



# Coverity 2020.12 Installation and Deployment Guide :

Planning, Installation, Tuning, and Supported Platforms Guidance for Coverity Analysis, Coverity Platform, and Coverity Desktop.

Copyright 2020 Synopsys, Inc. All rights reserved worldwide.

---

## Table of Contents

About this guide .....	iv
1. Coverity product overview .....	iv
2. Installation and deployment overview .....	vi
1. Installing Coverity Platform components .....	1
1.1. Installing Coverity Connect .....	1
1.2. Coverity Platform installation modes .....	7
1.3. Advanced installation options .....	18
1.4. Post-installation notes .....	23
1.5. Installing Coverity Reports .....	24
1.6. Coverity Reports installation modes .....	25
2. Installing Coverity Analysis components .....	27
2.1. Installing Coverity Analysis .....	28
2.2. Coverity Analysis installation modes .....	31
2.3. Coverity Analysis license options .....	36
2.4. Using an archive file to install Coverity Analysis .....	48
3. Installing Coverity Desktop components .....	49
3.1. Installing Coverity Desktop for Eclipse, Wind River Workbench, QNX Momentics, and IBM RTC .....	49
3.2. Installing Coverity Desktop for Microsoft Visual Studio .....	54
3.3. Installing Coverity Desktop for IntelliJ IDEA and Android Studio .....	55
3.4. Installing Coverity Analysis for local analysis .....	56
4. Installing Architecture Analysis .....	57
5. Deployment planning .....	61
5.1. Deployment checklist .....	61
5.2. How Coverity products integrate into a build system .....	63
5.3. Coverity Analysis deployment models .....	64
5.4. Coverity Connect deployment options .....	67
5.5. Other deployment models and features .....	72
6. Hardware and network recommendations and requirements .....	75
6.1. Coverity Connect hardware .....	75
6.2. Coverity Analysis hardware .....	76
6.3. Coverity Connect Network Connectivity Requirements .....	79
7. Supported platforms .....	81
7.1. Coverity product components: Supported versions and compatibility .....	81
7.2. Coverity Connect, Policy Manager, and Reports platforms .....	82
7.3. Coverity Analysis .....	84
7.4. Coverity Test Advisor SCM and platform support .....	91
7.5. Architecture Analysis .....	98
7.6. Coverity Desktop .....	99
8. Supported languages, compilers, and frameworks for Coverity Analysis .....	108
8.1. C/C++ compilers .....	108
8.2. C# compilers .....	126
8.3. CUDA compilers .....	126
8.4. Go compilers .....	127
8.5. Java compilers .....	128
8.6. Kotlin compilers .....	128

8.7. Scala compilers .....	129
8.8. Swift compilers .....	129
8.9. Visual Basic compilers .....	130
8.10. Third-party compilers .....	130
8.11. Frameworks .....	130
9. Coverity Connect system tuning, maintenance, and monitoring .....	134
9.1. Post-installation: what's next? .....	134
9.2. Coverity Connect system and database tuning .....	136
9.3. Monitoring and diagnosing Coverity Connect performance .....	139
A. Coverity Dynamic Analysis for Java .....	147
A.1. Supported compilers .....	147
B. Coverity Glossary .....	148
C. Legal Notice .....	160
C.1. Legal Notice .....	160

---

# About this guide

## Table of Contents

1. Coverity product overview .....	iv
2. Installation and deployment overview .....	vi

The *Coverity 2020.12 Installation and Deployment Guide* contains information to help you plan for a Coverity product deployment. This guide is intended for system architects, deployment architects, and build engineers who are responsible for the planning and installation of Coverity tools.

Product functionality, configuration, and operation instructions that are unrelated to deployment are not covered in this manual, so it is assumed that you have a knowledge of Coverity products and features. For more detailed information about Coverity tools, refer to the following documentation:

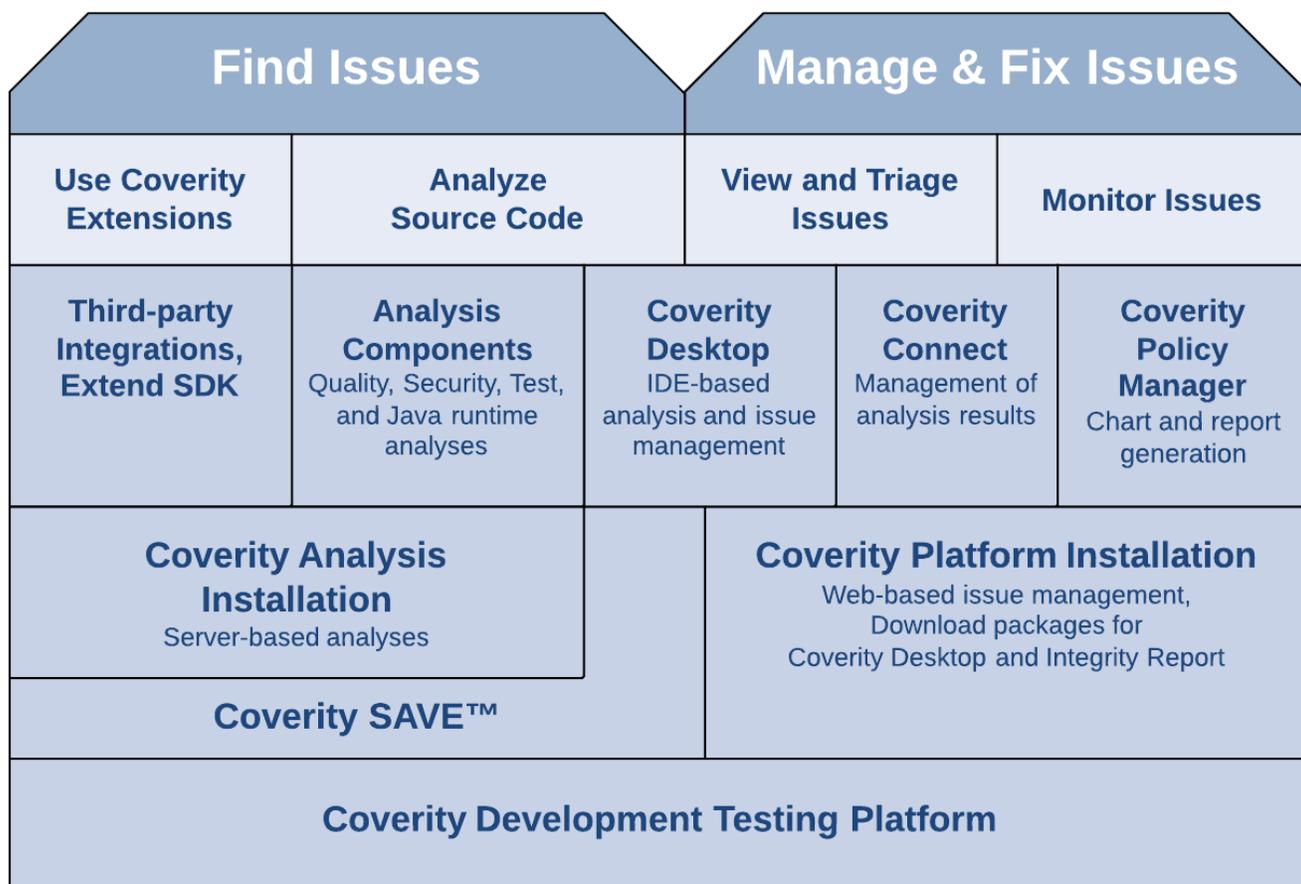
- *Coverity Platform 2020.12 User and Administrator Guide* - Guidance for system administrators and end users of Coverity Connect, Coverity Policy Manager, and Coverity Integrity Report.
- *Coverity Analysis 2020.12 User and Administrator Guide* - Guidance for build engineers or system architects to integrate and configure `cov-analysis` tools with the nightly build system.

The complete documentation set for Coverity products can be found under the root installation directory of either Coverity Platform or Coverity Analysis:

```
<coverity_product_install_root>/doc/
```

## 1. Coverity product overview

There are two installer applications that Coverity provides, Coverity Analysis and Coverity Platform. The following figure shows the relationship of issues found by Coverity tools (and third-party tools) to the Coverity components that help you manage and fix them.



### Coverity Analysis Installation

Coverity Analysis components and extensions are built on top of Coverity (Coverity Analysis), the set of foundational technologies that support the use of Coverity checkers to detect quality defects (Quality issues), Potential Security Vulnerabilities (Security issues), test violations (Test Advisor issues), and Java runtime defects (Dynamic Analysis issues).

- **Analysis Components**

- Coverity Analysis (formerly called Coverity Static Analysis)
- Test Advisor
- Dynamic Analysis

- Architecture Analysis
- **Analysis Extensions**
  - Third Party Integration Toolkit supports the addition of issues found by third-party products to the Coverity Connect database.
  - Compiler Integration Toolkit (CIT) allows you to extend the set of compilers that can be used to build source code for Coverity code analysis.
  - CodeXM supports the development of custom checkers.
- **Coverity Platform Installation**

Coverity Platform components support Web-based management of issues found by Coverity Analysis and third-party tools.

  - **Coverity Desktop for Eclipse and for Microsoft Visual Studio** supports IDE-based analysis, management, and remediation of issues. Coverity Desktop relies on Coverity Analysis to support local code analysis.
  - **Coverity Connect** is a Web-based application that helps software developers and team leaders manage and fix issues found using Coverity Analysis and third party tools. The Coverity Connect interface provides descriptions of the issues found by Coverity Analysis and shows where the issues exist in the source code.
    - **Downloads:** Coverity Connect also provides access to downloads for the Coverity Desktop plug-ins and the Coverity Reports installer. For details about these downloads, see Section 1.1, “Installing Coverity Connect”.
    - **Extensions:** Coverity Connect also provides access to the WSDL files for the Configuration and Defect services of the Coverity Platform Web Services API.
  - **Coverity Policy Manager** is a Web-based solution for code governance that enables software development organizations to set policies for code quality and security, and then manage, monitor, and report on these policies as code is tested. Coverity Policy Manager provides managers and leaders who have cross-organizational responsibilities with the insight needed to better align quality goals with business objectives, to enforce standards across the development organization, and to manage third-party code dependencies.

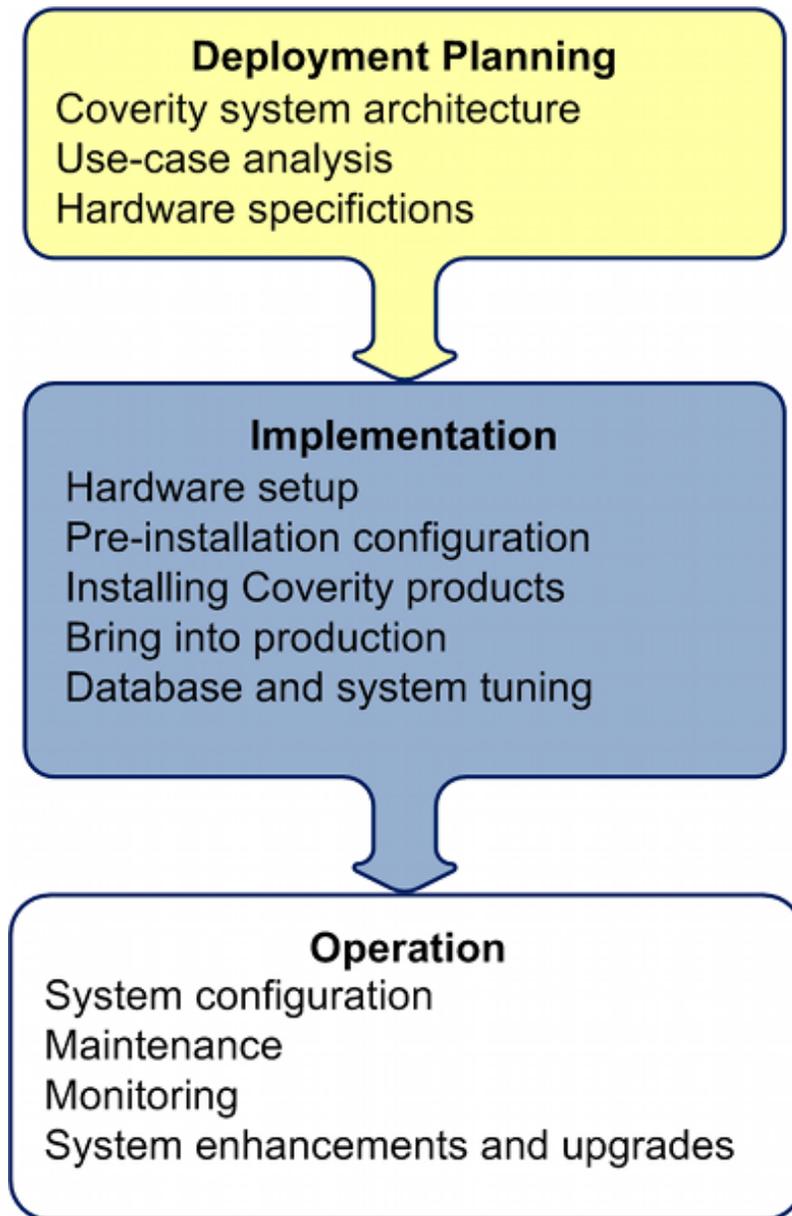
## 2. Installation and deployment overview

- **Deployment planning** represents the research and planning phase of the overall deployment. This section provides information to help you identify your organization's needs.
- **Installation** covers the phase during which you work from specifications and plans created during the deployment planning phase to install the Coverity tools. Installation is covered in several sections of this guide: Chapter 1, *Installing Coverity Platform components*, Chapter 2, *Installing Coverity Analysis*

*components*, Chapter 3, *Installing Coverity Desktop components*, Chapter 4, *Installing Architecture Analysis*.

- **Coverity Connect system tuning, maintenance, and monitoring** provides recommendations for system tuning and database maintenance procedures to ensure optimal performance. In addition, this section also provides tips for monitoring your system so that you can collect trend data to track the performance of your system over time.
- **Supported Platforms** provides matrices of Coverity products, compilers, components, and their supported operating systems and runtime environments.

Coverity deployment follows a process of planning, designing, and implementing the Coverity tools. The following diagram shows the deployment cycle and the cycle's phases that are discussed in this guide:



## 2.1. Deployment planning

Deployment planning is a critical step to successfully implement Coverity tools. Each organization has its own set of goals, requirements, and priorities. Successful deployment planning is the result of preparation, analysis, and implementation. Errors that might occur anywhere during the planning process can result in a system that might under-perform, be difficult to maintain, be too expensive to operate, could waste resources, or could be unable to scale to meet your organization's increasing needs.

#### Understanding the Coverity system architecture

During this step, you review the system deployment options to understand how Coverity products integrate with your existing build system, and choose the hardware infrastructure so that your deployed system runs efficiently and quickly. For more information, see Chapter 5, *Deployment planning*.

#### Hardware specifications

This section provides hardware recommendations to implement your hardware infrastructure on which Coverity products will run.

#### Security hardening

For information on maintaining the Coverity installation in a secure manner, please see "Securing Coverity Connect" in the *Coverity Platform 2020.12 User and Administrator Guide*.

## 2.2. Implementation

During the implementation phase of the deployment cycle, you work from specifications and plans created during the deployment planning phase to build and test the deployment, ultimately rolling out the deployment into production.

#### Hardware setup

The implementation of the hardware that will host the Coverity deployment.

#### Installing Coverity products

The process of installing the Coverity products based on the decisions that you made during the deployment planning phase. For installation instructions, see Chapter 1, *Installing Coverity Platform components*.

#### Database and system tuning

Tuning the database and system settings for Coverity Connect. For more information, see Section 9.2, "Coverity Connect system and database tuning".

#### Deployment scenario analysis

This section provides deployment use cases that describe the steps and decisions that Coverity customers have made to successfully integrate Coverity products into their system. You can review these use cases and apply the solutions to your deployment planning. For more information, see ???

#### Bring into production

Get the system up and running so that all components are functioning together (also see "System Configuration" below).

## 2.3. Operation

The Operating phase consists of post-deployment tasks to configure, maintain, and monitor your organization's Coverity deployment.

#### System configuration

The "What's Next" list provides an overview of tasks that you need to complete in order to set up your system (mainly administrative tasks).

#### System maintenance

System maintenance provides recommendations for you to enable features and settings that will maintain the size of the Coverity Connect database.

#### Monitoring the system

Monitoring the system is an important part of the process. It is suggested that you record trend data over time so you can make sure that the entire system is performing optimally. Guidelines for monitoring are provided in Section 9.3, “Monitoring and diagnosing Coverity Connect performance”.

