are adjacent or sequential, nor are the limited to comparison within the same stream. You can compare snapshots that exist in different streams.

- **Snapshot ID:**

  ```
  Show: 10015
  Compared to: 30034
  ```

  The preceding example compares snapshot 10015 from Stream 1 to snapshot 30034 from Stream 3. In addition, you can specify a group of multiple snapshots or series of snapshots from one stream and compare them to a group or series in different snapshot. For example (series):

  ```
  Show: 10012..10014
  Compared to: 30033..30035
  ```

Outstanding issues since the last release

This example shows a comparison scope intending to show all outstanding (unfixed) issues since the last release of a product (assume Stream 1 represents the product code base). The release is represented by a particular date, so in Stream 1, assume that the release date was 01/02/14 (represented by snapshot 10012).

- **Relative expressions:**

  ```
  Show: last()
  Compared to: firstAfter(2014-01-01)
  ```

Fixed issues since the last release

This example shows a show/comparison intending to display all fixed issues since the last release of a product (assume Stream 1 represents the product code base). The release is represented by a particular date, so in Stream 1, assume that the release date was 01/02/14 (represented by snapshot 10012).

- **Relative expressions:**

```
Show: firstAfter(2014-01-01)
Compared to: last()
```

An issue that is absent indicates that the issue was fixed or dismissed in the specified time frame.

2.4.4. View-specific email notifications

Coverity Connect allows you to schedule and receive periodic email notifications to help you track software issue information. Each notification configuration is tied to a view in Coverity Connect, and will display the same attributes as the view to which it is assigned.

Note

These notifications are not available for every view type.

Email notifications must be enabled at a global level by your Coverity Connect administrator. Configuration steps are detailed in Section 3.1.1.15, “Configuring email notification and delivery”.

The email address configured for the *View* owner is used as the email sender address.

2.4.4.1. Use cases

The following scenarios illustrate the use of Coverity Connect's email notifications. These scenarios provide a high-level view of how you might combine view and notification options to track various software issue data. These scenarios cannot cover every use case, because of the large number of possible configurations, but describe several potential solutions.

2.4.4.1.1. Scenario: Weekly team email to reflect all outstanding issues for the release

Goal: To send a team-wide email notification once a week highlighting any issues that have been introduced since the last software release.

Basic configuration: In this scenario, the team lead creates a view that filters on:

- *Status* = *New* or *Triaged*
- *First Detected* date is after last release date

The team lead then adds a notification scheduled for once a week, and adds each team member individually or adds a group including the team members to the list of recipients.

2.4.4.1.2. Scenario: User specific email to catch each new software issue

Goal: To notify any member of a team when a new issue is detected in the code for which they are personally responsible.

Basic configuration: In this scenario, the team lead creates a view that filters on:

- *First Detected* in the last day
- *Owner* = <*User*> (for relative user)

The team lead then adds a notification scheduled for every night at midnight, with each team member or group included in the list of recipients.

2.4.4.1.3. Scenario: Email to single user illustrating newly triaged issues

Goal: To notify a single developer of all issues triaged in the last 24 hours for a specific project (Project Y).

Basic configuration: In this scenario, a developer creates a view that filters on:

- *Last Triaged* in the last day
- *Only in projects* = *Project Y*

The developer then sets the view's columns to all triage values (Classification, Severity, Action, Owner, Fix Target, Ext. Reference) and adds a notification scheduled for every night at midnight, with no additional recipients configured.

2.4.4.1.4. Scenario: Notification upon completed commit

Goal: To trigger an email when a commit is complete, highlighting any new defects introduced to the project (Project Y) in the latest snapshot.

Basic configuration: In this scenario, a developer creates a view that filters on:

- *Only in projects* = *Project Y*
- *Streams & Snapshots* = *Newly detected*

Then, to trigger the notification after completing the commit step, the developer runs the `cov-manage-im` command in `notification` mode:

```
cov-manage-im --mode notification --execute --view <view name>
```

To accomplish this in one step, the developer can create a script which includes each step in the build process and ends with `cov-manage-im`. For example:

```
cov-build <build options> /
cov-analyze[-java] <analyze options> /
cov-commit-defects <commit options> /
```

```
cov-manage-im --mode notification --execute --view <view name>
```

For more information on using `cov-manage-im`, see the *Coverity 2020.12 Command Reference*.

Note

Alternatively, in the *Schedule* tab of the *Notification* dialog, selecting *Send email when a new snapshot is created* will trigger the notification after the commit occurs.

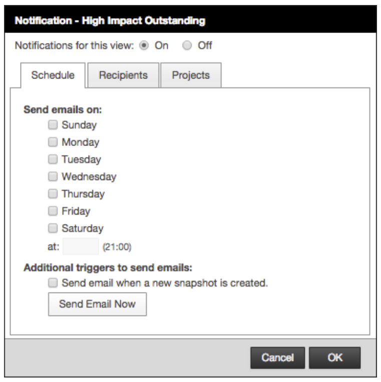
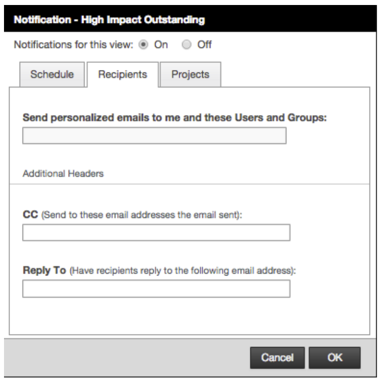
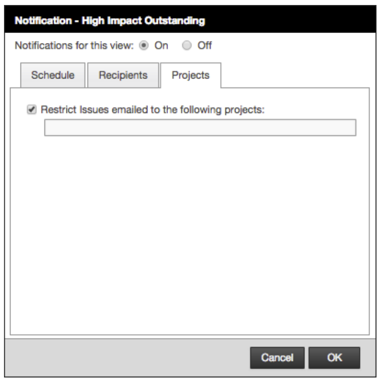
2.4.4.2. Scheduling a view-specific email notification

Once you have configured your view with the appropriate filters, you can schedule a periodic email notification by completing the steps below. You can only configure email notifications on views that you own. If you would like to set up notifications on a view that has been shared with you, duplicate the view and configure the notification on the duplicate.

Note

If an email notification is created for a *Dashboard* of an *Hierarchy*, the email will contain a graphic showing all of the reports on the dashboard. If the email notification is for a specific report within *Hierarchies*, then the single graphic for that report will show in the email.

Table 2.4.3. Notification dialogs

Notification - High Impact Outstanding

Notifications for this view: [✓ On] Off

Schedule | Recipients | Projects

Send emails on:
- ☐ Sunday
- ☐ Monday
- ☐ Tuesday
- ☐ Wednesday
- ☐ Thursday
- ☐ Friday
- ☐ Saturday

at: (21:00)

Additional triggers to send emails:
- ☐ Send email when a new snapshot is created.

Notification - High Impact Outstanding

Notifications for this view: [✓ On] Off

Schedule | Recipients | Projects

Send personalized emails to me and these Users and Groups:

Additional Headers

CC (Send to these email addresses the email sent):

Reply To (Have recipients reply to the following email address):

Notification - High Impact Outstanding

Notifications for this view: [✓ On] Off

Schedule | Recipients | Projects

- ✓ Restrict Issues emailed to the following projects:

1. Open the *View* menu and select *Notification*.

2. To send notifications for this view, select *On*.

3. On the *Schedule* tab, choose the days and the time notifications are sent. Notifications can be sent daily, and must be scheduled at least once a week. If the view does not include any issues at the scheduled time, no email is sent.

   Note

   Time should be entered in 24-hour format (5:00pm = 17:00)

4. Select *Send email when a new snapshot is created* to send an email when a commit of analysis data is completed.

5. Click *Send email now* to send the email report immediately.
6. On the *Recipients* tab, enter additional *Users* and *Groups* to receive the notification. The view owner is automatically included. They must be currently registered in Coverity Connect.

   Note

   Disabled users and those with no associated email address will not receive notifications. Any notifications owned by a disabled user will not be sent.
7. In the *CC* and *Reply To* boxes, enter email addresses for users or mailing lists as needed. These users do not need to be registered users in Coverity Connect.
8. On the *Projects* or *Hierarchies* tab (depending on which part of Coverity Connect the dialog is accessed from), enter the project or hierarchy whose issues are included in the email. If you uncheck *Restrict issues emailed to the following projects:*, the notification will be active for all of your projects.

Notification emails will contain up to a maximum of 100 issues by default. This level can be customized by your Coverity Connect administrator.

Each recipient may receive a different set of issues in the notification, relative to his/her permissions, and the use of the relative user filter. To receive an unfiltered version of the email notification, enter the user's email in the *CC* field.

2.4.4.3. Manually triggering a notification email

It is possible to manually trigger a notification using the `cov-manage-im` command in `notification` mode. To trigger the notification, enter the following command, where `viewName` is the name of the Coverity Connect view for which you want to send a notification:

```
cov-manage-im --mode notification --execute --view <viewName>
```

For more information on `cov-manage-im`, see the *Coverity 2020.12 Command Reference*.

2.4.5. Triage attributes

The Triage pane provides a number of attributes by default. Coverity Connect administrators can add and modify others. For information about using the attributes to triage a CID, see Section 2.2.4, “Triaging issues”.

Note

Coverity Connect allows an administrator with appropriate permissions to create and modify attributes and attribute values. For more information, see Section 3.3.2, “Configuring triage attributes”.

2.4.5.1. Classification

Unclassified
: Default for a new issue. It is intended for issues that have yet to be viewed by a developer.

Pending
: An issue that should be fixed eventually, but perhaps it is not critical enough to fix in the current source code base, or there are other dependencies that prevent it from being fixed at this time.

False Positive
: An issue that a developer has examined and deemed not a true (actual) bug. If a false positive appears to reflect shortcomings or flaws in the analysis engine, please report the issue to `software-integrity-support@synopsys.com`.

Intentional
: An issue that might be a true (actual) bug according to the code's programming language but that is not a bug in this code because either the code is not important or the code can never be exercised in a dangerous way in deployment environments.

Bug
: Reflects a determination that the issue found through a Coverity analysis process is an issue in the code, and is not a false positive or intentional issue.

Untested
: Reflects a determination that a section of the code analyzed by Test Advisor requires that a test be added to the code.

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.

Tested Elsewhere
: Indicates that a section of code is tested by a test outside of the set of tests that are specified in the Test Advisor analysis process.

2.4.5.2. Severity

Severities describe how critical an issue is, that is, how much damage the issue will cause to your program. The default severity attributes follow:

Unspecified
: Default for a new issue. Intended for results that have not been viewed by a developer.

Major
: Bugs that are the most urgent to fix.

Moderate
: Bugs that should be fixed in the near term.

Minor
: Bugs that are the least urgent to fix.

Note that a Coverity Connect administrator can add, delete, and rename severity attribute values in the Triage pane. For more information, see Section 3.3.2, “Configuring triage attributes”.

2.4.5.3. Action

You use these options to describe and track an issue:

Undecided
: Default for a new issue. Indicates that there is no decision yet whether to fix or ignore it.

Fix Required
: Indicates that the issue requires a fix.

Fix Submitted
: Indicates that the issue has been fixed. Note that issues will continue to appear in snapshots until they are absent from the source code, as determined by the analysis based on the checkers that are enabled and the checker options they use.

Modeling Required
: Indicates that incorrect modeling (or the absence of modeling) is confusing the analysis. You can address this issue by fixing or creating the model, which will help the analysis in generating the correct result.

Ignore
: Indicates that the issue can be ignored. This designation might be appropriate for some bugs of minor severity.

Note that a Coverity Connect administrator can add, delete, and rename these attributes. For more information, see Section 3.3.2, “Configuring triage attributes”.

2.4.5.4. Legacy

By default, Coverity Connect does not display this attribute. When displayed, you can indicate whether this is a legacy issue (for example, *Legacy* = *True*). For details about this attribute, see Section 3.3.5, “Configuring legacy issues”. See also, Alternative to using separate triage stores.

2.4.5.5. Fix Target

By default, Coverity Connect does not display this attribute. When displayed, you can select the release target (or other target milestone) by which an issue should be fixed. Typically, you use this attribute only if a CID has been classified as a *Bug* for which a fix is required (*Action* = *Fix Required*). See also, the Alternative to using separate triage stores.

2.4.5.6. Owner

You can assign an owner to an issue or a change the owner of an issue.

2.4.5.7. Ext. Reference

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

2.4.5.8. Comment

You can provide a comment about the issue in the text field. For example, if an issue is marked as a *Bug*, you might use the field to document the reason for that designation.

2.4.6. Preferences

Coverity Connect allows you to set the following preferences (shown in Figure 2.4.9, “Link to Coverity Connect Preferences”):

- General preferences: For modifying your name, email address, password, and locale.

  Supported locales:

  - *English (United States)* - default
  - *Japanese (Japan)*

  The language setting will take effect immediately.

- **Layout**: Allows you to control the following:

  View types
  : Select the view types that you want to be displayed in the left hand panel.

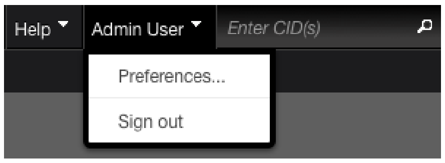
  Display views shared with you
  : You can either display all of the views that are shared with you, or you can select the *Custom* attribute and click the *edit* link to select the views that you want displayed in the left hand panel.

  Rows per page
  : Choose how many rows (for example, number of issues) that are displayed on a page. The default is 200.

- **Component Subscriptions**: For receiving email notifications when issues are found in selected components. For this feature to be available to you, a Coverity Connect administrator needs to set up email notification in Coverity Connect and to assign component-related RBAC permissions to you. See also, Section 3.1.1.15.1, “Setting up component subscription notification”.

- **Access Roles**: For seeing the Role-based Access Control (Section 3.2.3, “Roles and role based access control”).

Figure 2.4.9. Link to Coverity Connect Preferences



Part 3. Coverity Connect Administration

Administrators set up and manage Coverity Connect for end users through Configuration screens that are accessible from the *Configuration* menu, located in the toolbar.

End users view, triage, monitor, and fix issues found by Coverity Analysis and third party tools. For details about end usage, see Part 2, “Coverity Connect Usage”.

For installation instructions, information about supported platforms, and licensing options, see Chapter 1.1, *Installation, Product Licenses, and Supported Platforms*.

Coverity Connect also supports an API for creating Web services that communicate with the Coverity Connect database. For details, see the Coverity Connect Web Services API Reference.

Chapter 3.1. Performing system configurations

Table of Contents

3.1.1. Configuring and managing the Coverity Connect server 62
3.1.2. Configuring and managing the embedded database 129
3.1.3. Client-side SSL Certificate Management 135

3.1.1. Configuring and managing the Coverity Connect server

This section describes how to configure and manage the Coverity Connect server and embedded database.

For additional information specific to clustered deployments, see Chapter 3.5, *Configuring Coverity Connect enterprise clusters*

Coverity® Web Services

In addition to providing access to Coverity Connect through a browser, the *Coverity Connect 2020.12 Web Services API Reference* allows applications and scripts to remotely access functionality on the Coverity Connect server. The SOAP-based API provides the following services:

Defect Web Service
: Get defects, project metrics, trend data, source file contents, and checker subcategory information. Triage defects and copy stream state.

Configuration Web Service
: Create, delete, get, and update projects, streams, components, and attributes (actions, severities, classifications, defect statuses). Get and delete snapshots.

3.1.1.1. Configuring Coverity Analysis Updates

Synopsys allows you to automatically download Coverity Analysis updates through Coverity Connect. To configure the update settings, go to **Configuration** > **System** > **Analysis Update**. By default, updates are turned on in Coverity Connect. This means that you allow Coverity Analysis users to download and install Coverity Analysis updates when Synopsys releases them and makes them available to download. Coverity Connect sends a request for updates each day at 1:00 A.M. (Coverity Connect server's local time), by default.

Coverity Connect downloads and stores the update files locally to ensure fast delivery of the updates to the Coverity Analysis users.

3.1.1.1.1. Storage Capacity and Management

As Coverity Connect continues to download updates, you can manage the storage space used for these updates on the Coverity Connect server. The Storage Usage section displays the current amount of storage being used and also has settings to manually or automatically remove update files. You can remove updates based on storage limits or the age of the update file.

The smaller the storage capacity (5 GB is the minimum requirement), the faster the file purge turnover. This can be desirable if storage space is an issue. However, if network bandwidth is an issue, then a larger storage capacity can ensure that all update files are available to the Coverity Analysis user without the need to download update files from Synopsys each time Coverity Analysis requests an update.

- If no check boxes are selected, then there is no storage limit set and there are no daily purges.
- To manage storage on a daily schedule, based on storage limits, select the **Allow up to** check box, then in the **GB** text box, enter the number of GB to set as the storage limit. For example, if you want to set the storage capacity to 10 GB, enter 10 into the **GB** text box. (You must allow at least 5 GB of storage.)

  Now Coverity Connect stores update files only up to the set capacity limit. As Coverity Connect continues to download more update files and the storage capacity is surpassed, Coverity Connect removes the oldest update files until the combined size of the files is within the defined storage capacity.
- To manage storage based on the age of the update file, select the **Purge files older than** check box, then in the **days** text box, enter the number of days to represent the update file age limit. For example, if you want to set the age limit to 10 days, enter 10 into the **days** text box.

  Now Coverity Connect stores update files only up to the set age limit. As an existing update file's age limit is surpassed, Coverity Connect removes the update file.
- To immediately purge update files, based on the Storage Usage settings, click the **Purge Now** button. (One or both of the check boxes must be selected to use the Purge Now button.) Only update files that match the Storage Usage settings are purged. For example, you normally set the daily purge storage limit to 10 GB, but you want to free up some storage so that you can download more update files and store them locally for the Coverity Analysis user to download. In this case, you can enter 5 into the GB text box, and click **Purge Now**. Afterwards, return the storage capacity setting to what you previously had it set to. In this case, set it back to 10 GB.

3.1.1.1.2. Developer Workflow

When a Coverity Analysis user commits defects to Coverity Connect, using the `cov-commit-defects` command, Coverity Connect sends a notification if there are new updates. Coverity Connect determines which updates are relevant based on the commit. Coverity Connect notifies Coverity Analysis only about relevant updates and makes them available to download. Any other updates are ignored. In addition, if the request determines that an update is required, but it is not stored locally on Coverity Connect, then Coverity Connect requests the update from Synopsys. (This occurs whether or not there is enough storage available to store it locally; if there is not enough storage then Coverity Connect does not store it locally.) In this way, the Coverity Analysis user always gets the latest updates regardless of the amount of storage that Coverity Connect allows for storing the update files.

The Coverity Analysis user can use the `cov-install-updates` command with its sub-commands to query and list the available updates, install the updates in order, and if required, rollback an undesired update. For more information, see the *Coverity 2020.12 Command Reference*

3.1.1.1.3. External Network Access

Update information is hosted on the Synopsys Customer Portal, and access to this site is mediated by an authentication proxy server. When querying and downloading analysis updates, a Coverity Connect instance must present a valid license ID to the proxy server.

To support Coverity Analysis updates, a Coverity Connect instance must be configured so it can reach external web addresses. It may be necessary to configure your firewall to allow messages to and from the Authentication Proxy URL (`https://sig-updates.synopsys.com`).

Update packages are downloaded directly from Amazon Web Services. It may also be necessary to allow messages to and from `https://s3.amazonaws.com/cdtpincrementals/*`.

3.1.1.2. Configuring global owner assignment rules

This tab assigns global rules for automatic ownership using historical data derived from an SCM (Source Code Management) tool. For more information about the entire automatic ownership configuration process, see Chapter 3.4, *Configuring automatic owner assignment*.

3.1.1.3. Integrating with Jira

Coverity Connect allows you to export defects found by Coverity Analysis to an existing Jira instance. When configured, any CID can be easily added to a specified Jira project, with various fields automatically generated.

Note

If multiple defect export methods are configured, then the first one in this order is selected: `export-defect-handler`, URL, or Jira.

3.1.1.3.1. Requirements

Before configuring Coverity Connect to integrate with your Jira instance, be sure that you have all of the following:

- Jira version 5.0+ or Jira Cloud
- Coverity Connect version 7.7.0 or later
- Your Jira URL, username, and API token
- The Jira project(s) to which your CIDs will be exported

3.1.1.3.2. Configuring Jira export

The configuration process for Jira integration consists of two main tasks:

- Server configuration
- Project mapping

Note

Field maps carried over from earlier versions of Coverity Connect are duplicated so that there is one field map per project. This may cause some of the field mappings to be invalid for a given project. To check that the field mappings are valid, you should click **Test Mappings** to test all of the mappings, and edit or delete mappings found to be invalid.

After you have completed the configuration tasks, the *Triage* panel will include an **Export** button. When clicked, a new bug will be created in the specified Jira project with the information exported from the Coverity Connect issue.

Note that each issue can only be exported once.

3.1.1.3.2.1. Server configuration

Note

Before you begin the following procedure, you must first create an API token associated with the Jira account you intend to use. Refer to the Atlassian Jira documenation for information on creating API tokens.

The first step in configuring integration with Jira is establishing a connection to the Jira server. To do so, complete the following steps:

1. Navigate to Configuration → System → Bug Tracking System: JIRA and click **JIRA Server Configuration...**
2. Enter the URL of your Jira server in the *Location* field.
3. Enter the username for your Jira account into the *Username* field.
4. Copy and paste an API token associated with your Jira account into the *Password* field, and then click **Check Connection**.

   At this point, you will see a message confirming a successful connection, or an error message explaining how you might correct the failed connection.
5. When you have a successful connection, click **OK** and move on to the project mapping step.

Note

Note that it may be necessary to disable the Jira Captcha feature prior to connecting with the Jira server. To do so, complete the following steps:

1. Go to Jira administration and select *System*.
2. Click **Edit Settings**.
3. Change the value in *Maximum Authentication Attempts Allowed* to blank.
4. Click **Update**.

Note

In the case of connecting to a Jira server using SSL, it may be necessary to import the Jira self-signed certificate into the Java keystore for Coverity Connect.

1. Obtain the Jira SSL certificate (*.pem).
2. Execute `cd <install_directory>/jre/lib/security`
3. Make a backup of the `cacerts` file.
4. Execute `../../bin/keytool -importcert -keystore cacerts -file JIRA-CA.pem`
5. Restart Coverity Connect and confirm the connection.

3.1.1.3.2.2. Project mapping

After connecting with the Jira server, choose which Jira projects you want to export your Coverity Connect issues to. Project mapping allows you to associate your Jira projects with related Coverity Connect projects, so that when you export an issue, the bug is generated in the correct Jira project, with the appropriate information included.

You can create and edit project mappings by using the options in the *Bug Tracking System: JIRA* pane. After the Jira server is successfully configured, the project list displays existing mappings between Jira projects and Coverity Connect projects. Each project mapping consists of a Jira project, one or more associated Coverity Connect projects, and mappings to specify which information to include with each exported issue.

The *Bug Tracking System: JIRA* pane provides the following options:

Add...
Opens a pop-up dialog to create new project and field mappings. Here you can specify a Jira project and the associated Coverity Connect projects. After specifying the projects to map, you specify the appropriate Jira field along with an associated Coverity Connect field or constant value.

Edit...
Opens a pop-up dialog to edit the selected project mapping.

Delete...
Deletes the selected project mapping.

Test Mappings
Tests all of the defined project mappings to check for errors.

3.1.1.3.2.3. Adding a project mapping

1. Click **Add**.
2. Click **Select JIRA Project** and select one of the projects available on the Jira server.
3. In *Projects*, start entering the name of a Coverity Connect project to add. Alternatively, click **Edit** to choose one or more projects from a list of all available projects.

4. For *Mode*, select *Live* or *Test*. Use *Test* while you are developing the integration. In *Test*, Coverity Connect issues are not actually sent to Jira.

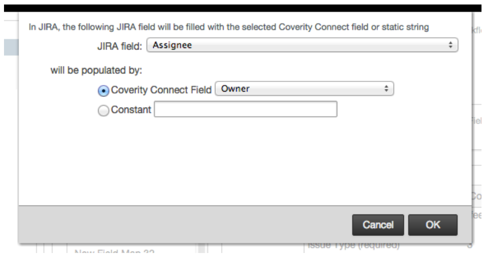
   Note

   When you are ready to start sending issues to Jira, change the *Mode* to *Live*.

5. On the next page, for *Issue Type*, choose the type of issue defined in the Jira project that will be assigned to all issues exported from Coverity Connect to that Jira project.

6. On the next page, assign a Coverity Connect field to each Jira field. Select each required Jira field and click **Edit**. To export information to other Jira fields, click **Add**.

   Figure 3.1.1. Add field mapping



In JIRA, the following JIRA field will be filled with the selected Coverity Connect field or static string

JIRA field: Assignee

will be populated by:

[✓ Coverity Connect Field] Owner

Constant

7. Select a Coverity Connect field to be the source of the value for the Jira field, or select *Constant* to set a constant value. Either or both options are shown, depending on settings in the Jira project.

8. After all Jira fields have been mapped to Coverity Connect fields, click **Next** to check the validity of the mappings. If they are valid, click **Finish** to save the map. If the Jira field values are not valid, helpful error messages will appear and you can click **Back** to edit them.

3.1.1.3.2.4. Using templates for the Constant static field

You can use simple variables in the *Constant* field to create templated static fields. To do so, use angle brackets around the variable name, and the relevant value will be substituted into the exported Jira field. For example, "<checker> found in <file>" will be formatted to something like "NULL_POINTER found in main.c".

The available variables are:

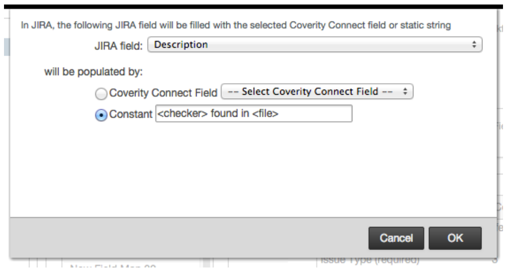
Table 3.1.1. Variables for Static Fields

Variable	Description
action	The action that is being taken for the issue.
category	The checker category describing the nature of the software issue.
checker	The checker that found the issue.
cid	The CID of the issue.
classification	The classification of the issue.

Variable	**Description**
comparison	Indicates if the issue is present in the snapshot used for comparison.
component	The component where the issue is found.
cwe	The Common Weakness Enumeration identifier for the issue.
file	The file where the issue was found.
firstdetected	Date when the issue was first detected
firstsnapshot	Snapshot when the issue was first committed.
firstdate	Date of snapshot when the issue was first committed.
firstdesc	Description of snapshot when the issue was first committed.
firststream	Stream in which the issue was first detected.
firsttarget	Target platform of the snapshot in which the issue was first detected.
firstversion	Version number of the snapshot in which the issue was first detected.
fixtarget	Target milestone for fixing the issue.
function	The name of the function where the issue is located.
functionmerge	Internal function name used as one of the criteria for merging separate occurrences of the same software issue, with the result that they are identified by the same CID.
impact	Issue impact as determined by Coverity Connect: High, Medium, Low, or Audit.
lastsnapshot	Snapshot where the issue was last detected.
lastdate	Date when the issue was last detected.
lastdesc	Description of the snapshot in which the issue was last detected.
laststream	Stream in which the issue was last detected.
lasttarget	Target platform of the snapshot in which the issue was last detected.
lastversion	Version number of the snapshot in which the issue was last detected.
lasttriaged	Date when the issue was most recently triaged.
legacy	The Legacy attribute of the issue.

Variable	Description
mergekey	Internal signature used to merge separate occurrences of the same software issue and identify them all by the same CID.
mergeextra	Internal property used as one of the criteria for merging occurrences of an issue.
username	The username of the owner of the issue.
owner	The first name and last name of the owner of the issue.
severity	Severity of the issue.
status	Issue status.
type	Type of issue.
url	URL to CID of issue.

Figure 3.1.2. Add static mapping



In JIRA, the following JIRA field will be filled with the selected Coverity Connect field or static string

JIRA field: Description

will be populated by:

Coverity Connect Field [-- Select Coverity Connect Field --]

[✓ Constant] <checker> found in <file>

3.1.1.4. Integrating with other (non-JIRA) bug tracking systems

Coverity Connect can be configured to export issues to other bug tracking systems. Currently this option supports the Bugzilla bug tracking system.

Note

If both JIRA and Bugzilla are configured to receive issues exported from the same project, those issues will only be exported to JIRA.

3.1.1.4.1. Requirements

This integration requires Bugzilla 4.x or 5.0. The integration utilizes the Bugzilla XML-RPC API. Ensure that XML-RPC is enabled. Please refer to the Bugzilla Administration Guide on how to enable the XML-RPC API for Bugzilla.

3.1.1.4.2. Server Configuration

1. In Configuration → System → Bug Tracking System: Other, click **Server Configuration**.

2. In *Location*, enter the URL of the bug tracking system.

3. Enter the appropriate *Username* and *Password*.

4. Click **Check Connection** to verify the settings.

5. Click **OK**.

Note

When attempting to connect to the Bugzilla server, do not exceed the maximum number of login attempts, otherwise the Bugzilla account may be locked. Contact the Bugzilla administrator in this case.

3.1.1.4.3. Configuring the project mapping from Coverity Connect to Bugzilla

The data that is exported to Bugzilla is defined by a project mapping. A project mapping specifies how the values of certain fields in a specific Coverity Connect project are exported to a specified Bugzilla product. One or more project mappings are defined in a JSON configuration file. To set up the bug tracking system integration, the Coverity Connect administrator exports a JSON file from Coverity Connect and modifies it as needed, and then imports the file. The example shows the required fields.

```
{
     // File type and format identifier. These fields are required
     // and must exactly match the values shown here.
     "type": "Coverity Connect BTS configuration",
     "format_version": 1,
     "variables" : {
 "defaultComp" : "Cov Test Component",
 "configMap" : {
  "CC_Other" : "Other"

 }
      },
     "configurations": [
       {
         // This is a user-provided name for this configuration and should
         // appear as a column in the BTS config list. Required.
         "name": "Sample Defect Export",

         // This is an optional textual description of the configuration.
         "description": "This configures exports defects from the testVA project.",

         // This is the set of Coverity Connect projects for which this
         // configuration applies.  When a bug is exported, we look at the
         // currently selected project, and this configuration applies when
         // the current project is in this list.  If multiple configurations
         // list the same project, the last one in the JSON file wins.
         "applies_to_projects": [
               "i18n","sample-ces","race"
         ],

         // This specifies the name of a BTS plugin to use to export the
```

```
        // defects.  It is required, although in Coverity Connect 8.0 the only 
allowable
        // value is "bugzilla".
        "bts_plugin": "bugzilla",

        // Next is a set of name/value pairs that define the contents of
        // an exported bug as a function of the attributes of the defect
        // that is being exported.  The latter are referred to using <variable> 
syntax.
        // The exact set of meaningful attributes is dependent on the selected 
bts_plugin
        // and how the BTS itself is configured.
        //
        "export_attributes": {
           // This specifies a mode in which we want to export a defect. It can take 
a value of "live" or "test".
           // If it is in test mode, actual export does not happen. To do an actual 
export, it has to be in "live" mode.
           "mode" : "live",
     // Bugzilla product that receives the defect.
           "product": "Coverity Test Product",
      // Other BZ attributes.
           "component": "<configMap[component]|defaultComp>",
           "version": "1.0",
     "bug_status": "<status>",
       //  "platform": "All",
 //  "op_sys": "Linux",
     "bug_severity": "<severity>",

           "bug_file_loc": "<url>",
      "cf_eventtag": "<eventtag>",
      "cf_linenumber": "<linenumber>",
      "cf_eventdescription": "<eventdescription>",
           "bug_found_by": "PoP",
      "cf_bug_url": "<function>",
      // The bug title/ summary.
           "short_desc": "CID <cid>: <checker>",
      // The description field (comment 0).
           "description": "File: <file>:<linenumber> Function: <function> Checker: 
<checker> <eventtag>: <eventdescription> Click <url> for more details."

      }
    }
  ]
}
```

JSON file attributes:

`name`
: Name of the Bugzilla product.

`description`
: A description of the Bugzilla product.

`applies_to_projects`
: One or more Coverity Connect projects can be mapped to one Bugzilla product.

> **Note**
>
> A single Coverity Connect project can only be mapped to one Bugzilla product at a time. In case of conflict, the most recent Bugzilla product mapping will be used.

`bts_plugin`
: The value must be `bugzilla`.

The following `export_attributes` values are required by Bugzilla. A `Bugzilla Constant` is a picklist value defined in the Bugzilla product. If this value does not match the actual value in Bugzilla, an error will occur when exporting an issue. The actual name of fields in Bugzilla may differ from the display name shown in the Bugzilla interface. Contact the Bugzilla administrator to confirm the actual names. Coverity Connect templates may be used in combination with string text in some cases.

`mode`
: Allowable values are `test` or `live`. It must be set to `live` for defects to actually be exported.

component
: `Bugzilla Constant`

summary
: String text and template fields

version
: `Bugzilla Constant`

description
: String text and template fields

op_sys
: `Bugzilla Constant`

platform
: `Bugzilla Constant`

priority
: `Bugzilla Constant`

severity
: `Bugzilla Constant`

The Coverity Connect fields that may be exported are:

- action

- category
- checker
- cid
- classification
- comparison
- component
- cwe
- file
- firstdetected
- firstsnapshot
- firstdate
- firstdesc
- firststream
- firsttarget
- firstversion
- fixtarget
- function
- functionmerge
- impact
- lastsnapshot
- lastdate
- lastdesc
- laststream
- lasttarget
- lastversion
- lasttriaged
- legacy

- mergekey
- mergeextra
- username
- owner
- severity
- status
- type
- url
- eventtag
- eventdescription
- linenumber

3.1.1.4.3.1. Maintaining or replacing project mappings

To replace the current project mappings, import a new JSON configuration file containing new project mappings.

Note

To update the existing project mappings, click **Export** to extract the JSON file, edit the JSON file as needed, and then re-import the modified file.

3.1.1.4.4. Variable Substitution

In the JSON file, variables are substituted with values from two sources:

- Coverity Connect fields
- Optional `"variables"` section in the JSON file

For example, the following expression is replaced with values from both a Coverity Connect field and the `"variables"` section.

```
"component": "<configMap[component]|defaultComp>"
```

Given the following `"variables"` section, this expression will resolve as follows:

1. The Coverity Connect field `component` is used for the map lookup, and succeeds if it matches `"CC_Other"`. Then the result is `"Other"`
2. Otherwise, the value for `"defaultComp"` is substituted, which is `"Cov Test Component"`.

```
                 "variables" : {
                "defaultComp" : "Cov Test Component",
                "configMap" : {
             "CC_Other" : "Other"
                       }
```

The following rules apply to variable substitution:

1. Use "<foo>" syntax to mean the substitution of the value of input variable foo.
2. Use the set of defect fields listed in "JSON file to import" above as inputs. For example, "<fixtarget>" is replaced by the parser with the contents of the defect's fixtarget field.
3. Use a map of user-defined variables defined in the JSON as inputs, using the same syntax as the previous requirement. For example, if the variables include "foo" : "bar", then the syntax "<foo>" indicates that the string "bar" (without the quotes) should be substituted.
4. Variable names must be taken from [a-zA-Z0-9]. The parser will verify this.
5. Since the defect field names and the variable names are drawn from the same namespace, they must not collide. The parser implementation will require and verify this.
6. Use the syntax "<foo[bar]>" to mean that:
   - The value of "bar" and "foo" will be looked up in the input space.
   - The value of "foo" will be used as a map and a lookup performed using the value of "bar".
   - Both "foo" and "bar" must exist in the input space. The value of "foo" must be a map. The map must contain an entry for "bar".
7. Use the syntax "<foo[bar]|baz>" to mean the same as "foo[bar]" except that if the map lookup fails, the value of "baz" from the input space (which must be present, whether or not the lookup succeeds) is substituted.
8. In addition to any number of substitutions, value strings may include any number of characters expressible in JSON. To express the "<" character, "<<" is used.

3.1.1.4.4.1. Example: Map Coverity fields to Bugzilla fields

The following example maps Impact fields (High, Medium, Low) in Coverity Connect to the corresponding fields (Highest, Normal, Lowest) in Bugzilla. To accomplish this, create a `configMap` section in the `variables:` section of the JSON file.

```
"configMap" : {
                // Map Coverity Impact field to Bugzilla Priority Field
                    "High" : "Highest",
                    "Medium" : "Normal",
                    "Low" : "Low",
                    "Audit" : "Lowest"
```

```
            }
```

In the `export_attributes` section of the JSON file, add this line:

```
"priority" : "<configMap[impact]>",
```

When an issue is created in Bugzilla, the impact values from Coverity Connect are looked up in the `configMap` section, and replaced in the Bugzilla entry with the corresponding values.

3.1.1.4.5. Connecting to a Bugzilla instance using SSL

If the Bugzilla instance uses SSL, it is necessary to import the SSL certificate to Coverity Connect. The following example is shown for Windows.

1. Access the Bugzilla instance through a browser.
2. In the Address bar, click the "lock" icon and save the certificate.
3. Double-click the certificate and click *Install certificate*. Follow the instuctions to add the certificate to the *Trusted Root Certificate Authorities store* .
4. On the command line, launch `certmgr.msc`.
5. Navigate to the *Trusted Root Certificate Authorities store*.
6. Expand the tree and click *Certificates*.
7. Find the certificate saved previously.
8. Right-click on the certificate and choose *All tasks* and *Export*.
9. Navigate to the Coverity Connect installation directory.
10. Execute the command: `<installationfolder>/jre/bin/keytool -import -alias "bugzilla" -file exportedCert.cer -keystore <installationfolder>/jre/lib/security/cacert`
11. Restart Coverity Connect.

The following procedure is for Linux:

1. Obtain the Bugzilla certificate (for example, `bugzilla-cert.crt`) and copy it to the following directory:

   ```
   $ sudo cp bugzilla-cert.crt /usr/local/share/ca-certificates/bugzilla-cert.crt
   ```

   ```
   $ sudo update-ca-certificates
   ```

2. Import the Bugzilla certificate into Coverity Connect:

   ```
   $ keytool -importcert -file bugzilla-cert.crt -keystore ~/<install-dir>/jre/lib/security/cacerts
   ```

The default passphrase is: `changeit`

3. Restart Coverity Connect.

Note

If the SSL handshake fails and leaves an "unrecognized_name" error in `cim.log`, then set the following parameter in the `<install-dir>/config/system.properties` file:

```
java_opts_pre=-Djsse.enableSNIExtension=false
```

(The parameter `java_opts_post=-Djsse.enableSNIExtension=false` also works.)

Note

If the SSL handshake fails and leaves a "No name matching" error in `cim.log`, then the long name of the certificate must be used. For example, if the full certificate name is `bugzilla-cert.company.com`, using the short name will cause the SSL handshake to fail.

3.1.1.5. Configuring custom checker descriptions

Note

Note that the features described in this section are deprecated as of version 7.6.0. This functionality will be removed completely in a future release.

Custom checker properties are now reported during the commit process automatically, so importing checker descriptions via a CSV file is no longer necessary. Coverity Connect continues to support custom issue categories and impact - see Section 3.1.1.6, “Configuring custom issue categories”.

Coverity Connect allows you to add issue descriptions and information that will be displayed in the UI when a custom checker discovers an issue. The information you add can appear in various locations in the Coverity Connect UI, such as:

- In the issue filters for checkers, categories, and types.
- In the CWE listing and Information tab in the Triage pane.
- In the Source browser as event descriptions.

All unrecognized checkers, developed with the Extend SDK or some other custom solution, are categorized as *Miscellaneous* (a `category` value) and as type *Other violation* (a `name` value) in the UI unless you specify the checker descriptions in a CSV file (for example, `customCheckers.csv`) and then import the file to Coverity Connect. Alternatively, you can create a custom categorization map that recategorizes the defect type (see Section 3.1.1.6, “Configuring custom issue categories”).

Note

Note that all custom checkers will initially be assigned an impact of "Low". If you wish to change the impact value for a custom checker, create an Issue Categorization Map to map the checker `type` to your desired `impact` value.

The CSV file accepts the following values:

domain

Represents the programming language that the checker analyzes or the type of analysis. The following values are allowed:

- STATIC_C - C/C++ language.
- STATIC_JAVA - Java language.
- STATIC_CS - C# language.
- DYNAMIC_JAVA - Dynamic Analysis.
- OTHER - Denotes another language that might be used with importing third-party issues. See Using the Third Party Integration Toolkit for more information.

This is a required field.

Name

The display name of the checker. This is a required field.

subcategory

A checker subcategory, describing the issue produced by the checker. Checkers can have multiple subcategories. This is an optional field.

Extend SDK checkers do not support subcategories, so all issues will have a subcategory of *none*.

category

A string used to describe the nature of the issue. This must be a string between 2 and 256 characters long, and can be a previously known category or a custom category.

This a required field.

type

A short description of a checker that will be displayed under the *Type* filter in the issue view. For more information abut the checker filters, see Part 2, “Coverity Connect Usage”.

This is a required field.

CWE

The number that corresponds to an issue description in the Common Weakness Enumeration (CWE). The CWE provides further information and examples of the issue type. Leave this field empty if there is no plausible link, as with Test Advisor violations.

long description

The description that will appear in the upper section of the triage pane.

local effect

The description that will appear under the Information tab in the triage pane.

event set 0 caption

Describes the event(s) that leads to the issue as found by your checker. This description is displayed in the *Occurrences* tab of the Triage pane.

You can specify multiple events with *event set 1 caption* and *event set 2 caption*.

The file is simply a list of comma separated values. For example:

```
STATIC_C,checker1,*,apiUsageErrors,C API usage error,382,long description,local
 effect,event set 0 caption,event set 1 caption,event set 2 caption
STATIC_C,checker2,subcategoryA,buildSystemIssues,C build system issue,,long
 description,local effect,,,
STATIC_C,checker2,subcategoryB,controlFlowIssues,C control flow issue,398,long
 description,local effect,,,
STATIC_JAVA,checker1,*,apiUsageErrors,Java API usage error,382,long description,local
 effect,event set 0 caption,event set 1 caption,event set 2 caption
```

Be sure to format the CSV fields precisely, with no additional spacing or quoting around field values. Each entry should be typed on a single line.

Note

Wildcards ("`*`") are accepted in the following places in the CSV file:

1. After the period ("`.`") in a checker name. For example, `CustChkr.*` will return `CustChkr.1`, `CustChkr.2`, `CustChkr.3`, and so on (assuming they exist).

2. In the place of a *subcategory name*. The wildcard indicates that the specified *category*, *cwe*, *short description*, and so forth, should be applied to all subcategories of the specified checker.

To import the CSV file

1. Edit and save the CSV file with your custom checker descriptions.

2. Navigate to Configuration → System → Custom Checkers.

3. Click the **Import** button and browse to the location of the CSV file.

   - If the structure of the file is valid, you will receive a Success notification.

   - If it is not valid, you will receive a Failure notification. In this case, you need to check your file for errors. You can also search for errors in `cim.log`.

   After the file is successfully imported, the checker name, description, and language (domain) are displayed in the custom checker table. The custom checkers will also be added to all of your category maps, under the categories specified in the imported CSV.

To remove all custom checkers

1. Create an empty CSV file.

2. Click the **Import** button to import the empty CSV file.

This action will remove any existing custom checker settings.

To remove an individual custom checker

1. Click the *Export* button and save the `customCheckers.csv file`.
2. Open the `customCheckers.csv file.`
3. Remove the line or value (if it is not required) of the checker that you want to delete.
4. Save the `customCheckers.csv file`.
5. Click the **Import** button to import the new CSV file.

3.1.1.6. Configuring custom issue categories

Coverity Connect allows you to change the impact level and category of issue types through a custom issue (defect) category map file (JSON file). Issues are categorized by the following criteria:

- Impact levels (High, Medium, Low, or Audit). In Coverity Connect, you can filter software issues by impact level, which can help you to identify the issues that require attention first.
- Issue description and category. For example, a C/C++ FORWARD_NULL checker can report Medium priority *Unchecked dynamic_cast* issues that belong to the *Null pointer dereferences* category.

For details about issue categories, see the Coverity 2020.12 Checker Reference.

To customize issue categories:

1. Navigate to Configuration → System → Issue Categorization.
2. Click **Download Value File** to generate a list of available values for "`type`" and "`impact`", along with all currently known category names. This information will be useful in editing/creating the issue categorization map.
3. If you already have an issue categorization map you want to edit, select it from the *Issue Categorization Map Name* list, and click **Export**. Then you can open and edit the file in your text editor. Otherwise, you will have to create a new issue categorization map.

   See Section 3.1.1.6.1, “Issue categorization map JSON format” for information on formatting the map file.
4. Click **Import** and browse to the JSON file location.
5. Click **Next**. A message will be displayed to tell you whether the import was successful. If the import fails, an error message will be displayed, describing the problem.

   Note that a successful import may still return one or more warning messages, describing potentially unintended results.

6. Issue categorization maps are applied to issues based on their stream. Therefore, to apply the mapping, you must configure your desired stream to apply the newly created map. See Section 3.3.1.2.7, “Selecting an issue categorization map for a stream” for configuration instructions.

Note

Issue categorization maps can be synchronized across a Coordinator/Subscriber cluster. In order to work correctly, however, all subscribers should be upgraded to match the coordinator version. Otherwise, issue categorization may not work properly on all subscriber instances.

3.1.1.6.1. Issue categorization map JSON format

An imported issue categorization map must match a defined JSON format. Each map must contain a "`name`" object, along with a list of "`types`". Each individual `type` object consists of a type name, and an associated "`category`" value, "`impact`" value, or both. This will map the specified `category` and/or `impact` value with all issues of the given `type`.

For example, the following JSON file creates an issue categorization map called "My default map". It maps all issues of type "Allocation size error" to the "Memory - corruptions" category and "High" impact. It also maps all issues of type "Calling deprecated method" to the "Code maintainability issues" category. The built-in impact for "Calling deprecated method" type issues remains unchanged.

```
{
    "name": "My default map",
    "types": {
        "Allocation size error": {
            "category": "Memory - corruptions",
            "impact": "High"
        },
        "Calling deprecated method": {
            "category": "Code maintainability issues"
        }
    }
}
```

Please note the following when creating or editing an issue categorization map:

- To override built-in "category" or "impact" values for any "types", import a map that only provides the new values. This is shown in the example above.
- It is not an error to use type names that Coverity Connect does not recognize, however a warning will be raised upon importing the file. This is to ensure that the issue type is a valid value and not a typo.

  Once imported, those type names will be recognized by Coverity Connect. In future commits, any issues with a matching type will be mapped accordingly.

  Note

  A custom categorization can be tied to either a snapshot or stream. Each stream and snapshot also has its own categorization. Therefore, we recommend that you tie the categorization to a stream, so that the categorization persists for both the snapshot and stream. New changes will be seen only after a commit has been made to that stream, or when a new snapshot has been created.

- The value for "`category`" must be a string between 2 and 256 characters long. You can use a previously known category, or create a custom category.
- Impact value must be "High", "Medium", "Low", or "Audit".
- "`category`" and "`impact`" are both optional, but at least one must be specified for each type.
- Null values are not valid for any field.
- The name shown for each issue type corresponds to a checker shown in the *Coverity 2020.12 Checker Reference* .

3.1.1.7. Configuring sign-in settings

To control user access to Coverity Connect, sign in to Coverity Connect using a web browser as the `admin` user. Then go to Configuration → System → Authentication and Sign In.

To configure sign-in settings:

1. Select from the sign-in options:

   Limit failed sign in attempts to
   Set the number of failed name-password sign-in attempts before the user is locked out. Once this happens, unless password recovery is enabled, the administrator must reset the password for this user. See Section 3.2.1.2.3, “Locking and unlocking a user account” for more information.

   Disable local password authentication
   Entirely disable local account access, and use LDAP for authentication. Requires previous LDAP configuration.

   Allow email password recovery
   If the user is locked out due to incorrect password attempts, the password can be requested using email. Requires previous email configuration.

   Time out signed in users after [N] minutes of inactivity
   Set the number of minutes of inactivity before requiring users to sign in again. Note that this option is permanently enabled. Its default setting is 120 minutes.

2. If you are using LDAP, select from the following options:

   Create LDAP users automatically on sign-in
   Creates users in Coverity Connect upon successful authentication with your LDAP server. Requires LDAP configuration.

   You can also choose to only create users that belong to imported LDAP groups.

   Only create users that are members of imported group(s)
   Provides for backward compatibility with Microsoft Active Directory products that require LDAP users to be members of an LDAP group.

3. Click **Done** to finalize your changes and exit the screen.

Sign In Log
The *Sign In Log* shows a history of user session activity.

Note

If you need to reset the Administrator password, use the `reset-admin-password` option of the `cov-admin-db` command. For more information, see the Coverity 2020.12 Command Reference.

3.1.1.8. Configuring Coverity Connect for TLS/SSL

You can configure Coverity Connect to encrypt communications using TLS/SSL.

This process summarizes steps described in the Tomcat documentation, as well as the Java keytool command. You can refer to this documentation for more in-depth information on configuring the embedded Tomcat server and managing a Java keystore. Refer to the appropriate keytool documentation for the JDK or JRE version that you are using. See Section 3.1.1.8.1, “Commit encryption use cases” for information on committing over TLS/SSL use cases.

Encryption for Coverity Connect works in tandem with the Coverity Analysis client, and some configuration by the user may be necessary. For information on client-side configuration (including the `ca-certs.pem` file) and authentication, see Coverity Analysis Trust Store.

Coverity Connect supports the session cookie flags of `secure` and `httpOnly` to prevent misuse of session cookies. The `secure` flag is only available when using HTTPS, and is on by default.

Cookie settings can be confirmed using the web developer tools in a modern browser. With the web developer tools active, connect to Coverity Connect, and then look for a cookie named `COVJSESSIONID-*`. Check the *HttpOnly* and *Secure* columns.

- *HttpOnly* should always be on. It prevents client-side scripts (such as JavaScript) from accessing the cookie.
- *Secure* should be on if and only if the connection is HTTPS. It prevents the cookie from being sent on an unsecure (HTTP) channel.

Refer to http://tomcat.apache.org/tomcat-8.0-doc/ssl-howto.html for information on setting up TLS/SSL for Tomcat servers like Coverity Connect. Additionally, see http://en.wikipedia.org/wiki/X.509 for background on digital certificates, and https://www.openssl.org/docs/manmaster/apps/x509.html for information on working with certificates using OpenSSL.

To configure Coverity Connect to use TLS/SSL:

The example commands illustrate the general procedure for importing certificates. You may need to alter the parameters depending on the requirements of your certificate authority. The certificate authority may have specific instructions for how their certificates are to be installed on various types of servers.

1. Select one of the following options:
   - Obtain an SSL certificate from a certificate authority and proceed to the next step.

- Use the self-signed SSL certificate created by the installer. It is in the keystore at `<install_dir_cp>/server/base/conf/keystore.jks`. If you use this option, proceed to Step 3.

  Note

  For information on Windows domain certificates, refer to the Microsoft documentation.

2. Create and import a certificate, as shown in sub-steps a through h.

   If you need additional assistance with completing this step, contact `software-integrity-support@synopsys.com`.

   a. Go to the `<install_dir_cp>/server/base/conf` directory.

   b. Rename your existing keystore, for example:

   ```
   mv keystore.jks keystore_bak.jks
   ```

   Renaming the keystore removes it from operation and backs it up. You should keep the backed-up keystore at least until you complete this procedure and are satisfied that the new keystore you create is functioning properly.

   c. Create a new keystore with a new private key. For example (substitute your Coverity Connect server's hostname or IP address for `<hostname>`):

   ```
   <install_dir_cp>/jre/bin/keytool -keystore keystore.jks -storepass changeit -
   alias tomcat -genkeypair \
   -validity 730 -sigalg SHA256withRSA -keyalg RSA -keysize 2048 -dname
    'CN=<hostname>' -keypass changeit \
   -deststoretype JKS
   ```

   Note

   You may need to change the `-sigalg` parameter to match the signature algorithm used by the SSL certificate. For example, `SHA1withRSA`, `SHA256withRSA`, and `SHA1withDSA` are commonly used.

   Note

   The password specified for the `-storepass` parameter must be the same as the password specified for the `-keypass` parameter.

   d. Generate a certificate signing request (CSR) from the private key. The following command creates a file called `csr.pem`:

   ```
   <install_dir_cp>/jre/bin/keytool -keystore keystore.jks -storepass changeit -
   alias tomcat \
   -certreq -sigalg SHA256withRSA -file csr.pem
   ```

   e. Send the CSR to your certificate authority, from which you will receive a "user" certificate.

You might also receive a chain certificate (also called an "intermediate" certificate). If you do, perform steps Step 2.f and Step 2.g

If you only receive a user certificate, only perform Step 2.g.

If your certificate authority sends your user certificate together with chain certificates in PKCS7 format, only perform Step 2.h.

f. Put the chain certificate into a file named `chain-cert.pem` and add the chain certificate to the same keystore that contains your private key:

```
<install_dir_cp>/jre/bin/keytool -keystore keystore.jks -storepass changeit  \
-alias chaincert -importcert -file chain-cert.pem -trustcacerts -deststoretype
 JKS
```

g. Put the user certificate in a file named `cert.pem` and then add the certificate to the keystore using the alias for the existing private key, `tomcat`:

```
<install_dir_cp>/jre/bin/keytool -keystore keystore.jks -storepass changeit  \
-alias tomcat -importcert -file cert.pem -trustcacerts -deststoretype JKS
```

h. Save the certificate blob in a file called `certs.pk7.pem` and add it to the keystore:

```
<install_dir_cp>/jre/bin/keytool -keystore keystore.jks -storepass changeit  \
-alias tomcat -importcert -trustcacerts -file certs.pk7.pem -deststoretype JKS
```

3. Stop the server, for example:

```
> <install_dir_cp>/bin/cov-im-ctl stop
```

4. Enable TLS/SSL on the Coverity Connect server as follows:

Note

If you enabled TLS/SSL during the installation process, skip this step.

Edit the Tomcat configuration file located at `<install_dir_cp>/server/base/conf/server.xml`. Uncomment the `Connector` section, which resembles the following:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443

The default connector configuration obeys all of the following rules:
  - allow only TLSv1.2 connections;
  - the enabled cipher suites must use all of the following:
    - Ephemeral Diffie-Hellman for key exchange
      (optionally, allow RSA for key exchange if necessary for supporting some
      clients);
    - block ciphers with key lengths of at least 128 bits;
    - block ciphers in GCM mode;
      (if CBC mode must be allowed for supporting some clients,
      use only CBC mode cipher suites that use the SHA-2 family of hash
      functions);
  - do not enabled the following cipher suites because there are problems with
    them when they are used with Firefox/Chrome:
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_RSA_WITH_AES_128_GCM_SHA256.

This connector is not compatible with some older CIM clients which only support
TLSv1, (e.g. clients that use JRE 6, because TLSv1.2 is not enabled by default
there).
If you need to enable TLSv1 (such a configuration is considered to be insecure),
do the following:
  - append TLSv1 to the set of enabled protocols, e.g.
    sslEnabledProtocols="TLSv1.2,TLSv1" (do not remove TLSv1.2 from the set);
  - append TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA cipher suite to the set of enabled
    cipher suites.

If you know that your web browser supports TLSv1.2, but it still can not
communicate with this server (this must not happen to web browsers which are
officially supported by this server), please consider appending the following
cipher suites to the set of enabled cipher suites:
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
-->
<!-- Enable this connector to add SSL support. -->
<!--
<Connector port="8443"
    protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="150" ciphers="TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_128_CBC_SHA256"
    scheme="https" secure="true" SSLEnabled="true" sslEnabledProtocols="TLSv1.2"
    keystoreFile="conf/keystore.jks" keystorePass="changeit"
    clientAuth="false" sslProtocol="TLS"/>
```

```
-->
```

Note

Do not copy this example into your `server.xml` file, as there may be other settings that are specific to your installation. Also, note that the long `ciphers` list is important for correcting certain security issues with Firefox and Chrome browsers and TLS v1.2 in Tomcat.

Depending on your use case, you might wish to revise the attribute values in `server.xml`. For example, you might need to disable weak ciphers or protocols, or reference custom resources.

`ciphers`
: A comma-separted list of ciphers to be enabled. You can add or remove cipher names from the list.

`sslEnabledProtocols`
: A comma-separated list of TLS protocols to be enabled. To enable a protocol, add it to the list.

  TLS 1.2 (`"TLSv1.2"`) is enabled by default.

  Note

  As of Coverity 2018.09, the TLS 1.0 protocol is no longer enabled by default. We continue to support it in order to maintain backward compatibility, but please be aware that TLS 1.0 is now considered to be insecure.

`keystoreFile`
: The value here should match the name of the keystore file that you are using.

`keystorePass`
: The password set here must match that used for your keystore file.

If you make changes to `server.xml`, restart Coverity Connect with the `cov-im-ctl` command, and then test that your browser can communicate with Coverity Connect using HTTPS.

5. Edit the Coverity Connect properties file located at `<install_dir_cc>/config/cim.properties`. Set the standard encryption level of incoming commits by adding a line, `commit.encryption=<value>`, to `cim.properties`. Acceptable values for `commit.encryption` are:

`required`
: Each commit connection must be encrypted. Any commit from a `cov-commit-defects` client that declines to open a TLS/SSL connection will be rejected.

`preferred`
: Coverity Connect will open a TLS/SSL connection if required or preferred by the client. If not, the commit will continue unencrypted. This is the default value for `commit.encryption`.

`none`
: Coverity Connect will refuse any client request to connect through TLS/SSL, and will only receive commits in the clear.

6. Restart the server, for example:

   ```
   > <install_dir_cp>/bin/cov-im-ctl start
   ```

7. Inform users to access Coverity Connect using a URL that contains `https://` and the port specified as `<Connector port="<port_number>"` in the server configuration file, instead of previous `http://`. For example, if the insecure URL was previously:

   ```
   http://coverity_server:8086
   ```

   then the new URL (using the previous configuration example) is:

   ```
   https://coverity_server:8443
   ```

   Users can still access the insecure HTTP port unless you explicitly disable it.

8. Once you are certain that Coverity Connect is accessible through the HTTPS URL, disable the insecure HTTP access.

Note

If errors occur, check for the following common mistakes:

- The argument `< -trustcacerts>` was not included on the Java keytool command line.
- The private key in the keystore does not match the public key of the server certificate.

To disable insecure access to Coverity Connect:

1. Stop the server with the `cov-stop-im` command.
2. Open the tomcat `server.xml` file to edit it.
3. Comment out the Connector section that enables the default port 8080, which starts with the following line:

   ```
   Connector port="8080" protocol="HTTP/1.1
   ```

4. Copy certain property specification lines from the default Connector section that was commented out to the TLS/SSL Connector section that was uncommented (see below). You should copy all of the lines that do not specify port usage. Typically, the default lines that you should copy are:

   ```
   connectionTimeout="20000"
   compression="10240"
   compressableMimeType="text/html,text/xml,text/plain,application/json"
   ```

5. Restart the server with `cov-start-im`.

If you are committing to a TLS/SSL-enabled instance of Coverity Connect, use the `--https-port` option instead of `--port`. For more information, see *Coverity 2020.12 Command Reference*.

3.1.1.8.1. Commit encryption use cases

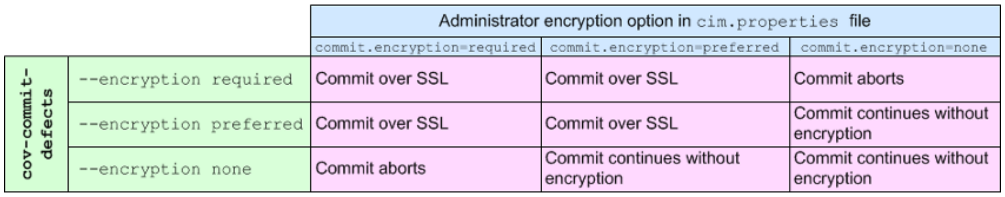
Depending on your standards, Coverity Connect and its clients may need to alter their configurations slightly to allow for committing over TLS/SSL. The typical use-cases for encrypted commits are as follows:

TLS/SSL is not important for sending or receiving commits
: Follow the instructions in Step 5 to set `commit.encryption` to `none`.

TLS/SSL is important to Coverity Connect server or administrator
: Leave `commit.encryption` with its default value of "`preferred`", or follow the instructions in Step 5 to set `commit.encryption` to `required`.

TLS/SSL is required by the user
: The Coverity Analysis developer must set the `cov-commit-defects --encryption` parameter to `required` for each commit.

The following table illustrates the expected outcomes for `cov-commit-defects` based on the encryption values of the user and Coverity Connect administrator.



		Administrator encryption option in `cim.properties` file		
		`commit.encryption=required`	`commit.encryption=preferred`	`commit.encryption=none`
`cov-commit-defects`	`--encryption required`	Commit over SSL	Commit over SSL	Commit aborts
	`--encryption preferred`	Commit over SSL	Commit over SSL	Commit continues without encryption
	`--encryption none`	Commit aborts	Commit continues without encryption	Commit continues without encryption

3.1.1.9. Integrating with LDAP servers

Lightweight Directory Access Protocol (LDAP) is used in organizations to centralize the storage of information common to many applications, particularly related to authentication, such as a user name (ID) or password. LDAP allows site administrators to enter this information only once, and all LDAP-compliant applications can use this central resource.

LDAP is used for authentication of users and for retrieving external properties stored in LDAP that are of interest to Coverity Connect, such as an email address and a user's full name. These attributes are replicated in the Coverity Connect database. Coverity Connect uses LDAP group information.

You can use LDAP authentication for user sign-in so that Coverity Connect does not have to save user passwords locally.

Note

It's important that Coverity Connect users retain their identities even in the event of configuration or organizational changes. If a person's identity is not maintained in Coverity Connect, issues that include that person in triage records (for example, issue owner designations) will appear to have originated with a different person.

LDAP servers maintain user identities. When considering whether to add an LDAP server to Coverity Connect, ask yourself whether the new server will refer to the same persons as the old server. If the answer, generally, is "yes", you should modify the existing LDAP server configuration in Coverity Connect to point to the new server rather than add an additional LDAP server configuration.

3.1.1.9.1. LDAP server requirements

Coverity Connect has the following requirements for LDAP server configuration:

- The LDAP server must support at least one of the following types of bind operations:

  general bind with shared bind DN
  : In this type of bind operation, a shared DN (distinguished name) exists and is allowed to query the LDAP server for user records and group definitions. To use this type of bind operation, the DN's name and password must be defined in the LDAP server configuration screen.

  anonymous bind
  : In this type of bind operation, the LDAP server allows the querying of users and groups without having to authenticate first. To use this type of bind operation, select *Use anonymous bind* in the LDAP server configuration screen.

- Your server must be configured to use case-insensitive user names. Imported LDAP usernames are normalized to lower case characters.

- Coverity Connect only imports first-level LDAP users and groups. For example:

  ```
  Group1
    -user1
    -user2
    -user3
    -Group1A
      --user4
      --user5
  ```

  In this case, Coverity Connect imports user1, user2, user3, and Group1A, but not user4 and user5.

- If an LDAP user and a local user have the same user name, the local user will need to append `@local` to their user name when logging in.

- The SASL (simple authentication and security layer) framework is not supported.

- If the LDAP server is unreachable and an LDAP user attempts to sign in, the sign in will fail and the user will receive the following error:

  ```
  Error during sign in.
  ```

  You can find more information about the failure in the `cim.log` file.

- LDAP usernames can only be between 1 and 32 characters long. User names can include unicode characters starting from U+0020 (space), and up to, but not including, U+007f (delete).

- All required LDAP configuration values must be set in the LDAP server configuration screen before you can save the configuration.

  The LDAP server configuration screen provides links that automatically populate common user and group search settings for Active Directory and OpenLDAP servers.

3.1.1.9.2. Read-only access to LDAP data

Coverity Connect never adds to or modifies the LDAP server data.

3.1.1.9.3. LDAP server types

Coverity Connect provides basic configuration for Microsoft Active Directory and OpenLDAP servers. Other types of servers might require custom configuration values in the LDAP Configuration screen.

3.1.1.9.4. LDAP server configuration overview

To configure LDAP:

1. Configure LDAP settings with Configuration → System → LDAP Configuration. More information.
2. Go to Configuration → System → Authentication and Sign In, and select Authenticate with:`LDAP`. Click **Done** to finalize your changes and exit the screen.
3. Import LDAP users using one of the following methods:
   - Add LDAP users
   - Import LDAP groups
   - Create LDAP users automatically on sign-in

   More information.
4. Synchronize Coverity Connect with data from the LDAP server by re-importing the LDAP group, using the Configuration → Users & Groups menu. More information.

3.1.1.9.5. Configuring LDAP server settings

You configure LDAP server settings for Coverity Connect in Configuration → System → LDAP Configuration.

Warning

Some LDAP server implementations do not allow spaces in configuration settings. To avoid possible problems, do not use spaces in your Base DN, User Search Base DN, and Group Search Base DN settings.

To configure LDAP server settings:

1. Create or edit the LDAP server settings.
   - Click **Add** from the LDAP Configuration screen to create a new configuration.

This action opens configuration pane.

Note

If your instance of Coverity Connect is configured as a subscriber in a clustered Coverity Connect environment, you can not configure the LDAP settings (the **Add** button will be disabled). The coordinator is the only Coverity Connect instance in a clustered environment that can configure these settings. For more information, see Section 3.5.1, “Synchronizing multiple Coverity Connect instances”.

- Click the name of an existing LDAP configuration to open its configuration pane.

2. Provide a display name.

   This is used to identify an instance in a multi-server setup.

3. If there are multiple LDAP server configurations, indicate whether this one is the *Primary server*.

   If checked, this configuration is treated as the primary configuration.

   Note

   When a user logs in with their username only, Coverity Connect attempts to authenticate the username against the LDAP server specified as *Primary* in the *LDAP Configuration* pane. If the username is not found in the Primary LDAP server, authentication fails.
   To log in with a username set in a non-primary LDAP server, the user must log in as `username@other_ldap_server`.

4. Complete the configuration.

   For information about the configuration pane, see:

   - Section 3.1.1.9.5.1, “Connection settings”
   - Section 3.1.1.9.5.2, “Pre-fill settings”
   - Section 3.1.1.9.5.3, “User search settings”
   - Section 3.1.1.9.5.4, “Group Search Settings”

5. Click **Done** to finalize your changes and exit the screen.

   You must set all required LDAP server configuration values before you can save them.

3.1.1.9.5.1. Connection settings

The connection settings are:

Host Name
: The host name of the LDAP server (using the IP address is not supported). The host name must be resolvable from the Coverity Connect host.

Port
The TCP port number where the LDAP server listens for connections (defaults to 389).

Base DN
The LDAP domain name in host format, such as <`domain`>.com or <`corp.domain`>.<`com`>, or base DN such as `ou=corp` .

Note

If you specify the domain as a hostname, it is translated into DN (distinguished name) format, where each element of the hostname becomes a domain component (DC) in the DN. For example, `<corp.domain>.com` becomes `dc=<corp>,dc=<domain>,dc=<com>`.

If this does not match your LDAP directory structure, use DN format.

Security
Specify one of the following security protocols:

- SSL
- TLS

Select *None* if you do not want to use SSL or TLS security.

For more information, see Section 3.1.1.9.6, “LDAP security”.

Use anonymous bind
Specify this option to use an anonymous LDAP connection. If *Use anonymous bind* is enabled, *Bind DN* and *Password* are disabled.

If *Use anonymous bind* is not enabled, you must provide a value for *Bind DN* for an unauthenticated bind request, or you must provide a value for both *Bind DN* and *Password* for an authenticated bind request.

Bind DN
The username needed to access the LDAP server.

You can specify the user in `username@domain` or LDAP DN format.

Bind Password
Bind user password for LDAP user queries. This field is required for authenticated binding requests.

If you do not enter a password, and if your LDAP server is configured to accept unauthenticated binding requests, Coverity Connect will attempt to gain unauthenticated access to the LDAP server.

If an unauthenticated LDAP connection fails, Coverity Connect will display an `LDAP Server Configuration failed` message and an explanation of the failure. You can find more information about the failure in the `cim.log` file.

Test Connection Settings
Click **Connect to LDAP Server** to test the connection.

If the connection fails, there might be a problem with your configuration. You can find more information about the failure in the `cim.log` file.

3.1.1.9.5.2. Pre-fill settings

If you are using Microsoft Active Directory or OpenLDAP servers, you can use the following links to automatically populate standard user and group search settings for these servers:

- **Pre-Fill Microsoft Active Directory Settings**
- **Pre-Fill OpenLDAP Settings**

Make sure that you use the correct auto-fill form for your server type.

If you are using an LDAP server type other than Microsoft Active Directory or OpenLDAP, it might require custom configuration values for the User and Group search settings..

3.1.1.9.5.3. User search settings

The user search configuration settings are:

User Search Base DN
A user search base DN to prepend to the base DN. Used to limit user search.

Note

The base DN is configured as *Domain* as part of the basic LDAP settings.

For example:

```
cn=users,ou=corp
```

If the search base is incorrectly set and an LDAP user attempts sign in, the sign in will fail and the user will receive the following error:

```
Error during sign in.
```

You can find more information about the failure in the `cim.log` file.

User ObjectClass
The LDAP ObjectClass associated with users on your LDAP server. For OpenLDAP, the default is `inetOrgPerson` and for Active Directory, the default is `user.`

Username Attribute
LDAP attribute that maps to the Coverity Connect username.

Given Name Attribute
LDAP attribute that maps to the given name.

Surname Attribute
LDAP attribute that maps to the surname.

Email Attribute
LDAP attribute that maps to the email.

Test User Search Settings
: Coverity Connect will attempt to connect to the LDAP server and will test the filter based on a valid LDAP username that you are required to enter. Click **Connect to LDAP Server** to test the connection. If the connection fails, or if authentication fails, there might be a problem with your configuration.

  Username
  : Enter a known LDAP username from your LDAP server. Coverity Connect searches for this user when you click **Test User Search Settings**.

3.1.1.9.5.4. Group Search Settings

The group search configuration settings are:

Group Search Base DN
: The group search base DN to prepend to the base DN. Used for group searches. The base DN is configured as *Domain* as part of the basic LDAP settings.

  For example:

  ```
  cn=groups,ou=corp
  ```

Groups ObjectClass
: The LDAP `objectClass` value that identifies user groups. For OpenLDAP, the default is `groupofnames`. For Active Directory, the default is `group`. This field defines a component of the LDAP user group search query that Coverity Connect creates.

Group Name Attribute
: The name attribute for the group, for example `cn`.

Member Attribute
: The LDAP attribute that defines the members of a group. Group members can be referred to by their DN or username.

Members stored by
: Setting that indicates whether members of your LDAP groups are stored by their full DN or by the username attribute.

Additional Search Filter
: Additional filter arguments to include certain entries in the subtree and to exclude others. This filter supports standard LDAP query syntax.

  For example, the following entry limits the search to all groups where the name is either `dev`, or starts with `eng-`:

  `(|(name=eng-*)(name=dev))`

  In this case, the group search filter that Coverity Connect creates is:

  ```
  (&(objectClass=group)(|(name=eng-*)(name=dev)))
  ```

Test Group Search Settings

Coverity Connect will attempt to connect to the LDAP server and retrieve the list of groups. Click **Retrieve Groups** to test the connection. If the connection fails, or the list of groups is empty, there might be a problem with your configuration. You can find more information about the failure in the `cim.log` file. If the number of groups returned is greater than 1000, you need to use an additional group search filter to reduce the number of groups.

3.1.1.9.6. LDAP security

For security reasons, LDAP authentication tokens (passwords) are never stored in the Coverity Connect database, except for the special case of the bind DN/password. The bind/DN password is encrypted in the database for security.

3.1.1.9.6.1. Configuring Coverity Connect with SSL and TLS-enabled LDAP servers

In addition to setting the security type during LDAP configuration, Coverity Connect requires certificates to connect to an SSL or TLS-enabled LDAP server. For information on creating certificates and enabling your LDAP server for SSL or TLS, refer to your server's documentation.

Interoperability with SSL/TLS-capable LDAP servers is provided, but only simple authentication is supported. The SASL framework is not supported.

For LDAP SSL/TLS integration, Coverity Connect must be able to find the root CA (certificate authority) public keys (certificates) to verify the signature of the LDAP SSL or TLS server certificate. Coverity Connect uses the Java Secure Socket Extension (JSSE) format to manage certificates and key stores. The J2SE SDK ships with the `keytool` utility, which enables you to set up and work with JSSE digital certificates.

There are two scenarios for certificates for setting up SSL/TLS: LDAP server authentication and client authentication, in which Coverity Connect is the client. In both scenarios, a truststore is required. A basic set of steps for accessing an LDAP server via SSL is outlined below, with additional details available in the subsequent sections:

Note

The examples in the following section use Linux-based syntax. The Windows syntax is different. For example in Linux, the following configuration:

```
-Djavax.net.ssl.trustStore=<install_dir_cc>/config/LDAP_cers.jks
```

Would be represented in Windows as the following:

```
-Djavax.net.ssl.trustStore=<drive>\:<install_dir_cc>\\config\\LDAP_cers.jks
```

A backslash is required in front of the colon (:) and each of the path separators.

1. Set up your certificates

   a. If a truststore is not provided for you, use the `keytool` utility to create the truststore; it may be a randomly chosen file and location. For example:

```
export trst=<install_dir_cc>/config/LDAP_cers.jks
export kt=<install_dir_cc>/jre/bin/keytool
$kt -genkeypair -dname "CN=<some_user>, OU=<People>, DC=<somedom>, DC=<com>" -
keystore
$trst -storepass <store_password> -keypass <key_password> -alias <some_user-
keypair>
```

See Section 3.1.1.9.6.1.1, “Using a truststore” for more information.

b. Import the `<some_user>` client certificate.

```
$kt -importcert -trustcacerts -file <some_user_client.cer> -keystore $trst -
storepass <store_password> -alias <some_user_client_key>
```

c. Import the LDAP server Root CA.

```
$kt -importcert -trustcacerts -file LDAP_Root_CA.cer -keystore $trst -storepass
 <store_password> -alias LDAP_Root_CA_key
```

d. Set the needed info in the `system.properties` file, located in `<install_dir_cc>/config`.