#### System enhancements and upgrades

This step represents the ongoing improvements to the deployment, through upgrading new versions and implementing new features.

---

# Chapter 1. Installing Coverity Platform components

## Table of Contents

1.1. Installing Coverity Connect .....	1
1.2. Coverity Platform installation modes .....	7
1.3. Advanced installation options .....	18
1.4. Post-installation notes .....	23
1.5. Installing Coverity Reports .....	24
1.6. Coverity Reports installation modes .....	25

This part of the guide is for administrators who install Coverity Platform, the name of the tool that installs and upgrades the following components:

- Coverity Connect
- Coverity Policy Manager

### Upgrades

Upgrade procedures are described separately in the [Coverity 2020.12 Upgrade Guide](#), which not only provides steps to perform the upgrade, but also includes important upgrade notes on new features that will affect your upgrade.

## 1.1. Installing Coverity Connect

This procedure is referred to as a Coverity Connect installation because all the Coverity Platform components (including Coverity Policy Manager) are installed as part of Coverity Connect. In addition, once Coverity Connect is running, IDE users can download and install Coverity Desktop plug-ins for Eclipse, QNX Momentics, Wind River Workbench, IBM RTC, Visual Studio, IntelliJ, Android Studio, and Gradle from the *Downloads* link in Coverity Connect. The link also provides access to the Coverity Reports installer, which installs Coverity Integrity Report, Security Report, and Coverity MISRA Report.

Before installing to a production environment:

- You need to determine which database you intend to use. The Coverity Platform installer provides the option of using a Coverity Connect database that is embedded in the installation package, or connecting to a pre-existing external PostgreSQL database for Coverity Connect.
- It is important to understand deployment considerations and hardware recommendations described in Chapter 5, *Deployment planning* and Chapter 6, *Hardware and network recommendations and requirements*.
- Coverity recommends that you or your database administrator choose and tune your file system settings. Database-dependent application behavior relies on I/O (input/output), and the system I/O can be further tuned for better performance. A good source for such information can be found in *PostgreSQL 9.0 High Performance* authored by Gregory Smith and published by Packt Publishing.

Note that if you are simply performing a preliminary installation for trial purposes, you might opt to use the embedded database and proceed with the installation without reference to the deployment considerations

described above. However, if you want to fully evaluate your installation, you should heed all preliminary requirements and recommendations.

Also be sure that you have the following information prior to launching the installer:

- Desired installation directory.
- Location of your Coverity Connect license file.
- Desired ports for Coverity Connect communications.
- External PostgreSQL credentials (if not using the embedded database).
- Amount of available RAM (for embedded database performance).

### To install Coverity Connect:

1. Verify that your installation environment meets Coverity software and platform requirements.  
For details, see Section 7.2, “Coverity Connect, Policy Manager, and Reports platforms”.
2. On the machine where you want to install Coverity Connect and other Coverity Platform components, download the Coverity Platform installer file for your operating system.

**On Linux systems:** If the `/tmp` directory on this machine is set to `noexec`, you must set the Java property `java.io.tmpdir` to a location that is not set to `noexec`.

3. If you do not plan to run Coverity Connect as a service, choose or create an operating system user under which the Coverity Connect server software will run.

On Windows and Linux systems, this is the user who can start or stop Coverity Connect.

**On Linux systems:** If this user's home directory is set to `noexec`, you must set the Java property `jna.tmpdir` to a location that is not set to `noexec`.

On Windows systems that run Coverity Connect as a service, the Administrator can start and stop Coverity Connect.

4. Run the installer script or program for your operating system:

Linux, 64-bit

```
cov-platform-linux64-2020.12.sh
```

Windows, 64-bit

```
cov-platform-win64-2020.12.exe
```

#### **Note**

On Linux systems, you must not be logged in as root to install Coverity Platform.

For Windows, double-click the `.exe` program. It is recommended that you install Coverity Connect with Windows Administrator privileges (installing Coverity Connect as a service requires it). To do so, right click the installer and choose *Run as Administrator*.

For Linux, run the installer script in a Bourne shell, for example:

```
> ./cov-platform-linux64-2020.12.sh
```

The installer uses a text-based console mode or a graphical mode. The installation choices for graphical and console modes are equivalent.

- To change the installer mode, see Section 1.2, “Coverity Platform installation modes”.

5. Complete the installation process:

- a. Select and accept the license agreement for your region of the world.
- b. Select the *Fresh Installation* option.

Note that if you intend to upgrade an existing Coverity Connect instance, you should follow one of the upgrade procedures in the *Coverity 2020.12 Upgrade Guide* [↗](#) instead of using the steps in this guide.

- c. Enter the destination directory for the installation.

The destination directory should be on a local file system. In this documentation, the Coverity Platform installation directory is referred to as `<install_dir_cp>`.

 **Note**

For a fresh installation, you must use an empty directory. If an earlier version of Coverity Connect is installed in the specified destination directory, the Coverity Platform installer will treat the installation as an upgrade and attempt to install over the existing instance.

 **Note**

Coverity Platform must be installed to a location where the user has full permissions. Specifically, it is recommended that Coverity Platform not be installed into a location that is only accessible by root.

- d. Enter the location of the Coverity license file (`license.dat`).

 **Note**

If your license is invalid, you will not be able to log into Coverity Connect. The installer will not alert you if there is a problem with your license.

- e. Choose the database type for Coverity Connect.

You can choose the embedded PostgreSQL database that is bundled within the installer (which is recommended) or opt to connect to an external database that is hosted on a PostgreSQL server. For supported PostgreSQL versions, see Section 7.2.1, “Coverity Connect software requirements”.

Note that the external database option is only intended for experienced PostgreSQL DBAs. For the installer settings for an external database, see Section 1.3.1, “Using an external PostgreSQL database with Coverity Connect”.

- f. If you chose the Embedded database option, enter the Coverity Connect database port and the database location.

The TCP port that the embedded database server uses to listen for connections is only used for localhost connections, so you should not need to configure your firewall. The default is 5432.

Choose the location for the embedded Coverity Connect database files. This location should be on a local volume that has at least 2GB of free space. The default is `<install_dir_cp>/database`.

- g. If you chose the External database option, enter the following PostgreSQL configuration parameters for the database you will use:

- PostgreSQL server name
- Database port
- Database name
- Database user
- Database password

If you have not already done so, you must ask your database administrator (DBA) to create a database and user role for Coverity Connect. You (the user) must have privileges to create

and alter tables in the database. For more information, see Section 1.3, “Advanced installation options”.

- h. Select your Coverity Connect database performance tuning option.

Coverity Platform will warn you if your system does not meet the RAM requirements on your system and you will not be able to select that tuning option.

To help with performance, you can choose from the following options:

1. Production - This tuning configuration is allowed to use all of the installed RAM on your system.
2. Demo - Will run on a small computer and does not require the full 8GB of recommended RAM. *You should not use this option for any production system. This option should be used for proof-of-concept or testing environments only.*

Depending on your selection, the installer will suitably configure the JVM settings and PostgreSQL configuration. For more information, See sections Section 9.2.1, “PostgreSQL database tuning: embedded database” and Section 9.2.2, “JVM settings”.

 **Note**

These deployment tunings do not affect database memory allocation when installed with an external database; The PostgreSQL settings will not be modified. However, the JVM settings are set whether an embedded or external database is used.

- i. (Windows only) Choose whether or not to create Start Menu entries.

If you choose to have Start Menu entries created, you have the option to change the default Start Menu folder, and you can choose whether or not to create shortcut entries for all users.

- j. Choose and confirm the Coverity Connect administrator password.

This is the password for the Coverity Connect administrator (`admin`) account. Administrators can use this account to log in with a web browser and configure users, create projects, and manage other administrative settings.

- k. (Windows only) Choose to run Coverity Platform as a service.

You must have Windows Administrator privileges to install and run Coverity Connect as a service. The service runs as the built-in Windows account `LocalService`. You cannot change this setting during the installation process.

- l. Choose the host name configuration.

Choose from the host name of your machine, or the IP address.

- m. Enter the HTTP port number.

This is the general purpose port for the Coverity Connect server and is the preferred port for transferring analysis data to the server. Users connect to the HTTP port to access Coverity Connect with a web browser and web services clients. If you are using a firewall, make sure it is configured to allow incoming connections on this port. This is referred to as the `<http_port>`. The default is 8080

If you want to configure Coverity Connect with a secure server, select *Provide HTTPS service*.

The HTTPS protocol requires an installed server certificate from a certificate authority. If you choose this option, you must specify the *HTTPS Port number*. The default is 8443.

n. Configure the following ports:

- Commit port - (Optional) Although the HTTP(S) port is the preferred port for receiving data from the build and analysis processes, you *can* use the Commit port instead. Note, however, that the Commit port will be deprecated for this purpose in a future release. If you plan to use the Commit port and have a firewall in place, make sure the firewall is configured to allow incoming connections on this port. The default is 9090.

 **Note**

The Commit port will be deprecated for committing analysis data but will continue to be used in clustered environments for Coordinator-to-Subscriber communication.

- Control port - The Port used by the embedded application server for internal server communication. The default is 8005.

6. If you changed any of the default ports, make a note of the port number(s) for later use, such as for the `cov-commit-defects --dataport` command.

After the installation completes, a record of some of this information is also available in the following files:

- `<install_dir_cp>/config/cim.properties`
- `<install_dir_cp>/config/web.properties`
- `<install_dir_cp>/config/system.properties`

- The Tomcat configuration files located at:

`<install_dir_cp>/server/base/conf/server.xml`

For example, see `commitPort` in `cim.properties`.

7. Check your installation:

a. Launch Coverity Connect by entering one of the following URLs into your web browser:

- `http://<hostname>:<http_port>`

- `https://<hostname>:<https_port>`
- b. Sign into Coverity Connect with user name `admin`, and the administrator password that you previously created.

If you are using HTTPS and you open Coverity Connect in a web browser before you have enabled the server for SSL, you will receive an error message that you do not have the correct certificate installed. After you have configured Coverity Connect to use the appropriate certificates, you will be able to log into the system. For more information, see the "Configuring Coverity Connect to use SSL" section in the *Coverity Platform 2020.12 User and Administrator Guide*.

- c. Review Section 1.4, "Post-installation notes".

### To uninstall Coverity Platform:

1. Go to `<install_dir_cp>`.
2. On Linux, run the `uninstall` script.

On Windows, run the `uninstall.exe` program, and follow the prompts. Note that you should uninstall as an administrative user, particularly if you were running Coverity Connect as a service.

The `uninstall` script provides an option for removing user-supplied data. If you want to retain these files for your own purposes, you should back them up before uninstalling Coverity Connect.

## 1.2. Coverity Platform installation modes

You can complete the series of installation steps using either a graphical or console (command-line) interface. Alternatively, you can run the silent installer to complete the installation from the command line in a single step.

### 1.2.1. Selecting an installation mode

On Linux-based systems, the console mode (with text-based prompts) is the default, and on Windows systems graphical mode is the default. For both Coverity Platform and Coverity Analysis, you can use the options describe below to select any of the other modes.

#### To specify an installer mode:

- Console mode: Use the command-line option `-c` for console mode.
- Graphical mode: Use the `-g` option for graphical mode.
- Silent mode: Use the `-q` option to run the silent (quiet) installer.

For example, to start the graphical installer on Linux:

```
> cov-platform-linux64-2020.12.sh -g
```

On Windows, when using the command prompt, you must precede the command with a `start /wait` command. Additionally, if the executable filename contains spaces, you must include empty, double quotes even if the filename is double-quoted:

```
> start /wait "" "<executable name>" -c
```

You can use the empty quotes even when the executable name does not contain spaces. For example:

```
> start /wait "" cov-platform-win64-2020.12.exe -c
```

### 1.2.2. Coverity Platform graphical installer

The Coverity Platform installer (see Section 1.2.1, “Selecting an installation mode”) provides options for completing the most common Coverity Platform installation and upgrade use cases:

- *Fresh install*
- *In-place upgrade*
- *Backup-and-restore upgrade*
- *Intermachine upgrade*
- *Upgrade preparation*

For detailed instructions on all Coverity Connect upgrades, including external database upgrades, see "Upgrading Coverity Connect" in the *Coverity 2020.12 Upgrade Guide* [↗](#).

### 1.2.3. Coverity Connect silent installer

The Coverity Connect silent installer allows you to specify all of the installation configuration details on the command line so you do not need to run the "step-through" process either through the command line (-c) or the graphical (-g) installer modes.

To run the silent installer, specify the installation utility with the `-q` option, followed by the installation parameters. The `-q` option and the installation parameters must all be on the same command line. The following example installs a "production" Coverity Platform deployment with an embedded database.

```
./cov-platform-linux64-2018.06.sh -q \  
--installation.dir ~/cov-platform-linux64-2020.12 \  
--license.region=0 \  
--license.agreement=agree \  
--license.path=/tmp/license.dat \  
--db.type=0 \  
--db.embedded.performance=0 \  
--admin.password.env=ADMIN_PASSWORD \  
--hostname=myhostname \  
--http.port=14800 \  

```

## Installing Coverity Platform components

```
--commit.port=14801 \  
--control.port=14802 \  
--db.embedded.port=14803 \  

```

If you are installing on Windows, use the `-q -console` options preceded by a `start /wait` command. If the executable filename contains spaces, precede it with empty double quotes, even if the filename itself is double-quoted, for example:

```
> start /wait "" "<my executable name>" -q -console
```

### Note

You can include the empty double quotes whether or not the executable name contains spaces.

The silent installer accepts the following options. Note that not all of the options are required. For example, you do not need to specify the options for an external (`postgres`) database if you are installing Coverity Connect with an embedded database. If you use any of the following parameters, you should provide specifically assigned values. Some values, if left blank, will accept the default value, but this is not a recommended practice. For more information about the installation options, see Section 1.1, “Installing Coverity Connect”.

### Installer command line options

#### Note

Do not use the `-v` prefix with these options:

Option	Description
<code>-q</code>	Required. Enables the silent installer.
<code>-console</code>	Required on Windows. Displays status messages in the console from which you invoked the silent installer.

### General parameters

Parameter	Description
<code>--install.type=&lt;0   1   2   3   4&gt;</code>	Sets the installer to backup-and-restore mode. For this option, the following values are available: <ul style="list-style-type: none"><li>• 0 - Fresh install</li><li>• 1 - In-place upgrade</li><li>• 2 - Backup and restore</li><li>• 3 - Intermachine upgrade</li><li>• 4 - Upgrade preparation</li></ul>

## Installing Coverity Platform components

---

Parameter	Description
	Otherwise, the default value is set to 0.
<code>--license.agreement=&lt;agree&gt;</code>	Required. Agree to the terms of the Coverity product license.
<code>--license.region=&lt;0   1   2   3   4   5   6&gt;</code>	<p>Required. Specifies your region, used for selecting the correct product license. The following values (and representative region) are available:</p> <ul style="list-style-type: none"> <li>• 0 - Americas, Africa, and Israel</li> <li>• 1 - Japan</li> <li>• 2 - Taiwan</li> <li>• 3 - China</li> <li>• 4 - Republic of Korea</li> <li>• 5 - International license (for any countries not mentioned in 0-4)</li> <li>• 6 - Evaluation Only. This installation is being used solely for evaluation purposes, and is not for production use</li> </ul>

### Fresh installation parameters

These parameters specify the options used for completing a fresh installation of Coverity Connect. For a fresh installation, the `--install.type` setting value must be set to 0.

Parameter	Description
<code>--accept.https=&lt;true   false&gt;</code>	Specifies that Coverity Connect should use SSL (HTTPS) communication. The HTTPS protocol requires an installed server certificate from a certificate authority. The default value is <code>false</code> .
<code>--admin.password=&lt;password&gt;</code>	<b>Deprecated:</b> This option is not recommended because the password might be visible to any user who has access to the list of running processes during the installation. Post installation, the password might also be visible to users who have access to the command history. Use either the <code>admin.password.file</code> or <code>admin.password.env</code> option instead.
<code>--admin.password.env=&lt;environment variable name&gt;</code>	The name of an environment variable containing the administrator password that you will use to log into Coverity Connect upon installation.

Parameter	Description
<code>--admin.password.file=&lt;absolute file path&gt;</code>	The absolute file path of a file containing the administrator password that you will use to log into Coverity Connect upon installation.
<code>--commit.port=&lt;port&gt;</code>	Optional. Specifies the Coverity Connect Commit port. Default value is 9090.  The HTTP(S) port is recommended over the Commit port. The Commit port will be deprecated in a future release.
<code>--control.port=&lt;port&gt;</code>	Recommended. Specifies the port used by the embedded application server for internal server communication. Default value is 8005.
<code>--db.type=&lt;0   1&gt;</code>	Specifies the type of database that is used with the Coverity Connect application server. The following option values are available: <ul style="list-style-type: none"> <li>• 0 - Embedded database parameters  This includes the <code>--db.embedded.data</code> and <code>--db.embedded.port</code> options.</li> <li>• 1 - External database parameters  This includes the following options: <code>--db.external.host</code>, <code>--db.external.password</code>, <code>--db.external.port</code>, <code>--db.external.name</code>, and <code>db.external.user</code>.</li> </ul> <p> <b>Note</b></p> <p>The external database must be created before you run the installer.</p> <p>For more information, see Section 1.3, “Advanced installation options”.</p> <p>Otherwise, the default value is 0. This specifies that Coverity Connect will connect to an existing, external PostgreSQL database.</p>
<code>--db.embedded.performance=&lt;0   1&gt;</code>	Selects the Coverity Connect database performance tuning option. Based on your installation type, choose one of the following values: <ul style="list-style-type: none"> <li>• 0 - Production</li> </ul>

## Installing Coverity Platform components

Parameter	Description
	<ul style="list-style-type: none"> <li>1 - Demo</li> </ul> <p>The default value is 0.</p>
<code>--hostname=&lt;host-name&gt;</code>	Required. Specifies the hostname of the Coverity Connect server.
<code>--http.port=&lt;port&gt;</code>	Recommended. Specifies the HTTP port number for Coverity Connect. Default value is 8080.
<code>--https.port=&lt;port&gt;</code>	Recommended. Specifies the HTTPS port number for Coverity Connect. Default value is 8443. Valid if the <code>--accept.https</code> value is set to <code>true</code> .
<code>--installation.dir=&lt;directory-path&gt;</code>	Location to which Coverity Platform is installed. This must be an absolute directory path.
<code>--license.path=&lt;path&gt;</code>	Required. Specifies the full directory path of a Coverity <code>license.dat</code> file.
<code>--service.enable=&lt;true   false&gt;</code>	Specifies that Coverity Connect will be enabled as a Windows service. This parameter is optional and will not affect any installations on non-Windows platforms. Default value is <code>false</code> .

### In-place upgrade parameters

These parameters specify the options used for completing an in-place upgrade of an installed instance of Coverity Connect. To upgrade a previously installed instance, the `--install.type` setting value must be set to 1.

Prior to beginning an upgrade, see the [Coverity 2020.12 Upgrade Guide](#) for important upgrade information and procedures.

Parameter	Description
<code>--db.backup=&lt;true   false&gt;</code>	Default value is set to <code>false</code> . If set to <code>true</code> , this option determines if the in-place upgrade process creates a backup of the existing embedded database.
<code>--db.backup.dir=&lt;directory-path&gt;</code>	Required. Specifies the directory where the backup database is stored. Valid if the <code>--db.backup</code> option is set to <code>true</code> .
<code>--db.backup.file= &lt;filename&gt;</code>	Required. File name of backup. Valid if the <code>--db.backup</code> option is set to <code>true</code> .
<code>--db.embedded.performance=&lt;0   1&gt;</code>	<p>Selects the Coverity Connect database performance tuning option. Choose values depending on your installation type.</p> <p>The following setting values are available:</p>

Parameter	Description
	<ul style="list-style-type: none"> <li>• 0 - Production</li> <li>• 1 - Restore</li> </ul> <p>Otherwise, the default value is 0.</p>
<code>--installation.dir=&lt;directory-path&gt;</code>	Location to which Coverity Platform is installed. This must be an absolute directory path.
<code>--license.path=&lt;path&gt;</code>	Required. Specifies the full directory path of a Coverity <code>license.dat</code> file.

#### Backup-and-restore upgrade parameters

These parameters specify the options used for completing a backup-and-restore upgrade of an installed instance of Coverity Connect. To backup and restore an upgrade, the `--install.type` setting value must be set at 2.

Parameter	Description
<code>--accept.https=&lt;true   false&gt;</code>	Specifies that Coverity Connect should use SSL (HTTPS) communication. The HTTPS protocol requires an installed server certificate from a certificate authority. Default value is taken from the previous Coverity Connect instance that is stored.
<code>--backup.dir=&lt;directory-path&gt;</code>	Specifies the directory where the backup data is stored. Required if the <code>--backup.automated</code> option is set to <code>false</code> .
<code>--db.type=&lt;0   1&gt;</code>	<p>Specifies the type of database that is used with the Coverity Connect application server. The following option values are available:</p> <ul style="list-style-type: none"> <li>• 0 - Embedded database parameters</li> </ul> <p>This includes the <code>--db.embedded.data</code> and <code>--db.embedded.port</code> options.</p> <ul style="list-style-type: none"> <li>• 1 - External database parameters</li> </ul> <p>This includes the following options: <code>--db.external.host</code>, <code>--db.external.password</code>, <code>--db.external.port</code>, <code>--db.external.name</code>, and <code>db.external.user</code>.</p>

Parameter	Description
	<p> <b>Note</b></p> <p>The external database must be created before you run the installer.</p> <p>For more information, see Section 1.3, “Advanced installation options”. Otherwise, the default value is 0. This specifies that Coverity Connect will connect to an existing, external PostgreSQL database.</p>
<pre>--backup.automated=&lt;true   false&gt;</pre>	<p>Initiates the automated backup and restore process if set to true. Default value is true. When set to false, the installer upgrades from a previously created backup.</p> <p>Note that when -- backup.automation=false, the -- backup.dir option must be specified.</p>
<pre>--commit.port=&lt;port&gt;</pre>	<p>Recommended. Specifies the Coverity Connect commit port. Default value remains the same from the previous Coverity Connect instance that is stored.</p>
<pre>--control.port=&lt;port&gt;</pre>	<p>Recommended. Specifies the port used by the embedded application server for internal server communication. Default value remains the same from the previous Coverity Connect instance that is stored.</p>
<pre>--db.embedded.performance=&lt;0   1&gt;</pre>	<p>Selects the Coverity Connect database performance tuning option. Choose values depending on your installation type.</p> <p>The following setting values are available:</p> <ul style="list-style-type: none"> <li>• 0 - Production</li> <li>• 1 - Restore</li> </ul> <p>Otherwise, the default value is 0.</p>
<pre>--existing.instance.dir=&lt;directory-path&gt;</pre>	<p>Required. The path of the existing Coverity Connect.</p>
<pre>--hostname=&lt;host-name&gt;</pre>	<p>Required. Specifies the hostname of the Coverity Connect server.</p>
<pre>--http.port=&lt;port&gt;</pre>	<p>Recommended. Specifies the HTTP port number for Coverity Connect. Default value remains</p>

## Installing Coverity Platform components

Parameter	Description
	the same from the previous Coverity Connect instance that is stored.
<code>--https.port=&lt;port&gt;</code>	Recommended. Specifies the HTTPS port number for Coverity Connect. Default value remains the same from the previous Coverity Connect instance that is stored. Valid if the <code>--accept.https</code> option is set to <code>true</code> .
<code>--installation.dir=&lt;directory-path&gt;</code>	Location to which Coverity Platform is installed. This must be an absolute directory path.
<code>--license.path=&lt;path&gt;</code>	Required. Specifies the full directory path of a Coverity <code>license.datfile</code> .
<code>--service.enable=&lt;true   false&gt;</code>	Specifies that Coverity Connect will be enabled as a Windows service. This parameter is optional and will not affect any installations on non-Windows platforms. Default value is <code>false</code> .

### Intermachine upgrade parameters

These parameters specify the options used for completing an intermachine upgrade of an installed instance of Coverity Connect. This is similar to the backup-and-restore upgrade, but the upgraded instance will be on a different machine than the original. To prepare your system for an intermachine upgrade, the `--install.type` setting should be set to 3.

Parameter	Description
<code>--accept.https=&lt;true   false&gt;</code>	Specifies that Coverity Connect should use SSL (HTTPS) communication. The HTTPS protocol requires an installed server certificate from a certificate authority. Default value is taken from the previous Coverity Connect instance that is stored.
<code>--backup.dir=&lt;directory-path&gt;</code>	Required. Specifies the directory where the backup data is stored.
<code>--commit.port=&lt;port&gt;</code>	Recommended. Specifies the Coverity Connect commit port. Default value remains the same from the previous Coverity Connect instance that is stored.
<code>--control.port=&lt;port&gt;</code>	Recommended. Specifies the port used by the embedded application server for internal server communication. Default value remains the same from the previous Coverity Connect instance that is stored.
<code>--db.backup.dir=&lt;directory-path&gt;</code>	Required. Determines if the in-place upgrade process creates a backup of the existing embedded database.

Parameter	Description
<code>--db.embedded.performance=&lt;0   1&gt;</code>	<p>Selects the Coverity Connect database performance tuning option. Choose values depending on your installation type.</p> <p>The following setting values are available:</p> <ul style="list-style-type: none"> <li>• 0 - Production</li> <li>• 1 - Demo</li> </ul> <p>Otherwise, the default value is 0.</p>
<code>--db.type=&lt;0   1&gt;</code>	<p>Specifies the type of database that is used with the Coverity Connect application server. The following option values are available:</p> <ul style="list-style-type: none"> <li>• 0 - Embedded database parameters</li> </ul> <p>This includes the <code>--db.embedded.data</code> and <code>--db.embedded.port</code> options.</p> <ul style="list-style-type: none"> <li>• 1 - External database parameters</li> </ul> <p>This includes the following options: <code>--db.external.host</code>, <code>--db.external.password</code>, <code>--db.external.port</code>, <code>--db.external.name</code>, and <code>db.external.user</code>.</p> <p> <b>Note</b></p> <p>The external database must be created before you run the installer.</p> <p>For more information, see Section 1.3, “Advanced installation options”.</p> <p>Otherwise, the default value is 0. This specifies that Coverity Connect will connect to an existing, external PostgreSQL database.</p>
<code>--hostname=&lt;host-name&gt;</code>	<p>Required. Specifies the hostname of the Coverity Connect server.</p>
<code>--http.port=&lt;port&gt;</code>	<p>Recommended. Specifies the HTTP port number for Coverity Connect. Default value remains the same from the previous Coverity Connect instance that is stored.</p>

## Installing Coverity Platform components

Parameter	Description
<code>--https.port=&lt;port&gt;</code>	Recommended. Specifies the HTTPS port number for Coverity Connect. Default value remains the same from the previous Coverity Connect instance that is stored. Valid if the <code>--accept.https</code> option is set to <code>true</code> .
<code>--license.path=&lt;path&gt;</code>	Required. Specifies the full directory path of a Coverity <code>license.datfile</code> .
<code>--service.enable=&lt;true   false&gt;</code>	Specifies that Coverity Connect will be enabled as a Windows service. This parameter is optional and will not affect any installations on non-Windows platforms. Default value is <code>false</code> .

### Upgrade preparation parameters

These parameters specify the options used for completing an upgrade preparation of an installed instance of Coverity Connect. This creates a backup for future use by the upgrade installer. To prepare your system for an upgrade, the `--install.type` setting should be set to 4.

Prior to beginning an upgrade, see the *Coverity 2020.12 Upgrade Guide* [🔗](#) for important upgrade information and procedures.

Parameter	Description
<code>--backup.dir=&lt;directory-path&gt;</code>	Required. Specifies the directory where you want to store the backup data when using the automated backup and restore process. You can also use this setting to retrieve backed up database information if you are upgrading (from a previously created backup).
<code>--existing.instance.dir=&lt;directory-path&gt;</code>	Required. The path of the existing Coverity Connect.

### Embedded database parameters

These options set the parameters for installing Coverity Connect with an embedded PostgreSQL database. The installer will automatically install and configure the database.

Parameter	Description
<code>--db.embedded.data=&lt;path&gt;</code>	Specifies the full path of the directory to which the database files will be installed. If you do not enter this option, the database directory defaults to <code>&lt;install_dir_cp&gt;/database</code> .
<code>--db.embedded.port=&lt;database_port_number&gt;</code>	Optional. Specifies the database port. Default value remains the same from the previous Coverity Connect instance that is stored. However, for a fresh installation, the default value is 5432.

### External database parameters

These options set the parameters for installing Coverity Connect with an external PostgreSQL database.

Parameter	Description
<code>--db.external.host=&lt;host_name&gt;</code>	Required. Specifies the hostname of the external PostgreSQL database.
<code>--db.external.password=&lt;database-password&gt;</code>	Required. Specifies the administrative password of the external PostgreSQL database.
<code>--db.external.port=&lt;port_number&gt;</code>	Required. Specifies the external PostgreSQL database port.
<code>--db.external.name=&lt;database-name&gt;</code>	Required. Specifies the name of the external PostgreSQL database.
<code>--db.external.user=&lt;admin-user-name&gt;</code>	Required. Selects the name of the administrative user that created the external PostgreSQL database.

## 1.3. Advanced installation options

The following installation options are available.

### 1.3.1. Using an external PostgreSQL database with Coverity Connect

Coverity Connect can use a PostgreSQL database as an external database.

If you use an existing, external PostgreSQL database you are responsible for the database, including implementing a database backup and recovery system.

#### **Caution**

This section is intended only for experienced PostgreSQL DBAs (database administrators).

#### **Configuring PostgreSQL for Coverity Connect**

You should be familiar with the following before performing this procedure.

#### **Prerequisites for external PostgreSQL configuration**

- A working installation of a supported PostgreSQL database: For supported PostgreSQL versions, see Section 7.2.1, “Coverity Connect software requirements”.
- The ability to create a database role and a database.
- A database back-up and recovery system. Coverity Connect does not back up external databases.

To configure an external PostgreSQL database for Coverity Connect:

1. Create a new database and a role that has permissions to create, modify, and drop tables for Coverity Connect. For example:

```
createdb -O <role name>...
```

The permissions allow the installer to create the database schema and to alter or drop tables when you upgrade Coverity Connect.

It is required that you create the database with the following encoding settings:

```
--encoding UTF8 --locale C
```

These settings might not be available on some existing PostgreSQL installations, depending on the version, the operating system, and the options used when initially creating the database cluster. For more information consult the PostgreSQL documentation.

2. Record the following information prior to configuring Coverity Connect:
  - PostgreSQL role name and password.
  - Database name.
  - PostgreSQL database server name.
  - Port of the database listener.
3. Using the Coverity Connect installer, specify *Connect to an existing PostgreSQL database*, and follow the prompts to complete the installation. The settings used by the installer for this configuration are described in Table 1.1.

**Table 1.1. External PostgreSQL Installer settings**

Name	Description	Default
PostgreSQL server name	Usually the host name where the PostgreSQL server runs.	None
Database port	The TCP port on which the server is listening for connections.	5432
Database name	Name of the database that you previously created for use by Coverity Connect.	None
Database user	User name that owns the database name previously specified.	None
Database password	Password for previously specified user.	None
Run as service	(Windows only) If you have Windows Administrator	Yes

Name	Description	Default
	privileges and want to run Coverity Connect as a service. The service runs as the built-in Windows account LocalService.	
Launch web browser	(Windows only) Opens a web browser on the URL to the Coverity Connect server.	Yes

4. Start Coverity Connect.
5. Sign in to Coverity Connect by entering the following URL in your web browser's location bar:

`http://hostname:<http_port>`

When installing Coverity Connect with an external PostgreSQL database, you are not asked to specify an administrator account password. Instead, sign in with user name `admin`, and password `coverity`. For security reasons, after the initial sign-in, change the password for this account.

In the event of difficulties connecting to an external database, you might want to examine and edit the property file (`cim.properties`) that controls the external database configuration. If you want to enable SSL on your external database connection, you must edit this property file. See Section 1.3.2, “Understanding database information in the `cim.properties` file”.

### 1.3.2. Understanding database information in the `cim.properties` file

To edit the `<install_dir_cc>/config/cim.properties` file:

1. Stop Coverity Connect:

Linux:

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

Windows:

Go to `<install_dir_cc>\bin` and double click the `cov-stop-im` program.

2. Check, and if necessary edit, the properties so that the following property values match:

- `embeddedDatabase=false`

This property is set to `true` when Coverity Connect uses an embedded database. However, after you have installed Coverity Connect with an external database, you cannot just change this value to `true` to change to an embedded database.

- `maindb.name=<database_name>`

- `maindb.user=<ROLE_name>`
- `maindb.url = jdbc\:postgresql\://<database_server_name>\:<port_number>/<database_name>`

This property value takes the form of a JDBC URL.

To enable SSL on the external database connection, append `?ssl=true` to this property value (as shown in the following example).