```
java_opts_post=-Djavax.net.ssl.trustStoreType=jks
-Djavax.net.ssl.trustStore=<install_dir_cc>/config/LDAP_cers.jks

-Djavax.net.ssl.trustStorePassword=<store_password>
```

See Section 3.1.1.9.6.1.3, “Client authentication” for more details.

2. Restart Coverity Connect.

3. Create a new LDAP configuration and set the needed login option for the LDAP. See Section 3.1.1.9.5, “Configuring LDAP server settings” for more information.

3.1.1.9.6.1.1. Using a truststore

Both server authentication and client authentication can use the default JSSE truststore. The truststore is in a JKS format file and contains the root or intermediate CA certificates. These CA certificates determine which endpoints are allowed for communication.

Coverity Connect connects to the LDAP server, which presents a certificate that is signed by one of the truststore's CA certificates. Using this truststore, Coverity Connect attempts an SSL handshake with all LDAP servers that present a certificate signed by the CA. JSSE looks for the truststore as follows:

1. If the `javax.net.ssl.trustStore` system property is defined, then the value of this property is used as the truststore's location.

2. If the `lib/security/jssecacerts` file is defined in the `java.home` directory, then the `jssecacerts` file is used as the truststore.

3. If the file `lib/security/cacerts` file is defined in the `java.home` directory, then the `cacerts` file is used as the truststore.

You can also use a custom truststore in the Coverity Connect server to accept a generated certificate. To do so:

1. Use the `keytool` utility to generate the truststore. For more information, see the `keytool` documentation at http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html.

2. Import the certificate into the newly created truststore.

3. Edit the `java_opts_post` property in the Coverity Connect `system.properties` file to define the location of the truststore. By default, `system.properties` is located at `<install_dir_cc>/config`. For example:

   ```
   java_opts_post=-Djavax.net.ssl.trustStoreType=jks \
   -Djavax.net.ssl.trustStore=/openldap-cert/truststore.jks \
   -Djavax.net.ssl.trustStorePassword=adminadmin
   ```

   Note

   Do not use quotation marks (`" "`) to specify a full path or other multiple string values. If a path contains spaces (for example, `Program Files`), use an equivalent path without spaces (for example, `Progra~1`) or change to a location that does not contain spaces.

4. Stop and restart Coverity Connect.

   For more information, see Section 3.1.1.9.8, “Stopping and starting Coverity Connect”.

3.1.1.9.6.1.2. LDAP server authentication

With server authentication, client certificate authentication is not enforced, meaning the LDAP server does not expect to receive a certificate from Coverity Connect for authentication. Coverity Connect verifies that any SSL/TLS-enabled LDAP server is valid. The truststore must be available, as specified in Section 3.1.1.9.6.1.1, “Using a truststore”.

3.1.1.9.6.1.3. Client authentication

With client authentication, Coverity Connect must provide a certificate for authentication. The truststore needs to have the Certificate Authority certificate available in it. Through this truststore, Coverity Connect verifies the server that is responding to the request. Coverity Connect must have a keystore, which contains the private and public key pair.

The separation of the truststore and keystore is not mandatory, but it is recommended. They can be the same physical file.

1. Use the `keytool` utility to create the keystore.

2. Import the certificate into the keystore.

3. Edit the `java_opts_post` property in the Coverity Connect `system.properties` file to define the location of the keystore. By default, `system.properties` is located at `<install_dir_cc>/config`. For example:

```
java_opts_post=-Djavax.net.ssl.keyStoreType=jks \
-Djavax.net.ssl.keyStore=/openldap-cert/keystore.jks \
-Djavax.net.ssl.keyStorePassword=adminadmin
```

Note

Do not use quotation marks (`" "`) to specify a full path or other multiple string values.

4. Stop and restart Coverity Connect.

   For more information, see Section 3.1.1.9.8, "Stopping and starting Coverity Connect".

3.1.1.9.7. Upgrading LDAP server configuration settings

If you upgrade from a version of Coverity Connect earlier than 5.3, it is possible that your LDAP server configuration settings will not successfully upgrade due to your user and group objectClass settings.

In previous versions, the LDAP server configuration screen contained the *User Search filter* and *Group Search Filter* fields which required you to define the respective objectClass as part of the filter definition. As of 5.3, these fields are removed. Coverity Connect now automatically builds the user search filter and group search filter based on the values set in the *User objectClass* and *Group objectClass* fields. The defaults for these values are:

- *User objectClass*: `inetOrgPerson` (OpenLDAP), `user` (Active Directory)
- *Group objectClass*: `groupofnames` (OpenLDAP), `group` (Active Directory)

If your objectClass definitions from a previous Coverity Connect version match the defaults listed above, your LDAP server configuration settings should successfully upgrade. If the objectClass definition do not match, configuration settings will not update successfully. In this case, you will have to manually set User objectClass and Group ObjectClass to your desired value after you upgrade your Coverity Connect server.

To aid you in locating possible mismatches, Coverity Connect writes your LDAP configuration settings to the upgrade log before you upgrade and after the upgrade is finished. If there are conflicts, Coverity Connect alerts you. You can use the upgrade log to locate the mismatch and make the appropriate changes.

3.1.1.9.8. Stopping and starting Coverity Connect

To stop or start Coverity Connect:

- On Linux, go to `<install_dir_cc>/bin` and run the `cov-im-ctl` command.

  The command accepts the `maintenance`, `start`, `stop`, or `status` options. For example, to get status information:

  ```
  > cd <install_dir_cc>/bin
  > ./cov-im-ctl status
  ```

- On Windows, go to `<install_dir_cc>\bin` and run the `cov-im-ctl.exe` program.

When Coverity Connect is installed as a service, the `cov-im-ctl.exe` program is often unnecessary because Coverity Connect starts and stops automatically when the system boots up or shuts down. When Coverity Connect is installed as a service, any administrator can use this program.

When Coverity Connect is not installed as a service, only the user who installed Coverity Connect is able to use this program to start or stop it.

Note

If you need to restart your external PostgreSQL database, see the PostgreSQL documentation.

3.1.1.9.9. LDAP debugging tools

To set up the LDAP environment, customers need to examine their internal LDAP configuration in order to find and test values for *Connection Settings*, *User Search Settings*, and *Group Search Settings*. The following tools are suggested for this purpose.

- `ldapsearch`
- `JXplorer` (see JXplorer download page.)
- `Apache Directory Studio` (see Apache Directory Studio download page.)

3.1.1.9.9.1. Integrating Coverity Connect with Active Directory (AD) Global Catalog

The Global Catalog component of Active Directory is accessed using 3268/3269, and the AD server is accessed using ports 389/636. When connecting to the Global Catalog, the AD attributes typically used by Coverity are not available, and **Test Connection Settings** results in an error.

To avoid this error, edit the `ldap.properties` file in the `<CC_INSTALL_DIR>/server/base/webapps/ROOT/WEB-INF/classes/` directory, as follows:

1. Set the "ldap.ldapValidationEnabled" property to "false". This will turn off LDAP validation on Coverity Connect and thus should suppress the error.

Note

Make sure to verify that you can import users and groups, by using **Test User Search Settings** and **Test Group Search Settings**.

To enable **Test Connection Settings** to work, do the following:

1. Edit the "ldap.connectFilter=(objectClass=classSchema)" line and replace the "classSchema" with a known attribute in your AD catalog.

Note

Use the tools mentioned above to inspect your AD catalog.

3.1.1.10. Configuring Coverity Connect to use Kerberos

3.1.1.10.1. Kerberos Overview

Kerberos is a suite of services that implement Single Sign-On (SSO). Kerberos enables a client and a server (security principals) to authenticate each other securely on an unsecure network, and conduct encrypted communications. The Kerberos protocol is operated by the Key Distribution Center (KDC). The KDC comprises two services: an Authentication Server (AS) and a Ticket Granting Service (TGS).

When a client wishes to access a resource on a server, the client first contacts the KDC and provides an account name and password. The AS accesses Active Directory to verify the credentials, and if verified, grants a Ticket Getting Ticket (TGT) to the client. When the client needs to access a server in the domain, the client submits the TGT back to the Ticket Granting Service of the KDC, which then grants a service ticket and session keys to the client. The client presents the service ticket to the server, which then authenticates the ticket and the client.

The service ticket is typically valid for a day, but the interval can be set by the Kerberos administrator. Each time the user needs to obtain a new service ticket, he will have to execute `kinit`. See Section 3.1.1.10.4.3, “Obtaining an initial Kerberos ticket”.

In Coverity, when Coverity Connect is configured to use Kerberos, it will use Kerberos authentication for LDAP users from LDAP servers associated with the Kerberos server.

Warning

Coverity Connect does not use Kerberos for local users, including the built-in Admin user account.

3.1.1.10.2. Key Concepts

- Keytab

  The keytab file (“key table”) contains one or more entries, where each entry consists of a timestamp (indicating when the entry was written to the keytab), a service principal name, a key version number, an encryption type, and the encryption key itself. The keytab file is generated on each host in the Kerberos realm, and is used by Coverity Connect to authenticate clients.

- Fully Qualified Domain Name

  The fully qualified domain name (FQDN) is the complete domain name for a specific computer, or host, on a network. The FQDN consists of a hostname and a domain name. For example, in `mycomputer.mycompany.com`, `mycomputer` is the hostname, and `mycompany.com` is the domain name.

- Realm

  A Kerberos realm shares a common Kerberos database, and security principals within it can be authenticated to each other. It is conventionally written as an upper-case ASCII string.

3.1.1.10.3. Setup

Coverity Connect uses a keytab file generated on each host to authenticate service tickets from clients. A keytab file can be generated by the Kerberos administrator using the following command:

Generating a Keytab file (Mac/Linux)

1. On the command line, start `kadmin`.
2. To export the keytab file, enter:

   ```
   ktadd host/name_of_service_principal
   ```

3. Exit `kadmin`.

Configure Coverity Connect to use Kerberos using the following procedure:

1. In *Configuration – System*, select *Kerberos Configuration*.
2. For *Kerberos Keytab*, click **Browse** to select the Keytab file.
3. In *Host FQDN*, enter the fully-qualified domain name of the Coverity Connect host server.
4. In *Realm*, enter the name of the realm covered by Kerberos authentication.
5. Click **Validate** to confirm the settings.
6. Click **Done** to save your changes.
7. In Configuration → System → Authentication and Sign In, under *Authentication Method for LDAP Users*, select Authenticate with: **Kerberos**.
8. Click **Done** to save your changes and exit.

3.1.1.10.4. Configuring the Browser

Browsers must be configured to use Kerberos. The procedure varies by browser and operating system. It may be necessary to obtain a Kerberos ticket for the browser as well.

Note

In some networks, a server may be configured to share login credentials with other servers. This is called "credential delegation." Credential delegation can expose the credentials to third parties, which in theory is a security vulnerability. The procedures for configuring browsers for Kerberos include a step for enabling credential delegation. Coverity Connect does not use credential delegation, so unless otherwise needed, consider refraining from enabling credential delegation. The specific settings are indicated in the respective procedures.

3.1.1.10.4.1. Configuring the Firefox browser on Linux and Mac OS

To configure the Firefox browser to use Kerberos:

1. Enter `about:config` in the address bar.
2. In *Filter*, enter `negotiate`.

3. Double-click *network.negotiate-auth.trusted-uris*.

4. Enter the Coverity Connect domain name.

5. Repeat the procedure for the *network.negotiate-auth.delegation-uris* entry, using the same domain.

   Note

   This setting enables credential delegation. It is optional.

6. Double-click *network.negotiate-auth.allow-non-fqdn* to change the *Value* to *true*.

3.1.1.10.4.2. Configuring the Chrome browser on Linux and Mac OS

To configure the Chrome browser to use Kerberos, start the browser from the command line with the following arguments:

```
--auth-server-whitelist="*.example.com" --auth-negotiate-delegate-
whitelist="*.example.com"
```

Replace the `"*.example.com"` with the domain used in your organization.

Note

The setting `--auth-negotiate-delegate-whitelist` enables credential delegation. It is optional.

3.1.1.10.4.3. Obtaining an initial Kerberos ticket

If the client is not in a Windows Active Directory network, then it is necessary to obtain an initial Kerberos Ticket Granting Ticket (TGT). This ticket provides evidence that the client (browser) is authenticated in the network. This procedure will work if the client is in a Kerberos realm.

1. In a command shell, type `kinit` and enter the user password. This retrieves the Kerberos ticket from the KDC.

2. To view the Kerberos tickets on the client, enter `klist`.

3.1.1.10.5. Troubleshooting

Kerberos authentication relies on the proper configuration of various network resources outside of Coverity Connect. If problems occur, the following tips may help you resolve them.

3.1.1.10.5.1. How can I test that my Kerberos-enabled client is working with a Kerberos-enabled web server?

The end to end test involves having network access from the client to the kerberos server and a keytab file for the web server. Settings for the kerberos server and associated realm are specified in the kerberos configuration file (e.g. krb5.conf or krb5.ini).

1. If the initial kerberos token (Ticket Granting Ticket) is required, authenticate to the Kerberos server:

```
kinit kuser1@YOUR-REALM
kuser1@YOUR-REALM's Password:
```

2. Verify that a valid kerberos token (TGT) is available:

```
klist
Credentials cache: API:678E11A7-D99A-45AE-8FBE-C8ED45AD8338
Principal: kuser1@YOUR-REALM
Issued                Expires               Principal
Nov 23 11:03:13 2015  Nov 23 21:03:08 2015  krbtgt/YOUR-REALM@YOUR-REALM
```

3. Send a web request with a kerberos-enabled web client, such as a browser:

```
http://your-server.company.com:8080/
```

3.1.1.10.5.2. Why am I getting errors about unsupported encryption types?

The Kerberos server may be using encryption that is not supported by the Java runtime environment. For information on updating the JRE with more encryption types, see: Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8 Download. In addition, the Kerberos server may be configured to use weaker encryption (that is available in the default JRE) by editing the Kerberos configuration file:

```
# use weak encryption so as not to require the unlimited strength security jars
default_tgs_enctypes = des3-hmac-sha1
default_tkt_enctypes = des3-hmac-sha1
permitted_enctypes = des3-hmac-sha1
```

3.1.1.11. Adding and configuring a reverse proxy server

This section describes how to add and configure a reverse proxy (RP) server in front of the Coverity Connect server.

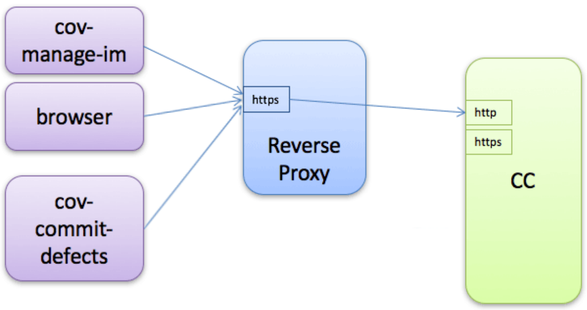
Note

This is a prerequisite for enabling Reverse Proxy Authentication (RPA) in Coverity Connect.

The RP server configuration described in this section can handle requests from, and responses to, any HTTP or HTTPS client. Examples include:

- browsers
- `cov-manage-im`
- `cov-run-desktop`
- `cov-commit-defects`

As shown in the diagram below, requests from these clients are addressed to the HTTPS port on the RP server.



3.1.1.11.1. Setup using Apache HTTP Server

This section describes how to add and configure *Apache HTTP Server 2.4* (apache2) to act as a reverse proxy (RP) in front of the Coverity Connect server. It assumes the reader has some familiarity with reverse proxy concepts, usage, and administration.

Note

If desired, web servers other than apache2 can be used. However, instructions specific to other web servers are not provided.

The configuration described in this section also enables the transformation of HTTPS requests to HTTP requests.

To add and configure the reverse proxy:

Note

In addition to the steps below, note the following:

- If you plan to use RPA, set up the reverse proxy on the same host as the Coverity Connect server.
- Assuming you use the `--host --port` method of connecting, the Coverity Connect implementation can provide service only for RPs for which the “context path” part of the URL is empty. For example, the RP works if it uses the URL `https://coverity.example.com/` for Coverity Connect, but not if it uses `https://example.com/coverity/`. If you wish to use context paths, then you must use the `--url` method of connecting instead of `--host --port`.
- You need to take steps to ensure that clients cannot reach CC directly--that is, by bypassing the proxy.

Note

These steps are for configuring apache2 on Ubuntu. The steps for other platforms are similar.

1. On the reverse proxy host machine, make sure the necessary apache2 modules are installed and enabled:

   a. In the `/etc/apache2` directory, run the following command to get a list of available apache2 modules:

```
sudo a2enmod
```

b. In response to the prompt asking you which module(s) you want to enable, enter the following list of modules:

```
auth_basic authn_core authn_file authz_core authz_host
authz_user env headers log_debug mime mpm_event proxy_http
proxy rewrite setenvif socache_shmcb ssl
```

2. In the `/etc/apache2/sites-available` directory, create a `coverity.conf` file and enter the following lines:

```
Include coverity/ssl.conf
Include coverity/authentication.conf
Include coverity/proxy.conf
```

Note

The `authentication.conf` and `proxy.conf` files are located in`<cim-installer-location>/extras/proxy/apache2`.

3. Enable the `coverity` site:

```
sudo a2ensite coverity
```

4. Remove any unneeded sites from `/etc/apache2/sites-enabled` using the `a2dissite` command.

5. In the `/etc/apache2` directory, create a `coverity` directory:

```
sudo mkdir coverity
```

6. `cd` into the `coverity` directory, and create these configuration files:

   - `ssl.conf`—Modify your Apache SSL (HTTPS) configuration file as necessary to provide SSL service for your environment. If you want to use an RP URL without a port number, SSL service should be provided on port 443.

   - `authentication.conf`—This file is empty if RPA is not used. For details about using this file for RPA, see Section 3.1.1.12, “Adding Reverse Proxy Authentication”.

   - `proxy.conf`—To specify that requests to the RP using `http://<ccserver_hostname>.com/foo` be converted to a proxy request to the Coverity Connect address, `http://localhost:8080/foo`, copy the content from `<cim-installer-location>/extras/proxy/apache2/proxy.conf` and paste it into the `proxy.conf` file.

   Note

   This `<cim-installer-location>/extras/proxy/apache2/proxy.conf` configuration is for the scenario when the RP server is located on the same host as the Coverity Connect server.

7. Restart apache2 to activate the new configuration:

   ```
   sudo service apache2 restart
   ```

3.1.1.12. Adding Reverse Proxy Authentication

Coverity Connect supports Reverse Proxy Authentication (RPA). RPA is an authentication method that can be used to transmit authentication information from a Single Sign-On (SSO) implementation to Coverity Connect.

Note

This section provides detailed instructions for adding RPA and configuring it for SSO. It does not cover adding SSO itself. Instructions for adding SSO are beyond the scope of this guide.

3.1.1.12.1. RPA key concepts

After adding RPA:

- *Only LDAP users are accessible. The built-in admin user is not accessible, so you must assign the System Admin role to an LDAP user before enabling RPA.*

  In Coverity Connect, RPA is one of three “LDAP-only” authentication methods (the others are LDAP and Kerberos) that you can use. An “LDAP-only” authentication method is an authentication method that only works for LDAP users. Only one LDAP-only authentication method can be enabled at a time. However, each LDAP-only authentication method can be used at the same time as one or more non-LDAP-only authentication methods.

- *Authentication and session management is delegated to the proxy. Coverity Connect does not time out users’ sessions. The “log out” menu item is disabled.*

- *Optionally, for Web Services access and cov-commit-defects requests, password-based authentication can be disabled. Authentication keys can be used instead when password-based authentication is disabled.*

  When enabled, RPA authenticates Coverity Connect UI requests. In addition, after enabling RPA, you can optionally disable password-based authentication for SOAP and REST Web services (WS), and for `cov-commit-defects`. Authentication keys can be used instead to authenticate those three services. For more information, see Section 3.2.1.6, “Working with authentication keys”.

  Disabling password-based authentication for WS and `cov-commit-defects` provides assurance that the only way users can authenticate to Coverity Connect is by entering their password at the SSO Identity Provider's screen.

  The disadvantages of disabling password-based authentication for WS and `cov-commit-defects` are as follows:

  - After password-authentication is disabled, users cannot use `cov-manage-im` or desktop plugins to create authentication keys. Instead, users must use the Coverity Connect UI.

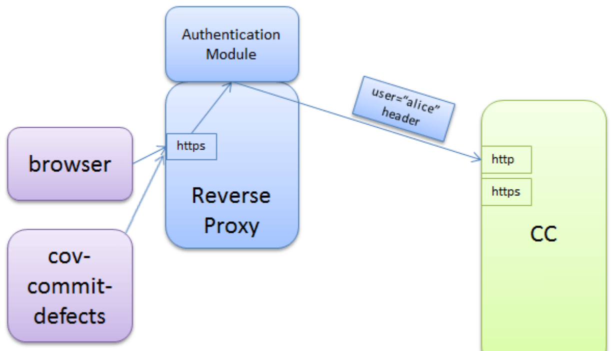
- After password-authentication is disabled, users must store authentication keys on their computer's file system. This makes the keys vulnerable to security attacks unless they are properly protected.

  To disable password-based authentication, add the following line to `<install_dir_cp>/config/cim.properties` and restart the server:

  ```
  authentication.proxy.disable.password=true
  ```

- *Coverity Connect will accept HTTP and HTTPS connections only from clients on the same host as Coverity Connect. The proxy must be on the same host.*

  When RPA is used, authentication of HTTP and HTTPS Coverity Connect UI requests is delegated to the RP. For security reasons, the RP must reside on the same host machine as the Coverity Connect server. After the RP server authenticates an HTTPS request, it forwards the request to the Coverity Connect server as an HTTP request and adds a header to the request. The header lets the Coverity Connect server know that the request has been authenticated, and the header also identifies the authenticated user.



  In addition, with RPA, session management is delegated to the RP server, Coverity Connect does not time-out users' sessions, and the "Log out" menu item is disabled.

- *Verify the forward proxy connection for a successful commit. Client calls to cov-commit-defects are directed to the HTTP or HTTPS proxy server, which are then forwarded on to the hostname specified in the URL.*

  An existing (nonempty) value for the `http_proxy` or `https_proxy` environment variable indicates that a proxy is used. You can use the `printenv` command to see what values exist. (The port default value is 1080.)

  Note

  When `https_proxy` is defined and `cov_commit_proxy` is undefined (or empty), then the commit protocol uses the proxy for its connection, regardless of whether the connection is secure or not. We recommend that you set `cov_commit_proxy` to `none`, so that no proxy is used for the commit protocol.

To ensure a working proxy connection, check for the following:

- Correct syntax for your proxy address.
- Schemes that match the protocol of the environment variable. The port syntax should look like this: `hostname` or `scheme://hostname[:port]`.
- If the `--dataport` option is not set, then an HTTP or HTTPS request is sent to retrieve the dataport before the commit protocol begins. It uses `http_proxy` or `https_proxy` accordingly.

- The `no_proxy` setting overrides and disables the `cov_commit_proxy`, `http_proxy`, and `https_proxy` settings.

3.1.1.12.2. Setting up RPA

This section describes how to set up RPA using basic authentication. You can modify the steps provided such that RPA is set up using your SSO authentication method.

To set up RPA:

1. If you have not done so already, configure a reverse proxy server on the same host machine as the Coverity Connect server as described in Section 3.1.1.11, "Adding and configuring a reverse proxy server"
2. `cd` into the `/etc/apache2/coverity` directory, and replace the contents of `authentication.conf` with the content found inside the `<cim-installer-location>/extras/proxy/apache2/authentication.conf` file.
3. To enable SSO, replace the following line of `authentication.conf` as needed for your SSO protocol:

```
# SSO implementation invoked here.
```

3.1.1.12.3. Configuring Coverity Connect for RPA

After setting up RPA, and enabling and verifying SSO capability, you need to configure Coverity Connect to use RPA.

To configure Coverity Connect for RPA:

1. In a Web browser, enter the URL for the Coverity Connect server (using either the `http://` or `https://` protocol).
2. When the sign-in page opens, log in as the *admin* user.
3. Configure LDAP if it is not already configured. See Section 3.1.1.9, "Integrating with LDAP servers".
4. Choose one or more LDAP users to be administrators, and assign the *System Admin* role to each user. See Section 3.2.1.2.2, "Managing roles for a user".

5. Modify the value of the `web.url` property in the file `<install_dir_cp>/config/web.properties` to be the URL of the RP. For example:

   ```
   web.url=https\://coverity.example.com
   ```

6. Modify the file `<install_dir_cp>/server/base/conf/server.xml` by adding the `address` attribute to all of the `Connector` entities that are not commented out. The `address` attribute must have a value of localhost or 127.0.0.1 as shown below:

   ```
   <Connector port="8080" protocol="HTTP/1.1" URIEncoding="UTF-8"
       connectionTimeout="20000" compression="10240"
       compressableMimeType="text/html,text/xml,text/plain,application/json"
       address="127.0.0.1" redirectPort="8443"/>
   ```

7. Stop and restart the server. For example:

   ```
   <install_dir_cp>/bin/cov-im-ctl stop
   <install_dir_cp>/bin/cov-im-ctl start
   ```

8. In Configuration > System > Authentication and Sign In, in the Authentication Method for LDAP Users section, select Authenticate with: *Reverse Proxy*.

   Note

   The Reverse Proxy option is disabled if the `address` attribute is not added correctly, as described in Step 6.

9. Click *Done* to save your changes and exit.

10. Check the `<install_dir_cp>/logs/cim.log` file for errors that indicate RPA was not enabled.

11. Verify HTTPS browser access by connecting to the proxy. Log in as one of the LDAP users to which you assigned the System Admin role in Step 4.

    Following successful SSO login, you should see the Coverity Connect Web application.

12. In Configuration > System > Authentication and Sign In, in the Authentication Method for LDAP Users section, verify that it still says Authenticate with: *Reverse Proxy*. If it says *LDAP* instead of *Reverse Proxy*, then the changes made to `server.xml`, as described in Step 6, are incorrect or incomplete.

3.1.1.13. Coverity Connect license information

You can check your Coverity Connect license information at Configuration → System → Coverity Connect License. This page shows the status of the terms of your license. This is useful if you are using Coverity Connect as an evaluation and you want to keep track of the usage limitations defined in the license. If you exceed any of the following terms defined in your license, you will no longer be able to use Coverity Connect:

Licensed to
: The name of the organization to which Coverity Connect is licensed.

Allowable users
: The number of Coverity Connect users that you have added. A disabled user is not counted as an allowable user.

Lines of code
: The sum of all lines of code that are analyzed by Coverity tools. Any files that have duplicate names or content are not counted.

Expires
: The date on which the terms of the license agreement expires.

Use the **Import license** button to upload a new license.

3.1.1.14. Analysis license files

You can use Coverity Connect to host your Coverity Analysis license files, and associate them with their respective Coverity Connect projects. This allows Desktop Analysis users to be covered by the analysis license file specified by their associated Coverity Connect project.

To view or update your analysis license files, navigate to Configuration → System → Analysis License Files. This page displays your previously uploaded analysis licenses, and allows you to perform the following tasks:

Default License
: Select your default analysis license from a list of your previously uploaded license files. The default license will be associated with each Coverity Connect project that isn't specifically configured to use another license.

Import...
: Import a new analysis license or Flexnet file. Use the pop-up dialog to choose a file and edit the license name.

Update...
: Upload a license file to replace the selected analysis license.

Delete...
: Delete the selected analysis license. If the default license is deleted, the value in the *Default License* drop-down will change to "None."

Export
: Export the selected analysis license file.

Once you have uploaded your analysis license files, you can associate them with specific projects. To do so, complete the following steps:

1. Navigate to Configuration → Projects & Streams.

2. Select the relevant project from the list in the left pane.

3. Click **Edit** and choose the correct license file from the *Analysis License File* drop-down menu.

Note

All projects not specifically associated with a license file using these steps will use the default license.

3.1.1.15. Configuring email notification and delivery

You can enable Coverity Connect to send email from Configuration → System → Email Configuration. You can now change the default email listed in the *From Address* field of your email configuration settings to send view-specific notification emails. Once enabled, users can schedule periodic notifications to help track issue data in their specific views. See Section 2.4.4, “View-specific email notifications” for more information.

The administrator account can also be configured to use email, but a valid email address must be used. You can view or change the default configuration settings for your account by navigating to Configuration → Users & Groups.

Email configuration consists of the following options:

Allow Coverity Connect to send email
: Select to enable email. This can include name-password information when creating new users. If this is not enabled, all Coverity Connect email functionality is disabled. Selecting this option is required for self-service user password recovery.

Suspend email delivery
: Select to hold all email notification until this attribute is unselected. All email notifications are queued and then sent after the attribute is unselected. This feature is useful, for example, if there is a problem connecting to your mail server, or if your are in the process of changing or updating your users' email addresses.

Host Name
: Specify the SMTP or server name.

Port
: Type a port number for the mail server.

No encryption
: Email is sent without encryption. This typically uses port 25.

SMTP over SSL
: Email is sent by means of SMTP using SSL. This typically uses port 465.

Require STARTTLS
: Email is sent using the STARTTLS protocol. This typically uses port 587 or port 25.

Username
: Specify a username to authorize name/password on the mail server.

Password
Specify a password to authorize name/password on the mail server.

From Address
Specify the email address of the sender.

View notification emails use the email address of the View owner as the sender address.

Plain-Text only
If `On` is selected, Coverity Connect will send emails of Content-Type: text/plain. If set to `Off` CIM: will send emails of Content-Type: text/html. The default is `Off`.

Send Test Email
(Optional) Send an email to test these settings. Provide a destination email address, and click **Send**. If there is a failure, you can find more information about it in the `cim.log` file.

Click **Done** to finalize your changes and exit the screen.

Note

Before you click **Done**, ensure that the browser's autofill function has not entered spurious values in the *Username* or *Password* fields.

3.1.1.15.1. Setting up component subscription notification

In order to allow Coverity Connect to send issue notification based on user component subscription, you must create an external Coverity Connect Web Services script that runs periodically (for example, nightly), executes the appropriate query for users and their component assignments, and then invokes the notification API to send out a message in a predetermined format, such as HTML.

To ensure that notification will work, you should be aware of the following in your Web Services implementation:

- The usernames of all users that subscribe to the component are available in the `subscribers` element in the `componentDataObj` type for Configuration Web Services.
- The `notify()` method in the Administration Web Services works.

For more information, see the *Coverity Connect 2020.12 Web Services API Reference*.

3.1.1.15.2. Setting the maximum included defects in notification emails

By default, notification emails triggered by Coverity Connect include a maximum of 100 defects. To update the maximum number of included defects, add the `notification.maximum.rows` property to the `cim.properties` file, with a value equal to the new maximum.

For example, if you wanted to include a maximum of 50 defects in notification emails, you would add the following line to `cim.properties`: `notification.maximum.rows=50`

3.1.1.15.3. Changing the URL for email notifications

You can change your hostname by specifying the `cim.rootcontext` property in the `<install_dir_cc>/config/cim.properties` file. For example:

```
cim.rootcontext=http://<hostname>:<port>
```

This URL will be used to generate hyperlinks in emails generated by Coverity Connect. If you specify a hostname that is only available on your company intranet, keep in mind that your users will only be able to click on these links from within the intranet.

3.1.1.15.4. Configuring Coverity Connect with an SSL-enabled mail server

If the mail server used by Coverity Connect is protected with SSL authentication, then the mail server's SSL certificate must be added to the Coverity Connect keystore.

1. Obtain the certificate (for example, `exportedCert.cer`) from the mail server administrator.
2. In the Coverity Connect installation folder, run the following command:

   ```
   jre/bin/keytool -import -alias "mailserver" -file exportedCert.cer -
   keystore jre/lib/security/cacert
   ```

3. Restart Coverity Connect.

3.1.1.15.5. Configuring your default email to send view-specific notification emails

If you receive an access denied error message while sending a notification email, you may need to configure your email configuration settings. You also need to modify your `cim.properties` file.

To fix this error message, perform the following steps:

1. Navigate to the right-hand menu in Coverity Connect and click Configuration → System → Email Configuration.
2. Update your `cim.properties` file (via the command line) with the following setting:

   ```
   notify.using.configured.from.address.only=true
   ```

3.1.1.16. Configuring logging

Coverity Connect automatically logs system event information to the `cim.log` file. You can increase the amount of information that Coverity Connect records to this file by enabling additional logging options to work with Coverity Support on an issue. Coverity recommends that you leave all of the logging options un-enabled and only enable them by request from Support.

3.1.1.16.1. Coverity Connect log files

If an error occurs, you can examine the following log files to help diagnose the issue.

`catalina.out`
: Log that is used only in case of a catastrophic Tomcat failure, when the file can contain standard error and output.

`cim.log`
: Basic information about startup, shutdown, and access to Coverity Connect. It also records output from the `cov-manage-im` command. Errors during commits are stored in this log. This log is rotated

daily. The size of the log can grow over time, taking up database space. It is recommended that you monitor the log size and remove logging information as needed.

`cov-admin-db.log`
: Records `cov-admin-db` activity, such as a change to a password or the creation of a database archive file.

`cov-archive.log`
: Records stream export and import actions resulting from the execution of the `cov-archive` command.

`coverity_service.log`
: Windows-only log file that records activity pertaining to Coverity Connect as a service. The file can be present only if Coverity Connect is installed as a service.

`catalina.log`
: Internal information about the embedded Tomcat server. This log is rotated daily.

The log files are written to the `<install_dir_cc>/logs` directory. Note that most Coverity Connect system commands log process and environment data.

3.1.1.16.2. Enhanced Usage Logging

Coverity Connect records events related to user activity in usage logs. In the 8.6 release, Coverity Connect adds additional logging entries focused on user events such as changing user preferences, viewing source files and issues, creating users, and logging in to Coverity Connect.

Events are recorded as JSON-formatted log entries. Each event is recorded on a single line of the machine-readable log file. Log files are stored in the Coverity Connect logs directory. Each log file contains one day's worth of logging, based on the local time zone. Customers may write programs that consume these log files.

The system administrator is responsible for deleting log files as needed. Usage logging is always enabled; there is no provision to turn it off.

The formats of some of the values in the JSON key:value pairs are given in the following table.

Table 3.1.2. Standard value formats

Format name	Format	Example	Notes
datetime	RFC-3339	2016-07-08T17:20:39+0000	Date and time.
username	user@domain	maribel@example	A domain of "local" refers to a user whose repository is Coverity Connect. Domains other than "local" are names of LDAP domains. The @domain portion may be dropped; in this case

Format name	Format	Example	Notes
			any customer program that consumes the log file should assume the domain to be @local.
host	IP address (IPv4 or IPv6)	192.138.8.145	
id	nonnegative number	10014	A low-level identifier for an object.
filename	slash-separated absolute or relative path	path/to/a/file.cpp	

Customer programs that consume the log files should ignore the following:

- unparsable lines
- unexpected types
- unexpected fields

Some pairs appear in every entry, as described in the following table. The “Value type” column refers to a JSON type name.

Table 3.1.3. Standard pairs

Name	Value type	Value format	Value example	Semantics
@type	string		LogInEvent	Describes the semantics of the entry, including what name:value pairs will appear in it.
timestamp	string	datetime	2016-07-08T17:20:39+0000	The time and date at which the event occurred.
userId	number	id	12	Persistent, unique user id of the user executing the operation.
remoteHost	string	host		IP address of the remote host from which the operation was initiated.
ipv6	boolean		false	True if remoteHost is an IPv6 address.

The following events were added to Coverity Connect logging.

- Creation of a user. All user creations are logged.

Table 3.1.4. UserCreatedEvent

Name	Value type	Value format	Semantics
targetId	number	id	User id of the new user.
userName	string		The user name of the new user.

- Deletion of a user. All user deletions, whether local or LDAP in origin, are logged.

Table 3.1.5. UserDeletedEvent

Name	Value type	Value format	Semantics
targetId	number	id	User id of the deleted user.

- Changing of a user's password.

Table 3.1.6. UserPasswordChangeEvent

Name	Value type	Value format	Semantics
targetId	number	id	User id of the user whose password is being changed.

- Change of a user's preference.

Table 3.1.7. UserPreferenceChangeEvent

Name	Value type	Value format	Semantics
setting	string		Name of the user preference setting that was changed.

- The user has changed the project scope for subsequent requests.

Table 3.1.8. ProjectScopeChangeEvent

Name	Value type	Value format	Semantics
projectName	string		Name of the new project scope

- The user has viewed a source file.

Table 3.1.9. SourceFileViewedEvent

Name	Value type	Value format	Semantics

filename	string	filename	Name of the file viewed

- The user has viewed an issue in the context of the source file in which it occurs. Note that other views of issues, such as tables, are not logged.

Table 3.1.10. IssueViewedInSourceEvent

Name	Value type	Value format	Semantics
cid	number	id	Identifier of the issue that was viewed
filename	string	filename	Name of the file viewed

- A session has been created for a user.

Table 3.1.11. SessionCreatedEvent

Name	Value type	Value format	Semantics
sessionId	number	id	Identifies the session

- A session has been deleted.

Table 3.1.12. SessionDeletedEvent

Name	Value type	Value format	Semantics
sessionId	number	id	Identifies the session

- Attempted user authentication.

Table 3.1.13. LogInEvent

Name	Value type	Value format	Semantics
userName	string		The user name of the authenticated user.
protocol	string	one of “GUI”, “WS”, “commit”	GUI means use of the Web UI. WS means use of Web Services. Commit means use of the commit protocol.
logInSucceeded	boolean		True if login was successful.
failureReason	string		If login failed, the reason for the failure.
authenticationSource	string	LOCAL or LDAP or AUTHENTICATION_KEY	How the user authenticated.

- A new role is associated with an entity (project, stream, triage store, or component), user, or group.

Table 3.1.14. RoleAssignmentAddEvent

Name	Value type	Value format	Semantics
entityID	number	id	Identifies the project, stream, triage store or component.
entityType	string	one of “project”, “stream”, “triageStore”, “component”	Type of entity to which the role is being added.
groupName	string		Group to which the roles are being assigned.
roleAssignment	string	id	Role being assigned.
roleAssignments	string	comma separated list of role ids	List of roles being assigned.
username	string		User to whom the roles are assigned.

- A check was done to verify that the current user is not attempting to modify their roles or permissions.

Table 3.1.15. RoleAssignmentCheckSelfOperationEvent

Name	Value type	Value format	Semantics
checkMessage	string		Explanation about the outcome of a check operation.
currentUserId	number	id	User trying the role assignment change.
roleAssignment	number	id	Role assignment being checked to forbid self-changes.
roleAssignments	string	comma separated list of role ids	List of roles assignments being checked to forbid self-changes.
success	boolean		True if check succeeds.
targetUserId	number	id	User for which the role assignment change is attempted.

- Global removal of role assignments.

Table 3.1.16. RoleAssignmentGlobalRemoveEvent

Name	Value type	Value format	Semantics

groupName	string		Group from which the role assignment(s) is being removed.
username	string		User from whom the role assignment(s) is removed.

- Removal of role assignments at the entity level.

Table 3.1.17. RoleAssignmentRemoveEvent

Name	Value type	Value format	Semantics
entityID	number	id	Identifies the project, stream, triage store or component.
entityType	string		Type of entity from which the role is being removed.
groupName	string		Group from which the entity role assignment(s) is being removed.
roleAssignment	string	id	Role assignment being removed.
roleAssignments	string	comma separated list of role ids	List of role assignment being removed.
username	string		User from whom the role assignment(s) is(are) removed.

- A new role was created.

Table 3.1.18. RoleCreateEvent

Name	Value type	Value format	Semantics
role	string	JSON representation of the role: { "roleId": "*roleId*", "name": "*name*", "description": "*description*", "permissions": ["*permission1*", … ,"*permissionN*"]}	Role being created.

- An attempt to update a non-editable role was made.

Table 3.1.19. RoleForbiddenUpdateAttemptEvent

Name	Value type	Value format	Semantics
role	string	JSON representation of the role: { "roleId": "*roleId*", "name": "*name*", "description": "*description*", "permissions": ["*permission1*", … ,"*permissionN*"]}	Role on which the rejected attempt was made.

- A role was updated.

Table 3.1.20. RoleUpdateEvent

Name	Value type	Value format	Semantics
role	string	JSON representation of the role: { "roleId": "*roleId*", "name": "*name*", "description": "*description*", "permissions": ["*permission1*", … ,"*permissionN*"]}	Role being updated: the state after udpate.

3.1.1.17. Collecting and delivering Use and Compliance data (UDC)

Starting in version 5.4, Coverity Connect collects Use and Compliance data from streams on a regular basis. If permitted under the terms of your Product License Agreement (PLA), Coverity Connect also sends a set of this data to Coverity. The PLA sets forth the frequency of data delivery and the types of stream data that Coverity Connect can send to Coverity.

Data Collection

Coverity Connect collects data from the most recent commit to Coverity Connect streams. If the Coverity Connect server is down during the collection period, Coverity Connect skips the data collection process that week and resumes the next week. Data collection takes place on Saturdays.

You can receive notification when Coverity Connect collects data. For details, see Section 3.1.1.17.1, “Administering Use and Compliance data”.

Data Delivery

Coverity Connect delivers the data to Coverity at an interval that is specified in the PLA. Each PLA applies to Coverity commands that analyze source code and commit the results to Coverity Connect streams, for example, the Coverity Analysis `cov-analyze` and `cov-commit-defects` commands. PLAs can indicate which portion of the data collected from the stream, if any, Coverity Connect will deliver to Coverity.

The data is contained in a set of CSV files, which Coverity Connect packages into an encrypted file that it sends to Coverity.

If the PLA allows, you can enable or disable the delivery of data to Coverity. For details, see Section 3.1.1.17.1, “Administering Use and Compliance data”.

Important

For Coverity Connect to email data to Coverity, you must configure and enable Coverity Connect email. Setting up Coverity Connect email also allows Coverity Connect to send data collection and delivery notifications to you. For the setup procedure, see Section 3.1.1.15, “Configuring email notification and delivery”.

The `<install_dir_cc>/logs/cim.log` file provides license information for each commit of analyzed data to Coverity Connect.

3.1.1.17.1. Administering Use and Compliance data

The Use and Compliance Data screen allows you to download data that Coverity Connect sends to Coverity. In addition, some of the fields on the screen, such as *Notification Email*, are editable. However, most are informational only and therefore not editable.

To use the screen:

1. Open the Use and Compliance Data screen.

   Navigate to Configuration → System → Usage Data Collection.

2. If you have not done so already, configure and enable email delivery in Coverity Connect.

   Navigate to Configuration → System → Email Configuration.

   For the setup procedure, see Section 3.1.1.15, “Configuring email notification and delivery”.

3. Determine which tiers of data Coverity Connect can deliver to Coverity:

   - *Compliance data*: Coverity product license compliance data.
   - *Aggregate Usage data*: Basic Coverity Connect usage data, plus Compliance data.
   - *Detailed data*: Additional Coverity Connect usage data, plus Aggregate Usage data and Compliance data.

   Note

   The data tiers are fully described in the PLAs.

   The following data delivery rules are possible:

- *Mandatory*: Coverity Connect delivers a given tier of data to Coverity at the specified interval. This option is not editable.
- *Optional*: You can enable or disable the delivery of a given tier of data to Coverity. This option defaults to Off. You cannot edit the delivery interval.
- *Disabled*: Coverity Connect will not deliver a given tier of data to Coverity. This option is not editable.

  Note

  A more inclusive tier might allow for the delivery of data that is disabled at a less inclusive tier. For example, if an *Aggregate usage* data tier is disabled but the *Detailed usage* data tier is enabled, Coverity Connect will send Detailed and Aggregate usage data to Coverity.

The following delivery intervals are possible:

- *Weekly*: Once per week.
- *Monthly*: Once per month.
- *Quarterly*: Four times per year.
- *Semi-annually*: Two times per year.
- *Annually*: Once per year.

4. Receive notification by email of Coverity Connect data collection and delivery dates.

   - *Notification Email*: Coverity Connect sends data collection notifications to the email address that you provide in the subscription field.

     - To activate notification emails, enter your email address, and click **Done**.

     Upon collection, Coverity Connect will send you one notification per enabled tier (*Compliance*, *Aggregate Usage*, *Detailed*). All Mandatory tiers are enabled. Administrators can enable Optional tiers.

     The notification contains a link to the Use and Compliance Data screen and information about the next date on which the data will be sent to Coverity. It also identifies the PLA that applies to the data.

     Note

     The field will be uneditable if Coverity Connect email is not enabled and configured. For the setup procedure, see Section 3.1.1.15, “Configuring email notification and delivery”.

     If Coverity Connect email was disabled after an email address was added, the disabled subscription field will display that email address. To edit the field, you must re-enable Coverity Connect email.

5. Download data files that Coverity Connect sends to Coverity:

   - Click the **Download** button to download an *unencrypted* zip file that contains the CSV files. These files are generated from the most recent data collection.

     This feature allows you to see what Coverity Connect sends to Coverity. On the specified delivery date, Coverity Connect emails the most current version of this data to Coverity in an encrypted file.

   - Click the **Download Encrypted** button to download an encrypted file that contains CSV files from the most recent data collection. The encrypted file is used for official Coverity audits.

   Note

   If data delivery is disabled for all tiers, neither button will appear.

   If the buttons are not visible even though delivery is enabled for at least one of the tiers, then no data files are available for download yet. This scenario can occur when you commit data to a new Coverity Connect stream. The buttons will appear after the first data collection from the stream takes place.

3.1.1.18. Enabling Issue Export by URL or Export Defect Handler

Issues can be exported from Coverity Connect to an export defect handler program, a URL, or a JIRA instance. When any of these three methods is configured, the **Export** button displays in the *Triage* pane. It is recommended that you create a URL to receive and display the exported defect data. However, you can also create an `export-defect-handler` program to do the same thing. Instructions for both configurations are listed below. For information on JIRA configuration, see Section 3.1.1.3, “Integrating with Jira”.

If multiple issue export methods are configured, then the first one in this order is selected: `export-defect-handler`, URL, or JIRA.

Note

The **Export** button will not be displayed in the *Triage* pane if JIRA, a URL, or an export defect handler program is not properly configured.

To export issue data to your URL (recommended method):

1. Edit your `cim.properties` file to include the following properties:

   - `export.issue.url=https://<yourURL>/?mergedDefectId={mergedDefectId}&projectId={projectId}`

   - `export.issue.request.confirmation=true`

2. Restart Coverity Connect.

   To restart, run `<install_dir_cc>/bin/cov-stop-im`, then run `<install_dir_cc>/bin/cov-start-im`.

When the user clicks **Export**, Coverity Connect opens a new window to your configured URL, with the current defect data displayed. The `{mergedDefectId}` and `{projectId}` variables in the `export.issue.url` parameter will be replaced with their respective exported values. You can also include additional variables at the end of the URL, separated by ampersands (&).

For more information on available issue data variables, see the export XML file in Chapter 3.9, *Exported defect output*.

To create a program to handle an exported issue XML file:

1. Name your program `export-defect-handler`.

   On non-windows systems, do not use a file extension. On Windows systems, the program must have an extension, such as `.com`, `.exe`, or `.bat`. It can be any extension, but make sure that the extension is in the `PATHEXT` environment variable so the program can run.

2. Copy the program to the following directory:

   ```
   <install_dir_cc>/bin
   ```

3. Restart Coverity Connect.

   To restart, run `<install_dir_cc>/bin/cov-stop-im`, then run `<install_dir_cc>/bin/cov-start-im`.

   The **Export** button is enabled when Coverity Connect locates `export-defect-handler` upon start-up.

When the user clicks **Export**, Coverity Connect runs the `export-defect-handler` program and it passes the exported XML file name as the first argument. What `export-defect-handler` writes to `stdout` is written to the `<install_dir_cc>/logs/cim.log` file. If the program returns a non-zero status, the program writes to `stderr` which is also written to the `<install_dir_cc>/logs/cim.log` file.

For more information, see the export XML file in Chapter 3.9, *Exported defect output*.

3.1.1.19. Disabling project dashboards in the Coverity Connect interface

By default, Coverity Connect displays graphs and charts that provide an overview of the state of the issues in the currently selected project. Coverity Connect provides the following Dashboard views:

- The Quality charts graph data on issues found with quality-related checkers.
- The Security charts graph data that is related to security-related checkers.
- The Test Advisor charts graph data that is related to Test Advisor policy violations.

Coverity Connect allows you to remove the display of the dashboards from the Coverity Connect interface. To remove the dashboards, add the following variables and definitions to the `<install_dir_cc>/config/cim.properties` file:

For Quality:

```
display.quality=false
```

For Security:

```
display.security=false
```

For Test Advisor:

```
display.ta=false
```

After you edit the properties file, restart Coverity Connect.

3.1.1.20. Specifying commit options

You can use the following properties of `cim.properties` to modify the queue-related behavior for commits of analysis results to the Coverity Connect database:

- `commitWorkQueueCapacity`: Specifies the number of commits that can wait in the queue. Minimum 15. Maximum 255. Default 80.
- `commitPoolThreads`: Specifies the number of concurrent threads to process commits off of the queue. Minimum 1. Maximum 50. Default 5.

Note

Prior to 6.5.1, `commitWorkQueueCapacity` had lower Max/Default values, which were: Minimum 15. Maximum 100. Default 20.

3.1.1.21. System Diagnostics

System Diagnostics provides information about the current health of the Coverity Connect server, and gives access to system logs. The function is available only to administrative users. It is most helpful for diagnosing performance problems in conjunction with assistance from Coverity Support. In that case, the command-line tool cov-support may be used to gather support information. See `cov-support` for more information.

To access the feature, select *Help*, then select *System Diagnostics*.

3.1.1.21.1. System Diagnostics Overview

The *System Diagnostics* page provides the following features:

- Check the current operating status of the server

  The *Overview* tab displays information and statistics about the current operating condition of the Coverity Connect server. The data are updated in real time.

- Check the load on the system

  The *Graph* tab provides five different types of graphs, each measuring different processes that contribute to or indicate system load. The graphs will update in real time if the check box is selected, as long as the tab remains open.

- Check the status of the cluster

  If the Coverity Connect server is a subscriber or coordinator, the *Cluster* tab displays information about the cluster status.

- Check user information

  The *Users* tab displays information about several categories of users. If the number of allowed users has been reached, this tab will help you know which ones to delete.

- Check database information

  The *Database* tab displays information about database size and current query.

- Check configuration files, environment variables, and Java properties

  The *Config Files* tab displays a list of configuration files, environment variables, and Java properties used by Coverity Connect. The contents of the files can be viewed by clicking on each item, or the files can be downloaded as a set.

- Check logs and log information

  The *Logs* tab displays a list of log files used by Coverity Connect. The contents of the files can be viewed or downloaded.

3.1.1.21.2. Custom Database Queries

An administrator can create custom database queries and set them to appear in the Database tab. The queries are entered into a file as a YAML document, with the format shown in this example:

```
queries:
       - name: A
         sql: select * from users
       - name: B
         sql: select * from project
       - name: C
         sql: select * from user_group
```

Note

Do not perform any data definition language (DDL) or data manipulation language (DML) operations on the content of a Coverity Connect database unless Coverity Support specifically instructs you to do so. Otherwise, all of your Coverity Connect data may become unusable and unrecoverable. This restriction applies whether you are using the embedded database or an external one.

Coverity Support will not assist you in the recovery of data that gets corrupted due to such an update of the database.

For more information about YAML, a type of markup language, see http://www.yaml.org/start.html.

To create a custom query:

1. Create a text file using the format shown above.

2. Save the text file in the installation directory of Coverity Connect, with the name `customQueries.yml`.

3. Select the **Query** drop-down on the **Database** tab, and select the name of a custom query.

3.1.1.22. Securing Coverity Connect

There are several features of Coverity Connect that are used to control access to the system and protect the critical data it contains. Review the following sections to learn how to enhance system security in your data environment.

Configuring Coverity Connect to use SSL
: SSL provides authentication and encryption between clients and servers. See Section 3.1.1.8, “Configuring Coverity Connect for TLS/SSL”.

Integrating with LDAP servers
: LDAP provides centralized user authentication and management. See Section 3.1.1.9.6, “LDAP security”.

Configuring Coverity Connect to use Kerberos
: Kerberos provides authentication among clients and servers in a realm. See Section 3.1.1.10, “Configuring Coverity Connect to use Kerberos”.

Role-Based Access and Control (RBAC)
: RBAC provides detailed control over users and what parts of the Coverity Connect system they have access to. See Section 3.2.3, “Roles and role based access control”.

To operate Coverity Connect in a safety- and security-conscious manner, please keep the following tips in mind:

- Make sure physical access is restricted to the machine where Coverity Connect is installed.

- Verify that file permissions for installed files (including Postgres files) are set only to the Coverity Connect user. Note that the installer correctly applies permissions to all files and directories during the initial install. It is not recommended that these privileges be modified as it may affect the security of the application.

- Use standard IT password rules for users, such as long and complex passwords, and requiring that passwords be changed periodically.

- Note that Coverity Connect will not run under the root or Administrator accounts. The installer requires a dedicated user account on Unix-like systems and should refuse to install or start if attempted to run under root. On Windows, by default Coverity Connect uses a Service Account when configured to start as a service. Although the installer requires Administrator privileges to run, this requirement addresses installation location privileges only, not the privileges required to run Coverity Connect.

- It is highly recommended that Coverity Connect be configured to use HTTPS using a corporate SSL certificate authority.

- Application configuration files ($CIM_HOME/config) contain sensitive information. By default, the $CIM_HOME/config/pgpass and $CIM_HOME/config/cim.properties are accessible only by the account running CIM and administrators (root/Administrator).
- When installing Coverity Connect with the embedded DB option, the database listens exclusively to `localhost` and does not accept network connections. Only a single application account is configured for accessing the database. This mechanism should not be subverted by adding additional users. If this functionality is required, an external database should be used with configuration vetted by a certified database administrator (DBA).
- For even stronger security, use system-level access logging (i.e. syslogNG, Splunk, etc.) to report when connections are made to the Coverity Connect account.
- To ensure that configuration files are not modified without notification, customers can use a tripwire application to watch the configuration directory (`$CIM_HOME/config`).
- To protect data on the drive from being read by untrusted third parties in the event the drive is moved, make sure to use hardware which supports the Trusted Computing standard.

3.1.2. Configuring and managing the embedded database

This section describes database maintenance operations for the embedded database in a Coverity Connect *stand-alone deployment*.

Note

For additional information specific to managing databases in *clustered deployments*, see Section 3.5.2, “Managing multiple database instances”

3.1.2.1. Important notes

- Do not perform any data definition language (DDL) or data manipulation language (DML) operations on the content of a Coverity Connect database unless Coverity Support specifically instructs you to do so. Otherwise, all of your Coverity Connect data may become unusable and unrecoverable.

  DDL operations include SELECT, UPDATE, INSERT, or DELETE. DML operations include DROP INDEX, CREATE INDEX, DROP TABLE, CREATE TABLE, CREATE FOREIGN KEY, and others. This restriction applies whether you are using the embedded database or an external one.
- Coverity Support will not assist you in the recovery of data that gets corrupted due to such an update of the database.
- Use the backup and restore solution selected by your company to backup and restore an *external* PostgreSQL database in a stand-alone deployment.
- Backups can take a long time for large databases. In addition to using the `--dir` option to create directory backups, it is recommended that you place the backup file or directory on a device with fast

read/write speeds (such as a solid state disk). The use of a fast file system is also recommended as it may affect performance when dealing with a large number of files.

- Restores can also take a long time. Backup files, databases, or directories can be restored more quickly if you use a faster file system.

3.1.2.2. Maintaining the size of the database

With each commit to Coverity Connect or any additions to the number of Coverity Connect entities (users, groups, components), the size of your database will grow. The rate that the database grows is based on many factors including the number of defects and snapshots (which will vary for every organization) and the number of lines of code in a given project. Because of this, predicting the size of database over time is extremely difficult, however is it important to both monitor and maintain the size of your database over time.

The database password is located in the `<install_dir_cc>/config/cim.properties` file.

3.1.2.2.1. Database optimization

To maintain the size of your embedded database, run the command `cov-admin-db optimize`. This command should be scheduled to run nightly on databases that regularly see heavy commit traffic. The command vacuums and analyzes the database, which compresses it and updates the query planner data.

For more information, see the `cov-admin-db` description in the *Coverity 2020.12 Command Reference*.

3.1.2.2.2. Purge snapshot details

The Purge Snapshot Details feature allows you to schedule a clean-up process in which Coverity Connect automatically removes snapshot information that you might no longer need if more current snapshot information is available. This feature is recommended because it allows you to reduce and maintain the size of a Coverity Connect database. This feature is also recommended instead of deleting snapshots, if you have been using snapshot deletion in previous versions to save database size. Snapshot purging is faster, more efficient, and preserves your snapshot and triage history.

This feature is implemented as follows:

- During the installation of Coverity Platform - This installation option sets a preconfigured interval for this feature. For more information, see the *Coverity 2020.12 Installation and Deployment Guide*.

- Through the Coverity Connect administration configuration menu - This allows you to customize your interval settings. These configuration setting override the installation settings if you chose to implement them. For more information, see the *Coverity Platform 2020.12 User and Administrator Guide*.

3.1.2.2.2.1. Configuring Purge Snapshot Details

This section describes purging snapshot details throught the Coverity Connect adminstration configuration menu.

Note

For considerations when purging snapshots during or after an upgrade, see Section 3.1.2.2.2.2, “Important upgrade notes”.

Note

It is highly recommended to back up and restore your database after the purge process completes.

To configure snapshot detail purging, define the following:

Scope:
Allows you to designate the purge snapshot details process based on their age (in days) AND identifies the number of snapshots within a stream that will not be removed. Coverity Connect will not remove snapshot details for the most recently committed snapshot, so the value to designate the number of snapshots per stream must be one or greater.

If you chose to enable automatic snapshot details deletion during your Coverity Connect installation process, the scope is set for snapshots that are older than 120 days and to keep snapshot details for 5 snapshots per stream. These settings can be changed at any time.

Schedule:
Allows you to choose the days and time of day that the snapshot purge process will run. If you chose to enable automatic purging during your Coverity Connect installation process, the snapshot purging schedule is set for every day at 5:00 AM. The time value is set in 24-hour notation.

- The purge snapshot details feature removes the following snapshot data (and not the entire snapshot itself):

  - out-of-date source code
  - source code symbol references (xrefs)
  - function metrics and instances

  Because the Purge Snapshot Details feature does not remove the entire snapshot, you can still view some information for a CID that occurred in a "purged" snapshot. For example, if you view a CID that is in the FIXED state that belonged to a snapshot that has been cleaned, you can still see some of the basic information (such as triage states). However, you are not able to see the source code in which the issue occurred or any function data.

- The snapshot purging process is global to all streams within a given Coverity Connect instance. If you have an enterprise cluster deployment, the snapshot purging configuration for a given Coverity Connect instance is not shared with other Coverity Connect instances within the cluster.

- This feature removes rows from the database table. To reclaim disk space, you need to back-up and restore the database.

- In Coverity Policy Manager, if you have changed the configuration and it needs to fetch the existing history (and the history includes purged snapshots) of a stream, the following information is not available (and thus not displayed in Coverity Policy Manager):

- Code coverage numbers (they will be set to 0)
- Function count
- CCM information

3.1.2.2.2.2. Important upgrade notes

If you have enabled this option during the upgrade process or if you have configured the schedule after the upgrade, it is important to note that the first time that the clean-up process runs it might take a long to complete and might result in performance degradation due to the following:

- If your pre-6.5.1 Coverity Connect database contains a large amount of deleted snapshot data.

  The clean-up process that is executed by the Purge Snapshot Details feature performs a full search and removal (garbage-collection) of all of the old and superficial information that was not handled by snapshot deletion in previous releases. This only occurs the first time the clean-up process is executed.

- If your build and commit processes are scheduled at the same time as the Purge Snapshot Details process.

  While the snapshot purging process is running, commits might be significantly delayed, especially if some of the commits take a long time to run.

To decrease the time of the clean-up process and to avoid performance degradation, it is highly recommended to reconfigure the default settings (if set) and schedule "incremental" clean-up processes separately from your scheduled build/commit process.

The first run should specify a date range of older snapshot data. Subsequent processes should specify newer date ranges until the process time is manageable.

For example, assume that your pre-6.5.1 database is 1095 days (3 years) old and you do not have any large commits scheduled to run during the weekend (Friday night through Sunday night):

Note

The values in the following steps are examples and are not suggested values to be used in a real deployment. If you need help to determine a work-able clean-up schedule based on the size and age of your database, contact Coverity Support.

1. Set the Removes information from snapshots that are older than number> days to 730 and schedule the clean-up process to run on Friday night at 23:59.

   When the process runs at the scheduled time, Coverity Connect will remove snapshot information that existed from (day age) 1095 through (day age) 730.

2. After the clean-up process completes, set the Removes information from snapshots that are older than <number> days to 365.

   The clean-up process will, again, run at the scheduled time and will remove the snapshot information that existed from (current day age) 730 to (current day age) 365.

3. Continue this process once a week, lowering the Removes information from snapshots that are older than <number> days number until the setting has reached the level at which you want it set long-term (the recommended number of days is 120, which is the default).

To ensure that the entire Purge Snapshot Details process is complete, see the cim.log file. Completion status is displayed as follows:

For the snapshot details purge process:

```
Starting snapshot details purge for {number} streams
Finished snapshot details purge for {number} snapshots in {number}ms
```

For the garbage-collection process:

```
Starting garbage collection batch
Finished garbage collection batch
```

Note that while garbage collection is running, the above message are continually printed (not just once). You can identify that the process ends when the "finished" message is not followed by a new "started" message.

3.1.2.3. Backing up the database

This section describes how to backup the embedded PostgreSQL database in a Coverity Connect *stand-alone deployment*. It covers using the Coverity Connect UI or the command line to perform or schedule the backup.

Note

For information specific to backing up databases in *clustered deployments*, see Section 3.5.2, “Managing multiple database instances”

As you would do with any system that contains important data, you should develop a process for regularly backing-up and restoring the database. The database contains all of the sensitive data for your system, including source code and defects, so choose an appropriate location when creating copies for back up purposes.

It is up to you to decide the backup schedule (and this might depend on how large the database is and how long the backup takes), but it is a good idea to always have a relatively recent backup. For example, if anything were to go wrong with your system at some point, you will have a successful backup that you can restore into production. It is also important to make a backup of your database when you make changes to the system, such as new feature implementation, major system configuration changes, major tuning changes, upgrades, and so forth.

Once you have a process for regular backups, you should also implement a process for periodically testing the backups by attempting to restore them to an instance of the Coverity Connect server that is the same version the backup was created from. This practice will verify the media reliability and information integrity in the event restoring a backup is required.

3.1.2.3.1. Important notes

- Commit and backup can now run in parallel. There is no risk to the backup if a commit is in progress while the backup occurs. If restored, such a backup will show a partial commit, which will be removed by the next commit to the affected stream.
- Backups work the same whether performed using the Coverity Connect UI or the command line.
- The backup does not store a property that is used by Coverity Connect to retain passwordsset up for Coverity Connect email or for LDAP or Jira integrations. If you are using any ofthese features, you should keep a backup of the `cim.ldap.key` value (in `<install_dir>/config/cim.properties`) in an accessible location. Otherwise, you will need to re-enter the passwords if you need to restore a backup of the database to a different instance of CoverityConnect.
- Any errors are written to `cim.log`. Within the file, backup statements are denoted by BackupJob.

3.1.2.3.2. Using the Coverity Connect UI

The Coverity Connect Configuration menu provides a menu item for making or scheduling a backup of the current state of your database. This feature only supports backing up an embedded database. External database backup is not supported by this feature.

Note

If Coverity Connect is running on a Windows computer as a service, it will not be able to backup to a drive letter. Drive letters are mapped on a per-user basis and the local service user does not have the mappings. It can, however, backup to volumes using UNC paths. For example:

```
\\server\volume\path\to\backup
```

To make or schedule a database backup in a stand-alone deployment:

1. Before you perform or schedule a backup, enter the directory location in which you want the backup to be saved. The default directory is:

   ```
   <install_dir_cc>/backup
   ```

   When the backup completes, the backup file is saved with the following naming convention:

   ```
   <date>,<time>.backup.
   ```

2. Do one of the following:

   - To perform a backup, click the *Backup Now* button.
   - To schedule a backup, choose one or more days and set the time of the backup and click *Done*.

   When the backup is started without any errors, Coverity Connect displays a Success message.

3.1.2.3.3. Using the command line

You can manually run (or create a script to automatically run) `cov-admin-db backup` to backup your database. For more information, see the cov-admin-db description in the *Coverity 2020.12 Command Reference*.

To backup the database in a stand-alone deployment:

1. Verify that the Coverity Connect database is running. To check the status of your database, type the following command:

   ```
   > cov-im-ctl status
   ```

2. Back up your database by entering the `cov-admin-db` command. For example:

   ```
   > cov-admin-db backup daily_cim_database_backup
   ```

3.1.2.4. Restoring the database

Once you have a process for regular backups, you should also implement a process for periodically testing the backups by attempting to restore them to an instance of the Coverity Connect server that is the same version the backup was created from. This practice will verify the media reliability and information integrity in the event restoring a backup is required.

Note

For information specific to restoring databases in *clustered deployments*, see Section 3.5.2, “Managing multiple database instances”

To restore the database in a stand-alone deployment:

Note

Use caution when restoring a database with `cov-admin-db`, because it deletes data in an existing database. For more information, see the `cov-admin-db` description in the *Coverity 2020.12 Command Reference* and the `pg_restore` documentation at `http://www.postgresql.org/docs/8.4/static/app-pgrestore.html`.

1. Place the embedded database in maintenance mode using the `cov-im-ctl` command. For example:

   ```
   > cov-im-ctl maintenance
   ```

2. Use the `cov-admin-db` command to restore the database from an archive file. For example:

   ```
   > cov-admin-db restore daily_cim_database_backup
   ```

3. Start Coverity Connect using the `cov-im-ctl` command. For example:

   ```
   > cov-im-ctl start
   ```

3.1.3. Client-side SSL Certificate Management

3.1.3.1. Overview

Secure Sockets Layer (SSL) is a suite of industry-standard protocols for creating encrypted connections between servers and applications. It is used in Coverity for a variety of applications. This section focuses only on clients (such as `cov-commit-defects` and `cov-security-report`) of Coverity Connect.

SSL is important because it makes possible encryption and authentication. Encryption allows a client and a server to communicate across an open network, with the assurance that anyone monitoring the communication will not be able to understand it. Authentication allows the client to know that it is actually communicating with whom it expects, and not with an impostor.

The name “SSL” actually refers to an older version of the software. Transport Layer Security (TLS) is the name of the newer version, but it is also referred to as SSL/TLS, or just SSL.

3.1.3.2. SSL Key Concepts

Certificates

SSL uses digital certificates. Certificates are passed from servers to clients in order to authenticate the server and create encrypted connections. Certificates are data files that contain information such as the Subject and Issuer of the certificate.

The Subject identifies the party that proffers the certificate. In effect, “this is who I am.”

The Issuer identifies the party that asserts the veracity of the Subject. In effect, “this is who vouches for me.”

The certificates are digitally signed, which achieves two goals:

- Makes the certificates attributable: you can confirm whom they come from.
- Makes the certificates non-forgeable: anyone can verify that the contents are exactly as the issuer intended.

Certificate Authority

A Certificate Authority (CA) is an entity that digitally signs the certificates it issues. When a Certificate Authority issues a certificate, it is making the public claim that it has investigated the party named in the certificate’s Subject, and found it to be authentic. When a CA signs a certificate, it puts its own identifier in the Issuer field.

Self-signed certificates

A self-signed certificate has the same Issuer and Subject. In effect, it says “I vouch for myself.” This feature implies that a self-signed certificate can act as its own CA certificate in an SSL handshake (see below).

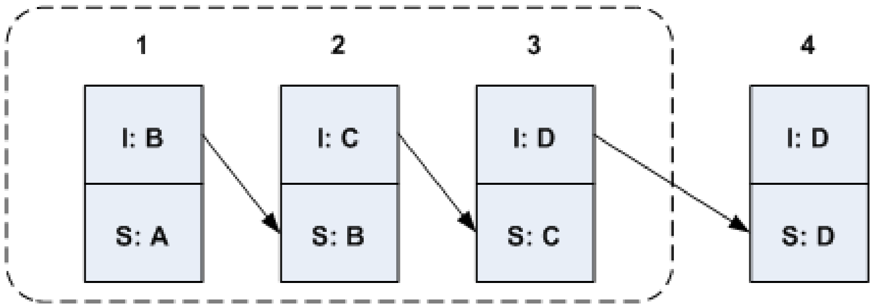
It can be used to establish encrypted communications between a client and server, but it does not provide authentication. However, within a secure corporate network, the convenience of self-signed certificates may outweigh the absence of authentication.

Certificate chain

A certificate chain is a series of one or more certificates returned by the server. Each certificate has a Subject, “who I am”, and an Issuer, “who vouches for me.” The Issuer of one certificate is the

Subject of the next certificate in the chain. The chain links the certificate from the server itself (called the server certificate) back to a certificate issued by the Certificate Authority. A client receives this certificate chain, and verifies that each certificate vouches for the next one. The final certificate, called the CA root certificate, is a self-signed certificate. If the client "trusts" the CA root certificate (explained below), the chain is called a “chain of trust.”

Figure 3.1.3. SSL Certificate Chain



In the figure, each numbered box represents a certificate, with an Issuer (I) and a Subject (S). Certificate 1 is the Server Certificate. The letters A, B, C, and D represent entities that issue or are the subject of certificates. Certificates 1, 2, and 3 comprise the certificate chain sent by the server to the client; certificate 4 is the CA root certificate.

SSL Handshake

To establish a connection, the client contacts the server. Following that, the client and server exchange messages that verify the server’s identity and set up an encrypted channel. This exchange is called the "SSL handshake."

The first part of the handshake is to authenticate the server. First, the server sends its certificate chain, except for the CA root certificate, which the client supplies. The client must construct a verified “chain of trust” starting with the server certificate and ending with a CA root certificate. The client application verifies each certificate in the chain. Since the CA root certificate is implicitly trusted by the client, and that trust flows from the CA root certificate to the server certificate, the client can trust that the server’s identity is truly what is stated in the server certificate’s subject.

After establishing a chain of trust, the client verifies that the server hostname in the server certificate Subject field matches the hostname given by the user in the `--host` option or connection page. This match must be exact, except for letter case. For example, if `cov-commit-defects` has `--host foo` but the server certificate has `foo.example.com` in its Subject, the handshake will fail. This step is called "hostname verification" and is the last step in the authenticating the server.

Note that a server with a self-signed certificate can put anything it wants in the Subject field. Coverity software therefore does not do hostname verification when accepting self-signed certificates.

Trust Stores

A client application knows it can trust a CA root certificate because it fetches the certificate from a trusted location within itself or in its environment. These trusted locations store collections of CA root certificates, and they are called trust stores.

The browser is a common Coverity Connect client that comes preconfigured with a trust store containing CA certificates from the largest CA vendors. The browser checks the server's certificate chain against the certificates in its Trust Store to find one that completes the chain. If a certificate is found in the trust store that completes the chain, then the client authenticates the server, and an SSL connection can be established.

3.1.3.3. Server Certificate Use Cases

Server Certificates fall into three categories according to their origin and how they are handled by Coverity client applications:

1. Self-signed certificates: These certificates are generated by the server administrator. Also, Coverity Connect is preconfigured by the installer with a self-signed certificate. Coverity client software lets the user decide whether to trust a self-signed certificate.

   Note

   Port redirection is not supported with the default self-signed certificate `server/base/conf/server.xml`.

2. Well-known Certificate Authority: the client is preconfigured with these CA certificates in the trust store already.

3. Private Certificate Authority: the administrator has to put the private CA certificate into a trust store that clients can use. Well-known CAs are inherently less secure than private CAs, because they are well known. For this reason, many organizations establish private CAs to provide certificates for servers on their internal networks.

3.1.3.4. Improvements in Certificate Management

Prior to Coverity 8.0 there were various difficulties in managing certificates. Coverity 8.0 solves these difficulties.

- Few Coverity SSL client applications accept self-signed certificates. Other applications needed to be provided with certificates in other ways. In Coverity 8.0, all clients can accept self-signed certificates.
- CA root certificate propagation was difficult. Customers with private CAs had to manually propagate CA root certificates to every SSL client. Now, several features make this task easier.
- There were multiple types of trust stores, and several ways to update each type, which was difficult to document and manage. Now, most trust stores are in PEM format, facilitating updates.

3.1.3.5. Self-signed certificates and Trust-First-Time SSL

This section is relevant if the Coverity Connect server is configured with self-signed certificates. This is the default configuration, since the installer generates a self-signed certificate for the server.

Self-signed certificates do not permit the client to authenticate the server, so they are inherently less secure than CA-signed certificates, which do allow authentication. However, CA-signed certificates require more administrative overhead than self-signed certificates. With this in mind, Coverity applications allow the server to use self-signed certificates for the benefit of customers for whom the additional overhead does not justify the benefit of authentication of the server.

All Coverity SSL client applications now accept self-signed certificates, provided the user permits it. (Previous to Coverity 8.0, only `cov-commit-defects`, `cov-run-desktop` and `cov-manage-history` accepted self-signed certificates.) Coverity's algorithm for conditionally accepting them, called Trust First Time (TFT), is designed to minimize the number of times the user is asked to permit an SSL handshake with self-signed certificates. The algorithm, modeled after that of SSH, has these characteristics:

- A handshake with a self-signed certificate will fail unless the user permits it to succeed.
- This permission is stored by the client.
- Subsequent handshakes with the same self-signed certificate for the same server host and port automatically succeed.
- An attempt by the server or an impostor to use a different self-signed certificate is rejected by the client.

TFT has two modes of operating: one for attended applications and one for unattended applications. It is assumed that the former will always have a user present to be able to accept alerts and answer questions, while this may not be true of the latter. The attended applications in Coverity 8.0 are all the GUI applications. All the command-line applications are considered unattended.

TFT is triggered when the server sends a self-signed certificate to the client. If the server's certificate is not present in the trust store of self-signed certificates, the application's response is based on the user's intentions with respect to self-signed certificates:

- Attended applications describe the self-signed certificate for the user and ask whether it should be trusted (i.e., accepted for the handshake) and stored for future handshakes. If the user assents, e.g. by clicking the OK button, the application completes the handshake and stores the certificate.
- Unattended applications cannot ask the user what to do. Instead they rely on a command-line directive. A new option, `--on-new-cert`, specifies what to do. It has two values: "trust" and "distrust". The default is "distrust". (The default behavior previous to Coverity 8.0 was "trust".)
  - "trust" indicates that the application should behave as if the user assented to use of the self-signed certificate: the handshake is completed and the certificate is stored.
  - "distrust" indicates that the application should fail the handshake and not store the certificate.

    The old `--authenticate-ssl` option for `cov-commit-defects` is a synonym for `--on-new-cert distrust`.

If the server sends a different self-signed certificate than in the past, the TFT algorithm detects this and displays a warning for the user. This is important because it may indicate that the server is an impostor. In this situation,

- Attended applications ask the user if the stored certificate should be replaced with the new one. If the user says yes, it stores the new certificate and permits the handshake to succeed.

- Unattended applications fail.

The trust store for self-signed certificates is a directory under the user’s home directory:

- (Windows) `%APPDATA%\Coverity\certs\tft`
- (elsewhere) `$HOME/.coverity/certs/tft`

3.1.3.6. Types of Coverity Trust Stores

This section is relevant to customers whose server certificates are issued by private CAs. In this situation, the administrator must provide users with CA root certificates, and a way for client applications to use them.

In order to facilitate use of private CAs, Coverity provides several different trust stores. Generally, trust stores used by Coverity software are sequences of PEM-formatted certificates. They are in ASCII, which makes them easy to manipulate.

Installation Trust Store
: This file is installed with Coverity client applications. It is called `ca-certs.pem` and is located in the `certs` directory. It contains public CA certificates, and private CA root certificates can be added to it. If it is not present, Java applications use the CA certificate file `cacerts` installed with Java. This is not a PEM file but a Java keystore file with the password `changeit`.

Extra CA Trust Store
: This optional CA certificates file is passed to command-line applications using the `-–certs` option or to GUI applications via their *Connection* pages.

User Trust Store
: This file is stored under the user's home directory:

  - (Windows) `%APPDATA%\Coverity\certs\ca\ca-certs.pem`
  - (Unix/Linux) `$HOME/.coverity/certs/ca/ca-certs.pem`

Operating System Trust Store
: Some operating systems (Windows, Linux, Mac OS) provide mechanisms for storing CA root certificates.

  Windows: Active Directory administrators can remotely manage the user’s Root Certificate Store, or users can update the store themselves using a wizard, the certmgr.msc executable.

  Additional References:

  - Active Directory Certificate Services
  - Active Directory Certificate Services Step-by-Step Guide

  Mac OS X: Certificates are stored in the user and system keychains. The user uses the `Keychain` graphical application or the `security` command to add certificates. The Keychain API may also be used.

Additional References:

- Keychain Access: Add certificates to a keychain

Unix/Linux: certificates are stored in /etc/ssl/certs/ca_certificates.crt, a PEM-format file. The administrative procedures for updating certificates vary by OS brand, but they all require root permission. Trust stores can be updated remotely using scripting or SSH.

Additional References:

- OpenSSL Cookbook

3.1.3.7. Procedures for propagating CA certificates

These procedures are needed for propagating private CA root certificates. Guidelines are presented below depending on network environment.

On Windows, use Active Directory
By storing a certificate in AD, a network administrator can push the certificate to users' Root Certificate Store. See your AD documentation for information on how to do this.

On Unix/Linux, use the OS trust store
Using remote administration software such as Puppet, append CA certificates to `/etc/ssl/certs/ca-certificates.crt`, a PEM-formatted file.

Share a file via shared storage
If your client executables are provided via a shared volume, consider putting the CA certificates in the Installation trust store.

Otherwise, consider putting the CA certificates in shared storage and referring to them as the Extra CA trust store.

Store a certificate file in your codebase
It can be used as an Extra CA trust store or copied to the User trust store.

Propagate using email
If you email a PEM-format certificate file as an attachment with an extension of `.cer`, a Windows user can add it to their Root certificate store by double-clicking on it.

Similarly, a Macintosh user can add the certificate to their keychain.

If your Unix/Linux users have root access, they can append the file to their OS trust store file.

Otherwise, the attachment can be put in the User trust store.

3.1.3.8. Obtaining the CA's certificate

You can obtain the CA's certificate from the .pem file on the server.

To obtain the certificate, on the Coverity Connect host, use `keytool` to export the CA's certificate in PEM format (-rfc) to a file. For example, where the CA's certificate's alias is "root" in the CC keystore:

```
<CC_host>$ <CC_install_dir>/jre/bin/keytool \
-keystore <CC_install_dir>/server/base/conf/keystore.jks -storepass changeit \
-exportcert -rfc -alias root -file root-CA-cert.pem
Certificate stored in file root-CA-cert.pem
```

Chapter 3.2. Managing users, groups, and roles

Table of Contents

3.2.1. Managing users 143
3.2.2. Managing groups 152
3.2.3. Roles and role based access control 157

3.2.1. Managing users

You manage users in Configuration → Users & Groups. If your user role has permission to manage users and groups, you can add, delete, copy, import, and edit users. When adding and editing a user, you can assign the user to a group and set one or more roles for the user.

Built-in users

Coverity Connect automatically creates the following users during the installation process:

- The *admin* user is the overall administrator for the system. This user cannot be deleted. Also, this user is outside of the scope of the RBAC feature.
- The *reporter* user is a specialized process (as opposed to an actual person) in the Coverity Connect system that collects nightly trend and metrics data.

  The *reporter* is assigned the *System Report Generator* role to streamline the trend data collection process. By default, the role assignment is global, meaning that the *reporter* collects data for all components, projects, and streams on your system. To change this default behavior, see Section 3.2.1.5, “Limiting the scope of data collection”.

3.2.1.1. Setting up a local or LDAP user

When a user is created, it is automatically added to the *Users* group. This built-in group has predefined access control rules that allow the user to perform certain tasks, such as signing into Coverity Connect. After the user is created, you can assign other permissions through the *Roles* tab.

Note

Compare to Section 3.2.1.4, “Importing users”

To set up a new user:

1. Navigate to Configuration → Users & Groups.
2. Click **Add** to open the edit fields for the new user.
3. Specify the user properties.

   For guidance with this step, see Section 3.2.1.2, “Editing a local or LDAP user”.
4. Click **Create** to save your changes and exit.

Note

Values for the Coverity Connect *username* field are stored in lower case.

3.2.1.2. Editing a local or LDAP user

After creating a user, you can edit its properties.

To edit a user:

1. Navigate to Configuration → Users & Groups.
2. Select the user to edit. In *User Details*, click *Edit*.
3. Edit the user properties.

   - *Username*: Required identifier for the users.

     New user names cannot match any existing user in the system.

     Note

     If an existing user name is changed, then any authentication keys that were generated for it will need to be regenerated using `cov-manage-im`.

   - *First Name*: Optional entry for the first name of the user.
   - *Last Name*: Optional entry for the last name of the user.
   - *Email*: Optional e-mail address of between 6 and 256 characters. The e-mail address is required for Notification.
   - *Password*: Required entry of between 6 and 32 characters.
   - *Confirm Password*: Required entry that must match the password entry.
   - *Account Type*: Required user account type (*Local* or *LDAP*).

     All information about local users is stored in the Coverity Connect database. Information about LDAP users is stored both in the Coverity Connect database and an LDAP server, with the exception of passwords.

     As an administrator, you can control which types of user accounts are allowed to access Coverity Connect by disabling various authentication mechanisms through the Configuration → System → Authentication and Sign In screen.

     You can change the *Type* at any time by editing the user.

     LDAP Users

     If you change an LDAP user to a local user, all LDAP groups are removed.

After you add an LDAP user and apply your changes, the user is automatically synchronized with the LDAP server. For information about manually synchronizing users with LDAP groups, see Refreshing LDAP group membership.

You cannot change the user name, domain, first name, last name, email, or password of an LDAP user because this information is stored on the LDAP server.

You can change the *Account Type* from LDAP to Local (or vice versa).

To email the new user an account notification that contains a Coverity Connect URL, name, and sign-in information, select *Email sign-in instructions once you apply*.

Requirement

You can only send email sign-in instructions if you have configured email in Coverity Connect. The check box will be disabled if email is not configured.

- *Locale*: Required UI display language for the particular user. You can select from the following:
  - *English (United States)* - default
  - *Japanese (Japan)*

  The language setting will take effect the next time the user signs in.

  Click *OK* to close the *Edit* dialog.

4. If necessary, in the *Group Memberships* tab, assign the user to an additional user group. For guidance, see Section 3.2.1.2.1, “Assigning a user to a group”.

   By default, all users belong to the *Users* group.

5. If necessary, in the *Roles* tab, change the role settings for the user at the global, project, stream, or component level.

   For guidance with this task, see Section 3.2.1.2.2, “Managing roles for a user”. Note that it is typical to reserve role assignment at the user level for special cases in which role assignment at a group level is not feasible or where you need to override a group-level role (see Section 3.2.1.2.1, “Assigning a user to a group”). For details about roles, see Section 3.2.3.1.3, “Understanding how Coverity Connect applies roles”.

6. Click **Done** to save your changes and exit.

3.2.1.2.1. Assigning a user to a group

A user can belong to one or more groups. Groups are an efficient way of assigning roles to a user. The user inherits the roles of the groups to which it belongs, unless overridden at the user level (see Section 3.2.3, “Roles and role based access control”).

Note

If you want to assign multiple users to a group at once, you can follow the procedure in Section 3.2.2.2.1, “Assigning one or more users to a group”.

To assign a user to a group:

1. Select a user to display the *Group Memberships* tab for the user.
2. Click *Edit*, and start typing in *Select group to add*. Select one or more groups from the drop-down menu in the tab.

   For example, you might select the *Administrators* group. You might need to scroll down the list.
3. Click **OK** to save your changes and exit.

Removing a user from a group

To remove a user from a group, select the group, and click **Remove**.

3.2.1.2.2. Managing roles for a user

When you select a user from the *Users & Groups* menu, the *Roles* tab will display any roles that are currently assigned to the user at each level (global, component, project, stream, and triage store). You can use this tab to view role assignments at all levels, or add, edit, and remove roles for the selected user at the global level.

Note

You can edit roles at a lower level (by project, stream, component map, or triage store) from the respective Configuration menu.

For information about roles, including role and permission definitions, see Section 3.2.3, “Roles and role based access control”.

To manage a role for a user:

1. Select a user from the list of users.
2. Click the *Roles* tab for the user.

   a. In the *Roles* tab, ensure that you've selected the Global level from the *Show* pull-down menu.

   b. Click **Edit** in this tab.

   c. Assign one or more roles to the user.
3. Click **OK** to save your changes and exit.

Removing a role from a user:

To remove a role assignment from a user, click **Edit** in the *Roles* tab, and de-select the role that you want to remove. Then apply your changes.

3.2.1.2.3. Locking and unlocking a user account

You can lock or unlock a user account. Users that are locked out of Coverity Connect can regain access through a password recovery email, if you have enabled password recovery (for details, see Section 3.1.1.7, “Configuring sign-in settings”) and have configured email.

Note

Compare to Section 3.2.1.2.5, “Disabling a user account”.

To lock or unlock a user account:

1. Select a user from the user list.
2. In *User Details*, click *Edit*.
3. Under *Additional Actions*, select *Lock account* to lock a user account.

   Note

   If a user has been locked out automatically due to a timeout interval, you need to deselect *Lock account* to re-enable access.

3.2.1.2.4. Unlocking all temporarily locked user accounts

You can unlock all temporarily locked user accounts. A user who is temporarily locked out of an account as a result of repeated unsuccessful login attempts can simply wait 30 minutes for the account to be automatically unlocked. As an alternative, you can immediately unlock all such users. Unlocking temporarily locked user accounts does not affect accounts that you have explicitly locked.

To unlock all temporarily locked user accounts:

1. Navigate to Configuration → System → Authentication and Sign In → Sign In Log.
2. Click **Unlock All**.

3.2.1.2.5. Disabling a user account

You can disable a user account to prevent that user from gaining access to Coverity Connect.

Note

Compare to Section 3.2.1.2.3, “Locking and unlocking a user account”.

To disable a user account:

1. Select a user from the user list.
2. In *User Details*, click *Edit*.
3. Under Additional Actions, select *Disable account*.

Users that are disabled cannot log in to the system, cannot use password recovery, and cannot become owners of new issues. An administrator must unset this option to reestablish access (or create a new user account).

To disable users who have been disabled or deleted in LDAP

If a user has been deleted from your LDAP server, you need to change the user account type to Local before you can disable the user account. To change the account type, see Section 3.2.1.2, “Editing a local or LDAP user”.

You can also disable users who have been disabled in LDAP. The following steps will result in disabling all users matching the specified filter, at midnight.

1. Go to **Configuration > System > User Search Settings**.

2. In the **Disabled user filter** field specify an LDAP filter.

   The filter depends on the LDAP sever implementation. For example, it might be `UserAccountControl:1.2.840.113556.1.4.803:=2` for Active Directory server.

To change the default schedule from 12am every night, you can add the `ldap.users.disabled.sync.cron.schedule` property to the `cim.properties` file. This property accepts a cron string in the same format as cron schedules in the policy manager. For example, the following string disables users every night at 3 a.m.: `ldap.users.disabled.sync.cron.schedule= 0 0 3 * * *`

3.2.1.3. Deleting a user

To delete a user:

1. Navigate to Configuration → Users & Groups.

2. Select each user to delete.

3. Click **Delete**.

   Click **Delete** in the confirmation window.

4. Click **Done** to save your changes and exit.

Note

When you delete a user who is not associated with an issue, snapshot, or triage record, that user is removed from the system. On the other hand, a deleted user that *is* associated with an issue, snapshot, or triage record is, by default, merely disabled and removed from the Users & Groups list. If you want all users removed from the system upon deletion, then you must add the following property to the `$CIM_HOME/config/cim.properties` file:

```
retain.user_information=false
```

You must restart Coverity Connect for this change to take effect. Setting this property does not affect users who were deleted prior to the setting of the property. Such users will continue to exist in a disabled state.

3.2.1.4. Importing users

Instead of setting up users one at time, you can import multiple local or LDAP users at once.

1. Navigate to Configuration → Users & Groups.
2. Click **Import** to open the *Import Users* window.
3. To import local users, you can either:

   - Upload a comma-separated CSV file from a local directory.
   - Import users from an LDAP server.

   An entry has the following structure:

   ```
   User Name, First Name, Last Name, Email, Group1; Group2
   ```

   Notice the semi-colon (rather than a comma) between the group names.

   Note

   The format for the exported `users.csv` file (generated by clicking the **Export** button) contains two additional fields:

   - A boolean value between the `Email` value and `Group` values specifies whether or not the user is built-in.
   - The final field in the exported file (after the group names) specifies the date of the user's last login, or, if the user has never logged in, the time of the export.

   These fields are for informational purposes only, and must not be specified when importing. Also note that values in the exported file are escaped by quotation marks (""), but these should not be included when importing.

4. If you opt to import users from an LDAP server:

   a. If multiple domains appear in a drop-down list, select the correct domain. Click **Next**.

   b. Select one or more groups to import, and then click **Next**.

   c. Confirm the list of users by clicking **Next**.

      Note

      This process can take a few minutes to complete. Once complete, a table with attributes of the imported user will appear.

d. If you receive an error notification, click **Back** and fix these problems, or click **Next** to continue the import without these users.

e. Select any of the following options:

- *Lock accounts (users must use recover password to unlock)*
- *Disable accounts (users cannot sign in)*
- *Email sign in instructions to users after adding*
- *Refresh LDAP group membership nightly*

Note

For the *Lock accounts* and *Email sign in instructions* to be enabled, it is necessary for Coverity Connect e-mail to be enabled. For details, see Section 3.1.1.15, “Configuring email notification and delivery”.

f. Click **Finish** to complete the process.

g. Click **Done** to save your changes and exit.

5. Click **Done** to save your changes and exit.

3.2.1.5. Limiting the scope of data collection

You can change the scope of the data that the *reporter* [p. 143] collects by disabling its access to that data for any of the following items:

- Project
- Stream
- Component

For example, you might want to omit data collection from certain streams so that the issue data contained in those streams are not added to the overall issue trend data.

To limit the scope of data collection on a project or stream:

1. Navigate to the Configuration → Projects & Streams menu.
2. Select the appropriate project or stream from the file tree on the left.
3. Select the *reporter* user from the *Roles* tab and click **Edit**. If *reporter* isn't listed, click *Add* and type it in the *Group/User* field.
4. Assign the *No Access* permission to the user.

Note

The *No Access* permission will only be assigned to the *reporter* in the context of this specific project or stream.

To limit the scope of data collection on a component:

1. Navigate to the Configuration → Component Maps menu.
2. Select the appropriate component under *Component Maps*.
3. On the *Components* tab, under *Group/User*, select the *reporter* user and click **Edit**. If *reporter* isn't listed, click *Add* and type it in the *Group/User* field.
4. Assign the *No Access* permission to the user. This permission will only be assigned to the *reporter* in the context of this specific component.

Note

Streams created through the Coverity Desktop plug-in automatically have the *User Group - No Access* permission assigned to them by default. This will cause data collection from these streams to be omitted. To enable data collection for a stream created by the Coverity Desktop plug-in, remove the User Group from the stream. Note that doing so effectively makes the stream visible to others.

3.2.1.6. Working with authentication keys

Coverity Connect users can create authentication keys and use them to securely connect to Coverity Connect without specifying a password.

3.2.1.6.1. Creating keys

Users can create authentication keys from the Coverity Connect UI or from the command line.

To create a key from the UI:

1. After logging in to Coverity Connect, select <User> → Authentication Keys...

   The "Manage My Authentication Keys" page displays.
2. In the *New Authentication Key* field, enter a name for the key.
3. Click *Create and Download*.

To create a key from the command line:

- Create the key using the `cov-manage-im` command in `auth-key` mode. For example:

```
cov-manage-im --host <host_name> --port <port_number> \
          --user <user_name> --password <password> --mode auth-key \
          --create --output-file <keyfile>
```

3.2.1.6.2. Using keys

After creating an authentication key, `cov-manage-im`, `cov-commit-defects`, and `cov-run-desktop` will accept the `--auth-key-file` option for connecting to Coverity Connect, rather than requiring `--password`.

See the command reference for `cov-manage-im`, `cov-commit-defects`, and `cov-run-desktop` for more information.

3.2.1.6.3. Viewing and revoking keys

You can view a list of the active authentication keys that you created, and you can revoke one or more of them. To do so, log in to Coverity Connect and select <User> → Authentication Keys...

3.2.1.6.4. Use cases

There are three main scenarios in which a user should consider using an authentication key:

Desktop Analysis with the `cov-run-desktop` command
: When performing desktop analysis, `cov-run-desktop` needs to communicate with Coverity Connect (unless in disconnected mode). Using an authentication key is a secure alternative to typing a password on every analysis.

Certain requests when password-based authentication is disabled
: When RPA is enabled, password-based authentication can be optionally disabled (for details, see Section 3.1.1.12.1, “RPA key concepts”. if password-based authentication is disabled, you must use authentication keys (instead of password-based authentication) for SOAP and REST Web services requests and for `cov-commit-defects` requests.

Periodic commits of new analysis snapshots with `cov-commit-defects`
: A user or administrator might use a script to run periodic analyses and commits. Using an authentication key is more secure than having the Coverity Connect password hard-coded directly in the script.

Authentication keys can be used for all administrative actions that can be performed by means of a web service. For example, authentication keys can be used for scripts that create or update projects and streams, or create triage stores.

Note

When an authentication key is created by `cov-manage-im`, its file permissions will only allow the user who created the key file to read it. If the file permissions are changed to allow other users to read the file, the `cov-*` tools will no longer accept the key.

3.2.2. Managing groups

Use Configuration → Users & Groups to manage groups. All users must belong to at least one group.

If your role has the *Manage users and groups* permission, you can add, copy, edit, delete, and import groups. When adding and editing, you can assign or remove users from the group and set the roles that apply to the group at any level of the system (global, component, project, stream).

3.2.2.1. Built-in groups

By default, built-in groups are assigned specific roles that allow each group to perform functions:

Administrators
: By default, members of this group have the *Server Admin* role at the *Global* level, which provides them system-wide access to Coverity Connect.

Configuration Managers
: By default, members of this group have the *Project Admin* and *Project Owner* roles, which allow them to configure projects, streams, components, and attributes.

Users
: By default, members of this group have *Committer* and *Developer* roles at the *Global* level, and they have the *Developer* role at the *Component* level. Every user that is created is, by default, a member of this group.

Note that you can customize access control permissions by changing roles for these and other groups. For more information, see Section 3.2.2.2.2, “Managing roles for a group”. For additional details about these roles, see Figure 3.2.1, “Built-in roles”.

Note

For user administration and system management tasks, rather than using the default *Administrator* account, it is recommended that an alternative administrator account be created. Follow the procedures in Chapter 3.2, *Managing users, groups, and roles* to create a new user account and assign it to the *Administrators* group. That user account will have all of the permissions of the *Server Admin* role.

3.2.2.2. Setting up a group

Additional groups are useful if you need to assign specialized roles or LDAP properties to a group of users.

Note

You can also import groups from external files or an LDAP server. See Section 3.2.2.4, “Importing LDAP groups”.

To add a new group:

1. Select Configuration → Users & Groups.
2. In the *Groups* tab, click **Add** to create a new group.
3. Specify a unique *Name* for the group.

4. Specify one of the following group types:

   - *Local*
   - *LDAP*

     Select the LDAP domain, if available. The LDAP group must exist on the LDAP server before you add it.

5. Click *Create*.

6. If you selected LDAP for the *Group Type*, click **Edit** in **Group Details** to select the following options:

   Refreshing LDAP group membership

   Refresh LDAP group membership nightly
   : This option synchronizes members of the LDAP group with users on the system on a nightly basis. Users are not deleted or disabled, and no user attributes are refreshed. The nightly synchronization time occurs at 2 AM (02:00:00), and this time cannot be changed.

   Refresh LDAP group membership now
   : This option synchronizes members of the LDAP group with users on the system immediately after you apply this setting to the group membership.

   In both cases, new group members are added as users, and users who are no longer members of the group are removed from the group.

7. Use the *Members* tab to assign users to the new group.

   For guidance, see Section 3.2.2.2.1, “Assigning one or more users to a group”.

8. Use the *Roles* tab to assign one or more roles to the group.

   For guidance, see Section 3.2.2.2.2, “Managing roles for a group”

9. Select **Done**.

3.2.2.2.1. Assigning one or more users to a group

When you create or edit a group, you can assign users to it.

To assign one or more users to a group:

1. In the *Groups* tab, select a group.

2. In the *Members* tab, click *Edit*.

3. Start typing in *Select user to add*, and select a user from the drop-down list. Repeat to add more users.

4. Click **OK** to save your changes.

Removing a user from a group

To remove a the user from a group, click *Edit*, select the user, and click **Remove**.

It is not possible to remove users from the *Users* group.

3.2.2.2.2. Managing roles for a group

When you select a group from the *Users & Groups* menu, the *Roles* tab will display any roles that are currently assigned to the group at each level (global, component, project, stream, and triage store). You can use this tab to view role assignments at all levels, or add, edit, and remove roles for the selected group at the global level.

Note

You can edit roles at a lower level (by project, stream, component map, or triage store) from the respective Configuration menu.

For information about roles, including role and permission definitions, see Section 3.2.3, “Roles and role based access control”.

To manage a role for a group:

1. Select a group from the list of groups.
2. Click the *Roles* tab for the group.

   a. In the *Roles* tab, ensure that you've selected the Global level from the *Show* pull-down menu.

   b. Select *Global* in the list and click **Edit** in this tab.

   c. Assign one or more roles to the group.
3. Click **OK** to save your changes.

Removing a role from a group:

To remove a role assignment from a group, click **Edit** in the *Roles* tab, and deselect the role that you want to remove. Then apply your changes.

3.2.2.3. Editing a group

To edit a group:

1. Select Configuration → Users & Groups.
2. In the *Groups* tab, select the group to edit.
3. In the *Group Details* tab, click *Edit*.

   For guidance with this task, see the information on editable group fields in Section 3.2.2.2, “Setting up a group”.

You cannot change the group name for an LDAP group.

4. Click **OK** to save your changes.

3.2.2.4. Importing LDAP groups

You can import one or more LDAP groups at once.

Note

The Group Filter can now be applied to either top-level and nested groups, or just top-level groups. By default it applies to both.

To import only top-level groups, go to **Configuration > System > LDAP Configuration > ldap** and uncheck **Retrieve group members from nested groups**.

1. Navigate to Configuration → Users & Groups.
2. Click **Import** to open the *Import LDAP Groups* window.
3. Import one or more groups:
    a. If multiple domains appear in a drop-down list, select the correct domain, and then click **Next.**

       If not, proceed to the next step.

    b. Select one or more groups to import, and then click **Next**.
    c. Confirm the list of groups by clicking **Next**. A table that lists all of your group memberships should appear.
    d. If you receive an error notification, click **Back** and fix these problems, or click **Next** to continue the import without these groups.
    e. Select any of the following options:
        - *Lock accounts (users must use recover password to unlock)*
        - *Disable accounts (users cannot sign in)*
        - *Email sign in instructions to users after adding*
        - *Refresh LDAP group membership nightly*

        Note

        For the *Lock accounts* and *Email sign in instructions* to be enabled, it is necessary for Coverity Connect e-mail to be enabled. For details, see Section 3.1.1.15, “Configuring email notification and delivery”.
    f. Click **Finish**.

   g. Click **Done**.

4. Click **Done**.

3.2.2.5. Deleting a group

To delete a group:

1. Select Configuration → Users & Groups.

2. On the *Groups* tab, select each group to delete.

3. Click **Delete**. In the confirmation dialog, click *Delete* again.

3.2.3. Roles and role based access control

Role-based access control (RBAC) is a feature that restricts system access to authorized Coverity Connect users. Within Coverity Connect, roles represent various job functions that can be assigned to users or groups.

Roles consist of permissions that grant or restrict actions to certain operations or to certain areas of the user interface. Users and groups are not assigned permissions directly; rather, they acquire them through their role (or roles).

Managing individual user access rights becomes a matter of simply assigning the appropriate roles to the user or group. This also simplifies common operations, such as accounting for user turnover or changing a user's department or job function.

This chapter provides the following information:

- Key concepts, including role and permission definitions, role assignments, and role creation.

- Use cases that give examples of the use of roles in Coverity Connect.

3.2.3.1. Key concepts

The following sections provide descriptions of RBAC-related topics. They also provide definitions of permission rules and list the available roles in Coverity Connect and the role permissions that they possess.

3.2.3.1.1. Coverity Connect Permissions

Permissions are rules that can be added to a role to define access rights for a user. Generally, similar types of rules are assigned to a role based on a person's responsibilities within the organization.

Global permissions

System permissions are access rules that are intended primarily for administrative tasks, such as Coverity Connect server configuration, user and group creation and management, and so forth. The global permissions are described in the following table:

Permission	**Description**
Log in to Coverity Connect	Allows the user to sign in to Coverity Connect through the web interface, assuming that the user has proper authentication credentials.
Access web services	Allows the user to access the Coverity Connect Web services API.
Manage server parameters	Allows the user to access the Configuration → System menu and to edit the Coverity Connect server system settings.
Manage users and groups	Allows the user to access the Configuration → Users & Groups menu, and to create and manage user and group settings.
Manage role definitions	Allows the user to access the Configuration → Roles menu and to create roles and assign permissions.
Manage attribute definitions	Allows the user to access the Configuration → Attributes menu and to create, configure, and delete Coverity Connect triage attributes and their values.
Manage component maps	Allows the user to access the Configuration → Component Maps menu. The user can create component maps, add components, add RBAC access rules, and so forth.
View global dashboard	Allows the user to access the dashboard for the current project for Quality, Security, and Test Advisor (if a valid Test Advisor license exists) results. The dashboards shows graphs and charts that provide an overview of status of all of the project.
Create projects	Allows the user to access the Configuration → Projects & Streams menu to create new projects and streams in the system.
Create triage stores	Allows the user to create new triage stores in the system by accessing the Configuration → Triage Stores menu.
View Policy Manager	Allows the user to access Coverity Policy Manager to view and create charts that are used for monitoring and reporting on the status of the code base. Note that Coverity Policy Manager

Permission	Description
	might not be enabled by your Coverity Connect license.
Manage Hierarchies	Allows the user to configure Coverity Policy Manager Hierarchies. Note that Coverity Policy Manager might not be enabled by your Coverity Connect license.

Project permissions

Project permissions are access control rules that are applied at the project level. Users with project permissions can access items and features in the *Projects* screen. For example, your organization more than likely has project leads or project owners. Such users are typically assigned roles that contain these permissions. After the roles are assigned, users with project permissions can then assign roles or regulate access in the project to the developers that are assigned to it. The project permissions are described in the following table:

Permission	Description
Manage projects	Allows the user to edit and manage the project, including updating the name and description, assigning roles to users in the project, linking streams, and setting trend data calculation formulas.
Create streams	Allows the user to create new streams within the project.
View project history and dashboard	Allows the user to view the project trends and reports.

Stream permissions

Stream permissions are access control rules that are applied to users at the stream level. Stream permissions are intended for users (typically, developers) who are examining and triaging issues in the code. The stream permissions are described in the following table:

Permission	Description
Manage streams	Allows the user edit and manage the stream, including updating the name and description, assigning roles for users in the stream, and so forth.
View issues	Allows to the user to view, but not triage, issues that occur in the stream to which the user is assigned.
View source	Allows the user to view issue occurrences in the Source browser.
Commit to stream	Allows the user to commit analysis results to Coverity Connect using the `cov-commit-defects` command.

Permission	Description
Preview commits	Allows the user to preview commits to streams using the `--preview-report` option with the `cov-commit-defects` command.
Triage issues	Allows the user to change and update triage states for issues that exist in the stream. Users who do not possess this permission cannot access the triage form in the *Source* tab.
Classify issues	Allows the user to change and update classification states for issues that exist in a stream that is associated with a triage store. Users who do not possess this permission cannot access the triage form in the **Source** tab.

Triage Store permissions

Triage Store permissions are access control rules that are applied to users at the triage store level. Triage store permissions are intended for users (typically, developers) who are examining and triaging issues in the code. The triage store permissions are described in the following table:

Permission	Description
Manage triage stores	Allows the user edit and manage the triage store, including updating the name and description, and branching triage stores, associated streams with a triage store, and assigning roles for groups and users in the triage store.
Classify issues	Allows the user to change and update classification states for issues that exist in a stream that is associated with a triage store. Users who do not possess this permission cannot access the triage form in the **Source** tab.
Triage issues	Allows the user to change and update triage states for issues that exist in a stream that is associated with a triage store. Users who do not possess this permission cannot access the triage form in the *Source* tab.
View issues	Allows to the user to view, but not triage, issues that occur in the stream to which the user is assigned.

Component permissions

Component permissions are access control rules that are applied to users at the component level. Component permissions are intended for users (typically, developers) who are examining and triaging issues in the code. The component permissions are described in the following table:

Permission	Description
View source	Allows the user to view issue occurrences in the Source browser.
View issues	Allows the user to view, but not triage, issues that occur in the stream to which the user is assigned.
Triage issues	Allows the user to change and update triage states for issues that exist in a stream that is associated with a triage store. Users who do not possess this permission cannot access the triage form in the *Source* tab.

3.2.3.1.2. Coverity Connect roles

Roles are entities that contain any number of permissions to grant or restrict access to a specific "type" of Coverity Connect user. For example, an administrator will most likely only need to access certain parts of Coverity Connect for configuring the system, while a developer does not need to access system configuration screens, but does need to view and triage issues.

Note

The *System Report Generator* is a specialized role that is specifically assigned to a unique type of user in Coverity Connect, the *reporter* user. The *reporter* is a process (rather than an a person) that is automatically created by Coverity Connect. For details about the role of this user, see Built-in users.

Roles can be assigned to both users and groups. For more information, see Section 3.2.3.1.3, “Understanding how Coverity Connect applies roles”.

Coverity Connect has the following types of roles:

- Built-in roles. For details, see Section 3.2.3.1.2.1, “Managing built-in roles”.
- Custom roles. For details, see Section 3.2.3.1.2.3, “Creating new roles”.

3.2.3.1.2.1. Managing built-in roles

Built-in roles are pre-defined roles that you cannot edit or delete. You can, however, copy a built-in role and edit the name and permissions of the copy.

The following table lists the permissions that are assigned for each built-in role:

Figure 3.2.1. Built-in roles

		System Admin	System Report Gen.	Visitor	Project Owner	Stream Owner	Triage Store Owner	No Access	Hierarchy Admin	Policy Manager User
Global Permissions	Log in to Coverity Connect	●		●	●	●	●		●	
	Access web services	●		●	●	●	●		●	●
	Manage server parameters	●								
	Manage users and groups	●								
	Manage role definitions	●								
	Manage attributes	●								
	Manage component maps	●								
	View global dashboard	●								
	Create projects	●								
	Create triage stores	●								
	View policy manager	●							●	●
	Manage hierarchies	●							●	
Project	Manage projects	●			●					
	Create streams	●			●					
	View project history	●			●					
Stream	Manage streams	●	●		●	●				
	**View issues	●	●		●	●	●			
	Classify issues	●			●	●	●			
	*View source	●			●	●				
	Commit to stream	●			●	●				
	Preview commits				●	●				
Triage Stores	Classify issues	●			●	●	●			
	*Triage issues	●			●	●	●			
	Manage triage stores	●					●			

Permissions marked with an asterisk (*****) also belong to the component level. For more information, see Section 3.2.3.1.3, “Understanding how Coverity Connect applies roles”.

Permissions marked with two asterisks (******) also belong to the component level and triage store levels. For more information, see Section 3.2.3.1.3, “Understanding how Coverity Connect applies roles”.

Note

As of release 2018.06 the new permission category, Classify Issues, is available by default to all users. See Stream permissions.

To copy and edit a built-in role:

1. Select the name of the role that you want to copy.
2. Click **Duplicate**.

   A dialog with the copied role name followed by a number (for example, *Project Owner 73*) appears.
3. Change the name and description of the role, as needed.
4. Select or remove any permissions that you want included or excluded in the role.
5. Click **Create** to finalize your changes and exit the dialog.

3.2.3.1.2.2. Managing custom roles

Custom roles are pre-configured roles that you can edit by adding or removing permissions to match your requirements. The following table lists the permissions that are assigned for default roles:

Figure 3.2.2. Custom roles

		Server Admin	Project Admin	Stream Admin	Developer	Observer	Committer	Web Srvc Reporter
Global Permissions	Log in to Coverity Connect	●	●	●	●	●		
	Access web services	●	●	●	●	●		●
	Manage server parameters	●						
	Manage users and groups	●						
	Manage role definitions	●						
	Manage attributes		●					
	Manage component maps		●					
	View global dashboard							
	Create projects		●					
	Create triage stores		●					
	View policy manager							
	Manage hierarchies							
Project	Manage projects				●			
	Create streams			●	●			
	View project history				●	●		
Stream	Manage streams							
	**View issues				●	●		●
	Classify issues				●			
	*View source				●	●		
	Commit to stream						●	
	Preview commits				●			
Triage Stores	Classify issues				●			
	*Triage issues				●			
	Manage triage stores							

Permissions marked with an asterisk (*****) also belong to the component level. For more information, see Section 3.2.3.1.3, “Understanding how Coverity Connect applies roles”.

Permissions marked with two asterisks (******) also belong to the component level and triage store levels. For more information, see Section 3.2.3.1.3, “Understanding how Coverity Connect applies roles”.

To edit a custom role:

1. Select the name of the role.
2. In *Role Details*, click **Edit**.
3. Change the name or description of the role, if desired. Select or remove any permissions that you want included or excluded from the role.
4. Click **OK** to finalize your changes and exit the dialog.

Note

To delete a default role, see Deleting roles.

3.2.3.1.2.3. Creating new roles

Coverity Connect allows you to create your own, custom sets of access permissions for specific types of users or groups in your organization. In general, you need to create a new role if one or more of the built-in or default roles does not apply the permissions you need for a given user or group.

To create a new role:

1. Click **Add**.

   A dialog appears with the Name set to *New Role* followed by a number (for example, *New Role 47*).
2. Change the name and description of the role.
3. Select any of the permissions that you want included in the role.
4. Click **Create** to finalize your changes and exit the screen.

   You can now apply the role to users and groups.

Deleting roles

To delete one or more roles, select the name of each role to delete, and click **Delete**. In the confirmation dialog, click **Delete** again.

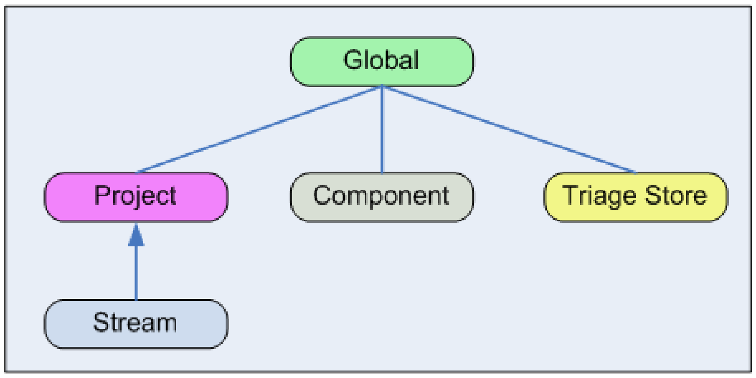
3.2.3.1.3. Understanding how Coverity Connect applies roles

Coverity Connect allows you, as an administrator, to assign roles to specific users or groups at a number of different levels. Role assignments at a more specific level override role assignments at a more general level. This allows role assignments at a specific level to grant a user (or group) additional permissions OR to remove permissions within that level. The Coverity Connect RBAC implementation provides the following levels:

- Global-level roles grant permissions to users throughout Coverity Connect, except when they are overridden by role assignments on a triage store, component, project or stream.
- Triage store-level roles grant permissions to users within the confines of a given triage store.
- Component-level roles grant permissions to users within the confines of a given component map.
- Project-level roles grant permissions to users within the confines of a given project, except when they are overridden by role assignments on a stream.
- Stream-level roles grant permissions to users within the confines of a given stream.

Some of the levels combine to form hierarchies by which Coverity Connect determines access roles, as illustrated in the following diagram:

Figure 3.2.3. RBAC roles on a Coverity Connect



Coverity Connect enforces RBAC permissions by determining an "effective" role for the user or group that is trying to perform some action. This effective role, in turn, allows Coverity Connect to decide if the action is permitted or not.

Effective role

To determine the effective role, Coverity Connect examines the most specific level for the action that is being attempted. For example, if a user is attempting to triage an issue on a specific stream, Coverity Connect determines access rights by examining the role assignments on that stream. If Coverity Connect identifies one or more role at that level, it stops the evaluation at that level and does not continue to evaluate role definitions at "higher" levels. This is because the role permissions at the more specific level hide (or, override) any role permissions that exist at more general levels.

Coverity Connect examines roles defined at a higher level only when the more specific level does not have any roles assigned to it. Therefore, a role assigned to a user or group at the global level can be overridden by assigning a role at a more specific level.

When Coverity Connect finds a role at a given level, it gathers all role definitions at that level and combines them to form the effective role. The effective role contains all of the permissions that are assigned to the user at that level, so the user can use any permission defined in any role at that level.

Furthermore, a user's role can vary depending on context. For example, a user could be assigned the Developer role on one stream, and the Observer role on another stream.

As an example of how Coverity Connect determines role access, consider the following:

- `user1` has a global Developer role.
- `user1` also has the No Access role assigned at the project level for `projectA`.

In this configuration, `user1` has permission to view and triage issue anywhere on the system, except for in `projectA`, which `user1` cannot see at all.

As an additional example:

- `user2` has a global Visitor role.
- `user2` has the Developer role defined at the project level for `projectB`.
- `user2` has a role with the *view issues* permission to view issues, and the *triage issues* permission at the component and triage store level to triage issues.

In this configuration, `user2` can log into the system, but can only view and triage issues on `projectB`.

3.2.3.1.3.1. Permissions for triaging issues

Most actions for which a user is allowed or denied permission take affect on a single object. Because of this, these permissions take place in a single hierarchy. A few common operations, however, require multiple objects and are therefore controlled by permission settings of multiple hierarchies at one time.

One very common and important example of this is applying issue triage permissions. This involves role settings at the triage store, component, and stream levels. The user must be authorized at all three levels in order for the action to succeed. All three effective roles must grant at least the *Triage issues* permission in order for the triage action to be allowed. For an example, see Section 3.2.3.2.1, “Scenario: Granting triage permissions on a synchronized Coverity Connect enterprise cluster ”

The *View issues* permission is available for definition at the triage store level, so a user must have this permission at the triage store, component, and stream levels in order to view issues.

3.2.3.1.3.2. Triaging workflows and classifying issues

By default all users have the right to triage as well as to classify an issue. If you want to prevent certain users from being able to dismiss an issue, you can do so by limiting their ability to classify an issue. You can apply the restriction to groups of users as defined by their assigned role. In this way triaging may be done in two steps: when a new issue is detected,

- A junior developer can triage the defect; then depending on the action needed
- A senior developer can determine the appropriate action; for example, by setting its classification to `False Positive` or `Intentional`

You can control which users can do further classification by enabling the **Classify issues** checkbox when defining the user role.

If a user does not have the **Classify issues** permission, when triaging an issue using the **Triage** pane, They may still specify the severity of the issue and the action taken. If the user tries to classify an issue in some other way, for example, through Web Services, an error is returned.

3.2.3.1.3.3. Assigning roles to users and groups

All levels of roles can be assigned to users and groups in the Configuration → Users & Groups menu. When you assign a role to a group, all members of the group possess the permission rules for each role. Assigning roles directly to users and groups is typically performed by an administrator who must be assigned a role that contains the *Manage users and groups* permission.

Roles assigned directly to the user take precedence over those assigned to the user through group membership. If a user belongs to a group that is assigned a role with permissions that are different than permissions that are assigned to the user, Coverity Connect applies the permissions defined for the user and ignores the group permissions.

For information about adding roles to users, see Section 3.2.1.2.2, “Managing roles for a user”. For information about adding roles to groups, see Section 3.2.2.2.2, “Managing roles for a group”.

3.2.3.1.3.4. Assigning roles to levels

Coverity Connect allows you to assign roles directly to users and groups at each of the levels:

Adding roles at the triage store level
: Specific users and groups can be assigned roles through Configuration → Triage Stores. In order to add users and group role assignments at this level, the granter must have the *Manage triage stores* permission. For more information, see Section 3.3.4, “Managing triage stores”.

Adding roles at the component level
: Specific users and groups can be assigned roles through Configuration → Component Maps. In order to add users and group role assignments at this level, the granter must have the *Manage component maps* permission. For more information, see Section 3.3.3, “Using components”.

Adding roles at the project level
: Specific users and groups can be assigned roles through Configuration → Projects & Streams. Users that are assigned roles through the project are granted permissions that apply only to the project. You must have a role with the *Manage projects* permission to add users and assign project-level roles. For more information, see Section 3.3.1.2.6, “Assigning roles per project or stream”.

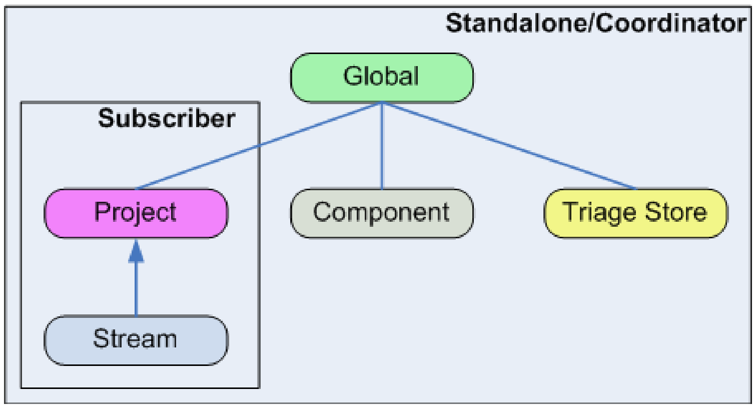
Adding roles at the stream level
: Specific users and groups can be assigned roles at the stream level through Configuration → Projects & Streams. You must have a role with the *Manage streams* permission to authorize users and assign stream-level roles. For more information, see Section 3.3.1.2.6, “Assigning roles per project or stream”.

3.2.3.1.4. Assigning RBAC roles on a synchronized Coverity Connect cluster

If you are using a Coordinator to synchronize multiple Coverity Connect instances, the global, triage store, and component level roles must be set on the Coordinator. Any of these roles that are set on the

Coordinator are shared with each Subscriber within the enterprise. Roles for the project and stream levels, however, can be set on and are local to each Subscriber instance. If you are using Coverity Connect as a standalone instance, roles can be set for each level on that Coverity Connect instance. Please note that **in order for a developer to be able to triage issues, the developer must have the appropriate roles set at the triage store, component, and stream levels**. For an example, see Section 3.2.3.2.1, “Scenario: Granting triage permissions on a synchronized Coverity Connect enterprise cluster ”.

Figure 3.2.4. RBAC roles on a Coverity Connect cluster



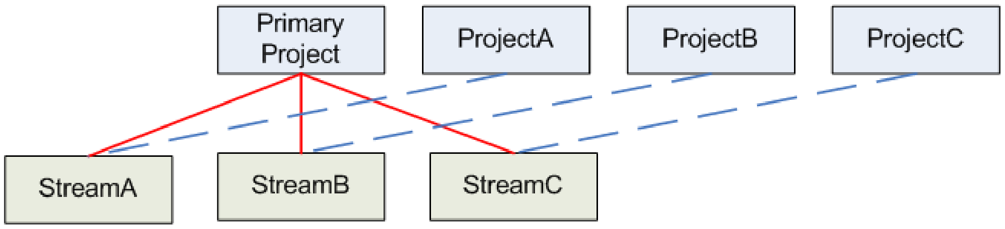
For more information about Coverity Connect synchronization, see Section 3.5.1, “Synchronizing multiple Coverity Connect instances”.

3.2.3.1.5. Primary projects and stream links

When you associate a stream with a project, that project becomes the primary project for the stream. The project also becomes the primary project for any stream links that were created from those streams. A stream link is a reference to a stream that can occur in other projects.

Streams and stream links that are associated with the primary project inherit the roles from the primary project. In this way, primary projects provide a way for project owners to centrally define and manage access role permissions for streams and stream links.

After you create and associate streams with the primary project, the streams can be associated with other projects by stream links (for details, see Section 3.3.1.2.3, “Associating a stream with multiple projects”). Stream links behave the same as the stream to which you create the link. The following figure shows the relationship of multiple project/stream associations. The red lines represent the association of streams to a primary projects, while the dotted blue lines represent a stream link association:



See the Primary project use case for an example of the process.

3.2.3.2. Use cases

The following scenarios illustrate the use of RBAC. These scenarios provide a high-level view of how roles and permissions apply to different types of Coverity Connect groups and users. These scenarios cannot cover every use case, because of the large number of combinations and ways to establish access control, but the scenarios covered in this section describe common use cases. Furthermore, these use cases assume that you (or the user types described in the scenarios) are familiar with Coverity Connect operations related to that user. These scenarios do not go into detail, for example, about triaging issues, setting up components, creating projects and streams, and so forth.

3.2.3.2.1. Scenario: Granting triage permissions on a synchronized Coverity Connect enterprise cluster

Goal: To grant permissions for groups of users to be able to triage issues on a specific stream on a subscriber instance of Coverity Connect.

In order to accomplish this, the group must have appropriate triage permissions at the following levels:

- Triage store level (set on the coordinator)
- Component level (set on the coordinator)
- Stream level (set on a subscriber)

Note

If this scenario were to be deployed on a standalone instance of Coverity Connect, the procedures are basically the same. Instead of performing these steps on a coordinator or subscriber, respectively, they would be performed on the single Coverity Connect instance.

Scenario assumptions: The organization has two product lines in their code base, represented as `ProductA` and `ProductB`.

Coverity Connect is deployed as an enterprise cluster as follows:

- `coordinator` is the name of the installed coordinator for the system.
- `subscriber1` is a configured subscriber of `coordinator` and contains the analysis streams for ProductA.
- `subscriber2` is a configured subscriber of `coordinator` and contains the analysis streams for ProductB.

On `coordinator`, the following users and entities exist:

- Users and groups:

  - `SysAdmin` possesses one or more roles that contain the *Manage users and groups*, *Manage component maps*, *Manage triage stores*, and *Manage streams* permissions (for example, the System Administrator role).
  - The `users` group contains all users on the system.
  - Users who are a group of engineers that belong to `DevGroupA`. This group represents the engineers that develop `ProductA`.
  - Users exist on the system, some of which are a group of engineers that belong to `DevGroupB`. This group represents engineers that develop `ProductB`.
  - Some users that belong to `DevGroupA` and `DevGroupB` belong to a group called`SecurityGroup`.
- Triage stores:
  - `ProductAStore` is the triage store that contains all of the development streams that represent ProductA; `ProdAVer1.0`, `ProdAVer2.0`, `ProdAVer3.0` (see subscriber 1, below).
  - `ProductBStore` is the triage store that contains all of the development streams that represent ProductB; ; `ProdBVer1.0`, `ProdBVer2.0`, `ProdBVer3.0` (see subscriber 2, below).
- Components:
  - `Default` contains component maps that are associated with streams pertaining to ProductA and ProductB.
  - `3rdParty` contains a component map of the same name that is associated with third-party libraries that are to be viewed by select users.

On `subscriber1`, the following entities exist:

- `ProjectA` exists and contains the following streams, which represent release versions of ProductA.
  - `ProdAVer1.0` - This stream has been marked as EOL (End of Life). No development or bug fixes occur on the code represented by this stream.
  - `ProdAVer2.0` - This stream represents code that is currently supported, but no new features are under development. Bug fixes are expected.
  - `ProdAVer3.0` - This stream represents code that is under active development.

On `subscriber2`, the following entities exist:

- `ProjectB` exists and contains the following streams, which represent release versions of ProductB.
  - `ProdBVer1.0` - This stream has been marked as EOL (End of Life). No development or bug fixes occur on the code represented by this stream.
  - `ProdBVer2.0` - This stream represents code that is currently supported, but no new features are under development. Bug fixes are expected.

- `ProdBVer3.0` - This stream represents code that is under active development.

Scenario procedures:

1. On `coordinator`, `SysAdmin` assigns the Developer role at the global level.

   `SysAdmin` accesses Configuration → Users & Groups, selects `DevGroupA`, and in the *Roles* tab assigns the Developer role at the global level. `SysAdmin` then repeats the role assignment for `DevGroupB` and `SecurityGroup`.

   - At this point, all groups have the Developer role assigned to them at the global level. Because there are currently no roles assigned at any of the "lower" roles levels, the members of the group will have the Developer role permissions defined for every level on the system to which the group will be assigned.

   - For triage purposes, the permissions that are of most interest are *View issues*, *View source*, and *Triage issues.*

2. `SysAdmin` assigns groups and access roles at the Triage Store level.

   a. `SysAdmin` accesses Configuration → Triage Stores.

   b. `SysAdmin` selects `ProductAStore`.

   c. On the *Roles* tab, `SysAdmin` clicks **Add**, starts typing in the *Group / User* box and selects `DevGroupB` to associate that group with `ProductAStore`.

   d. `SysAdmin` selects the Observer role to `DevGroupB`, and clicks **OK**.

   e. `SysAdmin` repeats the process but for `ProductBStore` and `DevGroupA`, respectively.

   - Access is now starting to be restricted at lower levels. `DevGroupA` has triage access only to the streams that are associated with `ProductAStore`, while `DevGroupB` has triage access only to the streams that are associated with `ProductBStore`. Each group has Observer permissions on the triage store of the other groups.

3. On `coordinator`, `SysAdmin` assigns groups and roles at the component level.

   a. `SysAdmin` accesses Configuration → Component Maps.

   b. `SysAdmin` selects the `Other` component (under the *Default* component map, *Components* tab), and in the *Group/Users* selects `DevGroupA` and `DevGroupB` to associate those groups with the component map. The groups inherit the Developer role on `Other` by virtue of the global Developer role assignment.

   c. `SysAdmin` selects the `3rdParty` component map and under *Group/Users*, adds `SecurityGroup` to associate that group with `3rdParty`.

   d. `SysAdmin` clicks **Edit** and applies the Observer role to `SecurityGroup`. This role does not contain triage permissions, but allows the members of the group to view issues.

e. `SysAdmin` adds `DevGroupA` and `DevGroupB` to the `3rdParty` component map and assigns them the No Access role.

- Users in `DevGroupA` and `DevGroupB` have triage permissions at the component-level role because the Developer role is inherited by virtue of that role's global assignment.
- Users in `SecurityGroup` are the only users on the system that can view issues in the streams that are associated with that component. Furthermore, because the Observer role is assigned to the group at the component level, the permissions in that role take precedence over the permissions that are assigned to the Developer role at the global level. So, the members of the group can view issues and the source, but not triage issues in the streams that are associated with the component map.

4. On `subscriber1`, `SysAdmin` assigns groups and roles at the stream level.

   a. `SysAdmin` accesses Configuration → Projects & Streams, and selects `ProjectA`.

   b. In both `ProdAVer2.0` and `ProdAVer3.0`, `SysAdmin` associates `DevGroupA` to them in *Group/Users*.

   c. In both `ProdAVer2.0` and `ProdAVer3.0`, `SysAdmin` associates the `users` group to them in *Group/Users* and then assigns the No Access role to the `users` group. This ensures that no users on the system can access these two streams, unless explicitly given access permissions.

   d. In `ProdAVer1.0`, `SysAdmin` associates `DevGroupA` to it in *Group/Users*.

   e. In the *Roles* tab for `ProdAVer1.0`, `SysAdmin` assigns the No Access role to the group.

   - `DevGroupA` now has the Developer role on the `ProdAVer2.0` and `ProdAVer3.0` streams because the role is inherited from the levels above it. The members of the group can now view and triage the issues that exist in each stream.
   - In the `ProdAVer1.0` stream, `DevGroupA` has the No Access role assigned. Because Coverity Connect evaluates the role defined at the most granular level (in this case, the stream level) and because the role definition does NOT contain the *View issues*, *View source*, and *Triage issue* permissions, the members of the group cannot view or triage issues in the stream.

5. On `subscriber2`, `SysAdmin` assigns groups and roles at the stream level.

   `SysAdmin` performs the same procedures described in the previous step for `DevGroupB` and `users` as they apply to the `ProBAVer1.0`, `ProdBVer2.0`, and `ProdBVer3.0`. The results are similar:

   - The members of `DevGroupB` can view and triage issues in the `ProdAVer2.0` and `ProdAVer3.0` streams.
   - The members of `DevGroupB` can not view or triage issues in the `ProdBVer1.0` stream.

3.2.3.2.2. Scenario: Delegating project management

Goal: To delegate management of a project to that project’s owner

Scenario assumptions:

- `ProjectManager` exists on the system with a role that includes the *Manage projects* permission.
- `user1` exists on the system.
- `Project A` exists on the system.

1. `ProjectManager` goes to Configuration → Projects & Streams.
2. `ProjectManager` edits `ProjectA`.
3. `ProjectManager` adds `user1` to the project's *Roles*.
4. `ProjectManager` assigns the *Project Owner* role to `user1` so that he/she is allowed to administer the project.

   Note

   Roles can also be assigned to a user group that represents users that are the project's owners.
5. `user1` logs into Coverity Connect and can create streams in `ProjectA` and assign roles to other users and groups in their respective projects.

Stream ownership can be delegated similarly to this procedure by assigning the Stream Owner role.

3.2.3.2.3. Scenario: Limiting access for project and stream administration

Goal: To allow project owners to create and configure their own projects/streams, but to not be able to access other owner's projects/streams.

Scenario assumptions:

- `Admin` exists on the system with a role that includes the *Manage users and groups* permission.
- `user1` and `user2` exist in the system and are designated to be project owners.

1. `Admin` assigns `user1` and `user2` the Project Admin role at the global level.

   This role has a *Create projects* global permission, but no project or stream-level permissions.
2. `user1` creates a `ProjectA` and is given the Project Owner role on that project by the system.
3. `user2` creates a `ProjectB` and is given the Project Owner role on that project by the system.
4. `user1` creates several streams, adds users to `ProjectA`, and assigns them the Stream Owner role on those streams.
5. `user2` also creates several streams, adds users to `ProjectB`, and assigns them the Stream Owner role on those streams.

6. `user1` cannot add any streams that exist in `ProjectB`, because `user1` does not have the *Manage stream* for those streams.

   The same is true for `user2` regarding the streams in `ProjectA`.

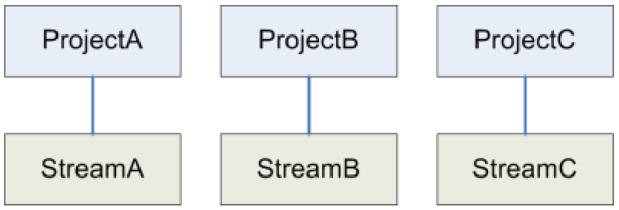
7. The users in each project cannot add streams to their respective projects, because they do not have the *Manage project* or `Create streams` permissions.

3.2.3.2.4. Scenario: Managing roles through primary projects

Goal: To manage multiple projects that have the same access control restrictions to avoid having to replicate access control settings on each project.

Scenario assumptions:

- `SysAdmin` is a user that exists in the system. This user has the *Create projects* and *Manage users and groups* permissions.
- `ProductManager` creates the following projects and associated streams:



- Multiple users exist on the system and will have access to one (or more) of the streams. (Users could alternatively be members of separate groups).

1. `ProductManager` creates a separate project called `Primary Project` that will manage all permissions.
2. `ProductManager` assigns all of the users a global role that includes the *View project* permission, but no stream permissions. (For example, Visitor).
3. `ProductManager` associates all of the streams with `Primary Project` and lists it as their primary parent. The streams now inherit access control settings from `Primary Project` and are associated with the other projects as stream links.
4. `ProductManager` assigns the appropriate roles for the users (or groups) on the `Primary Project`.
5. Because the role assignments in `Primary Project` cascade down to the streams, `ProductManager` does not need to manage stream permissions on `ProjectA`, `ProjectB`, or `ProjectC`.

3.2.3.2.5. Scenario: Granting access to create streams

Goal: To allow other users to create streams in a project.

Scenario assumptions:

- `ProjectManager` exists on the system and has a role with the *Manage project* and *Create project* permissions.
- `user1` and `user2` exist on the system, both with a global Visitor role.

1. `ProjectManager` creates `Project1`.
2. `ProjectManager` goes to Configuration → Projects & Streams and edits `Project1`.
3. In the *Roles* tab, `ProjectManager` adds `user1` and `user2` and assigns them a role that includes the *create stream* permission (for example, Stream Admin).
4. `user1` and `user2` each creates a stream, and the system grants them the Stream Owner role for each stream that they have created.

3.2.3.2.6. Scenario: Restricting project access for Coverity Desktop users

Goal: To grant Coverity Desktop users access only to the Coverity Connect projects to which they are assigned.

Scenario assumptions:

- `ProjectManager` exists on the system and has a role with the *Manage project* and *Create project* permissions.
- Multiple Coverity Desktop users exist on the system.

1. In the *Roles* tab, `ProjectManager` creates a new role, `Desktop Developer`, selects the same permissions that are assigned to the `Developer` role, and adds the *Create Triage Stores* permission.
2. `ProjectManager` creates a new Group, `Group1`, and adds the Coverity Desktop developers to the list of members.
3. In the *Projects & Streams* menu, `ProjectManager` creates a new Project, `Project1`, and adds `Group1` with the `Committer`, `Desktop Developer`, and `Triage Store Owner` roles assigned.
4. In the *Triage Stores* menu, `ProjectManager` creates a Triage Store, `TriageStore1`, and adds `Group1` with the `Desktop Developer`, and `Triage Store Owner` roles assigned.
5. In the *Triage Stores* menu, `ProjectManager` adds `Group1`, to the `Default Triage Store` and `Empty Triage Store`, with the `Desktop Developer`, and `Triage Store Owner` roles assigned.
6. In the *Component Maps* menu, `ProjectManager` creates a Component Map, `CompMap1`, and adds `Group1` with the `Committer`, `Desktop Developer`, and `Triage Store Owner` roles assigned.
7. In the *Component Maps* menu, `ProjectManager` adds `Group1`, to the `Default` component map with the `Desktop Developer` and `Triage Store Owner` roles assigned.

3.2.3.3. Guidelines for restricting access

Often it is necessary to restrict visibility of source and defects to specific users or groups of users. RBAC can be configured to allow access to specific projects, components, or triage stores. Usually, the simplest approach is to choose one way to control access, and remove restrictions on other ways:

- If you choose to control access by project, then remove restrictions on access by components and triage stores.
- If you control access by components, then remove restrictions by projects and triage stores.
- If you control access by triage stores, then remove restrictions by projects and components.

Listed below are steps to control access by project. These steps are similar to those for controlling access by components or triage stores.

1. Assign the *Visitor* role to the *Users* group at the Global level. This allows all users to login, but does not allow access to any projects, components, or triage stores.
2. Assign the *Developer* role to the *Users* group for every component and triage store. This removes restrictions on access by components and triage stores.
3. For each project, assign the roles to each specific user and group that needs access to the project. Depending on the access the user or group needs, you may assign the *Developer* role and/or other roles. This grants only the specific user or group access to the project.

Chapter 3.3. Managing data on software issues

Table of Contents

3.3.1. Working with projects and streams 178
3.3.2. Configuring triage attributes 192
3.3.3. Using components 196
3.3.4. Managing triage stores 213
3.3.5. Configuring legacy issues 223

3.3.1. Working with projects and streams

You create a stream to support issue data on a portion of your code base. Each stream is organized into a project, which can support multiple streams. You can create one or more projects in Coverity Connect.

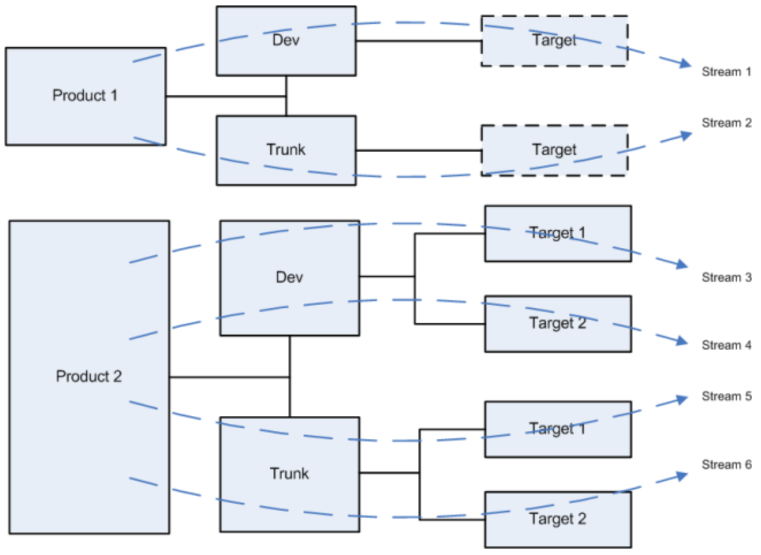
You must create a stream before you can commit issue data to Coverity Connect. To add issue data to a stream, you first need to run an analysis on some part of your code base, then commit the resulting issue data to the stream, for example, by using Coverity Analysis.

3.3.1.1. Planning your project and stream configuration

To effectively view and manage the data on issues in your code base, you need to think carefully about how to set up streams for your code base and how to organize those streams into Coverity Connect projects.

The following figure provides examples of possible Coverity Connect stream configurations for two products. `Product 1` contains a development branch and an integration (trunk) branch. Each branch has one assumed target (represented by dotted lines). `Product 2` also contains a development branch and a trunk branch, but in this case, both branches are built for two target platforms.

Figure 3.3.1. Coverity Connect streams



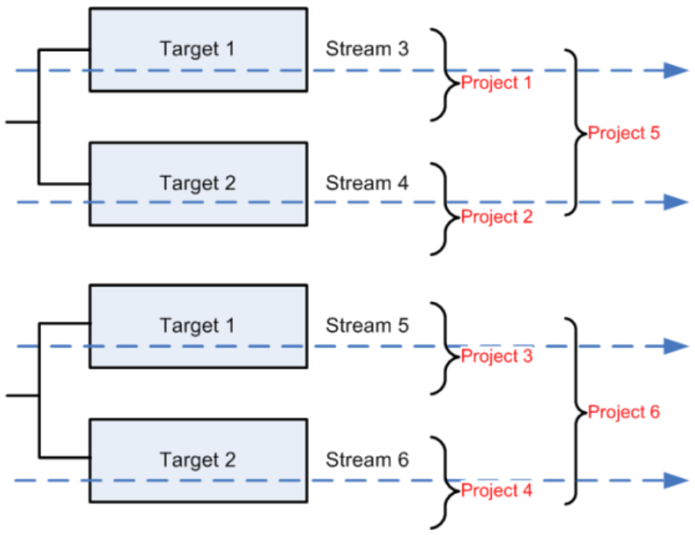
In this example, there are six possible streams. Certain streams, or combinations of streams, are useful to certain roles within your enterprise. For example:

- Developers might be interested in branches for a specific product. Developers for `Product 1` will focus on `Stream 1` or `Stream 2`. Developers for `Product 2` will focus on `Stream 3` and `Stream 4`, or `Stream 5` and `Stream 6`.
- QA engineers might be interested in a particular platform target, so they will focus on each individual stream, `Stream 1` through `Stream 6`.
- A Program manager might want to monitor the integrity of all of the products on a particular branch, so a person in this position will focus on `Stream 1`, `Stream 3`, and `Stream 4` of the `Dev` branches, and `Stream 2`, `Stream 5`, and `Stream 6` of the `Trunk` branches.

Coverity Connect allows you to organize your streams into projects to give you easy access to the most important information for your job. The following figure focuses on `Product 2` in Figure 3.3.1, “Coverity Connect streams” and shows how the streams can be associated with projects appropriate to the role discussed above:

- `Project 1` through `Project 4` contain individual streams for use by QA engineers.
- `Project 5` contains `Stream 3` and `Stream 4`. `Project 6` contains `Stream 5` and `Stream 6`. Both projects are organized by branch for use by developers and program managers.

Figure 3.3.2. Coverity Connect projects



Note

If you want to associate the same stream with multiple projects, you can use stream links. For more information, see Section 3.3.1.2.3, “Associating a stream with multiple projects” and Section 3.2.3.1.5, “Primary projects and stream links”.

3.3.1.2. Creating, copying, and deleting projects and streams

All project and stream configuration takes place in the Configuration → Projects & Streams menu. You must have appropriate Project and Stream permissions in order to create, edit, or delete them.

Important preconditions

Plan your project and stream configuration before using them in a production environment. For guidance, see Section 3.3.1.1, “Planning your project and stream configuration”. Also, decide whether and how streams should share triage data within and across Coverity Connect instances. For guidance, see Chapter 3.5, *Configuring Coverity Connect enterprise clusters*.

3.3.1.2.1. Setting up projects

When you create a project, you give it a name and description. You can also assign RBAC roles to it. For more information about roles and the permissions that are associated with those roles, see Section 3.2.3, “Roles and role based access control”.

To set up a project:

1. In the *Projects & Streams* menu, click **+Project**.

   Coverity Connect displays a dialog with a name similar to `New Project 141` and a Description field. Select the *Analysis License File*, if necessary, and click **Create**.

2. Select the project, and in the *Roles* tab, select one or more groups.

   Note

   Make sure that the roles assigned to that group are appropriate for the project. Click **Edit** to change the assigned roles. For guidance, see Section 3.3.1.2.6, “Assigning roles per project or stream”.

3. If necessary, assign one or more streams to your project.

   From the *Projects & Streams* menu, simply select and drag one or more streams to the project.

   If you need to assign a new stream, see Section 3.3.1.2.2, “Setting up streams”.

4. Click **Done** to save your changes and exit. If you need to change any of the information, click **Edit** in *Project Details*.

Note

To use local analysis, Coverity Desktop requires special projects to be created. For more information, see Section 3.12.1, “Configuring Coverity Desktop and shared files through the Downloads page”.

3.3.1.2.2. Setting up streams

The process of creating a stream is similar to creating a project. When you create a stream, you give it a name and description. In addition, you must associate the stream with a Triage Store. You can also select a component map, an issue category map (if more than one is set up), and one or more groups with permission to the stream. For more information about roles and the permissions that are associated with those roles, see Section 3.2.3, “Roles and role based access control”.

Note

See Important preconditions for using streams in a production environment.

To create a stream:

1. Select Configuration → Projects & Streams.

2. Decide where to create the stream:

   - If you select a project before creating a stream, Coverity Connect will automatically associate the stream with the selected project. Creating a stream under a project also makes the project the primary project for the stream (see Section 3.2.3.1.5, “Primary projects and stream links” for more information).

   - To create an unassociated stream, you need to select `Other Streams` and click the **+Stream** button. You can then associate the new stream with any existing project.

   Note that you can always associate a stream with another project (but not `Other Streams`) after creating it.

3. Click **+Stream**.

   Coverity Connect displays a dialog with a name similar to `New Stream 280`.

4. Select a programming language for the stream:

   - Any
   - C/C++
   - C#
   - Java
   - Dynamic Analysis
   - Other

   These settings determine the programming language's analysis results that you can commit to the stream. The default for a newly created stream is *Any*, indicating that the stream can accept commits from any of the supported languages. Furthermore, you can commit multiple analyses of mixed languages to the same *Any* stream. Coverity highly recommends that you choose *Any* for each new stream that you create.

   The other languages are provided for backward compatibility for previously released Coverity Connect versions (in implicitly designated languages were required for streams). Coverity recommends that you only select a specific language when you want to commit to the stream using pre-7.0 clients. If you define a specific language to a stream, Coverity Connect prohibits you from committing results from a different language.

   After a stream has been defined and an initial commit has been executed, it is possible to change the language for the stream and then commit the newly defined language analysis results to it. Use extreme caution if you plan to switch languages because the stream will not necessarily indicate any information about the stream's past commits from another language.

5. Optionally, select a component map for the stream.

   This setting associates the source stream with an existing component map. For information about component maps, see Step 1.

6. Optionally, select an *Issue Categorization*. If one or more issue categorization maps have been set up in Coverity Connect (see Section 3.1.1.6, “Configuring custom issue categories”), you can apply one to the stream. Issues found in future commits to the stream will use those issue categories.

7. Select a Triage Store for the stream.

   Each stream must be associated with a single Triage Store. For information about Triage Stores, see Section 3.3.4, “Managing triage stores”.

8. Optionally designate the stream as *Outdated.*

When you select this option, the stream and its data are effectively hidden from Coverity Connect user-oriented operations such as metric calculation, issue reporting, views displays, and so forth. Certain marked non-issue views (Functions, Files) will continue to include outdated issues in their aggregate counts, though following the link can bring the user to a view that excludes the specific outdated issues.

Note

A user can still commit to an outdated stream.

9. Click **Create** to save your changes and exit. If you need to change any of the information, click **Edit** in *Stream Details*.

Snapshots

Once you create the stream, it will include a *Snapshots* tab. For details, see Section 3.3.1.4, “Managing snapshots of streams”.

3.3.1.2.3. Associating a stream with multiple projects

You can use stream links to associate a single stream with more than one project. A stream link is a reference that points to a stream. A stream link inherits all groups (and associated roles) that are assigned to the stream to which it is a link and to the project with which that stream (not the stream link) is associated.

To create a stream link:

1. Select Configuration → Projects & Streams.
2. Select a stream to which you want to create a stream link.
3. Click **+Link**.
4. Select a *Project* from the drop-down list and click *Create*.
5. Click **Done** to save your changes and exit.

3.3.1.2.4. Copying and deleting projects and streams

Projects and streams have the same mechanism for copying and deleting:

- You can create a copy of a project or stream that will contain its original configuration settings. Copying is useful to preserve complex stream and component configuration that you might update in the original stream or project later.

  To create a copy, select a project or stream and click **Duplicate**. You can then edit the copied project or stream. CLick **Create** to save your changes.

- To delete a project or stream, select it, and click **Delete**. When the delete prompt appears, click **Delete** to continue with the deletion, or **Cancel** to exit without deleting the project.

The data belonging to a stream is deleted in the background while Coverity Connect is running. Some parts of this background activity conflict with other activities: for example ETL, garbage collection, commit actions, intra-cluster synchronization. The property `cim.cleanup.stream.delay.min` defines a delay that Coverity Connect introduces between deleting separate streams to give an opportunity for other conflicting activities to start. Before version 2020.03, the default value of this property was 30 (minutes), and starting with 2020.03 the default value is 2 (minutes). While it is possible to set the value of the property to 0, we do not recommend doing so. It might be worth doing temporarily to delete a large number of streams as fast as Coverity Connect can do it provided that starving all the conflicting activities is not considered to be a problem.

The property `cim.cleanup.stream.delay.min` also affects auto-deletion of expired streams. See "Designating a stream for auto-deletion of expired streams" for more information.

3.3.1.2.5. Exporting and importing streams

A system administrator can use the `cov-archive` command to export a set of streams into an archive file (and optionally delete the exported streams), import streams from an archive, or get information about an archive file (its version, the date and time of creation, the streams contained, and so on). You, the system administrator, can *export* one or more streams to an archive file that contains those streams and some associated entities. You may specify the streams to export either by name or by the project to which they belong.

Exporting and importing streams can be useful in situations like the following:

- You need to save one or more streams offline and to restore these to online status in future.
- You need to transfer a stream to another Coverity Connect instance for coarse-grained load balancing or for some other administrative purpose.
- You need to remove streams from Coverity Connect as part of an offlining or moving operation, or simply to reclaim resources.

Export and import actions are logged in the file `<CC_install_dir>/logs/cov-archive.log`. For more information and for examples, please see the `cov-archive` command in the *Command Reference*.

Note

You may import an archive into a Coverity Connect instance that has the same or a newer version as the Coverity Connect instance used to create the archive. You can check the Coverity Connect version used to create the archive using the `cov-archive list` command.

3.3.1.2.6. Assigning roles per project or stream

Coverity Connect supports role-based access control (RBAC) roles to projects and streams. For example, if you want a specific group of users (Group A) to be able to create streams in a given project, you can assign the *Stream Admin* role to that group. This role has permission to create streams. If you want another group of users to be able to manage issues in all streams in a given project, you can assign the *Developer* role to that group. For details about these roles, see Section 3.2.3, “Roles and role based access control”.

To assign a role per project or stream

1. In the *Projects & Streams* menu, click the name of the project or stream.
2. In the *Roles* tab, select a user or group to which you want to assign a role.

   Note

   If the user or group is not listed, you can add it by clicking *Add*.
3. Click **Edit** to open the *Roles* window.

   This window provides a list of roles that you can assign to the user or group.
4. Select one or more roles for the user or group.

   Note

   You can also deselect one or more roles to unassign the role.
5. Click **OK** to close the *Roles* window.
6. Click **Done** to finalize your changes and exit the screen.

Note

You can remove a group from the project or stream by clicking *Remove*.

3.3.1.2.7. Selecting an issue categorization map for a stream

Checkers automatically categorize issues and assign an impact level (High, Medium, Low, and Audit). For example, in the *Coverity* categorization, a C/C++ FORWARD_NULL checker can report a number of medium priority issues, one of which is an `unchecked dynamic_cast`.

Custom categorizations can be imported through Configuration → System → Issue Categorization, and applied to individual streams.

To select an issue categorization map:

1. Select Configuration → Projects & Streams.
2. Select the stream with which you want to associate an issue categorization map.
3. In *Stream Details*, click **Edit**.
4. Use the *Issue Categorization* drop-down to select your preferred issue category map.
5. Click **OK** to save your changes and exit.

Note

Note that the issue categorization map is applied to defect instances at commit time. Changes to issue category mapping will only take effect for future snapshots, and will not get applied to existing defects.

3.3.1.2.8. Associating an analysis license with a project

Coverity Connect projects are automatically associated with your default analysis license, however you may want certain projects to use different license files.

To select an analysis license file:

1. Select Configuration → Projects & Streams.
2. Select the project with which you want to associate the analysis license.
3. In *Project Details*, click **Edit**.
4. Use the *Analysis License File* drop-down to select your preferred license file.
5. Click **OK** to save your changes and exit.

3.3.1.2.9. Using search filters to find projects and streams

If you have a large number of projects and streams on your system, you can use the search filter field to locate them quickly. The search filter accepts a glob pattern and returns a list of projects and streams that match your search pattern.

To use the search filter:

- Enter a full or partial pattern. For example, if you enter `*dev*`, the filter will return all projects and streams that contain `dev` in their names.
- Click the clear icon (x) to clear the filter pattern and re-display all projects.

3.3.1.3. Editing projects and streams

The *Projects & Streams* menu lists all of the projects and streams that are in Coverity Connect.

To edit a project or stream:

1. Select the project or stream that you want to edit from the list in the *Projects & Streams* menu.

   This action opens the edit fields and tabs for the project or stream.

   You can filter the projects and streams. See Section 3.3.1.2.9, “Using search filters to find projects and streams”.

2. Click *Edit* or use the tabs to modify the project or stream.

   - Editing projects: For information about editable project properties, see Section 3.3.1.2.1, “Setting up projects”.
   - Editing streams: For information about editable stream properties, see Section 3.3.1.2.2, “Setting up streams”.

Note

Stream links are not editable. For details, see Section 3.3.1.2.3, “Associating a stream with multiple projects”.

3. Click **Done** to save your changes and exit.

3.3.1.3.1. Associating a stream with a project

You can associate one or more streams with a project.

To associate a stream with a project:

1. From the *Projects & Streams* menu, simply select and drag one or more streams to the project.

   Note

   You can use one or more stream links to assign a stream to multiple projects. For details, see Section 3.3.1.2.3, “Associating a stream with multiple projects”.

2. Click **Done** to save your changes and exit.

3.3.1.3.2. Managing daily trend records at the project level

The *Daily Trend Records* tab allows you to view and rebuild project-wide issue trends on a daily basis. To exclude the data that belongs to a deleted snapshot, rebuild the trend data.

By default, Coverity Connect gathers trend issue data nightly at 1 AM. However, you can choose to rebuild the trend data at any time (see Rebuilding trend data now for information about the data collection date).

Trend data management options:

- Viewing trend data:
  - Total daily trend record
  - Latest daily trend record
  - Earliest daily trend record
- Deleting and rebuilding all trend records:

  This option deletes all trend records from your system and rebuilds the trending data based on the most current state of the project.
- Deleting all records and rebuilding only trend records since a certain date:

  This option deletes trend records from a specified date and recomputes trending data based on the most current state of the project. You click the calender widget to select a date from which you want to recompute. Trending records that were computed before the specified date are preserved.

- Deleting and rebuilding the latest trend record:

  This option deletes the most recent trend record and recomputes trending data based on the most current state of the project. All previous trending records are preserved.

- Rebuilding trend records now:

  The data for the rebuilt records was collected up to 1 AM of the day on which you run the rebuild. For example, if you choose to rebuild trend records on October 5 at 2:30 PM, Coverity Connect will recompute data available up to October 5 at 1 AM.

For information about monitoring your trend data, see Chapter 2.3, *Monitoring issues*.

3.3.1.3.3. Designating a stream for auto-deletion of expired streams

Coverity Connect can automatically delete streams after a period of inactivity. Only streams that are specifically configured for this feature are eligible for automatic deletion. Streams can opt in to automatic deletion in one of two ways:

1. The Coverity Desktop plug-ins for Eclipse and Visual Studio create private streams, and these private streams are configured for automatic deletion.

2. Other streams can be configured for automatic deletion using Web Services API.

The configuration parameter which enables automatic deletion is only visible via Web Services. Every night, Coverity Connect deletes streams which meet all of the following criteria:

- The stream must be configured for automatic deletion, with the Web Services API setting `autoDeleteOnExpiry=true`.

- The stream must have at least one snapshot in it (it must have commit data).

- The stream must not have any snapshots (commits) within the last 28 days.

Streams meeting all three criteria will be deleted automatically. The Coverity Connect log will record each stream deleted.

To configure a stream for automatic deletion, the value `autoDeleteOnExpiry` must be set to true in the `StreamSpecDataObj` passed to the `ConfigurationService` operation `createStream`, `createStreamInProject`, or `updateStream`. See the Coverity Connect 2020.12 Web Services API Reference for details on how to access the SOAP web services.

The inactivity threshold is also configurable, by adding a line to `cim.properties`. The property name is `stream.expiration.inactivity.days`, and the value is specified in days. For example, to set the inactivity threshold to 60 days, add the following line:

```
stream.expiration.inactivity.days=60
```

The default value of 28 days is used if the property is not specified. The Stream Expiry feature can be turned off entirely by setting the inactivity threshold to 0; with this setting Coverity Connect will never delete streams automatically, even if they meet all 3 criteria for automatic deletion.

When copying a stream, either through Web Services or through the **Duplicate** button in the UI, the the hidden stream setting that controls whether the stream should be deleted automatically is not preserved. The copied stream will have this setting turned off.

3.3.1.4. Managing snapshots of streams

A snapshot of the stream is created when a stream receives issue data, at which point this tab will add the new Snapshot ID, the snapshot creation date, the user name of the person who committed the analysis results (the *committer*), along with optional data that can also be committed (the version, target, and description) to Coverity Connect through the `cov-commit-defects` command or can be defined in the Snapshots view type (users must have appropriate RBAC permissions).

You can click the ID to view additional snapshot attributes, build details, and analysis details. You can also delete snapshots from a stream.

3.3.1.4.1. Viewing snapshot data

Each stream that is listed in the *Projects & Streams* menu contains a *Snapshots* tab, which displays the values of attributes that are associated with each snapshot of the stream.

To view snapshot data for a stream:

1. Select a stream in the *Projects & Streams* menu.
2. On the *Snapshots* tab, select the *ID* to examine.
3. Click **Details** to view information about the snapshot.

   The following table provides information about properties of the snapshot.

   Table 3.3.1. Snapshot Attributes

Snapshot Attributes	**Value**
ID	A number that uniquely identifies the snapshot
Version	The version of the product being analyzed. This attribute is optionally set in Coverity Connect in the Snapshots view type, or on the `cov-commit-defects` command line.
Target	Intended to represent a milestone of a given product or code base, such as a release version number. This field is optionally defined in Coverity Connect in the Snapshots view type, or on the `cov-commit-defects` command line.
Description	The description of the snapshot. This field is optionally defined on the `cov-commit-defects` command line.
Date Committed	The date when the snapshot was committed.

Snapshot Attributes	Value
Committer	The username of the person who committed the issue data to Coverity Connect.

The following table provides build information on the snapshot.

Table 3.3.2. Build Details

Build Details	Description
Working Directory	The directory in which the source code is built.
Command Line	The command, options, and values used for the build.
Intermediate Directory	The location of the intermediate directory specified for the build command.
Configuration File	The name of the configuration file used during analysis. By default, it is `coverity_config.xml`
Build Host	The host name of the committed snapshot.
Build Time	The duration of the build.

The following table provides code analysis information on the snapshot.

Table 3.3.3. Analysis Details

Analysis Details	Description
Working Directory	The directory in which the analysis takes place.
Command Line	The command, options, and values used for the analysis.
Intermediate Directory	The location of the intermediate directory specified for the analysis command.
Configuration File	The name of the configuration file used during analysis. By default, it is `coverity_config.xml`
Analysis Host	The host name of the committed snapshot.
Analysis Time	The duration of the analysis.
Enabled Checkers	The list of checkers that were enabled for the analysis.

Note

If you committed Architecture Analysis data, the snapshot record will contain a *CVA* column that contains a `*.cva` file entry, from which you can download a CVA file. For more information, see Section 3.12.2, “Downloading a CVA file for Architecture Analysis”.

3.3.1.4.2. Deleting snapshots

Coverity Connect allows you to delete snapshots from a specified stream. Deleting a snapshot reverses the effects of a commit and can be used to undo a commit of mistaken, erroneous, or experimental analysis results.

When you delete the most recent snapshot in a stream, Coverity Connect removes all the data that comprises the snapshot (including issue states and file states) from the stream. For example, if you commit a snapshot and then immediately delete that snapshot, Coverity Connect will be in a state equivalent to the state it was in before the snapshot was committed.

When you delete an older snapshot, Coverity Connect updates its history as if the snapshot had never occurred. For example, assume that you have committed snapshot1, snapshot2, and snapshot3 (in order). Then you delete snapshot2. Coverity Connect's history now appears as if you only committed snapshot1 and snapshot3.

Deleting a snapshot affects other aspects of the system. The following important notes outline things you should consider before you delete a snapshot:

Trends and metrics

Trends and metrics are not automatically recalculated after a snapshot is deleted. If you delete a snapshot before the nightly metrics and trends data is collected, the data for the deleted snapshot will not be recorded as part of the trend and metrics data for that stream or issue. If you delete an older snapshot (one that has already been factored into trends and metrics data), you must manually recompute the data. For more information on recomputing trend records, see Section 3.3.1.3.2, “Managing daily trend records at the project level”.

Defect Manager run deletion vs. Coverity Connect snapshot deletion

Coverity Defect Manager had the ability to delete runs. This is a very different mechanism than the ability in Coverity Connect to delete snapshots. If you are accustomed to using Defect Manager and are new to Coverity Connect, it is important to note the distinction.

Deleting a run in Defect Manager deleted some of the information from a run but retained all of the triage information. Basically, it archived the run that was used to save space in the database while retaining the most important state information from a run. It worked on the assumption that a run was valid, but that you no longer needed to keep the additional data around and wanted to aggregate the information.

In Coverity Connect, you use snapshot deletion when a snapshot is not valid. This feature is used to correct mistakes. A snapshot should be deleted only if it was a mistake or otherwise invalid.

Do not use snapshot deletion to save space, as it will alter the history of your issues. See Section 3.1.2.2.2.2, “Important upgrade notes” for the proper procedure.

To delete a snapshot:

1. Navigate to Configuration → Projects & Streams, and select a `<Project>`.
2. Click the corresponding triangle icon to open the project, and select a stream name under the project.

3. In the *Snapshots* tab, select a snapshot to delete.
4. Click *Delete*.
5. In the confirmation window, click *Delete*.

Note

Snapshot deletion may not be an instantaneous process. To indicate that a deletion operation has been queued for a given snapshot, a yellow, rounded square icon will appear in the snapshot ID column.

3.3.1.5. Setting automatic issue ownership in streams

This tab assigns owners for unassigned issues in the selected stream. For more information about the entire automatic ownership configuration process, see Chapter 3.4, *Configuring automatic owner assignment*.

3.3.1.6. Setting Desktop Analysis options for streams

The Desktop Analysis tab (located under Configuration → Projects & Streams → Desktop Analysis) allows you to enable Desktop Analysis for a stream, and access advanced configuration options, described in the following list. See Chapter 3.7, *Server administration for Desktop Analysis* for additional information on server administration for Desktop Analysis. Click **Edit** to change the following settings.

Enable Desktop Analysis
: Click the *Enable Desktop Analysis* check-box to configure the selected stream for use with Desktop Analysis. The stream in question becomes a central repository for analysis summary information, as used by Desktop Analysis and Coverity Desktop.

Days before Summary Purge
: Summary data older than the number of days specified will be deleted.

3.3.2. Configuring triage attributes

Developers use these attributes to triage software issues through the *Triage* panel in Coverity Connect. The *Attributes* menu allows you to edit some of the built-in attributes and to create and edit custom attributes.

There are many conceivable scenarios for using a custom attribute. The following scenario uses a custom attribute to identify the version (or branch) of the code base in which a bug should be fixed.

Scenario: Targeting the code branch in which to fix a CID. Assume that developers are working on multiple branches of the same code base and that issue data from each branch is submitted to separate streams in a Coverity Connect project (Project 1):

- Branch 1 of the code base is analyzed and committed to streamV1 in Coverity Connect.
- Branch 2 of the code base is analyzed and committed to streamV2 in Coverity Connect.

In such a case, you might use the built-in *Fix Target* attribute to list all versions (or releases) in which bugs might be fixed (for example, v1 and v2). Developers can then mark the version in which the bug should be fixed. For example, assume that both code branches contain the same issue (CID 123). If CID 123 is a bug, developers might set the *Fix in* attribute to Version 2 (v2), for example, if there is no time to fix it in Version 1, or if the bug is of lower priority than others. However, if time permits, they might reset the *Fix in* attribute for the CID to Version 1 (v1). Both branches will share the same setting for the CID if the stream for each branch is associated with the same triage store (see Section 3.3.4, “Managing triage stores”). Note that after creating the appropriate values for the *Fix in* attribute, you also need to select the *Show in triage panel* option. For details, see Section 3.3.2.3, “Editing triage attributes”.

3.3.2.1. Creating a custom triage attribute

You can create up to 10 custom attributes and specify their attribute values.

To create a custom attribute:

1. Select Configuration → Attributes.
2. Click the *Add* button that is below the list of built-in and custom attributes.

   A screen displays with a name similar to *New Attribute Definition 108*.
3. Type a name and description for the attribute.
4. Indicate whether to display this attribute in the Triage panel so that developers can use it to triage issues.

   This panel appears in Projects → `<a project>` → `<a CID>` → Source tab.
5. For *Type*, select a value.
   a. Select the value type:
      - *Text*: Allows developers to type a value of their choice when triaging the issue.
      - *Pick list*: Allows developers to select a value from a list of pre-specified values when triaging the issue.

   b. If you selected *Pick list*:
      i. Click **+**.
      ii. Type a name for the value.
      iii. Select whether the value is the *Default* value for the pick list, and/or if it is *Deprecated*.
6. Click **Create** to save your changes and exit.

3.3.2.1.1. Custom attribute limits

Custom attributes have the following limits:

- Maximum number of custom attributes is 10
- Maximum custom attribute name length is 128 characters
- Maximum custom attribute description length is 255 characters
- Maximum length for a custom attribute text value is 1024 characters
- Maximum length for the name of a custom attribute pick-list value is 256 characters
- Maximum number of elements in a custom attribute pick-list is 50

In addition, there is a limit of 4096 characters for the combined length of all the custom attribute text values on a single issue.

3.3.2.2. Copying a triage attribute

You can make a copy of any attribute. Except for the name of the new attribute, all the properties of the copy will match those of the attribute you copied.

To copy an attribute:

1. Select Configuration → Attributes.
2. Select the attribute to copy.
3. Click **Duplicate**.
4. Edit the copy, as needed.

   For guidance, see Section 3.3.2.3, “Editing triage attributes”.
5. Click **Create** to save your changes and exit.

3.3.2.3. Editing triage attributes

You can modify *Custom* attributes, *Built In* (only *Action*, *Severity*, and *Fix Target*) attributes, and their values.

Note

You cannot change or delete the *Classification* or *Ext. Reference* attributes.

To edit an attribute:

1. Select Configuration → Attributes.
2. Select an editable attribute.
3. Click *Edit* to modify the attribute, as needed:

   To change the name or description of an attribute:

   - Type a new name or description for the attribute.

To display or hide an attribute from the Coverity Connect Triage panel:

- Set *Display*.

  On means that the attribute will appear the panel. *Off* means that it will not appear in the panel.

To add a new value to an attribute:

- Click + to show the *Values* field.

  You can create one or more predefined values or a text box into which developers can enter the value.

4. Modify any attribute values, as needed:

   To change the name or description of an attribute value:

   - Type a new name or description for the value.

     Note that CIDs associated with an old attribute *value* name will still refer to the old name.

   To delete an attribute value:

   - Select the value, and click - to remove it.

     Note

     You cannot delete an attribute value if an issue is using it.

     Note that you can deprecate an attribute value if you do not want to delete it.

   To deprecate an attribute value:

   - Click the *Deprecated* check box that is associated with the value.

     This action strikes through the name of the value. Developers cannot triage a deprecated issue. You can un-deprecate the value at a later date, if necessary.

     Note that you can delete an attribute value if you do not need it anymore.

   To reorder triage attribute values:

   - Select the value that you want to move, and then click *Up* or *Down* to change the position of the selected attribute value.

     The order that you specify affects the order in which the values appear in lists and the order used when sorting on its column in tables throughout Coverity Connect.

   To select a default value for an attribute:

   - Click the *Default* check box that is associated with the value that you want to set as the default.

5. Click **OK** to save your changes and exit.

3.3.3. Using components

The Coverity Connect components feature allows you to logically group source code files in named entities. Defining components allows you to:

- Filter issues and files to show the relationship between source code and development teams.
- Assign issues to only the users or groups that are responsible for a particular section of the code.
- Limit access to code and issues, for example, to address, intellectual property concerns, to prevent exposing third party code, or to prevent exposing vulnerabilities in the code.

Generally, you create components with source files of related functionality, such as libraries or software subsystems. For example, if a particular software group is working on a specific set of functions within a product, the group might only be interested in issues found in the source code of those functions.

A user with configuration privileges can create a component for the group containing the source files for these functions. For information on setting configuration privileges, see Section 3.2.3, “Roles and role based access control”.

Members of the group can view and filter these issues in the Source screen and receive automatic email notification of issues found in these files.

Note

After components have been added, deleted, or changed, users need to logout and login again for the changes to take effect.

3.3.3.1. Configuring components

Configuring and using the Coverity Connect components feature requires multiple procedures and options over several areas of the interface. This section provides procedures for setting up components. Subsequent sections provide more detailed User Scenarios that provide examples of component configuration and user interaction.

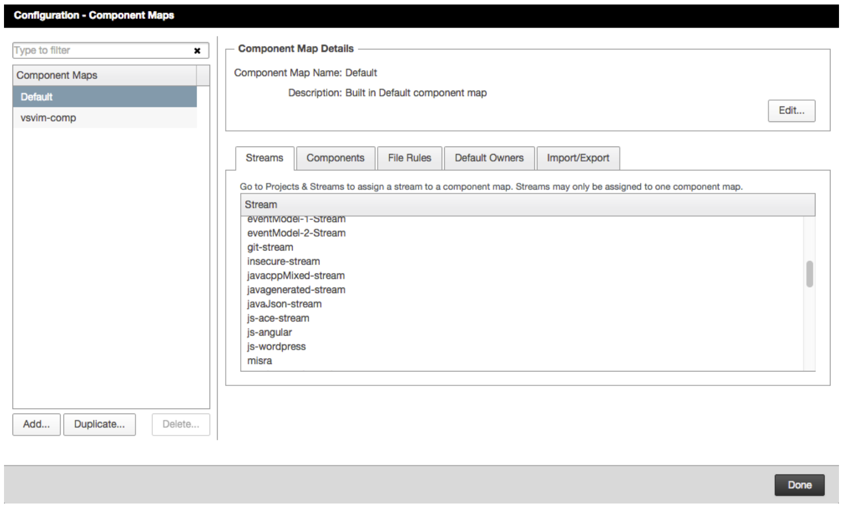
To configure a component:

1. Create a component map.

   A user with configuration privileges creates a component map that describes how to map source code files and analysis issues into components.

   UI Location: Configuration → Component Maps

Figure 3.3.3. Component map settings



Component Map Name: Default

Description: Built in Default component map

Go to Projects & Streams to assign a stream to a component map. Streams may only be assigned to one component map.