For example:

```
#Mon Sep 14 11:13:39 PDT 2009
embeddedDatabase=false
maindb.name=jdoe_main
db.driver=org.postgresql.Driver
db.dialect=org.hibernate.dialect.PostgreSQLDialect
maindb.password=afy687
maindb.user=jdoe
maindb.url=jdbc\:postgresql\://t-postgres-03\:5432/jdoe_main?ssl=true
dir.log=/usr/local/jdoe/CIM/server/base/logs
commitPort=9090
dir.temp=/usr/local/jdoe/CIM/server/base/temp
```

You must restart Coverity Connect after you add this attribute.

### 1.3.2.1. Changing the limit on active connections to the database pool

You can add this property to `cim.properties`.

`db.maxActiveConnections`

Specifies the maximum number of active connections allowed to the database pool. The default is 50.

Example: `db.maxActiveConnections=75`

### 1.3.3. Optimizing JVM utilization of additional RAM in `config/system.properties`

The application is started with the following JVM defaults.

For Production deployments (default):

`-Xms=512m`

`-Xmx=<75% of the Total System Memory>`

### 1.3.4. Changing the number of concurrent commits

These properties can be set in the `cim.properties` file.

### `commitPoolThreads`

Specifies the number of concurrent threads to process commits off of the queue. Minimum 1. Maximum 50. Default 5. To change this number, insert the following property:

```
commitPoolThreads=N (where N is the number of commits)
```

As commit performance and scale are affected by a number of factors (lines of code, number of defects/issues, complexity of events, etc.) it is difficult to predict the concurrency load on the application. While `commitPoolThreads` allows for greater levels of concurrency, it is possible to subject the application to too much load.

When this occurs, there are errors in the `$(CIM_HOME)/logs/cim.log` that are indicative of this issue. In some cases, these errors can be mitigated by increasing the heap memory allocation via the `-Xmx java_opts_post` setting in the `$(CIM_HOME)/config/system.properties` file with the provision that there is sufficient available system RAM. If not, the application load must be reduced and/or limited by reducing `commitPoolThreads` value.

The errors are as follows:

1. `java.lang.OutOfMemoryError: Java heap space` - suggests that the maximum heap size is insufficient for the JVM load.
2. `java.lang.OutOfMemoryError: GC overhead limit exceeded` - suggests that the application is spending a significant amount of time performing garbage collection instead of processing the application requests. This error can be disabled by appending the `-XX:-UseGCOverheadLimit` to the `java_opts_post` in the `$(CIM_HOME)/config/system.properties` file. This not recommended however as messages of this nature suggest extreme memory pressure on the application.

### `commitWorkQueueCapacity`

Specifies the number of commits that can wait in the queue. Minimum 15. Maximum 255. Default 80.

#### 1.3.4.1. Concurrent commit requirements

Concurrent commits require additional RAM. Given projects with the following characteristics:

- Files analyzed: ~3000
- Total lines of code input to `cov-analyze`: ~3 million
- Functions analyzed: ~90,000
- Paths analyzed: ~7.5 million
- Defect occurrences found: ~4,000

The amount of memory required is as follows:

- 2 concurrent commits - 8Gb
- 5 concurrent commits - 16Gb (default `commitPoolThreads` value)

- 8 concurrent commits - 24Gb

Ideally, you should have 1 CPU core per concurrent commit.

## 1.4. Post-installation notes

After installing Coverity Platform, you should review the sections in this chapter along with the information in Chapter 9, *Coverity Connect system tuning, maintenance, and monitoring*.

### 1.4.1. Important recommendations

After installing Coverity Connect, keep in mind these important recommendations:

- Do not move Coverity Connect from its installation location.
- Do not rename the Coverity Connect installation location.
- Do not start Coverity Connect from a different machine.
- Do not start Coverity Connect with a user other than the user who installed it.
- Do not use version numbers in the name of your installation directory. While using version numbers is permitted, it could cause problems when you upgrade to a new version. When you upgrade, the installation directory name is unmodified.

If you have to make these types of configuration changes, you should create a new installation, backup your old system, and restore the backup into the new installation instance. See the *Coverity 2020.12 Upgrade Guide* [🔗](#).

### 1.4.2. Getting Coverity Platform licensing information

Coverity Connect and Coverity Platform licensing information is provided in the *Help* menu, in the *About* option in Coverity Connect.

You can verify the type of license by clicking the *About* link in Coverity Connect and examining the *License Packaging* field. To request information about the restrictions of your license, or to inquire about upgrading your license, contact your Synopsys sales representative.

### 1.4.3. Installing multiple Coverity Connect licenses

You can update the Coverity Connect license by navigating to *Configuration > System > Coverity Connect License > Import the new file*.

The new license file must be named in the following format:

```
license<name>.dat
```



#### Note

We recommend that you add the new license file before an old license file has expired to ensure uninterrupted service when a license is renewed.

### 1.4.4. 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](#).

### 1.4.5. Changing the Coverity Connect owner

In Unix, when installing Coverity Connect the current user becomes the owner of all unpacked files, and is the only user who can execute command-line tools such as `cov-im-ctl`. The user name is also recorded as the `os_user` in the `config/system.properties` file.

If the owner's account is removed, then a new owner must be assigned. Perform the following two actions to change the Coverity Connect owner:

- At the command line, execute `chown -R new_owner_username[:new_group] <installation_dir>`.
- Edit the `config/system.properties` file and change `os_user` to the new user name.

This has no effect on the database user name.

## 1.5. Installing Coverity Reports

Coverity Reports include the Integrity Report, Security Report, and MISRA Report. They are installed separately from Coverity Connect. You can obtain installers from the Downloads page in Coverity Connect.

1. In Help → Downloads, select the Coverity Reports installer that is appropriate for your operating system.
2. Save the installer application to an appropriate folder on your system.

3. Launch the installer, and follow the instructions in the wizard.

## 1.6. Coverity Reports installation modes

You can complete the series of installation steps using either a graphical or console (command-line) interface. Alternatively, you can run the silent installer to complete the installation from the command line in a single step.

### 1.6.1. Selecting an installation mode

This procedure is covered for all the Coverity product installers in Section 1.2.1, “Selecting an installation mode”.

### 1.6.2. Coverity Reports silent installer

The Coverity Reports silent installer allows you to specify all of the installation configuration details on the command line so you do not need to run the "step-through" process either through the command line (`-c`) or the graphical (`-g`) installer modes.

To run the silent installer, specify the installation utility with the `-q` option, followed by the installation parameters. The `-q` option and the installation parameters must all be on the same command line. The following example installs Coverity Report version 2020.12 to the `home/cov-reports-linux64-2020.12` directory.

```
./cov-reports-linux64-2018.06.sh --installation.dir=~ /cov-reports-linux64-2018.06 -q
```

If you are installing on Windows, use the `-q -console` options preceded by a `start /wait` command. If the executable filename contains spaces, precede it with empty double quotes, even if the filename itself is double-quoted, for example:

```
> start /wait "" "<my executable name>" -q -console
```

#### Note

You can include the empty double quotes whether or not the executable name contains spaces.

The silent installer accepts the following options. Note that not all of the options are required. If you use any of the following parameters, you should provide specifically assigned values. Some values, if left blank, will accept the default value, but this is not a recommended practice.

Installer command line options

#### Note

Do not use the `-v` prefix with these options:

Option	Description
<code>-q</code>	Required. Enables the silent installer.

---

## Installing Coverity Platform components

---

<b>Option</b>	<b>Description</b>
-console	Required on Windows. Displays status messages in the console from which you invoked the silent installer.
--installation.dir= <directory-path>	Required. This option sets the location to which the Coverity Reports product is installed. The default value is <cwd>/Coverity/Coverity Reports/.

---

## Chapter 2. Installing Coverity Analysis components

### Table of Contents

2.1. Installing Coverity Analysis .....	28
2.2. Coverity Analysis installation modes .....	31
2.3. Coverity Analysis license options .....	36
2.4. Using an archive file to install Coverity Analysis .....	48

This part of the guide is for administrators who install Coverity Analysis.

Coverity Analysis provides tools that you use to analyze your code bases and programs. This document guides you through the steps of running the Coverity Analysis installer.

Some of the analysis components that you can install include:

- Coverity Analysis – for analysis of compiled and interpreted code bases, including code used for Web applications.
- Dynamic Analysis
- Architecture Analysis – see the Architecture Analysis installation instructions for further information about launching Architecture Analysis.
- Coverity Extend SDK
- Coverity Wizard – GUI-based configuration tool that is installed as part of Coverity Analysis.
- Coverity Desktop Analysis
- .NET Core SDKs
- Documentation:
  - English version
  - Japanese version
  - Korean version
  - Chinese version

 **Note**

If you are upgrading an existing version of Coverity Analysis, see *Coverity Upgrade Guide* .

Note that certain Coverity Analysis components are only available for certain platforms. So, depending on the platform on to which you are installing, you might only be able to access a subset of the analysis tools. For details, see Section 7.3, “Coverity Analysis”.

## 2.1. Installing Coverity Analysis

Prior to starting this procedure, you should have your license ready. For details, see Section 2.3, “Coverity Analysis license options”. Prior to installing to a production environment, you should also make sure that you understand the deployment considerations and hardware recommendations in Chapter 5, *Deployment planning*.

In general, you should use the installation procedure described in this chapter. For rare cases where it is not possible to do so, an alternative is described in Section 2.4, “Using an archive file to install Coverity Analysis”.

### To install Coverity Analysis:

1. Verify that your operating system, compiler versions, and the Coverity Analysis products that you want to install on your platform are supported.

Depending on the Coverity Analysis components you intend to use, you will need to check one or more of the following sections:

- Section 7.3, “Coverity Analysis”
- Section 7.4, “Coverity Test Advisor SCM and platform support”
- Section 7.3.3, “Supported platforms for Coverity Wizard”
- Section 7.5, “Architecture Analysis”

2. On the machine on which you want to install Coverity Analysis, download the installer file.
3. Run the installer script or program for your operating system.

The installer file name is similar to `cov-analysis-<platform>-<version>.sh` or `.exe`. For example, the Linux 64-bit installer file is named `cov-analysis-linux64-2020.12.sh`, and you might run it as follows:

```
> cov-analysis-linux64-2020.12.sh
```

For Windows systems, double-click the `.exe` program.

Depending on whether you are using Linux or Windows, the installer uses a text-based console mode or a graphical mode. The installation choices for graphical and console modes are identical.

For guidance with changing the installer mode, see Section 1.2, “Coverity Platform installation modes”.

4. Select and accept the End User Software License and Maintenance Agreement for your region of the world.
5. Choose a destination directory for the Coverity Analysis components.

In this document, the Coverity Analysis installation directory is referred to as `<install_dir_ca>`.

6. Choose the Coverity Analysis components that you want to install.

Select from the following options:

- Coverity Static and Dynamic Analysis

Note that this option cannot be deselected.

- Extend SDK
- .NET Core SDKs
- Documentation
  - English Documentation
  - Japanese Documentation
  - Korean Documentation
  - Chinese Documentation
- Coverity Wizard

7. Choose your license file type:

- *Coverity* - Choose this option if you are using a license file with a `.dat` extension, for example `license.dat`. To obtain or inquire about your license file from Coverity, send mail to `software-integrity-license@synopsys.com`.
- *FLEXnet* - Choose this option if you are using a FLEXnet license file.
- *Both* - Choose this option if you have FLEXnet licensing for Coverity Analysis and want to use Dynamic Analysis (Dynamic Analysis only uses Coverity licensing).
- *Obtain from server* - If you want the desktop analysis tool to automatically obtain a license file from the Coverity Connect server. In order to use this option the administrator must first install the analysis license file on the server. This is only valid when performing strictly Desktop Analysis, and should not be used for Central Analysis servers.

 **Note**

The Coverity Analysis installation process will not continue if you do not specify a license file or license server location.

8. For Coverity licensing: Specify the location of your Coverity License file.

If you chose the *Coverity* or *Both* option for the license type, the installer prompts you to specify the location of your license (`.dat`).

- 
9. For FLEXnet licensing: Set up your FLEXnet `license.config` file.

If you chose the *FLEXnet* or *Both* option for the license type, this step sets up the configuration file that tells Coverity Analysis where the FLEXnet license server is. Choose from the following options:

1. *Basic* - Choose this option if you use a single license server. You will be prompted to enter the license server hostname and license server port for your FLEXnet server.
2. *Advanced* - Choose this option if your license servers are a redundant triad. This option prompts you to enter a comma-separated list of three <port>@<hostname> values. For example:

```
28000@flex1,28001@flex2,28002@flex3
```

For more information about setting up FLEXnet licensing, see the *Coverity Analysis 2020.12 User and Administrator Guide* (located in the <install\_dir\_ca>/doc directory).

3. *Use an existing license.config file* - Choose this option if you have an existing FLEXnet configuration file. This option prompts you to specify the location of the configuration file.
10. On Windows, choose whether you want to create Start Menu folders and shortcuts.
11. Indicate whether to launch Coverity Wizard and the Coverity documentation from launching after the installation is complete.

Note that `cov-wizard` is a GUI-based tool that you can use to run analyses. For Coverity Analysis documentation, see *Coverity Wizard 2020.12 User Guide* and "Analyzing source code from the command line" in *Coverity Analysis 2020.12 User and Administrator Guide*.

12. After the installation finishes, add <install\_dir\_ca>/bin to your `PATH` environment variable.
13. If you intend to install Architecture Analysis see Chapter 4, *Installing Architecture Analysis*.
14. [Optional] Check your installation directory.

Access to Coverity Analysis components in the following directories depends on the scope of your license and the platform on which you are running Coverity Analysis (see Section 7.3, "Coverity Analysis").

- **Coverity Analysis**

Coverity Analysis tools and binaries are located at the top level installation directory. For example, the Coverity Analysis commands are located in:

```
<install_dir_ca>/bin
```

All of the Coverity Analysis product documentation is accessible from the following locations:

- <install\_dir\_ca>/doc/en/index.html (English)
- <install\_dir\_ca>/doc/ja/index.html (Japanese). Please see the notice regarding the availability of Japanese-language documentation in this file.

- **Dynamic Analysis**

If your platform and license supports Dynamic Analysis, the tools and binaries are located in the following directory:

```
<install_dir_ca>/dynamic-analysis
```

- For Coverity Analysis commands and the Coverity Wizard application (for example, `cov-wizard.exe` on Windows):

```
<install_dir_ca>/bin
```

Note that Coverity Wizard starts automatically after installation of Coverity Coverity Analysis. For information, see *Using Coverity Wizard to Get Started with Coverity Analysis* [🔗](#).

### To uninstall Coverity Analysis:

1. Go to `<install_dir_ca>`.
2. On Linux, run the `uninstall` script.

On Windows, run the `uninstall.exe` program, and follow the prompts.

The `uninstall` script does not remove files that contain user-supplied data. To remove user-supplied data, manually delete the installation directory after running the `uninstall` or `uninstall.exe` script.

## 2.2. Coverity Analysis installation modes

You can complete the series of installation steps using either a graphical or console (command-line) interface. Alternatively, you can run the silent installer to complete the installation from the command line in a single step.

### 2.2.1. Selecting an installation mode

This procedure is covered for both the Coverity Analysis and Coverity Platform installers in Section 1.2.1, “Selecting an installation mode”.

### 2.2.2. Coverity Analysis silent installer

The Coverity Analysis silent installer allows you to specify all of the installation configuration details on the command line so you do not need to run the “step-through” process either through the command line (`-c`) or the graphical (`-g`) installer modes.

To run the silent installer, specify the installation utility with the `-q` option, followed by the installation parameters. The `-q` option and the installation parameters must all be on the same command line. The following example installs Coverity Analysis version 2020.12 to the `home/cov-analysis-linux64-2020.12` directory.

```
./cov-analysis-linux64-2018.06.sh -q \
```

## Installing Coverity Analysis components

```
--installation.dir=cov-analysis-linux64-2018.06 \  
--license.region=0 \  
--license.agreement=agree \  
--license.type.choice=0 \  
--license.cov.path=/tmp/license.dat \  

```

If you are installing on Windows, use the `-q -console` options preceded by a `start /wait` command. If the executable filename contains spaces, precede it with empty double quotes, even if the filename itself is double-quoted, for example:

```
> start /wait "" "<my executable name>" -q -console
```

### Note

You can include the empty double quotes whether or not the executable name contains spaces.

The silent installer accepts the following options. Note that not all of the options are required. If you use any of the following parameters, you should provide specifically assigned values. Some values, if left blank, will accept the default value, but this is not a recommended practice. For more information about the installation options, see Chapter 2, *Installing Coverity Analysis components*.