```
eventModel-1-Stream
eventModel-2-Stream
git-stream
insecure-stream
javacppMixed-stream
javagenerated-stream
javaJson-stream
js-ace-stream
js-angular
js-wordpress
misra
```

To create a new component map, select **Add**. To copy an existing component map, select a component map from the list and click **Duplicate**. The **Delete** button removes a selected component map.

After you have added or duplicated a component map, you define the following:

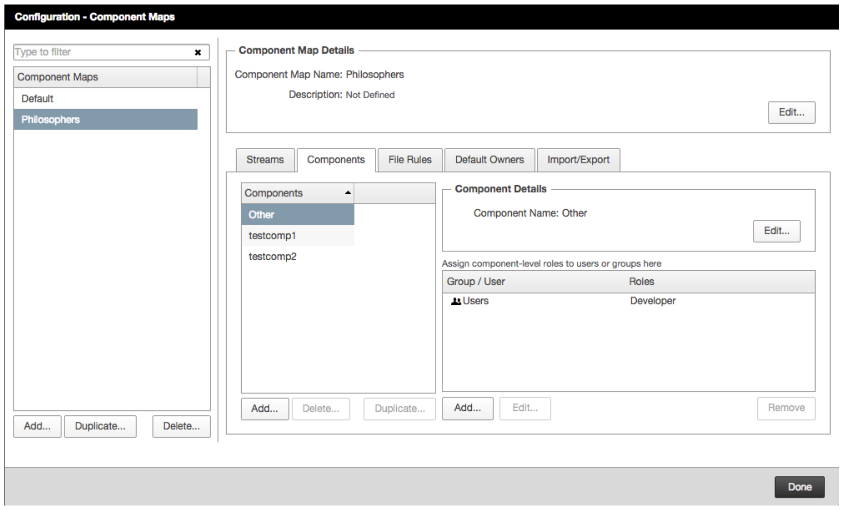
- *Name* - Uniquely identifies the component map.
- *Description* - Optional description of the component map.
- Stream associations
- Components
- File Rules
- Default owner

The *Default* component map is a Coverity Connect-created component map that consists of a single rule that maps to the catch-all `Other` component. Newly created streams and streams that have no component map associations are automatically associated with *Default*. *Default* can be edited or copied, but cannot be removed. When an existing component map is removed, the streams associated with it are re-associated to the *Default* component map.

2. Create the component.

   UI Location: Configuration → Component Maps → Components

Figure 3.3.4. Component settings



Component Maps: Default, Philosophers

Component Map Details

Component Map Name: Philosophers

Description: Not Defined

Components: Other, testcomp1, testcomp2

Component Details

Component Name: Other

Assign component-level roles to users or groups here

Group / User	Roles
Users	Developer

To add a new component, click **+**. To remove an existing component, select one and click -. The *Other* component cannot be removed.

After you add a new component, you define the following:

1. **Name**

   Uniquely identifies the component.

2. **Roles**

   The access control feature allows you to restrict users to viewing and triaging issues contained in a component. You define access privileges for users based on users and/or groups through the Coverity Connect RBAC feature.

   Each component can be associated with an ordered list of user or groups, in which each group can be assigned RBAC roles to limit or grant usage permissions. Any role can be assigned to a user or group within a component, but the only permissions that effectively limit or grant access are:

- *View issues*
- *View source*
- *Triage issues*

Coverity Connect ships with a number of pre-defined roles that contain these permissions, but you can create your own customized rules to limit or grant access. For more information about RBAC, see Section 3.2.3, “Roles and role based access control”.

For example, suppose that your system contains the following groups:

- `Users` contains all users on your system.
- `offshore` contains users that represent a part of the engineering department that are in a different locale from the local developers.

Within a component, you grant access for code and issues at the component level for the `users` group, but wish to restrict access to `offshore`:

`Users` - Assign a global role, such as Developer, that includes permissions for viewing and triaging issues

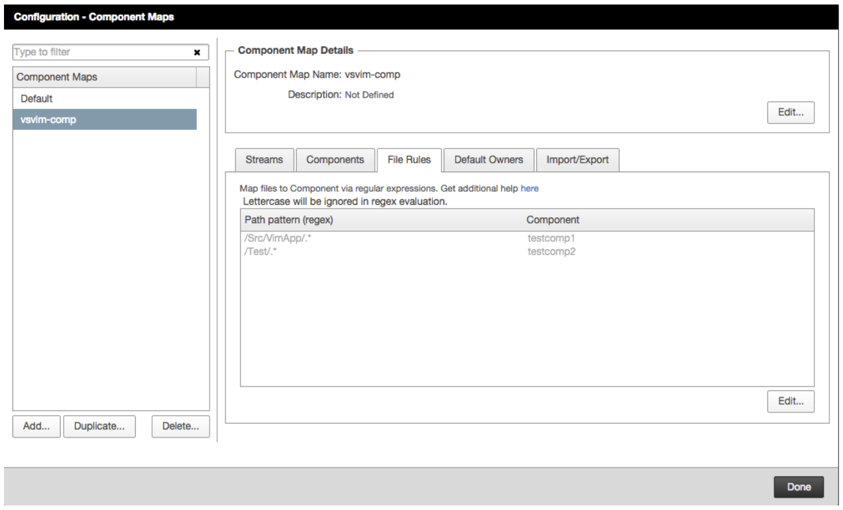
`offshore` - Assign a role, such as No Access on a given component.

Alternatively, you could grant a global No Access role for the `User` group, and grant a Developer role for `Offshore` at the component level.

3. File Rules

   UI Location: Configuration → Component Maps → File Rules

Figure 3.3.5. File Rules settings



Component Map Name: vsvim-comp

Description: Not Defined

Map files to Component via regular expressions. Get additional help here

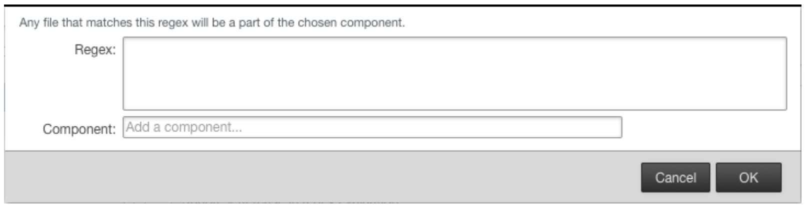
Lettercase will be ignored in regex evaluation.

Path pattern (regex)	Component
/Src/VimApp/.*	testcomp1
/Test/.*	testcomp2

To add a file rule:

a. Click **Insert Rule...**.

This displays the file rules edit dialog:



Any file that matches this regex will be a part of the chosen component.

Insert Rule... add a new rule expression immediately after the selected rule (if a rule is selected). If there is no selection, the new rule is added at the end of the list.

b. Enter a regular expression to map source files (see the notes below for more information).

c. Add the file rule to an existing component.

d. Click **OK**.

A file represents a source file in your code. It is identified by its fully-qualified file path. You can exclude path prefixes for filenames during the commit process by using the `cov-commit-defects --strip-path` option. For more information, see the `cov-commit-defects` documentation.

File rules contain path patterns to establish which files are mapped to the component. Use regular expressions to match a component's set of file names. For example, the following path pattern for `component1` matches any file contained in any directory named `/temp` within your directory structure:

`/temp/.*` (component1)

However, `/temp` might exist in multiple subdirectories, so for `component2`, you might want to match only a set of specific file types that exist under a specific directory. The following file rule returns only files with the `.c` extension in the `/subdir/temp` directory:

Caution

Avoid using a leading `.*` in these regular expressions. This can make the search considerably slower.

`/subdir/temp/.*\.c` (component2)

Coverity Connect maps each file to a component using the first regular expression that matches the entire absolute path of that file. Coverity Connect allows you to add multiple file rules per component map, and to set the order of precedence in which the file rules are executed.

Continuing with the example, if you set the following file rule order by selecting the rule and using the **Up** or **Down** buttons:

`/subdir/temp/.*\.c` (component2)

`/temp/.*` (component1)

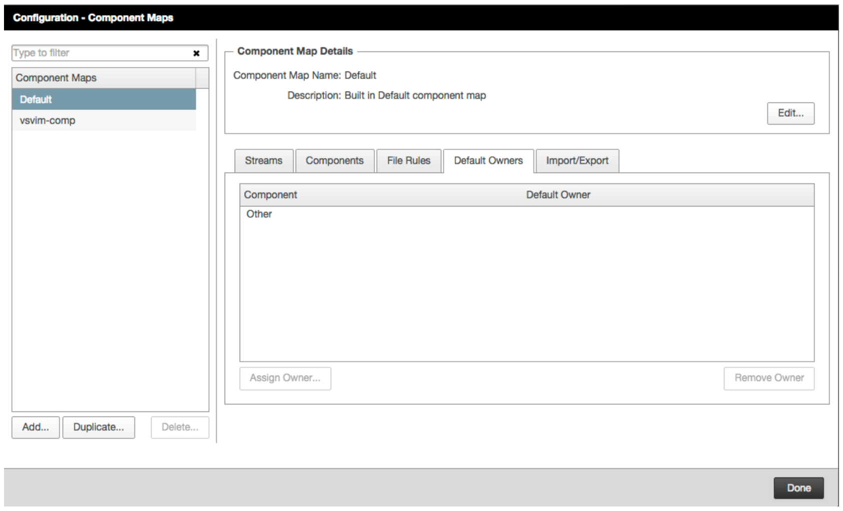
`component2` will contain only `.c` files contained in the `/subdir/temp/` path, while `component1` will contain any file under any `/temp` directory, *except* for `.c` files that exist under `/subdir/temp/`.

If you select *Ignore lettercase in regex evaluation*, all regular expressions will be evaluated without regards to case. If this option is not selected, all regular expressions will be evaluated with regards to case sensitivity.

4. Assign the default owners

   UI Location: Configuration → Component Maps → Default Owners

Figure 3.3.6. SCM system settings for automatic owner assignment



This optional step allows you to assign an owner to issues upon a commit based on the first issue component that matches the rule. Any user that is permitted to access a component can be designated as the default owner for all issues within the component. This feature is one of the configuration options for Automatic owner assignment.

To assign a default owner, select a component name and click **Assign Owner...**, and then type a user name in the *Default owner* field. `

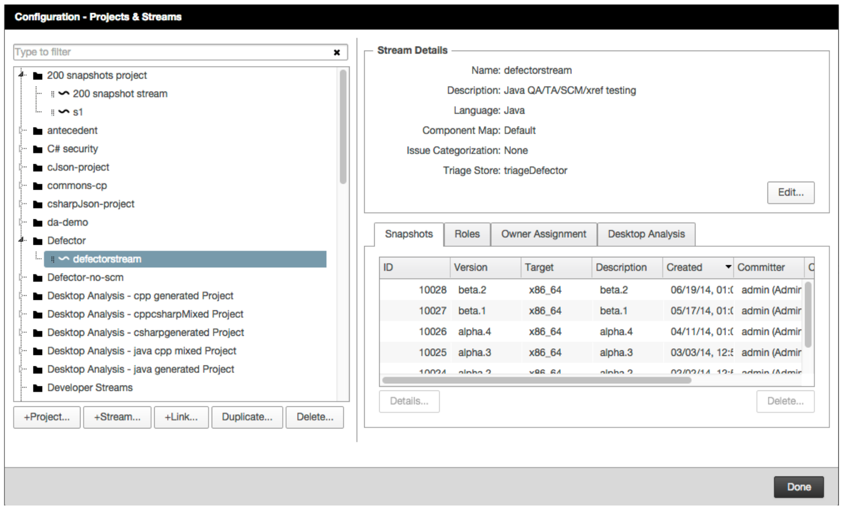
5. Associate streams with your component map.

   UI Location: Configuration → Projects & Streams

   Note

   The *Streams* tab under *Components* only displays the stream that are currently assigned to the component map. Stream association is configurable in the location listed above.

Figure 3.3.7. Stream settings



Stream Details

Name: defectorstream
Description: Java QA/TA/SCM/xref testing
Language: Java
Component Map: Default
Issue Categorization: None
Triage Store: triageDefector

ID	Version	Target	Description	Created	Committer
10028	beta.2	x86_64	beta.2	06/19/14, 01:(	admin (Admir
10027	beta.1	x86_64	beta.1	05/17/14, 01:(	admin (Admir
10026	alpha.4	x86_64	alpha.4	04/11/14, 01:(	admin (Admir
10025	alpha.3	x86_64	alpha.3	03/03/14, 12:5	admin (Admir

To associate a stream:

a. Select a stream from the Projects & Streams list.

b. Enter a component map name in the *Component Map* field.

c. Click **Done**.

When you associate a stream with a component map, it maps the source files in that stream into a component.

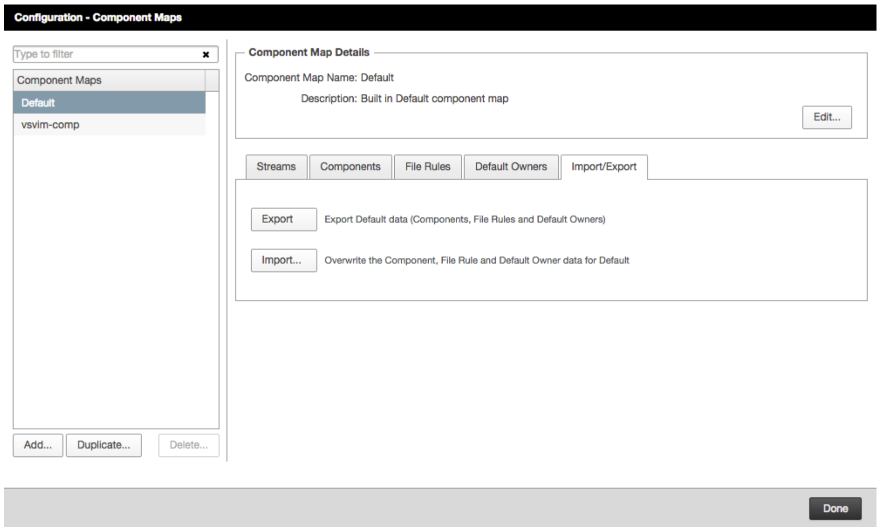
A given stream can only be associated with exactly one component map. However, a component map can contain multiple streams. If a component map is not specified, the stream is associated with the `Default` component map.

3.3.3.2. Importing and exporting component maps

The *Component Maps* configuration screen also allows you to import and export previously configured component maps.

UI Location: Configuration → Component Maps → Import/Export

Figure 3.3.8. Component map import/export


Configuration - Component Maps
Type to filter
Component Maps
Default
vsvim-comp
Add...
Duplicate...
Delete...
Component Map Details
Component Map Name: Default
Description: Built in Default component map
Edit...
Streams
Components
File Rules
Default Owners
Import/Export
Export
Export Default data (Components, File Rules and Default Owners)
Import...
Overwrite the Component, File Rule and Default Owner data for Default
Done

Export

The **Export** feature creates a JSON file with all of the relevant information for the selected component map. This includes components, file rules, and default owners.

The downloaded file can then be edited and imported back into Coverity Connect.

See Section 3.3.3.2.1, “Exported component map JSON elements” for details on the exported JSON elements.

Import

The **Import** feature accepts a component map JSON file, and uses it to overwrite the components, file rules, and default owners for the selected component map. Section 3.3.3.2.1, “Exported component map JSON elements” contains information on the various component map JSON elements.

There are several important considerations for formatting the import file:

- The format of the import file should match the exported JSON. Aside from required elements, missing fields will be considered empty, and will be deleted if they existed prior to the import. The only exceptions to this are the `defaultOwners` and `rbacSettings` fields. An empty `defaultOwners` or `rbacSettings` field will simply be ignored.

  If you want to delete `defaultOwners` or `rbacSettings` through import, set their value to `null`.

- The imported JSON must contain exactly one component with the name "`Other`".

- Import can not be used to create component maps. If you want to import into a new component map, you need to create a new component map by clicking **Add**, and then import the component map data.

- If a component's `name` is changed in the JSON, any file rules (`fileRules`) that refer to that component must also use the new component name (via the `componentName` field).

3.3.3.2.1. Exported component map JSON elements

The following table lists and describes the elements contained in the JSON file that is produced when you use the **Export** button in the Component Maps menu.

Table 3.3.4. Exported component map JSON elements

Element	Description
`version`	Specifies the version of the component map JSON format. This may be excluded, but if present, the value must be 1.
`name`	The name of the component map. This is a required element.
`description`	Optional description of the component map.
`components`	Contains each of the components within the component map. Each component object contains the following child elements: `name, description, defaultOwner,` and `rbacSettings`.
`name`	The name of the component. This is a required element. No duplicate names are allowed in the JSON file.
`defaultOwner`	The username of the default owner in the format `username@ldap` (or just `username` if `ldap` is null.
`rbacSettings`	Contains desired RBAC settings for the component map. This element is optional for import. If it is missing, the existing RBAC settings will be unchanged. The rbacSettings element contains the following child elements: `groupOrUser, principalName,` and `roles`.
`groupOrUser`	Specifies whether the RBAC settings apply to a group or a single user. This element is required (if `rbacSettings` is present) with a value of "`group`" or "`user`".
`principalName`	Specifies the group or username the RBAC settings apply to. This is a required element if `rbacSettings` is present.
`roles`	A list of role names that apply to the group or user specified by `principalName`. This is a required element (if `rbacSettings` is present), and must contain a non-empty list of role names.
`fileRules`	Contains each of the file rules within the component map. Each file rule object contains the `componentName` and `pathPattern` child elements. This element may be excluded, which will be interpreted as a request to delete all existing file rules.

Element	Description
`componentName`	Component name that exists in this JSON file. This component will be mapped to all files that match the sibling `pathPattern` element. This is a required element if `fileRules` is present.
`pathPattern`	A valid regex pattern. Any files matching the pattern will be mapped to the component specified by the sibling `componentName` element. This is a required element if `fileRules` is present.

3.3.3.3. User scenarios: components

The following scenarios illustrate the use of components through the Coverity Connect interface. The first scenario focuses on component configuration and the second scenario describes the work-flow of a user who is assigned to the component.

The recommended configuration pattern is to have streams which are based on the same code base share a single component map. Streams which are unrelated should generally not share a component map. For example:

- Code-A version1 - Uses component map "X"
- Code-A version2 - Uses component map "X"
- Code-B version1 - Uses component map "Y"

This way, the same components and file rules will apply to any stream associated with a component map, while maintaining the best performance possible.

3.3.3.3.1. System overview

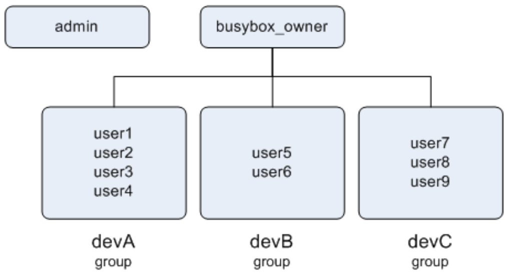
The examples in the following scenarios use the BusyBox open-source project to represent a product's code base for an unnamed organization. The codebase was analyzed using Coverity Analysis, and was committed to the following streams, each representing a different way in which the software is built:

- `allyesconfig`
- `allnoconfig`
- `defconfig`
- `randconfig`

The user scenarios in this section concentrate on a project called `busybox_dev` which contains the `allyesconfig` and `allnoconfig` stream associations.

The following organizational chart displays the users that are key participants in the `busybox_dev` project, and the development groups to which they belong:

Figure 3.3.9. busybox_dev users



admin
: The Coverity Connect system administrator. Responsible for creating/importing users, configuring groups, setting user permissions, and so forth.

busybox_owner
: Leads a team of nine developers and is responsible for team's overall productivity and for assigning issues and project tasks to developers. Knows the exact issues contained in the project areas for assignment. Manages multiple groups and requires the groups to have limited access to certain portions of the codebase. Has configuration (Project Owner) privileges.

user1-user9
: Use Coverity tools to find and resolve issues in the codebase. Have User privileges to view, filter, and triage issues.

Note

For information about creating users, creating groups, and adding users to groups, see Chapter 3.2, *Managing users, groups, and roles*.

3.3.3.3.2. Scenario 1: Component configuration

User: `busybox_owner`

Goal: To create and configure a component map called `busybox` that contains the following components:

core_libs
: This component represents a section of core library files that exist in the following directories:

- `libbb`
- `include`

The `devA` group is the team of developers that primarily works on this section of code. The `devC` group occasionally works on this section of the code.

IO

This component represents input and output utilities that exist in the `archival` directory. The `devB` group is the team of developers that primarily work on this code. The `devA` group occasionally works on this section of the code.

util

This component represents a series of utilities located in the following directories:

- `util`
- `util-linux`

The `devB` group is the team of developers that work on this section of code.

3rd_party_code

This component represents sections of the code developed by a third party. The files and issues contained in this component are not to be exposed to Coverity Connect users.

Configuring components:

1. `busybox_owner` creates a component map.

   a. Goes to Configuration → Component Maps.

      Coverity Connect automatically creates the `Default` component map. This component map can be copied or edited, but not deleted.

   b. Clicks **Add** to create a new component map.

   c. Enters `busybox` as the component map name.

   d. Adds a description of the component map (this step is optional).

   e. Instead of adding a new component each time, `busybox_owner` can use the **Duplicate** button to copy an existing component map and then edit it.

2. Adds the first component definition for the `busybox` component map (more components are added later in this scenario).

   a. While the `busybox` component map is selected, `busybox_owner` clicks **+** to add a new component and names it `busybox`.

   b. Selects `busybox` to edit it.

3. Defines file rules in the *File Rules* tab.

   a. Uses a regular expression pattern to match files that belong to a component. File rules are added by using the **Insert Rule...** button.

      `busybox_owner` adds the following file rules:

`/libbb/.*`
: Maps all files and issues contained in the `/libbb` directory to this component.

`/1_9_2/include/.*`
: Maps all files and issues contained only in the `/1_9_2/include/` directory to this component. This file rule specifies the `/1_9_2` because other `/include` subdirectories exist, but the files contained within them are not applicable to the development groups associated with this component.

   b. For each file rule added, `busybox_owner` selects and assigns `core_libs` from the *Components* auto-complete field..

4. (This step is optional) - In the *Default Owner* tab, `busybox_owner` chooses to define `user1` as the default owner for the `core_libs` component.

5. Goes to the *Components* tab to add access control for the development groups working on the files and issues defined in the component.

   a. Clicks Add open the RBAC selection list.

   b. Enters `devA` in the *Group/User* field.

   c. Assigns the *Developer* role to the group.

   d. `busybox_owner` enters `devC` and assigns the *Developer* role.

6. `busybox_owner` associates streams with the component map.

   a. Goes to Configuration → Projects & Streams.

   b. Selects the `allnoconfig` stream and adds the `busybox` component map to it.

   c. Selects the `allyesconfig` stream and adds the `busybox` component map to it.

   d. Clicks **Done** to finalize the changes and exit the screen.

   Note

   If `busybox_owner` copies a stream (for example, `allyesconfig`), the copied stream is associated with the same component map (`busybox`).

7. Repeats the Add Component procedure for the `IO` component with the following configuration:

   File Rules: `/archival/.*`

   default owner: `user6`

   Access Control: `devB - Developer`, `devA - Developer`

8. Repeats the Add Component procedure for the `util` component with the following configuration:

File Rules: `/runit/.*\.c`

`/util-linux/.*\.c`

These files rules return files with the `.c` suffix in the designated directories.

default owner: `user6`

Access Control: `devB - *Developer*`

9. Repeats the Add Component procedure for the `3rd_party_code` component. This component is defined so that third party code, and its associated issues, are not exposed. In this scenario, header (`.h`) files that exist in directories other than those defined in the previous components represent third-party code. The configuration is:

   File Rules: `\.h`

   default owner: Not set (no action required).

   Access Control: `devC - Developer`, `devA - Developer`

10. Establishes the file rule order in the *File Rules* tab.

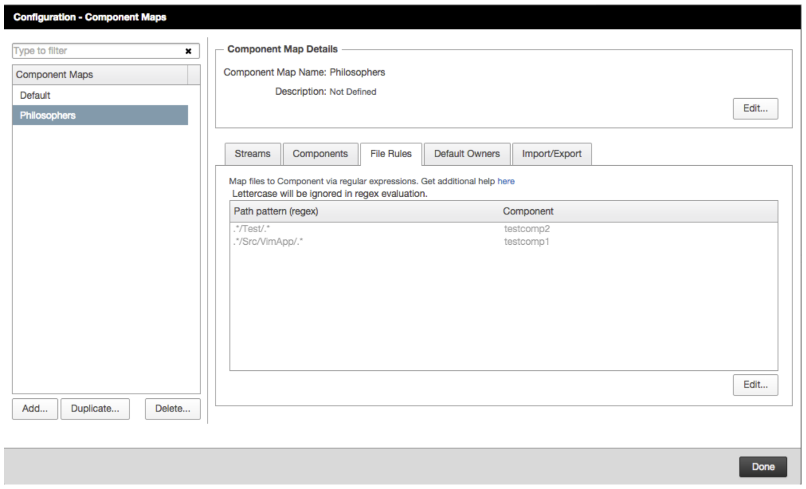
    If a file matches multiple component specifications, it is assigned to the first component in the list (from top to bottom). `busybox_owner` changes the order of components as follows:

    a. Selects the file rule, and then clicks the **up** or *down* buttons.

    b. Repeats this process (if necessary) to get the desired ordering.

    c. Clicks **Done** to finalize the changes and exit the screen.

    Figure 3.3.10. File Rules tab



Component Map Name: Philosophers

Description: Not Defined

Map files to Component via regular expressions. Get additional help here

Lettercase will be ignored in regex evaluation.

Path pattern (regex)	Component
.*/Test/.*	testcomp2
.*/Src/VimApp/.*	testcomp1

11. Observes the effects of changes to the component rules in the configuration area prior to applying the changes.
12. Clicks **Done** to update all changes in the system.

3.3.3.3.3. Scenario 2: User work-flow

User: `user1`

Goal: This scenario describes the work-flow of the default owner, `User1`. From the `User1`'s perspective, the work-flow does not greatly differ from a "normal" work-flow; that is, one without component definitions. So, this scenario describes the specifics of how the component mechanism works within the flow.

Filtering issues:

1. `user1` signs into Coverity Connect and selects the `busybox` project.
2. Observes *Component* as a default column in the Issues: By Snapshot list.
3. Observes that either `core_libs` or `IO` (the components to which `user1` belongs) is the component name for each issue.

   Note

   If more than one component map exists in the project, the component name is identified as follows:

   `<component_map_name>.<component_name>`

   For example, `busybox.core_libs`

   In this case, only one component map exists, so the name displays as `core_libs`.
4. Observes that `user1` is the owner for all issues that occur in `core_libs`.

   `user1` was designated as the default owner during the component's configuration.
5. Selects `core_libs` and *Include* in the Components filter.

   The issues list displays only the issues contained in `core_libs`.

   Component filtering is CID-based, so an issue is included even if only one of its occurrences is in the included component. Components that are invisible to `user1` (through access control or exclusion) do not appear in the filter.

   The *Include* selection displays only the issues for the component specified in the filter. The *Exclude* button displays all of the issues that `user1` has access to outside of the specified component.

Examining, locating, and triaging issues:

1. Clicks the Files view type and then the In Latest Snapshot view to observe the correct issue markers and file trees that indicate the existence of issues in files, directories, and components.

`user1` can only view and open the files and issues to which he or she is assigned access.

2. Observes issues that are not in the latest snapshot (fixed/dismissed) as associated with components according to the current component mapping.

3. Clicks the gear icon to edit filters.

4. Selects `core_libs` and *Include* in the Components filter to focus the list of issues. This includes issue links in source file display (when no issue is open). Filtering is CID-based; that is, an issue is included even if only one of its occurrences happens to be in the included component.

5. Observes the issue markers in the file tree, which was filtered according to issue filtering criteria.

6. After the list of issues is filtered and a group of issues is selected into the desired order, `user1` begins to triage the list of issues.

   The process of triaging issues is described in Section 2.2.4, “Triaging issues”. However, the following notes apply to triaging issues through the use of components:

   - `user1` can only select an owner for the issue from the list of users that can access the issue based on their component assignment.

   - `user1` can only retrieve or update stream issues that are contained in the assigned components (`core_libs` and `IO`).

   - `user1` can only view issues that are contained in the assigned components (`core_libs` and `IO`) in the issue history.

Monitoring issues:

Monitoring issues allows `user1` to track and create reports of the issues in the components to which he or she is assigned. For more information, see Part 2, “Coverity Connect Usage”.

Setting up component-based notification:

1. Clicks *Preferences* from the User Session tool bar.

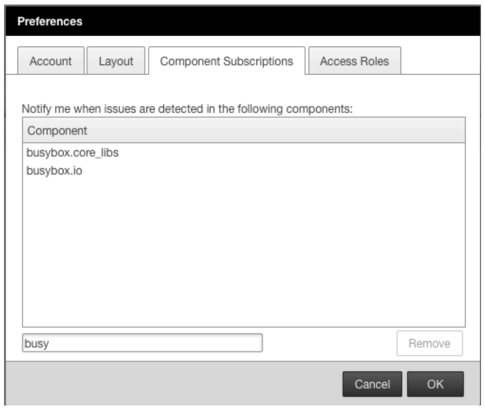
2. Clicks the Components Subscriptions tab.

   Note

   Component notification requires the setup of a notification script using the Web Services API.

3. Subscribes to new issue notification by selecting the `busybox` component map, and the `core_libs` and `IO` components.

Figure 3.3.11. Component Subscriptions window



4. Clicks **Done**.

   `user1` will receive new issue notifications when new issues are committed.

   Users are alerted if they subscribe to a restricted component to which they have no access.

3.3.4. Managing triage stores

A triage store is a repository for the current and historical triage values of CIDs. The state of triage values for a CID changes when a user triages a CID, for example, by changing its classification from *Unclassified* to *Bug*.

In Coverity Connect, each stream must be associated with a single triage store so that users can triage issues (instances of CIDs) found in the streams. You can use one triage store for all Coverity Connect streams, or you can associate streams with separate triage stores. When streams are associated with the same triage store, any instances of a CID that are found in the streams will share the same triage values, regardless of the project to which the streams belong.

Triage store configuration and management takes place through the *Triage Stores* menu.

Figure 3.3.12. Example: *Triage Stores* menu

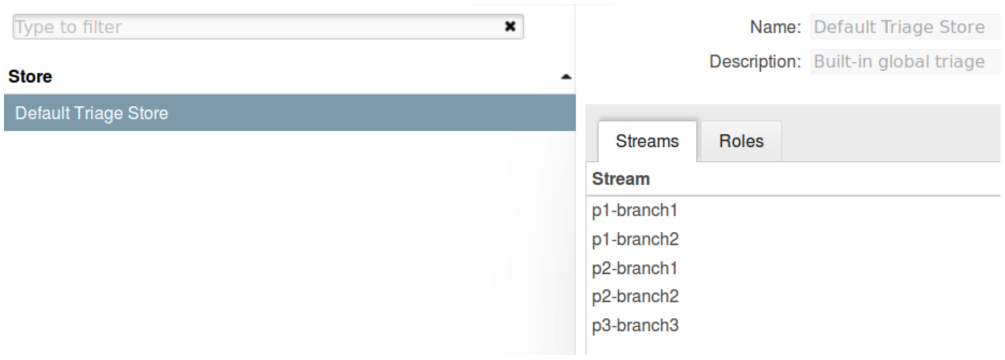


Figure 3.3.12, “Example: Triage Stores menu” shows a *Default Triage Store* configuration.

Note

The *Triage Stores* menu is viewable but not editable on Coverity Connect Subscribers. Subscribers are instances of Coverity Connect that share their triage values through one or more triage stores that are set up in a Coordinator instance. For details, see Section 3.5.1, “Synchronizing multiple Coverity Connect instances”.

The Coordinator instance of Coverity Connect can only manage streams that are local to the Coordinator. For example, it is not possible to use the Coordinator instance to associate a stream from a Subscriber instance with a triage store. Instead, an administrator would need to use the *Projects & Streams* menu on the Subscriber for this purpose. The Coverity Connect Web Services API also supports this functionality (see the Coverity Connect Web Services Reference).

The *Triage Store menu* supports the following actions:

- Adding a triage store:

  Creates a triage store. The new store does not contain any triage values, triage history, or stream associations. Note that if you associate a stream with a newly created triage store, any CIDs found in the stream will acquire the default triage values.

  For required permissions, see Assigning roles. A user who adds a triage store becomes the *Triage Store Owner* of the new triage store. The new store does not have any other role associations.

  The **Add** button supports this functionality.

- Associating one or more streams with a triage store:

  Assigns the triage values that are stored in a triage store to CIDs in the associated stream (or streams). Note that these CIDs will not retain triage data from any triage store with which their stream was previously associated. If a CID has not been triaged in a triage store with which a stream is associated, the CID will receive the default triage values. For example, assume Scenario 1: Triage Stores:

  Scenario 1: Triage Stores

  - Stream A is currently associated with TS1, and Stream A contains an instance of CID 123 and CID 456, both with a classification of *Pending* in TS1.

  - The instances of CID 123 in streams currently associated with TS2 are classified as *Bug*. CID 456 has never been triaged in TS2.

  In this case, if you change the association of Stream A from TS1 to TS2, the classification of CID 123 in Stream A will change from *Pending* to *Bug*. However, the classification of CID 456 will be *Unclassified*, the default triage value for the *Classification* issue attribute.

  If you disassociate a stream from a triage store, the store will retain the triage values for CIDs in that stream.

  For required permissions, see Assigning roles.

The *Streams* tab supports this functionality.

Note

Stream links are associated with a triage store through the streams that they reference, so stream links share the same triage data as the streams they reference. For this reason, stream links are not listed in the *Triage Stores* menu. Triaging a CID in a project that, for example, contains only a stream link will update the triage store with which the referenced stream is associated.

For additional information about the relationship of stream links to projects, see Section 3.2.3.1.5, “Primary projects and stream links”.

- Deleting a triage store:

  Removes a triage store from Coverity Connect along with all its triage data and role associations. Note that you cannot delete the *Default Triage Store* or a triage store that has streams associated with it. If you need to delete a triage store, you must first associate its streams with another triage store.

  For required permissions, see Assigning roles.

  The **Delete** button supports this functionality.

  Note that if you delete a triage store on a Coordinator instance of Coverity Connect when streams from a Subscriber instance are associated with the store, Coverity Connect will associate the streams from the Subscriber instance with the Empty Triage Store.

- Exporting a triage store:

  The **Export** button creates a JSON file with all relevant triage data from the selected triage store. All triage history and attribute information, including any custom attributes, is included in the exported file.

  See Section 3.3.4.5, “Exported triage store JSON elements” for details on the exported JSON elements.

- Importing a triage store:

  The **Import** button accepts a JSON triage data file and uses it to create a new triage store. Section 3.3.4.5, “Exported triage store JSON elements” contains information on the various triage store JSON elements.

  There are several important considerations for formatting the import file:

  - In general, if a triage data file does not specify a value for a particular field, it will be given the default value (e.g. owner will be “unassigned”).
  - Validation of imported triage data will only succeed when all the referenced data, such as users and custom triage attributes, are already available on the target Coverity Connect instance.
  - Importing the JSON triage data file creates a new triage store, it does not overwrite an existing one.

  - Import is not available to subscriber instances of Coverity Connect. Coordinators may import triage stores, which will be reflected on subscriber instances.

  - The importing of triage data will create Merged Defects (identified by their merge key value) if they do not yet exist on the target Coverity Connect instance. If a CID is specified, the merged defect will be assigned that CID value.

    This can cause collision errors in the following scenarios:

    1. The merged defect already exists on the target Coverity Connect instance, but has a CID value that differs from the imported triage data.

    2. The merged defect does not yet exist on the target Coverity Connect instance, but its specified CID value is already used by an existing defect.

    In the event of a collision, the error will be flagged, and the user should edit the import file to remove or modify the offending CID value.

- Assigning roles:

  Allows you to assign to users or groups one or more roles that have permissions to a triage store (or to other features in the UI).

  To add a triage store, a user or group requires the *Create Triage Stores* permission. The *System Admin* and *Project Admin* roles have this permission by default.

  To delete triage stores, a user or group requires the *Manage Triage Stores* permission. The *System Admin* and *Triage Store Owner* role have this permission by default.

  To associate a stream with a triage store, a user or group requires the *Manage Streams* permission on the stream. The user or group also must have one of the following permissions to the triage store: *Manage Triage Stores*, *View issues*, or *Triage issues*. To discover the built-in roles that have these permissions by default, see Figure 3.2.1, “Built-in roles”.

  To triage CIDs found in streams that are associated with a triage store, a user (or group to which the user belongs) requires the *View issues* and *Triage issues* permissions at the global level or at three separate lower levels (a triage store, a component, and a project or stream). By default, the *User* group is assigned the *Developer* role, which has this permission at the global level. To use the permission at lower levels, you might add to a triage store, component, and stream a user (or group) with a role that includes the *Triage issues* and *View issues* permissions. (Note that you also need to assign any additional permissions that the user requires to work with the stream and component, such as the ability to view issues and view the source.) For a detailed example, see Section 3.2.3.2.1, “Scenario: Granting triage permissions on a synchronized Coverity Connect enterprise cluster ”. To understand permissions and associate them with a role, see Section 3.2.3, “Roles and role based access control”.

  The *Roles* menu supports this functionality.

Coverity Connect provides two built-in triage stores.

Built-in Triage Stores

- *Default Triage Store*:

  Serves as a default store for new streams (though it is possible to associate a stream with a different store when creating or editing the stream). In addition, this built-in triage store cannot be deleted.

  If you decide that all your streams should share a single triage store, Coverity recommends that you use this store instead of creating a new one for that purpose. For details on this topic, see Section 3.3.4.1, “Scenario: Sharing triage data across code branches (Recommended)”.

  For required permissions to the *Default Triage Store*, see Assigning roles.

- *Empty Triage Store*

  This store will be empty except in the case that it contains orphaned streams.

  Users cannot triage instances of CIDs contained in streams that are associated with this store. To enable triage, you need to associate the streams with another triage store. This is the only operation you can perform with the *Empty Triage Store*.

  Coverity Connect associates streams with this store under the following conditions:

  - When creating a stream, if a user or administrator does not have permission to any triage stores. In other words, the user is not assigned a role that has the *Manage Triage Stores*, *View issues*, or *Triage issues* permission on *any* triage store.

  - After deleting a triage store on a Coordinator instance when streams from a Subscriber instance are associated with the store. Coverity Connect will associate the streams from the Subscriber instance with the *Empty Triage Store*.

Triage scope. When a user triages a CID in a project, the update applies by default to all triage stores that are associated with streams (in the project) that contain that CID. However, it is possible for users to perform triage on select triage stores. See ???TITLE??? for more information.

User-set attributes for CIDs:

- *Classification* attribute (default value: *Unclassified*)
- *Severity* attribute (default value: *Unspecified*)
- *Action* attribute (default value: *Undecided*)
- *Fix Target* attribute (default value: *Untargeted*). This built-in attribute is hidden from the Triage pane by default.
- *Ext. Reference* attribute (default value: an empty field)
- *Owner* attribute (default value: *Unassigned*)
- *Comment* attribute (default value: an empty field)
- Custom issue attributes

For more information about issue attributes and values, see Section 2.4.5, “Triage attributes”.

Note

Users *do not* set the *Status* of a CID (*New*, *Triaged*, *Dismissed*, or *Fixed*), which is set by Coverity Connect.

3.3.4.1. Scenario: Sharing triage data across code branches (Recommended)

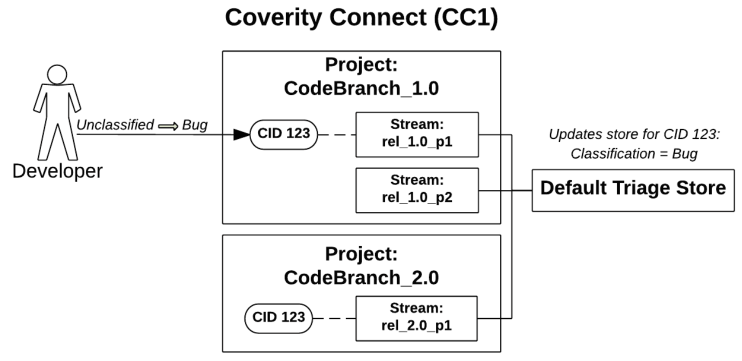
Issues share the same CID when Coverity determines that the same fix is required for them, and most instances of a given CID are very similar issues that are found in separate code branches.

In general, managing and tracking issues is simplified if all instances of a given CID are triaged in the same way, regardless of the code branches in which they are found. Historically, Coverity customers have usually triaged the same CID differently (for example, marking the CID as a *Bug* in the main development branch but as *Intentional* in the maintenance branch) when it was risky to fix the CID in the maintenance branch. However, the *Fix Target* attribute (introduced in version 6.0.2) diminishes the need to triage the same CID differently (for details, see Alternative to using separate triage stores). Further, in many cases, the same developer is the *Owner* of a given CID. Because of such considerations, Coverity recommends using a single triage store (the Default Triage Store) for all streams.

In the simplest case, where all streams across projects in a given Coverity Connect instance share the same triage store, the triage values for a CID that occurs in these streams will always be unified across streams in all projects.

In Figure 3.3.13, “Example: Updating Triage Value, Single Triage Store”, the Coverity Connect streams share the same triage store (Default Triage Store). When a user triages CID 123 in the `CodeBranch_1.0` project (by changing the classification from *Unclassified* to *Bug*), the Default Triage Store is updated with the new triage values for CID 123. Because the streams `rel_1.0_p1` and `rel_2.0_p1` are associated with the same triage store, they share the same triage values for CID 123.

Figure 3.3.13. Example: Updating Triage Value, Single Triage Store



The following procedure explains how to set up the scenario described in Figure 3.3.13, “Example: Updating Triage Value, Single Triage Store”.

To use a single triage store:

1. When creating a stream, always associate the stream with the *Default Triage Store*.

For information about creating streams, see Section 3.3.1.2.2, “Setting up streams”.

2. If any streams are currently associated with another (non-default) triage store, determine whether to re-associate the streams with the *Default Triage Store*. You can use the drop-down menu in the *Streams* tab to associate the streams with the *Default Triage Store*.

3. If necessary, use the *Roles* tab for the *Default Triage Store* to assign any necessary roles to users or groups.

   If a user or group needs a different set of roles, you can use the **Edit** button to select or deselect its roles.

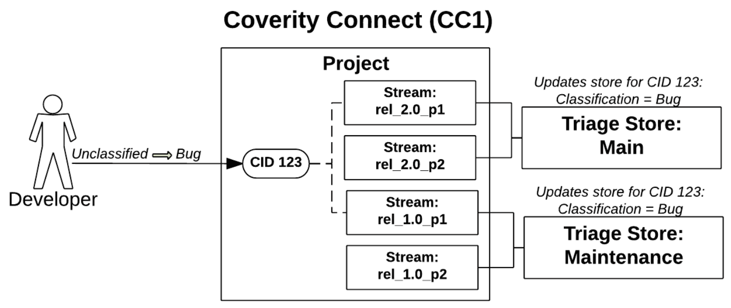
   For example, one role might allow your developer groups to triage issues in the store, while another role might allow an administrative group to manage the *Default Triage Store*.

4. Click **Done** to save your changes and exit.

3.3.4.2. Scenario: Separating triage data by code branch

A company might decide to create separate triage stores for the currently active (main) and maintenance (for example, hotfix or patch) code branches so that users can triage the same CID differently depending on the code branch. Figure 3.3.14, “Example: Updating Triage Value, Multiple Triage Stores” shows an example of such a case. Here, there is a single Coverity Connect project for streams that contain issue data from active branches and maintenance branches of the code base. This setup allows users (with the appropriate permissions) to triage instances of a CID (for example, CID 123) in both triage stores (the default) or only in one of them by selecting the triage stores. (See ???TITLE??? for more information.) For example, users might decide to ignore an instance of a CID in the maintenance branch (setting the *Action* attribute to *Ignore*) but fix the CID in the main branch (setting the *Action* attribute to *Fix Required*).

Figure 3.3.14. Example: Updating Triage Value, Multiple Triage Stores



To complete the scenario:

1. When setting up Coverity Connect for the first time, the administrator uses the **Add** button to create the triage store that is intended for streams that contain issues from the active code branch.

The example calls this triage store the Main store for convenience only. A company could give it any name that fits its naming conventions. Here, an active code branch is the branch for an upcoming release of the products.

2. The administrator makes sure that the streams are associated with the correct triage store:

   - When creating a stream for the active code base (here, release branch 2.0), the administrator associates the stream with Main triage store.

     For information about creating streams, see Section 3.3.1.2.2, “Setting up streams”.

   - When putting a branch of the code base into maintenance, the administrator creates a branch of the Main triage store, then associates the streams that contain issues for the maintenance release (here, `rel_1.0_p1` and `rel_1.0_p2`) with this copy (here, named the Maintenance store for convenience).

     This action creates a triage store that contains the current triage history of the Main triage store. The administrator associates the `*_1.0_*` streams with the Maintenance triage store so that users will be able use the *Advanced Triage* feature to update CIDs in the Main triage store without triaging CIDs in the Maintenance triage store (and vice versa).

     Alternative to using separate triage stores

     To avoid using separate triage stores for the streams shown in the example, you could use a single triage store along with a *Fix Target* attribute that identifies the releases (for example, `rel_1.0` or `rel_2.0`) in which a given CID could be fixed. In this way, a user who triages CID 123 could set the *Fix Target* attribute to `rel_2.0` and set the *Action* attribute to *Fix Required*. Though the CID would be marked as *Fix Required* for both branches, the *Fix Target* attribute would tell developers to fix the CID only in the active (`rel_2.0`) code branch, not in the maintenance branch. Such a scenario might be required to minimize the testing of the maintenance branch that is required prior to releasing a hotfix or patch.

     With this configuration, users no longer need to use advanced triage to triage the *Action* attribute of the CID differently in two triage stores. For more about issue attributes, see Section 3.3.2, “Configuring triage attributes”.

3. To provide or restrict access to the Maintenance streams, the administrator can use the *Roles* menu for the Maintenance store to assign any necessary roles to users or groups.

   If a user or group requires a particular role, the administrator can use the **Edit** button to select that role. For example, in some cases, a company might not want users to triage CIDs in the Maintenance store. In such a case, the administrator could assign a role that gives the a developer group permission to view (*View issues*) but not triage (or see the triage history of) the CIDs in the maintenance streams, or that prevents the group from viewing or triaging instances of CIDs that are in the maintenance streams. For details about roles and permissions, see Section 3.2.3, “Roles and role based access control”.

4. Click **Done** to save your changes and exit.

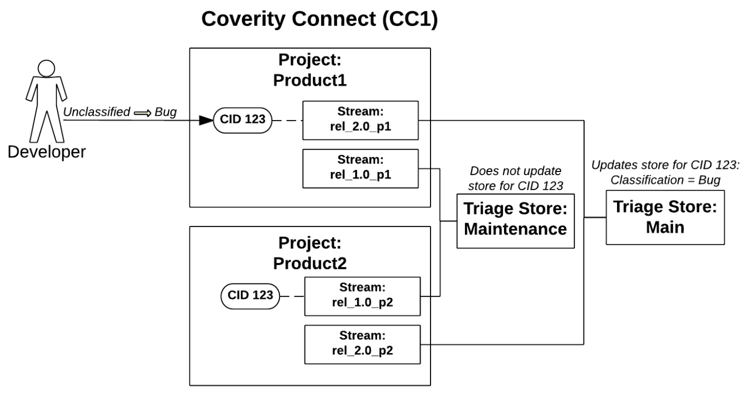
Note

In Figure 3.3.14, “Example: Updating Triage Value, Multiple Triage Stores”, streams for the different code branches are stored in the same project so that users can take advantage of advanced triage. However, if there is no need for advanced triage, the set of streams for each release could be in separate project. Assuming the same triage store associations, such a configuration would keep the triage data for each set of streams separate. However, triaging instances of a CID that occur in both projects would require triaging the CID twice, once in the 2.0 project, another time in the 1.0 project.

3.3.4.3. Scenario: Separating triage data in product-based projects

A company might keep the streams for each product in a separate project. Figure 3.3.15, “Example: Updating Triage Value, Multiple Stores, Product-based Projects” illustrates such a case. Each stream in a project is for issues from a separate branch (here, rel_1.0_* and rel_2.0_* streams) of the code base for a given product (here, *_p1 or *_p2). Streams for the active branch of the code base (here, rel_2.0_* streams) for all products share the Main triage store, while streams for the maintenance branch (here, rel_1.0_* streams) share the Maintenance triage store. As in Section 3.3.4.2, “Scenario: Separating triage data by code branch”, when triaging a CID in a project, users (with the appropriate permissions) can triage the CID in all triage stores (the default) or only in a subset of them by selecting triage stores. (See ???TITLE??? for more information.)

Figure 3.3.15. Example: Updating Triage Value, Multiple Stores, Product-based Projects



To complete the scenario:

1. Use the procedure in Section 3.3.4.2, “Scenario: Separating triage data by code branch” as a guide.

   The only difference to keep in mind is that streams for each release branch (rather than for each product) are in separate projects. Nevertheless, both scenarios use triage stores in the same way.

2. Click **Done** to save your changes and exit.

3.3.4.4. Configuring triage stores after upgrading from version 6.0 or 6.0.1

All triage store configurations from version 6.0 or 6.0.1 are retained after the upgrade. However, the *Roles* menu is new in version 2020.12. After upgrading to version 2020.12 from version 6.0 or 6.0.1, you can assign roles manually, as needed.

3.3.4.5. Exported triage store JSON elements

The following table lists and describes the elements contained in the JSON file that is produced when you use the **Export** button in the Triage Store menu.

Table 3.3.5. Exported triage store JSON elements

Element	Description
`<Name>`	The root element of the JSON file is a string containing the name of the triage store. The triage store may contain multiple defect objects, which each consist of the following attributes: `checker`, `cid`, `dateOriginated`, `detectedBy`, `domain`, `mergeKey`, `preventVersionExternal`, `preventVersionInternal`, and `triageStates`.
`checker`	The name of the checker that found the defect.
`cid`	The CID (Coverity ID) of the defect.
`dateOriginated`	The date and time that the defect was originally committed to Coverity Connect.
`detectedBy`	The process that was responsible for reporting the defect.
`domain`	The defect's programming language.
`mergeKey`	A unique identifier that maps a CID across multiple projects or Coverity Connect instances.
`preventVersionExternal`	The version of the Coverity Analysis tools that found the defect.
`preventVersionInternal`	The internal identifier for the Coverity Analysis version.
`triageStates`	A group of name/value pairs for the defect's triage attributes. `triageStates` contains the following child attributes: `action`, `classification`, `comments`, `customTriage`, `dateCreated`, `dateEnded`, `fixTarget`, `legacy`, `severity`, `userCreated`, and `userCreatedLdapServerName`.
`action`	Triage attribute used to specify the action to take with regard to the defect.
`classification`	Triage attribute that identifies the classification of the defect.
`comments`	User comments (if any) included by a user during triage of the defect.
`customTriage`	The `customTriage` object contains two child objects, `picklistAttributes` and `stringAttributes`, which contain

Element	Description
	any custom triage attributes and values created by the user or administrator.
`dateCreated`	Date and time when the current triage attribute values were stored.
`dateEnded`	Date and time when the current triage attribute values were updated or replaced.
`fixTarget`	Triage attribute used to set the release in which to fix an issue.
`legacy`	Triage attribute used to indicate whether a defect is a legacy issue or not.
`severity`	Triage attribute that identifies the severity of the defect.
`userCreated`	The user name that created the current defect triage state.
`userCreatedLdapServerName`	The LDAP server associated with the `userCreated` user name.
`Export Summary`	The `Export Summary` object contains information on the success or failure of the export process. It contains the following child objects: `failCount`, `successCount`, and `totalProcessed`.
`failCount`	The number of defect objects within the triage store that failed to export properly.
`successCount`	The number of defect objects within the triage store that exported properly.
`totalProcessed`	The total number of defect objects within the triage store.

3.3.5. Configuring legacy issues

A legacy issue is any issue that existed in a company's code base prior to adopting Coverity Analysis, or one that was undetected prior to completing a Coverity Analysis upgrade. Differentiating legacy issues from others returned by Coverity Analysis can be helpful for prioritizing newly introduced issues over those which were part of the company's backlog.

An issue's legacy status is defined by it's `legacy` attribute, which is set to `false` by default. To set the legacy status of a set of issues to `true`, use the `cov-manage-im` command in update mode, with `--set legacy:True` as an argument. See `cov-manage-im` in the *Coverity 2020.12 Command Reference* for more details on legacy issues and the `cov-manage-im` command.

3.3.5.1. Setting all new issues to legacy status

The most common scenario for using the legacy attribute is identifying all issues in a code base which existed before the adoption of Coverity Analysis. A company may want to implement a policy where all code check-ins have been checked by Coverity Analysis and are free of new issues. To do so, all existing issues in a given stream must be highlighted as legacy by running the following command after committing your first analysis to Coverity Connect:

```
> cov-manage-im --mode defects --stream <streamName> --update --set legacy:True
```

This will set the legacy status of all previously existing issues to true.

3.3.5.2. Creating legacy issues after upgrading Coverity Analysis

When a company upgrades to a newer version of Coverity Analysis, improvements in Coverity checkers may reveal additional issues that existed in the company's code base prior to upgrading. To mark these newly revealed issues as legacy issues, complete the following steps:

1. Run an analysis on your code base using the original Coverity Analysis version, and commit the results.
2. Complete the upgrade to the newer Coverity Analysis version.
3. Run the analysis again, and commit the results. Any issues introduced in this step are the cause of improvements in the Coverity checkers.
4. Use the following command to mark the issues found after upgrade as legacy issues:

   ```
   > cov-manage-im --mode defects --stream <streamName> --update --newest --set
    legacy:True
   ```

   The `--newest` option makes sure that only the issues introduced in the most recent snapshot (after the upgrade) are marked as legacy issues.

   Note

   An optional `[snapshotId]` parameter can be added after the `--newest` option to compare the newest snapshot directly against the specified older snapshot. In the following example, all issues that are present in the latest snapshot but not present in snapshot 10001 will be marked as Legacy issues.

   ```
   > cov-manage-im --mode defects --stream <streamname> --update --newest 10001
    --set legacy:True
   ```

3.3.5.3. Working with legacy issues in Coverity Connect

Coverity Connect provides the ability to assign the legacy attribute to individual issues, as you would triage any other attribute. The legacy attribute doesn't appear in the triage panel by default, but can be added by completing these steps:

1. Navigate to the Configuration → Attributes menu.
2. Click on *Legacy* in the left pane.
3. In *Attribute Details*, click **Edit**.
4. For *Display*, select **On (Shown in triage panel)**.
5. Click **OK** and **Done** to exit from the Attributes menu.

Now you can select an issue's legacy status directly in Coverity Connect's triage panel. You can also update the legacy status of multiple issues at once by highlighting all the issues to be updated and selecting the new legacy value in the triage panel.

Chapter 3.4. Configuring automatic owner assignment

Table of Contents

3.4.1. Overview of the automatic owner assignment process 226
3.4.2. Setting global SCM rules and the SCM user map 227
3.4.3. Setting stream-level rules 233

You can configure Coverity Connect to automatically assign an owner to an unassigned issue when that issue is committed to a new snapshot. After the username is determined, it is displayed in the *Owner* field in the triage pane.

This feature alleviates the cumbersome process of inspecting issues and manually assigning each one to a developer who has the expertise to investigate and fix it.

Consult the subsequent implementation instructions to configure owner assignment.

3.4.1. Overview of the automatic owner assignment process

The following steps provide an overview of the automatic owner assignment configuration process, as well as pointers to specific configuration/implementation details:

1. Determine the method in which you want Coverity Connect to automatically assign owners:

   - **By retrieving SCM (Source Code Management) history to determine the assigned owner.**

     The SCM history consists of when, where, and by whom the code was changed and the assignment rules derive the owner of an issue based on the SCM history. Determining the owner using SCM data is accomplished by a combination of configuring Coverity Connect UI properties and Coverity Analysis command utilities. Coverity Connect supports the same SCM tools that are supported by the Test Advisor product. See Coverity Test Advisor SCM and Platform Support for a list of supported SCM tools.

     There are two types of users that are referred to in this section:

     - SCM user - A user identity that modified or checked the code into the SCM system.
     - Coverity Connect user - A user identity that exists in the Coverity Connect database.

   - **By defining the default owner in a component.**

     This method uses the functionality associated with Coverity Connect components. If you choose this method, skip to Step 4.

   - **Alternatively, you can choose not to enable automatic owner assignment at all.**

   This allows organizations with multiple code bases to gradually transition to automatic owner assignment, one code base at a time and according to a schedule convenient to each development team.

2. If you are planning to use SCM data for owner assignment, define the global SCM derivations rule under the system configurations. See Section 3.4.2, “Setting global SCM rules and the SCM user map”.

3. Configure the SCM to Coverity Connect user map file. See Section 3.4.2.2, “Configuring the SCM to Coverity Connect user map”.

4. At the stream level, define the owner assignment rule. See Section 3.4.3, “Setting stream-level rules”.

   If you configure owner assignment with SCM data, continue to the next step in this workflow.

   If you are using the component mechanism, or choose not to enable owner assignment, the remaining steps are not applicable.

5. Update your Coverity Analysis scripts(s)/commit process to include the SCM options to `cov-commit-defects`.

   If you are not already using the `cov-import-scm` command to retrieve SCM data, then you will need to add the `--scm` option (and other optional SCM-related options) for `cov-commit-defects`. For more information, see `cov-commit-defects`.

6. Optionally test the results of different derivation rules without having to commit them to Coverity Connect by using the `cov-blame` command. For more information, see the `cov-blame`.

3.4.2. Setting global SCM rules and the SCM user map

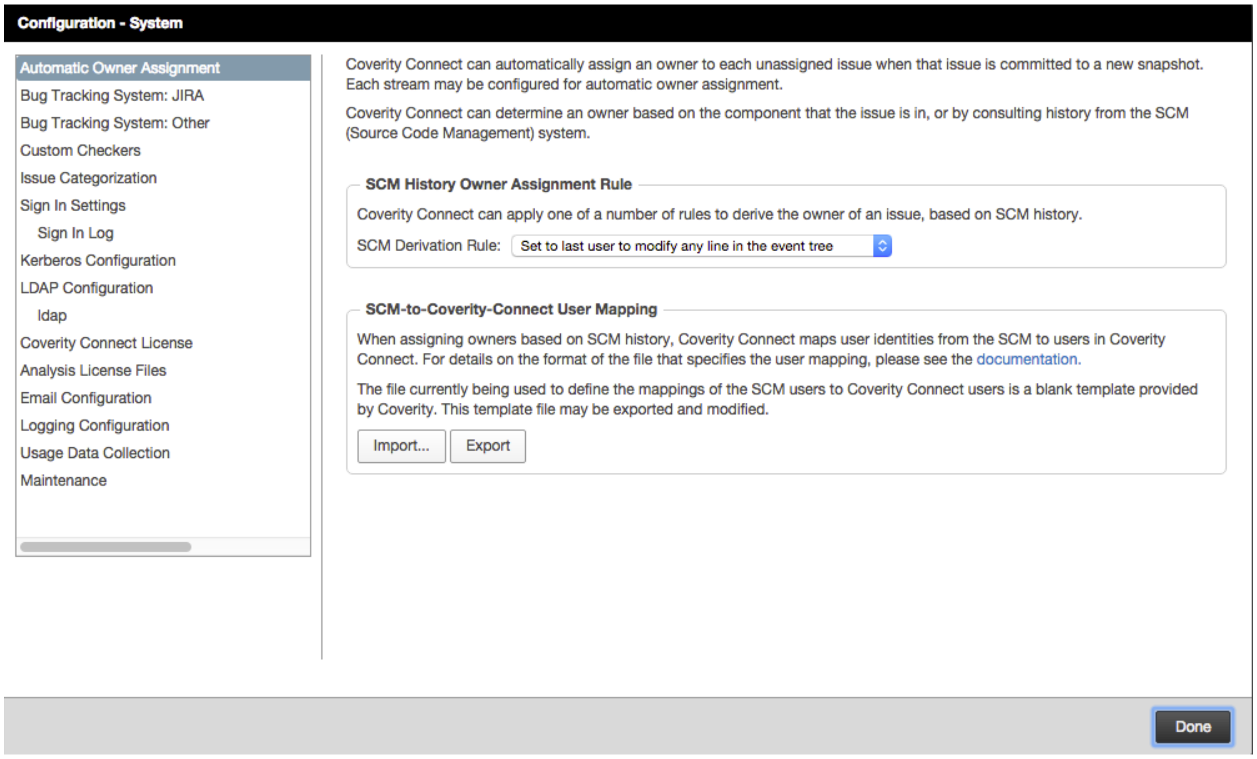
If you have made the decision to enable Coverity tools to assign owners based on SCM data, the first step is to configure the following globally-defined items:

1. Go to Configuration → System → Automatic Owner Assignment.

2. Choose a global SCM derivation rule.

3. Define an SCM to Coverity Connect user map file.

Figure 3.4.1. SCM system settings for automatic owner assignment



Coverity Connect can automatically assign an owner to each unassigned issue when that issue is committed to a new snapshot. Each stream may be configured for automatic owner assignment.

Coverity Connect can determine an owner based on the component that the issue is in, or by consulting history from the SCM (Source Code Management) system.

SCM History Owner Assignment Rule

Coverity Connect can apply one of a number of rules to derive the owner of an issue, based on SCM history.

SCM Derivation Rule: Set to last user to modify any line in the event tree

SCM-to-Coverity-Connect User Mapping

When assigning owners based on SCM history, Coverity Connect maps user identities from the SCM to users in Coverity Connect. For details on the format of the file that specifies the user mapping, please see the documentation.

The file currently being used to define the mappings of the SCM users to Coverity Connect users is a blank template provided by Coverity. This template file may be exported and modified.

Note

In a Coverity Connect clustered environment, all of these settings are synchronized, so you can only set them on the coordinator and not on the subscribers. For more information, see Section 3.5.1, “Synchronizing multiple Coverity Connect instances”.

3.4.2.1. SCM derivation rules

Coverity Connect can apply one of a number of rules to derive the owner of an issue based on SCM history. The rule that you choose is applied to all streams in the project that are configured to accept SCM data for owner assignment (see Section 3.4.3, “Setting stream-level rules”).

It is not required that you configure this property, as there is the default setting, *Set to last user to modify any line in the event tree*. However, you might find that a different rule is more appropriate for assigning owners within your system.

The following table lists the derivation rules available for configuration in Coverity Connect and shows the corresponding rule name used in the `cov-blame --owner-assignment-rules` command line option. For descriptions of the rules, see the `--owner-assignments-rules` option for `cov-blame`.

Table 3.4.1. Automatic ownership assignment rules

SCM rule	cov-blame rule
Set to last user to modify file of main event	`file`
Set to last user to modify line of main event	`line`
Set to last user to modify the function of main event	`function`
Set to last user to modify top of the event tree	`top_events`
Set to last user to modify any line in the event tree	`all_events`
Set to last user to modify any function in the event tree	`all_functions`
Set to last user to modify any file in the event tree	`all_files`

3.4.2.2. Configuring the SCM to Coverity Connect user map

When owners are assigned based on SCM history, Coverity Connect maps user identities from the SCM data to users in Coverity Connect through an imported file.

The default user map file is in JSON format. It is set globally, so it is shared by all streams in Coverity Connect that are configured to accept owners derived from SCM data.

Coverity Connect provides a default user map file that you can access by clicking the **Export** and then saving to your preferred location. After you finish editing the file, use the **Import** button to import it for use by Coverity Connect.

The file template is as follows:

```
{
  "map" : [ {
    "scmUsername" : "(.*)",
    "cimUsername" : "$1"
  } ]
  }
```

The JSON elements for user mapping are:

`"scmUsername"`

Represents the SCM username expressed as a regular expression using full string matching. It uses Java's built-in regular expression syntax. For more information about the regular expression syntax, see the Javadoc at http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html.

The following table describes the username mapping format for supported SCM tools:

Table 3.4.2. SCM username mapping format

SCM tool	username format description	Example regex for Coverity Connect mapping
Accurev	The "user" of the transaction.	`(.*)`

SCM tool	username format description	Example regex for Coverity Connect mapping
Clearcase	The "login name" and "group" of the user for the revision, in the form `<user>.<group>`.	`(.*)\\.(.*)`
CVS	The "author" of the revision.	`(.*)`
Git	The "author-mail" (`user.email`) of the revision, in the form `<user>@<domain>`.	`(.*)@(.*)`
Hg (Mercurial)	The short representation of the "author" of the changeset.	`(.*)`
Perforce	The "user" of the revision.	`(.*)`
Plastic	The "owner" of the changeset.	`(.*)`
SVN	The "username" of the revision.	`(.*)`
TFS	The "author" of the changeset.	`(.*)`

Note

For all SCM tools except for Clearcase and Git, you might have an informal policy for the value of the relevant field that Coverity Connect chooses for the username. This might allow for a more specific regex to be used in the Coverity Connect user map.

`"cimUsername"`

Represents the user in the Coverity Connect database to which the SCM user is transformed. In the example below, the value is represented as a reference to a captured subsequence. For information about capturing, see the Javadoc for Groups and Capturing.

This element can not be used in the same mapping as `"cimEmail"`.

Note

Values for the Coverity Connect *username* field are stored in lower case.

`"cimEmail"`

This optional element represents the email address of the user in the Coverity Connect database to which the SCM user is transformed. This is useful for cases where the `scmUsername` is the user's email address.

This element can not be used in the same mapping as `"cimUsername"`.

`"ldapServer"`

This element is optional and is only required if you have duplicate user names from LDAP servers or a combination of LDAP users and local users. You can also explicitly specify a local user with `"ldapServer" : "local"` in the case that a user exists on both the local Coverity Connect server and the LDAP server. When `"ldapServer"` is left blank, is null, or is absent, the entry will only match if there is exactly one Coverity Connect user with the expected username.

Mapping rules are processed in order until a match is found, or there are no more rules to process. If `"scmUsername"` matches the username of the SCM user, but the corresponding `"cimUsername"` (after variable expansion) is not a valid Coverity Connect user, the process continues to the next `"scmUsername" rule.`

Example 3.4.1. Example - SCM user mapping

In the following example, the mapping file attempts to match an SCM user in the format of <`first_name`>.<`last_name`> and transform it to a Coverity Connect user in the form of <`first_letter_first_name`><`last_name`>. Otherwise, the file attempts to match <`username`>@<`domain.com`> and transform it to <`username`> at an LDAP server <`domain.com`>.

```
{
  "map" : [ {
    "scmUsername" : "([a-z])[a-z]*\\.([a-z]+)",
    "cimUsername" : "$1$2"
  }, {
    "scmUsername" : "(.*)@(.*)",
    "cimUsername" : "$1",
    "ldapServer" : "$2"
  } ]
}
```

The following example demonstrates literal mappings as well (`jdoe -> john.doe`):

```
{
  "map" : [ {
    "scmUsername" : "([a-z])[a-z]*\\.([a-z]+)",
    "cimUsername" : "$1$2"
  }, {
    "scmUsername" : "(.*)@(.*)",
    "cimUsername" : "$1",
    "ldapServer" : "$2"
  }, {
    "scmUsername" : "jdoe",
    "cimUsername" : "john.doe"
  } ]
  }
```

The following example attempts to match an SCM <`username`> and transform it to the Coverity Connect user's <`email_address`>:

```
{
  "map" : [ {
    "scmUsername" : "(.*)",
    "cimEmail" : "$1"
  } ]
}
```

The following example shows how to map `DOMAIN\username` to `username`.

```

{
  "map" : [ {
     "scmUsername" : "(.*)\\\\(.*)",
     "cimUsername" : "$2"
   } ]
}
```

In the above example, the backslash character must be escaped twice. The backslash is escaped once to be properly interpreted by the regular expression parser: `DOMAIN\\username`. Next, because that expression is encoded in JSON, each backslash must be escaped again: `DOMAIN\\\\username`.

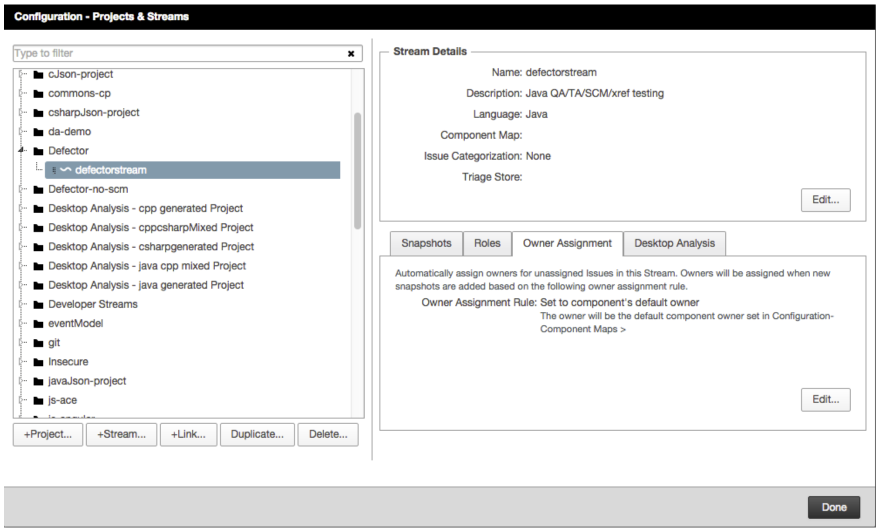
Note

All SCM to Coverity Connect user map debugging messages are registered in the `cim.log` file. You can view these debug messages by enabling the `Commit` option in Coverity Connect's logging configuration settings.

3.4.3. Setting stream-level rules

The *Owner Assignment* tab allows you to set the owner assignment rule that will compute automatic owner assignment for issues displayed in Coverity Connect.

Figure 3.4.2. Stream settings for automatic owner assignment



Stream Details

Name: defectorstream
Description: Java QA/TA/SCM/xref testing
Language: Java
Component Map:
Issue Categorization: None
Triage Store:

Automatically assign owners for unassigned Issues in this Stream. Owners will be assigned when new snapshots are added based on the following owner assignment rule.

Owner Assignment Rule: Set to component's default owner
The owner will be the default component owner set in Configuration-Component Maps >

1. Go to Configuration → Projects & Streams.

2. Expand a project to see the list of streams.

3. Select the stream for which you want to apply the rule.

4. Select the *Owner Assignment* tab.

5. Click **Edit** and choose one of the following rules:

   Set to component's default owner.

   The owner will be the default component owner for a component configuration. This option is the default and will not trigger Coverity Analysis to attempt to retrieve any SCM historical data. If this option is selected, but the component to which the stream belongs does not have the default owner assigned, the owner will be left as *Unassigned* for the issue.

   For more information, see Section 3.3.3, “Using components”.

Derived from SCM (Source Code Management)
: The owner will be based on the SCM Derivation Rule set in Table 3.4.1, “Automatic ownership assignment rules”.

None (leave issues unassigned)
: Unassigned Issues will not automatically be assigned an owner. The *Owner* field in Coverity Connect will be left as *Unassigned*.

This tab assigns owners for unassigned issues in the selected stream. Owners will be assigned when new snapshots are added based on the owner assignment rules.

Chapter 3.5. Configuring Coverity Connect enterprise clusters

Table of Contents

3.5.1. Synchronizing multiple Coverity Connect instances 235
3.5.2. Managing multiple database instances 243

3.5.1. Synchronizing multiple Coverity Connect instances

Coverity Connect Coordination allows you to deploy clusters of Coverity Connect instances on which centrally managed data is synchronized. Importantly, developer-set triage data can be updated automatically across the cluster. For Coverity Connect to synchronize data, it is necessary to set up one instance of Coverity Connect as the central Coordinator and to configure other Coverity Connect instances into a cluster of Subscribers with which the Coordinator can communicate.

Coordinator Responsibilities

The following functionality (available through the *Configuration* menu) is viewable but not configurable on the Subscribers so that the Coordinator can manage it centrally:

- *Configuration - Users & Groups*: All users in the cluster are set up and managed in the Coordinator. Note that the triage options available to developers include the *Owner* attribute, which is populated by a comprehensive list of all users in the cluster. All users that are managed by the Coordinator belong to at least one group, so all groups in the cluster are also created and managed in the Coordinator. See Section 3.2.1, “Managing users” and Section 3.2.2, “Managing groups” for details.

  For LDAP settings, also use System - Configuration on the Coordinator. Note that Subscribers cannot modify LDAP configurations in the System configuration. You must contact the Coordinator to make changes.

- *Configuration - Roles*: RBAC permissions and roles are configured on the Coordinator. However, roles for projects and streams can be applied on each Subscriber through the *Projects & Streams* menu item. See Section 3.2.3, “Roles and role based access control” for details.

- *Configuration - Triage Stores*: One or more Triage Stores in the Coordinator support the developer-set triage data that gets synchronized across the cluster. When you set up a stream in a Subscriber, you select from a list of these Triage Stores. See Section 3.3.4, “Managing triage stores” for details.

- *Configuration - Component Maps*: All component maps and components are managed through the Coordinator. See Section 3.3.3, “Using components” for details.

- *Configuration - Attributes*: The custom attributes and attribute values that are available to each Coverity Connect instance in the cluster are set up and managed in the Coordinator. When a developer who is triaging an issue changes an attribute value for an issue, that change gets propagated through the cluster. See Section 3.3.2, “Configuring triage attributes” for details.

- *Configuration - System* features: The Coordinator supports *Issue Categorization* and *LDAP Configuration* (if you are using LDAP). See Section 3.1.1.6, “Configuring custom issue categories” and Section 3.1.1.9, “Integrating with LDAP servers” for details.
- *Configuration - System - Automatic Owner Assignment*: The Coordinator controls *Automatic Owner Assignment*. See Chapter 3.4, *Configuring automatic owner assignment*.

Locally Configured Coordinator and Subscriber Features

- *System - Projects & Streams*: You set up and manage separate sets of projects and streams locally. Note that sharing projects and streams on both the Coordinator and a Subscriber is not a supported use case. See Section 3.3.1, “Working with projects and streams” for details.

  Each stream is associated with a Triage Store that is set up on the Coordinator. You create this association through the *Configuration - Projects & Streams* on each Coverity Connect instance. For guidance, see Section 3.3.1.3, “Editing projects and streams”.

If possible, the coordinator instance should act solely as coordinator, and have no projects, streams, or snapshots configured. This will ensure that no individual project or stream information is lost in the event of failure.

Local configuration outside the scope of the Coordinator/Subscriber model:

Local *System - Configuration* features: *Except when setting up Issue Categorization* and *LDAP Configuration*, you perform system configuration locally. See Chapter 3.1, *Performing system configurations*.

Configuration - Hierarchies: This page is accessible only if your license covers Coverity Policy Manager. See Part 5, “Coverity Policy Manager Administration”.

Version requirements

The Subscriber Coverity Connect servers must be the same or previous release as the Coordinator server. For example, if the Coordinator is version 8.0, the Subscriber servers may be version 8.0 or 7.7.

To set up Coverity Connect Coordination:

1. Take time to understand the synchronization process.

   Review Section 3.5.1.1, “Synchronizing data across the cluster”

2. Set up shared features.

   For guidance, see Coordinator Responsibilities.

3. Configure synchronization properties.

   For guidance, see Section 3.5.1.2, “Configuring data synchronization across the cluster”.

Note

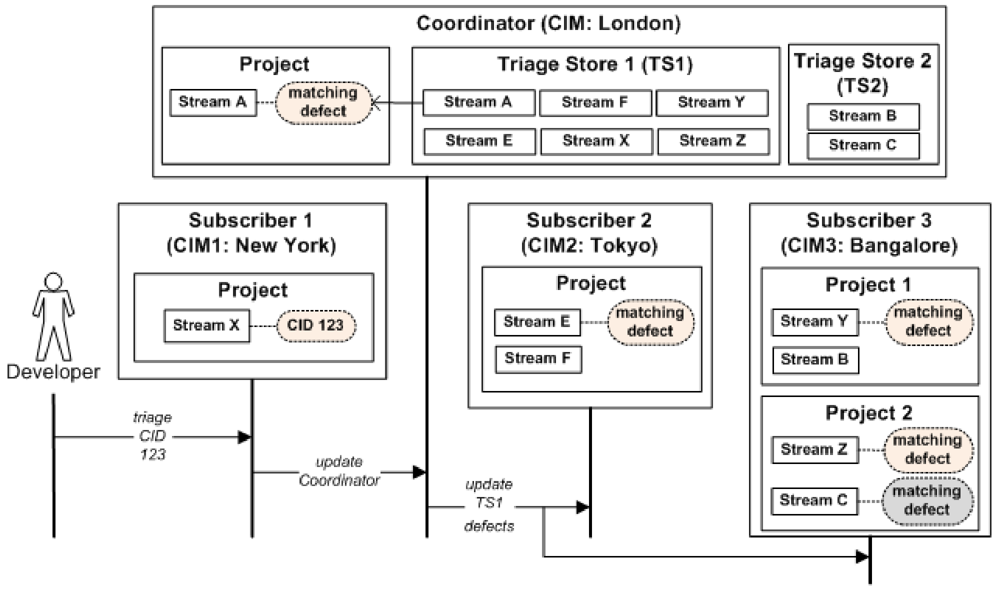
If the Coverity Connect coordinator goes down, it will synchronize data on the subscribers once it becomes available again. During the time the coordinator is down, new issues that are committed to a subscriber are marked as *Pending* in the subscriber UI. Such issues will receive a CID after the coordinator becomes available again.

If one of the Coverity Connect subscribers in the cluster becomes unavailable (for example, due to a network failure), the coordinator will update that instance once it becomes available again. The coordinator will also receive the latest configuration settings and triage values for the instance that was down once that instance becomes available again. After receiving the updated data, the coordinator will propagate it to the other subscribers, as needed.

3.5.1.1. Synchronizing data across the cluster

When a developer triages an issue through the Coordinator or through a Subscriber, the update propagates by way of one or more Triage Stores in the Coordinator to other members of the cluster. In this way, the Coordinator is responsible for synchronizing triage data updates across Coverity Connect Subscribers.

Figure 3.5.1. Example: Coverity Connect Coordination



As Figure 3.5.1, “Example: Coverity Connect Coordination” shows, a developer triages CID 123 in Stream X of CIM1 (a Subscriber), a stream that is associated with Triage Store 1 (TS1) on the Coordinator. After receiving notification of the change in triage data from the Subscriber, the Coordinator updates the other Subscribers (CIM2 and CIM3). The new triage data is synchronized for each matching issue found in TS1 streams.

Note that the triage data of the matching issue in Stream C of Project 2 (CIM 3) is not updated because Stream C belongs to TS2, not TS1. For this reason, the Stream C issue appears in a different color than the others in the figure.

Table 3.5.1, “Triage Data Updates” identifies the updates to the triage data of CID 123 that are shown in Figure 3.5.1, “Example: Coverity Connect Coordination”.

Table 3.5.1. Triage Data Updates

Coverity Connect	Location	Project	Matching issues	Stream	Triage Store 1	Data
Subscriber CIM1	New York	—	✓	X	✓	updated
Coordinator CIM	London	—	✓	A	✓	updated
Subscriber CIM2	Tokyo	—	✓	E	✓	updated
			no	F		not updated
Subscriber CIM3	Bangalore	1	✓	Y	✓	updated
			no	B	no	not updated
		2	✓	Z	✓	updated
				C	no	not updated

Note

It is possible that the CIDs will not display properly on the subscriber if the coordinator is unavailable. The CIDs will display correctly when the coordinator becomes available again.

3.5.1.2. Configuring data synchronization across the cluster

Setting up data synchronization across the cluster requires identifying the Coordinator and Subscribers and setting up SSL certificates and TrustStores.

3.5.1.2.1. Identifying the Coordinator and Subscribers

This section describes how to edit the cim.properties file to identify the Coordinator and Subscribers in a cluster. It also describes how to remove a Subscriber from the cluster.

To identify the Coordinator and Subscribers:

This configuration takes place through settings in a configuration file (`cim.properties`) on each Coverity Connect instance. Each Subscriber instance must specify the address of the Coordinator in this file.

Important Subscriber Requirement

If you are setting up a new Subscriber instance, the instance should not contain any pre-existing issue data or any settings or records that will be managed by the Coordinator. As Coordinator

Responsibilities explains, the Coordinator handles all centrally managed configurations, so they should not be set up on any of the Subscribers.

1. Configure one instance of Coverity Connect as the Coordinator.

   You need to add the following properties to `<install_dir>/config/cim.properties` on the Coordinator:

   - `remoteconfig.mode`: Set the remote configuration mode to `coordinator`.

   Coordinator configuration example:

   ```
   remoteconfig.mode=coordinator
   ```

   - Set up SSL communications.

2. Configure one or more *clean* instances of Coverity Connect as Subscribers.

   You need to add the following properties to `<install_dir>/config/cim.properties` on each Subscriber:

   - `remoteconfig.mode`: Set the remote configuration mode to `subscriber`.
   - `remoteconfig.coordinator`: Provide the IP address or fully qualified domain name (not an alias) followed by the Commit port number (the Coverity Connect default is `9090`) for the Coordinator on each Subscriber in the cluster.

     Note

     Even if you use the Coverity Connect HTTPS port for commiting defect data, you must continue to use the Commit port for Coordinator to Subscriber communication.

   Subscriber configuration example:

   ```
   remoteconfig.mode=subscriber
   remoteconfig.coordinator=cim.london.ex.com:9090
   ```

   - Set up SSL communications.

Since the `remoteconfig` SSL negotiation uses the operating system's host name resolution system to validate SSL certificates from other hosts in the cluster, that infrastructure must be capable of resolving the host names of the peers in the cluster.

Normally this is not an issue since Coverity Connect servers are assumed to be installed in an environment with a working host name resolution system. However, in some environments this may not be the case. In such environments, you may need to enable host name resolution on your server or edit `/etc/hosts`.

Removing a subscriber from the cluster

In order to remove a Coverity Connect instance from the cluster, navigate to `<install_dir>/config/cim.properties` and set the `remoteconfig.mode` property to `none`.

```
remoteconfig.mode=none
```

After your properties file is updated, restart Coverity Connect.

Note

Once a subscriber changes to a non-subscriber, it cannot be changed back. So it is recommended that you back up all subscriber data before configuration.

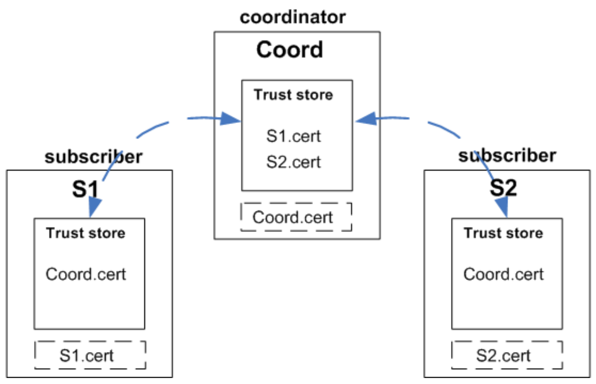
3.5.1.2.2. Setting up SSL certificates and TrustStores

This section describes how to set up SSL certificates and TrustStores to configure a cluster for secured communication.

Coverity Connect uses SSL to authenticate and encrypt communication between the Coordinator and Subscribers. Unlike in the HTTPS protocol, authentication is mutual, meaning that the Subscriber authenticates the Coordinator, and the Coordinator authenticates to the Subscriber. This authentication can use self-signed SSL certificates generated by each Coverity Connect instance, however you can also implement SSL certificates that are applicable to your system. This section only describes how to set up SSL with Coverity-generated certs.

To establish the trusted relationship between the Coordinator and the Subscribers, you must exchange the SSL certificates between them. The procedure in this section describes how to do so.

Figure 3.5.2. Coverity Connect certificate exchange between Coordinator and Subscribers



Initial installation setup

The first time you start Coverity Connect it automatically creates these files in <install_dir_cp>/config:

- **keystore.jks--**The Java KeyStore file is a repository of a private key and certificate. The certificate is essentially the “lock” that requires the key to establish SSL communication. To set up SSL between two hosts, the hosts must first exchange their certificates.

- **truststore.jks--**The Java TrustStore file is used to store certificates of trusted hosts. Those trusted hosts can authenticate using their private key. For a Subscriber, the TrustStore will store one certificate: the Coordinator's. For a Coordinator, the TrustStore will store one certificate for each Subscriber. Deleting a TrustStore invalidates all trust relationships that it had recorded.

- **<truststore.jks--**The certificate file is a copy of the certificate in the KeyStore. The file is created to make it easy to transport the certificate to other hosts in the cluster.

After these files are created on all of the instances in a cluster, you can exchange certificates between the Coordinator and Subscribers as described later in this section.

Post-upgrade manual setup

The trust relationships define which Coverity Connect instances can participate in the cluster. If you make a copy of an instance, using an Intermachine or Backup-and-restore upgrade, both the copy and the original will be able to participate in the cluster. Having that duplication can damage the cluster. You have to prevent that from happening by re-forming the trust relationships to exclude the original instance.

A Coverity Connect *Intermachine upgrade* copies non-database state to a new host. Therefore, after you perform an Intermachine upgrade on an instance, the old KeyStore, certificate, and TrustStore will exist on the old and new instance. This is undesirable and requires deleting and re-creating the KeyStore, certificate, and TrustStore on the new host. That change also necessitates updates to the TrustStores of other instances in the cluster. The same can be true when you perform a *Backup-and-restore upgrade* to a new location on the same host, except that the KeyStore, certificate, and TrustStore are mirrored in a new directory location on the same host.

Given the duplication of KeyStores, certificates, and TrustStores that result from Intermachine upgrades and Backup-and-restore upgrades to new directory locations, it's critical to delete and re-create KeyStores, certificates, and TrustStores as necessary after upgrading Coverity Connect.

You can force a Coverity Connect instance to re-generate its certificate by removing the current certificate file and `keystore.jks` from the `config/` directory and then re-starting Coverity Connect. Also, when re-starting Coverity Connect, if truststore.jks is absent Coverity Connect will create a new, empty `truststore.jks`.

Once you have established the desired KeyStores, certificates, and TrustStores on the upgraded instances, you can proceed with exchanging certificates between the Coordinator and Subscribers as described in the following procedure.

Note

You can add the `remoteconfig.localHostName` property to `cim.properties` to define the hostname returned by the system when Coverity Connect generates certificates for remote configurations. Specifying this property in the configuration file overrides the hostname value that Coverity Connect automatically searches for when the certificate is generated. The automatic mechanism sometimes can not resolve hostnames, so setting this property allows you make the hostname explicit.

To exchange SSL certificates between a Coordinator and a Subscriber:

1. On the Coordinator go to the `<install_dir_cp>/config` directory and copy the certificate to the Subscriber. For example:

```
scp Coord.cert subscriber1.domain.com:S1/cov-platform/cim/config
```

2. On the Subscriber go to the `<install_dir_cp>`/config directory and copy the certificate to the Coordinator. For example:

   ```
   scp S1.cert Coordinator.domain.com:/Coord/cov-platform/cim/config
   ```

3. On the Coordinator, import the Subscriber's certificate into the Coordinator's trust store. For example:

   ```
   cov-import-cert S1.cert truststore.jks
   ```

4. On the Subscriber, import the Coordinator's certificate into the Subscriber's trust store. For example:

   ```
   cov-import-cert Coord.cert truststore.jks
   ```

5. Restart both the Coordinator and Subscriber instances of Coverity Connect.

6. To ensure that the Coordinator and Subscriber are correctly communicating, examine the `<install_dir_cp>/logs/cim.log` file on both the Coordinator and the Subscriber.

   If there are no exceptions in the log file, encrypted communication should be functioning as expected.

7. Apply your updates by restarting all Coverity Connect instances on which you updated `cim.properties`.

   On Linux:

   ```
   > cd <install_dir>/bin
   > ./cov-stop-im
   > ./cov-start-im
   ```

   On Windows:

   - Go to `<install_dir>\bin`, and double-click `cov-stop-im`.
   - After Coverity Connect stops, restart it by double-clicking `cov-start-im`.

Repeat these steps for each new Subscriber on your system.

3.5.1.2.3. Synchronizing Coverity Policy Manager data across the cluster

All Policy Manager data is synchronized across the cluster once a day, for each subscriber in the cluster (function counts are excluded). The coordinator aggregates the data collected from all of its subscribers, but the subscribers do not receive any Policy Manager data back from the coordinator. This process is detailed in the following steps:

1. The daily trend report update runs on Subscriber 1. This starts sometime between midnight and 1:00 am local time for the subscriber, and may take several hours to finish.

2. Once the update is complete on Subscriber 1, the Coordinator (which checks periodically for any newly available data from its subscribers) requests the new data, and Subscriber 1 sends it.
3. Extract Transform Load (ETL) runs on the Coordinator and picks up the new data from Subscriber 1. The Coordinator runs ETL periodically throughout the day, waiting one hour after the end of each update to start the next one. So if the ETL run takes 30 minutes, the Coordinator will be updated approximately every 90 minutes.

   Note

   The Policy Manager needs to run the ETL process whether a cluster is in use, or not. See Chapter 5.3, *Scheduling the Extract Transform Load (ETL) Process*.

Note

Hierarchies are not synchronized between subscribers and coordinators. Instead, when creating or editing a hierarchy on the coordinator, you can create individual nodes that contain projects from subscriber instances of Coverity Connect. See Section 5.1.2, “Configuring a Hierarchy” for more information.

3.5.2. Managing multiple database instances

When backing up and restoring databases in a clustered Coverity Connect deployment, use the workflows in this section to ensure Coverity Connect data integrity.

3.5.2.1. Workflow for backing up databases in a cluster

Important notes:

- A Coordinator database backup must be made from the same or later point in time than from when the Subscriber database backup is made.
- If possible, put Subscriber and Coordinator database instances into maintenance mode (`cov-im-ctl maintenance`) prior to using this workflow. Do this when the cluster is not synchronizing data and when commits are not runnning. If you can put the database instances into maintenance mode, start with the Subscriber databases and put the Coordinator database in maintenance mode last.
- If your cluster’s workload does not provide a convenient time when data is not synchronizing and commits are not running, schedule the backups to a time with the least activity and make sure the Subscriber database backups complete before backing up the Coordinator database.

To back up the embedded PostgreSQL database for a Subscriber instance or a Coordinator instance in a clustered deployment:

1. Read the important notes above.
2. Back up the database. See Section 3.1.2.3, “Backing up the database”
3. Regularly test the integrity of the backups by restoring them as described in Section 3.1.2.3, “Backing up the database”

3.5.2.2. Workflow for restoring databases in a cluster

To restore the embedded PostgreSQL database for a Subscriber instance or a Coordinator instance in a clustered deployment:

1. On each *Subscriber* instance, put the embedded database into maintenance mode:

   ```
   > cov-im-ctl maintenance
   ```

2. After all Subscriber databases are in maintenance mode, put the Coordinator database into maintenance mode:

   ```
   > cov-im-ctl maintenance
   ```

3. On each Subscriber instance, run the following commands to restore the database and immediately put it back into maintenance mode:

   ```

   > cov-im-ctl maintenance
   > cov-admin-db restore <archive_file>
   > cov-im-ctl maintenance
   ```

4. On the Coordinator instance, run the following commands to restore the database.

   ```

   > cov-im-ctl maintenance
   > cov-admin-db restore <archive_file>
   ```

5. After running the `cov-admin-db restore` command on the Coordinator instance, wait until the Coordinator instance starts.

6. Start each Subscriber instance:

   Note

   Wait until each instance has started before starting the next instance.

   ```
   > cov-im-ctl start
   ```

7. After all Subscriber instances have started, allow the cluster synchronization process to complete before carrying out new commits, triage, or other updates on the Subscriber instances.

Chapter 3.6. Configuring a Server to Support Code Sight

Table of Contents

3.6.1. Server administration for the Synopsys Code Sight Plug-in 245
3.6.2. Configuring a Coverity Connect server to support Code Sight clients 245

3.6.1. Server administration for the Synopsys Code Sight Plug-in

Coverity Connect requires specific configuration in order to provide analysis summary data to users of the Code Sight plug-in. This chapter describes the administrative tasks required to enable proper communication with Code Sight.

3.6.2. Configuring a Coverity Connect server to support Code Sight clients

The Coverity Connect server needs to deploy both the installer for Coverity Analysis, and the customer's `license.dat` file.

- The server administer must ensure that both the installer for Coverity Analysis and the `license.dat` file are located in a directory named as follows: `<server-install-dir>/server/base/webapps/downloads`.

  These are the specific file names:

  - `license.dat`
  - `cov-analysis-win64-2018.12.exe`
  - `cov-analysis-linux64-2018.12.sh`
  - `cov-analysis-macosx-2018.12.sh`

  Note

  If you are using a different version of Coverity Analysis, use the file names for that specific version instead of the ones associated with version 2018.12.

Chapter 3.7. Server administration for Desktop Analysis

Table of Contents

3.7.1. Setting up streams 246
3.7.2. Creating analysis summaries 247
3.7.3. Summary expiration 247
3.7.4. Desktop Analysis usage tracking 248
3.7.5. Analysis license service 248

Coverity Connect requires specific configuration in order to provide analysis summary data to Desktop Analysis users. This chapter describes the administrative tasks required to enable proper communication with Desktop Analysis.

Note

The `Developer` role is required for using Desktop Analysis.

3.7.1. Setting up streams

Desktop Analysis relies on a "reference snapshot" to provide analysis summary data. Reference snapshots should be organized into one or more streams for user convenience. As a Coverity Connect administrator, you must choose that stream organization.

As a general principle, if you choose to use more streams, developers will be able to select a reference snapshot that is more similar to the code they are developing, and hence get more accurate analysis results. However, the cost of more streams is the necessity to set up and run more central analysis jobs, also requiring more storage space on the Coverity Connect server.

The following list describes some of the most common stream design options.

One stream (simplest)
: The simplest solution is to have only one stream enabled for Desktop Analysis. In this scenario, each Desktop Analysis user uses the same stream for analysis summaries.

Stream per branch (recommended)
: The recommended organization is to have one stream per branch of your code base that is under active development. For example, you may have a separate branch for version X, Y, and Z of your code. By setting up a stream for each branch, developers working on separate versions will get the most accurate summary data for their version of the code.

Stream per branch and platform (most accurate)
: By setting up one stream per platform, the analysis summary data will be specific not only to the version of your project, but also to its relative platform. While this configuration offers the most accurate summary information, it also requires the most maintenance. Unless your project contains significant differences between platforms, it is recommended that you use the "stream per branch" configuration.

Make sure that you select **Enable Desktop Analysis** from the *Desktop Analysis* tab for each stream that will used by Desktop Analysis developers. Any stream that is not meant for use with Desktop Analysis should leave the **Enable Desktop Analysis** box unchecked, as checking this box will cause more storage space to be used.

As a rough guide to space usage, if your code base has 10 million non-blank, non-comment lines of code, the first snapshot with analysis summary data will require 2 GB of storage space. Each additional snapshot will require another 100 MB of storage space.

See Section 3.3.1, “Working with projects and streams” for additional information on creating and editing streams.

3.7.2. Creating analysis summaries

Desktop Analysis relies on analysis summary data for accurate and efficient analysis results. In order to provide this data, it is required that each stream used by Desktop Analysis have at least one snapshot containing analysis summaries.

Therefore, as part of the stream configuration process, you must complete a full Coverity Analysis for each stream in your configuration, and commit the results before any developers can start using Desktop Analysis.

Note

Analysis summaries will be captured and committed by default, unless the `cov-analyze --export-summaries` option is explicitly set to `false`.

3.7.3. Summary expiration

The Coverity Connect database can become quite sizeable when it contains multiple streams and summary data. To mitigate this, you can configure your streams to purge summary data after a set number of days. To do so, complete the following steps:

1. Navigate to Configuration → Projects & Streams.
2. Select the correct stream from the *Projects & Streams* file explorer.
3. Open the *Desktop Analysis* tab.
4. Click **Edit** and enter the desired number in the *Days before Summary Purge* field.
5. Click **Done**.

Note

Analysis summaries are generated by `cov-run-desktop` and used by Fast Desktop Analysis. If Fast Desktop is being used, purging Analysis summaries is recommended prior to Snapshot purging because it reduces database size. Analysis Summaries belonging to Streams that do not have a configured expiration value will be purged with the default scope set here.

3.7.4. Desktop Analysis usage tracking

You can use Coverity Connect to get information on the defects found specifically by Desktop Analysis users. You may wish to know, for example, how many defects are detected and fixed locally, before ever being committed to the server. See the *Coverity Desktop Analysis 2020.12: User Guide* for information on tracking Desktop Analysis usage data.

3.7.5. Analysis license service

Coverity Connect allows you to associate projects with analysis license files so that Desktop Analysis users don't need to keep license files locally. See Section 3.1.1.14, “Analysis license files” for configuration information.

Chapter 3.8. Coverity Connect URL construction

To reconstruct Coverity Connect URLs so that you can query for issues programmatically, you can use the following query prefix followed by the parameters that you need:

`/query/defects.htm`

Supported parameters:

- `cid`
- `mergeKey`
- `outstanding`
- `project` or `projectId`
- `snapshotId`
- `stream`

All parameters are optional, but you must specify at least one of `project`, `projectId`, or `stream`. If you do not specify a project, by name or ID, Coverity Connect will use the default project for the first named stream. Both `project` and `stream` expect names, while `projectId` accepts the numeric project ID. The `stream` and `cid` parameters can both appear multiple times. The `outstanding` parameter is boolean (true/false); an absent value means false. Only a single `mergeKey` parameter can be specified at one time.

If a project is not specified, and the first named stream does not belong to any projects, the URL will redirect to the *Projects* list. If an unrecognized project or stream name is given, Coverity Connect will return an error page.

Examples:

Show CIDs 1, 2 and 3 in project *ProjectA*:

```
http://machine1.eng.company.com:8080/query/defects.htm?
project=ProjectA&cid=1&cid=2&cid=3
```

Show all outstanding CIDs from streams *StreamA* and *StreamB* inside project ID *10001*:

```
http://machine1.eng.company.com:8080/query/defects.htm?
projectId=10001&stream=StreamA&stream=StreamB&outstanding=true
```

Show CIDs 1, 2 and 3 from stream *StreamA*:

```
http://machine1.eng.company.com:8080/query/defects.htm?
stream=StreamA&cid=1&cid=2&cid=3
```

Show a single issue in project ID *10001* using the issue's mergeKey:

```
http://machine1.eng.company.com:8080/query/defects.htm?
projectId=10001&mergeKey=f7e9b5e82a9046ca51c136e8786c20b2
```

Show all outstanding issues in stream *StreamA*:

```
http://machine1.eng.company.com:8080/query/defects.htm?
stream=StreamA&outstanding=true
```

Show issues for the specified snapshotId

```
/query/defects.htm?project=ProjectA&snapshotId=10040&snapshotId=10041
```

```
/query/defects.htm?project=ProjectA&cid=10020&snapshotId=10040
```

You can pass multiple shapshot ids to the url, and you can combine the snapshot id with other parameters.

Chapter 3.9. Exported defect output

Table of Contents

3.9.1. Exported defect URL variables 251
3.9.2. Exported defect XML elements 251
3.9.3. Exported defect XML file 253

3.9.1. Exported defect URL variables

The following table lists and describes the variables supplied to your chosen URL (specified in the `cim.properties` `export.issue.url` property) when you click the **Export** button.

If you have configured an `export-defect-handler` program to work with your exported defect data, see Section 3.9.2, “Exported defect XML elements”.

Table 3.9.1. Exported defect URL replacement strings

Variable	Description
`{mergedDefectId}`	If the issue being shown has a numeric CID, its ASCII decimal representation; otherwise returns an empty string.
`{projectId}`	The ASCII decimal representation of the numeric project ID.
`{userName}`	The user name of the logged in user who presses **Export**.
`{userLdapDisplayName}`	If the logged in user is an LDAP user, the administrator-provided "display name" of the LDAP server; otherwise returns an empty string.
`{mergeKey}`	The merge key of the issue shown, as 32 hexadecimal characters in `[0-9a-f]`.
`{projectName}`	The name of the Coverity Connect project associated with the issue.

3.9.2. Exported defect XML elements

The following table lists and describes the elements contained in the XML file that is produced when you use the **Export** button in the Defect Listing pane. The XML file is saved to the following location:

```
<install_dir_cc>/server/base/temp/
```

Because a defect can exist in multiple streams, and can be in different states, the XML file displays the merged state information, as well as the state information for each stream in which the defect exists.

Table 3.9.2. Exported defect XML elements

Element	Description
`<cxp:exportedDefect>`	Root element of the XML file containing a single exported (merged) defect.
`<user>`	The name of the user that exported the defect XML file.
`<ldapDomain>`	The LDAP domain of the user that exported the defect XML file. If the user is local, the LDAP domain will also be `local`.
`<project>`	The name of the project that contains the exported defect.
`<timeStamp>`	The date and time that the defect export file was created.
`<cxp:mergedDefect>`	Container element that represents the merged defect.
`<checkerName>`	The name of the checker that found this defect.
`<checkerSubcategory>`	The sub-category of the defect reported by `<checkerName>`.
`<cid>`	The CID (Coverity ID) of this defect.
`<componentName>`	The map and name of the component, joined by a period, that contains this defect. If the defect is in multiple components, the name is `Various`.
`<defectStateAttributeValues>`	The name and value an assigned defect attribute. This element contains the following child elements:
`<attributeDefinitionId>`	The name of the attribute, inside <name> tags.
`<attributeValueId>`	The value of the given attribute, inside <name> tags.
`<domain>`	The type of analysis performed to find this defect: static C/C++, static C#, static Java, or dynamic.
`<filePathname>`	The complete path to the file that contains the defect.
`<firstDetected>`	The date and time in which the defect was first detected by the analysis.
`<firstDetectedSnapshotId>`	The snapshot in which the defect was first detected.
`<functionDisplayName>`	The name of the function that contains the defect.
`<lastDetected>`	The date and time in which the analysis most recently detected the defect.

Element	Description
`<lastDetectedSnapshotId>`	The most recent snapshot containing the defect. This is the same value as in `<latestSnapshotId>`.
`<mergeKey>`	A unique identifier that maps a CID across multiple projects or Coverity Connect instances.
`<occurrenceCount>`	The number of streams where the defect is found.
`<latestSnapshotId>`	The most recent snapshot containing the defect. This is the same value as in `<lastDetectedSnapshotId>`.
`<streamDefects>`	Container element that represents the occurrences of this CID in all streams within the project.
`<cxp:streamDefect>`	Container element that represents a single stream defect.
`<checkerSubcategoryId>`	The sub-category of this defect within the class of defects that are discoverable by the checker reported in the `<checker>` element.
`<subcategory>`	The sub-category of the defect reported by the checker.
`<checkerName>`	The name of the checker that caught this defect.
`<domain>`	The type of analysis performed to find this defect: static C/C++, static C#, static Java, or dynamic.
`<subcategory>`	The sub-category of the defect reported by the checker.
`<cid>`	The CID (Coverity ID) of this defect.
`<id>`	The identification container for information identifying the stream defect.
`<defectTriageId>`	Internal reference to latest triage ID.
`<defectTriageVerNum>`	The version number of the latest triage.
`<id>`	The ID of this stream defect.
`<verNum>`	An internal field used to prevent data corruption.
`<streamId>`	The name of the stream containing the stream defect, inside <name> tags.

3.9.3. Exported defect XML file

This example shows a defect (CID 310001) that exists in stream `emp_java` and stream `ns1`. The triage states differ in each stream. The exported XML file shows the merged triage state in the `<cxp:mergedDefect>` element, as well as the defect's triage state for each stream in the `<cxp:streamDefect>` elements.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cxp:exportedDefect xmlns:cxp="http://export.coverity.com/v9">
    <user>admin</user>
    <ldapDomain>local</ldapDomain>
    <project>EMPv2.0</project>
    <timeStamp>2015-10-01T14:49:10.916-07:00</timeStamp>
    <cxp:mergedDefect>
        <checkerName>FORWARD_NULL</checkerName>
        <checkerSubcategory>deref_constant_null</checkerSubcategory>
        <cid>310001</cid>
        <componentName>Default.Other</componentName>
        <defectStateAttributeValues>
            <attributeDefinitionId>
                <name>DefectStatus</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name>New</name>
            </attributeValueId>
        </defectStateAttributeValues>
        <defectStateAttributeValues>
            <attributeDefinitionId>
                <name>Classification</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name>Unclassified</name>
            </attributeValueId>
        </defectStateAttributeValues>
        <defectStateAttributeValues>
            <attributeDefinitionId>
                <name>Action</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name>Undecided</name>
            </attributeValueId>
        </defectStateAttributeValues>
        <defectStateAttributeValues>
            <attributeDefinitionId>
                <name>Severity</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name>Unspecified</name>
            </attributeValueId>
        </defectStateAttributeValues>
        <defectStateAttributeValues>
            <attributeDefinitionId>
                <name>Fix Target</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name>Untargeted</name>
            </attributeValueId>
        </defectStateAttributeValues>
        <defectStateAttributeValues>
            <attributeDefinitionId>
```

```
                <name>Legacy</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name>False</name>
            </attributeValueId>
        </defectStateAttributeValues>
        <defectStateAttributeValues>
            <attributeDefinitionId>
                <name>Owner</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name>Unassigned</name>
            </attributeValueId>
        </defectStateAttributeValues>
        <defectStateAttributeValues>
            <attributeDefinitionId>
                <name>TranslatedOwner</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name>Unassigned</name>
            </attributeValueId>
        </defectStateAttributeValues>
        <defectStateAttributeValues>
            <attributeDefinitionId>
                <name>OwnerName</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name></name>
            </attributeValueId>
        </defectStateAttributeValues>
        <defectStateAttributeValues>
            <attributeDefinitionId>
                <name>Comment</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name></name>
            </attributeValueId>
        </defectStateAttributeValues>
        <defectStateAttributeValues>
            <attributeDefinitionId>
                <name>ExternalReference</name>
            </attributeDefinitionId>
            <attributeValueId>
                <name></name>
            </attributeValueId>
        </defectStateAttributeValues>
        <domain>STATIC_JAVA</domain>
        <filePathname>/users/sample/document/Test2.java</filePathname>
        <firstDetected>2015-07-16T07:55:31.241-07:00</firstDetected>
        <firstDetectedSnapshotId>10031</firstDetectedSnapshotId>
        <functionDisplayName>Test2.main(java.lang.String[])</functionDisplayName>
        <lastDetected>2015-10-01T14:47:17.347-07:00</lastDetected>
        <lastDetectedSnapshotId>10031</lastDetectedSnapshotId>
        <mergeKey>d9228ebd8c4a79e6f152e6e700e1ef95</mergeKey>
        <occurrenceCount>1</occurrenceCount>
    </cxp:mergedDefect>
    <latestSnapshotId>10031</latestSnapshotId>
    <streamDefects>
        <cxp:streamDefect>
            <checkerSubcategoryId>
                <checkerName>FORWARD_NULL</checkerName>
                <domain>STATIC_JAVA</domain>
                <subcategory>deref_constant_null</subcategory>
            </checkerSubcategoryId>
            <cid>310001</cid>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>DefectStatus</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>New</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Classification</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>Unclassified</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Action</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>Undecided</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Fix Target</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>Untargeted</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Severity</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>Unspecified</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
```

```
                <attributeDefinitionId>
                    <name>Legacy</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>False</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Comment</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name></name>
                </attributeValueId>
            </defectStateAttributeValues>
            <id>
                <defectTriageId>1</defectTriageId>
                <defectTriageVerNum>0</defectTriageVerNum>
                <id>310001</id>
                <verNum>0</verNum>
            </id>
            <streamId>
                <name>ns1</name>
            </streamId>
        </cxp:streamDefect>
        <cxp:streamDefect>
            <checkerSubcategoryId>
                <checkerName>FORWARD_NULL</checkerName>
                <domain>STATIC_JAVA</domain>
                <subcategory>deref_constant_null</subcategory>
            </checkerSubcategoryId>
            <cid>310001</cid>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>DefectStatus</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>New</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Classification</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>Unclassified</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Action</name>
                </attributeDefinitionId>
                <attributeValueId>
```

```
                    <name>Undecided</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Fix Target</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>Untargeted</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Severity</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>Unspecified</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Legacy</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name>False</name>
                </attributeValueId>
            </defectStateAttributeValues>
            <defectStateAttributeValues>
                <attributeDefinitionId>
                    <name>Comment</name>
                </attributeDefinitionId>
                <attributeValueId>
                    <name></name>
                </attributeValueId>
            </defectStateAttributeValues>
            <id>
                <defectTriageId>1</defectTriageId>
                <defectTriageVerNum>0</defectTriageVerNum>
                <id>310007</id>
                <verNum>0</verNum>
            </id>
            <streamId>
                <name>emp_java</name>
            </streamId>
        </cxp:streamDefect>
    </streamDefects>
    <cxp:checkerProperties>
        <subcategoryShortDescription>Explicit null dereferenced</
subcategoryShortDescription>
    </cxp:checkerProperties>
</cxp:exportedDefect>
```

Chapter 3.10. Suppressing built-in views for new users

Coverity Connect allows you prevent the display of any of the built-in views from new users. This feature is useful for any views that are not needed by your company. This process requires you to modify a Coverity Connect properties file.

For example, if you want to suppress the built-in *Low Impact Outstanding* and *Medium Impact Outstanding* views, you need to add the following line to `cim.properties` (located in `<install_dir>/config`):

```
suppressedFactoryViews=issues.mediumimpactoutstanding,issues.lowimpactoutstanding
```

Note

You do not need to restart Coverity Connect after suppressing views through `cim.properties`.

This change affects new users only. You cannot suppress the views from users who already have them. However, the users could delete the views through the UI. See Section 2.4.2.5, "Delete".

Table 3.10.1. Coverity Connect View Properties

View Type	Internal View Type Property	View Name	Internal View Property
Issues: By Snapshot	`issues`	*High Impact Outstanding*	`highimpactoutstanding`
		Outstanding Untriaged	`alluntriaged`
		My Outstanding	`myoutstanding`
		Outstanding Defects	`outstandingdefects`
		Outstanding Security Risks	`outstandingsecurityrisks`
		Outstanding Test Rules Violations	`outstandingtestruleviolations`
Issues: Project Scope	`issues`	*All in Project*	`allinproject`
Files	`files`	*In Latest Snapshot*	`inlatestsnapshot`
		Uncovered By Tests	`uncoveredbytests`
Functions	`functions`	*High CCM (>15)*	`highccm`
		In Latest Snapshot	`inlatestsnapshot`
		Uncovered By Tests	`uncoveredbytests`
Components	`components`	*All In Project*	`allinproject`

View Type	**Internal View Type Property**	**View Name**	**Internal View Property**
		High Issue Density (> 1)	`highdefectdensity`
		With Outstanding Issues	`withoutstandingissues`
		With Untriaged Issues	`withuntriagedissues`
Checkers	`checkers`	*All In Project*	`allinproject`
Owners	`owners`	*All In Project*	`allinproject`
Snapshots	`snapshots`	*All In Project*	`allinproject`
Tests	`tests`	*All Tests*	`inlatestsnapshot`
		Currently Failing	`currentlyfailing`

Chapter 3.11. Changing the date format display in Coverity Connect

You can change the way in which the date format is displayed for the English, Japanese, Chinese, and Korean locales in certain view types and columns in Coverity Connect.

You can change the date format as follows:

1. Create a new properties file.

   For the English locale:

   ```
   <install_dir_cp>/config/format_en.properties
   ```

   For the Japanese locale:

   ```
   <install_dir_cp>/config/format_ja.properties
   ```

   For the Chinese locale:

   ```
   <install_dir_cp>/config/format_zh_CN.properties
   ```

   For the Korean locale:

   ```
   <install_dir_cp>/config/format_ko.properties
   ```

2. Add a new date format to the `dateFormat` property in the `format_<locale>.properties file`. The following table lists the date format properties that can be changed, their defaults, and where in Coverity Connect they will be displayed.

3. Restart the Coverity Connect server to enable the changes.

Table 3.11.1. Date properties

Date property	Defaults to	Displayed in
`dateFormat`	`MM/dd/yy` for English, `yyyy/MM/dd` for Japanese, Chinese, and Korean	• Date-valued columns in the Issues: by Snapshot view type (*First Detected*, *Last Detected*, *Last Triaged*) and the Issues: Project Scope view type (*First Detected*, *Last Detected*). • The same columns (as above) in email notification table data. • Dates for *Detection History* and *Triage History* sidebars.

Date property	Defaults to	Displayed in
`timeFormat`	`hh:mm a` for English, Japanese, Chinese, and Korean	Date and time values in Coverity Connect configuration (that is, through the Configuration menu). The following use a combination of `dateFormat` and `timeFormat`: • Projects & Streams → Snapshot → Created column • System → Sign In Log → Session Start column
`timeStampFormat`	`yyyy-MM-dd HH:mm:ss` for English, Japanese, Chinese, and Korean	Date+time-valued columns in Functions views (Last Impacted, Last Modified) and Snapshots (Date) views
`scmDateFormat`	Defaults to `MMM d, yyyy` for English, `yyyy/MM/dd` for Japanese, Chinese, and Korean	SCM display from the Show Source Gutter Menu in the source browser.
`emailNotificationDateFormat`	`MMMM d, yyyy` for English, `yyyy年M月d日` for Japanese, `yyyy年M月d天`for Chinese, and `yyyy년 M월 d일` for Korean	The date used in the body text of an email notification (table data uses `dateFormat` like the web view).
`policyManagerTrend.dayFormat`	`MMM dd` for English, Japanese, Chinese, and Korean	The date used for day-to-day Policy Manager trend reports.
`policyManagerTrend.weekFormat`	`MMM dd` for English, Japanese, Chinese, and Korean	The date used for weekly Policy Manager trend reports.
`policyManagerTrend.monthFormat`	`MMM yyyy` for English, Japanese, Chinese, and Korean	The month format used for monthly Policy Manager trend reports.
`policyManagerTrend.yearFormat`	`yyyy` for English, Japanese, Chinese, and Korean	The year format used for year-to-year Policy Manager trend reports.

The following example shows the `format_ja.properties` file, with the date format set to the American date format.

```
dateFormat=MM/dd/yy
timeFormat=hh:mm
timeStampFormat=yyyy-MM-dd HH:mm:ss
scmDateFormat=yyyy/MM/dd
emailNotificationFormat=MMMM d, yyyy
policyManagerTrend.dayFormat=MMM dd
policyManagerTrend.weekFormat=MMM dd
policyManagerTrend.monthFormat=MMM yyyy
policyManagerTrend.yearFormat=yyyy
```

For more information about how date time strings function, see http://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html.

Chapter 3.12. Configuring Coverity Desktop and Architecture Analysis through Coverity Connect

Table of Contents

3.12.1. Configuring Coverity Desktop and shared files through the Downloads page 264
3.12.2. Downloading a CVA file for Architecture Analysis 267

3.12.1. Configuring Coverity Desktop and shared files through the Downloads page

This chapter provides the following information about configuring Coverity Connect to work with Coverity Desktop and shared custom files:

- Configuring Coverity Connect for use with Desktop Analysis.
- Configuring the Coverity Desktop download packages and including shared files on the Coverity Connect *Downloads* page.

3.12.1.1. Configuring the Coverity Connect Downloads page

The *Downloads* page is available through the Coverity Connect User menu. This page provides a central location in which users can:

- Download the Coverity Desktop product packages for Eclipse, QNX Momentics, Wind River Workbench, IBM RTC, Visual Studio, IntelliJ, and Android Studio.
- Eclipse Update site
- Coverity Desktop Analysis Tools
- Gradle Plug-In
- Coverity Desktop Reports, which include the MISRA Report, Security Report, and Coverity Integrity Report
- Software Development Kits
- Other content, such as scripts or additional documentation. For information about making this content available, see Section 3.12.1, “Configuring Coverity Desktop and shared files through the Downloads page”.

By default, the Coverity Connect installer includes the Coverity Desktop 2020.12 packages so that they are ready to download. There is no extra configuration required. A user with appropriate credentials can sign into Coverity Connect, go to the *Downloads* page, and download the appropriate Coverity Desktop installation file. Eclipse and Visual Studio users can also obtain a link to the Coverity Desktop update or gallery site, so the IDE can search for and install the most current version of the plug-in that is made available through the *Downloads* page.

In order for the Eclipse update site to work properly with an SSL connection, the update site requires a valid certificate installed for the Coverity Connect server. Otherwise users will have to download the plug-ins and install them locally.

3.12.1.1.1. Adding Coverity Analysis to the Downloads page

Coverity Connect allows you to add Coverity Analysis product packages and license files to the *Downloads* page, so that Coverity Desktop users can obtain and install Coverity Analysis from a central location:

1. Obtain the Coverity Analysis packages (`.exe` for Windows systems or `.sh` for Unix) that you want to make available to your users.
2. Copy the Coverity Analysis installer packages into the `<install_dir_cc>/server/base/webapps/downloads` directory.
3. Optionally copy the Coverity Analysis license (`license.dat`) file into same directory.

Users with proper sign-in credentials and permissions can now download the Coverity Analysis package and install it on their system.

Note

Make sure the Coverity Analysis package version is compatible with the Coverity Connect version. Version mismatches can cause runtime failures. For more information, see *Coverity 2020.12 Installation and Deployment Guide*

3.12.1.1.2. Adding Coverity Desktop updates

If an incremental release of the Coverity Desktop product becomes available, you can update the central download page with the new packages by completing the following:

1. Go to the `<install_dir_cc>/server/base/webapps/` directory.
2. Remove the following files:
   a. In the `downloads` directory, remove the following files:
      - `cov-desktop-eclipse-2020.12.zip`
      - `cov-desktop-windriver-2020.12.zip`
      - `cov-desktop-qnx-2020.12.zip`
      - `cov-desktop-microsoft-visual-studio-2020.12.zip` (for Visual Studio versions 2008-2010)

  - `Coverity.Desktop.vsix` (for Visual Studio 2013-2017)

b. Remove the following directories:

  - `coverity-desktop-eclipse/update`
  - `coverity-desktop-windriver/update`
  - `coverity-desktop-qnx/update`
  - `coverity-desktop-visual-studio/gallery`

3. Download updated Eclipse, Wind River WorkBench, QNX Momentics, and Visual Studio `.zip` packages.

4. For Eclipse, Wind River, QNX, and Visual Studio versions 2008-2010, save the `.zip` packages in the `<install_dir_cc>/server/base/webapps/downloads` directory.

   For Visual Studio 2013-2017, extract the `cov-desktop-microsoft-visual-studio-gallery-2020.12.zip` file. Then copy `Coverity.Desktop.vsix` into the `downloads` directory.

   Note

   `Coverity.Desktop.vsix` is found in the extracted `.zip` package, under `coverity-desktop-vs/gallery`.

5. For Eclipse, unpack `cov-desktop-eclipse-2020.12.zip`, then move the extracted directories to `coverity-desktop-eclipse/update`.

   For Wind River, unpack `cov-desktop-windriver-2020.12.zip`, then move the extracted directories to `coverity-desktop-windriver/update`.

   For QNX, unpack `cov-desktop-qnx-2020.12.zip`, then move the extracted directories to `coverity-desktop-qnx/update`.

   For Visual Studio 2013-2017, copy the extracted `gallery` directory to `coverity-desktop-visual-studio/gallery`.

The user can now download the new versions of the plug-in files and also point at a new version of the Eclipse update or Visual Studio 2013-2017 gallery site.

3.12.1.1.3. Adding and removing custom files

Coverity Connect administrators can add custom files, such as configuration files, documentation, scripts, and so forth. Developers can then access these files from the central Downloads page.

To add a custom file:

1. Copy your file to the following directory:

   `<install_dir_cc>/server/base/webapps/downloads`

2. In the `/downloads` directory, edit the `fileConfig.xml` and add new file nodes for each custom file. For example:

```
<customFiles>
       <!-- example file node entry -->
       <file>
           <fileName>example.txt</fileName> <!-- mandatory field -->
           <displayName>example display</displayName> <!-- optional field -->
           <fileDescription>example description</fileDescription> <!-- optional
field -->
       </file>
       </customFiles>
```

   - The `<fileName>` tag is mandatory and corresponds to the name of the hosted file. If this is the only entry is present in the `<file>` node, then a link to this file name is displayed on the Downloads page.
   - The `<displayName>` tag is optional and when it is defined, it overwrites the `<fileName>` tag in the link that is displayed on the Downloads page.
   - The `<fileDescription>` tag is optional and when it is defined, it adds a description to the link on the Downloads page for the file in the same `<file>` tag.

To remove a custom file:

1. Remove file from the `/downloads` directory.
2. Edit the `fileConfig.xml` in the `/downloads` directory and remove the corresponding `<file>` tag.

3.12.2. Downloading a CVA file for Architecture Analysis

After you use `cov-export-cva` with the `--output-file` option to commit issues into a Coverity Connect stream, you need to navigate to the stream configuration in Coverity Connect.

This task requires a user role that has administrative permissions to the stream.

To download a CVA file:

1. Navigate to Configuration → Projects & Streams.
2. Select the stream to which the issues were committed.
3. Click the *Snapshots* tab for that stream.
4. Scroll to the snapshot record for the commit from which you need the CVA file.

   The snapshot record will contain a *CVA* column that contains a `*.cva` file entry.
5. Click the CVA file name (for example, `10002.cva`).
6. Download the CVA file.

Part 4. Coverity Policy Manager Usage

Leaders of software development teams and developers use Coverity Policy Manager charts to monitor issues found by Coverity Analysis and third-party tools.

Important

In general, Policy Manager does not store data indefinitely. In the absence of other activity, daily data is kept for 40 days, weekly data is kept for 30 weeks, monthly data is kept for 24 months, and only yearly data is kept indefinitely. However, Policy Manager does not save historical data in terms of previous configurations of component map and stream settings, and if the current project configurations are changed, Policy Manager data is recomputed based on those changes.

Chapter 4.1. Coverity Policy Manager Overview

Table of Contents

4.1.1. Hierarchies 269
4.1.2. Metrics 270
4.1.3. Filters 270
4.1.4. Reports 271
4.1.5. Dashboards 274
4.1.6. Heatmaps 274
4.1.7. Policies 278
4.1.8. Summary Metrics 278
4.1.9. Views 281

Coverity Policy Manager charts (reports and heatmaps) use metrics (and filters on metrics) to get information about data that is stored in the Coverity Connect database. For example, a chart might display the number of outstanding, high-impact issues found in a portion of the code base that is associated with a certain Coverity Connect project. Charts get their structure from a hierarchy that is set up by a Coverity Policy Manager administrator.

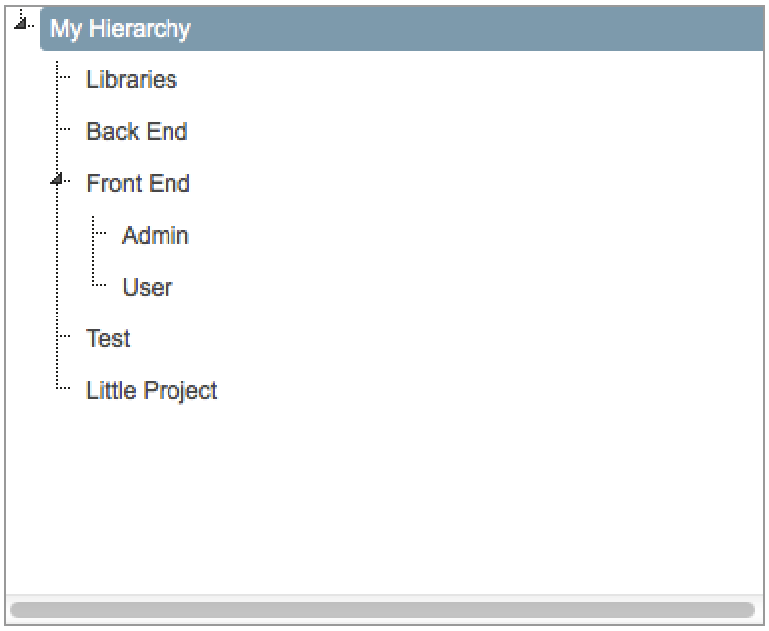
After an administrator configures one or more hierarchies, it becomes possible to navigate to them through the Main Menu (see Figure 1.3.2, “Example: All Hierarchies”) and then set up and view Coverity Policy Manager heatmaps, reports, and dashboards. (Permissions pertaining to viewing and managing Coverity Policy Manager apply; see Chapter 5.5, *Coverity Policy Manager Roles and Permissions*).

4.1.1. Hierarchies

Coverity Policy Manager administrators specify one or more hierarchies through the *Hierarchies* configuration screen. For details, see Section 5.1.2, “Configuring a Hierarchy”.

A hierarchy is a tree data structure (a **node tree**) that underlies one or more Coverity Policy Manager heatmaps or reports. Each node in a hierarchy represents some part of your code base that has been analyzed and committed to an instance of Coverity Connect that shares data with Coverity Policy Manager.

Figure 4.1.1. Example: Node Tree for a Hierarchy



The terminal (leaf) nodes in a hierarchy are the source of data for the higher level nodes, which aggregate data from lower level nodes. This aggregation allows you to examine the code base at increasingly inclusive levels, as described in the Coverity Policy Manager Use Case in Chapter 5.1, *Managing a Coverity Policy Manager Hierarchy*.

Leaf nodes are associated with a Coverity Connect project. Coverity Policy Manager Administrators can limit the scope of data for a leaf node to one or more Coverity Connect components that are associated with the selected project.

4.1.2. Metrics

Coverity Policy Manager metrics provide numeric information on data that is stored in the Coverity Connect database. Metrics are available for use in Coverity Policy Manager heatmaps and reports. Examples include outstanding issue count and lines of code. For information about all the metrics that are available in Coverity Policy Manager, see Chapter 4.3, *Coverity Policy Manager Metrics*.

See also Summary Metrics.

4.1.3. Filters

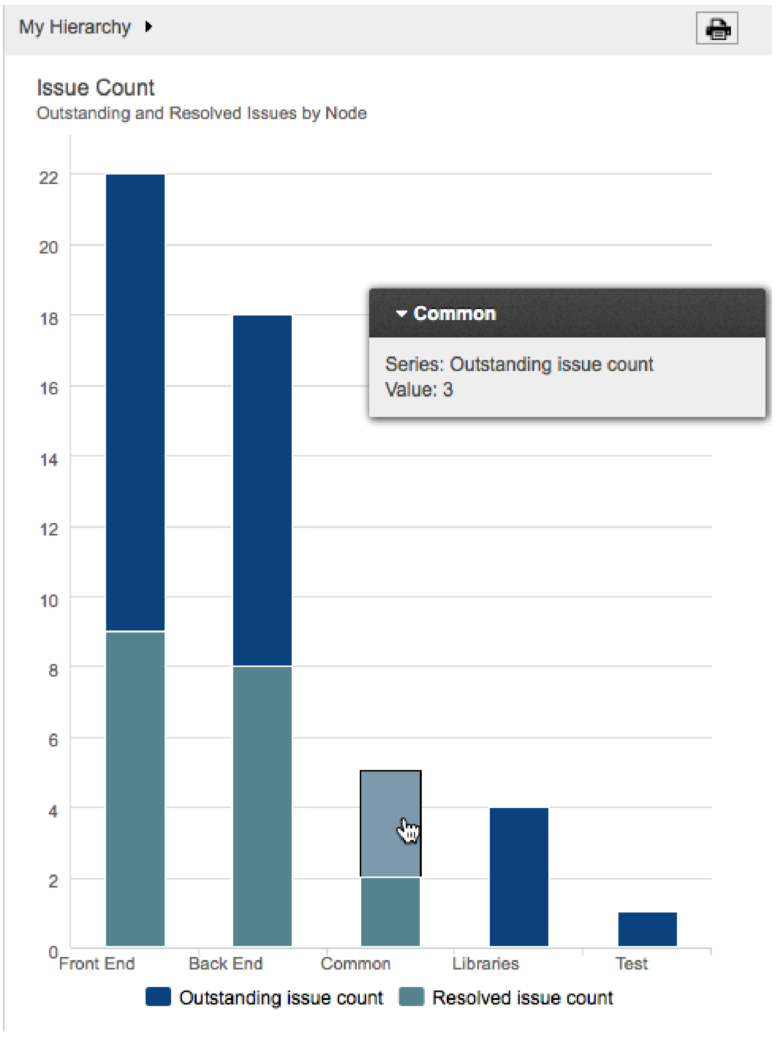
Coverity Policy Manager filters narrow the scope of the data found by a metric. For example, you might want to see data on issues of medium impact only, excluding the data on issues of high and low impact. For descriptions of filters, see Chapter 4.4, *Coverity Policy Manager Filter and Segmentation Properties*.

4.1.4. Reports

Coverity Policy Manager reports display numerical data that is derived from metrics. Reports can display data in bar, column, pie, table, line, and area formats. All reports provide navigation breadcrumbs (located above each report) so that you can view report data for a node at any level in the hierarchy (from the root, to any branch, to any leaf). You can also save a report to file and print out hard copies of the file (see Figure 4.2.5, “Printer Icon”). In addition, you can add one or more reports to a Coverity Policy Manager dashboard.

- **Status Reports:** Status reports show one or more specified metrics plotted against either components or hierarchy child nodes, as a bar chart, column chart, pie chart, or table.

Figure 4.1.2. Example: Status Report

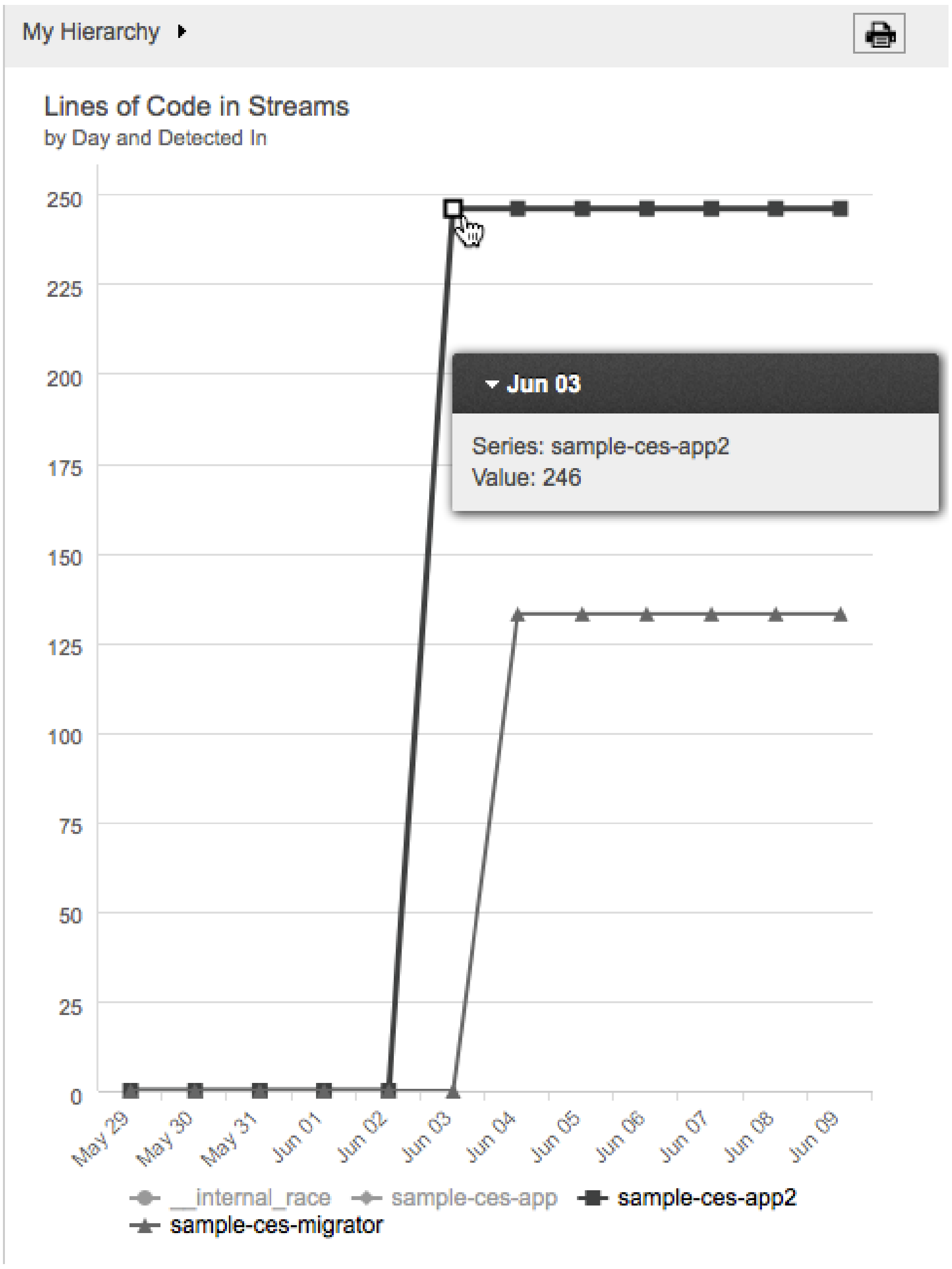


The example shows data for each child node of the root node (My Hierarchy). In the report, the Common pop-up window displays data on the selected portion of the Common node, *Outstanding issue count*.

To set up a Status Report, see Section 4.2.5, “Setting up Coverity Policy Manager Status Reports”.

- **Trend Reports:** Trend reports display changes in data points over time.

Figure 4.1.3. Example: Trend Report



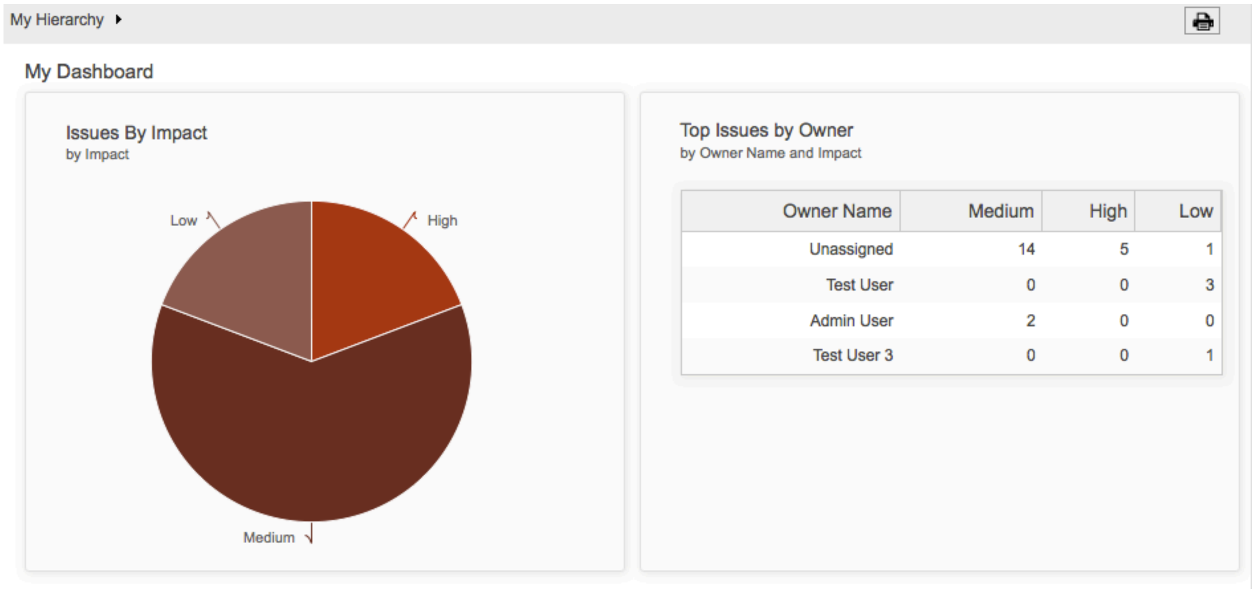
The example selects a data point. The pop-up window associated with the selected data point displays details on that data.

To set up a Trend Report, see Section 4.2.6, “Setting up Coverity Policy Manager Trend Reports”.

4.1.5. Dashboards

Dashboards display a collection of reports and/or heatmaps. You can save a dashboard to file and print out hard copies of the file (see Figure 4.2.5, “Printer Icon”).

Figure 4.1.4. Example: Dashboard



Owner Name	Medium	High	Low
Unassigned	14	5	1
Test User	0	0	3
Admin User	2	0	0
Test User 3	0	0	1

To set up a dashboard, see Section 4.2.3, “Setting up Coverity Policy Manager Dashboards”.

4.1.6. Heatmaps

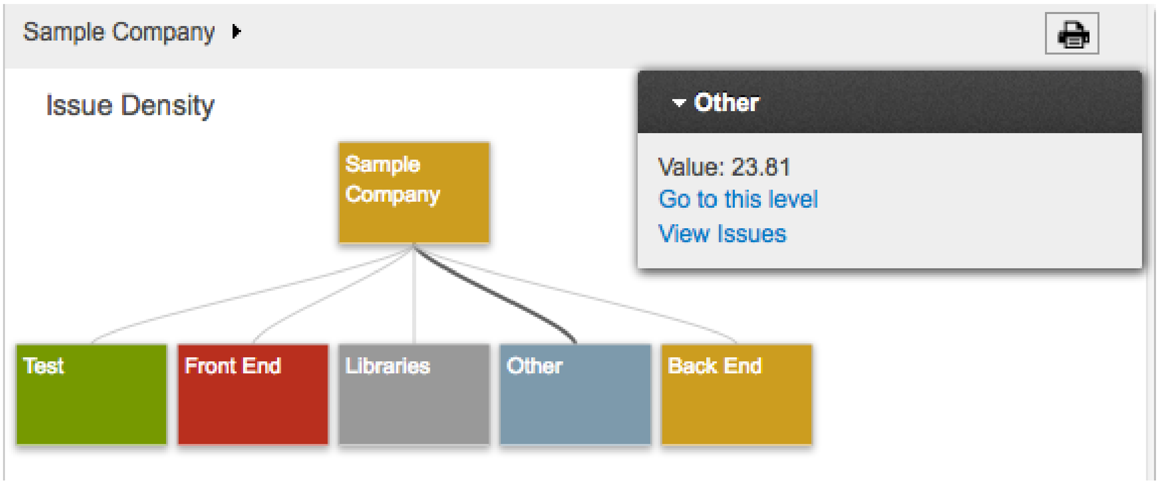
Tree, Sunburst, and Banded Trend maps divide your code base into segments that are color-coded according to their level of compliance with policies that you specify for them. Higher-level map nodes reflect overall compliance based on policy-related data that is aggregated from lower-level segments.

In addition to containing red (in violation), yellow (at risk), and green (in compliance) segments, heatmaps can also contain gray segments. Data within the scope of a gray segment (for example, the Libraries node in the following figure) is excluded from higher-level segments that contain the gray segment. The gray segments might include uninteresting or peripheral data on libraries, certain third-party code, infrequently used legacy code, or some other area of your code base. (Note that the Other node is blue because it has been selected.)

Tree Map

- Tree maps contain an all-inclusive root with branching segments.

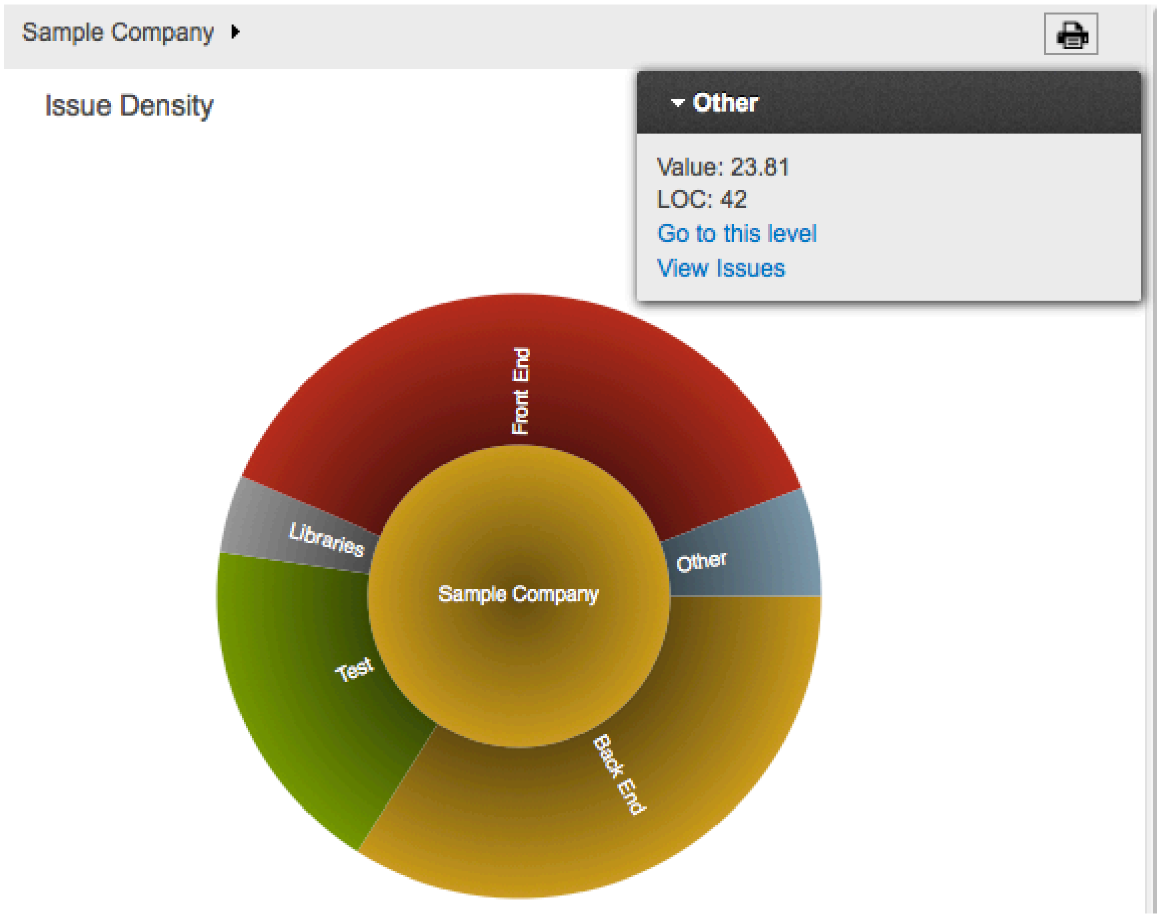
Figure 4.1.5. Example: Tree Map



Sunburst Map

- The Sunburst map displays data in rings, with the center ring as the most inclusive node (for example, the root node) and the surrounding rings as segments of the parent ring. The segments of the Sunburst are sized according to the relative number of lines of code they represent. Otherwise, the Tree and Sunburst maps provide identical data.

Figure 4.1.6. Example: Sunburst Map



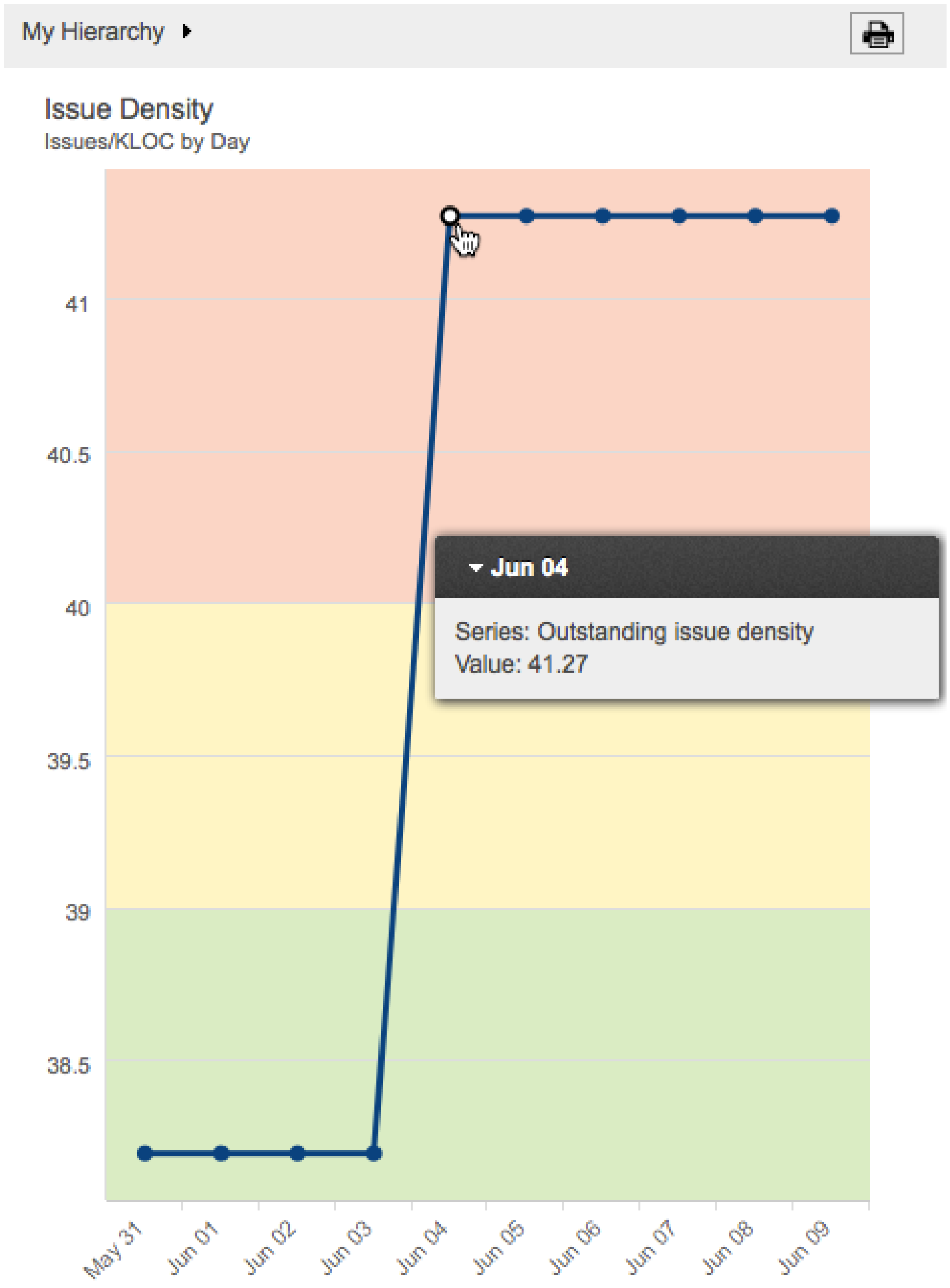
Value: 23.81
LOC: 42

Notice that the Other node is selected in both Figure 4.1.6, “Example: Sunburst Map” and Figure 4.1.5, “Example: Tree Map” to display information about it in the *Other* pop-up window. In addition to showing the issue density value, this window provides a link to the issue list (see *View issues*) for this node and allows you to display this node and any subnodes it contains (see *Go to this level*).

Banded Trend Map

- The Banded Trend map applies policy bands to data in a trend report. Figure 4.1.7, “Example: Banded Trend Map” shows the issue density for the past 10 weeks. It also shows the selection of the data point to reveal the precise issue density on the selected day.

Figure 4.1.7. Example: Banded Trend Map



Series: Outstanding issue density
Value: 41.27

The Tree and Sunburst maps can display up to four levels of the node tree. However, all heatmaps provide navigation breadcrumbs (located above each heatmap) so that you can view heatmap data for a node at any level in the hierarchy (from the root, to any branch, to any leaf). You can also save a heatmap to file and print out hard copies of the file (see Figure 4.2.5, “Printer Icon”). In addition, you can add one or more heatmaps to a Coverity Policy Manager dashboard.

To set up a heatmap, see Section 4.2.4, “Setting up Coverity Policy Manager Heatmaps”.

4.1.7. Policies

Coverity Policy Manager policies are thresholds you set in heatmaps so that you can evaluate the integrity of your code base, and so that you can check other information, such as Coverity Connect activity levels. In Coverity Policy Manager, a policy is expressed as a simple formula that, when applied to report data, determines whether a given data point is compliant (displayed in green), at risk (displayed in yellow), or in violation (displayed in red) of your standards. For example, an Issue Density report might display in a yellow band the data points that fall within 1.0-1.8 software issues per thousand lines of code (issues/KLOC) and show in a red band the data points that exceed 1.8 issues/KLOC. Data points below 1.0 issues/KLOC would appear in a green band. Coverity Policy Manager permits one policy per heatmap.

For an example that shows how to set a policy for a heatmap, see Figure 4.2.8, “Example: Heatmap Edit Settings Window”.

4.1.8. Summary Metrics

Summary metrics aggregate data from built-in metrics. An example is the built-in summary metric, Technical Debt, which by default aggregates data from metrics on outstanding issue and function counts. Like all Summary metrics, its contributor metrics can be filtered (see Section 4.1.3, “Filters”) and weighted. As shown in Table 4.1.1, “Built-in Summary Metric: Technical Debt”, outstanding issue metrics are filtered according to their impact, and the function count metric is filtered according to Cyclomatic Complexity (CCM). The value of each contributor to the summary metric is weighted by a multiplier to produce a relative value for each metric.

Table 4.1.1. Built-in Summary Metric: Technical Debt

Metrics	**Label**[a]	**Filter**[b]	**Multiplier**[c]
Outstanding Issue Count	High-impact issues	Impact=High	20
Outstanding Issue Count	Medium-impact issues	Impact=Medium	10
Outstanding Issue Count	Low-impact issues	Impact=Low	5
Outstanding Issue Count	Audit-impact issues	Impact=Audit	1
Function Count	Functions with CCM > 15	CCM > 15	10

[a]Administrator-provided name of a filtered metric that serves as a contributor to a summary metric. The name appears in charts and heatmaps.

[b]Default filters: Filter outstanding issues by impact, and filter functions by complexity.
[c]Default weight applied to each filtered metric.

Note

Before using the *Technical Debt* summary metric, you should find out if an administrator has tuned it to meet the needs of your organization. Alternatively, you can create a test report and select a data point to see what values are set (see Figure 4.1.8, “Example: Summary Metric used in a Heatmap”).

Technical Debt
: Technical debt often accrues in code where the demand for rapid development can take precedence over established code design and development standards. High levels of technical debt can make a code base difficult and time-consuming for development teams to successfully maintain, modify, and troubleshoot.

Just as the weighted values of the contributor metrics can serve as a data points in your Coverity Policy Manager charts and heatmaps, so too can the sum of these values. In the following Technical Debt example (Table 4.1.2, “Example: Technical Debt of 80”), the summary metric value of 80 is simply the sum of its contributor values. The value of each contributor is equal to the product of its filtered metric and multiplier.

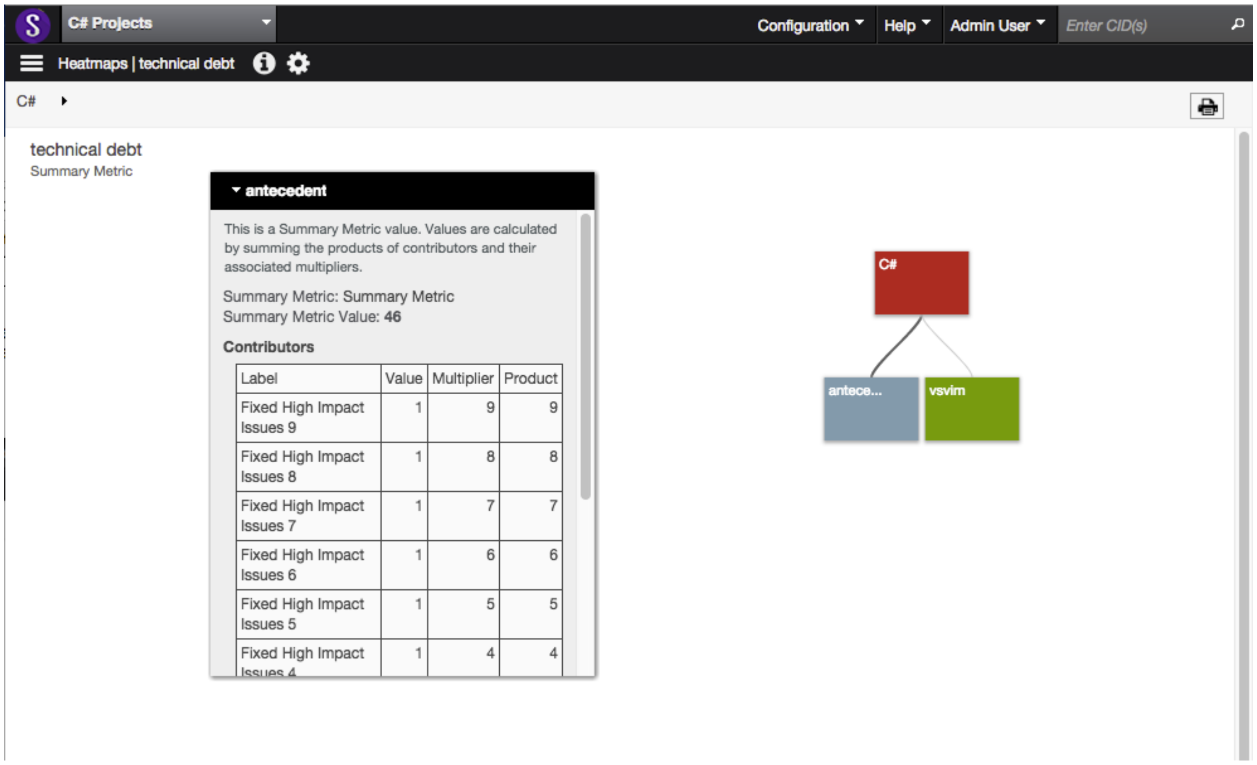
Table 4.1.2. Example: Technical Debt of 80

Technical Debt Contributors[a]	**(A) Filtered Metric Value**	**(B) Multiplier Value**	**(A x B) Contributor Value**	**Summary Metric Value**
High-impact issues	1	20	20	80 (= Sum of Contributor Values)
Medium-impact issues	2	10	20	
Low-impact issues	4	5	20	
CCM > 15	2	10	20	

[a] **Contributors:** Set of one or more filtered metrics and the multipliers used to weight their values.

In charts, Summary metrics look and behave much like other metrics. You can select them as data sources for reports and heatmaps. As shown in the following examples, you can click nodes and other data points in charts to see their properties, including the values of contributors that make up Summary metrics.

Figure 4.1.8. Example: Summary Metric used in a Heatmap



This is a Summary Metric value. Values are calculated by summing the products of contributors and their associated multipliers.

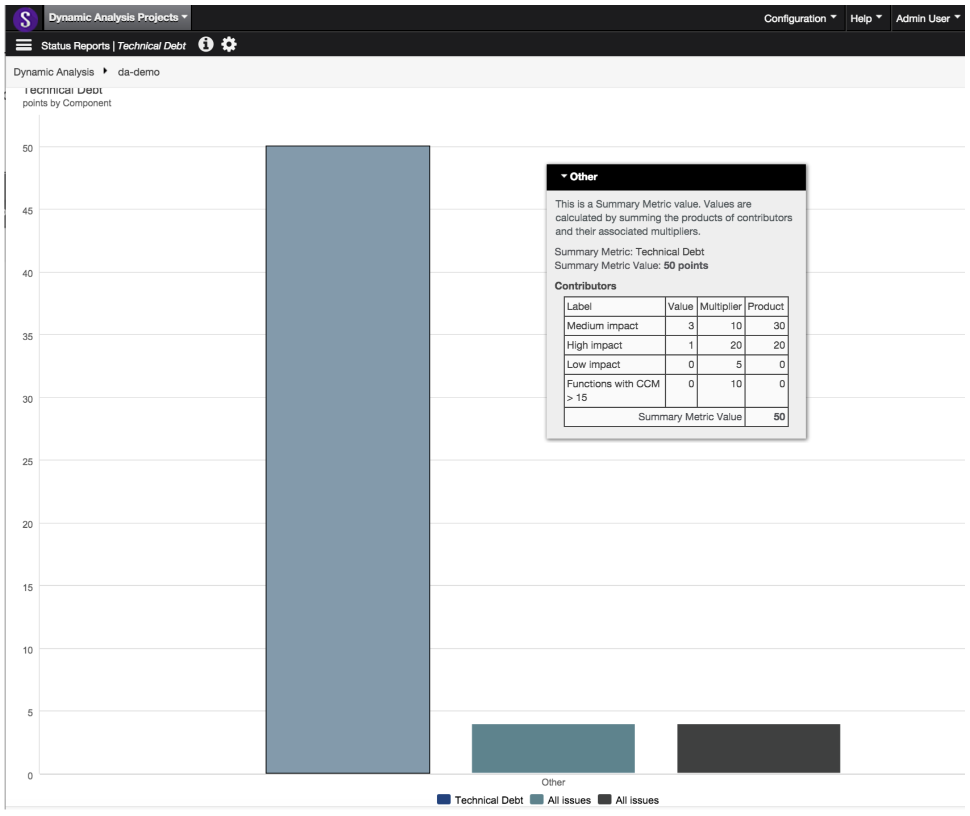
Summary Metric: Summary Metric
Summary Metric Value: **46**

Contributors

Label	Value	Multiplier	Product
Fixed High Impact Issues 9	1	9	9
Fixed High Impact Issues 8	1	8	8
Fixed High Impact Issues 7	1	7	7
Fixed High Impact Issues 6	1	6	6
Fixed High Impact Issues 5	1	5	5
Fixed High Impact Issues 4	1	4	4

As shown in the following example, it is possible to segment reports by contributors to a summary metric. (See also, Section 4.3.4, “Summary Metrics: Technical Debt”.)

Figure 4.1.9. Example: Summary Metric used in a Status Report



Other

This is a Summary Metric value. Values are calculated by summing the products of contributors and their associated multipliers.

Summary Metric: Technical Debt
Summary Metric Value: **50 points**

Contributors

Label	Value	Multiplier	Product
Medium impact	3	10	30
High impact	1	20	20
Low impact	0	5	0
Functions with CCM > 15	0	10	0
Summary Metric Value			**50**

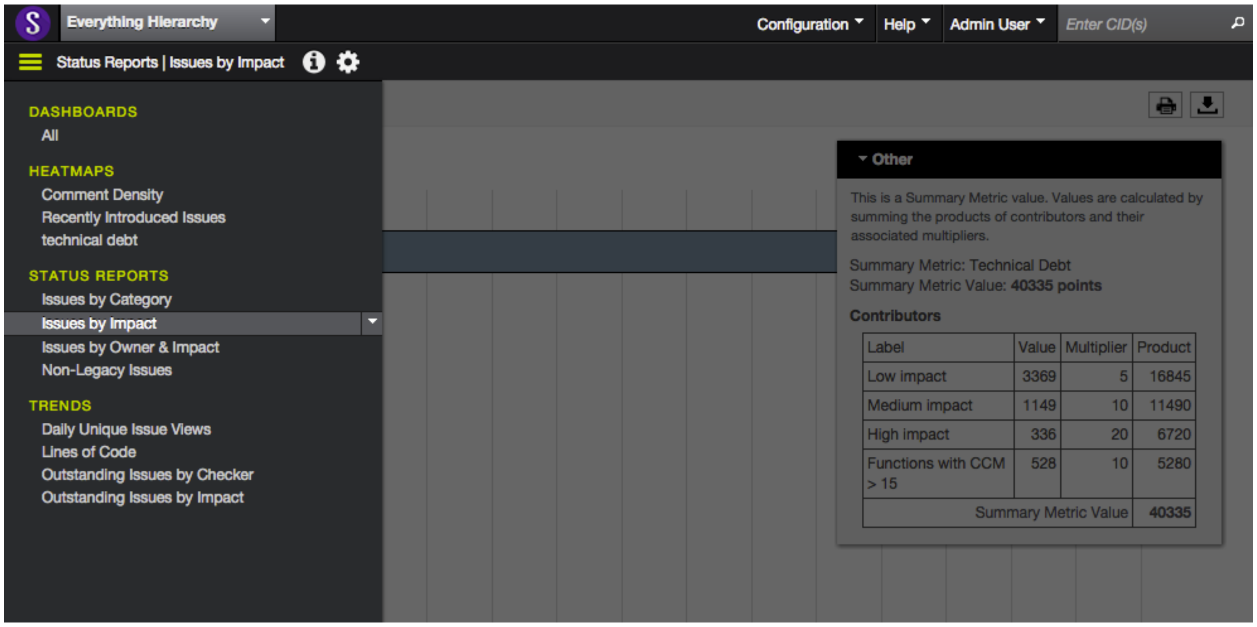
Coverity Policy Manager administrators are responsible for creating and modifying Summary metrics (see Chapter 5.4, *Creating Summary Metrics*).

4.1.9. Views

The left-side pane of a Coverity Policy Manager Hierarchy window lists views under the following view types: *Dashboards*, *Heatmaps*, *Status Reports*, and *Trends*.

When you create a heatmap, dashboard, or report, you generate a specification (a view) that is associated with your username and only visible to you (unless you share it with another user or group). The name and specification of a view appears in all hierarchies. For example, if you create a status report called Outstanding Issues, a link to the report will appear under the STATUS REPORTS heading for all the hierarchies in Coverity Policy Manager, not just in the hierarchy where you created the report.

Figure 4.1.10. Example: View Pane in Coverity Policy Manager Window



This is a Summary Metric value. Values are calculated by summing the products of contributors and their associated multipliers.

Summary Metric: Technical Debt
Summary Metric Value: **40335 points**

Contributors

Label	Value	Multiplier	Product
Low impact	3369	5	16845
Medium impact	1149	10	11490
High impact	336	20	6720
Functions with CCM > 15	528	10	5280
		Summary Metric Value	40335

At a higher level in the UI (*Projects & Hierarchies*), the Coverity Policy Manager *Hierarchies* view type (and Coverity Connect *Projects* view type) heading contain their own views (for example, the default *All Hierarchies* view). In Coverity Policy Manager, each of these high-level views supports the collection of hierarchies.

For information about using views, see Section 4.2.2, “Performing Common Coverity Policy Manager Actions”.

Chapter 4.2. Using Coverity Policy Manager Hierarchies

Table of Contents

4.2.1. Navigating through Charts 283
4.2.2. Performing Common Coverity Policy Manager Actions 283
4.2.3. Setting up Coverity Policy Manager Dashboards 286
4.2.4. Setting up Coverity Policy Manager Heatmaps 288
4.2.5. Setting up Coverity Policy Manager Status Reports 291
4.2.6. Setting up Coverity Policy Manager Trend Reports 297

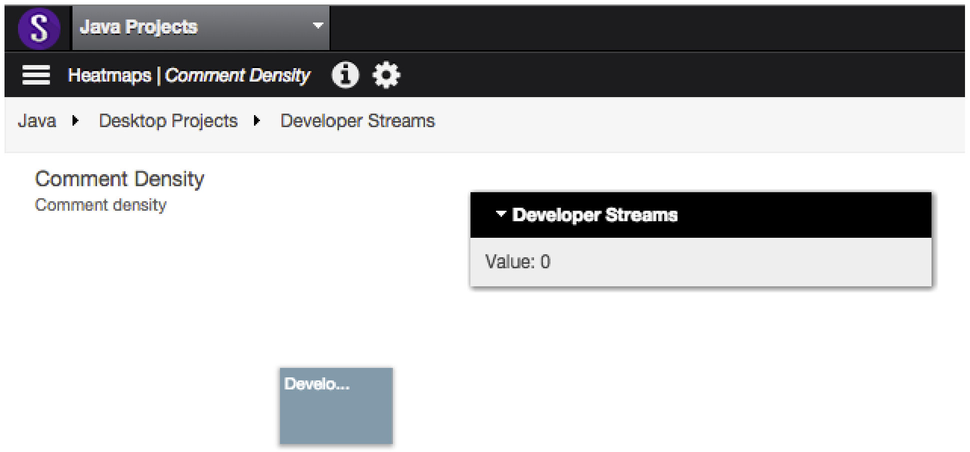
Coverity Policy Manager allows you to create, edit, share, duplicate, and delete Coverity Policy Manager heatmaps, reports, and dashboards. Creating and editing consists of specifying the metrics, filters and/or split-by and group-by options, policies (for heatmaps only), and other properties to use and display.

Navigation
: For guidance with navigation to hierarchies, see Chapter 1.3, *Shared Navigation and Main menu tools*.

4.2.1. Navigating through Charts

Navigation breadcrumbs located above each report and heatmap allow you display data that falls within the scope of any node in the hierarchy. The following figure shows how to navigate from the root node (Java) of a hierarchy, through a branch node (Desktop Projects), to leaf nodes (Developer Streams).

Figure 4.2.1. Example: Chart Navigation Breadcrumbs (Java → Desktop Projects → Developer Streams)

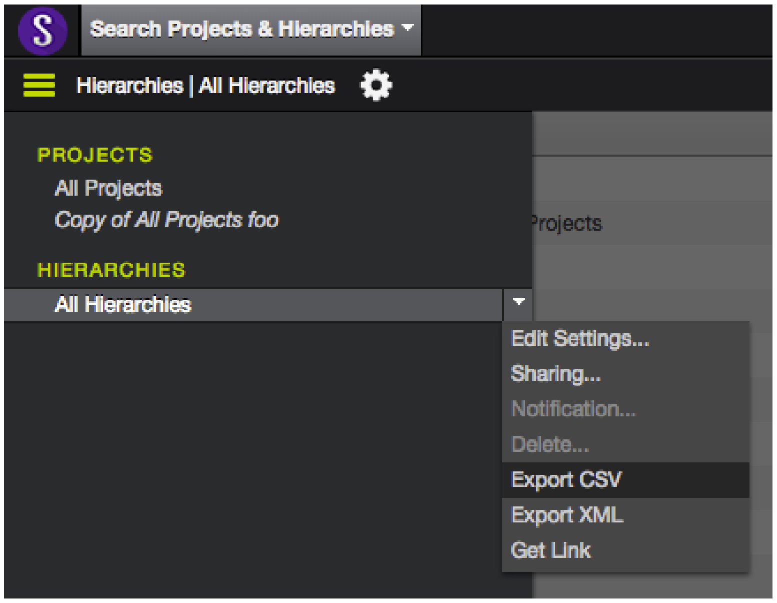


4.2.2. Performing Common Coverity Policy Manager Actions

High-level views (such as *All Hierarchies*) provide a menu that you can use to perform the same sort of actions that Coverity Connect views support (see Section 2.4.2, “View Management” for Coverity

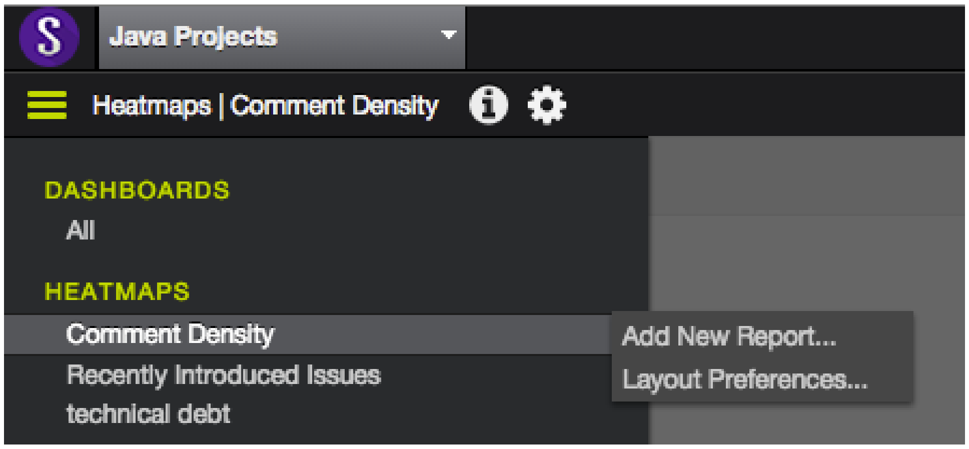
Connect). Though the particulars (such as filters and column names) differ between Coverity Policy Manager and Coverity Connect, the functionality of each menu item is largely the same.

Figure 4.2.2. Example: High-level View Menu



Add New Report...

Figure 4.2.3. Example: Add Report



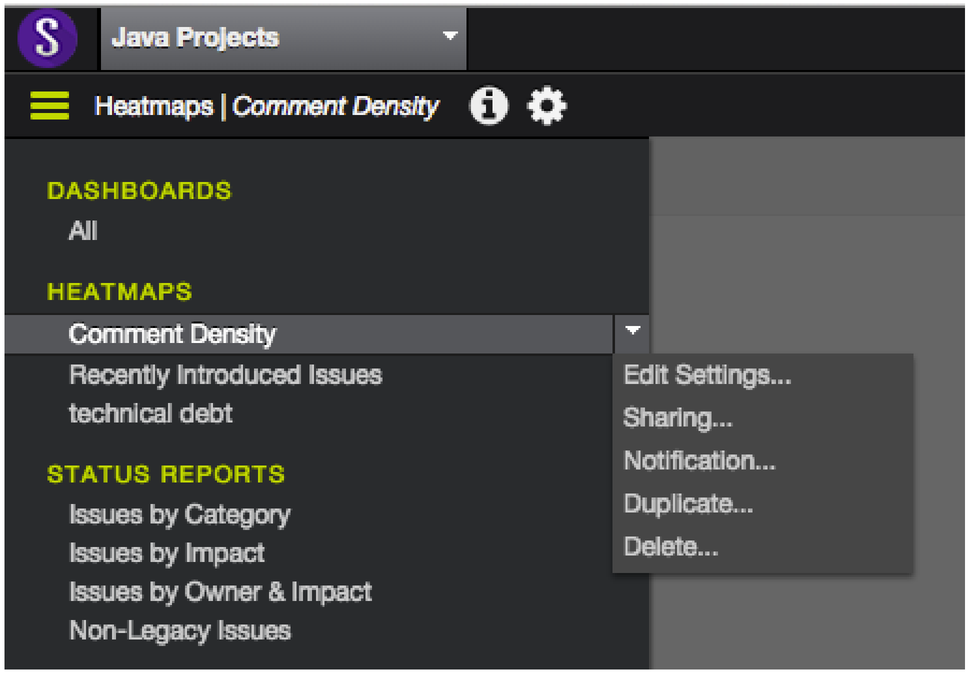
This option allows you to generate a new heatmap, report, or dashboard. You can find it by clicking the section title in the left-side Hierarchy view pane (for example, HEATMAPS or DASHBOARDS), then clicking the down arrow on the right side of the item.

Note that you can also create a new report by selecting the **Duplicate** button in the Action Menu (see Figure 4.2.4, “Example: Action Menu”.)

At the level of a particular hierarchy (for example, the Java Projects hierarchy), you can use Coverity Policy Manager heatmap, report, and dashboard views to perform the actions described in Table 4.2.1, “Common Actions”. The following features support these actions:

Action Menu

Figure 4.2.4. Example: Action Menu



This menu supports common actions. You can open this menu by mousing over any heatmap, report, or dashboard listed in the left-side pane and clicking the down arrow on the right side of the item.

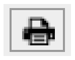
Table 4.2.1. Common Actions

Action	Description
Create, Edit Settings	To set up Coverity Policy Manager dashboards, reports, and heatmaps, see the following: • Section 4.2.3, “Setting up Coverity Policy Manager Dashboards”. • Section 4.2.4, “Setting up Coverity Policy Manager Heatmaps”. • Section 4.2.5, “Setting up Coverity Policy Manager Status Reports”. • Section 4.2.6, “Setting up Coverity Policy Manager Trend Reports”
Share	Selecting users and/or groups with which to share your view of a Coverity Policy Manager dashboard, report, or heatmap.
Duplicate	Making a copy of a Coverity Policy Manager dashboard, report, or heatmap.

Action	Description
Delete	Deleting a Coverity Policy Manager dashboard, report, or heatmap. Note that this action deletes the item from all hierarchies, not just the hierarchy from which you are deleting it. See Views.

You can also print a dashboard, heatmap, or report to file and print out a report. You use the printer icon in the chart for this purpose.