### Installer command line options

#### Note

Do not use the `-v` prefix with these options:

Option	Description
<code>-q</code>	Required. Enables the silent installer.
<code>-console</code>	Required on Windows. Displays status messages in the console from which you invoked the silent installer.
<code>--installation.dir=&lt;directory-path&gt;</code>	Required. This option sets the location to which the Coverity Analysis product is installed. The default value is <code>&lt;cwd&gt;/cov-analysis-&lt;platform&gt;-&lt;version&gt;</code> .

### General parameters

Parameter	Description
<code>--desktop.link=&lt;true   false&gt;</code>	This option creates a desktop icon for Coverity Analysis. This option is only valid on Windows. Default value is <code>false</code> .
<code>--license.agreement =&lt;agree&gt;</code>	Required. Confirms agreement to the terms of the license.
<code>--license.region=&lt;0   1   2   3   4   5   6&gt;</code>	Required. Specifies the region of the End User License and Management Agreement (EULM). It used to select the correct product license. The

Parameter	Description
	<p>available values (and representative region) are as follows:</p> <ul style="list-style-type: none"> <li>• 0 - Americas, Africa, and Israel</li> <li>• 1 - Japan</li> <li>• 2 - Taiwan</li> <li>• 3 - China</li> <li>• 4 - Republic of Korea</li> <li>• 5 - International license (for any countries not mentioned in 0-4)</li> <li>• 6 - Evaluation Only. This installation is being used solely for evaluation purposes, and is not for production use</li> </ul>
<code>--license.type.choice=&lt;0   1   2   3&gt;</code>	<p>Specifies the type of license.</p> <ul style="list-style-type: none"> <li>• 0 - Coverity .dat license file.</li> <li>• 1 - FLEXnet config.file license file. "license.dat".</li> <li>• 2 - Both Coverity .dat and FLEXnet config.file license files.</li> <li>• 3 - Obtain license file from server (in Desktop mode)</li> </ul> <p>Otherwise, the default value is 0.</p>

Included component parameters

The following optional parameters specify which components will be included in the Coverity Analysis installation. If unspecified, only Coverity Analysis and Dynamic Analysis will be installed.

Parameter	Description
<code>--component.cov-wizard=&lt;true   false&gt;</code>	Installs the Coverity Wizard component. Default value is true.
<code>--component.dotnet-sdk=&lt;true   false&gt;</code>	Installs the .Net Core SDKs. This component contains selected .NET Core SDKs that can be used for buildless capture on C# .NET Core SDK projects. By installing this component, you might not need to install a .NET Core SDK in advance. You need this component only if you intend to

## Installing Coverity Analysis components

Parameter	Description
	use buildless capture to capture and analyze C# projects. This option is valid only for 64-bit Windows platforms. Default value is <i>false</i> .
<code>--component.sdk=&lt;true   false&gt;</code>	Installs the Coverity Extend SDK. This option is valid only for platforms that support Extend SDK. Default value is <i>false</i> .
<code>--component.skip.documentation=&lt;true   false&gt;</code>	Skips all documentation components. This option overrides all individual documentation component options. Default value is <i>false</i> .
<code>--component.en_doc=&lt;true   false&gt;</code>	Installs the English documentation component. This option is ignored if <code>--component.skip.documentation=true</code> . Default value is <i>true</i> .
<code>--component.ja_doc=&lt;true   false&gt;</code>	Installs the Japanese documentation component. This option is ignored if <code>--component.skip.documentation=true</code> . Default value is <i>true</i> .
<code>--component.ko_doc=&lt;true   false&gt;</code>	Installs the Korean documentation component. This option is ignored if <code>--component.skip.documentation=true</code> . Default value is <i>true</i> .
<code>--component.zh-cn_doc=&lt;true   false&gt;</code>	Installs the Chinese Simplified documentation component. This option is ignored if <code>--component.skip.documentation=true</code> . Default value is <i>true</i> .

### Coverity licensing parameters

The following parameters are optional configurations using the Coverity license. To use these options, the `--license.type.choice` option must be set to 0 or 2.

Parameter	Description
<code>--license.cov.path=&lt;path&gt;</code>	Required. Specifies the full directory path of a Coverity <code>license.dat</code> file. This option is valid if the <code>--license.type.choice</code> option is set to a 0 or 2.

### FLEXnet licensing parameters

The following parameters are optional configurations using FLEXnet licensing. To use these options, the `--license.type.choice` option must be set to 1 or 2.

Parameter	Description
<code>--license.flex.choice=&lt;0   1   2&gt;</code>	Specifies your FLEXnet license server type. You can set the value to one of the following:

Parameter	Description
	<ul style="list-style-type: none"> <li>• 0 - Basic, single license server</li> <li>• 1 - Advanced, redundant triad of license servers</li> <li>• 2 - Existing <code>license.config</code> file.</li> </ul> <p>Otherwise, the default value is 0.</p>

Basic FLEXnet licensing parameters

Basic FLEXnet licensing parameters (where the `--license.flex.choice` value is 0) have the following configurations available:

Parameter	Description
<code>--license.flex.basic.host=&lt;host-name&gt;</code>	Specifies the host name of your FLEXnet server. This option is required if you are using a basic FLEXnet server or if the <code>--license.flex.choice</code> option is set to 0.
<code>--license.flex.basic.port=&lt;port&gt;</code>	Specifies the port number of your FLEXnet server. This option is required if you are using a basic FLEXnet server or if the <code>--license.flex.choice</code> option is set to 0.

Advanced FLEXnet licensing parameters

Advanced FLEXnet licensing parameters (where the `--license.flex.choice` value is 1) have the following configurations available:

Parameter	Description
<code>--license.flex.advanced.triad=&lt;triad&gt;</code>	<p>A FLEXnet server triad is specified by a comma-separated list of three <i>port@host-name</i> values. For example, <code>28000@flex1,28001@flex2,28002@flex3</code>.</p> <p>This option is required if the <code>--license.flex.choice</code> option is set to 1.</p>

Existing FLEXnet licensing parameters

Existing FLEXnet licensing parameters (where the `--license.flex.choice` value is 2) have the following configurations available:

Parameter	Description
<code>--license.flex.config.path=&lt;path&gt;</code>	Specifies the full directory path to the <code>license.config</code> file. The <code>license.config</code> filename must be included at the end of the path.

Parameter	Description
	This option is required if the <code>--license.flex.choice</code> option is set to 2.

## 2.3. Coverity Analysis license options

Coverity Analysis licenses have options that enable or disable functionality in Coverity Analysis and its infrastructure. Any combination of the following software packages can be enabled:

- Quality analysis.
- Web application security analysis.
- Test Advisor analysis.
- Third Party Integration Toolkit usage.
- Dynamic Analysis analysis.

These options are provided through both of the Coverity Analysis licensing mechanisms:

- Coverity Analysis licenses: `license.dat` (available from `software-integrity-license@synopsys.com`).
- FLEXnet licenses: FLEXnet licenses provide floating-node features, including the counting of license usage against a centrally maintained limit. See Section 2.1, “Installing Coverity Analysis” [p. 29] for details.

### Note

Coverity Analysis is supported on virtual machines (VMs) only if you use FlexNet licensing or a default license that is not node-locked to a particular host machine.

Pre-6.5 licenses continue to be supported for backward compatibility. They do not have the options mentioned above. When Coverity Analysis 6.5 sees a pre-6.5 license, it infers that quality analysis is enabled and the other three options are disabled. Therefore, you don’t need a new license to run Coverity Analysis 6.5 unless you want to use the three new features of 6.5.

Note that it is possible to reset your license after installing Coverity Analysis. This procedure is necessary if you need to replace a trial license with a production license, if you need to update your license, or if your current license file is invalid for some other reason.

### 2.3.1. Setting up a license.dat file

**To set up default licensing:**

1. Obtain a license file from Coverity by sending an e-mail to `software-integrity-license@synopsys.com`.

2. Verify that the license file is named `license.dat`. If it is not, rename it to `license.dat`.
3. Copy the license file to the `<install_dir>/bin` directory.

 **Note**

A missing license leads to a fatal `No license found` error when you attempt to analyze your code.

On some Windows platforms, you might need to use administrative privileges when you copy the Coverity Analysis license to `<install_dir>/bin`. Due to file virtualization in some versions of Windows, it might look like `license.dat` is in `<install_dir>/bin` when it is not.

Typically, you can set the administrative permission through an option in the right-click menu of the executable for the command interpreter (for example, `Cmd.exe` or `Cygwin`) or Windows Explorer.

### 2.3.2. Installing multiple Coverity Analysis licenses

You can update the Coverity Analysis license by placing multiple license files into the `$COVERITY_STATIC_ANALYSIS\bin` folder.

The new license file must be named in the following format:

```
license<name>.dat
```

 **Note**

We recommend that you add the new license file before an old license file has expired to ensure uninterrupted service when a license is renewed.

### 2.3.3. Installing FLEXnet licenses

Coverity Analysis supports FLEXnet licensing. FLEXnet licensing is an alternative to the default licensing process described in Section 2.3, “Coverity Analysis license options”.

 **Note**

FlexLM software is now called FlexNet Publisher. For more information about FlexNet Publisher (including support contact information), visit the FlexNet Publisher Community page at:

[https://community.flexera.com/t5/FlexNet-Publisher/ct-p/FlexNet\\_Publisher](https://community.flexera.com/t5/FlexNet-Publisher/ct-p/FlexNet_Publisher) 

The following table lists how the Coverity Analysis commands map to FLEXnet licensing features. This information is helpful for troubleshooting purposes.

**Table 2.1. Coverity commands mapped to FLEXnet licensing features**

Coverity command	FLEXnet licensing feature
<code>cov-analyze</code>	<code>analysis.master</code> and <code>analysis.worker</code>

Coverity command	FLEXnet licensing feature
<code>cov-commit-defects</code>	<code>analysis.infrastructure</code>
<code>cov-format-errors</code>	<code>analysis.infrastructure</code>
<code>cov-make-library</code>	<code>analysis.master</code> and <code>analysis.worker</code>
<code>cov-configure</code>	<code>analysis.infrastructure</code>

There is one `analysis.master` for each analysis job.

There is one or more `analysis.worker`; one for each analysis worker. The number of parallel analysis workers for `cov-analyze` is specified in the `-j <number-of-workers>` option. There is exactly one worker each for `cov-make-library`.

To set up FLEXnet licensing for Coverity Analysis:

1. Verify that your platform is supported (see Chapter 7, *Supported platforms*).
2. Run the `generate-flexnet-hostid` command and send the results to `software-integrity-license@synopsys.com`.

Coverity will send you, attached to an e-mail, a license file named `coverity.lic`.

The `generate-flexnet-hostid` command generates the MAC addresses of all NICs that will be used for FLEXnet licensing.

 **Note**

You must have the Linux standard base `lsb` package installed on your machine in order to run `generate-flexnet-hostid`.

3. Rename the `sample-license.config` file to `license.config` and put it in the `<install_dir_sa>/bin` directory.

 **Important**

The `<install_dir_sa>/bin/license.config` file must exist.

If this file is empty, Coverity Analysis uses localhost for the default license server during the analysis and committing of defects.

4. Start the license server manager from the `<install_dir_sa>/bin` directory by running the following command:

```
> [nohup] lmgrd -c coverity.lic
```

The `lmgrd` command runs in the background. To run `lmgrd` in the foreground or to debug FLEXnet license server problems on Windows, use the `-z` option. Avoid the `-z` option on UNIX or Linux systems because the `lmgrd` daemon cannot be stopped with Ctrl-C.

To capture the output in a log file, use the `-l <log-file>` option.

 **Note**

On some UNIX or Linux systems, if you do not use the `nohup` command, it is not possible to log out of the license server from the shell where the `lmgrd` command ran. To avoid this, use the `nohup` command on UNIX or Linux systems.

5. Edit the `<install_dir_sa>/bin/license.config` file to add your license server configuration information using the following syntax:

```
license-server [<port1>]@<server1>[, [<port2>]@<server2> ... ]
```

 **Important**

The `license.config` file that is used for FlexNet licensing needs to end with a newline character - that is - end with a blank line. If it does not, `cov-analyze` will not recognize the last line of the file.

Example:

```
license-server @localhost
```

Example:

```
license-server 28000@flex1,28001@flex1,28000@flex2
```

For information about troubleshooting FLEXnet licensing, see [Troubleshooting for FLEXnet licensing](#).

For more information about the license server manager and FLEXnet licensing, see the License Administration Guide, FLEXnet Publishing Licensing Toolkit 11.5 .

### 2.3.4. Troubleshooting for FLEXnet licensing

You might encounter the following common issues with licensing of Coverity Analysis. For more information about the license server manager and FLEXnet licensing, see the License Administration Guide, FLEXnet Publishing Licensing Toolkit 11.5 .

 **Note**

FlexLM software is now called FlexNet Publisher. For more information about FlexNet Publisher (including support contact information), visit the FlexNet Publisher Community page at:

[https://community.flexera.com/t5/FlexNet-Publisher/ct-p/FlexNet\\_Publisher](https://community.flexera.com/t5/FlexNet-Publisher/ct-p/FlexNet_Publisher)

2.3.4.1. I get an <code>lmutil not found</code> error when I run <code>generate-flexnet-hostid</code> . .....	40
2.3.4.2. A long message displays when I run the <code>lmgrd</code> command. ....	40
2.3.4.3. The <code>lmgrd</code> command uses a different port number than the one in my <code>license.config</code> file. ....	41
2.3.4.4. The license server cannot verify my license. ....	41
2.3.4.5. The license manager cannot initialize. ....	41
2.3.4.6. The license file is different than the one that you expect. ....	42

2.3.4.7. The `lmutil lmdown` command cannot shut down the license server. .... 43

2.3.4.8. The `lmutil lmdiag` command returns a start date of 1-jan-1990. .... 43

2.3.4.9. The clock difference is too large between the client and server systems. .... 44

2.3.4.10. The behavior of the following `lmutil lmdown` command query can cause confusion: .... 44

2.3.4.11. The `.flexlmrc` file settings are not recognized. .... 44

2.3.4.12. The `/usr/tmp/.flexlm` file cannot be created. .... 44

2.3.4.13. The `lmgrd -z` option leaves `lmgrd` in the foreground on UNIX and Linux systems. .... 45

2.3.4.14. The `covlicd` command exits unexpectedly. .... 45

2.3.4.15. The `lmutil lmhostid --help` command does not display the `-ether` option. .... 46

2.3.4.16. The message (`covlicd`) `UNSUPPORTED:`  
`"prevent.ccpp" (PORT_AT_HOST_PLUS )` displays. .... 46

2.3.4.17. License failure for systems configured to use IPv6. .... 47

2.3.4.18. `cov-analyze` will not recognize the last line of the file. .... 47

2.3.4.19. On Linux environments, the FLEXnet licensing does not work with Ethernet devices that  
 have the LAN port mapped to anything other than `eth0`. .... 47

**2.3.4.1.** I get an `lmutil not found` error when I run `generate-flexnet-hostid`.

**Solution**

You must have the Linux standard base `lsb` package installed on your machine.