Figure 4.2.5. Printer Icon

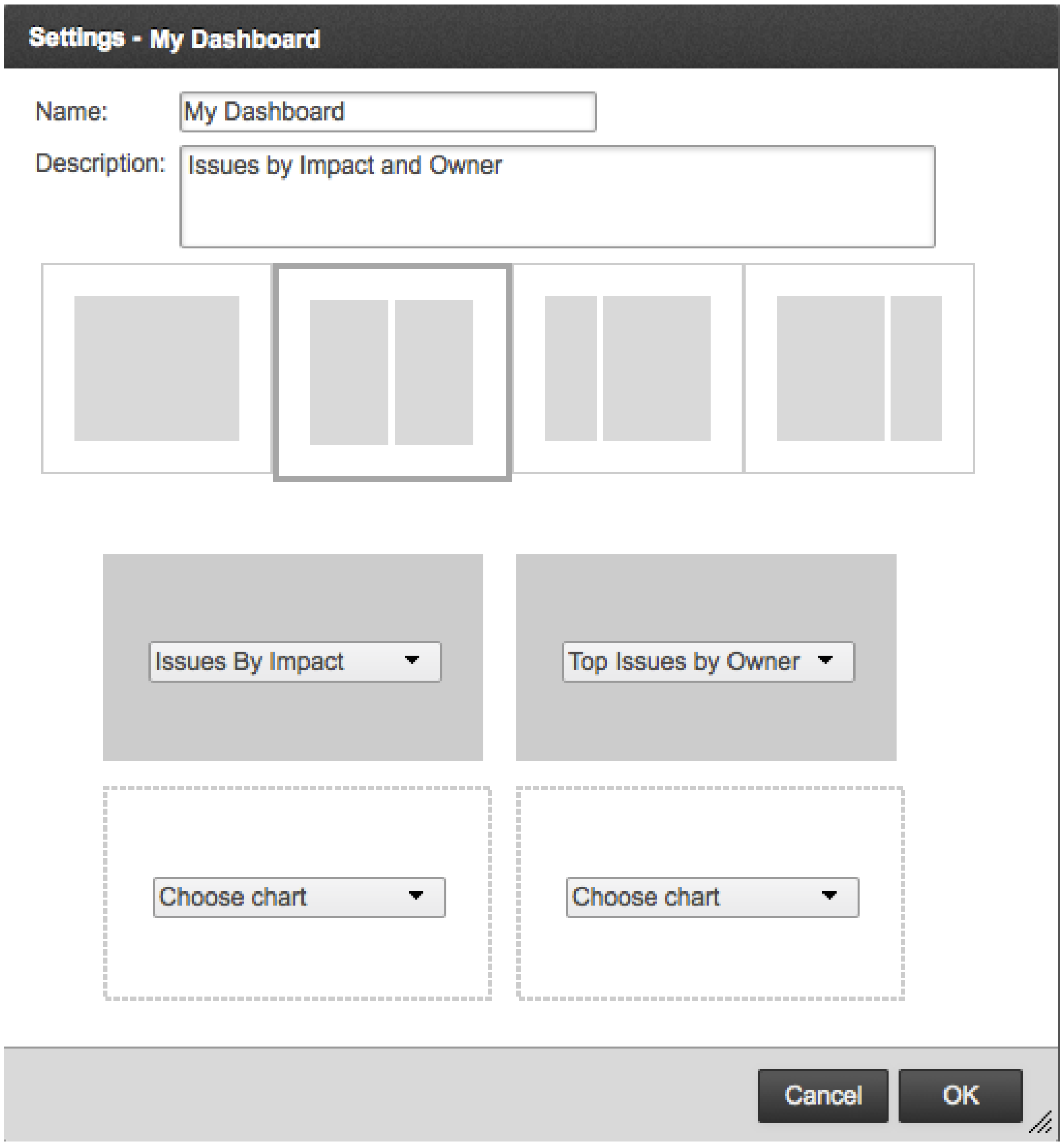


For an example of the printer icon in a chart, see Figure 4.1.2, “Example: Status Report”.

4.2.3. Setting up Coverity Policy Manager Dashboards

You can create, edit, share, duplicate, and delete one or more dashboards.

Figure 4.2.6. Example: Dashboard Edit Settings Window



Name: My Dashboard

Description: Issues by Impact and Owner

Issues By Impact

Top Issues by Owner

Choose chart

Choose chart

To create a new dashboard, click the main *DASHBOARDS* label and from the drop-down list, choose *Add New Report*. To add items to the new dashboard, click the gear-shaped *Properties* icon.

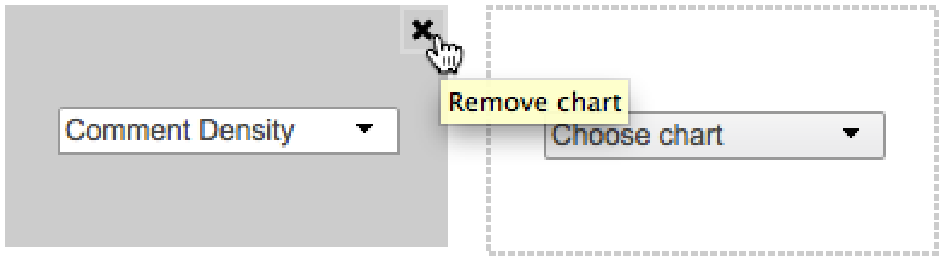
Table 4.2.2. Dashboard Properties

Property	Description
Name	Unique name for a Coverity Policy Manager dashboard.
Description	Description of the dashboard.

Property	Description
Layout	Layout in which to display columns of selected heatmaps or reports. The window provides multiple layout options.

You can delete a chart from the dashboard by clicking it in the Edit Settings window, then clicking the *Remove chart* button.

Figure 4.2.7. Example: Dashboard Chart Deletion

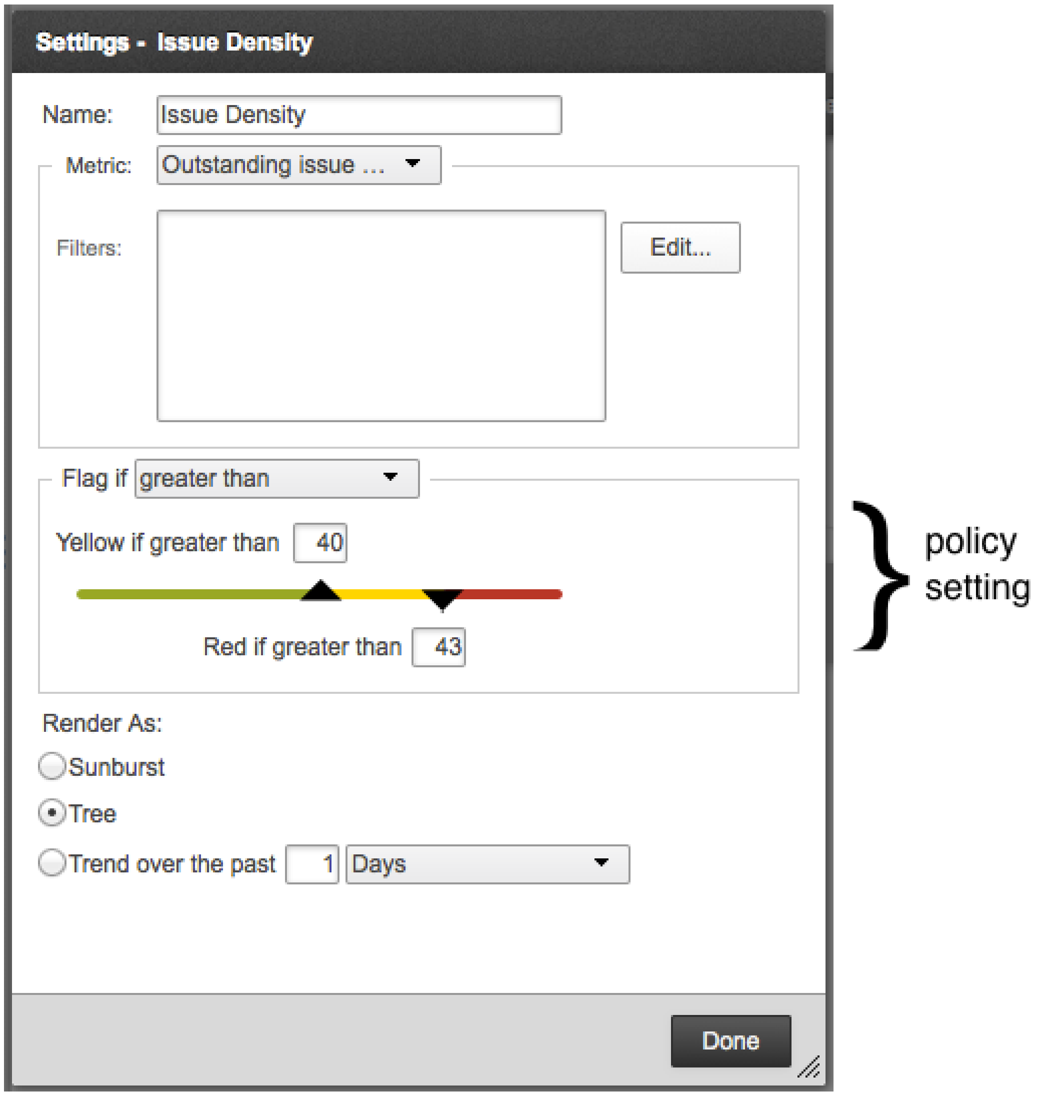


For additional information about creating, editing, sharing, duplicating, and deleting dashboards, see Section 4.2.2, “Performing Common Coverity Policy Manager Actions”.

4.2.4. Setting up Coverity Policy Manager Heatmaps

You can create, edit, share, duplicate, and delete one or more heatmaps. You can also save a heatmap to a file and print out hard copies of the file (see Figure 4.2.5, “Printer Icon”).

Figure 4.2.8. Example: Heatmap Edit Settings Window



Settings - Issue Density

Name: Issue Density

Metric: Outstanding issue ...

Filters:

Flag if greater than

Yellow if greater than 40

Red if greater than 43

Render As:

Sunburst

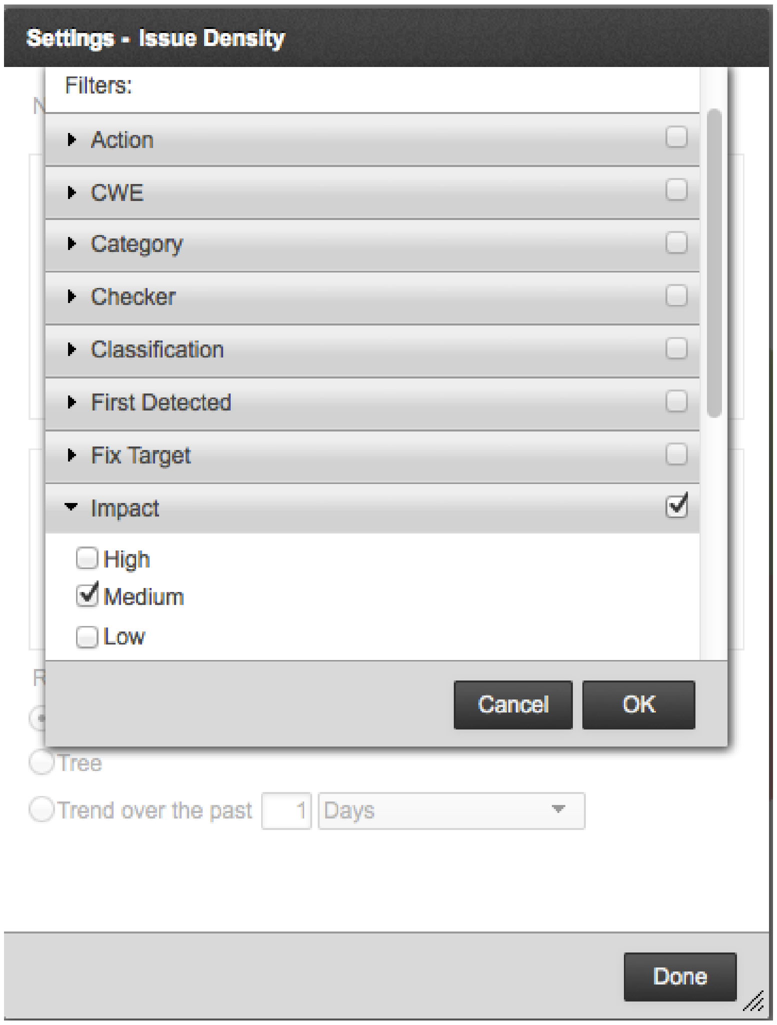
[✓ Tree]

Trend over the past 1 Days

The example shows a heatmap specification in an Edit Settings window. This heatmap sets an issue density policy where a density between 40 and 43 issues per 1000 lines of code (1 KLOC) is at risk of violating the policy (and appears in yellow on the heatmap), and a density over 43 issues per KLOC violates the policy (and appears in red on the heatmap). A density below 40 issues per KLOC meets the specified policy criteria (and appears in green on the heatmap). For an example of a heatmap with this specification, see Figure 4.1.5, “Example: Tree Map”.

Table 4.2.3. Heatmap Properties

Property	**Description**
Name	Unique name for a Coverity Policy Manager heatmap.

Property	Description
Metric	Metric to use for data in the heatmap. For an example, see the metric in Figure 4.2.8, “Example: Heatmap Edit Settings Window”.
Filter	One or more filters on the data to display in the heatmap. **Figure 4.2.9. Example: Filter Edit Settings Window**  Settings - Issue Density Filters: Action CWE Category Checker Classification First Detected Fix Target Impact High Medium Low Cancel OK Tree Trend over the past 1 Days Done To open the Edit window for filters, you click the **Edit** button in the Filters area (see Figure 4.2.8, “Example: Heatmap Edit Settings Window” for an example of this button). **Note** Summary Metrics do not provide filters.

Property	**Description**
Policy	Policy that applies to the data in your heatmap. For an example, see the policy setting in Figure 4.2.8, “Example: Heatmap Edit Settings Window”.
Render As	Display option for a heatmap: Sunburst, Tree, or Trend (Banded Trend map). For the Banded Trend map, you can specify a number of days, weeks, or months over which to plot data. The map displays a data point for each day in the specified period.

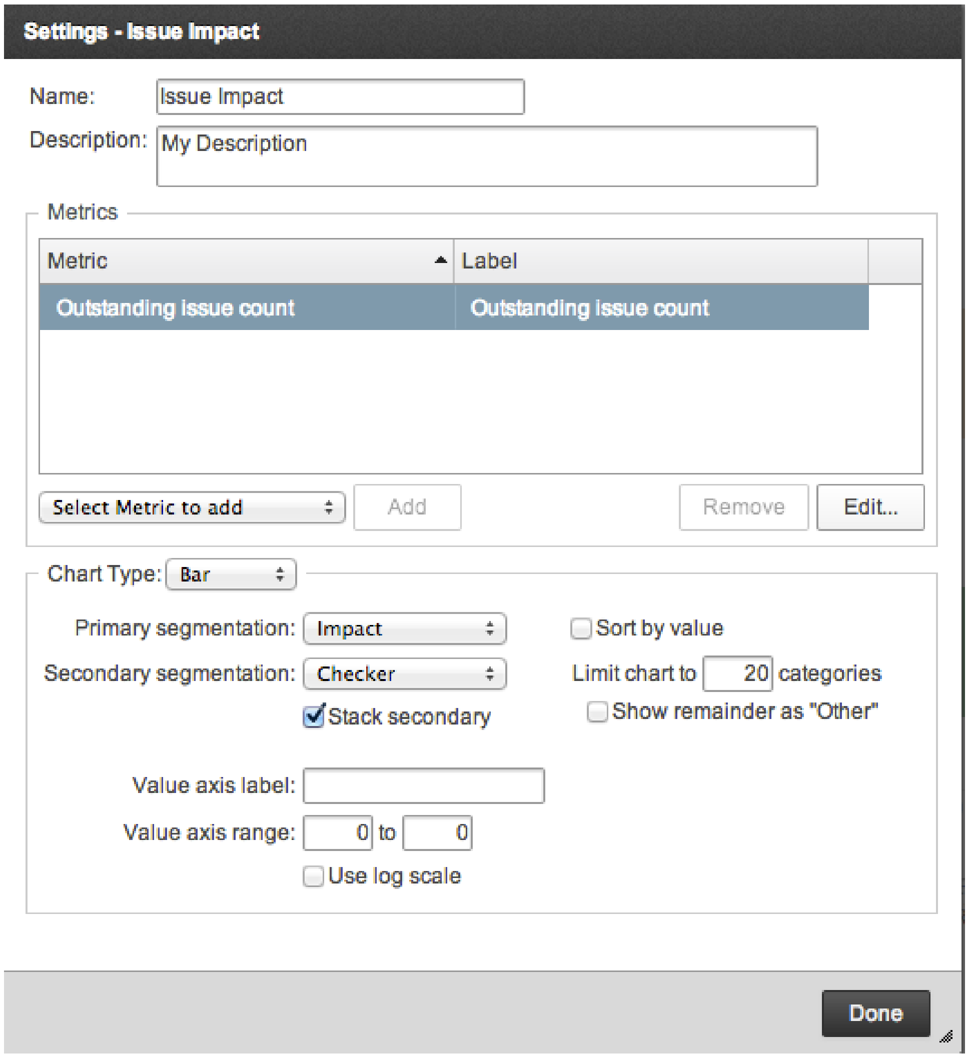
For additional information about creating, editing, sharing, duplicating, and deleting heatmaps, see Section 4.2.2, “Performing Common Coverity Policy Manager Actions”.

4.2.5. Setting up Coverity Policy Manager Status Reports

You can create, edit, share, duplicate, and delete status reports. You can also save a status report to file and print out hard copies of the file (see Figure 4.2.5, “Printer Icon”).

The Edit Settings window in the following figure specifies a single metric for a status report.

Figure 4.2.10. Example: Status Report Edit Settings Window (Single Metric)



Settings - Issue Impact

Name: Issue Impact

Description: My Description

Metrics

Metric	Label
Outstanding issue count	Outstanding issue count

Chart Type: Bar

Primary segmentation: Impact

Secondary segmentation: Checker

[✓] Stack secondary

[ ] Sort by value

Limit chart to 20 categories

[ ] Show remainder as "Other"

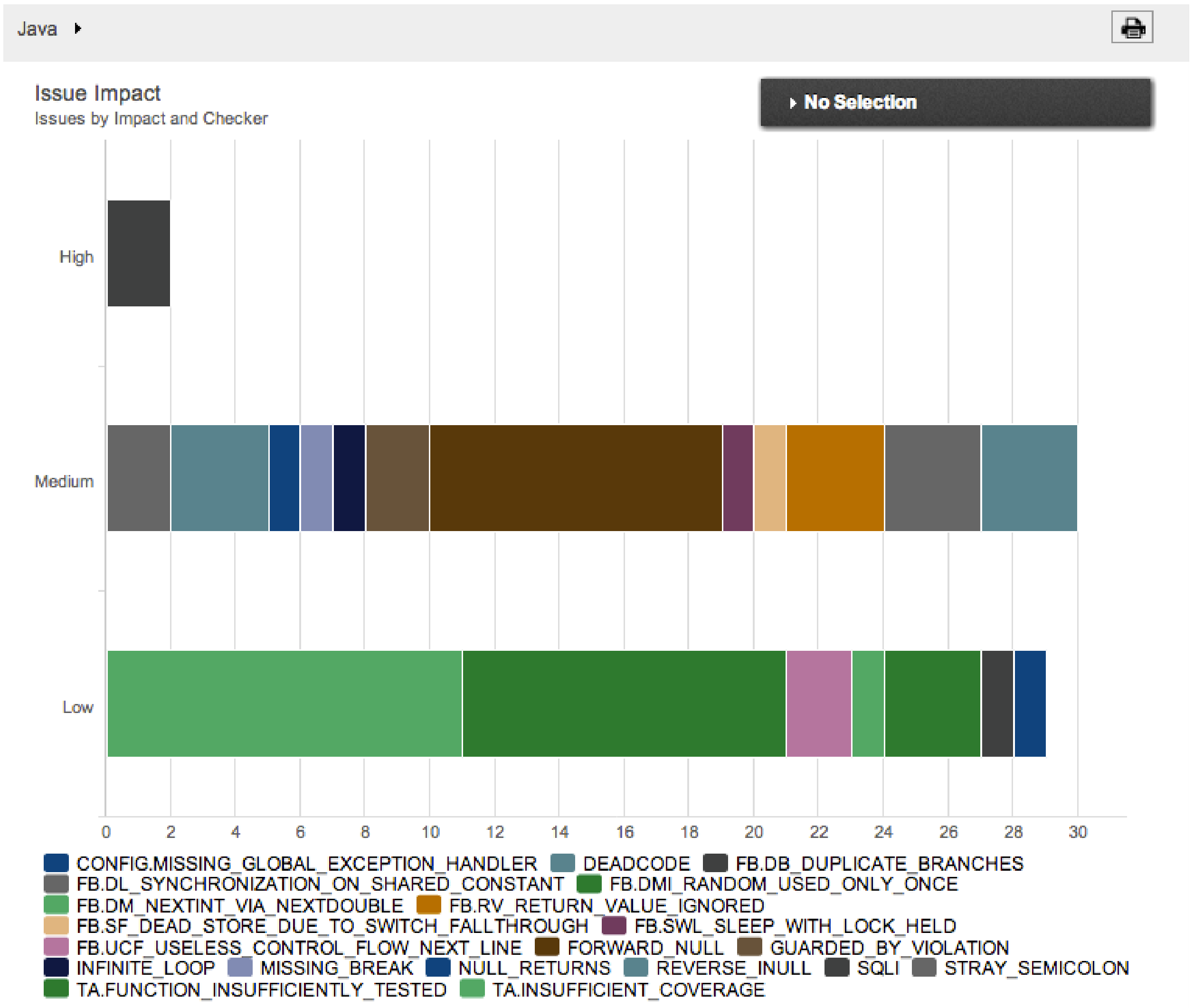
Value axis label:

Value axis range: 0 to 0

[ ] Use log scale

The chart that results from the preceding example:

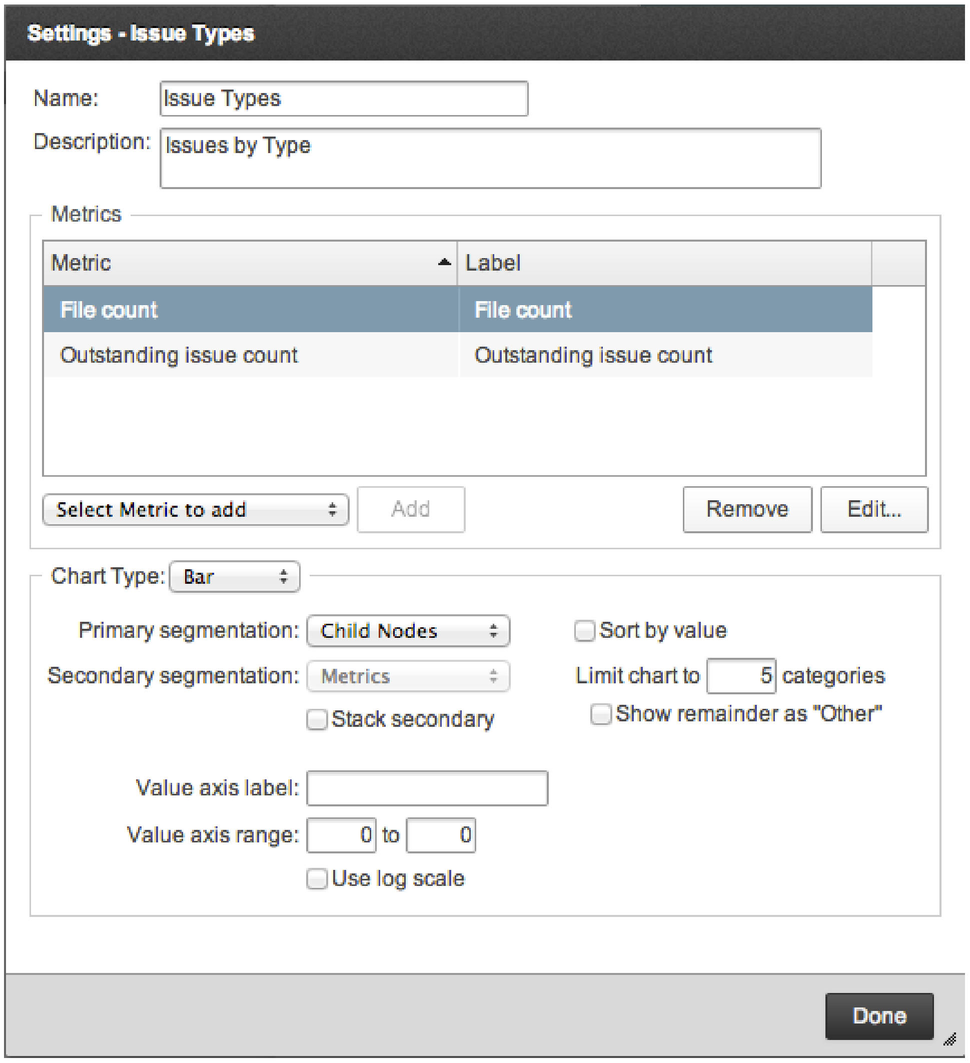
Figure 4.2.11. Example: Resulting Status Report (Single Metric)



Notice that in the report above, the primary segmentation of the issues is classification (Unclassified, Pending, Bug) and the secondary segmentation is based on the checkers that found the issues. The checkers are listed at the bottom of the report.

The Edit Settings window in the following figure specifies two metrics for a status report. Note that when two metrics are being compared, the secondary segmentation is automatically set to *Metrics*.

Figure 4.2.12. Example: Status Report Edit Settings Window (Two Metrics)



Settings - Issue Types

Name: Issue Types

Description: Issues by Type

Metrics

Metric	Label
File count	File count
Outstanding issue count	Outstanding issue count

Chart Type: Bar

Primary segmentation: Child Nodes

Secondary segmentation: Metrics

☐ Stack secondary

☐ Sort by value

Limit chart to 5 categories

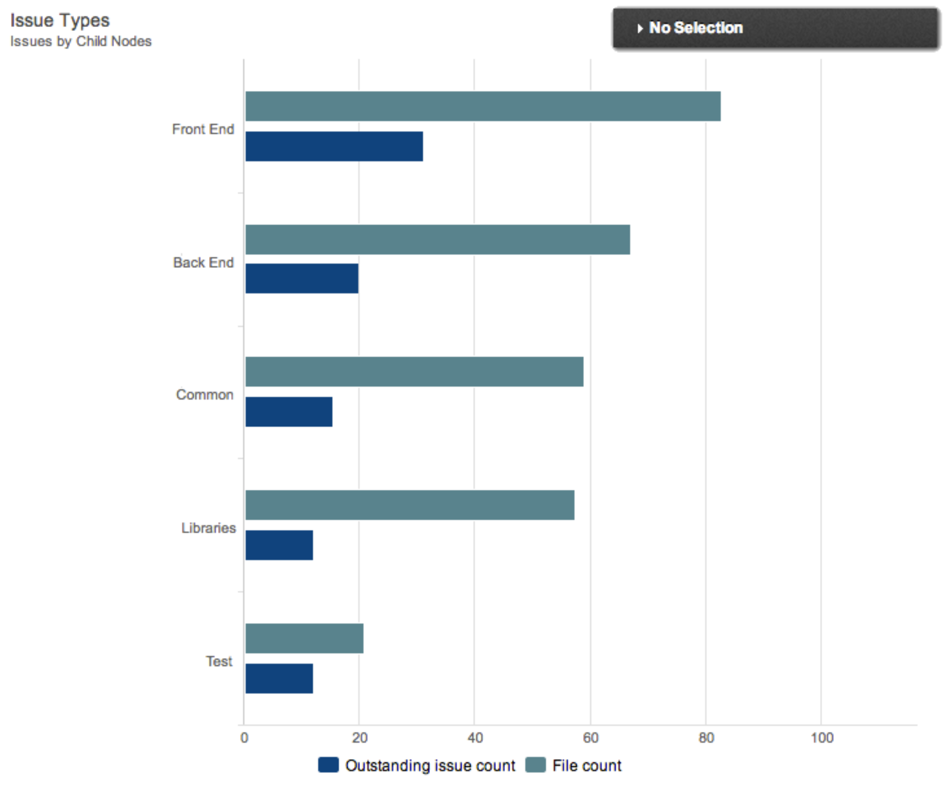
☐ Show remainder as "Other"

Value axis label:

Value axis range: 0 to 0

☐ Use log scale

Figure 4.2.13. Example: Resulting Status Report (Two Metrics)



The following table describes the properties that you can specify in the Edit Settings window for a status report.

Table 4.2.4. Status Report Properties

Property	**Description**
Name	Unique name for the report. Available to all chart types.
Description	Description for the report. Available to all chart types.
Metrics	One or more metrics to use in the report. The metric specifies the type of data to use in the data. For example, the chart in Figure 4.2.11, “Example: Resulting Status Report (Single Metric)” reports issues by their classification, and the chart in Figure 4.1.2, “Example: Status Report” reports outstanding and resolved issues.

Property	**Description**
	You can use the *Edit* button to set one or more filters on data for a selected metric. For an example, see Figure 4.2.9, "Example: Filter Edit Settings Window". Available to all chart types.
Chart Type	Chart display options: Bar (data graphed in horizontal bars), Column (data graphed in vertical columns), Pie (data divided into a standard pie chart), table (data listed in a standard table format).
[Primary] Segmentation	Primary division of the data found through the specified metric. Available to Bar, Column, and Pie charts only. See also, Secondary Segmentation.
Secondary Segmentation	Secondary division of the data found through the specified metric. If multiple metrics are selected, this option will be disabled, and the chart will divide data by metric. Available to Bar and Column charts only. See also, [Primary] Segmentation.
Rows	Primary division of the data found through the specified metric, listed as rows in a table. Available to Table charts only. See also, Columns.
Columns	Secondary division of the data found through the specified metric, represented as the columns in a table. Available to Table charts only. See also, Rows.
Stack sections	For reports that specify a *Split By* property, this property stacks the resulting data segments together (see Figure 4.2.11, "Example: Resulting Status Report (Single Metric)"), instead of presenting them side-by-side. . Available to Bar and Column charts only.
Sort by Value	Option to sort data points from highest to lowest. Available to all chart types.
Limit chart to [specified_number_of] categories	Option to limit the number of *Group By* categories to display in the chart. Typically used for items such as users or checkers, where it might be impractical to display all of them. Defaults to the maximum of 40. See also, *Show remainder as "Other"*. Available to all chart types.

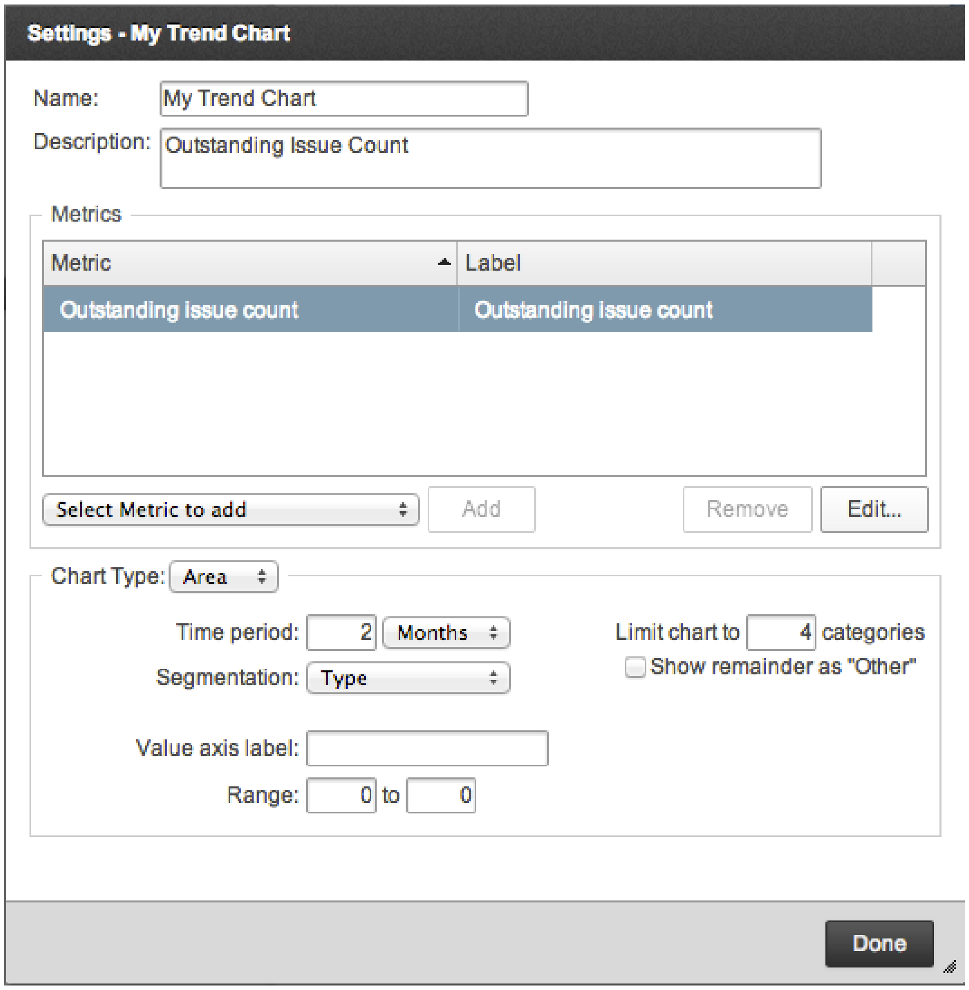
Property	**Description**
Show remainder as "Other"	When using Limit chart to [number_of] categories, you can select this option to generate a single *Other* data point that aggregates values from all categories not otherwise displayed. Available to all chart types.
Value axis label	Name that you can prepend to the subtitle of the report. By default, the subtitle displays the values of the *Group By* and *Split By* fields. For example, Figure 4.2.10, “Example: Status Report Edit Settings Window (Single Metric)” specifies *Issue Count* as the value of this property, so the subtitle displays the following: *Issue Count by Classification and Checker*. Available to Bar and Column charts only.
Value axis range	Numeric range of values to display in the chart. If you set the values to `0`, Coverity Policy Manager will automatically determine the range to use. If the actual values exceed the specified range, the chart will crop the data. Available to Bar and Column charts only.
Log Scale	Option to scale data points logarithmically. Typically used to display of large-scale differences between the data points in a practical way. Available to Bar and Column charts only.

For additional information about creating, editing, sharing, duplicating, and deleting status reports, see Section 4.2.2, “Performing Common Coverity Policy Manager Actions”.

4.2.6. Setting up Coverity Policy Manager Trend Reports

You can create, edit, share, duplicate, and delete Trend reports.

Figure 4.2.14. Example: Trend Report Edit Settings Window



Settings - My Trend Chart

Name: My Trend Chart

Description: Outstanding Issue Count

Metrics

Metric	Label
Outstanding issue count	Outstanding issue count

Select Metric to add | Add | Remove | Edit...

Chart Type: Area

Time period: 2 Months

Segmentation: Type

Limit chart to 4 categories

☐ Show remainder as "Other"

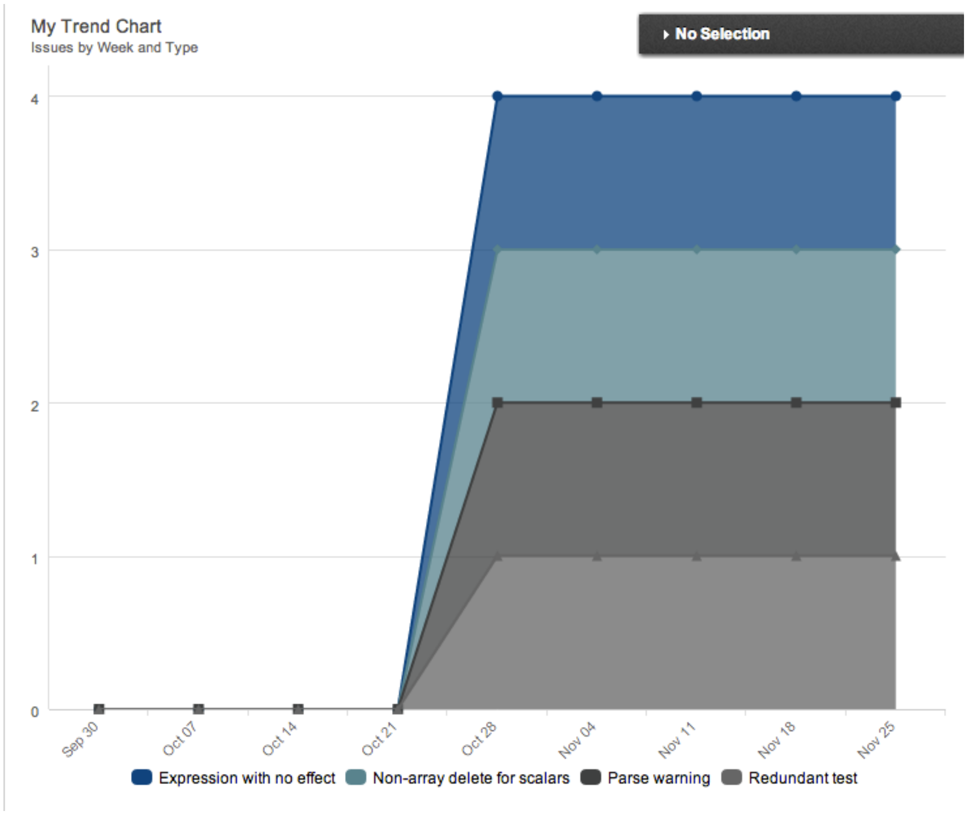
Value axis label:

Range: 0 to 0

Done

The chart that results from the preceding example:

Figure 4.2.15. Example: Resulting Trend Report



The following table describes the properties that you can specify in the Edit Settings window for a trend report.

Table 4.2.5. Trend Report Properties

Property	Description
Name	Unique name for the report. Available to all chart types.
Description	Description for the report. Available to all chart types.
Metrics	One or more metrics to use in the report. The metric supplies the report data. For example, see the metric in Figure 4.2.14, “Example: Trend Report Edit Settings Window”. You can use the *Edit* button to set one or more filters on data for a selected metric. For an example, see Figure 4.2.9, “Example: Filter Edit Settings Window”.

Property	**Description**
	Available to all chart types.
Chart Type	Trend report display options: Area (data graphed in a standard area chart), Line (data points on a standard line graph), Table (data listed in a standard table format).
Segmentation	Filter used to subdivide the data points or segments. For example, the trend report in Figure 4.2.15, “Example: Resulting Trend Report” subdivides the data points by triage *Action* values (Undecided, Fix Required, Fix Submitted, Modeling Required, and Ignore). Note that trend reports automatically group their data by day, so they *do not* provide a configurable Secondary Segmentation field (which is available to status reports). Available to all chart types.
Limit chart to [specified_number_of] categories	Option to limit the number of *Split By* categories to display in the chart. Typically used for items such as users or checkers, where it might be impractical to display all of them. Defaults to the maximum of 40. See also, *Show remainder as "Other"*. Available to all chart types.
Show remainder as "Other"	When using Limit chart to [number_of] categories, you can select this option to generate a single *Other* data point that aggregates values from all categories not otherwise displayed. Available to all chart types.
Time Period	Number of days, weeks, or months over which to track changes to the data. Note that each data point in the trend report will represent a day in the specified time period. Available to all chart types.
Range	Numeric range of data values to display in the charts. If you set the values to `0`, Coverity Policy Manager will automatically determine the range to use. If the actual values exceed the specified range, the chart will crop the data. Available to Bar and Column charts only.

For additional information about creating, editing, sharing, duplicating, and deleting trend reports, see Section 4.2.2, “Performing Common Coverity Policy Manager Actions”.

Chapter 4.3. Coverity Policy Manager Metrics

Table of Contents

4.3.1. Metrics: User Activity 301
4.3.2. Metrics: Analysis 302
4.3.3. Metrics: Code 304
4.3.4. Summary Metrics: Technical Debt 305

The tables in the following sections describe Coverity Policy Manager metrics, list charts to which you can apply the metrics, and identify the filter and segmentation properties that you can use with the metrics.

4.3.1. Metrics: User Activity

User activity metrics track usage of the Coverity Connect UI as it pertains software issues, which are identified by a CID.

Table 4.3.1. Coverity Policy Manager User Activity Metrics

Metric	**Description**	**Available To**[a]	**Filters**[b]	**Segmentation | Primary Segmentation | Secondary Segmentation**[c]
Daily unique users	Number of unique users within the set of Daily unique issue views on a given day. For example, assume that a total of 2 different users viewed one or more CIDs on a day that the *Daily unique issue views* is 5. In this case, *Daily unique users* is 2 for that day. See also, Monthly unique users.	Trend Reports	Component, Owner, Owner Name	None, Child Nodes, Component, Owner, Owner Name
Daily unique issue views	Total number of unique issues viewed by a unique user in the Coverity Connect Triage pane on a given day. If 5 different users view the same CID on the same day, this metric counts 5 separate unique views (of that CID). If a single user views 5 different CIDs on the same day, this metric counts these actions as 5 separate unique views. If a single user views the same CID a total of 5 separate times in one day, this metric counts only 1 unique view (of this CID for this user).			

Metric	Description	Available To[a]	Filters[b]	Segmentation | Primary Segmentation | Secondary Segmentation[c]
Daily unique issue views per user	Number of Daily unique issue views divided by the *Daily unique users*. Assume the following occurs on the same day: User 1 inspects CID 11 and CID 22. User 2 inspects CID 22. In this case, this metric is 1.5 (because the *Daily unique issue views* is 3, and the *Daily unique users* is 2).			
Daily unique issue triages	Total number of unique CIDs that have been triaged by a unique user on a given day. Like Daily unique issue views, this metric counts unique CID and user combinations. If the same CID is triaged by the same user more than once on a given day, this metric counts only one triage action (for this CID), regardless of whether the user changed the values of different triage attributes.			
Monthly unique users	The number of unique users per month. For example, if for ten days there is only one unique user per day (and the rest of the days in the month have zero unique users), the monthly unique users could be anywhere betwen 1 and 10, depending on whether the same user visited on ten different days (monthly unique users = 1), or ten different users visited during that period (monthly unique users = 10), or some number between 1 and ten.			

[a] Lists of charts to which the metric is available: Heatmaps, Status Reports, and/or Trend Reports.

[b] List of filters that you can apply to the data retrieved by the metric. For descriptions of filters, see Chapter 4.4, *Coverity Policy Manager Filter and Segmentation Properties*.

[c] List of *Segmentation*, *Primary Segmentation* or *Secondary Segmentation* properties that you can apply to the chart data. For descriptions of these properties, Chapter 4.4, *Coverity Policy Manager Filter and Segmentation Properties*.

4.3.2. Metrics: Analysis

Analysis metrics provide information about the state of software issues (CIDs).

Table 4.3.2. Coverity Policy Manager Analysis Metrics

Metric	Description	Available To	Filters[b]	Segmentation | Primary Segmentation | Secondary Segmentation[c]
All issues	Total number of issues in a given node of a hierarchy.	Heatmaps, Status Reports, Trend Reports	Action, Category, Checker, Classification, Coding standards and vulnerability reports, Component, CWE, First Detected, Fix Target, Impact, Issue Kind, Legacy, Owner, Owner Name, Severity, Status, Type Also lists any picklist-type custom attributes.[a]	None, Child Nodes, Action, CWE, Category, Checker, Classification, Coding standards and vulnerability reports,Component, Fix Target, Impact, Legacy, Owner, Owner Name, Severity, Status, Type Also lists any picklist-type custom attributes.
Outstanding issue count	Number of CIDs that are classified in Coverity Connect as *Unclassified*, *Pending*, *Bug*, or for Test Advisor, *Untested*.			
Daily issues introduced	The number of CIDs introduced to a particular stream for the first time in a day.	Trend Reports		
Daily issues dismissed	Number of CIDs that were dismissed on a given day.			
Daily issues fixed	Number of CIDs that were fixed on a given day.			
Outstanding issue density	Number of outstanding issues per 1000 lines of code.	Heatmaps, Status Reports, Trend Reports		Node, Component
Daily commit count	Number of snapshots committed to a given node of a hierarchy on a given day.	Trend Reports	Description, Target, Version	Stream, Node
Daily issues committed	Number of issues committed to a given node of a hierarchy on a given day. Returns a raw number of issues, with duplicate issues counted separately.			
Daily lines of code committed	Total number of lines of code committed to a given node of a hierarchy on a given day. Returns a raw number of lines, with duplicate lines of code counted separately.			

Metric	Description	Available To	Filters[b]	Segmentation | Primary Segmentation | Secondary Segmentation[c]
Daily files committed	Number of source code files committed to a given node of a hierarchy on a given day. Returns a raw number of files, with duplicate files counted separately.			

[a]Picklist attributes provide a preconfigured set of values.

4.3.3. Metrics: Code

Code metrics provide information about the source code used in your analyses.

Table 4.3.3. Coverity Policy Manager Code Metrics

Metric	Description	Available To	Filters[b]	Segmentation | Primary Segmentation | Secondary Segmentation[c]
Lines of code	Number of lines of code in the source code files within the scope of a given node of a hierarchy. Does not include lines fully composed of comments or blank lines in the source code. However, any line that includes both code and a comment counts as a line of code.	Heatmaps, Status Reports, Trend Reports	Component	None, Child Nodes, Component, Detected In[a]
Comment lines	Number of lines of comments in the source code files within the scope of a given node of a hierarchy. Does not include comments that occur on lines that also contain source code.			
Comment density	Comment lines as a proportion of the total lines of code in the source code files. The density is equal to number of comment lines divided by the sum of comment lines and lines of source code. This metric does not include blank lines in the density calculation.			

Metric	**Description**	**Available To**	**Filters**[b]	**Segmentation | Primary Segmentation | Secondary Segmentation**[c]
Policy coverage	Test Advisor metric for the percentage that is equal to the count of lines covered by tests divided by coverable lines.			
Policy covered lines	Test Advisor measure of the number of lines covered by tests.			
Policy uncovered lines	Test Advisor measure of the number of lines not covered by tests.			
Raw coverage	Test Advisor metric for the percentage that is equal to the raw covered lines divided by raw coverable lines.			
Raw covered lines	Test Advisor measure of the number of lines covered by tests, as reported by the coverage tool.			
Raw uncovered lines	Test Advisor measure of the number of lines not covered by tests, as reported by the coverage tool.			
File count	Number of source code files within the scope of a given node of a hierarchy. Note that files with the same name and file path count as a single file.		Code Lines (LOC), Issue Density, Oustanding, Policy Coverage, Raw Coverage	
Function count	Number of functions in the source code files within the scope of a given node of a hierarchy. Returns a raw number of functions, with duplicate functions counted separately.		CCM	None, Child Nodes, Component, Cyclomatic Complexity

[a]See Detected In.

4.3.4. Summary Metrics: Technical Debt

The *Technical Debt* summary metric is built into Coverity Policy Manager. Administrators can create additional summary metrics, as needed.

You will see a label when adding a Summary metric to a heatmap, status report, or trend report. Note that Summary metrics provide no filters. Instead, their contributor metrics can be filtered.

Table 4.3.4. Coverity Policy Manager Code Metrics

Summary Metric	**Label**	**Available To**	**Filters**[b]	**Primary Segmentation | Secondary Segmentation** [c]
Technical Debt	Technical Debt	Heatmaps, Status Reports, Trend Reports	(No Filters)	None, Child Nodes, Component, Contributor Only Summary metrics offer the Contributor segmentation property. It is never available to reports that include other metrics.

For more detail on this topic, see Section 4.1.8, “Summary Metrics”.

Chapter 4.4. Coverity Policy Manager Filter and Segmentation Properties

This section describes the filters, split-by values, and group-by values that you can use when specifying a Coverity Policy Manager chart. The availability of these items varies by metric and chart type.

A group-by value sets the primary division of the data in a chart. A split-by value sets a secondary division of data in a map. For an example, see Figure 4.2.10, “Example: Status Report Edit Settings Window (Single Metric)”.

Table 4.4.1. Coverity Policy Manager Filters

Filter or Segmentation Property	**Filter**[b]	**Segmentation**[c]	**Description**
Action	Yes	Yes	Triage attribute used to specify the action to take with regard to a software issue.
Category	Yes	Yes	Category of software issue found by a checker. Examples include *Memory - corruptions*, *Resource leaks*, *Null pointer dereferences*.
Checker	Yes	Yes	Name of a Coverity checker.
Coding standards and vulnerability reports	Yes	Yes	Coding standards and vulnerability reports, such as MISRA-C and OWASP Top Ten.
Component	Yes	Yes	Coverity Connect component. If a hierarchy configuration component filter is enabled, then the **Component** acts as a secondary filter.
Contributors	No	Yes	Filtered and weighted metrics that make up a summary metric. If a chart includes a summary metric and one or more regular (non-summary) metrics, this segmentation property will not be available.
Custom Attributes	Yes	Yes	Custom picklist-type triage attributes. Such attributes contain a preconfigured list of one or more attribute values.
CWE	Yes	Yes	Common Weakness Enumeration documentation of a software issue. The Coverity Connect Triage pane displays a CWE for a software issue.
Detected In	(Not a Filter)	Yes	Name of a Coverity Connect stream in which the issue was detected.
First Detected	Yes	(Not a *-by Option)	Filter for narrowing the scope of issues detected (or not detected) in snapshots up to

Filter or Segmentation Property	**Filter**[b]	**Segmentation**[c]	**Description**
			40 days in the past. Accepts a value of zero (0) to 40.
Fix Target	Yes	Yes	Triage attribute used to set the release in which to fix an issue. See Fix Target.
Impact	Yes	Yes	Issue impact as determined by Coverity Connect: High, Medium, Low, or Audit.
Issue Kind	Yes	(Not a *-by Option)	Any of the following kinds of issues: Quality, Security, Test, or Various issue.
Legacy	Yes	Yes	Triage attribute used to indicate whether an issue is a legacy issue or not. Some companies mark as Legacy those issues that existed undetected in the code base prior to a Coverity Analysis upgrade or change to checkers used to analyze the code base.
LOC	Yes	(Not a *-by Option)	Lines of code in the source code files.
Owner	Yes	Yes	Filter for entering the username of the owner of software issues.
Owner Name	Yes	Yes	Filter for entering a glob pattern that matches the first or last name (or names) of the owner of software issues.
Severity	Yes	Yes	Triage attribute that identifies the severity of software issues. Built-in Severity values: Unspecified, Major, Minor, Moderate. Alternative custom values are possible.
Status	Yes	Yes	Triage attribute that identifies the status of software issues. Status values: New, Triaged, Dismissed, Fixed.
Type	Yes	Yes	Descriptions of software issues (sometimes called checker subcategories) found by a checker.

Part 5. Coverity Policy Manager Administration

Coverity Policy Manager administrators set up the hierarchies that are used to specify the data source and structure of Coverity Policy Manager heatmaps and charts. For your organization to use Coverity Policy Manager, it is necessary to specify at least one hierarchy and to assign the appropriate Coverity Policy Manager role to users and/or groups (see Chapter 5.5, *Coverity Policy Manager Roles and Permissions*).

Coverity Policy Manager is a companion product to Coverity Connect that shares the Coverity Connect user database and servlet container and is fully integrated into the Coverity Connect UI.

Licensing

If you need to update or import a license for Coverity Policy Manager, you need to import it through Coverity Connect. For details, see Section 3.1.1.13, “Coverity Connect license information”.

Daily data synchronization

Coverity Policy Manager synchronizes Trend Report data with the Coverity Connect database once per day, starting between midnight and 1:00 am. The time needed to complete the data synchronization depends on the amount of data being transferred.

Alternately, Status Reports are updated periodically throughout the day, waiting one hour after the end of each update to start the next one. For example, if an update takes 30 minutes to run, the Status Reports will be updated approximately every 90 minutes.

Important

In general, Policy Manager does not store data indefinitely. In the absence of other activity, daily data is kept for 40 days, weekly data is kept for 30 weeks, monthly data is kept for 24 months, and only yearly data is kept indefinitely. However, Policy Manager does not save historical data in terms of previous configurations of component map and stream settings, and if the current project configurations are changed, Policy Manager data is recomputed based on those changes.

For information about data synchronization between coordinators and subscribers, see Section 3.5.1.2.3, “Synchronizing Coverity Policy Manager data across the cluster”.

Chapter 5.1. Managing a Coverity Policy Manager Hierarchy

Table of Contents

5.1.1. Planning a Hierarchy 310
5.1.2. Configuring a Hierarchy 320

When you set up Coverity Policy Manager, it is necessary to specify at least one hierarchy.

5.1.1. Planning a Hierarchy

Before you attempt to create or modify a hierarchy, it is important to understand the needs of your Coverity Policy Manager users so that you can determine a useful way of organizing the data that the hierarchy will support. Though it is possible to create a hierarchy that simply organizes each Coverity Connect project into a separate node in the hierarchy, Coverity Policy Manager users are likely to require a different structure. As a basic goal, your hierarchy should allow Coverity Policy Manager users to see data that matters to them.

It helps to start by thinking of your development organization from the top down, rather than from the bottom up. For example, a vice president of engineering might want to see information on all products. A manager who reports to the vice president might need to check data that is aggregated from all the teams that build one of the products. A team lead primarily needs to get the status of code managed by that group. Most likely, the vice president and managers also need to be able to navigate to the areas of the code base that concern them. So it would be helpful to build one or more hierarchies that could support such views.

Coverity Policy Manager Use Case

Scenario

Assume that company has used Coverity products for several years. Twenty project managers oversee a total of sixty software development projects, some of which are separate Coverity Connect projects, while others (about half) are components of a single Coverity Connect project.

The director who is in charge of this project management team wants to see charts (Status Reports and Trend Reports) that organize data on software issues (for example, impact, count, density, and so on) by project manager:

1. You need to set up a hierarchy in which the analyzed source code files than contain issues from the company's software projects are associated with the appropriate project manager.

    a. You create a separate branch node for each project manager (for example, the node might name the project manager).

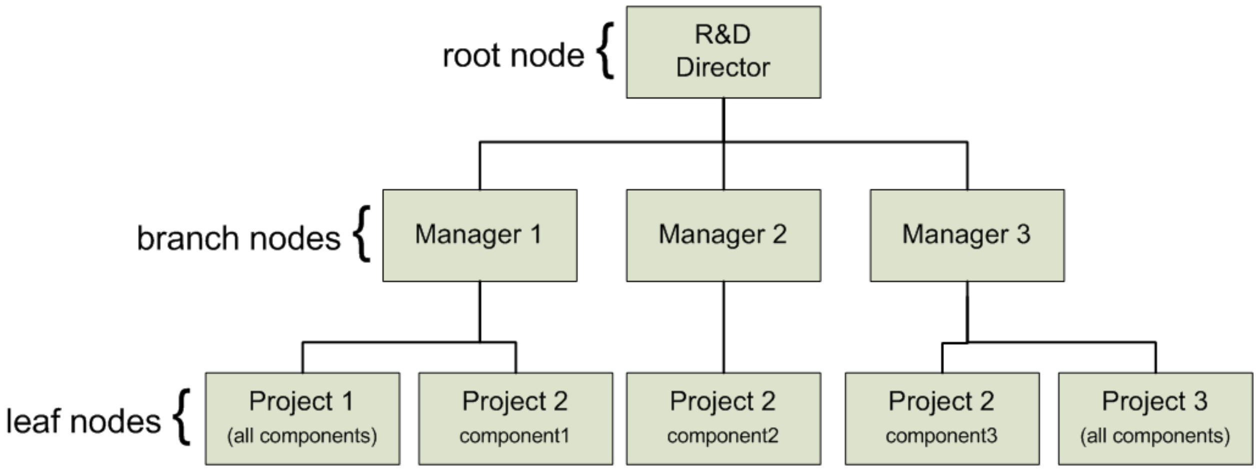
    b. For each branch that represents a separate project manager, you create one or more leaf nodes, and associate the leaves with the project or component for which each project manager is responsible.

       In the case that a given project manager is reponsible for managing all software issues in a given Coverity Connect project, you can associate the leaf with the entire project.

In the case that the project manager is responsible for managing software issues from a subset of source files from a Coverity Connect project, you can associate the leaf with a component. If the components do not exist, a Coverity Connect administrator will need to set them up for you (see Section 3.3.3, “Using components”).

The following figure represents a portion of the node tree for the hierarchy described in this step.

Figure 5.1.1. Example: Organizing a Hierarchy by Management Personnel



For guidance, see Section 5.1.2, “Configuring a Hierarchy”. For additional node tree examples, see the figures below.

2. You need to assign a Coverity Connect role to the director (or to a Coverity Connect user group to which the director belongs) that allows the director access to the Coverity Policy Manager user interface (see Chapter 5.5, *Coverity Policy Manager Roles and Permissions*).

   The director can then use the Coverity Policy Manager heatmap and charts that are supported by the hierarchy to view and compare data on issues associated with each project manager.

   The Coverity Policy Manager charts (heatmaps and reports) allow users to navigate to data that is associated with the entire set of project managers, with an individual project manager, and with individual project or component data that is associated with a leaf node. For example, the director can configure the heatmap to show policy violations on defect density. A trend chart might show changes to issues over time (for example, how many issues are introduced and resolved each day). A status report might show the number outstanding issues that are of high impact.

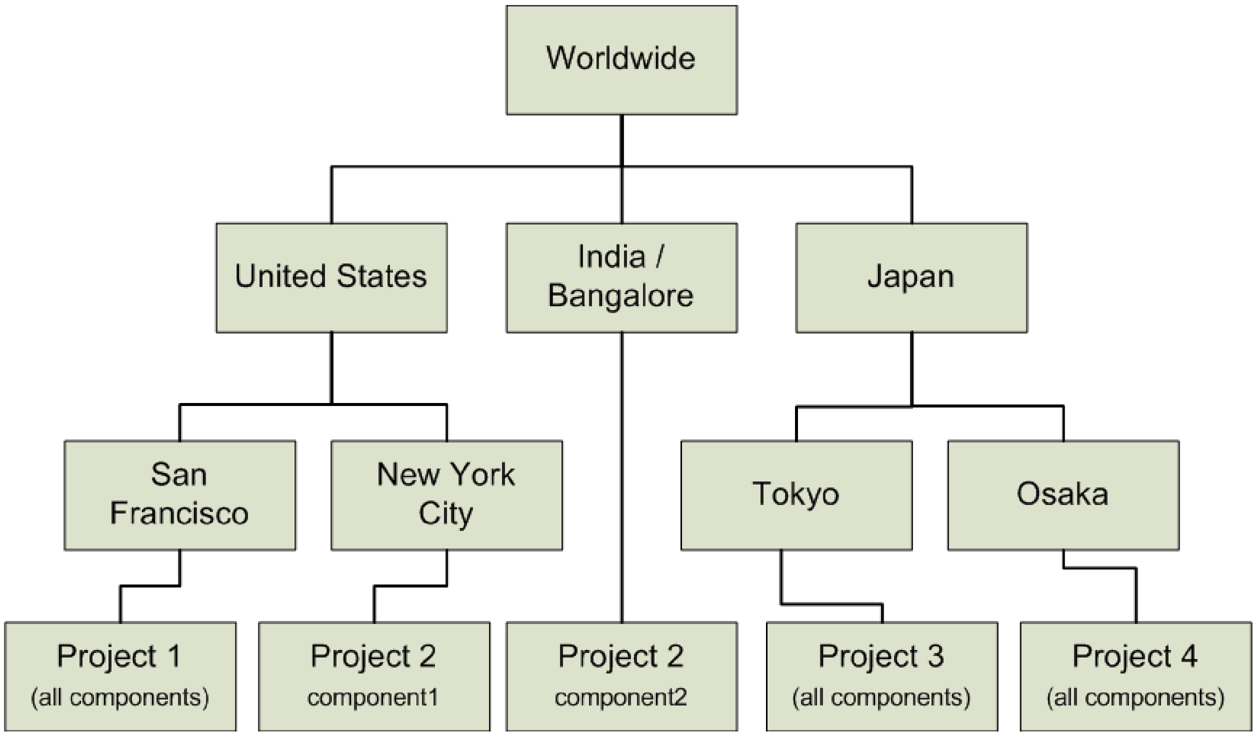
Other ways of organizing data are possible. Coverity Policy Manager users might need to see data organized in any of the following ways:

By geography

- A company might have engineering groups in the United States, India, and Japan. In such a case, the root node of the hierarchy could be called *Worldwide*, and child nodes could be called *U.S.*, *India*, *Japan*. Further subdivisions might identify the various cities where the development takes place. In this way, an executive in charge of worldwide development and managers in charge of the various regions could get the status of all code bases in aggregate or broken down by country or city.

  Figure 5.1.2, “Example: Organizing a Hierarchy by Geographical Regions” divides the hierarchy by geography. The leaf nodes of this hierarchy contain Coverity Connect projects and components that are associated with the cities.

  Figure 5.1.2. Example: Organizing a Hierarchy by Geographical Regions

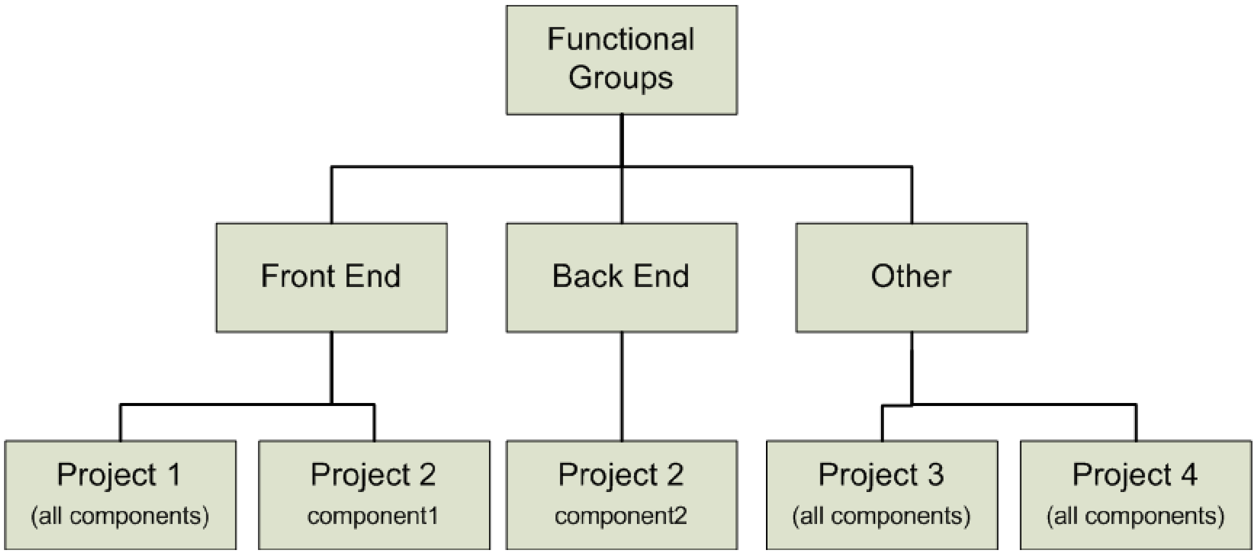


By functionality

- A company might divide its code base into functional units such as front end, back end, and so on. In such a case, the children of the root node could be named according to those functional divisions, and, if needed, their children could be named according to subdivisions of those functional units.

  Figure 5.1.3, “Example: Organizing a Hierarchy by Functionality” divides the hierarchy into functional units. The leaf nodes of this hierarchy contain Coverity Connect projects and components that are associated with them.

Figure 5.1.3. Example: Organizing a Hierarchy by Functionality

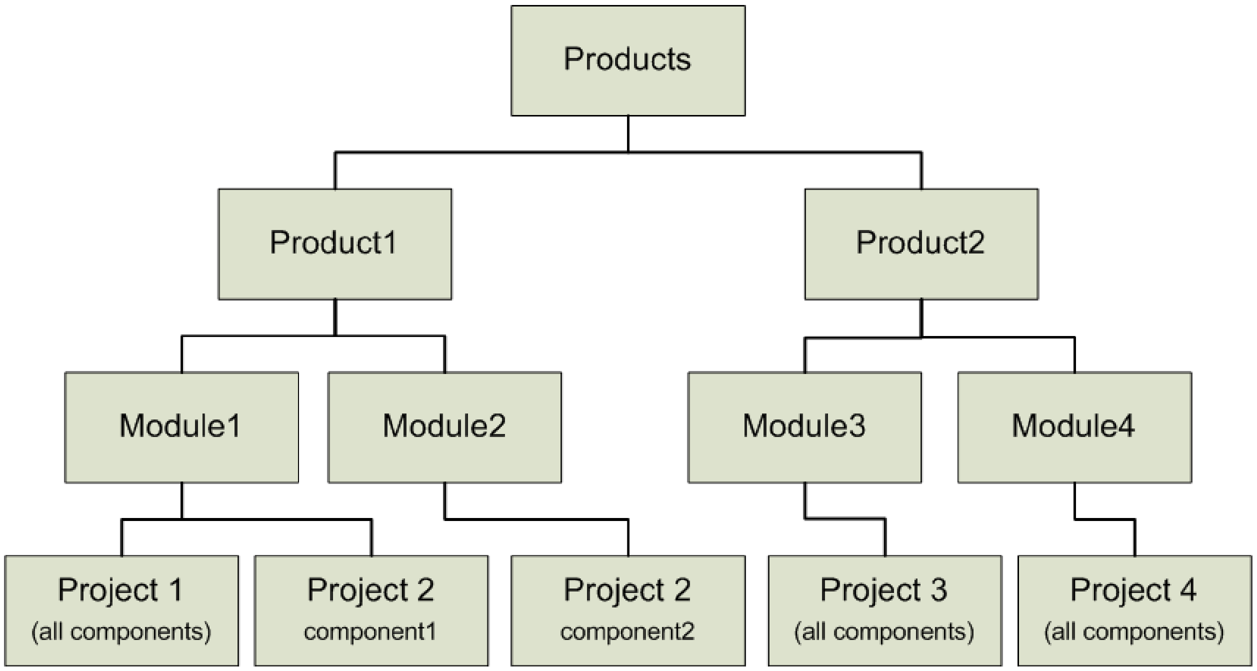


By product or project

- A company might build multiple products. In such a case, the children of the root node could be named according to product name then subdivided further by product modules and, if needed, by submodules.

  Figure 5.1.4, “Example: Organizing a Hierarchy by Product” divides the hierarchy by product. The leaf nodes of this hierarchy contain Coverity Connect projects and components that are associated with the product modules.

Figure 5.1.4. Example: Organizing a Hierarchy by Product

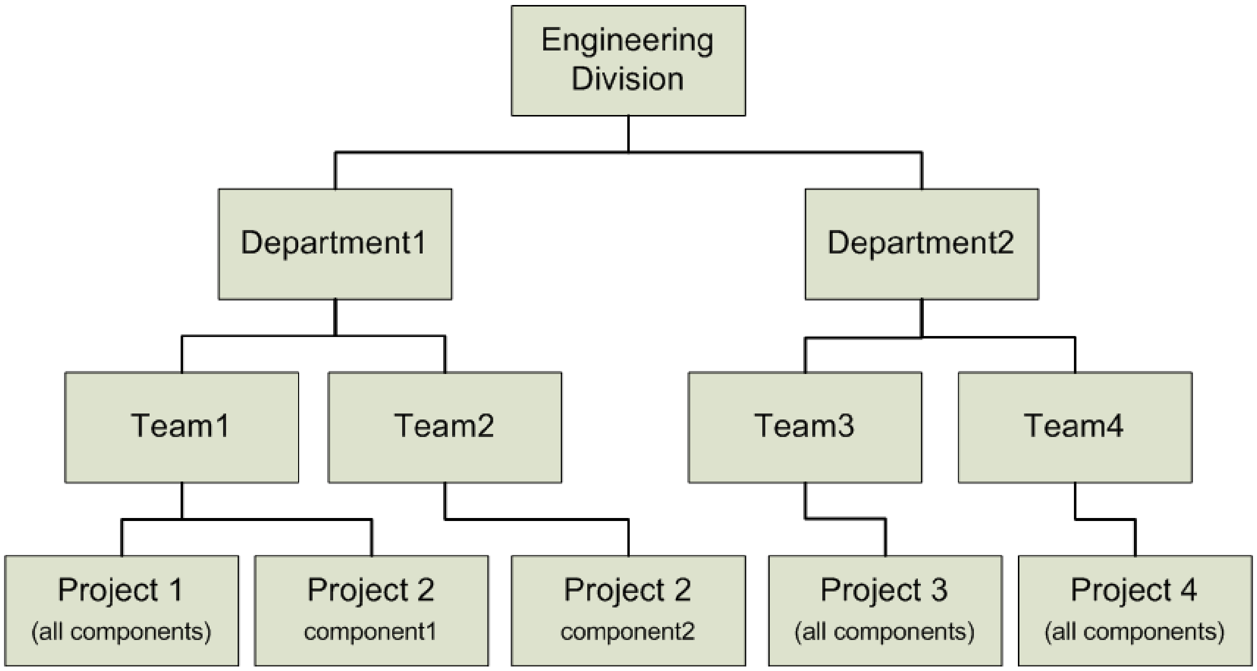


By company department or division

- A company might have engineering teams that span multiple departments or corporate divisions. In such a case, the children of the root node could be named according those departments or divisions. Each department could be made up of further subdivisions, such as teams.

  Figure 5.1.5, “Example: Organizing a Hierarchy by Department” divides the hierarchy by department. The leaf nodes of this hierarchy contain Coverity Connect projects and components that are associated with the teams in each department.

Figure 5.1.5. Example: Organizing a Hierarchy by Department



By a hybrid structure

- A company might have separate front end and back end teams for different products. In such a case, the children of the root node could represent each product, and the subdivisions of each product could identify the functional units of the product. Alternatively, the children of the root could be the front end and back end nodes, and the subdivisions of these functional units could identify the products. The leaf nodes of this hierarchy could contain Coverity Connect projects and components that are associated with submodules of each product or with subdivisions of the functional units.

  Figure 5.1.6, “Example 1: Organizing a Hierarchy by a Hybrid Structure” divides the hierarchy first by function, then by product.

Figure 5.1.6. Example 1: Organizing a Hierarchy by a Hybrid Structure

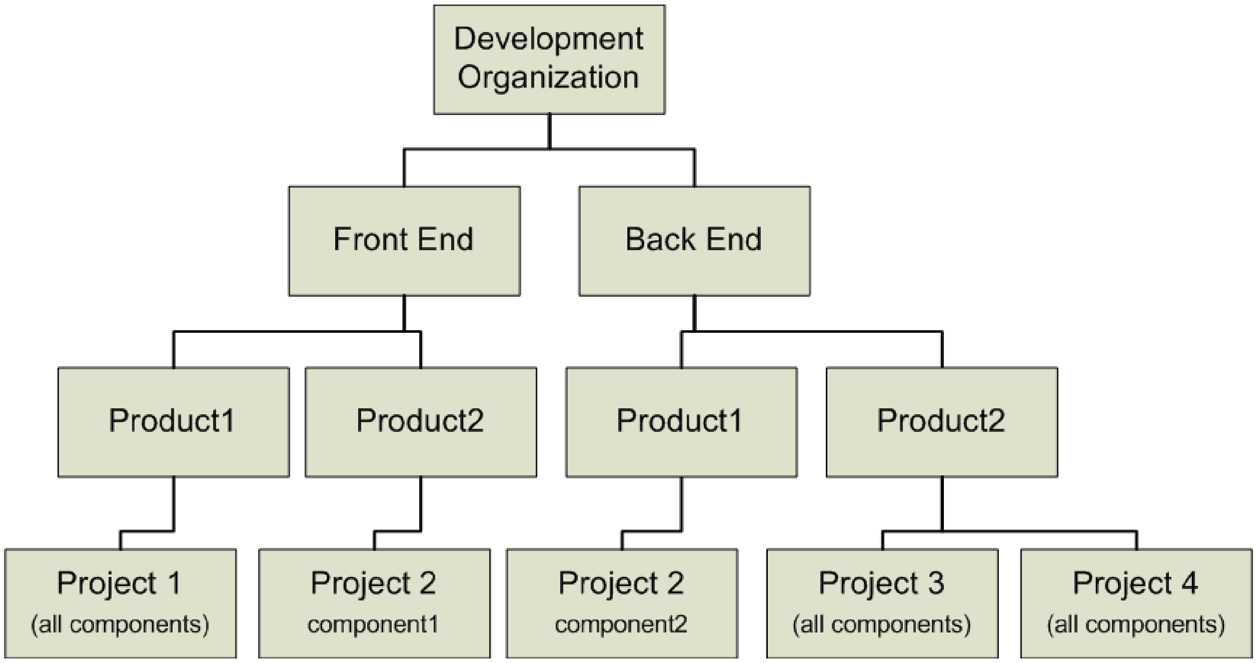
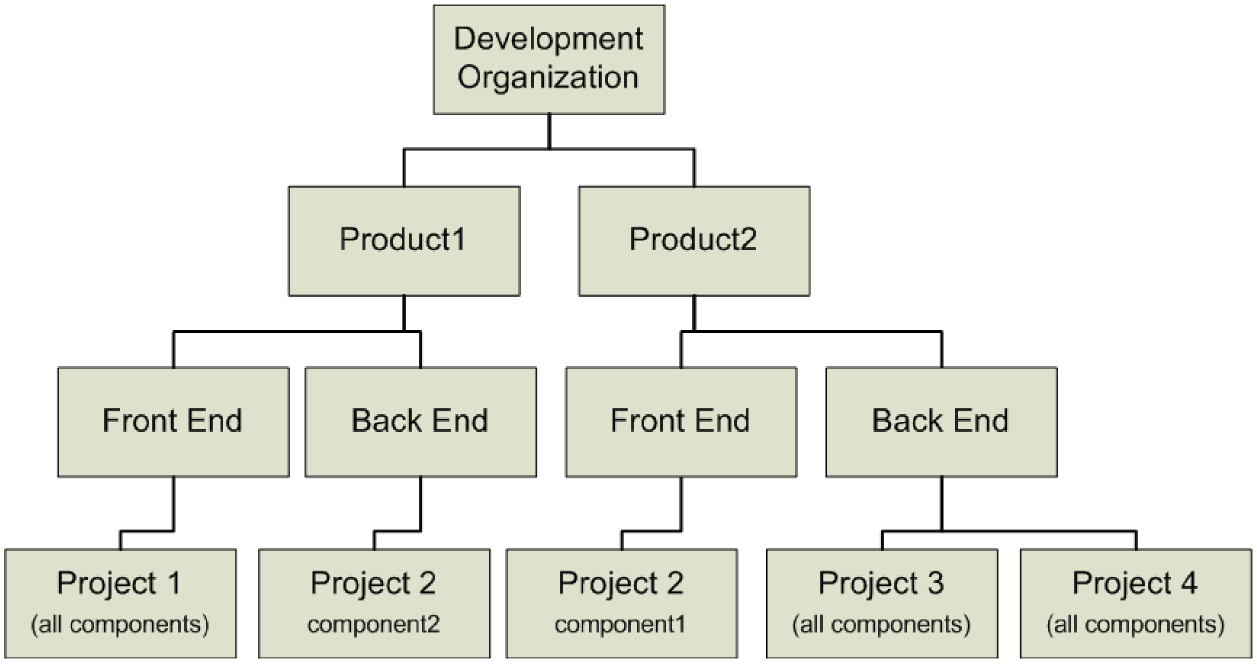


Figure 5.1.7, “Example 2: Organizing a Hierarchy by a Hybrid Structure” divides the hierarchy first by product, then by function.

Figure 5.1.7. Example 2: Organizing a Hierarchy by a Hybrid Structure



By Coverity Connect instance

- A company might host multiple instances of Coverity Connect that pass data to Coverity Policy Manager. In this case, the hierarchy could get data from projects and components in each Coverity Connect instance. Though the children of the root node could identify each Coverity Connect instance, they need not do so. The children could represent any other organizational structure that makes sense for the company.

  For example, assume a case in which one instance of Coverity Connect contains projects for a set of front end modules that are used by multiple products. Assume another instance contains projects for back end modules used by those products. A third instance might contain other modules for these and other products. In this case, the hierarchy could divide the code base by product and then break down each product by module and submodule. The leaf nodes could contain Coverity Policy Manager projects and components that are associated with the submodules of each product.

  A simpler example, Figure 5.1.8, “Example: Organizing a Hierarchy Explicitly by Coverity Connect Instance”, explicitly divides the hierarchy by Coverity Connect instance.

Figure 5.1.8. Example: Organizing a Hierarchy Explicitly by Coverity Connect Instance

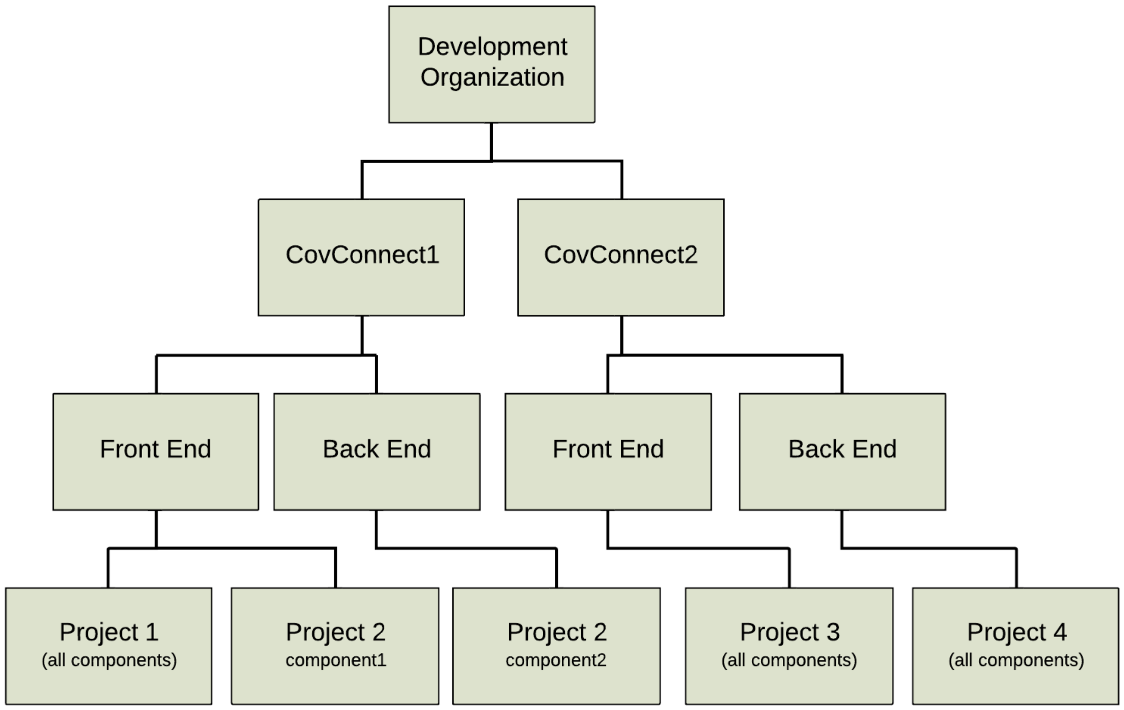
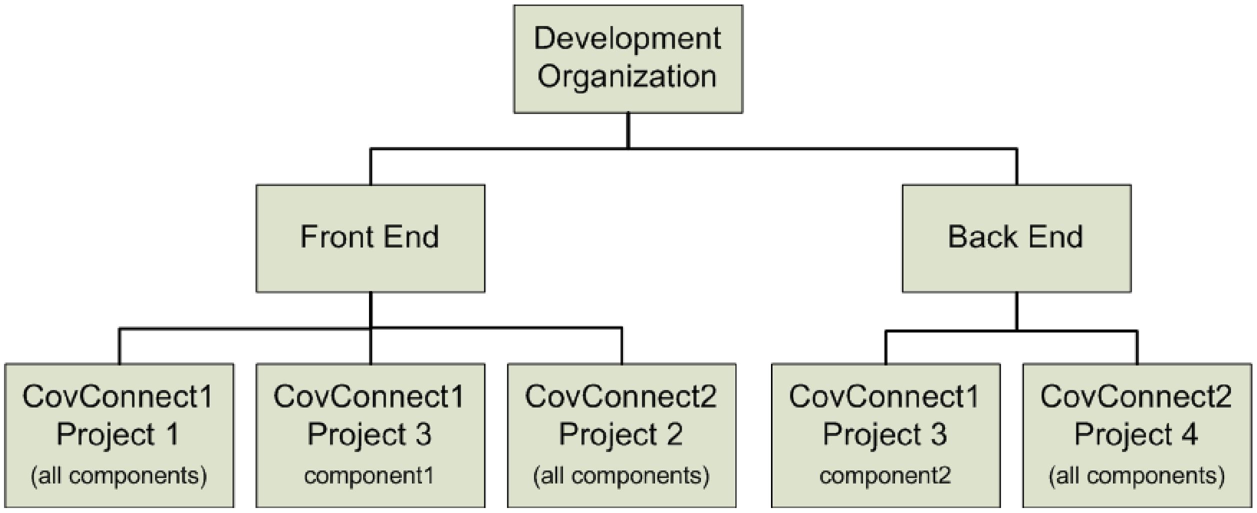


Figure 5.1.9, “Example: Organizing a Hierarchy Implicitly by Coverity Connect Instance” incorporates projects and components from multiple Coverity Connect instances without explicitly identifying

the instances. Note that this figure includes the same projects and components that Figure 5.1.8, “Example: Organizing a Hierarchy Explicitly by Coverity Connect Instance” includes.

Figure 5.1.9. Example: Organizing a Hierarchy Implicitly by Coverity Connect Instance

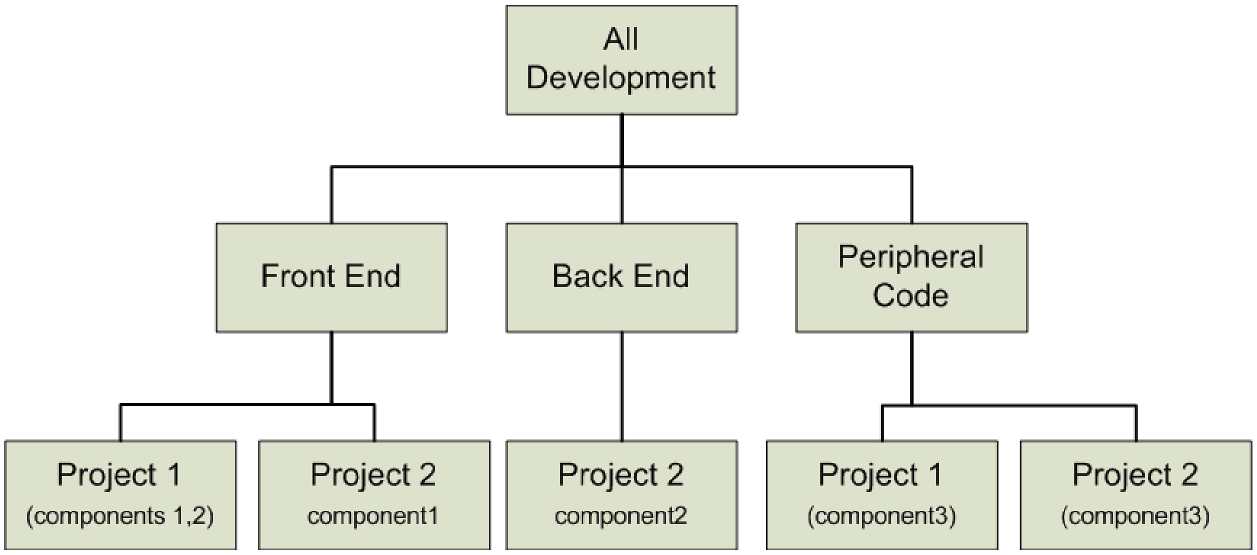

Development Organization
Front End
Back End
CovConnect1 Project 1 (all components)
CovConnect1 Project 3 component1
CovConnect2 Project 2 (all components)
CovConnect1 Project 3 component2
CovConnect2 Project 4 (all components)

With peripheral code bases

- Perhaps a company uses common libraries, third party code, or legacy code that managers want to examine separately from the primary code base. It is possible to create separate, high-level nodes in the hierarchy that are just for these items. In this case, one of the children of the root could be called *Peripheral*, and its children could be called *Libraries*, *Third Party*, and *Legacy*. The leaf nodes could then contain the projects and components that make up the divisions of the peripheral code bases.

  A simpler example, Figure 5.1.10, “Example: Configuring Peripheral Code Bases in a Hierarchy”, contains all the peripheral code in a single, high-level node.

Figure 5.1.10. Example: Configuring Peripheral Code Bases in a Hierarchy



Note that if users do not need to get status of some or all of the peripheral code base, Coverity Connect projects and components for that code can be omitted from the hierarchy. It is also possible to specify peripheral code bases within subnodes that use the *Exclude from rollup* feature; for instance, see the Libraries node in Figure 4.1.5, “Example: Tree Map”.