**2.3.4.2.** A long message displays when I run the `lmgrd` command.

**Solution**

The following FLEXnet marketing message displays when you run the `lmgrd` command. Real error messages often proceed or follow it. Make sure to scroll up to the top of your screen to read any error messages that display. For the sake of brevity, this content is removed from the remaining questions and solutions.

```
13:53:54 (lmgrd) -----
13:53:54 (lmgrd)   Please Note:
13:53:54 (lmgrd)
13:53:54 (lmgrd)   This log is intended for debug purposes only.
13:53:54 (lmgrd)   In order to capture accurate license
13:53:54 (lmgrd)   usage data into an organized repository,
13:53:54 (lmgrd)   please enable report logging. Use Macrovision's
13:53:54 (lmgrd)   software license administration solution,
13:53:54 (lmgrd)   FLEXnet Manager, to readily gain visibility
13:53:54 (lmgrd)   into license usage data and to create
13:53:54 (lmgrd)   insightful reports on critical information like
13:53:54 (lmgrd)   license availability and usage. FLEXnet Manager
13:53:54 (lmgrd)   can be fully automated to run these reports on
13:53:54 (lmgrd)   schedule and can be used to track license
13:53:54 (lmgrd)   servers and usage across a heterogeneous
13:53:54 (lmgrd)   network of servers including Windows NT, Linux
13:53:54 (lmgrd)   and UNIX. Contact Macrovision at
13:53:54 (lmgrd)   www.macrovision.com for more details on how to
13:53:54 (lmgrd)   obtain an evaluation copy of FLEXnet Manager
13:53:54 (lmgrd)   for your enterprise.
13:53:54 (lmgrd) -----
```

**2.3.4.3.** The `lmgrd` command uses a different port number than the one in my `license.config` file.

**Solution**

If you provide a port number in the `license.config` file, use the port number of `lmgrd`. In the following example, `lmgrd` uses port 27000 and `covlicd` (the Coverity vendor daemon) uses port 60185. Use the `lmgrd` port in the `license.config` file.

```
$ ./lmgrd -c coverity.lic
13:53:54 (lmgrd) -----
...
13:53:54 (lmgrd) -----
13:53:54 (lmgrd) FLEXnet Licensing (v11.5.0.0 build 56285 amd64_re3) started on
bl-1-4 (linux) (4/3/2008)
13:53:54 (lmgrd) Copyright (c) 1988-2007 Macrovision Europe Ltd. and/or
Macrovision Corporation. All Rights Reserved.
13:53:54 (lmgrd) US Patents 5,390,297 and 5,671,412.
13:53:54 (lmgrd) World Wide Web: http://www.macrovision.com
13:53:54 (lmgrd) License file(s): coverity.lic
13:53:54 (lmgrd) lmgrd tcp-port 27000
13:53:54 (lmgrd) Starting vendor daemons ...
13:53:54 (lmgrd) Started covlicd (internet tcp_port 60185 pid 32023)
13:53:54 (covlicd) FLEXnet Licensing version v11.5.0.0 build 56285 amd64_re3
13:53:54 (covlicd) Server started on bl-1-4 for: prevent.platform
13:53:54 (covlicd) prevent.dotnet prevent.java prevent.ccpp
13:53:54 (covlicd) EXTERNAL FILTERS are OFF
13:53:54 (lmgrd) covlicd using TCP-port 60185
```

**2.3.4.4.** The license server cannot verify my license.

**Solution**

The `license.config` file might be incorrectly configured or empty. Verify that the license server in the `<install_dir_sa>/bin/license.config` file correctly points to where the `lmgrd` command is running.

The DNS server might be down. If the DNS server is down, use an IP address in the `license.config` file instead of a hostname.

The license server might not be running. If it is not running, use the `lmgrd` command to start it.

The firewall on the license server might be blocking the `lmgrd` port. If so, make sure that the server where the `lmgrd` command is running does not block ports 27000 through 27009 for incoming TCP messages.

**2.3.4.5.** The license manager cannot initialize.

**Solution**

If you use the `lmgrd` command without the `-c` option, the command fails and displays an error message that includes the text `Cannot find license file` at the top of the output on the screen. Retry the `lmgrd` command using the `-c` option.

```
$ ./lmgrd coverity.lic
license manager: can't initialize: Cannot find license file.
The license files (or license server system network addresses) attempted are
```

---

## Installing Coverity Analysis components

---

```
listed below. Use LM_LICENSE_FILE to use a different license file,
or contact your software provider for a license file.
Filename:      /usr/local/flexlm/licenses/license.dat
License path:  /usr/local/flexlm/licenses/license.dat:
FLEXnet Licensing error:-1,359. System Error: 2 "No such file or directory"
For further information, refer to the FLEXnet Licensing documentation,
available at "www.macrovision.com".
14:04:49 (lmgrd) -----
...
14:04:49 (lmgrd) -----
14:04:49 (lmgrd) Using license file "/usr/local/flexlm/licenses/license.dat"
```

### 2.3.4.6. The license file is different than the one that you expect.

#### Solution

There are two reasons why the license file seems to be different than the one that you expect.

If the date on the license server is incorrect, the following type of error displays and likely causes confusion. Change the date on the license server to today's date.

```
21:04:27 (lmgrd) -----
...
21:04:27 (lmgrd) -----
21:04:27 (lmgrd) FLEXnet Licensing (v11.5.0.0 build 56285 i86_re3) started on
rh el3x86 (linux) (5/11/2007)
21:04:27 (lmgrd) Copyright (c) 1988-2007 Macrovision Europe Ltd. and/or
Macrovis ion Corporation. All Rights Reserved.
21:04:27 (lmgrd) US Patents 5,390,297 and 5,671,412.
21:04:27 (lmgrd) World Wide Web: http://www.macrovision.com
21:04:27 (lmgrd) License file(s): coverity.lic
21:04:27 (lmgrd) lmgrd tcp-port 27000
21:04:27 (lmgrd) Starting vendor daemons ...
21:04:27 (lmgrd) Started covlicd (internet tcp_port 35916 pid 5362)
21:04:27 (covlicd) FLEXnet Licensing version v11.5.0.0 build 56285 i86_re3
21:04:27 (covlicd) Feature prevent.platform is not enabled yet, starts on
12-may -2008
21:04:27 (covlicd) Feature prevent.ccpp is not enabled yet, starts on
12-may-200 8
21:04:27 (covlicd) License server system started on rhel3x86
21:04:27 (covlicd) No features to serve, exiting
21:04:27 (covlicd) EXITING DUE TO SIGNAL 36 Exit reason 4
21:04:27 (lmgrd) covlicd exited with status 36 (No features to serve)
21:04:27 (lmgrd) covlicd daemon found no features. Please correct
21:04:27 (lmgrd) license file and re-start daemons.
21:04:27 (lmgrd)
21:04:27 (lmgrd) This may be due to the fact that you are using
21:04:27 (lmgrd) a different license file from the one you expect.
21:04:27 (lmgrd) Check to make sure that:
21:04:27 (lmgrd) coverity.lic
21:04:27 (lmgrd) is the license file you want to use.
```

If you created the license file `coverity.lic` by copying and pasting from a Windows machine, the space character (ASCII 32) might be replaced with a 160 character (hex 0xA0). To resolve

this, replace all the 0xA0 characters in the license file with spaces. In this case, the following message displays:

```
$ ./lmgrd -c coverity.lic
9:58:10 (lmgrd)
9:58:10 (lmgrd) -----
...
9:58:10 (lmgrd) -----
9:58:10 (lmgrd)
9:58:10 (lmgrd)
9:58:10 (lmgrd) FLEXnet Licensing (v11.5.0.0 build 56285 amd64_re3) started on
bl-1-4 (linux) (4/23/2008)
9:58:10 (lmgrd) Copyright (c) 1988-2007 Macrovision Europe Ltd. and/or
Macrovision Corporation. All Rights Reserved.
9:58:10 (lmgrd) US Patents 5,390,297 and 5,671,412.
9:58:10 (lmgrd) World Wide Web: http://www.macrovision.com
9:58:10 (lmgrd) License file(s): coverity.lic
9:58:10 (lmgrd) lmgrd tcp-port 27000
9:58:10 (lmgrd) Starting vendor daemons ...
9:58:10 (lmgrd) Started covlicd (internet tcp_port 50476 pid 9281)
9:58:10 (covlicd) FLEXnet Licensing version v11.5.0.0 build 56285 amd64_re3
9:58:10 (covlicd) License server system started on bl-1-4
9:58:10 (covlicd) No features to serve, exiting
9:58:10 (covlicd) EXITING DUE TO SIGNAL 36 Exit reason 4
9:58:10 (lmgrd) covlicd exited with status 36 (No features to serve)
9:58:10 (lmgrd) covlicd daemon found no features. Please correct
9:58:10 (lmgrd) license file and re-start daemons.
9:58:10 (lmgrd)
9:58:10 (lmgrd) This may be due to the fact that you are using
9:58:10 (lmgrd) a different license file from the one you expect.
9:58:10 (lmgrd) Check to make sure that:
9:58:10 (lmgrd) coverity.lic
9:58:10 (lmgrd) is the license file you want to use.
```

**2.3.4.7.** The `lmutil lmdown` command cannot shut down the license server.

### Solution

The license server is not running. The following message displays:

```
$ lmutil lmdown
lmutil - Copyright (c) 1989-2007 Macrovision Europe Ltd. and/or Macrovision
Corporation. All Rights Reserved.
Shutdown failed: Cannot connect to license server system. (-15,570:115
"Operation now in progress")
```

**2.3.4.8.** The `lmutil lmdiag` command returns a start date of 1-jan-1990.

### Solution

This is bug OA-002429 from Acreso (FLEXnet). Although the start date is incorrect, the `lmutil lmdiag` command works as expected.

```
$ lmutil lmdiag
"commit" v2008.03, vendor: covlicd
```

```
License server: localhost
floating license  starts: 1-jan-1990,   expires: 26-mar-2008

This license can be checked out
```

**2.3.4.9.** The clock difference is too large between the client and server systems.

**Solution**

If the time difference between the client and server systems is larger than two days, the `cov-analyze`, `cov-commit-defects`, and `cov-format-errors` commands will not run because they cannot verify the license.

**2.3.4.10.** The behavior of the following `lmutil lmdown` command query can cause confusion:

```
Are you sure (y/n)?
```

**Solution**

Any letter after `y` or `Y` is ignored. If the first letter is not `y` or `Y`, the following message displays:

```
"No server selected, exiting"
```

**2.3.4.11.** The `.flexlmrc` file settings are not recognized.

**Solution**

Coverity command-line utilities do not use the `.flexlmrc` file. Store license settings in the `<install_dir>/bin/license.config` file.

**2.3.4.12.** The `/usr/tmp/.flexlm` file cannot be created.

**Solution**

To resolve this, run the license server from a supported platform and edit the `<install_dir_sa>/bin/license.config` file to point to the correct location of the license server. FLEXnet on Linux is supported on systems running Red Hat Enterprise Linux 3, Red Hat Enterprise Linux 4, or Red Hat Enterprise Linux 5.

```
23:41:25 (lmgrd) -----
...
23:41:25 (lmgrd) -----
23:41:25 (lmgrd)
23:41:25 (lmgrd) The license server manager (lmgrd) running as root:
23:41:25 (lmgrd)           This is a potential security problem
23:41:25 (lmgrd)           and is not recommended.
23:41:25 (lmgrd) Can't make directory /usr/tmp/.flexlm, errno: 2(No such file
or directory)
23:41:25 (lmgrd) Can't make directory /usr/tmp/.flexlm, errno: 2(No such file
or directory)
23:41:25 (lmgrd) Can't open /usr/tmp/.flexlm/lmgrdl.29777, errno: 2
license manager: can't initialize:
For further information, refer to the FLEXnet Licensing End User Guide,
available at "www.macrovision.com".
23:41:25 (lmgrd) Can't remove statfile /usr/tmp/.flexlm/lmgrdl.29777: errno No
such file or directory
```

**2.3.4.13.** The `lmgrd -z` option leaves `lmgrd` in the foreground on UNIX and Linux systems.

**Solution**

If you used the `lmgrd -z` option on UNIX or Linux systems, Ctrl-C does not terminate the license server. To terminate the license server, manually terminate all `lmgrd` and `covlicd` processes.

**2.3.4.14.** The `covlicd` command exits unexpectedly.

**Solution**

If the `lmgrd` and `covlicd` commands run on an unsupported platform, the following error message displays:

```
./lmgrd -c coverity.lic
18:20:04 (lmgrd) -----
...
18:20:04 (lmgrd) -----
18:20:04 (lmgrd)
18:20:04 (lmgrd)
18:20:04 (lmgrd) FLEXnet Licensing (v11.4.100.0 build 50818 i86_re3) started on
lee-linux (linux) (4/26/2008)
18:20:04 (lmgrd) Copyright (c) 1988-2007 Macrovision Europe Ltd. and/ or
Macrovision Corporation. All Rights Reserved.
18:20:04 (lmgrd) US Patents 5,390,297 and 5,671,412.
18:20:04 (lmgrd) World Wide Web: http://www.macrovision.com
18:20:04 (lmgrd) License file(s): coverity.lic
18:20:04 (lmgrd) lmgrd tcp-port 27000
18:20:04 (lmgrd) Starting vendor daemons ...
18:20:04 (covlicd) FLEXnet Licensing version v11.5.0.0 build 56285
i86_re3
18:20:04 (covlicd) lmgrd version 11.4, covlicd version 11.5
18:20:04 (lmgrd) Started covlicd (internet tcp_port 35481 pid 22149)
coverity@lee-linux:~/prevent-linux-4.0.0.beta1/bin$ 18:20:04
(covlicd) Server started on lee-linux for: prevent.platform
18:20:04 (covlicd) prevent.ccpp
18:20:04 (covlicd) EXTERNAL FILTERS are OFF
18:20:04 (lmgrd) covlicd using TCP-port 35481
18:20:04 (lmgrd) covlicd exited with status 0 signal = 17
18:20:04 (lmgrd) Since this is an unknown status, license server
18:20:04 (lmgrd) manager (lmgrd) will attempt to re-start the vendor daemon.
18:20:04 (covlicd) FLEXnet Licensing version v11.5.0.0 build 56285
i86_re3
18:20:04 (covlicd) lmgrd version 11.4, covlicd version 11.5
18:20:04 (lmgrd) REStarted covlicd (internet tcp_port 43652 pid 22155)
18:20:04 (covlicd) Server started on lee-linux for: prevent.platform
18:20:04 (covlicd) prevent.ccpp
18:20:04 (covlicd) EXTERNAL FILTERS are OFF
18:20:04 (lmgrd) covlicd using TCP-port 43652
18:20:04 (lmgrd) covlicd exited with status 0 signal = 17
18:20:04 (lmgrd) Since this is an unknown status, license server
18:20:04 (lmgrd) manager (lmgrd) will attempt to re-start the vendor daemon.
18:20:04 (covlicd) FLEXnet Licensing version v11.5.0.0 build 56285
i86_re3
```

## Installing Coverity Analysis components

---

```
18:20:04 (covlicd) lmgrd version 11.4, covlicd version 11.5
18:20:04 (lmgrd) REStarted covlicd (internet tcp_port 55809 pid 22161)
18:20:04 (covlicd) Server started on lee-linux for:      prevent.platform
18:20:04 (covlicd) prevent.cppp
18:20:04 (covlicd) EXTERNAL FILTERS are OFF
18:20:04 (lmgrd) covlicd using TCP-port 55809
18:20:04 (lmgrd) covlicd exited with status 0 signal = 17
18:20:04 (lmgrd) Since this is an unknown status, license server
18:20:04 (lmgrd) manager (lmgrd) will attempt to re-start the vendor daemon.
18:20:04 (covlicd) FLEXnet Licensing version v11.5.0.0 build 56285
i86_re3
18:20:04 (covlicd) lmgrd version 11.4, covlicd version 11.5
18:20:04 (covlicd) Cannot open lock file. errno=11 (/var/tmp/
lockcovlicd): Resource temporarily unavailable
18:20:04 (covlicd) EXITING DUE TO SIGNAL 41 Exit reason 9
18:20:04 (lmgrd) REStarted covlicd (internet tcp_port 56164 pid 22167)
18:20:04 (lmgrd) covlicd exited with status 41 (Exited because another server
was running)
18:20:04 (lmgrd) MULTIPLE "covlicd" license server systems running.
18:20:04 (lmgrd) Please kill, and run lmreread
18:20:04 (lmgrd)
18:20:04 (lmgrd) This error probably results from either:
18:20:04 (lmgrd)  1. Another copy of the license server manager
(lmgrd) is running.
18:20:04 (lmgrd)  2. A prior license server manager (lmgrd) was
killed with "kill -9"
18:20:04 (lmgrd)      (which would leave the vendor daemon running).
18:20:04 (lmgrd) To correct this, do a "ps -ax | grep covlicd"
18:20:04 (lmgrd)   (or equivalent "ps" command)
18:20:04 (lmgrd) and kill the "covlicd" process.
```

Make sure that the platform, upon which the `lmgrd` and `covlicd` commands run, is supported by Coverity.

**2.3.4.15.** The `lmutil lmhostid --help` command does not display the `-ether` option.

### Solution

The `lmutil lmhostid -ether` command generates the MAC addresses of all NICs that are used for FLEXnet licensing.

For more information about the `lmutil lmhostid -ether` option, see the `lmhostid` syntax in the License Administration Guide, FLEXnet Publishing Licensing Toolkit 11.5 [🔗](#). The *Ethernet address* term in the FLEXnet documentation is incorrect. An accurate term is *MAC address*.

**2.3.4.16.** The message `(covlicd) UNSUPPORTED: "prevent.cppp" (PORT_AT_HOST_PLUS )` displays.

### Solution

The following message might display if a specified feature is not listed in the license file. The message is benign; Coverity Analysis will attempt to use the "prevent.analysis" feature instead. If neither feature is present in the FLEXnet license file, Coverity Analysis will not run.

```
(covlicd) UNSUPPORTED: "prevent.cppp" (PORT_AT_HOST_PLUS )
```

```
<email_address>@<domain>.com  
(License server system does not support this feature. (-18,327))
```

#### 2.3.4.17. License failure for systems configured to use IPv6.

##### Solution

Systems configured to use IPv6, which is now standard for some Linux distributions, can have problems using FlexNet licensing with either Coverity Analysis for C/C++ or Coverity Analysis for Java. If the server pointed to in the `license.config` file is `@localhost` you might get one of these errors:

*For Coverity Analysis for C:*

```
cov-analyze --dir 'data'  
[FATAL] Licensing failure.  
[FLEXlm] License server machine is down or not responding.  
[FLEXlm] See the system administrator about starting the license server system,  
or  
[FLEXlm] make sure you're referring to the right host (see LM_LICENSE_FILE).  
[FLEXlm] Feature: prevent.analysis  
[FLEXlm] Hostname: localhost  
[FLEXlm] License path: 27000@localhost:
```

*For Coverity Analysis for Java:*

```
cov-analyze --max-mem 1024 --dir data junit-4.1.jar --findsource src  
  
Coverity Static Analyzer for Java  
Version 5.0.0 (pj5.0dev-push-2255)  
using Java 1.6.0_07 (Sun Microsystems Inc.)  
  
License check failed.  
Could not get FLEXnet License  
[ERROR] Could not verify the license.  
FlexlmException: Can't Connect to License Server (-15,3002) (Connection  
refused)
```

If you get one of the preceding error messages, check that the license server is running with `lmutil lmdiag -c <license>`. If it is running, try changing the `license.config` file to use `@127.0.0.1` instead of `@localhost`. Next, rerun `cov-analyze`.

#### 2.3.4.18. `cov-analyze` will not recognize the last line of the file.

##### Solution

The `license.config` file that is used for FlexNet licensing needs to end with a newline character - that is - end with a blank line. If it does not, `cov-analyze` will not recognize the last line of the file. Leave a blank line at the end of the `license.config` file.

#### 2.3.4.19. On Linux environments, the FLEXnet licensing does not work with Ethernet devices that have the LAN port mapped to anything other than **eth0**.

##### Solution

The FLEXnet licensing manager (FLEXlm), expects to get the MAC address only from the **eth0** device. If the licensing machine LAN port is mapped to **eth1** or higher, it will not get the correct MAC address. The solution is to map the LAN to **eth0**.

## 2.4. Using an archive file to install Coverity Analysis

Coverity recommends using the executable installers (`.sh`, `.exe` files described in Section 2.1, “Installing Coverity Analysis”) instead of the archive installers (`.tar.gz` and `.zip` files, for example, `cov-analysis-aix-2020.12.tar.gz`) because the executables set up user preferences that you will need, while the archive files do not. Archive installers are provided only for the very rare cases in which there is a problem with the executables. If you need a command line mechanism for automating the installation when using executables, see Section 2.2.2, “Coverity Analysis silent installer”.

### To install Coverity Analysis components using an archive file:

1. Verify that your operating system and compiler versions are supported.

For details, see Section 7.3, “Coverity Analysis”.

2. Verify that the archive installer has the MD5 checksum described in the Coverity Customer Center [🔗](#) (requires login).

You need to use the `md5sum` utility to calculate the MD5 hash.

3. On the machine where you intend to perform builds and analyses, decompress the contents of the archive into an installation directory that is *not* the root directory and that *does not* have a space character (" ") in the directory name.

This directory is your `<install_dir>`.

4. Set up licensing for Coverity Analysis.

For details, see Section 2.3, “Coverity Analysis license options”.

5. Add the `<install_dir>/bin/` directory to your PATH environment variable.

Once your PATH is set correctly, running command such as `cov-help --help` should display the help page for the command.

---

## Chapter 3. Installing Coverity Desktop components

### Table of Contents

3.1. Installing Coverity Desktop for Eclipse, Wind River Workbench, QNX Momentics, and IBM RTC .....	49
3.2. Installing Coverity Desktop for Microsoft Visual Studio .....	54
3.3. Installing Coverity Desktop for IntelliJ IDEA and Android Studio .....	55
3.4. Installing Coverity Analysis for local analysis .....	56

This part is for administrators who install Coverity Desktop for Eclipse, Microsoft Visual Studio, and IntelliJ IDEA.

### 3.1. Installing Coverity Desktop for Eclipse, Wind River Workbench, QNX Momentics, and IBM RTC

The Coverity Desktop product is available through the Coverity Connect *Downloads* page. From this page, you can:

- Obtain an update site location for installation or upgrade.
- Download the Coverity Desktop product packages and install them from your local desktop.
- Download Static Analysis and the corresponding license file for local analysis, if available.

To access the download page, sign in to Coverity Connect and click *Downloads* from the help menu. If you have difficulty signing in and cannot access the *Downloads* page, it might be because you do not have the appropriate role permissions or user credentials. If this occurs, contact your system administrator.

Installation software prerequisites (see IDE and Java Version Support for supported version numbers):

- Wind River WorkBench, QNX Momentics, IBM Rational® Team Concert™, or Eclipse
- Java Runtime Environment
- Eclipse C/C++ Development Tooling (CDT), only for the C/C++ analysis
- Eclipse Java Development Tools (JDT), only for the Java analysis

#### 3.1.1. Installing Coverity Desktop with the update site

Through Coverity Connect you can set up an update site to install Coverity Desktop or upgrade Coverity Desktop when the update site is updated with the new version.

1. Under the Coverity Connect Help menu, select *Downloads*.
2. Use the pull-down menu to select your IDE.

3. Copy the URL associated with your IDE (Eclipse, QNX Momentics, Wind River, IBM RTC), or click the copy link icon.
4. In the IDE, go to Help → Install New Software...

 **Note**

For Wind River Workbench only, you must go to the *Device Debug* perspective to make sure that the *Install New Software...* is present in the Help menu.

Additionally, do not use the Help → Install into Eclipse... option. This option does not install the Coverity plug-in.

5. Click the **Add...** button.
6. In the *Location/Work with* field, enter the update site URL and click **OK**.
7. Select Coverity Desktop C/C++ Analysis and Java Analysis (Eclipse only).
  - Coverity Desktop Java Analysis and C/C++ Analysis each allow you to view and triage Coverity issues in your project through central analysis.
  - Coverity Desktop Java Analysis allows you to run local analysis for Java Quality Defects and Security Risks.
  - Coverity Desktop C/C++ Analysis allows you to run local analysis for C/C++ Quality Defects.
8. Click **Next** and review the Coverity Product License Agreement.

 **Note**

The installation step, "Calculating requirements and dependencies," can take a long time if there are invalid or slow URLs in the *Available Software Sites* list. To solve this, remove the offending URLs or uncheck the box labeled *Contact all update sites during install to find required software*.

9. Select the *I accept the terms of the license agreement* radio button.
10. Click **Finish**.

 **Note**

The installation process might take a long time to complete because Eclipse checks for updates for all of the selected update sites installed on your environment. To speed up the installation, you can go to Help → Software Updates → Available Software → Manage Sites. Uncheck all of the update sites except for `cov-desktop-eclipse-2020.12`. When you want to update other components, go to Manage Sites and select them again.

11. Click **Restart Now** to restart the IDE.
12. Set up your workspace preferences.

For information about configuring Coverity Desktop, see the Coverity Desktop online help. To access the help, select Help → Coverity Help Center, then open *Coverity 2020.12 for Eclipse, Wind River Workbench, and QNX Momentics: User Guide*.

### 3.1.2. Installing Coverity Desktop from your local desktop

1. Under the Coverity Connect Help menu, select *Downloads*.
2. Use the pull-down menu to select your IDE.
3. Download and extract your preferred package:

For Eclipse

`cov-desktop-eclipse-2020.12.zip`

For Workbench

`cov-desktop-windriver-2020.12.zip`

For QNX Momentics

`cov-desktop-qnx-2020.12.zip`

For IBM RTC

`cov-desktop-ibmrtc-2020.12.zip`

4. From the IDE, select Help → Install New Software.

 **Note**

For Wind River Workbench, you must go to the *Device Debug* perspective to make sure that the *Install New Software...* is present in the Help menu.

Additionally, do not use the Help → Install into Eclipse... option. This option does not install the Coverity plug-in.

5. Click the **Add...** button.
6. Click **Local** and browse to your Coverity Desktop package directory:
  - `<CD_package_dir>/cov-desktop-eclipse-2020.12`
  - `<CD_package_dir>/cov-desktop-windriver-2020.12`
  - `<CD_package_dir>/cov-desktop-qnx-2020.12`
  - `<CD_package_dir>/cov-desktop-ibmrtc-2020.12`
7. Click **OK** for the Browse For Folder dialog.
8. Click **OK** for the Add Site dialog. In Eclipse 3.6, click **OK** for the Add Repository dialog.
9. Select Coverity DesktopC/C++ Analysis and Java Analysis (Eclipse only).

- Coverity Desktop Java Analysis and C/C++ Analysis each allow you to view and triage Coverity issues in your project through central analysis.
  - Coverity Desktop Java Analysis allows you to run local analysis for Java Quality Defects and Security Risks.
  - Coverity Desktop C/C++ Analysis allows you to run local analysis for C/C++ Quality Defects.
10. Click **Next** and review the Coverity Product License Agreement.
  11. Select the *I accept the terms of the license agreement* radio button.
  12. Click **Finish**.

 **Note**

The installation process might take a long time to complete because Eclipse checks for updates for all of the selected update sites installed on your environment. To speed up the installation, you can go to Help → Software Updates → Available Software → Manage Sites. Uncheck all of the update sites except for `cov-desktop-eclipse-*`. When you want to update other components, go to Manage Sites and select them again.

13. Click **Restart Now** to restart the IDE.
14. Set up your workspace preferences.

For information about configuring Coverity Desktop, see the Coverity Desktop online help, or *Coverity 2020.12 for Eclipse, Wind River Workbench, and QNX Momentics: User Guide* [↗](#).

To access the help, select Help → Help Topics. In the Contents pane, select *Coverity Desktop for use with Eclipse*.

### 3.1.3. Auto-configuring compilers for the QNX and WindRiver toolchains

For the QNX and WindRiver toolchains, it is convenient to automatically configure the compilers by using `coverity.conf`.

In `coverity.conf`, you can add a `add_compiler_configurations` element to the `settings` section.

For more information, see: *Coverity Desktop Analysis 2020.12: User Guide* [↗](#)

For QNX:

```
"add_compiler_configurations": [  
  { "cov_configure_args": ["--template", "--compiler", "qcc", "--comptype", "qnxcc"] }  
]
```

For WindRiver:

```
"add_compiler_configurations": [
  { "cov_configure_args": ["--template", "--compiler", "cmmips", "--comptype",
"gcc"]},
  { "cov_configure_args": ["--template", "--compiler", "ccpentium", "--comptype",
"gcc"]},
  { "cov_configure_args": ["--template", "--compiler", "ccppc", "--comptype", "gcc"]},
  { "cov_configure_args": ["--template", "--compiler", "c++arm", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "cpparm", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "g++arm", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "c++mips", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "cppmips", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "g++mips", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "c++pentium", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "cpppentium", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "g++pentium", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "c++ppc", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "cppppc", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "g++ppc", "--comptype", "g+
+" ]},
  { "cov_configure_args": ["--template", "--compiler", "dcc", "--comptype", "dcc"]}
]
```



### Note

If the user has a `data-coverity` folder in their workspace, the `"add_compiler_configurations"` setting will not take effect. To correct this issue, delete the `data-coverity` folder.

### 3.1.4. Uninstalling Coverity Desktop

This section describes the steps to remove Coverity Desktop from your IDE.

In Eclipse, Desktop C/C++ Analysis and Desktop Java Analysis are displayed as separate components in the installed components window. If you want to re-install or upgrade one of them, you must uninstall them both.

#### To remove Coverity Desktop:

1. In Eclipse, go to Help → About Eclipse.

In WorkBench, go to Help → About Wind River Workbench.

In QNX, go to Help → About QNX Momentics.

2. Click **Installation Details**.
3. Select *Coverity Desktop Analysis* (for C/C++, Java (for Eclipse), or both).
4. Click **Uninstall**.
5. Restart the IDE.

## 3.2. Installing Coverity Desktop for Microsoft Visual Studio

The Coverity Desktop plug-in is available through the Coverity Connect *Downloads* page. From this page, you can download the Coverity Desktop product package and install them from your local desktop or use the update URL through the Visual Studio Gallery feature.

To access the download page, sign in to Coverity Connect and click *Downloads* from the help menu. If you have difficulty signing in and cannot access the *Downloads* page, it might be because you do not have the appropriate role permissions or user credentials. If this occurs, contact your system administrator.

Installation software prerequisites (see IDE and Java Version Support for supported version numbers):

- Visual Studio
- You must have an instance of Coverity Connect to connect to in order to configure Coverity Desktop to run analyses and commit results.

### 3.2.1. Installing and updating Coverity Desktop for Visual Studio using a gallery

You can use a private gallery in Visual Studio (by providing its location) to install Coverity Desktop. This allows automatic updates for the Coverity Desktop plug-in from within Visual Studio, whenever a new version becomes available on the Coverity Connect server.

To use this feature, complete the following steps:

1. From the Coverity Connect Help menu, select *Downloads*.
2. Select Visual Studio from the *IDE* drop-down and copy the generated Gallery link.
3. In Visual Studio, navigate to Tools → Options → Environment → Extensions and Updates.
4. Click the **Add** button, enter a name for the extension, and paste the copied Gallery link in the *URL* field. Then click **Apply** to save.
5. Click **OK** to close the options window.
6. Navigate to Tools → Extensions and Updates → Online and select the name you chose for the Gallery.

7. Click **Download** to download the latest version of the plug-in.

For information about adding a private gallery to *Extensions and Updates* in Visual Studio, see <http://msdn.microsoft.com/en-us/library/hh266746.aspx> .

For more information about managing a private gallery by using registry settings, see <http://msdn.microsoft.com/en-us/library/hh266735.aspx> .

 **Note**

You can also install the Visual Studio plug-in from your local desktop by downloading and running the `Coverity.Desktop.vsix` file from the Coverity Connect Help → Downloads menu.

### 3.3. Installing Coverity Desktop for IntelliJ IDEA and Android Studio

The Coverity Desktop plug-in is available through the Coverity Connect *Downloads* page. From this page, you can download the Coverity Desktop product package and install it from your local desktop.

To access the download page, sign in to Coverity Connect and click *Downloads* from the help menu. If you have difficulty signing in and cannot access the *Downloads* page, it might be because you do not have the appropriate role permissions or user credentials. If this occurs, contact your system administrator.

Installation software prerequisites (see IDE and Java Version Support for supported version numbers):

- IntelliJ IDEA or Android Studio
- You must have an instance of Coverity Connect to connect to in order to configure Coverity Desktop to run analyses and commit results.

#### 3.3.1. Installing Coverity Desktop from your local desktop

1. From the Coverity Connect Help menu, select *Downloads*.
2. Select your IDE (IntelliJ or Android Studio) from the *IDE* pull-down menu.
3. Copy the displayed *Plug-in repository* URL.
4. In the IDE, open the *Settings* dialog (*Preferences* on Mac) and navigate to the *Plugins* page.
5. Click the *Browse repositories* button.
6. Select *Manage repositories* under the repositories list.
7. Click the *Add (+)* button to add a plug-in repository.
8. Enter the repository URL (copied from Coverity Connect) into the *Repository URL* field, and click *OK*.
9. Click *OK* to get back to the *Browse Repositories* dialog.
10. Right-click on "cov-desktop-\*" from the list of repositories, and select *Download and Install*.

11. Click *Yes* to download and install the Coverity Desktop plug-in.
12. Click *Close* then *OK* to close out of the *Settings* dialog.
13. Select *Restart* when prompted.



**Note**

You may encounter issues connecting to Coverity Connect when using an SSL connection. If you experience any connection issues when installing the Coverity Desktop plug-in, you can instead use the following steps to install:

1. From the Coverity Connect Help menu, select *Downloads*.
2. Select your IDE (IntelliJ or Android Studio) from the *IDE* pull-down menu.
3. Click the link to "download the plug-in".
4. In the IDE, open the *Settings* dialog (*Preferences* on Mac) and navigate to the *Plugins* page.
5. Click "Install plugin from disk".
6. Navigate to the downloaded `cov-desktop-intellij-2020.12.zip` file and click 'OK'.
7. Click 'OK' to close out of the *Settings* screen, and restart when prompted.

### 3.4. Installing Coverity Analysis for local analysis

Coverity Analysis is used to run a local analysis. If the Coverity Connect system administrator has configured the *Downloads* page to include the Coverity Analysis installer, you can download it (and optionally the `license.dat` license file) to your system. (For information on how to configure the *Downloads* page to include the Coverity Analysis installer, refer to the section "Adding Coverity Analysis to the Downloads page" in the Coverity Platform User and Administrator Guide.)