With new or modified code bases

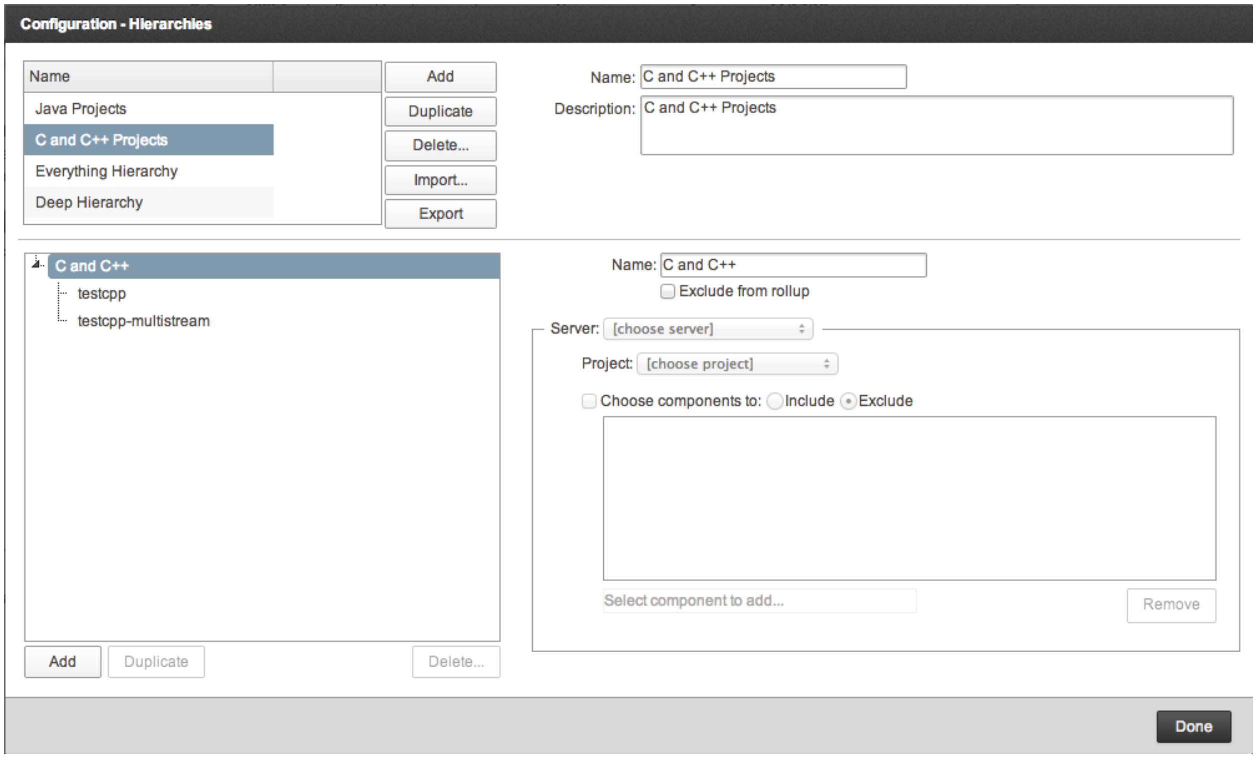
- Perhaps a company has added a product or an engineering group. It is possible to add nodes for them to an existing hierarchy. It is also possible to delete or reorganize existing nodes in the hierarchy.

For procedures on specifying one or more hierarchies, see Section 5.1.2, “Configuring a Hierarchy”.

5.1.2. Configuring a Hierarchy

In Coverity Policy Manager, a hierarchy specifies an ordered tree data structure (a node tree) for heatmaps and charts. You can create and maintain hierarchies through the Coverity Policy Manager UI or through a JSON file (see Chapter 5.2, *Importing/Exporting a Coverity Policy Manager Hierarchy*). The JSON import/export feature is preferred for large hierarchies or large-scale changes. The UI is preferred when creating smaller hierarchies or when making small-scale modifications to existing hierarchies.

Figure 5.1.11. Example: Hierarchy Configuration Window



Name: C and C++ Projects
Description: C and C++ Projects

Name: C and C++
Exclude from rollup
Server: [choose server]
Project: [choose project]
Choose components to: Include [✓ Exclude]
Select component to add...

As shown in Figure 5.1.11, “Example: Hierarchy Configuration Window”, the *Name* column (top-left portion of the window) lists all hierarchies that have been created. Selecting a name in this list allows you to use the edit settings fields for the hierarchy.

Hierarchy Buttons

- **Add**: Generates a new hierarchy. See Section 5.1.2.1, “Creating a Hierarchy”. Note the other Add button (located at the bottom of the screen) is for the node tree (see Node Tree Buttons).
- **Duplicate**: Creates a copy of the selected hierarchy that is identical except for the name. For example, a duplicate of *C and C++* is named *C and C++ Copy*.
- **Delete**: Deletes the hierarchy along with its node tree.
- **Import**: Uploads a hierarchy configuration to Coverity Policy Manager. See Chapter 5.2, *Importing/ Exporting a Coverity Policy Manager Hierarchy*.
- **Export**: Downloads a hierarchy configuration to a JSON file. This file is convenient for large-scale hierarchy configurations in which you edit this file and then import it back to Coverity Policy Manager instead of using the node configuration functionality in the UI. See Chapter 5.2, *Importing/Exporting a Coverity Policy Manager Hierarchy*.

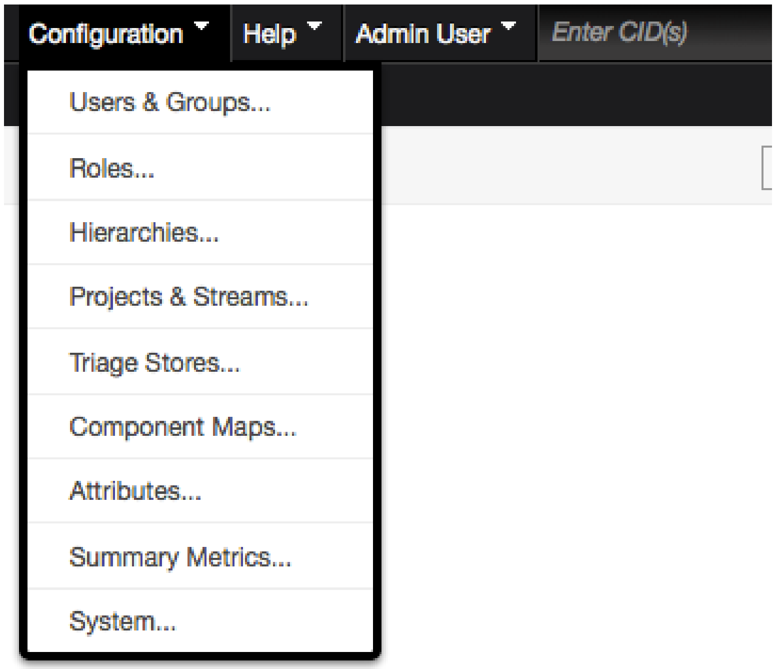
5.1.2.1. Creating a Hierarchy

Before creating a hierarchy, take time to plan. Think about the kind of data that your Coverity Policy Manager users need to see in the heatmaps and charts. For guidance, see Section 5.1.1, “Planning a Hierarchy”.

To create a hierarchy:

1. Navigate to the *Configuration - Hierarchies* window by selecting *Hierarchies* from the *Configuration* menu.

   Figure 5.1.12. Example: Configuration Menu

   Note

   If *Hierarchies* does not appear in this menu, you need to get permission to view and manage hierarchies from your Coverity Connect administrator (for more details, see Chapter 5.5, *Coverity Policy Manager Roles and Permissions*).

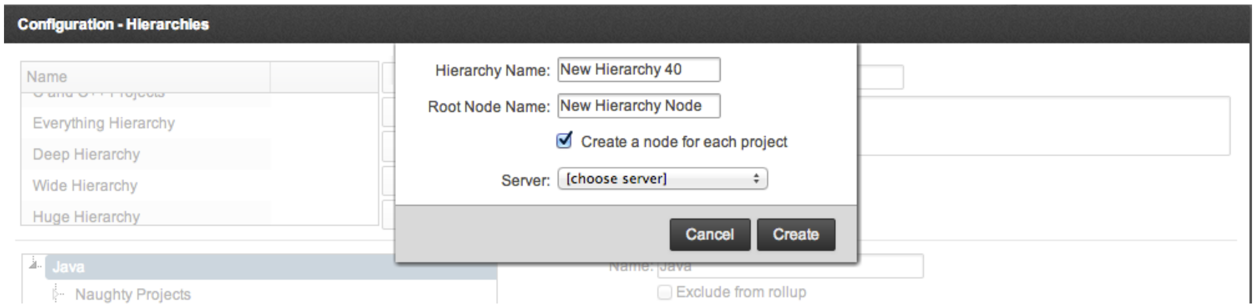
2. Use the **Add** button (in the top-left portion of the *Configuration - Hierarchies* window) to create the hierarchy.

   See an example of this window in Figure 5.1.11, “Example: Hierarchy Configuration Window”.

   This action opens a pop-up window in which you can type a name and description for the hierarchy (recommended), and opt to generate a flat hierarchy that contains a single root node with a separate leaf node for each Coverity Connect project. If you do not select this option, the new hierarchy will contain a single root node.

You can also select which server you would like the data to be drawn from. Your local server will be selected by default, but additional servers will be available to Coverity Connect instances that serve as Coordinators to one or more Subscribers. See Section 3.5.1, “Synchronizing multiple Coverity Connect instances” for more details.

Figure 5.1.13. Example: Hierarchy Creation Window



Hierarchy Name: New Hierarchy 40
Root Node Name: New Hierarchy Node
☑ Create a node for each project
Server: [choose server]

The name and description of the hierarchy will appear to end users in the Coverity Policy Manager list of hierarchies and in navigation menus. For examples, see the figures in Chapter 1.3, *Shared Navigation and Main menu tools*. The name must be unique.

Note

The following characters are not allowed in the name of a hierarchy: `: \ / * ' "`

3. Specify the node tree for the hierarchy:

   a. [Recommended] Change the automatically generated name of the root node for the hierarchy (New Hierarchy Node) by typing it into the lower-right *Name* field.

      The name of the root name will appear in Coverity Policy Manager charts (see the *Sample Company* root node in Figure 4.2.1, “Example: Chart Navigation Breadcrumbs (Java → Desktop Projects → Developer Streams)”). Note that the name of the root node can differ from the name of the hierarchy.

      The name of the root often indicates the relationship of the root to the children (for examples, see Section 5.1.1, “Planning a Hierarchy”). Note that it is not possible to create a sibling of the root node.

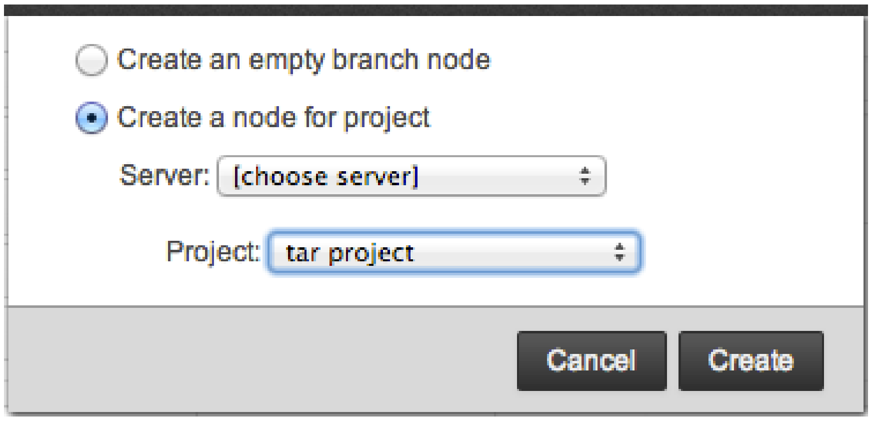
   b. Use the buttons at the bottom of the screen to generate nodes (add or duplicate) or delete nodes (along with all descendants of the deleted node).

      Note that nodes located at the same level in the tree must have different names.

      Node Tree Buttons

      - **Add**: Generates a child node for the selected node. You can opt to make this child a branch node or a leaf node. If you choose to generate a leaf node, you will be able to associate it with a project and, if applicable, a specific server.

Figure 5.1.14. Example: Node Creation Pop-up Window


Create an empty branch node [✓ Create a node for project]

Server: [choose server]

Project: tar project

- **Duplicate**: Generates a sibling node that is identical to the selected node except for the name. For example, a duplicate of *Front End* is named *Front End Copy*. This sibling will contain copies of any descendants of the duplicated node.

  It is not possible to create a duplicate of the root node.

- **Delete**: Deletes a node from the selected hierarchy. Deleting a node also deletes any descendents of that node. However, it is not possible to delete the root node.

  If you delete all the descendents of a given branch node, the node will have no data associated with it until you associate it with a project or create a leaf node for it that is associated with a project.

Tip

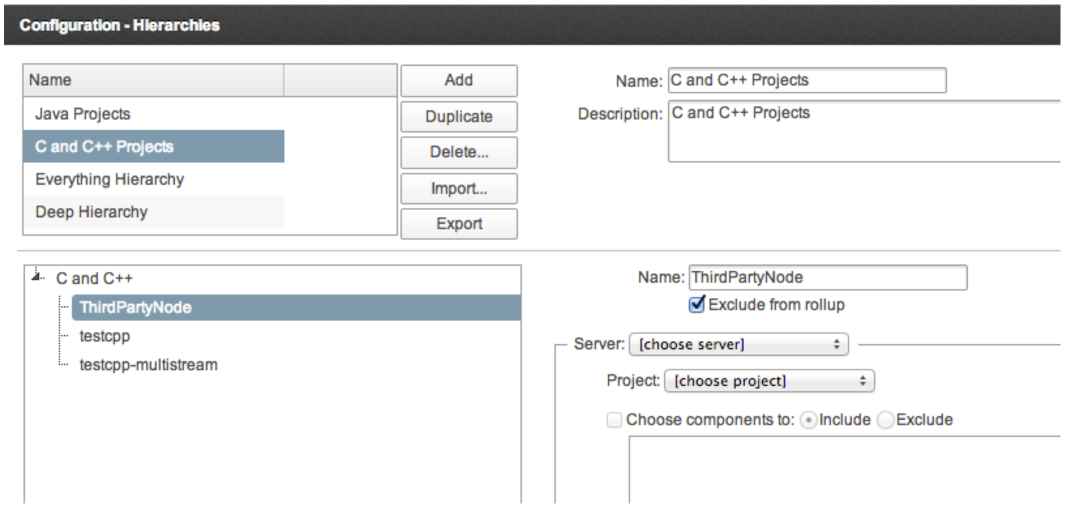
In general, the following limits are recommended:

- 1-100 child nodes for a single branch node.
- Upper bound of 2000 nodes for the entire node tree.
- Node depth of 1-10 levels, where 1 is the level of a child of the root node, 2 is a "grandchild" of the root node, and so on.

Drag and Drop functionality
: You can move a node to a new location in a hierarchy by selecting it, then dragging and dropping it to the desired location. Any descendents of a branch node will move along with the branch you select.

c. If you do not want data from a given node to contribute to data values of its parent and ancestor nodes, check *Exclude from rollup* for that node.

Figure 5.1.15. Example: Exclude Node From Rollup



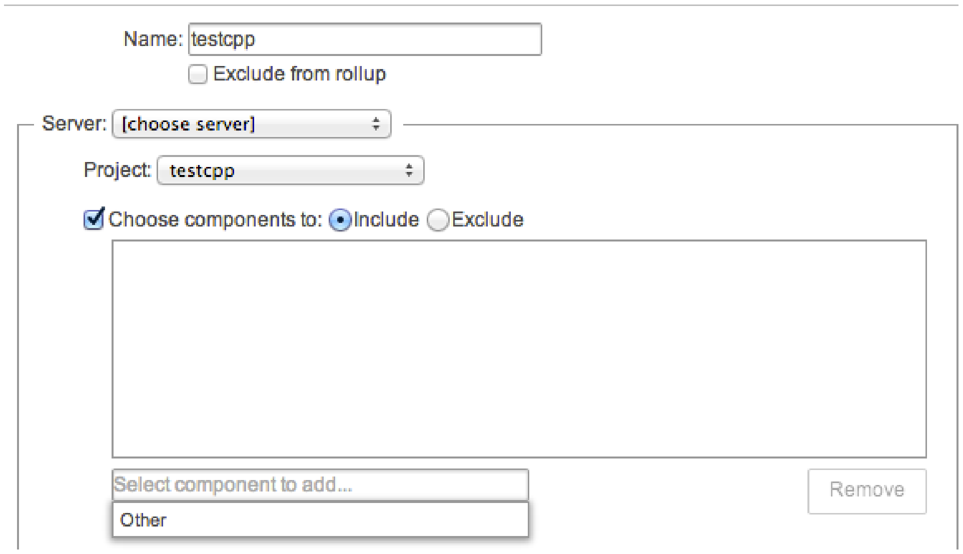
Name: C and C++ Projects
Description: C and C++ Projects
Name: ThirdPartyNode
[✓] Exclude from rollup

In the example, values from the selected, *third party* node are excluded from rollup.

This functionality can be useful for excluding data on issues from peripheral code bases (perhaps some third-party source) that your organization does not need to assess. For example, see With peripheral code bases.

d. Associate your leaf nodes with a project (and any components).

Figure 5.1.16. Example: Associating a Project and Component with a Node



Name: testcpp
[ ] Exclude from rollup
Project: testcpp
[✓] Choose components to: [✓ Include] Exclude

To support any data, a leaf must be associated with a single Coverity Connect project. Optionally, a leaf can narrow the scope of the project data by specifying a list of components. You can opt to *Include* or *Exclude* data that is derived from the specified components.

Note

If you add a node to a leaf node, the leaf will become a branch node and lose any association with a project (and components). The new node will become a leaf to which you can associate a project and any components.

The leaf will retain its association with a project or component even if the name of the project or component changes.

If the project with which a leaf is associated gets deleted, the leaf will no longer have data associated with it. Similarly, if the leaf is associated with a single component, and that component gets deleted, the leaf will no longer have data associated with it. The UI will indicate when a project or a component with which a leaf is associated gets deleted.

4. Click the **Done** button to save your changes.

5.1.2.2. Editing a Hierarchy

You can change the name and/or description of a selected hierarchy and change properties of any selected node in the hierarchy. For guidance with this process, see Section 5.1.2.1, “Creating a Hierarchy”.

Chapter 5.2. Importing/Exporting a Coverity Policy Manager Hierarchy

You can import and export one or more hierarchy files through the *Configuration - Hierarchies* window in Coverity Policy Manager (see Hierarchy Buttons). These editable JSON files specify the name, description, and node tree for the hierarchy.

Use Case

To expedite the specification of a hierarchy with many nodes, you might export an existing hierarchy to a file so that you can edit it manually (instead of using the UI for this purpose) and then import the edited version of the file to Coverity Policy Manager.

To see working examples of a hierarchy configuration in the file, you might want to to create a small, representative portion of the hierarchy through the Coverity Policy Manager UI before exporting it. For details, see Section 5.1.2.1, “Creating a Hierarchy”.

Hierarchy Objects use the following schema notation (not JSON):

- Braces and Brackets (examples: `{ }`, `[ ]`): These characters are verbatim.
- Italics (example: *`hierarchy`*): Items in italics represent objects defined elsewhere in the schema.
- Ellipsis (example: `thing...`): An item followed by an ellipsis represents zero or more repetitions of the item, separated by commas.
- Colon after word (example: `foo:`): A word followed by a colon represents that word in double quotes.
- Quotes (example: `"leaf"`): A string in quotes represents itself.
- `string`: This token represents a UTF-8 string in double quotes.
- Boolean (examples: `true`, `false`): This token represents either true or false.
- `(text)`: Text in parentheses is a comment. For example: `(constrained)`

All strings can be up to 256 characters long. Strings notated as (constrained) cannot include control characters or the following characters: `\ : / * ` ‘ “`

Hierarchy Objects

Top-level object

```
{ hierarchies : [<emphasis>hierarchy...</emphasis>] (You can import/export one or
 more hierarchies) }
```

hierarchy object

```
{
    name : string, (constrained)
    description : string,
    tree : <emphasis>tree</emphasis>
}
```

tree object
: Either a leaf or a branch object.

leaf object

```
{
    class : “leaf”,
    components : [ <emphasis>component</emphasis> … ],
    projectName : string, (constrained, may be null)
    componentsIncluded : boolean,
    name : string, (constrained)
    includeInPolicyEvaluation : boolean
}
```

branch object

```
{
    class : “branch”,
    children : [ <emphasis>tree</emphasis> … ],
    name : string, (constrained)
    includeInPolicyEvaluation : boolean
}
```

component object

```
{
    componentMap : string, (constrained)
    componentName : string  (constrained)
}
```

Example

```
{
  "hierarchies" : [ {
    "name" : "A tiny hierarchy",
    "description" : "Has 3 nodes",
    "tree" : {
      "class" : "branch",
      "children" : [ {
        "class" : "leaf",
        "components" : [ ],
        "projectName" : "sample-ces-app",
        "componentsIncluded" : true,
        "name" : "sample-ces-app",
        "includeInPolicyEvaluation" : true
      }, {
        "class" : "leaf",
        "components" : [ ],
        "projectName" : null,
        "componentsIncluded" : false,
        "name" : "i18n",
        "includeInPolicyEvaluation" : true
      } ],
      "name" : "Java",
      "includeInPolicyEvaluation" : true
    }
  } ]
}
```

Note

If you import a hierarchy and export it again, the JSON will be the same with one exception: Branch nodes with no children will be converted to leaf nodes with null projects.

If you import a hierarchy that has a name that matches an existing hierarchy, the existing hierarchy will be replaced by the imported one.

An error occurs under the following conditions:

- If Coverity Connect cannot parse your JSON.
- If a node name in the tree is not unique within its container.
- If a project or component name does not refer to an existing project or component.

Chapter 5.3. Scheduling the Extract Transform Load (ETL) Process

During the Extract Transform Load (ETL) process, Policy Manager extracts and aggregates raw data from the database, updating the information in the PM tables. You can schedule the ETL process to run at any (designated) time. By default, an ETL task is scheduled to run one hour after completion of the previous run. Note that this process can be time-consuming, however, and often requires extra disk space.

You can switch to `cron`-based scheduling by setting property values in `config/cim.properties`. You can either use the same schedule for both status and trend ETL processes, or you can set up separate schedules, depending on which properties you set.

Note

Data aggregated by the ETL process does not last indefinitely. Daily data is kept for 40 days, weekly data is kept for 30 weeks, monthly data is kept for 24 months, and yearly data is kept indefinitely. Older data is deleted after new data is inserted.

Also, be aware that the ETL process does not save a history of previous configurations (that is, component map and stream settings). Whenever the current project configuration is changed, Policy Manager data is recomputed based on the new settings.

To set up a single schedule for both status and trend ETL processes, set the following properties as indicated:

- `policymanager.etl.scheduled.disable=false`
- `policymanager.etl.cron.enable=true`
- `policymanager.etl.cron.schedule=`*`date-and-time`*

Note

The scheduled `cron` job follows a specific syntax. For example:

```
policymanager.etl.cron.schedule=0 30 2 * * *
```

The example above represents a `cron` job scheduled to run every night at 0230 hours.

You can disable this schedule by setting the properties as follows:

- `policymanager.etl.scheduled.disable=true`
- `policymanager.etl.cron.enable=false`

To set up separate schedules for status and trend ETL processes, set the following optional properties in addition to the three properties listed above:

- `policymanager.etl.trend.scheduled.disable=false`

- `policymanager.etl.trend.cron.enable=true`
- `policymanager.etl.trend.cron.schedule=`*`date-and-time`*

These properties control the schedule only for trend ETL processes. Once you set these trend properties, the previous three properties control only status ETL processes.

With all six properties set, you can specify different schedules for status and trend ETL processes. You can also enable and disable the status and trend schedules separately. For example, to disable the trend schedule, set the following properties as indicated:

- `policymanager.etl.trend.scheduled.disable=true`
- `policymanager.etl.trend.cron.enable=false`

Note

You can set only some of the trend properties, if you desire. For example, if you want separate status and trend schedules, but want to enable/disable them together, then set only this property (in addition to the first three):

- `policymanager.etl.trend.cron.schedule=`*`date-and-time`*

Similarly, if you want to use the same schedule for status and trend ETL processes but want to separately enable/disable them, set only these properties (in addtion to the first three):

- `policymanager.etl.trend.scheduled.disable=false`
- `policymanager.etl.trend.cron.enable=true`

Chapter 5.4. Creating Summary Metrics

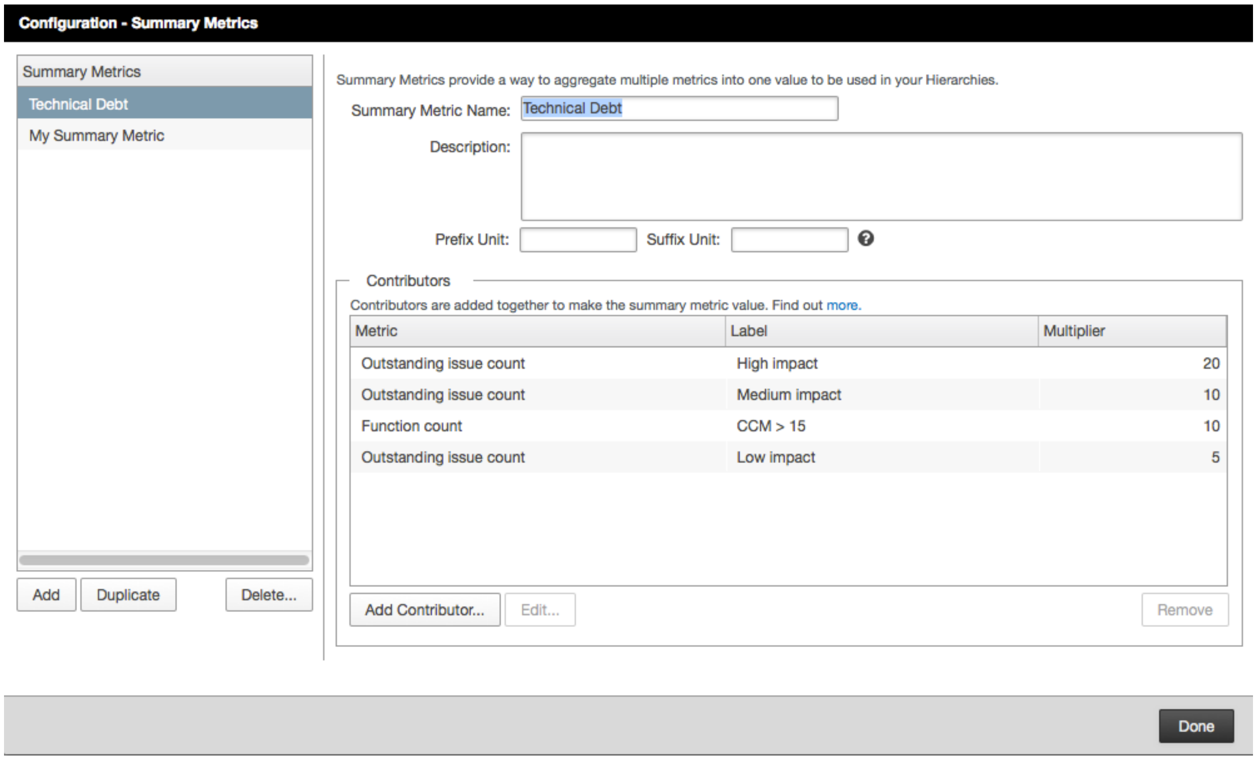
You use the *Configuration - Summary Metrics* screen to create and maintain Summary metrics that Coverity Policy Manager users can select for their heatmaps and charts.

By default, you can add up to 10 contributor metrics to a summary metric. You can change this default through the `summary.metric.contributor.limit` property in `<install_dir>/config/cim.properties`.

Recommended: Tuning the Technical Debt summary metric

You should tune the default values of the Technical Debt metric to match the needs of your organization, preferably *before* users can begin to incorporate the metric into their charts. You can change contributor properties (multipliers, metrics, and/or filters), remove contributors, or add new contributors, as needed.

Figure 5.4.1. Configuration - Summary Metrics screen



Summary Metrics provide a way to aggregate multiple metrics into one value to be used in your Hierarchies.

Summary Metric Name: Technical Debt

Description:

Prefix Unit: Suffix Unit:

Contributors

Contributors are added together to make the summary metric value. Find out more.

Metric	Label	Multiplier
Outstanding issue count	High impact	20
Outstanding issue count	Medium impact	10
Function count	CCM > 15	10
Outstanding issue count	Low impact	5

Summary metrics consist of the following components:

- *Summary Metric Name*: The name that you use to identify the metric to users.
- *Description*: A description that you can add to the Summary metric.
- *Prefix Unit*: Optional label (such as the "$" in $100) that precedes the value of a given summary metric. This unit is visible in the Coverity Policy Manager UI, for example, in the axes of graphs, in table

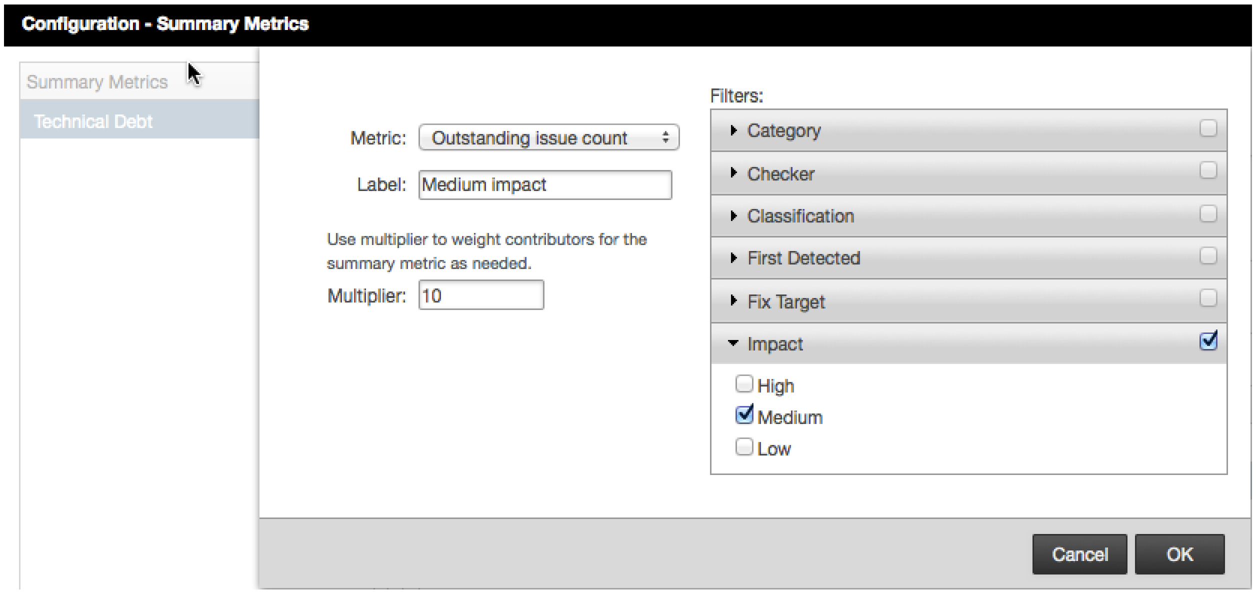
column or row headers, and in information boxes that identify data points that appear in a chart or heatmap.

- *Suffix Unit*: Optional label (such as the "K" in 100K, which is used an abbreviation used for 100,000) that follows the value of a given summary metric. This unit is visible in the Coverity Policy Manager UI, for example, in the axes of graphs, in table column or row headers, and in information boxes that identify data points that appear in a chart or heatmap.
- *Contributors*: List of one or more filtered metrics that you select for your summary metric. To establish the relative importance of a given contributor, you need to assign a non-negative, numeric weight as a multiplier. For example, the default multipliers used in the built-in summary metric, *Technical Debt*, weight the impact of issues and complexity of functions (where CCM refers to Cyclomatic Complexity).

To add a contributor to a summary metric:

1. Click the **Add Contributor** button to open the associated edit screen.

   Figure 5.4.2. Example: Edit screen for a contributor

   Configuration - Summary Metrics

   Summary Metrics
   Technical Debt

   Metric: Outstanding issue count
   Label: Medium impact
   Use multiplier to weight contributors for the summary metric as needed.
   Multiplier: 10

   Filters:
   - Category [ ]
   - Checker [ ]
   - Classification [ ]
   - First Detected [ ]
   - Fix Target [ ]
   - Impact [✓]
     - [ ] High
     - [✓] Medium
     - [ ] Low

2. Select a metric and filter.
3. Provide a label and multiplier for the contributor.

   Using a multiplier of zero (`0`) will disable a contributor.

Removing Summary Metrics

Use caution when deleting a summary metric. Deleted Summary metrics disappear without notification from any reports that use them. Obviously, this change can affect the meaning of a report or, in the case that the report used only one metric, lead to a completely empty report.

Chapter 5.5. Coverity Policy Manager Roles and Permissions

Coverity Connect controls user (and user group) access to Coverity Policy Manager components through role-based permissions described in Global Permissions. The username of all Coverity Policy Manager administrators and users must exist in a Coverity Connect user database. If not, you need to contact your Coverity Connect administrator.

By default, the *Policy Manager User* role allows you to log in and view Policy Manager The latter permission allows you to create and edit Coverity Policy Manager reports and heatmaps that you can share with other users. The default *Hierarchy Administrator* role allows you to log in, view Coverity Policy Manager, perform administrator-level configurations (which include creating Hierarchies and Summary Metrics), and use the Web services.

For details about additional role assignment options, see Step 5 in Section 3.2.1.2, “Editing a local or LDAP user”. For detailed information about roles, see Section 3.2.3, “Roles and role based access control”.

Part 6. Coverity Report Generators

Table of Contents

6.1. Configuring Report Generators 336
    6.1.1. Configuration overview 336
    6.1.2. General considerations 336
    6.1.3. Unified configuration file 349

Chapter 6.1. Configuring Report Generators

Table of Contents

6.1.1. Configuration overview 336
6.1.2. General considerations 336
6.1.3. Unified configuration file 349

6.1.1. Configuration overview

The Report Generators must be configured by using a unified `.yaml` configuration file, which indicates how the reports should be generated. The unified configuration file must contain the standard schema elements, as well as any additional schema elements that are required for a particular report. This section provides information about the different schema elements that should be used when configuring a report.

6.1.2. General considerations

The configuration file uses standard .yaml syntax as described in https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html. In addition, please observe the following:

- All configuration settings for various reports can be included in one `.yaml` file per project.
- Each Coverity project requires its own `.yaml` configuration file with a unique filename. For example:

  `coverityprojectconfig1.yaml`, `coverityprojectconfig2.yaml`, `coverityprojectconfig3.yaml`
- Configuration settings (or customized schema elements) that stand alone in the `.yaml` configuration file can be expressed outside of a nested block.
- Some elements that are required for a particular report are not present in the schemas because they must be provided via command line. For example, the password must be entered via command line.
- Some fields are file pathnames. A relative pathname is interpreted relative to the directory containing the configuration file. If the configuration did not come from a file, the pathname would be relative to the Report Generators' working directory.

  Pathnames may use a slash or backslash as a separator, whichever is more appropriate for their platform.

6.1.2.1. Standard schema elements

The standard schema is the same for all reports and is required for all reports. It looks like this:

```
version:
    schema-version: 5
```

```
connection:
    url: https://coverity.example.com:8443/
    username: admin
    ssl-ca-certs:
project:
title-page:
    company-name:
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:
locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:
```

The following sections describe these fields in alphabetical order.

6.1.2.1.1. Connection schema elements

The connection schema should include the following keys:

Key	Class Type	Description	Default	Required?
`ssl-ca-certs`	String	Lists the pathname to a file containing additional CA certificates that are used in establishing a secure HTTPS connection through an SSL handshake. Pathnames must be entered in PEM format.	N/A	No
`url`	String	Lists the URL of the Coverity Connect instance. This URL must not include user name and password.	N/A	Yes
`username`	String	Lists the user name that is used to connect to Coverity.	N/A	Yes

6.1.2.1.2. Issue cutoff schema element

The issue cutoff schema should include the following key:

Key	Class Type	Description	Default	Required?
`issue-cutoff-count`	Integer	Sets the limit for the maximum number of issues displayed in a report. This setting helps to control the size of the generated report. It is used for the CVSS, Security, PCIDSS, Mobile OWASP, and OWASP 2017 reports. For the Security Report, the maximum value is 10,000.	200	No

6.1.2.1.3. Locale schema element

The locale schema should include the following key:

Key	Class Type	Description	Default	Required?
`locale`	String	Sets the language or locale in which the report will be generated. The following locale settings are available: • `en_US` (English) • `ja_JP` (Japanese) • `ko_KR` (Korean)	en_US	No

Key	Class Type	Description	Default	Required?
		• `zh_CN` (Simplified Chinese)		

6.1.2.1.4. Project schema elements

The project schema must include the following key:

Key	Class Type	Description	Default	Required?
`project`	String	Lists the name of the Coverity Connect project.	N/A	Yes

Note

The project name you specify with the `--project` command option on the command line will override this key setting.

6.1.2.1.5. Snapshot schema elements

Elements for snapshot-id and snapshot-date are described below. If both keys are defined, `snapshot-id` will get the highest priority.

Key	Class Type	Description	Default	Required?
`snapshot-id`	Long	Retrieves the defects of a specific snapshot id, instead of using the latest snapshot id of all the streams associated with the project.	N/A	No
`snapshot-date`	String using the format MM/DD/YYY	Retrieves the most recent snapshot of each stream in the project whose date is less than or equal to the given date.	N/A	No

6.1.2.1.6. Report-filtering elements

Key	Class Type	Description	Default	Required?
`components`	String	An optional comma-separated list of Coverity Connect	The report includes data for all components.	No

Key	Class Type	Description	Default	Required?
		component names, which include component map names. If the components are listed here, the report will include data only for the listed components; for example: `Default.lib` or `Default.src`.		
`issue-kind`	String	An optional comma-separated list of Coverity Connect issue kinds. If issue kinds are listed here, the report will include only issues of the listed kinds. The possible values for `issue-kind` are as follows: • `Quality` • `Security` The following line is an example of using this option: `issue-kind: Quality`	The report includes both quality and security issues.	No

6.1.2.1.7. Title page schema elements

The schema for the title page should include the following key:

Key	Class Type	Description	Default	Required?
`company-name`	String	Lists the customer's company name.	N/A	Yes
`logo`	String	Lists the file pathname to display the logo of	N/A	No

Key	Class Type	Description	Default	Required?
		the company. Valid image types are `.bmp`, `.gif`, `.jpg`, and `.png`. The maximum image size allowed is 210 pixels wide by 70 pixels high. (If a backslash character is used in the file pathname, then it must be a double backslash. For example: `C:\\logo\\ourlogo.jpg`. You can also use a single forward slash, like this: `/var/logo/ourlogo.png`.)		
`organizational-unit-name`	String	Lists the name of your division, group, team, or organizational unit.	N/A	Yes
`organizational-unit-term`	String	Lists the unit term used for the organization.	N/A	Yes
`prepared-by`	String	Lists the name of the entity or individual that prepared the report.	N/A	Yes
`prepared-for`	String	Lists the name of the entity or individual for which the report was prepared.	N/A	Yes
`project-contact-email`	String	Lists the email address of the project contact; the email address of the recipient of the report.	N/A	Yes

Key	Class Type	Description	Default	Required?
		This element is used by the following reports: CIR, CVSS, PCIDSS, Mobile OWASP, and OWASP 2017.		
`project-name`	String	Lists the project name for the report.	N/A	Yes
`project-version`	String	Lists the project version number.	N/A	Yes

6.1.2.1.8. Schema version element

The schema version element must include the following key:

Key	Class Type	Description	Default	Required?
`schema-version`	Integer	Sets the version number for the schema. Changes that alter the semantics of the parts of the schema that are independent of the report in a non-additive way must trigger an increment of this number.	N/A	Yes

6.1.2.2. Schema elements for specific reports

In addition to the standard schema elements that are required for all of the report generators, some reports also require report-specific configuration settings. This section describes the different report-specific schema elements.

6.1.2.2.1. CERT Report

The CERT Report has an additional key for specifying and defining target levels:

Key	Class Type	Description	Default	Required?
`target-level`	String	Specifies the compliance standards for	F	No

Key	Class Type	Description	Default	Required?
		calculating the report. The following values are available: • F: Fully Compliant • L2: L2 Compliant • L1: L1 Compliant		

For more information about the CERT Report configuration, see Section 12.2.3, “Configuring the Coverity CERT Report”.

6.1.2.2.2. Coverity Integrity Report

The CIR Report has additional keys for report specification:

```
cir-report:
    project-description:
    project-details:
    target-integrity-level:
    high-severity-name:
    unspecified-severity-name:
    trial:
    loc-multiplier:
```

Key	Class Type	Description	Default	Required?
`high-severity-name`	String	Lists the name of the highest severity value.	Major	No
`loc-multiplier`	String	Sets the LOC multiplier for the number of lines of code that have been inspected.	1	No
`project-description`	String	States the project description.	CIM description	No
`project-details`	String	States the details of the project.	N/A	No
`target-integrity-level`	Integer	Defines the target's integrity levels. These are the standard target integrity levels:	1	No

Key	Class Type	Description	Default	Required?
		• 1: < 1 defect per thousand lines of code • 2: < .1 defects per thousand lines of code • 3: < .01 defects per thousand lines of code, and other requirements		
`trial`	Boolean	Activates the trial flag. The trial flag should be set to `True` if page three of the report should not be printed. (Page three contains severity data which is not applicable to a trial.)	false	No
`unspecified-severity-name`	String	Lists the names of any unspecified severity values.	Unspecified	No

For more information about the Integrity Report configuration, see Chapter 7.3, *Updating the Coverity Integrity Report configuration file*.

6.1.2.2.3. Security Report

The Security Report has additional keys for defining assurance levels and severity mappings:

```
security-report:
    assurance-level-score: 90
    assurance-level: AL1
    severity-mapping: Carrier Grade
    severity-mapping-description:
    custom-severity-mapping:
        modify-data: very high
        read-data: very high
        dos-unreliable-execution: very high
        dos-resource-consumption: very high
        execute-unauthorized-code: very high
        gain-privileges: very high
        bypass-protection-mechanism: very high
        hide-activities: very high
```

Assurance level

Key	Class Type	Description	Default	Required?
`assurance-level`	String	Indicates the level and its minimum acceptable score for the report to be considered passing. The following values are available: AL1, AL2, AL3, and AL4.	AL1	Yes
`assurance-level-score`	Integer	Indicates the assurance level score. There are four assurance levels, representing security scores that are greater than or equal to 60, 70, 80, and 90. When choosing the proper assurance level, consider its potential for damage to life, property, or reputation. For example, an application with high damage potential should have a high assurance level.	90	No

Custom severity mapping

Field	Class Type	Description	Default	Required?
`bypass-protection-mechanism`	String	Indicates if the attacker tries to bypass	Very high	No

Field	Class Type	Description	Default	Required?
		application or system protection mechanisms. The following values are available: very high, high, medium, low, very low, and informational.		
`custom-severity-mapping`	String	If the severity mapping is set to `Custom`, then update the `custom-severity-mapping` with specific settings and values. The following values are available: `very high`, `high`, `medium`, `low`, `very low`, and `informational`.	Very high	No
`dos-resource-consumption`	String	Indicates when a weakness creates a denial of service due to excessive resource consumption by the application. The following values are available: `very high`, `high`, `medium`, `low`, `very low`, and `informational`.	Very high	No
`dos-unreliable-execution`	String	Indicates when the weakness creates a denial of service due to unreliable	Very high	No

Field	Class Type	Description	Default	Required?
		execution of the application. The following values are available: `very high`, `high`, `medium`, `low`, `very low`, and `informational`.		
`execute-unauthorized-code`	String	Indicates when an attacker tries to execute code that they do not have authority to execute. The following values are available: `very high`, `high`, `medium`, `low`, `very low`, and `informational`.	Very high	No
`gain-privileges`	String	Indicates when an attacker tries to gain privileges that should not be available to them. The following values are available: `very high`, `high`, `medium`, `low`, `very low`, and `informational`.	Very high	No
`hide-activities`	String	Indicates that an attacker might try to hide activities from visibility in an audit. The following values are available: `very high`, `high`,	Very high	No

Field	Class Type	Description	Default	Required?
		`medium`, `low`, `very low`, and `informational`.		
`modify-data`	String	Indicates when an attacker tries to modify data in the application. The following values are available: `very high`, `high`, `medium`, `low`, `very low`, and `informational`.	Very high	No
`read-data`	String	Prevents the attacker from reading data that is private to the application. The following values are available: `very high`, `high`, `medium`, `low`, `very low`, and `informational`.	Very high	No
`severity-mapping-description`	String	Indicates the description for the custom severity mapping.	N/A	No

Severity mapping

This standalone mapping specifies the name of the severity map (formerly known as a vignette), which is used to calculate an issue's severity values.

Key	Class Type	Description	Default	Required?
`severity-mapping`	String	Indicates the name of the set of severity mappings used to determine the score of each issue. The first	N/A	Yes

Key	Class Type	Description	Default	Required?
		three mappings are built-in.		

For more information about Security Report configuration, see Section 8.2.3, “Configuring the Security Report”.

6.1.2.2.4. Synopsys Software Integrity Report

The Synopsys Software Integrity Report has additional keys for specifying and defining analysis date and legal text:

```
ssir-report:
    analysis-date: 01/15/2019
    legal-text:
```

Key	Class Type	Description	Default	Required?
`analysis-date`	String	Specifies when the analysis has been completed. Dates should be entered in MM/DD/YYYY format.	N/A	Yes
`legal-text`	String	Includes legal text. Multiline text should be placed inside double quotes. (For example: "This is Multiline legal text.")	N/A	No

For more information about the Synopsys Software Integrity Report configuration, see Section 10.3.2, “Configuring the Synopsys Software Integrity Report”.

6.1.3. Unified configuration file

You can use the same `.yaml` configuration file to configure all of the reports for the same project. You can have multiple configuration files (saved under different `.yaml` file names) for different projects.

Here is what the unified configuration file looks like with all sections included.

```
################## Sections that apply to all reports #############
version:
    schema-version: 4

connection:
    url: https://coverity.example.com:8443/
    username: admin
    ssl-ca-certs:
```

```
project:
title-page:
    company-name:
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:

locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:

################## CERT report #############
cert-report:
    target-level: F

################## Synopsys Software Integrity Report #############
ssir-report:
    analysis-date: 01/15/2019
    legal-text:

################## Coverity Integrity Report #############
cir-report:
    project-description:
    project-details:
    target-integrity-level:
    high-severity-name:
    unspecified-severity-name:
    trial:
    loc-multiplier:

################## Coverity Security Report #############
security-report:
    assurance-level-score: 90
    assurance-level: AL1
    severity-mapping: Carrier Grade
    severity-mapping-description:
    custom-severity-mapping:
        modify-data: very high
        read-data: very high
        dos-unreliable-execution: very high
        dos-resource-consumption: very high
        execute-unauthorized-code: very high
        gain-privileges: very high
        bypass-protection-mechanism: very high
        hide-activities: very high
```

Part 7. Coverity Integrity Report Generation

Leaders of software development teams use Coverity Integrity Report PDF-based charts to see the current impact of Coverity Connect issues. Administrators set up and run Coverity Integrity Report to produce these reports.

For extensive web-based charts that you can print to file, see Part 4, “Coverity Policy Manager Usage ”.

Chapter 7.1. Overview

Table of Contents

7.1.1. Defect ratings 352
7.1.2. Recommended analysis configuration 353
7.1.3. Requirements 353

Software Integrity Report is included with Coverity Analysis and Coverity Connect. The Software Integrity Report tool generates a high-level assessment of the code in a C/C++ application and its components. The report:

- Rates the integrity of the code and its components in relation to industry standards.
- Identifies the number of occurrences of important classes of high-risk and medium-risk defects in the code.
- Provides an overview of defect severity ratings and triage states that developers have assigned to the defects in Coverity Connect. The report identifies the number of:
  - High-severity, unspecified, and other defects in the code and code components.
  - Outstanding, dismissed, and fixed defects in the code.
- Can be localized.

Important

In order to invoke the Coverity Integrity Report, you must be assigned a role that contains, at minimum, the following permissions:

- *Access web services*
- *View project*
- *View defects*

For more information about RBAC roles and permissions, see the Coverity Platform 2020.12 User and Administrator Guide.

7.1.1. Defect ratings

To rate your code and code components in relation to the software industry, the report relies on defect density, which is equal to the number of high-risk and medium-risk defects found in 1000 lines of code (LOC). For example, 100 defects in 100,000 LOC yields a defect density of 1 (because 100/100000 = 1 defect per 1000 LOC). The defect densities identified in the report exclude low-risk defects, as well as defects that developers dismissed as intentional in Coverity Connect. Except for the Level 3 rating, the report also excludes defects that developers report as false positives.

Table 7.1.1. Software Integrity Report levels

Level	Description
Level Not Achieved	Indicates that the target level rating criteria are not met because the software has too many unresolved static analysis defects. To achieve the target integrity level rating, more defects should be reviewed and fixed.
Level 1	The defect density is close to the average for the software industry average: < 1 defect per 1000 lines of code.
Level 2	The defect density is in the 90th percentile for the software industry: < 0.1 defect per 1000 LOC.
Level 3	The defect density is in the 99th percentile for the software industry (< 0.01 defect per 1000 LOC). Developers have marked fewer than 20% of the defects as false positives and set no defects to major severity in Coverity Connect. A Coverity audit of the false positives can alter the 20% limit.

The report contains additional information about these levels. You can view this information in a sample report after extracting the Software Integrity Report file in Chapter 7.2, *Generating a Coverity Integrity Report*.

Note

The results of checkers that developers create by using the Coverity Software Development Kit or the earlier Coverity Extend are excluded from the defect density because they do not have risk categories.

7.1.2. Recommended analysis configuration

For optimal report results, Coverity recommends that you use the following options to the `cov-analyze` command when you use Coverity Analysis for C/C++ to run an analysis on your code:

```
cov-analyze --dir <intermediate_directory> --security --concurrency
  [--enable-constraint-fpp] [--enable-fnptr]
```

For details about these options, see the `cov-analyze` command documentation in the *Coverity 2020.12 Command Reference*. Note that the options that are surrounded by square brackets are optional.

7.1.3. Requirements

Software Integrity Report version 2020.12 is compatible with Coverity Connect version 2020.12. It is incompatible with prior versions of Coverity Connect. Similarly, an older version of Software Integrity Report is incompatible with Coverity Connect version 2020.12.

Chapter 7.2. Generating a Coverity Integrity Report

In this chapter, you install the Coverity Reports, configure the `config.yaml` file, and then run the `bin/cov-generate-integrity-report` executable.

The following procedure assumes that someone has already used Coverity Analysis to analyze your code base (see Section 7.1.2, “Recommended analysis configuration”) and commit the resulting defect data to Coverity Connect. For guidance with this process, see *Coverity Analysis 2020.12 User and Administrator Guide*.

To create a Coverity Integrity Report:

1. You can download the Coverity Reports installer from the *Downloads* page in Coverity Connect.

   Alternatively, obtain the installer for Coverity Reports from the Coverity Connect installation directory (for example, `cov-reports-linux64-2020.12.sh` or `cov-reports-win64-2020.12.exe`). Run the installer on a system that can connect to the Coverity Connect server.

   The installer files are located in the Coverity Platform installation directory: `<install_dir>/server/base/webapps/downloads` directory.

   Note

   After running the installer, you can find a sample Integrity Report in `<reports installation dir>/docs/`.

2. Open the directory where the Coverity Reports were installed.

   This directory contains a configuration file and a command that you need to use in subsequent steps:

   - Configuration file: `config.yaml`
   - Command: `bin/cov-generate-integrity-report`

3. Configure `config.yaml`.

   The `config.yaml` file contains pre-populated and customizable parameters that the `bin/cov-generate-integrity-report` command uses to generate a defect report.

4. Make sure that Coverity Connect is running.

   To start Coverity Connect, you can use the following command, which is located in the Coverity Platform `<install_dir>/bin`:

   ```
   > cov-start-im
   ```

5. Run the `bin/cov-generate-integrity-report` command.

   For example, on Linux:

   ```
   > bin/cov-generate-integrity-report
   ```

For example, on Windows:

```
> bin\cov-generate-integrity-report
```

Note

If you need to create a report more than once, close all open PDF-based reports before regenerating the report.

For summary information about the `cov-generate-integrity-report` command-line options, use the `--help` option.

Chapter 7.3. Updating the Coverity Integrity Report configuration file

You can specify configuration information using a `.yaml` file. A sample `config.yaml` file is shipped with the Report Generators and installed in the `config` directory.

Here is an example `.yaml` configuration file:

```
version:
     schema-version: 5
connection:
    url: https://coverity.example.com:8443/
    username: admin
    ssl-ca-certs:
project:
title-page:
    company-name:
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:
locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:

################## Coverity Integrity Report #############
cir-report:
    project-description:
    project-details:
    target-integrity-level:
    high-severity-name:
    unspecified-severity-name:
    trial:
    loc-multiplier:
    include-low-impact:
```

Note

The Coverity Integrity Report requires the additional report configuration settings listed in the example above. This ensures that the Coverity Integrity Report results are properly included in the output.

The following table describes each key in the file:

Key	Description
`cir-report`	Declares the configuration settings for the Coverity Integrity Report.
`company-name`	Name of your company. This is a mandatory field.
`components`	An optional comma-separated list of Coverity Connect component names, which inc map names. If the components are listed here, the report will include data only for th components; for example: `Default.lib` or `Default.src`.
`connection`	The URL of the Coverity Connect instance.
`high-severity-name`	Name of the highest severity value. The default value is `Major`.
`include-low-impact`	Whether to include Low impact for calculating the defect density; this would be in ad and Medium impact items shown in the report. Specify `true` to include Low impact, exclude. Default is `false`.
`issue-cutoff-count`	Some reports display information about individual issues. These reports bound the n displayed in order to control the size of the report. This bound is called the issue cut for CVSS, Security, PCIDSS, MobileOwasp, and Owasp2017 reports. Default value 10000 for Security report.
`issue-kind`	An optional comma-separated list of Coverity Connect issue kinds. If issue kinds are report will include only issues of the listed kinds. The possible values for `issue-kind` are as follows: • `Quality` • `Security` The following line is an example of using this option: `issue-kind: Quality`
`locale`	Locale of the report. Valid values are the following: `en_US`, `ja_JP`, `ko_KR`, and `zh_` `en_US`.
`loc-multiplier`	LOC multiplier for the number of lines of code that have been inspected. Default val
`logo`	Optional path to a logo file for your company. Valid image types are bmp, gif, jpg, an maximum allowed image size is 210 pixels wide by 70 pixels high. Note that backsla path must be doubled.
`on-cert-trust`	Allows users to trust self-signed certificates sent by Coverity Connect. There are two `trust` or `distrust`. Use `trust` to accept, use, and store a self-signed certificate `distrust` to reject connections from servers using self-signed certificates. The def
`organizational-unit-name`	Name of your division, group, team or other organizational unit. This is a mandatory
`organizational-unit-term`	Organizational unit term (e.g., division, group, team). This is a mandatory field.
`prepared-for`	Name of the entity for which the report was prepared. This is a mandatory field.
`prepared-by`	Name of the entity that prepared the report. This is a mandatory field.

Key	Description
`project`	Name of the Coverity Connect project.
`project-contact-email`	Project contact email address. It is used for the following reports: CIR, CVSS, PCID and OWASP2017. This is a mandatory field.
`project-description`	A short description of the project. The project description defaults to its description i if there is one available.
`project-details`	This field lists the details of the project.
`project-name`	Name of the software development project. May be distinct from the Coverity Conne This is an optional field.
`project-version`	Lists the project version number. This is a mandatory field.
`snapshot-date`	Retrieves the most recent snapshot of each stream in the project whose date is less the given date. Format is "DD/MM/YYYY".
`snapshot-id`	Retrieves the defects of a specific snapshot id, instead of using the latest snapshot associated with the project.
`target-integrity-level`	Sets the target integrity level. The default value is 1. The following target integrity levels are available: • 1: < 1 defect per thousand lines of code • 2: < .1 defects per thousand lines of code • 3: < .01 defects per thousand lines of code, and other requirements
`title-page`	Describes the fields in the title page of the report.
`trial`	This setting enables the trial flag. Uses the value `true` if page three of the report is generated. (Page 3 contains severity data which is not relevant for projects that do The default value is `false`.
`username`	Coverity Connect username. Password or other authentication key
`unspecified-severity-name`	Name of the unspecified severity value. The default value is `Unspecified`.
`version`	The version of the `.yaml` file's schema.

For more information about schema configurations, see Chapter 6.1, *Configuring Report Generators*

Part 8. Coverity Security Report

Table of Contents

8.1. Security Report Overview 360

  8.1.1. Basic Workflow 360

  8.1.2. Interpreting the Security Report 361

8.2. Working with the Security Report Generator 365

  8.2.1. Installing the Report Generator 365

  8.2.2. Connecting to Coverity Connect 366

  8.2.3. Configuring the Security Report 367

  8.2.4. Generating a Report 372

  8.2.5. Customizing Security reports 373

Chapter 8.1. Security Report Overview

Table of Contents

8.1.1. Basic Workflow 360
8.1.2. Interpreting the Security Report 361

The Security Report generator uses analysis results for a Coverity Connect project to evaluate the analyzed codebase. Based on this evaluation, it creates a Security Report. The codebase is evaluated against a policy, a set of rules or standards for determining pass or fail. The policy has 4 elements, and each element must pass for the policy to pass. The result for each element is presented in the Scorecard in the Executive Summary of the report. The elements include:

- **Security score** represents the severity levels of the issues found as a numerical value from 0 to 100, with issues of the highest reported severity level having the greatest impact on lowering the score.
- **OWASP Top 10 Count** specifies the number of security issues found that are included in OWASP's top ten count.The Top 10 from the year 2017 will be used.

  The OWASP (Open Web Application Security Project) Foundation publishes a report of the most critical web application security flaws, in a ranked order based on input from a worldwide group of security experts.
- **CWE/SANS Top 25 Count** specifies the number of security issues found that are included in the CWE/SANS Top 25 list.

  CWE (Common Weakness Enumeration) is a software community project responsible for creating a catalog of software weaknesses and vulnerabilities. The CWE/SANS Top 25 is a list of weaknesses, taken from the CWE, that are deemed to be the most widespread and critical errors that can lead to serious software vulnerabilities.
- **Analysis date** specifies the date when the analysis produced the data upon which the report is based.

The severity of the issues found during analysis is determined by a *severity mapping* that maps security flaws to severity levels. The report generator can use one of three default severity mappings, or it can use a custom severity mapping that you define when you configure the report generator.

In addition to providing a summary of findings, the report also includes sections that list the remediation actions required and increasingly detailed views of the issues found. The report also provides a detailed breakdown and cross references between technical findings and analysis results.

This chapter describes the workflow needed for generating a Security Report and explains how you interpret report findings.

8.1.1. Basic Workflow

The basic workflow for working with security reports is as follows:

1. Install the Security Report generator using the Coverity Reports Installer. See Installing the Security Report Generator for more information.
2. Set up a connection to Coverity Connect.

   You need to do this only the first time you run the report generator. See Connecting to Coverity Connect for more information.
3. Configure the report. This allows you to select the severity mapping you want to use and to specify other customization information.

   See Configuring a Security report for more information.
4. Generate the report.

8.1.2. Interpreting the Security Report

The report is divided into six basic sections that describe the issues found in increasing amount of detail:

- **Executive Summary** provides tabular and graphic summary information for the issues found.
- **Action Items** explains how the code base was evaluated and provides a summary of recommended remediation actions.
- **Security Details** shows the number of issues associated with each Technical Impact category.
- **Analysis Details** shows the number of issues associated with each OWASP Top 10 category and each CWE/SANS Top 25 category.
- **Detailed Issues Ranked by Severity** lists all issues, the name of the source file for that issue and the line number where the issue can be found. It also describes the Technical Impact associated with each group of issues and recommends remediation actions.
- **Methodology** describes basic elements associated with report output.

To interpret report results, you must understand the basic categories used to classify issues and how report output might vary depending on the severity mapping used when you configure the report generator. The following sections describe this basic information.

8.1.2.1. Severity mapping: Mapping technical defects to severity levels

A *severity mapping* is a mapping that determines the *severity* level of a given *technical impact* associated with a software issue. *Technical impacts* categorize the negative effects that can occur if an attacker exploits a particular weakness in the target software. Not all weaknesses are exploitable, and not all exploitable weaknesses are easy to exploit.

The Security Report is preconfigured with three built-in severity mappings (Carrier grade, Web application, and Desktop application): the `Carrier grade` severity mapping is the most stringent, and the `Desktop application` is the least stringent. When you configure the report generator, you can also specify that it use a custom severity mapping in which you can redefine severity levels for each Technical Impact category.

Technical Impacts are divided into eight categories (as defined by CWE):

Impact type	**Meaning**
Modify data	A weakness that could allow an attacker to modify memory or files on the host computer.
Read data	A weakness that could allow an attacker to read data that is not intended to be accessible to a user of the application.
Denial of service, unreliable execution	A weakness that can lead to crashes, freezes, and other malfunctions that can make the application unavailable to users.
Denial of service, resource consumption	A weakness that causes the application to use excessive CPU, memory, or storage resources, which degrades application performance.
Execute unauthorized code	A weakness that might allow an attacker to cause the application to execute code within or outside of the application in unintended ways.
Gain privileges	A weakness that might allow an attacker to perform privileged operations that are not intended to be available to a user of the application.
Bypass protection mechanism	A weakness that might allow an attacker to defeat or skirt protections that keep application users from reading, writing, or executing unauthorized resources.
Hide activities	A weakness that might allow an attacker to avoid detection.

The severity mapping associates each Technical Impact with one of the following severity levels:

- Informational (not included when calculating the Security Score)
- Very Low
- Low
- Medium
- High
- Very High

Issues that are found by Coverity Analysis might be associated with a CWE (Common Weakness Enumeration) ID number. CWE ID numbers refer to CWE records, each of which might have one or more Technical Impacts. If an issue has an **Issue Kind** of **Quality** or **Security** and its CWE record contains one or more of the eight types of Technical Impacts, that issue will be included in the Security Report.

For an issue where its CWE ID maps to more than one of the eight technical impact values, a single technical impact value will be assigned to the issue, where the highest relevant severity level will determine which technical impact value gets assigned, with ties for the highest severity level being broken arbitrarily.

Some issues found might not have a CWE ID; in that case, that issue cannot be associated with a Technical Impact value and cannot be included in the Security Score. In this case, the issue will be included in the **Issues Without CWE Numbers** count of a Security Report's "Additional Quality Measures" section. That section might also include issues that do have CWE IDs if the given CWE ID cannot be mapped to at least one of the eight Technical Impact categories.

Note

Setting the same severity level for different tehnical impacts might result in an issue being associated with different technical impacts from one generation of the report to the next. This should not affect the overall security score.

8.1.2.1.1. Creating a custom severity mapping

The first time you use the Security Report Generator, you should use one of the standard severity mappings to understand how the report is calculated. Thereafter, you might want to create a custom severity mapping. Consider the context in which your application is used, and select the severity of each technical impact with regard to the likelihood of its having a damaging effect. Because it is difficult to estimate the likelihood of a particular weakness being activated, likelihood should not be considered when choosing the severity level.

You can create a Security Report Plug-in that allows you to reuse customised severity mapping. For more information, see "Coverity Reports Customization Plugins."

8.1.2.1.2. Security scores, severity mappings, and assurance levels

The Security Report application analyzes the issues returned by Coverity Connect and calculates a Security Score, based on the severity mapping selected when you configure the report. The Security Report Generator compares this value to the required **Assurance Level** and determines if the Security Score passes or fails. For more information, see the next section.

When you configure the report generator, you select the **Assurance Level**. There are four Assurance Levels, representing Security Scores of greater than or equal to 60, 70, 80, and 90. When choosing the Assurance Level, consider the potential for damage to life, property, or reputation. An application with high damage potential should have a high Assurance Level.

8.1.2.2. Understanding scorecards

The report opens with an Executive Summary that features a Scorecard showing the target for each policy category, the actual result, and whether the codebase has passed for that category. Policy categories are described next; for detailed information on how each value is calculated, see the Methodology section of a generated report.

Security Score

This is a numerical value (0 to 100) calculated from the severity mapping specified when the report is configured. The score is compared to the configured Assurance Level to determine pass or fail.

Severity levels are used to determine the security score: possible severity levels are `Very High`, `High`, `Medium`, `Low`, and `Very Low`. The highest severity level that has at least one issue associated with it will greatly influence the security score. Additional issues with a relatively higher severity level will have a greater impact on reducing the security score than will additional issues with a relatively lower severity level. As such, it's important to address issues with the highest severity level.

While the full range of a possible security score is from 0 to 100, a project would need to contain more than 30 000 `Very High` severity level issues to receive a score lower than 30. Meanwhile, a project with a highest severity level of `Very Low` would need to contain more than 30 000 `Very Low` severity level issues to receive a score lower than 70.

To give some further context, consider the standard Target Assurance Levels plus their corresponding Target Security Score values of AL1 (90), AL2 (80), AL3 (70), and AL4 (60) relative to the highest severity level that has at least one issue associated with it.

- If `Very High` severity level issues exist, it will be nearly impossible to achieve AL3 (70), and it will be quite a challenge to achieve AL4 (60).
- If all of the `Very High` severity level issues have been addressed, but at least one `High` severity level issue exists, it will be nearly impossible to achieve AL2 (80), and it will be a reasonable challenge to achieve AL3 (70), with AL4 (60) being within easier reach.

  If all of the `Very High` and `High` severity level issues have been addressed, but at least one `Medium` severity level issue exists, it will be nearly impossible to achieve AL1 (90) and quite challenging to achieve AL2 (80), while AL3 (70) is more likely to be within reach, and AL4 (60) should be a relatively easy target to reach.

OWASP Top 10
This is a list of prioritized security weaknesses relating to web application security. If the policy prohibits these weaknesses, the target would be zero.

CWE/SANS Top 25
This is a list of software weaknesses that are thought to be widespread and critical. If the policy prohibits these weaknesses, the target would be zero.

Analysis Date
The codebase must have been analyzed within the last 30 days for this item to pass.

Chapter 8.2. Working with the Security Report Generator

Table of Contents

8.2.1. Installing the Report Generator 365
8.2.2. Connecting to Coverity Connect 366
8.2.3. Configuring the Security Report 367
8.2.4. Generating a Report 372
8.2.5. Customizing Security reports 373

The following sections explain how you install the Security Report generator and how you configure it.

8.2.1. Installing the Report Generator

The Security Report Generator is installed separately from Coverity Connect. You can obtain the Coverity Reports installer from the *Downloads* page in Coverity Connect. The Coverity Reports installer installs Security Report.

1. In Help → Downloads, select the Coverity Reports installer that is appropriate for your system.

2. Save the installer application to a folder in your system.

3. You have the option to use the install wizard or to install from the command line.

   - To use the wizard, launch the installer, and follow the instructions in the wizard.

   - To use the command line, proceed to the next step.

4. To run the installer from the command line, execute one of the following commands, using either quiet mode (`-q`) or console mode (`-c`):

Operating System	**Command**
Windows	`start /wait "" cov-reports-win64-<version_number>.exe -c`
Windows - quiet mode	`cov-reports-win64-<version_number>.exe -q -dir C:/target-dir`
Linux	`./cov-reports-linux64-<version_number>.sh -c`
Linux - quiet mode	`./cov-reports-linux64-<version_number>.sh -q -dir ~/target-dir`

   If you do not specify a `target-dir` for the installation, a default installation directory is used. The directory name is shown at runtime.

   You can use the following parameters in the command line:

Parameter	**Action**
`-c`	Run in console mode. User interaction is performed in the terminal window where the installer (or uninstaller) is invoked.

Parameter	**Action**
`-console`	On Windows, use with `-q` to open a console window to display output in quiet mode.
`-dir [directory]`	Set the installation directory in quiet mode.
`-Dname=value`	Set system properties.
`-h`	Display help.
`-manual`	On Windows, in GUI mode only, manually select a Java Runtime Environment.
`-overwrite`	Overwrite all files in quiet mode.
`-q`	Run in unattended (quiet) mode. There is no user interaction, and installation is performed automatically using default values.
`-splash [title]`	Display a progress bar in quiet mode.
`-varfile`	Use a response file.

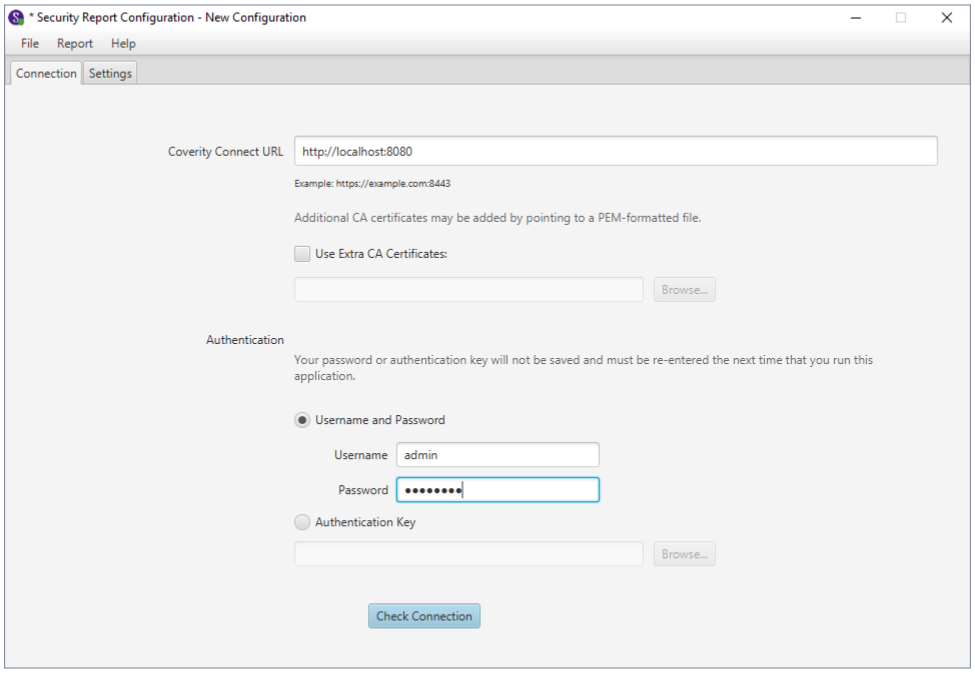
8.2.2. Connecting to Coverity Connect

In `<installation-directory>/bin`, locate the application executable, `cov-security-report`. The first time you run the application, you must set up the connection to your instance of Coverity Connect, which contains the issue information to include in the report.

1. Double-click on the application icon to launch it. It starts in *New Configuration* mode.

   If you have a saved configuration file, double-click the configuration file to launch the application with that configuration.

2. On the *Connection* tab, enter the *Host Name* and *Port Number*. If your Coverity Connect administrator has enabled SSL, enter the HTTPS port number associated with your Coverity Connect host. The default HTTPS port is 8443.



Coverity Connect URL: http://localhost:8080

Example: https://example.com:8443

Additional CA certificates may be added by pointing to a PEM-formatted file.

☐ Use Extra CA Certificates:

Authentication

Your password or authentication key will not be saved and must be re-entered the next time that you run this application.

● Username and Password

Username: admin

Password: ••••••••

○ Authentication Key

3. Check the Coverity Connect URL to make sure it is correct.
4. If SSL is used with the connection, select *Secured using SSL*.

   Note

   If an additional CA certificate is needed, select *Use Extra CA Certificate*, and click **Browse** to upload the file.

5. Enter the *Username* and *Password* for Coverity Connect.
6. Click **Check Connection** to test the connection to Coverity Connect.

8.2.3. Configuring the Security Report

You can configure a security report using the command line or using the GUI.

- If you use the GUI, you can save the resulting file as a `.yaml` file.
- Otherwise, you can modify the `.yaml` file described in the next section and use it from the command line or within a script.

The `cov-generate-security-report` command supports both types of config files.

8.2.3.1. Command-line configuration

You can specify configuration information using a `.yaml` file. A sample `config.yaml` file is shipped with the report generator and installed in the `config` directory.

Here is an example `.yaml` configuration file:

```
################## Sections that apply to all reports #############
version:
    schema-version: 5

connection:
    url: https://coverity.example.com:8443/
    username: admin
    ssl-ca-certs:

project:
title-page:
    company-name:
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:

locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:

################## Coverity Security Report #############
security-report:
    assurance-level-score: 90
    assurance-level: AL1
    severity-mapping: Carrier Grade
    severity-mapping-description:
    custom-severity-mapping:
        modify-data: very high
        read-data: very high
        dos-unreliable-execution: very high
        dos-resource-consumption: very high
        execute-unauthorized-code: very high
        gain-privileges: very high
        bypass-protection-mechanism: very high
        hide-activities: very high
```

Note

The Security Report requires the additional report configuration settings listed in the example above. This ensures that the Security Report results are properly included in the output.

The following table describes each key in the file:

Key	Description
`assurance-level`	A level indicating the minimum acceptable score for the report to be considered pas values are the following: AL1, AL2, AL3, and AL4. Default value is AL1.
`assurance-level-score`	There are four Assurance Levels, representing Security Scores of greater .than or e and 90. When choosing the Assurance Level, consider the potential for damage to l reputation. An application with high damage potential should have a high Assurance value is 90.
`company-name`	Name of your company. This is a mandatory field.
`connection`	The URL of the Coverity Connect instance.
`components`	An optional comma-separated list of Coverity Connect component names, which inc map names. If the components are listed here, the report will include data only for t components; for example: `Default.lib` or `Default.src`.
`custom-severity-mapping`	If the severity mapping is set to `Custom`, then update the `custom-severity-map` settings and values. The following values are available: `very high`, `high`, `medium`, `low`, and `very l` `high`.
`issue-cutoff-count`	Some reports display information about individual issues. These reports bound the displayed in order to control the size of the report. This bound is called the issue cut for CVSS, Security, PCIDSS, MobileOwasp, and Owasp2017 reports. Default value 10000 for Security report. You can increase the maximum of 10,000 to a number as large as 50,000 by setting `ISSUE_CUTOFF_COUNT` environment variable to the desired value.
`issue-kind`	An optional comma-separated list of Coverity Connect issue kinds. If issue kinds are report will include only issues of the listed kinds. The possible values for `issue-kind` are as follows: • `Quality` • `Security` The following line is an example of using this option: `issue-kind: Quality`
`locale`	Locale of the report. Valid values are the following: `en_US`, `ja_JP`, `ko_KR`, and `zh_` `en_US`.
`logo`	Optional path to a logo file for your company. Valid image types are bmp, gif, jpg, ar maximum allowed image size is 210 pixels wide by 70 pixels high. Note that backsla path must be doubled.
`on-cert-trust`	Allows users to trust self-signed certificates sent by Coverity Connect. There are tw `trust` or `distrust`. Use `trust` to accept, use, and store a self-signed certificate `distrust` to reject connections from servers using self-signed certificates. The def

Key	Description
`organizational-unit-name`	Name of your division, group, team or other organizational unit. This is a mandatory
`organizational-unit-term`	Organizational unit term (e.g., division, group, team). This is a mandatory field.
`prepared-for`	Name of the entity for which the report was prepared. This is a mandatory field.
`prepared-by`	Name of the entity that prepared the report. This is a mandatory field.
`project`	Name of the Coverity Connect project.
`project-contact-email`	Project contact email address. It is used for the following reports: CIR, CVSS, PCID and OWASP2017. This is a mandatory field.
`project-name`	Name of the software development project. May be distinct from the Coverity Conne This is an optional field.
`project-version`	Lists the project version number. This is a mandatory field.
`severity-mapping`	The name of the set of severity mappings used to determine the score of each issue documentation for a description of the severity mapping. The first three mappings a indicates that the mapping identified by `custom-severity-mapping` and `severi` `description` should be used. Valid values are `Carrier Grade`, `Web applicat` `application`, and `Custom`. Default value is `Carrier Grade`.
`security-report`	`severity-mapping-description` Optional descriptive text for custom severity mapping.
`snapshot-date`	Retrieves the most recent snapshot of each stream in the project whose date is less the given date. Format is "DD/MM/YYYY".
`snapshot-id`	Retrieves the defects of a specific snapshot id, instead of using the latest snapshot associated with the project.
`title-page`	Describes the fields in the title page of the report.
`username`	Coverity Connect username. Password or other authentication key
`version`	The version of the `.yaml` file's schema.
`custom-severity-mapping`	If the severity mapping is set to `Custom`, then this entry introduces some specific cu whose descriptions follow. Each custom field can be set to one of the following values: `very high`, `high`, `me` `low`, or `informational`.
`dos-unreliable-execution`	Custom field. Indicates when a weakness creates a denial of service due to unrelia application. Default value is `very high`
`dos_resource-consumption`	Custom field. Indicates when a weakness creates a denial of service due to excessi consumption by the application. Default value is `very high`
`execute-unauthorized-code`	Custom field. Indicates when an attacker tries to execute code that it does not have execute. Default value is `very high`
`gain-privileges`	Custom field. Indicates when an attacker tries to gain privileges that should not be a Default value is `very high`

Key	Description
`bypass-protection-mechanism`	Custom field. Indicates that an attacker has tried to bypass application or system pr mechanisms. Default value is `very high`
`hide-activities`	Custom field. Indicates when an attacker tries to modify data in the application. Defa `high`

For more information about schema configurations, see Chapter 6.1, *Configuring Report Generators*

8.2.3.2. GUI configuration

To configure a new security report, you use one of four panes to specify information:

- **Coverity Connect** to specify connection information.
- **Assurance Level** to specify the minimum passing score for the project.
- **Severity Mapping** to select one of the default severity mapping or to specify a custom severity mapping.
- **Customization** to specify company and organizational unit information; you can also select the locale for the report.

Note

Some reports display information about individual issues. These reports bound the number of issues displayed in order to control the size of the report. This bound is called the *issue cutoff count*. It is used for CVSS, Security, PCIDSS, MobileOwasp, and Owasp2017 reports. Default value is 200. Maximum is 10000 for Security report.

You can increase the maximum of 10,000 to a number as large as 50,000 by setting the `ISSUE_CUTOFF_COUNT` environment variable to the desired value.

To configure a report:

1. Select Settings → Coverity Connect.
2. Select **Coverity Connect Project**.

   The projects available through the connection to Coverity Connect are displayed in the drop-down list. If the list is empty, click *Refresh*.
3. In the *Assurance Level* pane, select the minimum passing score for the project.
4. In the *Severity mapping* pane, select a severity mapping. Default severity mappings are read-only. If you select *Custom*, you can edit the *Severity* level of each *Technical Impact*.
5. In the *Customization* pane, enter information to customize the report. The names and terms are used throughout the report, and the company name and logo are featured on the cover page.

Note

The *Project* mentioned here refers to the corporate project name, and should not be confused with the Coverity Connect project.

The company logo is optional.

You can use this pane to set the locale for your report

6. Click File → Save to save the configuration for future use.
7. You can now generate a report as described in the following section.

8.2.4. Generating a Report

After you have saved your report configuration, you can generate a report by clicking **Create Report**.

The configuration for a report can be saved as a `.yaml` file. This document can be used to regenerate the report, with the same settings, when the analysis data is updated. The report can be regenerated through the GUI or with the command line. The following sections describe each method.

8.2.4.1. Generating a previously configured report using the GUI

To regenerate a report through the GUI:

1. Launch the `cov-security-report` application.
2. Click File → Open Configuration, and select the saved `.yaml` configuration file.
3. Click **Create Report** to generate the report.

8.2.4.2. Generating a previously configured report from the command line

You can generate a report using a previously saved configuration by using the headless mode. On the command line, use `cov-generate-security-report` with the following settings.

```
cov-generate-security-report <config file> [--auth-key-file <filename>][--
company-logo <logo>] [--help] [--includeCusp] [--locale <locale>] [--on-new-
cert trust|distrust] [--output <filename>] [ --password <spec>] [--project
<project-name>] [--user <username>]
```

Parameter	**Meaning**
`--auth-key-file <filename>`	Coverity Connect authentication key file.
`--company-logo <logo>`	Path to company logo file.
`<config-file>`	Security Report `.yaml` configuration file.

Parameter	Meaning
`--help`	Displays this help message and exits.
`--includeCusp`	Include CWE/SANS Cusp in SANS top25 list.
`--locale <locale>`	Locale of the report: one of `en_US` (English, default), `ja_JP` (Japanese), `ko_KR` (Korean), or Simplified Chinese (`zh_CN`).
`--on-new-cert trust|distrust`	When the value is `distrust` (the default), an attempt to connect to the server using SSL might fail if the server provides an untrusted self-signed certificate.
`--output <output-file>`	Name of the PDF output file. The file will be replaced if present.
`--password <spec>`	Coverity Connect password specifier. The password <<`spec`>> argument is required, and has four forms: console The password is read from the keyboard without echoing to the console. file:<<`filename`>> The password is read from the first line of the file <filename>. file:- The password is read from standard input. This is for use with pipes and redirection, not for keyboard input. env:<<`variable`>> The password is read from the environment variable <<`variable`>>.
`--project <project-name>`	Name of the project in Coverity Connect.
`--user <user-name>`	Coverity Connect user name.

8.2.5. Customizing Security reports

You can customize reports in the following ways:

- You can localize reports.
- You can specify one of several additional output formats: xml, json, or csv.
- You can customize reports using Java plugins

8.2.5.1. Localizing reports

You can now localize Security reports for Japanese, Korean, and Chinese. When you configure the report, select your locale from the **Report's Locale** drop-down list in the **Customization** pane.

Important

You must have the same locale configured in Coverity Connect as you set for your report. Otherwise, portions of the report will be presented in the user's locale rather than the desired one. (Use the drop down list from **Admin User > Preferences > Locale** to select the desired locale.)

8.2.5.2. Using system environment variables

In addition to the PDF report output, the Security Report generator can also output three additional files:

- **A .csv (Severities CSV) file** that provides a mapping from CID and CWE values to the assigned Technical Impact and Severity Level values that are found in the Security Details' Technical Impact Table of a generated security report. This information is presented in a form that can be easily imported into an Excel spreadsheet.
- **An .xml file** that provides the report output in a form that makes it easy for you to include reported data in your own documentation or reports.
- **A .yaml file** that lists the CWE IDs with their respective severity values. Each line will list a name and value pair, where the CWE ID (an integer value) is listed next to name of the severity. The report generation will fail either because the file is invalid or because the user does not have proper read/write permissions.

You must use environment variables to specify which of these files you want the report generator to create and where it should be stored.

For example, on Windows, you'd set the environment variable like this:

```
set WRITE_REPORT_XML=<filename>
```

On Linux, you'd set the environment variable like this:

```
export WRITE_REPORT_XML=<filename>
```

For the Security Report, the following five environment variables are available:

- `IGNORE_ISSUES_DETAIL=true`: This variable produces a PDF report that only includes high level defect and issue data, omitting issue details. If this environment variable is set to `true`, then the *Detailed Issues Ranked by Severity* section is omitted from the generated report.
- `WRITE_ISSUES_JSON`: This variable writes defect and issue data to the JSON output file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.
- `WRITE_REPORT_XML`: This variable writes properties from the report's configuration to the XML output file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.
- `WRITE_SEVERITIES_CSV`: This variable writes severities and technical impacts of each CID to a CSV file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.

- `WRITE_CWES_YAML`: This variable writes CWE Partition data to the YAML output file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.

Part 9. MISRA Report

Table of Contents

9.1. MISRA Report Overview 377
    9.1.1. MISRA Compliance 377
9.2. Installing the MISRA Report Generator 379
    9.2.1. Connecting to Coverity Connect 380
    9.2.2. Configuring a MISRA Report 381
    9.2.3. Using system environment variables 383
    9.2.4. Generating a previously configured report 384
    9.2.5. Generating a report from the command line 385
    9.2.6. Localizing reports 386

Chapter 9.1. MISRA Report Overview

Table of Contents

9.1.1. MISRA Compliance 377

The MISRA Report uses analysis results for a project in Coverity Connect to evaluate a codebase and create a formatted report. The codebase is evaluated against a policy, which is a set of rules for determining whether the project's issues are consistent with MISRA compliance. The result is presented in the MISRA Compliance section in the Executive Summary of the report.

Certain elements of the report can be customized by means of Java plugins. For more information, see "Coverity Reports Customization Plugins" in the *Coverity Platform 2020.12 User and Administrator Guide*.

Important

To support report generation, you must use the `--coding-standard-config` option to the `cov-analyze` command. This option provides the path to a configuration file for a coding standard to run as part of the analysis. The configuration file can specify one of several MISRA standards; for example,

```
{   version : "2.0",
    standard : "misrac2012",
    title: "your_title_here",
    deviations : []
}
```

9.1.1. MISRA Compliance

The MISRA Compliance section of the report states whether the project was found to be MISRA compliant or not, and shows the number of violations in each of the following categories. For detailed information on how each scorecard element is calculated, see the *Methodology* section of a generated report.

Mandatory violation count
: This is the number of violations found that fall in the MISRA Mandatory category.

Required violation count
: This is the number of violations found that fall in the MISRA Required category.

Document violation count
: This is the number of violations found that fall in the MISRA Document category.

Advisory violation count
: This is the number of violations found that fall in the MISRA Advisory category.

Additional Quality Measures
: Additional measures are not considered in evaluating MISRA compliance, but are provided for informational purposes. They include:

- Issues marked "False Positive" or "Intentional"
- Non-MISRA Issues
- The codebase must have been analyzed within the last 30 days

Note

The way in which the MISRA report counts violations has changed. Previously, all violations of a given rule, were counted on the basis of the rule violated not on the basis of the number of occurrences. For example, if we detected 1000 violations of rule x, this was reported as 1 for the same instance. We now report 1000 to paint a realistic picture and to comply with MISRA standards.

Chapter 9.2. Installing the MISRA Report Generator

Table of Contents

9.2.1. Connecting to Coverity Connect 380
9.2.2. Configuring a MISRA Report 381
9.2.3. Using system environment variables 383
9.2.4. Generating a previously configured report 384
9.2.5. Generating a report from the command line 385
9.2.6. Localizing reports 386

The MISRA Report Generator is installed separately from Coverity Connect. You can obtain the Coverity Reports installer from the *Downloads* page in Coverity Connect. The Coverity Reports installer installs MISRA Report.

1. In Help → Downloads, select the Coverity Reports installer that is appropriate for your system.
2. Save the installer application to an appropriate folder in your system.
3. You have the option to use the install wizard or to install from the command line. To use the wizard, launch the installer, and follow the instructions in the wizard.

   To use the command line, proceed to the next step.
4. To run the installer from the command line, execute one of the following commands, using either quiet mode (`-q`) or console mode (`-c`):

Operating System	Command
Windows	`start /wait "" cov-reports-win64-<version_number>.exe -c`
Windows - quiet mode	`cov-reports-win64-<version_number>.exe -q -dir C:/target-dir`
Linux	`./cov-reports-linux64-<version_number>.sh -c`
Linux - quiet mode	`./cov-reports-linux64-<version_number>.sh -q -dir ~/target-dir`

   If you do not specify a `target-dir` for the installation, a default installation directory is used. The directory name is shown at runtime.

   You can use the following parameters in the command line:

Parameter	Action
`-c`	Run in console mode. User interaction is performed in the terminal window where the installer (or uninstaller) is invoked.
`-console`	On Windows, use with `-q` to open a console window to display output in quiet mode.
`-dir [directory]`	Set the installation directory in quiet mode.

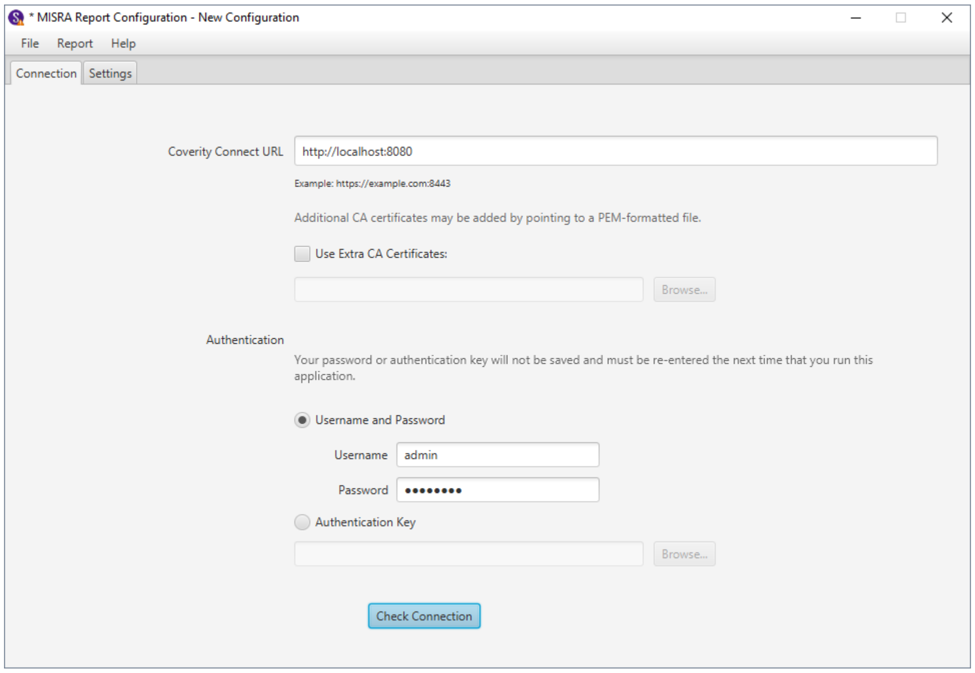
Parameter	Action
`-Dname=value`	Set system properties.
`-h`	Display help.
`-manual`	On Windows, in GUI mode only, manually select a Java Runtime Environment.
`-overwrite`	Overwrite all files in quiet mode.
`-q`	Run in unattended (quiet) mode. There is no user interaction, and installation is performed automatically using default values.
`-splash [title]`	Display a progress bar in quiet mode.
`-varfile`	Use a response file.

9.2.1. Connecting to Coverity Connect

In `<installation-directory>/bin`, locate the application executable, `cov-misra-report`. The first time you run the application, you set up the connection to your instance of Coverity Connect, which contains the issue information to include in the report.

1. Double-click on the application icon to launch it. It starts in *New Configuration* mode. Alternatively, double-click on a previously-created configuration file to launch the application with that configuration.

2. On the *Connection* tab, enter the *Host Name* and *Port Number*. If your Coverity Connect administrator has enabled SSL, enter the HTTPS port number associated with your Coverity Connect host. The default HTTPS port is 8443.



Coverity Connect URL: http://localhost:8080

Example: https://example.com:8443

Additional CA certificates may be added by pointing to a PEM-formatted file.

☐ Use Extra CA Certificates:

Authentication

Your password or authentication key will not be saved and must be re-entered the next time that you run this application.

● Username and Password

Username: admin

Password: ••••••••

○ Authentication Key

3. Check the Coverity Connect URL to make sure it is correct.

4. If SSL is used with the connection, select *Secured using SSL*.

   Note

   If an additional CA certificate is needed, select *Use Extra CA Certificate*, and click **Browse** to upload the file.

5. Enter the *Username* and *Password* for Coverity Connect.

6. Click **Check Connection** to test the connection to Coverity Connect.

9.2.2. Configuring a MISRA Report

To configure a MISRA Report:

1. Select Settings → Coverity Connect.

2. Select **Coverity Connect Project**. The projects available through the connection to Coverity Connect are displayed in the drop-down list.

   Note

   Before creating a report, make sure that the MISRA project has been analyzed with Coverity Analysis version 2020.12 prior to commit. MISRA analysis in previous versions will not work with this report.

3. In the *Customization* pane, enter information to customize the report. The names and terms are used throughout the report, and the company name and logo are featured on the cover page. The company logo is optional.

   Note

   The *Project* mentioned here refers to the corporate project name, and should not be confused with the Coverity Connect project.

4. Click File → Save to save the `.yaml` configuration file for future use.

   Note

   This document can be used to regenerate a report with the same settings whenever the analysis data is updated.

5. Click **Create Report** to generate the report.

9.2.2.1. Command-line configuration

You can specify configuration information using a `.yaml` file. A sample `config.yaml` file is shipped with the report generator and installed in the `config` directory.

The following shows a sample `.yaml` file:

```
################## Sections that apply to all reports #############
version:
    schema-version: 5

connection:
    url: https://coverity.example.com:8443/
    username: admin
    ssl-ca-certs:

project:
title-page:
    company-name:
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:

locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:
```

Note

The MISRA Report does not require any additional report settings to be configured.

The following table describes each key in the file.

Key	Description
`company-name`	Name of your company. This is a mandatory field.
`connection`	The URL of the Coverity Connect instance.
`components`	An optional comma-separated list of Coverity Connect component names, which inc map names. If the components are listed here, the report will include data only for t components; for example: `Default.lib` or `Default.src`.
`issue-cutoff-count`	Some reports display information about individual issues. These reports bound the displayed in order to control the size of the report. This bound is called the issue cu for CVSS, Security, PCIDSS, MobileOwasp, and Owasp2017 reports. Default value 10000 for Security report.
`issue-kind`	An optional comma-separated list of Coverity Connect issue kinds. If issue kinds are report will include only issues of the listed kinds. The possible values for `issue-kind` are as follows:

Key	Description
	• `Quality` • `Security` The following line is an example of using this option: `issue-kind: Quality`
`locale`	Locale of the report. Valid values are the following: `en_US`, `ja_JP`, `ko_KR`, and `zh_` `en_US`.
`logo`	Optional path to a logo file for your company. Valid image types are bmp, gif, jpg, ar maximum allowed image size is 210 pixels wide by 70 pixels high. Note that backsla path must be doubled.
`on-cert-trust`	Allows users to trust self-signed certificates sent by Coverity Connect. There are tw `trust` or `distrust`. Use `trust` to accept, use, and store a self-signed certificate `distrust` to reject connections from servers using self-signed certificates. The def
`organizational-unit-name`	Name of your division, group, team or other organizational unit. This is a mandatory
`organizational-unit-term`	Organizational unit term (e.g., division, group, team). This is a mandatory field.
`prepared-for`	Name of the entity for which the report was prepared. This is a mandatory field.
`prepared-by`	Name of the entity that prepared the report. This is a mandatory field.
`project`	Name of the Coverity Connect project.
`project-contact-email`	Project contact email address. It is used for the following reports: CIR, CVSS, PCID and OWASP2017. This is a mandatory field.
`project-name`	Name of the software development project. May be distinct from the Coverity Conne This is an optional field.
`project-version`	Lists the project version number. This is a mandatory field.
`snapshot-date`	Retrieves the most recent snapshot of each stream in the project whose date is less the given date. Format is "DD/MM/YYYY".
`snapshot-id`	Retrieves the defects of a specific snapshot id, instead of using the latest snapshot associated with the project.
`title-page`	Describes the fields in the title page of the report.
`username`	Coverity Connect username. Password or other authentication key
`version`	The version of the `.yaml` file's schema.

For more information about schema configurations, see Chapter 6.1, *Configuring Report Generators*

9.2.3. Using system environment variables

To use a system environment variable for your report generator, write the path and filename where you'd like the output file to be created and stored.

On Windows, you'd set the environment variable like this:

```
set WRITE_REPORT_XML=<filename>
```

On Linux, you'd set the environment variable like this:

```
export WRITE_REPORT_XML=<filename>
```

For the MISRA Report Generator, you can use these two environment variables:

- `WRITE_ISSUES_JSON`: This variable writes defect or issue data to the JSON output file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.
- `WRITE_REPORT_XML`: This variable writes properties from the report's configuration to the XML output file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.

9.2.4. Generating a previously configured report

The report can either be regenerated through the GUI or via command line.

To regenerate a report through the GUI:

1. Launch the `cov-misra-report` application.
2. Click File → Open Configuration, and select the configuration file that was previously made.
3. Click **Create Report** to generate the report.

To regenerate a report via command line, use `cov-generate-misra-report` with the following settings:

```
cov-generate-misra-report <configuration-file> [--help] --on-new-cert [ trust
| distrust ] --output <output-file> --password <spec> [--project <project-
name>]
```

Parameter	Meaning
`<configuration-file>`	`config.yaml` configuration file.
`--help`	Displays this help message and exits.
`--on-new-cert <value>`	<<value>> can be `trust` or `distrust`. When the value is `distrust` (the default), an attempt to connect to the server using SSL might fail if the server provides an untrusted self-signed certificate.
`--output <output-file>`	Name of the PDF output file. The file will be replaced if present.
`--password <spec>`	Coverity Connect password specifier. The password <<`spec`>> argument is required, and has four forms:

Parameter	Meaning
	console The password is read from the keyboard without echoing to the console. file:<<`filename`>> The password is read from the first line of the file <filename>. file:- The password is read from standard input. This is for use with pipes and redirection, not for keyboard input. env:<<`variable`>> The password is read from the environment variable <<`variable`>>.

9.2.5. Generating a report from the command line

You can generate a report using a previously saved configuration by using the headless mode. On the command line, use `cov-generate-misra-report` with the following settings.

```
cov-generate-misra-report <configuration file> [--help] --on-new-cert [ trust
| distrust ] --output <output-file> --password <spec> [--project <project-
name>]
```

<configuration file>
: MISRA Report `.yaml` configuration file

--help
: Display this help message and exit.

--on-new-cert <value>
: <value> can be "trust" or "distrust". If "distrust" (the default), an attempt to connect to the server using SSL will fail if the server provides an untrusted self-signed certificate.

--output <output-file>
: Name of the PDF output file. The file will be replaced if present.

--password <spec>
: Coverity Connect password specifier

The password `<spec>` argument is required, and has three forms:

console
: The password is read from the keyboard without echoing to the console.

file:<filename>
: The password is read from the first line of the file <filename>.

env:<variable>
: The password is read from the environment variable <variable>.

9.2.6. Localizing reports

You can now localize MISRA reports for Japanese, Korean, and Chinese. To localise the report, make your selection from the **Report's Locale** dropdown list in the Customization pane.

Important

You must have the same locale configured in Coverity Connect as you set for your report. Otherwise, portions of the report will be presented in the user's locale rather than the desired one. (Use the drop down list from **Admin User > Preferences > Locale** to select the desired locale.)

Part 10. Synopsys Software Integrity Report

Table of Contents

10.1. Introduction 388
10.2. Installation 389
10.3. Setup and configuration 390
    10.3.1. Setting up the Synopsys Software Integrity Report Generator 390
    10.3.2. Configuring the Synopsys Software Integrity Report 391
    10.3.3. Updating the report configuration file 391
    10.3.4. Using system environment variables 394
    10.3.5. Generating a report through the GUI 394
    10.3.6. Generating a report via command line 394
10.4. PDF report outline 396
10.5. Detailed results 397
    10.5.1. Coverity Results 397

Chapter 10.1. Introduction

The Synopsys Software Integrity Report summarizes integrity issues existing in a software development project. The report takes input from the Coverity Analysis testing tools. A report generator application pulls issue data from each contributing tool, aggregates the information, and generates data files and a formatted PDF report.

Chapter 10.2. Installation

The Synopsys Software Integrity Report software is available in the `cov-reports` installer. You can obtain the installer from the *Downloads* page in Coverity Connect. The Coverity Reports installer installs Synopsys Software Integrity Report.

Chapter 10.3. Setup and configuration

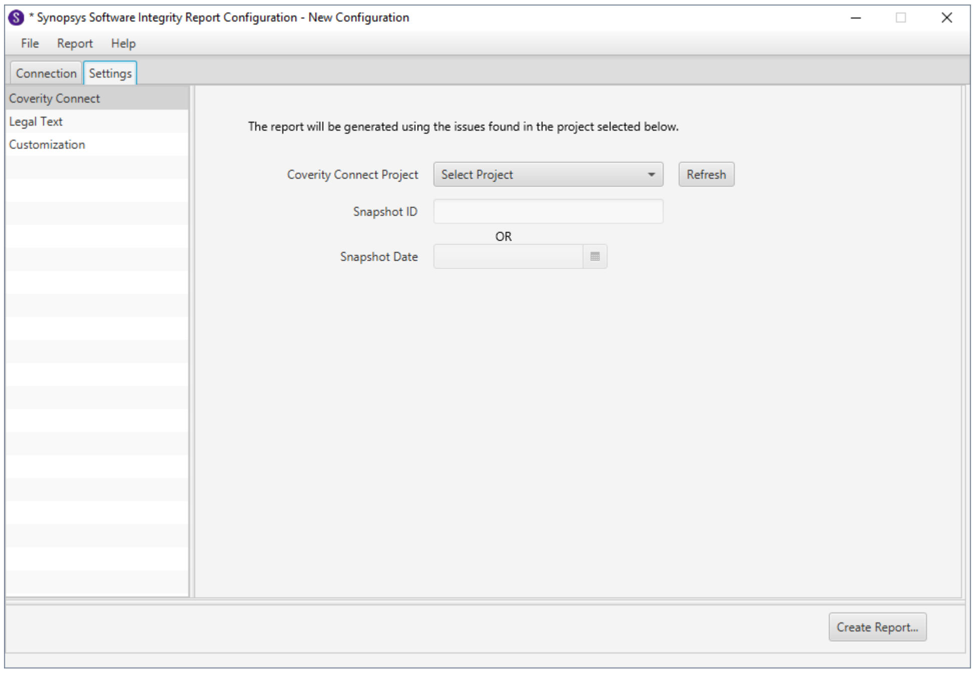
Table of Contents

10.3.1. Setting up the Synopsys Software Integrity Report Generator 390
10.3.2. Configuring the Synopsys Software Integrity Report 391
10.3.3. Updating the report configuration file 391
10.3.4. Using system environment variables 394
10.3.5. Generating a report through the GUI 394
10.3.6. Generating a report via command line 394

10.3.1. Setting up the Synopsys Software Integrity Report Generator

To set up the Synopsys Software Integrity Report Generator, perform the following steps:

1. Navigate to the Reports installation directory.
2. In the `/bin` subdirectory, launch `syn-integrity-report`.
3. On the *Report Settings* tab, in the *Tools* subtab, select the contributing tools to use for the report.



The report will be generated using the issues found in the project selected below.

4. Select *Cover Page*, and enter information on terminology to use in the report.
5. Select *Legal Text*, and enter or upload any legal disclaimers to be shown on the summary page of the report.
6. Select the tab for each enabled analysis tool.

7. In the *Connection* page, enter the *URL* and *Username* for the tool's server (Coverity Connect). Click **Check Connection** to test.

8. In the **Settings** page, click **Refresh** to get a list of projects from the server. Select the desired project or product from the drop-down menu.

9. Repeat the settings for each enabled tool.

10.3.2. Configuring the Synopsys Software Integrity Report

To configure a new SSIR report:

1. Select Settings → Coverity Connect.

2. Select **Coverity Connect Project**. The projects available through the connection to Coverity Connect are displayed in the drop-down list.

   Note

   Before creating a report, make sure that the Synopsys Software Integrity Report project has been analyzed with Coverity Analysis version 2020.12 prior to commit. SSIR analysis in previous versions will not work with this report.

3. In the *Customization* pane, enter information to customize the report. The names and terms are used throughout the report, and the company name and logo are featured on the cover page. The company logo is optional.

   Note

   The *Project* mentioned here refers to the corporate project name, and should not be confused with the Coverity Connect project.

4. Click File → Save to save the `.yaml` configuration file for future use.

   Note

   This document can be used to regenerate a report with the same settings whenever the analysis data is updated.

5. Click **Create Report** to generate the report.

10.3.3. Updating the report configuration file

You can specify and update configuration information in a `.yaml` configuration file. A sample `config.yaml` file is shipped with the report generator and installed in the `config` directory.

Here is an example `.yaml` file:

```
################## Sections that apply to all reports #############
version:
    schema-version: 5
```

```
connection:
    url: https://coverity.example.com:8443/
    username: admin
    ssl-ca-certs:

project:
title-page:
    company-name:
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:

locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:

################## Synopsys Software Integrity Report #############
ssir-report:
    analysis-date: 01/15/2019
    legal-text:
```

Note

The Synopsys Software Integrity Report requires the additional report configuration settings listed in the example above. This ensures that the Synopsys Software Integrity Report results are properly included in the output.

The following table describes each key in the file:

Key	Description
`analysis-date`	Lists date of report generation. The date should be written in MM/DD/YYYY format. field.
`company-name`	Name of your company. This is a mandatory field.
`components`	An optional comma-separated list of Coverity Connect component names, which in map names. If the components are listed here, the report will include data only for t components; for example: `Default.lib` or `Default.src`.
`connection`	The URL of the Coverity Connect instance.
`issue-cutoff-count`	Some reports display information about individual issues. These reports bound the displayed in order to control the size of the report. This bound is called the issue cu for CVSS, Security, PCIDSS, MobileOwasp, and Owasp2017 reports. Default value 10000 for Security report.

Key	Description
`issue-kind`	An optional comma-separated list of Coverity Connect issue kinds. If issue kinds are report will include only issues of the listed kinds. The possible values for `issue-kind` are as follows: • `Quality` • `Security` The following line is an example of using this option: `issue-kind: Quality`
`legal-text`	This setting displays legal text in the report. For example, it will display something li first line of multiline legal text and this is the second line." This field is optional.
`locale`	Locale of the report. Valid values are the following: `en_US`, `ja_JP`, `ko_KR`, and `zh_` `en_US`.
`logo`	Optional path to a logo file for your company. Valid image types are bmp, gif, jpg, ar maximum allowed image size is 210 pixels wide by 70 pixels high. Note that backsla path must be doubled.
`on-cert-trust`	Allows users to trust self-signed certificates sent by Coverity Connect. There are tw `trust` or `distrust`. Use `trust` to accept, use, and store a self-signed certificate `distrust` to reject connections from servers using self-signed certificates. The def
`organizational-unit-name`	Name of your division, group, team or other organizational unit. This is a mandatory
`organizational-unit-term`	Organizational unit term (e.g., division, group, team). This is a mandatory field.
`prepared-for`	Name of the entity for which the report was prepared. This is a mandatory field.
`prepared-by`	Name of the entity that prepared the report. This is a mandatory field.
`project`	Name of the Coverity Connect project.
`project-contact-email`	Project contact email address. It is used for the following reports: CIR, CVSS, PCID and OWASP2017. This is a mandatory field.
`project-name`	Name of the software development project. May be distinct from the Coverity Conne This is an optional field.
`project-version`	Lists the project version number. This is a mandatory field.
`snapshot-date`	Retrieves the most recent snapshot of each stream in the project whose date is less the given date. Format is "DD/MM/YYYY".
`snapshot-id`	Retrieves the defects of a specific snapshot id, instead of using the latest snapshot associated with the project.
`ssir-report`	Declares the Synopsys Software Integrity Report configuration settings.
`title-page`	Describes the fields in the title page of the report.
`username`	Coverity Connect username. Password or other authentication key
`version`	The version of the `.yaml` file's schema.

Once you have entered your settings, click File → Save As to save the configuration information for later use. The setting values (except passwords) are saved in a configuration file.

For more information about schema configurations, see Chapter 6.1, *Configuring Report Generators*

10.3.4. Using system environment variables

To use a system environment variable for your report generator, write the path and filename where you'd like the output file to be created and stored.

On Windows, you'd set the environment variable like this:

```
set WRITE_REPORT_XML=<filename>
```

On Linux, you'd set the environment variable like this:

```
export WRITE_REPORT_XML=<filename>
```

For the Synopsys Software Integrity Report Generator, you can use the following environment variable:

- `WRITE_REPORT_XML`: This variable writes properties from the report's configuration to the XML output file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.

10.3.5. Generating a report through the GUI

Click **Create Report** to start the data collection process. The report generator will access each analysis tool server, collect the data, and then compose the detailed results and PDF files.

1. In the Configuration tool, click **Create Report**.
2. In each **Password Entry** dialog, enter the password for the respective server.
3. In the *Save Detailed Data* dialog, enter the name and location of the `.zip` file.
4. In the *Save Report* dialog, enter the name and location of the `.pdf` file.
5. In the *Report Generation Completed* dialog, optionally open the PDF file.

10.3.6. Generating a report via command line

The `syn-generate-integrity-report` command line tool is used to generate a report based on an existing configuration file (`.yaml`) generated by `syn-integrity-report`. A password file must also be created in the following format using a text editor:

```
Coverity=Coverity_password
```

Note

Be sure to make this file readable only by the owner.

Execute the application (in the /bin subdirectory) from the command line as follows:

```
syn-generate-integrity-report <configuration-file> [--help] --output <output> --
password <console> | file:<filename> | file:- | env:variable>
        [--project <project-name>] --on-new-cert [trust | distrust]
```

Note

If the server is using SSL, the `--on-new-cert` option can be used to indicate whether to trust self-signed certificates, presented by the server, that the application has not seen before. Default is `distrust`.

The command uses the name specified by `--output` to generate .pdf and .zip files. The .zip file contains data files used to populate the PDF report.

Chapter 10.4. PDF report outline

The PDF report explains the data provided by the contributing tools in graphs and tables. Text is provided to clearly explain each category of data and the methodology of the report. The report is divided into the following sections:

Product Results
: Shows a high level summary of issues found by each contributing tool.

Coverity Summary
: Shows the results in terms of CWE vulnerabilities, OWASP Top 10 issues, and CWE/SANS Top 40 issues. A Methodology section explains the Coverity tools and issue categories.

Chapter 10.5. Detailed results

Table of Contents

10.5.1. Coverity Results 397

The report generator outputs a zip file containing detailed results for each contributing tool. The data is provided in a machine-readable form, for use by the developers tasked with fixing the issues.

10.5.1. Coverity Results

The Coverity results consist of a single table in CSV format (`coverity-issues.csv`). The table has one row per issue detected by Coverity analysis.

Table 10.5.1. Coverity Results

Column Title	Content	Notes
CID	The CID number of the issue.	This is the ID number of the issue, managed by Coverity Connect.
CWE	The CWE number of the issue, if any.	Not all issues have CWE numbers.
OWASP Top 10 rank	The rank of the issue in the OWASP Top 10, if any.	The Top 10 of 2017 is used.
CWE/SANS Top 40 rank	The rank of the issue in the CWE/SANS Top 40, if any.	The Top 40 of 2019 is used. The Top 40 weaknesses include the weaknesses in the Top 25 list, and 15 additional weaknesses that are considered important under various criteria.
file name	Name of the file where the main event occurs.	
line number	Line number of the main event in the above file.	
type	A short description of a checker.	E.g., “Unsigned compared against 0.”
category	Coverity category of the issue.	E.g., “Control flow issues.”
description	The English text describing the main event.	E.g., “This less-than-zero comparison of an unsigned value is never true. "i < 0U"”

Part 11. Coverity CVSS Report

Table of Contents

11.1. Introduction 399
  11.1.1. Scorecard elements 399
  11.1.2. Triage attributes 400
  11.1.3. CVSS Report profile 401
11.2. Installing the Coverity CVSS Report Report Generator 403
  11.2.1. Working with the Coverity CVSS Report Report Generator 403
  11.2.2. Downloading the Coverity Reports Installer 404
  11.2.3. Configuring a CVSS Report 405
  11.2.4. Generating a CVSS Report 407
  11.2.5. Using system environment variables 409
  11.2.6. CVSS Report Input Files 409

Chapter 11.1. Introduction

Table of Contents

11.1.1. Scorecard elements 399
11.1.2. Triage attributes 400
11.1.3. CVSS Report profile 401

The Coverity Common Vulnerability Scoring System (CVSS) Report details the application security activities carried out to assess software vulnerabilities. Based upon the CVSS framework, it calculates CVSS scores and provides a summary of findings. It uses CWE data and input from the master file or user-defined profile. The codebase is then evaluated against published policy requirements and its results are described in the report, along with required remediation actions. The CVSS Report also analyzes the issues returned by Coverity and calculates a **CVSS Score**, based on the triage attributes. Each policy element (or attribute) must pass for the policy to pass.

You can now localize a CVSS report using your `.yaml` file.

To use a `.yaml` file, specify one of the following for the locale field:

- `en_US` for English
- `ja_JP` for Japanese
- `ko_KR` for Korean
- `zh_CN` for Chinese

Important

You must have the same locale configured in Coverity Connect as you set for your report. Otherwise, portions of the report will be presented in the user's locale rather than the desired one.

11.1.1. Scorecard elements

The scorecard shows the *Target* for each category and the actual result, and indicates if the element passes or fails. For detailed information on how each scorecard element is calculated, see the *Methodology* section of a generated report.

CVSS Critical Count
: This is a list of security vulnerabilities that have scored as *Critical* (between 9.0-10.0 on the CVSS score calculator). The policy prohibits these weaknesses, so the target is zero.

CVSS High Count
: This is a list of security vulnerabilities that have scored as *High* (between 7.0-8.9 on the CVSS score calculator). The policy prohibits these weaknesses, so the target is zero.

CWE/SANS Top 25
: This is a list of software weaknesses that are thought to be widespread and critical. The policy prohibits these weaknesses, so the target is zero.

OWASP Top 10
: This is a list of prioritized security weaknesses relating to web application security. The Top 10 from the year 2017 will be used. The policy prohibits these weaknesses, so the target is zero.

11.1.2. Triage attributes

11.1.2.1. CVSS_Audited 400
11.1.2.2. CVSS_Score 400
11.1.2.3. CVSS_Severity 400
11.1.2.4. CVSS_Vector 400

Coverity administrators or users with proper permissions must create and configure the custom triage attributes or CVSS fields. Once the CVSS attributes have been configured, the `cov-generate-cvss-report --scores` command syntax updates the attributes and changes the values with the CWE-CVSS metric mappings.

Note

The following four attributes must be created prior to running the CVSS Report Generator.

11.1.2.1. `CVSS_Audited`

`CVSS_Audited` is a pick list type attribute. This field overrides generated vectors from the CVSS metric values in the master file and/or profile.

There are also two possible values: `Yes` and `No`. The default value should be set to `No`. When `CVSS_Vector` is reviewed and `CVSS_Audited` is set to `Yes`, the `CVSS_Score` is then calculated according to the `CVSS_Vector` value that is stored in Coverity Connect.

Note

If `CVSS_Audited` is set to `Yes`, `CVSS_Vector` pulls data from Coverity Connect instead of calculating it. If it is set to `No`, `CVSS_Audited` calculates `CVSS_Vector` based on the `config/Master_CWE_CVSS_Base_Score_Mapping-v1.json` file or `<SECURITY-PROFILE>.JSON` file.

11.1.2.2. `CVSS_Score`

`CVSS_Score` is a text type attribute. This field is computed from the `CVSS_Vector`. There is no default value. At each run, the CVSS score generator sets the field.

11.1.2.3. `CVSS_Severity`

`CVSS_Severity` is a pick list type attribute. This score is computed from the `CVSS_Score`.

`CVSS_Severity` has five possible values: `Critical`, `High`, `Medium`, `Low`, and `None`. There is no default value. At each run, the CVSS score generator sets the field.

11.1.2.4. `CVSS_Vector`

`CVSS_Vector` is a text type attribute. This value drives all of the other computations and is based on static analysis data. There is no default value. At each run, the field is set by the report generator but can be modified if the CVSS score needs to be adjusted.

11.1.3. CVSS Report profile

A CVSS Report profile contains application-specific CWE-CVSS vector mappings. It is based on application type and is created by the security expert. Examples include web applications, mobile applications, and embedded applications.

The CVSS Report profile should be structured like the following:

```
{
    "version"       : 1,
    "type"          : "CVSS profile",
    "AV"            : "N",
    "AC"            : "L",
    "PR"            : "L",
    "UI"            : "N",
    "cweMap"        : [{
            "cwe" : 4,
            "cvssMetrics" : {
                "S" : "U",
                "C" : "N",
                "I" : "N",
                "A" : "N"
            }
        },
        {
            "cwe" : 7,
            "cvssMetrics" : {
                "S" : "U",
                "C" : "N",
                "I" : "H",
                "A" : "H"
            }
        }
    ]
}
```

In creating your CVSS Report profile, please note the following:

- For initial validation, the `version` field should be set to the CVSS Report Generator product version number. The `type` field should be set to `CVSS profile`. If a user's profile contains a mismatch for these values, an error will occur causing CVSS Report Generator to quit.
- The values AC, AV, PR, and UI are independent of CWE. They differ based on application type.
- The `CWEMap` contains CWE-CVSS metric mappings.
- If a given defect has a CWE value of C and you provide F, a CWE-to-CVSS-vector mapping file, then the CVSS vector will be taken from:

  1. The value for C in F, or, if none,
  2. The value for the ancestor with the highest CVSS score of C defined in F, or, if none,

3. The value for C in the built-in configuration (master .json file) or, if none,
4. The value for the ancestor with the highest CVSS score of C in the built-in configuration, or, if none
5. A vector whose CVSS score is zero (i.e. CWE value of 0).

Note

These profiles do not need to cover OWASP Top 10/ CWE CANS Top 25 CWEs, as they are already covered in the Coverity CVSS Report.

Chapter 11.2. Installing the Coverity CVSS Report Report Generator

Table of Contents

11.2.1. Working with the Coverity CVSS Report Report Generator 403
11.2.2. Downloading the Coverity Reports Installer 404
11.2.3. Configuring a CVSS Report 405
11.2.4. Generating a CVSS Report 407
11.2.5. Using system environment variables 409
11.2.6. CVSS Report Input Files 409

The Coverity CVSS Report software is available with the `cov-reports` installer. Therefore, it is recommended that you download the `cov-reports` installer first. You can obtain the installer from the Downloads page in Coverity Connect. For more information, see Downloading the Coverity Reports installer.

11.2.1. Working with the Coverity CVSS Report Report Generator

This section describes the different user workflows for getting started with Coverity CVSS Report Generator.

Security Team:

1. From the Coverity Connect Downloads page, download the Coverity Reports installer. The Coverity Reports installer contains the `cov-generate-cvss-report` tool.

2. Create the following CVSS attributes: `CVSS_Audited`, `CVSS_Score`, `CVSS_Severity`, and `CVSS_Vector`.

3. Create the `<SECURITY-PROFILE>.JSON` file for your users. This file overrides the `config/Master_CWE_CVSS_BASE_SCORE_PROFILE_V1.json` file.

   Note

   The `<SECURITY-PROFILE>.JSON` file path must be specified in the `config/config.yaml` file.

4. Use the `scores` option to update the CVSS attributes. For example:

```
/bin/cov-generate-cvss-report --password <spec> --project <project-name> --profile
 <security-profile-file> --scores config/config.yaml
```

5. Verify that the `CVSS_Vector` value for each defect is correct.

6. Once you have verified that the `CVSS_Vector` value is correct, set the `CVSS_Audited` field to `Yes`.

7. [Optional:] If a change needs to be made, update `<SECURITY-PROFILE>.JSON` and run the `scores` option again.

Development Team:

1. Download the Coverity Reports installer from the Coverity Connect Downloads page. The Coverity Reports installer includes the `cov-generate-cvss-report tool`.
2. Create the following CVSS attributes: `CVSS_Audited`, `CVSS_Score`, `CVSS_Severity`, and `CVSS_Vector`.
3. In Coverity Connect, create a project with snapshots.
4. Use the `<SECURITY-PROFILE>.JSON` file for your CWE-CVSS vector mappings. The `<SECURITY-PROFILE>.JSON` file is created by the security team.
5. [Optional:] If the `<SECURITY-PROFILE>.JSON` file has not been created for you, then use the default `config/Master_CWE_CVSS_Base_Score_Mapping-v1.json` file.
6. Update the `config/config.yaml` file to reflect the users' settings.
7. Use the `scores` option to update the CVSS attributes. For example:

   ```
   /bin/cov-generate-cvss-report --password <spec> --project <project-name> --profile
    <security-profile-file> --scores config/config.yaml
   ```

8. Once all the updates are made, use the `report` option to generate the CVSS Report. For example:

   ```
   /bin/cov-generate-cvss-report --output <output-file> --password <spec> --project
    <project-name> --report config/config.yaml
   ```

11.2.2. Downloading the Coverity Reports Installer

1. In Help → Downloads, select the Coverity Reports installer that is appropriate for your system.
2. Save the installer application to an appropriate folder in your system.
3. You have the option to use the install wizard or to install from the command line. To use the wizard, launch the installer, and follow the instructions in the wizard.

   To use the command line, proceed to the next step.
4. To run the installer from the command line, execute one of the following commands, using either quiet mode (`-q`) or console mode (`-c`):

Operating System	**Command**
Windows	`start /wait "" cov-reports-win64-<version_number>.exe -c`
Windows - quiet mode	`cov-reports-win64-<version_number>.exe -q -dir C:/target-dir`
Linux	`./cov-reports-linux64-<version_number>.sh -c`
Linux - quiet mode	`./cov-reports-linux64-<version_number>.sh -q -dir ~/target-dir`

If you do not specify a `target-dir` for the installation, a default installation directory is used. The directory name is shown at runtime.

You can use the following parameters in the command line:

Parameter	Action
`-c`	Run in console mode. User interaction is performed in the terminal window where the installer (or uninstaller) is invoked.
`-console`	On Windows, use with `-q` to open a console window to display output in quiet mode.
`-dir [directory]`	Set the installation directory in quiet mode.
`-Dname=value`	Set system properties.
`-h`	Display help.
`-manual`	On Windows, in GUI mode only, manually select a Java Runtime Environment.
`-overwrite`	Overwrite all files in quiet mode.
`-q`	Run in unattended (quiet) mode. There is no user interaction, and installation is performed automatically using default values.
`-splash [title]`	Display a progress bar in quiet mode.
`-varfile`	Use a response file.

11.2.3. Configuring a CVSS Report

You can specify configuration information using a `.yaml` file. A sample `config.yaml` file is shipped with the report generator and installed in the `config` directory.

The following shows a sample `.yaml` file:

```
################## Sections that apply to all reports #############
version:
    schema-version: 5

connection:
    url: https://coverity.example.com:8443/
    username: admin
    ssl-ca-certs:

project:
title-page:
    company-name:
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:

locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:
```

Note

The CVSS Report does not require any additional report settings to be configured.

The following table describes each key in the file:

Key	Description
`company-name`	Name of your company. This is a mandatory field.
`components`	An optional comma-separated list of Coverity Connect component names, which inc map names. If the components are listed here, the report will include data only for th components; for example: `Default.lib` or `Default.src`.
`connection`	The URL of the Coverity Connect instance.
`issue-cutoff-count`	Some reports display information about individual issues. These reports bound the r displayed in order to control the size of the report. This bound is called the issue cut for CVSS, Security, PCIDSS, MobileOwasp, and Owasp2017 reports. Default value 10000 for Security report.
`issue-kind`	An optional comma-separated list of Coverity Connect issue kinds. If issue kinds are report will include only issues of the listed kinds. The possible values for `issue-kind` are as follows: • `Quality` • `Security` The following line is an example of using this option: `issue-kind: Quality`
`locale`	Locale of the report. Valid values are the following: `en_US`, `ja_JP`, `ko_KR`, and `zh_` `en_US`.
`logo`	Optional path to a logo file for your company. Valid image types are bmp, gif, jpg, an maximum allowed image size is 210 pixels wide by 70 pixels high. Note that backsla path must be doubled.
`on-cert-trust`	Allows users to trust self-signed certificates sent by Coverity Connect. There are tw `trust` or `distrust`. Use `trust` to accept, use, and store a self-signed certificate `distrust` to reject connections from servers using self-signed certificates. The def

Key	Description
`organizational-unit-name`	Name of your division, group, team or other organizational unit. This is a mandatory
`organizational-unit-term`	Organizational unit term (e.g., division, group, team). This is a mandatory field.
`prepared-for`	Name of the entity for which the report was prepared. This is a mandatory field.
`prepared-by`	Name of the entity that prepared the report. This is a mandatory field.
`project`	Name of the Coverity Connect project.
`project-contact-email`	Project contact email address. It is used for the following reports: CIR, CVSS, PCID and OWASP2017. This is a mandatory field.
`project-name`	Name of the software development project. May be distinct from the Coverity Conne This is an optional field.
`project-version`	Lists the project version number. This is a mandatory field.
`snapshot-date`	Retrieves the most recent snapshot of each stream in the project whose date is less the given date. Format is "DD/MM/YYYY".
`snapshot-id`	Retrieves the defects of a specific snapshot id, instead of using the latest snapshot associated with the project.
`title-page`	Describes the fields in the title page of the report.
`username`	Coverity Connect username. Password or other authentication key
`version`	The version of the `.yaml` file's schema.

For more information about schema configurations, see Chapter 6.1, *Configuring Report Generators*

11.2.4. Generating a CVSS Report

The CVSS Report can only be generated via command line, as it is unavailable in GUI mode.

Note

Before you begin, make sure the following CVSS attributes are configured in Coverity Connect: `CVSS_Audited`, `CVSS_Score`, `CVSS_Severity`, and `CVSS_Vector`.

To generate a CVSS Report, use the `cov-generate-cvss-report` command along with the following options:

```
cov-generate-cvss-report <configuration-file> [--company-logo <path-to-
company-logo>] [--help] [--on-new-cert <value>] [--output <output-path-to-
pdf>] --password <spec> [--profile <profile.json>] [--project <project-name>]
[--report] [--scores]
```

Note

When generating a report, it is recommended that you use the `--profile <security-profile-file>` and `--scores` options in the same command as the `--output <output-file>` and `--report` options.

You can also generate a report without using the `--profile <security-profile-file>` and `--scores` options, but you should ensure that the `CVSS_*` attributes are updated before generating the report.

Assign the CVSS score to your defects and generate a report.

Optional arguments:

`--company-logo`
: Path to the company logo file.

`--help`
: Shows the help message and exits.

`--on-new-cert <value>`
: `<value>` can be `trust` or `distrust`. If the server provides an untrusted self-signed certificate and the value is set to `distrust` (which is the default), an attempt to connect to the server using SSL may fail.

`--output <output-path-to-pdf>`
: The name of the PDF output file. The `--output <output-file>` option replaces any existing `<output-file>` files with the same name. Must be used in conjunction with the `--report` option.

`--profile <PROFILE>.JSON`
: A .json file with CWE-CVSS vector mappings.

`--project <project-name>`
: Name of the Coverity project to assign CVSS defect metrics. This can be set via command line or entered directly in the YAML configuration file.

  Note that if a user has already set the project in the `config.yaml` file and also tries to set the `--project` name through the command line, the command line will supercede what is written in the configuration file.

`--report`
: Must be specified with the `--output <output-file>` option. Can also be used with the `--profile <security-profile-file>` and `--scores` options. When specified without the `--scores` option, it generates a CVSS Report and does not update the `CVSS_*` attributes.

  When specified with the `--scores` option, the `--report` option generates the CVSS Report and also updates the `CVSS_*` attributes in Coverity Connect.

`--scores`
: When specified with the `--report` option, the `--scores` option updates the `CVSS_*` attributes in Coverity Connect when generating a report.

Required arguments:

`<configuration-file>`
: A `.yaml` file containing server configuration, Coverity Connect project, and other report-related parameters.

For detailed information about configuration values, please see the "Coverity Report Generators" section of the *Release Notes* and the README to which it refers.

`--password <spec>`
Coverity Connect password specifier

The password `<spec>` argument is required, and has four forms:

1. `console`

   The password is read from the keyboard without echoing to the console.

2. `file:<filename>`

   The password is read from the first line of the file `<filename>`.

3. `file:-`

   The password is read from standard input. This is used with pipes and redirection.

4. `env:<variable>`

   The password is read from the environment variable `<variable>`.

11.2.5. Using system environment variables

To use a system environment variable for your report generator, write the path and filename where you'd like the output file to be created and stored.

On Windows, you'd set the environment variable like this:

```
set WRITE_REPORT_XML=<filename>
```

On Linux, you'd set the environment variable like this:

```
export WRITE_REPORT_XML=<filename>
```

For the CVSS Report Generator, you can use these two environment variables:

- `WRITE_ISSUES_JSON`: This variable writes defect or issue data to the JSON output file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.
- `WRITE_REPORT_XML`: This variable writes properties from the report's configuration to the XML output file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.

11.2.6. CVSS Report Input Files

The CVSS Report input files use input data from the following files to update the CVSS metrics and generate the PDF report:

- `config/config.yaml`

This template file should be created or updated by the user via command line. Changes to the file name and content are allowed.

- `<SECURITY-PROFILE>.JSON`

  A .json file that is created by the security team (and not the user). Its structure is similar to the `Master_CWE_CVSS_Base_Score_Mapping-v1.json` file. Comments are allowed.

- `config/Master_CWE_CVSS_Base_Score_Mapping-v1.json`

  The master .json file which contains default mappings between CWE and CVSS metrics. This file must remain in the `config/` folder and should not be removed.

Part 12. Coverity CERT Report Guide

Table of Contents

12.1. Overview 412
12.2. Installing the CERT Report Generator 413
    12.2.1. Connecting to Coverity Connect 413
    12.2.2. Working with the CERT Report 414
    12.2.3. Configuring the Coverity CERT Report 415
    12.2.4. Using system environment variables 417
    12.2.5. Generating a previously configured report 418
    12.2.6. Generating the a report from the command line 418
    12.2.7. Interpreting the CERT Report 418

Chapter 12.1. Overview

The Coverity CERT Report aggregates the results of Coverity Static Analysis performed on a particular project. A project is a collection of one or more streams containing separately analyzed snapshots. The latest snapshot in each stream is used when reporting results for a project. A CERT Report provides information about CERT vulnerabilities detected by the Coverity CERT checkers described in the *Coverity 2020.12 Checker Reference*.

The CERT Report provides summary information for outstanding violations by component and rule, and for dismissed violations. Additionally, for each rule defined in the standard, the CERT Report describes the rule, its priority and level, and specifies whether it is supported and enabled, and the number of times that rule has been violated.

The *Methodology* section of the report explains in detail how rules are defined, and how their priority and level are determined. It also explains how CERT terminology for violations maps to Coverity terminology for defects.

The basic workflow for working with CERT reports is as follows:

1. Install the CERT Report Generator.
2. Connect to Coverity.
3. Configure the CERT Report.
4. Generate the report.
5. Interpret the report.

The sections that follow explain each of these steps.

Important

To support report generation, you must use the `--coding-standard-config` option to the `cov-analyze` command. This option provides the path to a configuration file for a coding standard to run as part of the analysis. For CERT, it might look like this:

```
{   version : "2.0",
    standard : "cert-c",
    title: "your_title_here",
    deviations : []
}
```

Chapter 12.2. Installing the CERT Report Generator

Table of Contents

12.2.1. Connecting to Coverity Connect 413
12.2.2. Working with the CERT Report 414
12.2.3. Configuring the Coverity CERT Report 415
12.2.4. Using system environment variables 417
12.2.5. Generating a previously configured report 418
12.2.6. Generating the a report from the command line 418
12.2.7. Interpreting the CERT Report 418

The Coverity CERT Report Generator is installed by the Coverity Reports installer. You can obtain the Coverity Reports installer from the *Downloads* page in Coverity Connect.

Follow these steps to install Coverity Reports:

1. In Help → Downloads, select the Coverity Reports installer that is appropriate for your system.

2. Save the installer application to an appropriate folder in your system.

3. Launch the installer using the GUI or the command line.

   - To run the installer using the GUI, follow the instructions in the wizard.

   - To run the installer from the command line, execute one of the following commands depending on your operating system and your preference for quiet mode (`-q`) or console mode (`-c`). In quiet mode, there is no user interaction: installation is performed automatically using default values.

Windows	`cov-reports-win64-<version_number>.exe -q -dir C:/target-dir`
Windows	`start /wait "" cov-reports-win64-<version_number>.exe -c`
Linux	`./cov-reports-linux64-<version_number>.sh -q -dir ~/target-dir`
Linux	`./cov-reports-linux64-<version_number>.sh -c`

   If you do not specify a target directory for the installation, a default installation directory is used. The directory name is shown at runtime.

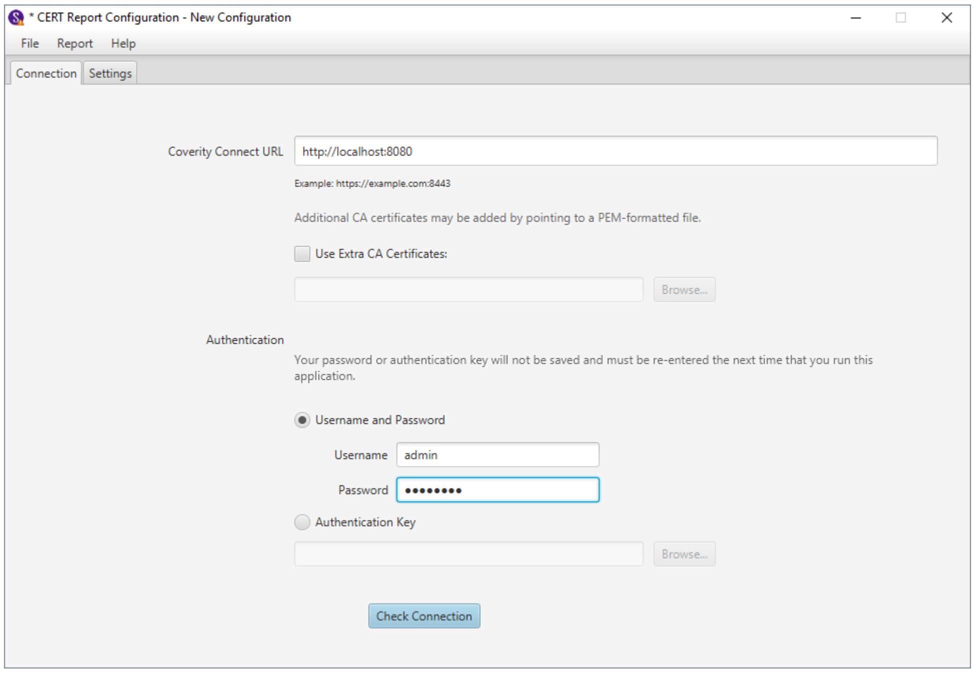
12.2.1. Connecting to Coverity Connect

Before you can configure and generate a report, you must set up a connection to your instance of Coverity Connect, which stores the information that is included in the report. Follow these steps to create your connection:

1. Locate the application executable `cov-cert-report` in `<installation-directory>/bin`

2. Double-click on the application icon to launch it. It starts in *New Configuration* mode. (Alternatively, double-click on a previously-created configuration file to launch the application with that configuration.)

3. On the *Connection* tab, enter the *Host Name* and *Port Number*. If your Coverity Connect administrator has enabled SSL, enter the HTTPS port number associated with your Coverity Connect host. The default HTTPS port is 8443.



Coverity Connect URL: http://localhost:8080

Example: https://example.com:8443

Additional CA certificates may be added by pointing to a PEM-formatted file.

☐ Use Extra CA Certificates:

Authentication

Your password or authentication key will not be saved and must be re-entered the next time that you run this application.

● Username and Password

Username: admin

Password: ••••••••

○ Authentication Key

4. Check the Coverity Connect URL to make sure it is correct.

5. If SSL is used with the connection, select *Secured using SSL*.

   Note

   If an additional CA certificate is needed, select *Use Extra CA Certificate*, and click **Browse** to identify the file.

6. Enter the *Username* and *Password* for Coverity Connect. The password is *not* stored in the configuration file.

7. Click **Check Connection** to test the connection to Coverity Connect.

If you want to regenerate a report using the same input data, you do not have to reconnect to Coverity.

12.2.2. Working with the CERT Report

The user workflow for working with the CERT Report is as follows:

1. In the *Settings* pane, select *Coverity Connect*.

2. Use the **Coverity Connect Project** drop-down list to select a project in Coverity Connect. This requires a valid connection, as established in the previous step.

3. In the *Customization* pane, enter information to customize the report. The names and terms you specify are used throughout the report, and the company name and logo (if specified) are featured on the cover page.

   If you elect to use a logo, click **Choose File** button to specify the name of an image file containing the logo. (Coverity can handle most standard image files. If it cannot handle the image you provide, it returns an error specifying the formats it can handle.)

4. Click File → Save to save the `.yaml` configuration file for future use.

5. Click **Create Report** to generate the report.

12.2.3. Configuring the Coverity CERT Report

You can specify configuration information using a `.yaml` file. A sample `config.yaml` file is shipped with the report generator and installed in the `config` directory.

Here is an example `.yaml` configuration file:

```
################## Sections that apply to all reports #############
version:
    schema-version: 5

connection:
    url: https://coverity.example.com:8443/
    username: admin
    ssl-ca-certs:

project:
title-page:
    company-name:
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:

locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:

################## CERT report #############
cert-report:
    target-level: F
```

Note

The CERT Report requires the additional CERT Report settings listed in the example above. This ensures that the CERT Report results are properly included in the output.

The following table describes each key in the file:

Key	Description
`cert-report`	Declares the CERT Report settings.
`company-name`	Name of your company. This is a mandatory field.
`components`	An optional comma-separated list of Coverity Connect component names, which inc map names. If the components are listed here, the report will include data only for th components; for example: `Default.lib` or `Default.src`.
`connection`	The URL of the Coverity Connect instance.
`issue-cutoff-count`	Some reports display information about individual issues. These reports bound the r displayed in order to control the size of the report. This bound is called the issue cut for CVSS, Security, PCIDSS, MobileOwasp, and Owasp2017 reports. Default value 10000 for Security report.
`issue-kind`	An optional comma-separated list of Coverity Connect issue kinds. If issue kinds are report will include only issues of the listed kinds. The possible values for `issue-kind` are as follows: • `Quality` • `Security` The following line is an example of using this option: `issue-kind: Quality`
`locale`	Locale of the report. Valid values are the following: `en_US`, `ja_JP`, `ko_KR`, and `zh_` `en_US`.
`logo`	Optional path to a logo file for your company. Valid image types are bmp, gif, jpg, ar maximum allowed image size is 210 pixels wide by 70 pixels high. Note that backsla path must be doubled.
`on-cert-trust`	Allows users to trust self-signed certificates sent by Coverity Connect. There are tw `trust` or `distrust`. Use `trust` to accept, use, and store a self-signed certificate `distrust` to reject connections from servers using self-signed certificates. The def
`organizational-unit-name`	Name of your division, group, team or other organizational unit. This is a mandatory
`organizational-unit-term`	Organizational unit term (e.g., division, group, team). This is a mandatory field.
`prepared-for`	Name of the entity for which the report was prepared. This is a mandatory field.
`prepared-by`	Name of the entity that prepared the report. This is a mandatory field.
`project`	Name of the Coverity Connect project.

Key	Description
`project-contact-email`	Project contact email address. It is used for the following reports: CIR, CVSS, PCID and OWASP2017. This is a mandatory field.
`project-name`	Name of the software development project. May be distinct from the Coverity Conne This is an optional field.
`project-version`	Lists the project version number. This is a mandatory field.
`snapshot-date`	Retrieves the most recent snapshot of each stream in the project whose date is less the given date. Format is "DD/MM/YYYY".
`snapshot-id`	Retrieves the defects of a specific snapshot id, instead of using the latest snapshot associated with the project.
`target-level`	This setting is used to determine the CERT configuration for target levels. The defa following values are valid: • F: Fully Compliant • L2: L2 Compliant • `L1`: L1 Compliant
`title-page`	Describes the fields in the title page of the report.
`username`	Coverity Connect username. Password or other authentication key
`version`	The version of the `.yaml` file's schema.

For more information about schema configurations, see Chapter 6.1, *Configuring Report Generators*

12.2.4. Using system environment variables

To use a system environment variable for your report generator, write the path and filename where you'd like the output file to be created and stored.

On Windows, you'd set the environment variable like this:

```
set WRITE_REPORT_XML=<filename>
```

On Linux, you'd set the environment variable like this:

```
export WRITE_REPORT_XML=<filename>
```

For the CERT Report Generator, you can use these two environment variables:

- `WRITE_ISSUES_JSON`: This variable writes defect or issue data to the JSON output file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.
- `WRITE_REPORT_XML`: This variable writes properties from the report's configuration to the XML output file. If a file with the same filename exists, it will be overwritten. A warning is issued if the file cannot be opened.

12.2.5. Generating a previously configured report

You can save the configuration for a report as a `.yaml` file, and then use this document to regenerate the report, with the same settings, when the analysis data is updated. You can regenerate the report using the GUI or the command line.

To regenerate a report through the GUI:

1. Launch the `cov-cert-report` application.
2. Click File → Open Configuration, and select an existing configuration file.
3. Click **Create Report** to generate the report.

12.2.6. Generating the a report from the command line

You can generate a report based on a previously saved configuration by running `cov-generate-cert-report` with the following settings:

```
cov-generate-cert-report <config-file> [--help] --on-new-cert {trust |
distrust} --output <output-file> --password <spec>
```

`<[config-file]>`	The name of a Coverity CERT Report configuration (`.yaml`) file. This is the configuration values you defined using the GUI.
[--help]	Display this help message and exit.
[--on-new-cert <`value`>]	<`value`> can be `trust` or `distrust`. If `distrust` (the default), an attempt to connect to the server using SSL will fail if the server provides an untrusted self-signed certificate.
[--output <`output_file`>]	Name of the PDF output file. The file will be replaced if present.
[--password <`spec`>]	Coverity Connect password specifier. <`spec`> has one of the following forms: console The password is read from the keyboard without echoing. file:<`filename`> The password is read from the first line of <`filename`>. env:<`variable`> The password is read from the environment variable <`variable`>.

12.2.7. Interpreting the CERT Report

The CERT division of the SEI (Software Engineering Institute) defines coding standards for commonly-used programming languages such as C, C++, Java, and Perl, and the Android platform. These standards provide a set of rules and recommendations to help you develop safe, reliable, and secure systems. It is important to understand how CERT rules are evaluated in order to understand reported violations and summaries.

CERT standards, described in Appendix B of the *Coverity Checker Reference*, divide their compliance tests into a set of rules. Not all rules can be checked using static analysis. Each rule has an assigned *priority*, which is the product of three risk assessment values multiplied together. These values are assigned on a scale of 1 to 3 for likelihood, severity, and remediation cost. This product is used to prioritize rules.

Priorities, in turn, may have ten possible values: from lowest to highest (1, 2, 3, 4, 6, 8, 9, 12, 18, 27). Each rule also has a *level*, which divides priorities into one of three buckets:

- L3 for the lowest priorities of 1, 2 ,3 ,4
- L2 for priorities 6, 8, 9
- L1 for the highest priorities of 12, 18, 27

Software can be assessed as L1, L2, or L3 fully conforming, depending on the set of rules used to validate the software. Compliance is evaluated as follows, from lowest to highest:

- **Not conformant**: has one or more L1 rule violations.
- **L1 conforming**: complies with all L1 rules, but has L2 violations.
- **L2 conforming**: complies with all L1 and L2 rules, but has L3 violations.
- **Fully conformant**: has no violations.

The software team should validate code by proceeding from the highest to the lowest priority level. CERT compliance requires that a software product have no defects or exploitable vulnerabilities.

Following several summaries, the CERT Report provides detailed information about CERT rules and your code's compliance. Each entry provides information like the following:

- A *rule identifier* that specifies the rule being validated; for example, PRE30-C.
- A *description* that explains the rule that is being validated; for example "Do not create a universal character name through concatenation."

  A reason might be provided if the rule is currently disabled in the analysis configuration.
- The *priority* of the rule.
- The *level* of the rule.
- Whether the rule is *supported* by the analysis.
- Whether the rule is *enabled* in the analysis configuration.
- The number of times the rule has been violated. This number is a count of individual occurrences of violations. By contrast, the violation (or issue) count in Coverity Connect counts "merged," or fully distinct, groups of violations.

Part 13. Coverity OWASP Web Top 10

Table of Contents

13.1. Overview 421
  13.1.1. Working with the OWASP Web Top 10 Report Generator 422
  13.1.2. Interpreting the OWASP Web Top 10 Report 422
13.2. Working with the OWASP Web Top 10 Report Generator 423
  13.2.1. Installing the Report Generator 423
  13.2.2. Configuring an OWASP Web Top 10 Report 423
  13.2.3. Generating an OWASP Top 10 Report 425

Chapter 13.1. Overview

Table of Contents

13.1.1. Working with the OWASP Web Top 10 Report Generator 422
13.1.2. Interpreting the OWASP Web Top 10 Report 422

The OWASP Web Top 10 report generator uses analysis results for a Coverity Connect project to evaluate the analyzed codebase. Based on this evaluation, it creates a OWASP Top Ten report, which details the assessments that were done, provides a summary of findings, and specifies the remediations needed. Information from this report is of special interest to application security assurance teams and their clients.

The OWASP (Open Web Application Security Project) Foundation is an international organization whose mission is to advance the cause of secure software. As part of its activities, OWASP publishes a report of the most critical web application security flaws in rank order based on the input of a worldwide group of security experts. The most recent version of this list and accompanying report is the OWASP Top 10 List for 2017; The OWSAP Top 10 List is referenced by many standards including MITRE, PCI DSS, DISA, and the FTC. For more information, see https://owasp.org.

The OWASP top ten include the following categories:

- Injection (A1:2017)
- Broken Authentication and Session Management (A2:2017)
- Sensitive Data Exposure (A3:2017)
- XML External Entities (A4:2017)
- Broken Access Control (A5:2017)
- Security Misconfiguration (A6:2017)
- Cross-site Scripting (A7:2017)
- Insecure Deserialization (A8:2017)
- Using Components with Known Vulnerabilities (A9:2017)
- Insufficient Logging and Monitoring (A10:2017)

The report provides additional information about each of these categories.

You can localize reports by setting the `locale` filed of your report generation configuration file.

This chapter describes the workflow needed for generating an OWASP Web Top 10 report and explains how you interpret report findings.

13.1.1. Working with the OWASP Web Top 10 Report Generator

The user workflow for working with the OWASP Web Top 10 Report Generator is as follows:

1. Install the Report Generator using the Coverity Reports Installer. See Installing the Security Report Generator for more information.
2. Configure the report. This allows you to specify information that is displayed on the report's cover page and to specify notification information.

   See Configuring a OWASP Web Top 10 report for more information.
3. Generate the report.

13.1.2. Interpreting the OWASP Web Top 10 Report

The report is divided into three basic sections that describe the issues found in increasing amount of detail:

- **Executive Summary** provides tabular and graphic summary information for the issues found.
- **Analysis Details** shows the number of issues associated with each OWASP Top 10 category. For each defect found, the report describes the issue type, the file name, and line number and the date it was first detected.
- **Methodology** describes basic elements associated with report output.

Each section of the report includes detailed information about the vulnerabilities found and remediation action.

Chapter 13.2. Working with the OWASP Web Top 10 Report Generator

Table of Contents

13.2.1. Installing the Report Generator 423
13.2.2. Configuring an OWASP Web Top 10 Report 423
13.2.3. Generating an OWASP Top 10 Report 425

The following sections explain how you install the OWASP Web Top 10 report generator and how you configure it.

13.2.1. Installing the Report Generator

The OWASP Web Top 10 report generator is installed separately from Coverity Connect. You can obtain the installer for the OWASP Web Top 10 report generator installer from the *Downloads* page in Coverity Connect.

To install on Windows or Linux

1. Log in to Coverity Connect.

   In Help → Downloads, download the cov-report installer.

2. Install `cov-reports`.

3. Navigate to the Coverity report installation directory: for example,

   ```
   C:\Program Files\Coverity\Coverity Reports\bin
   ```

4. Verify that the file `cov-generate-owasp2017-report` exists.

5. Before you run the `cov-generate-owasp2017-report` command to generate the report, you must configure report generation, as explained in Configuring a OWASP Top Ten report .

13.2.2. Configuring an OWASP Web Top 10 Report

Configuration values determine the information displayed on the title page of your report and specify notification information for your project. You can configure a security report by setting specifications in your `config.yaml` configuration file. (A `config.yaml` template file is shipped with the report generator and is installed in the `config` directory.)

Here is an example `.yaml` configuration file:

```
################## Sections that apply to all reports #############
version:
    schema-version: 5

connection:
```

```
    url: https://coverity.example.com:8443/
    username: admin
    ssl-ca-certs:

project:
title-page:
    company-name:
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:

locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:
```

Note

The OWASP Top 10 Report does not require any additional report settings to be configured.

The following table describes each key in the file:

Key	**Description**
`company-name`	Name of your company. This is a mandatory field.
`components`	An optional comma-separated list of Coverity Connect component names, which inc map names. If the components are listed here, the report will include data only for t components; for example: `Default.lib` or `Default.src`.
`connection`	The URL of the Coverity Connect instance.
`issue-cutoff-count`	Some reports display information about individual issues. These reports bound the displayed in order to control the size of the report. This bound is called the issue cu for CVSS, Security, PCIDSS, MobileOwasp, and Owasp2017 reports. Default value 10000 for Security report.
`issue-kind`	An optional comma-separated list of Coverity Connect issue kinds. If issue kinds are report will include only issues of the listed kinds. The possible values for `issue-kind` are as follows: • `Quality` • `Security` The following line is an example of using this option:

Key	Description
	`issue-kind: Quality`
`locale`	Locale of the report. Valid values are the following: `en_US`, `ja_JP`, `ko_KR`, and `zh_` `en_US`.
`logo`	Optional path to a logo file for your company. Valid image types are bmp, gif, jpg, ar maximum allowed image size is 210 pixels wide by 70 pixels high. Note that backsla path must be doubled.
`on-cert-trust`	Allows users to trust self-signed certificates sent by Coverity Connect. There are tw `trust` or `distrust`. Use `trust` to accept, use, and store a self-signed certificate `distrust` to reject connections from servers using self-signed certificates. The def
`organizational-unit-name`	Name of your division, group, team or other organizational unit. This is a mandatory
`organizational-unit-term`	Organizational unit term (e.g., division, group, team). This is a mandatory field.
`prepared-for`	Name of the entity for which the report was prepared. This is a mandatory field.
`prepared-by`	Name of the entity that prepared the report. This is a mandatory field.
`project`	Name of the Coverity Connect project.
`project-contact-email`	Project contact email address. It is used for the following reports: CIR, CVSS, PCID and OWASP2017. This is a mandatory field.
`project-name`	Name of the software development project. May be distinct from the Coverity Conne This is an optional field.
`project-version`	Lists the project version number. This is a mandatory field.
`snapshot-date`	Retrieves the most recent snapshot of each stream in the project whose date is less the given date. Format is "DD/MM/YYYY".
`snapshot-id`	Retrieves the defects of a specific snapshot id, instead of using the latest snapshot associated with the project.
`title-page`	Describes the fields in the title page of the report.
`username`	Coverity Connect username. Password or other authentication key
`version`	The version of the `.yaml` file's schema.

For more information about schema configurations, see Chapter 6.1, *Configuring Report Generators*

13.2.3. Generating an OWASP Top 10 Report

After you have saved your report configuration, you can generate an OWASP Top Report by running the following command:

```
cov-generate-owasp2017-report <configuration-file>
                              [--on-new-cert <value>]
                              [--output <output-file>]
                              [ --password <spec> | --auth-key-file <filename> ]
                              [--project <project-name>]
```

The parameters for this report generator are as follows:

`<configuration-file>`
The path to the `config.yaml` file you created

`--on-new-cert`
(Optional) When connecting to the Coverity Connect server via SSL/TLS, this parameter specifies how to handle a new certificate. The `<value>` parameter can have one of two values:

`distrust`
(The default) If the certificate is self-signed, the connect attempt will fail.

`trust`
The certificate will be accepted, even if it is self-signed.

`--output <output-file>`
(Optional) Specifies a PDF file in which to save the report.

If a file of this name already exists, the new report overwrites the old one.

`--password <spec>`
(Required if `--auth-key-file` is not present.) Specifies how to provide your Coverity Connect password.

The password `<spec>` has a few possible forms, listed here:

`console`
Tells Coverity Connect to prompt for the password from standard input. Coverity Connect does not echo the password characters.

`file:<filename>`
Tells Coverity Connect to read the password from the specified file.

Entering `file:-` tells Coverity Connect to prompt for the file name from standard input.

env:<variable>
Tells Coverity Connect to read the password from the specified environment variable.

`--auth-key-file <filename>`
(Required if `--password` is not present.) Specifies a file that contains your Coverity Connect authentication key.

Authenticating via a key value is an alternative to using a password. See Section 3.2.1.6, “Working with authentication keys”.

`--project <project-name>`
(Optional) Sets the project name.

The value is optional in this location: This option sets the project name here, at the command line. As an alternative, you can set the project name in the YAML configuration file.

For example,

```
cov-generate-owasp2017-report mydir/config.yaml --password console --project demo
```

To get help with command syntax, enter:

```
cov-generate-owasp2017-report --help
```

Part 14. Coverity Mobile OWASP Top 10

Table of Contents

14.1. Overview 429

  14.1.1. User workflow 429

  14.1.2. Interpreting the Mobile OWASP Top 10 Report 429

14.2. Working with the Mobile OWASP Top 10 Report Generator 431

  14.2.1. Installing the Report Generator 431

  14.2.2. Configuring a Mobile OWASP Top 10 Report 431

  14.2.3. Generating a Mobile OWASP Report 433

Chapter 14.1. Overview

Table of Contents

14.1.1. User workflow 429
14.1.2. Interpreting the Mobile OWASP Top 10 Report 429

The Mobile OWASP Top 10 report generator uses analysis results for a Coverity Connect project to evaluate the analyzed codebase. Based on this evaluation, it creates a Mobile OWASP Top Ten report, which details the assessments that were done, provides a summary of findings, and specifies the remediations needed. Information from this report is of special interest to application security assurance teams and their clients.

The Mobile OWASP Security Project provides developers and security teams the resources they need to build and maintain secure mobile applications. The project's goal is to classify mobile security risks and provide developmental controls to reduce their impact or likelihood of exploitation. The Mobile OWASP Top 10 lists the ten top security risks to mobile applications: The most recent version of this list is the Mobile Top 10 for 2017.

You can localize reports by setting the `locale` filed of your report generation configuration file.

This chapter describes the workflow needed for generating a Mobile OWASP Top 10 report and explains how you interpret report findings.

14.1.1. User workflow

The user workflow for working with the Mobile OWASP Top 10 report generator is as follows:

1. Install the Report Generators using the Coverity Reports Installer. See Installing the Security Report Generator for more information.

2. Configure the report. This allows you to specify information that is displayed on the report's cover page and to specify notification information.

   See Configuring a Mobile OWASP Top 10 report for more information.

3. Generate the report.

14.1.2. Interpreting the Mobile OWASP Top 10 Report

The report is divided into four basic sections that describe the issues found in increasing amount of detail:

- **Executive Summary** provides tabular and graphic summary information for the issues found.

- **Analysis Details** shows the number of issues associated with each Mobile OWASP Top 10 category.

- **Detailed Issues Ranked by Mobile OWASP Category** lists all issues, the name of the source file for that issue and the line number where the issue can be found.

- **Methodology** describes basic elements associated with report output.

Each section of the report includes detailed information about the vulnerabilities found and remediation action.

Chapter 14.2. Working with the Mobile OWASP Top 10 Report Generator

Table of Contents

14.2.1. Installing the Report Generator 431
14.2.2. Configuring a Mobile OWASP Top 10 Report 431
14.2.3. Generating a Mobile OWASP Report 433

The following sections explain how you install the Mobile OWASP Top 10 report generator and how you configure it.

14.2.1. Installing the Report Generator

The Mobile OWASP Top 10 report generator is installed separately from Coverity Connect. You can obtain the installer for the Mobile OWASP Top 10 report generator installer from the *Downloads* page in Coverity Connect.

To install on Windows or Linux

1. Log in to Coverity Connect.

   In Help → Downloads, download the cov-report installer.

2. Install `cov-reports`.

3. Navigate to the Coverity report installation directory: for example,

   ```
   C:\Program Files\Coverity\Coverity Reports\bin
   ```

4. Verify that the file `cov-generate-mobileowasp-report` exists.

5. Before you run the `cov-generate-mobileowasp-report` command to generate the report, you must configure report generation, as explained in Configuring a Mobile OWASP Top 10 report .

14.2.2. Configuring a Mobile OWASP Top 10 Report

Configuration values determine the information displayed on the title page of your report and specify notification information for your project. You can configure a security report by setting specifications in your `config.yaml` configuration file. (A `config.yaml` template file is shipped with the report generator and is installed in the `config` directory.)

Here is an example `.yaml` configuration file:

```
version:
     schema-version: 5
connection:
    url: https://coverity.example.com:8443/
    username: admin
```

```
    ssl-ca-certs:
project:
title-page:
    company-name:
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:
locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:
```

Note

The Mobile OWASP Report does not require any additional report settings to be configured.

The following table describes each key in the file:

Key	Description
`company-name`	Name of your company. This is a mandatory field.
`components`	An optional comma-separated list of Coverity Connect component names, which inc map names. If the components are listed here, the report will include data only for th components; for example: `Default.lib` or `Default.src`.
`connection`	The URL of the Coverity Connect instance.
`issue-cutoff-count`	Some reports display information about individual issues. These reports bound the r displayed in order to control the size of the report. This bound is called the issue cut for CVSS, Security, PCIDSS, MobileOwasp, and Owasp2017 reports. Default value 10000 for Security report.
`issue-kind`	An optional comma-separated list of Coverity Connect issue kinds. If issue kinds are report will include only issues of the listed kinds. The possible values for `issue-kind` are as follows: • `Quality` • `Security` The following line is an example of using this option: `issue-kind: Quality`
`locale`	Locale of the report. Valid values are the following: `en_US`, `ja_JP`, `ko_KR`, and `zh_` `en_US`.

Key	Description
`logo`	Optional path to a logo file for your company. Valid image types are bmp, gif, jpg, ar maximum allowed image size is 210 pixels wide by 70 pixels high. Note that backsla path must be doubled.
`on-cert-trust`	Allows users to trust self-signed certificates sent by Coverity Connect. There are tw `trust` or `distrust`. Use `trust` to accept, use, and store a self-signed certificate `distrust` to reject connections from servers using self-signed certificates. The def
`organizational-unit-name`	Name of your division, group, team or other organizational unit. This is a mandatory
`organizational-unit-term`	Organizational unit term (e.g., division, group, team). This is a mandatory field.
`prepared-for`	Name of the entity for which the report was prepared. This is a mandatory field.
`prepared-by`	Name of the entity that prepared the report. This is a mandatory field.
`project`	Name of the Coverity Connect project.
`project-contact-email`	Project contact email address. It is used for the following reports: CIR, CVSS, PCID and OWASP2017. This is a mandatory field.
`project-name`	Name of the software development project. May be distinct from the Coverity Conne This is an optional field.
`project-version`	Lists the project version number. This is a mandatory field.
`snapshot-date`	Retrieves the most recent snapshot of each stream in the project whose date is less the given date. Format is "DD/MM/YYYY".
`snapshot-id`	Retrieves the defects of a specific snapshot id, instead of using the latest snapshot associated with the project.
`title-page`	Describes the fields in the title page of the report.
`username`	Coverity Connect username. Password or other authentication key
`version`	The version of the `.yaml` file's schema.

For more information about schema configurations, see Chapter 6.1, *Configuring Report Generators*

14.2.3. Generating a Mobile OWASP Report

After you have saved your report configuration, you can generate a Mobile OWASP Report by running the following command:

```
cov-generate-mobile-owasp-report <configuration-file> --password <spec> [--project
 <project-name>]
```

- For `<configuration-file>`, specify the path to the `config.yaml` file you created.
- For password, specify the password you use when you connect to Coverity Connect.
- The project name can either be set (via command line) with the `--project` option or it can be entered directly into the YAML configuration file.

For example,

```
cov-generate-mobile-owasp-report mydir/config.yaml --password console --project demo
```

To get help with command syntax, enter:

```
cov-generate-mobile-owasp-report --help
```

Part 15. Coverity PCI DSS

Table of Contents

15.1. Overview 436
    15.1.1. Working with the PCI DSS Report Generator 436
    15.1.2. Interpreting the PCI DSS Report 436
15.2. Working with the PCI DSS Report Generator 438
    15.2.1. Installing the Report Generator 438
    15.2.2. Configuring a PCI DSS Report 438
    15.2.3. Generating a PCI DSS Report 440

Chapter 15.1. Overview

Table of Contents

15.1.1. Working with the PCI DSS Report Generator 436
15.1.2. Interpreting the PCI DSS Report 436

The PCI DSS report generator uses analysis results for a Coverity Connect project to evaluate the analyzed codebase. Based on this evaluation, it creates a report, which details the assessments that were done, provides a summary of findings, and specifies the remediations needed. Information from this report is of special interest to application security assurance teams and their clients.

The PCI Security Standards Council is a global forum for the ongoing development, enhancement, storage, dissemination and implementation of security standards for account data protection. The Payment Card Industry Data Security Standard (PCI DSS) was developed to encourage and enhance cardholder data security and facilitate the broad adoption of consistent data security measures globally. PCI DSS provides a baseline of technical and operational requirements designed to protect account data.

Important

You are required to generate a CVSS report before you can use the PCI DSS report generator because the latter depends upon the vulnerability scoring system defined by CVSS. For information about generating a CVSS report, see "Coverity CVSS Report" in the *Coverity Platform User and Administrator Guide*.

You can localize reports by setting the `locale` filed of your report generation configuration file.

This chapter describes the workflow needed for generating a PCI DSS report and explains how you interpret report findings.

15.1.1. Working with the PCI DSS Report Generator

The user workflow for working with the PCI DSS Report Generator is as follows:

1. Install the report generator using the Coverity Reports Installer. See Installing the Security Report Generator for more information.
2. Configure the report. This allows you to specify information that is displayed on the report's cover page and to specify notification information.

   See Configuring a PCI DSS report for more information.
3. Generate the report.

15.1.2. Interpreting the PCI DSS Report

The report is divided into three basic sections that describe the issues found in increasing amount of detail:

- **Executive Summary** provides tabular and graphic summary information for the issues found.
- **Analysis Details** shows the number of issues associated with each OWASP Top 10 category and PCI DSS v3.2 catgegory. For each defect found, the report describes the issue type, remediation information, the file name, and line number and the date it was first detected.
- **Methodology** describes basic elements associated with report output.

Each section of the report includes detailed information about the vulnerabilities found and remediation action.

Chapter 15.2. Working with the PCI DSS Report Generator

Table of Contents

15.2.1. Installing the Report Generator 438
15.2.2. Configuring a PCI DSS Report 438
15.2.3. Generating a PCI DSS Report 440

The following sections explain how you install the PCI DSS report generator and how you configure it.

15.2.1. Installing the Report Generator

The PCI DSS report generator is installed separately from Coverity Connect. You can obtain the installer for the PCI DSS report generator installer from the *Downloads* page in Coverity Connect.

To install on Windows or Linux

1. Log in to Coverity Connect.

   In Help → Downloads, download the cov-report installer.

2. Install `cov-reports`.
3. Navigate to the Coverity report installation directory: for example,

   ```
   C:\Program Files\Coverity\Coverity Reports\bin
   ```

4. Verify that the file `cov-generate-pci-dss-report` exists.
5. Before you run the `cov-generate-pci-dss-report` command to generate the report, you must configure report generation, as explained in Configuring a PCI DSS report.

15.2.2. Configuring a PCI DSS Report

Configuration values determine the information displayed on the title page of your report and specify notification information for your project. You can configure a security report by setting specifications in your `config.yaml` configuration file. (A `config.yaml` template file is shipped with the report generator and is installed in the `config` directory.)

Here is an example `.yaml` configuration file for the PCI DSS Report:

```
version:
     schema-version: 5
connection:
    url: https://coverity.example.com:8443/
    username: admin
    ssl-ca-certs:
project:
title-page:
    company-name:
```

```
    project-name:
    project-version: 0.9
    logo:
    organizational-unit-name: Widgets
    organizational-unit-term: Division
    prepared-for:
    project-contact-email:
    prepared-by:
locale:
issue-cutoff-count: 200
snapshot-id:
snapshot-date:
issue-kind:
components:
```

Note

The PCI DSS Report does not require any additional report settings to be configured.

The following table describes each key in the file:

Field	Description
`company-name`	Name of your company. This is a mandatory field.
`components`	An optional comma-separated list of Coverity Connect component names, which inc map names. If the components are listed here, the report will include data only for th components; for example: `Default.lib` or `Default.src`.
`connection`	The URL of the Coverity Connect instance.
`issue-cutoff-count`	Some reports display information about individual issues. These reports bound the r displayed in order to control the size of the report. This bound is called the issue cut for CVSS, Security, PCIDSS, MobileOwasp, and Owasp2017 reports. Default value 10000 for Security report.
`issue-kind`	An optional comma-separated list of Coverity Connect issue kinds. If issue kinds are report will include only issues of the listed kinds. The possible values for `issue-kind` are as follows: • `Quality` • `Security` The following line is an example of using this option: `issue-kind: Quality`
`locale`	Locale of the report. Valid values are the following: `en_US`, `ja_JP`, `ko_KR`, and `zh_` `en_US`.
`logo`	Optional path to a logo file for your company. Valid image types are bmp, gif, jpg, ar maximum allowed image size is 210 pixels wide by 70 pixels high. Note that backsla path must be doubled.

Field	Description
`on-cert-trust`	Allows users to trust self-signed certificates sent by Coverity Connect. There are tw `trust` or `distrust`. Use `trust` to accept, use, and store a self-signed certificate `distrust` to reject connections from servers using self-signed certificates. The def
`organizational-unit-name`	Name of your division, group, team or other organizational unit. This is a mandatory
`organizational-unit-term`	Organizational unit term (e.g., division, group, team). This is a mandatory field.
`prepared-for`	Name of the entity for which the report was prepared. This is a mandatory field.
`prepared-by`	Name of the entity that prepared the report. This is a mandatory field.
`project`	Name of the Coverity Connect project.
`project-contact-email`	Project contact email address. It is used for the following reports: CIR, CVSS, PCID and OWASP2017. This is a mandatory field.
`project-name`	Name of the software development project. May be distinct from the Coverity Conne This is an optional field.
`project-version`	Lists the project version number. This is a mandatory field.
`snapshot-date`	Retrieves the most recent snapshot of each stream in the project whose date is less the given date. Format is "DD/MM/YYYY".
`snapshot-id`	Retrieves the defects of a specific snapshot id, instead of using the latest snapshot associated with the project.
`title-page`	Describes the fields in the title page of the report.
`username`	Coverity Connect username. Password or other authentication key
`version`	The version of the `.yaml` file's schema.

For more information about schema configurations, see Chapter 6.1, *Configuring Report Generators*

15.2.3. Generating a PCI DSS Report

Now that you've configured your report and set your environment variable, you can generate the PCI DSS Report by running the following command:

```
cov-generate-pci-dss-report <configuration-file> --password <spec> [--project
 <project-name>] --report
```

- For `<configuration-file>`, specify the path to the `config.yaml` file you created.
- For password, specify the password you use when you connect to Coverity Connect.
- The project name can either be set (via command line) with the `--project` option or it can be entered directly into the YAML configuration file.

For example,

```
cov-generate-pci-dss-report mydir/config.yaml --password console --project demo
```

To get help with command syntax, enter:

```
cov-generate-pci-dss-report --help
```