To download Coverity Analysis from the Coverity Connect *Downloads* page:

1. In the *Downloads* page, download the Coverity Analysis package (if available).
2. If available, download the license file.
3. Run the Coverity Analysis installer.

It is required that you provide the location of the `license.dat` file during installation.



**Note**

In the case where Coverity Analysis uses a FLEXnet license, you will find a `license.config` file in `<SA_install_dir>/bin`.

4. When you configure Coverity Desktop for local analysis, you can point the configuration to your local Coverity Analysis instance.

---

## Chapter 4. Installing Architecture Analysis

Architecture Analysis is installed separately from Coverity Analysis.

### To install Architecture Analysis:

1. Download the Architecture Analysis installer.
2. Run the installer to install Architecture Analysis.
3. For your operating system, locate one of the following files:

**Table 4.1. Architecture Analysis executable files**

Host OS	File	Description
Windows	cov-aa-build-check-cva-all-<version>.zip	Contains Windows executable: Build-check the CVA Developer Desktop Application.  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.
	cov-aa-build-check-dotnet-win64-<version>.exe	Contains Windows (64-bit) executable: Build-check the .Net Developer Desktop Application.  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.
	cov-aa-build-eclipse-feature-<version>.zip	Contains C/C++ CVA-based Eclipse Architecture Analysis plugin, which is used to show Architecture Analysis information in an Eclipse IDE for developers.  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.
	cov-aa-build-check-java-all-<version>.zip	Contains Windows (64-bit) executable: Build-check the Java Developer Desktop Application.  <b>Deprecation Notice:</b> Support for this tool is deprecated as of

## Installing Architecture Analysis

Host OS	File	Description
		2019.12 and will be removed in a future release.
	cov-aa-build-cva-all- <version>.zip	Architecture Analysis Build CVA.
	cov-aa-build-dotnet- win64-<version>.zip	Windows (64-bit): Architecture Analysis Build .Net.
	cov-aa-build-java-all- <version>.zip	Architecture Analysis Build Java.
	cov-aa-cva-win64- <version>.exe	Executable for the C/C++ version of Architecture Analysis for Windows (64-bit).
	cov-aa-dotnet-win64- <version>.exe	Executable for Architecture Analysis for Windows Visual Studio (64-bit).
	cov-aa-java-intellij15- plugin-<version>.zip	Architecture Analysis plug-in for the IntelliJ IDE version 15. <sup>a</sup>  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.
	cov-aa-java-win64- <version>.exe	Executable for the Java version of Architecture Analysis for Windows (64-bit).
	cov-aa-server- <version>.zip	Web application to run Architecture Analysis as a service.
Linux/Unix	cov-aa-build-check-cva- all-<version>.zip	For Architecture Analysis on Unix: Build-check CVA Developer Desktop Application.  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.
	cov-aa-build-check-java- all-<version>.zip	For Architecture Analysis on Unix: Build-check Java Developer Desktop Application.  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.

## Installing Architecture Analysis

Host OS	File	Description
	cov-aa-build-cva-all- <version>.zip	Architecture Analysis Build CVA.
	cov-aa-build-eclipse- feature-<version>.zip	Contains C/C++ CVA-based Eclipse Architecture Analysis plugin, which is used to show Architecture Analysis information in an Eclipse IDE for developers.  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.
	cov-aa-build-java-all- <version>.zip	Architecture Analysis Build Java.
	cov-aa-cva-unix- <version>.tar.gz	Architecture Analysis CVA for Unix.
	cov-aa-java-intellij15- plugin-<version>.zip	Architecture Analysis plug-in for the IntelliJ IDE version 15. <sup>a</sup>  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.
	cov-aa-java-unix- <version>.tar.gz	Java version of Architecture Analysis for Unix.
	cov-aa-server- <version>.zip	Web application to run Architecture Analysis.
macOS	cov-aa-build-check-cva- all-<version>.zip	Includes macOS executable for Architecture Analysis: Build-check CVA Developer Desktop Application.  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.
	cov-aa-build-check-java- all-<version>.zip	Includes macOS executable for Architecture Analysis: Build-check Java Developer Desktop Application.  <b>Deprecation Notice:</b> Support for this tool is deprecated as of

Host OS	File	Description
		2019.12 and will be removed in a future release.
	cov-aa-build-cva-all- <version>.zip	Architecture Analysis Build CVA.
	cov-aa-build-eclipse- feature-<version>.zip	Contains C/C++ CVA-based Eclipse Architecture Analysis plugin, which is used to show Architecture Analysis information in an Eclipse IDE for developers.  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.
	cov-aa-build-java-all- <version>.zip	Architecture Analysis Build Java.
	cov-aa-cva-macos- <version>.dmg	Executable for the C/C++ version of Architecture Analysis for macOS.
	cov-aa-java-intellij15- plugin-<version>.zip	Architecture Analysis plug-in for the IntelliJ IDE version 15. <sup>a</sup>  <b>Deprecation Notice:</b> Support for this tool is deprecated as of 2019.12 and will be removed in a future release.
	cov-aa-java-macos- <version>.dmg	Executable for the Java version of Architecture Analysis for macOS.
	cov-aa-server- <version>.zip	Web application to run Architecture Analysis.

<sup>a</sup>See the official documentation for your IDE for information about installing plug-ins.

4. Install the package:

- If you are using one of the compressed files (.zip or .tar.gz), unpack the contents (usually to a custom location).
- If you are using an executable .dmg or .dmg), run it.

5. After launching the application, point to a valid Coverity license (license.dat).

6. See the Architecture Analysis online help for product documentation.

---

## Chapter 5. Deployment planning

### Table of Contents

5.1. Deployment checklist .....	61
5.2. How Coverity products integrate into a build system .....	63
5.3. Coverity Analysis deployment models .....	64
5.4. Coverity Connect deployment options .....	67
5.5. Other deployment models and features .....	72

The following sections provide information that will help you make decisions about deployment for your organization for the present -- with the goal of planning for future growth.

After you read this section, you will find it useful to read the ??? so you can see a practical implementation examples of Coverity tools.

### 5.1. Deployment checklist

The deployment checklist is a list of questions that helps you make decisions about how you will deploy Coverity products (for example, which deployment models you will choose) and what hardware you will choose.

When you are thinking about how you will deploy the tools, it is important to recognize that one of the largest contributing factors to the performance of your system is based on system load. System load consists of the following:

- Number of concurrent commits from Coverity Analysis to Coverity Connect - Commits represent the act of sending and processing the Coverity Analysis results and defect data from the intermediate directory to Coverity Connect. The number of and size of your commits can require a lot of hardware resources. The number of commits also increases the size of the database that Coverity Connect uses to store this information.

The basic criteria for commit load is the number of issue instances that are committed per unit of time. This is determined by a number of factors, including the number of code bases, the size of each code base, the number of branches and configurations per code base, and how many of the code bases will be analyzed and how frequently.

- Number of concurrent users of Coverity Connect - The number of concurrent users on a Coverity Connect system can affect the performance of the user interface.
- Number of concurrent Coverity Connect Web Services API calls - The Web Services API allows you to write web applications that communicate with Coverity Connect. The number of concurrent Web Service calls on a Coverity Connect system can affect the performance of the user interface.
- Number of concurrent desktop users - Desktop users fit into the desktop deployment model and as such use both concurrent web services API calls and concurrent commits (although the size of the commits tends to be much smaller than in a central build).

**Table 5.1. Deployment checklist**

Deployment considerations	Results notes	More information
<b>Questions regarding your organization's products</b>		
What platforms do you develop and build on?		Consult the Supported platforms section of this guide to determine operating systems, versions, and required patches for Coverity products.
How many products do you have in your organization?		The number of products can determine the number of streams you that you commit, thereby impacting the commit load.
How many targets are typically built for each product?		Targets are for a given product can determine the number of streams you that you commit, thereby impacting the commit load.
How many lines of code are in your product?		
<b>Questions regarding your organization's build process</b>		
When will a build be generated		Are builds made on demand or integrated as part of an automated process?
How frequently are builds initiated?		The frequency of the builds in relation to the analysis integration contributes to commit load, and thus affects the way you plan for your hardware deployment. For more information, see Chapter 6, <i>Hardware and network recommendations and requirements</i> .
What is the number of concurrent Coverity Analysis commits?		See Section 1.3.4, "Changing the number of concurrent commits".
How is the build command invoked?		Is it through a CI tool (such as Jenkins)? See Section 5.5.2, "Coverity Jenkins plug-in".
Do your developers develop on IDEs?		See the deployment descriptions for the desktop option.
<b>Questions regarding your organization's development process</b>		
How many organizations or business units are using Coverity products?		
How many developers are there in each organization or business unit?		The number of concurrent users will impact the performance of your system, particularly the Coverity Connect UI. Because of this, it is important to have an accurate number of users. Coverity has a list of maximum limits to help ensure optimum performance. If the number of users for your organization exceeds the limits, you could consider implementing one Coverity Connect deployment type over another.

Deployment considerations	Results notes	More information
How are your developers distributed geographically?		How developers in your organization distributed geographically? Do you want developers in other organizations to access a given Coverity Connect instance? You could consider implementing a Coverity Connect clustered environment.
Do you have a "clean before check-in" policy?		See the clean before check-in model.
Do you use a bug-tracking system?		See Section 5.5.3, "Export issues".
Do you have established standard back-up procedures?		See <i>Coverity Platform 2020.12 User and Administrator Guide</i> <a href="#">↗</a> .
Do you have a system validation and monitoring plan?		See Section 9.3, "Monitoring and diagnosing Coverity Connect performance".
Do you plan on using the Coverity deployment maturity model?		See Section 9.1.3, "The Coverity deployment maturity model".

## 5.2. How Coverity products integrate into a build system

Before you make decisions about how to deploy Coverity products into your development environment, it is important to understand how they can be integrated. This section shows how Coverity products are integrated into a typical build environment.

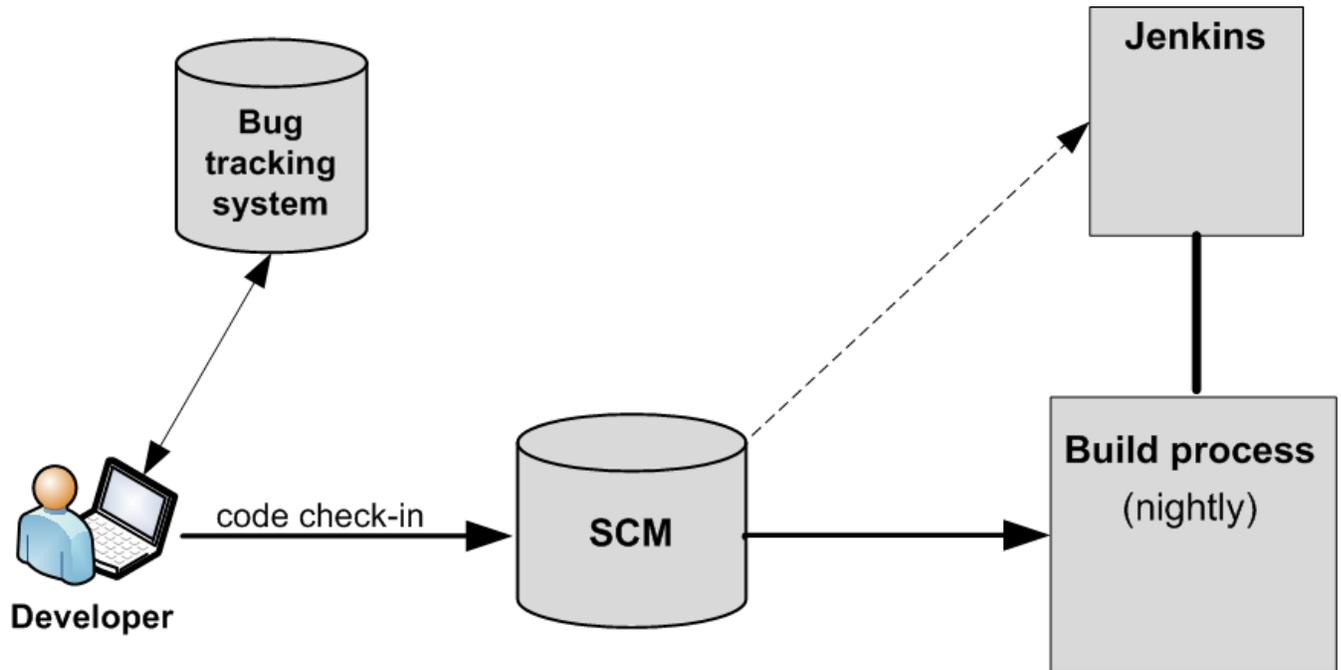
It is important to recognize the various deployment options and recommendations so that you can make decisions about the hardware to which you will install and implement Coverity components. After you have identified how you want to deploy your Coverity system, you should use the hardware recommendation chapter of this manual to set up your environment.

This section begins with a diagram and explanation of a basic build system. The subsequent deployment figures in the section below build upon the basic to show how Coverity tools are integrated into a build system.

### Note

Note that for the diagrams that depict Coverity deployments, the dotted lines represent features or tools that are optional.

The following image shows a high-level view of a basic build system:

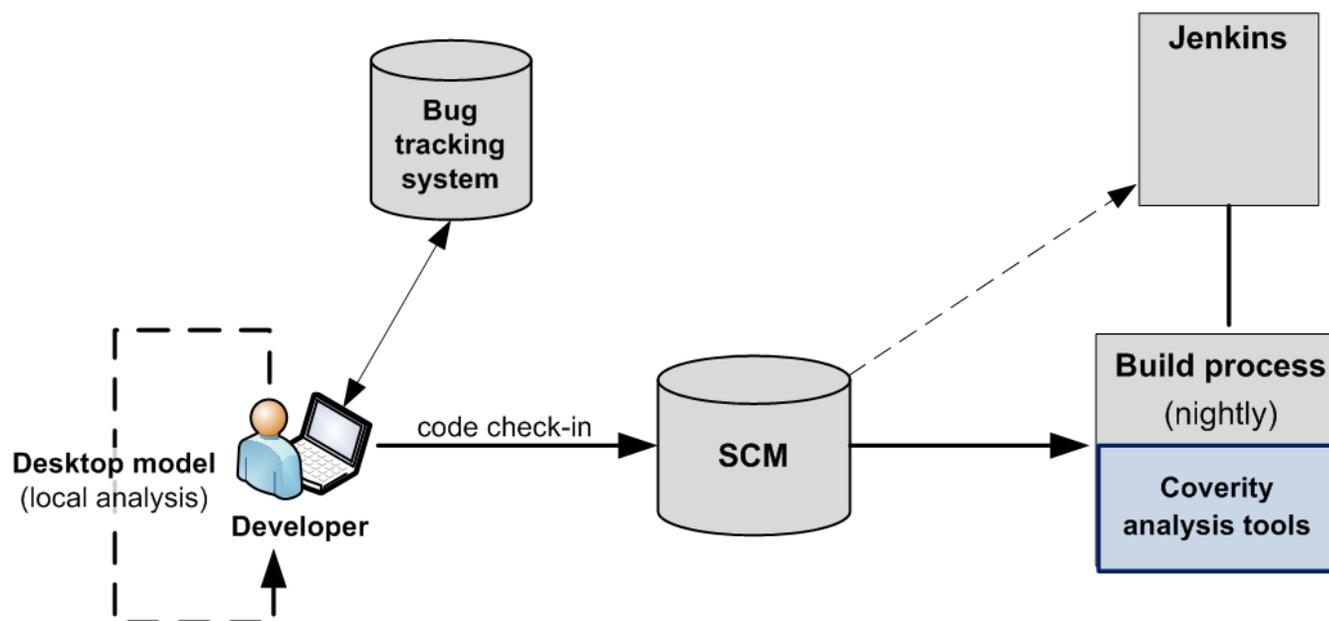


The build process flow is as follows:

1. The software developer makes changes or additions to a section of code.
2. When the developer is satisfied with her/his changes, the code is checked into a source control management (SCM) system.
3. Build execution instructions (such as Makefiles) are invoked through a continuous integration (CI) tool (such as Jenkins) upon receiving notification by the check-in process to the SCM.
4. The code is built at the scheduled interval. In this case, nightly.
5. Build success or failure is reported by the CI tool and notification of the result is sent to the organization.
6. Meanwhile, the developer uses a bug tracking system to locate and manage possible bugs that are found in the software.
7. The developer fixes the assigned bugs. The process repeats.

### 5.3. Coverity Analysis deployment models

The Coverity analysis tools are installed as part of the Coverity Analysis installation package and the goal of these tools is to find issues in your code:



Software organizations generally produce several products, and each product tends to consist of a number of related code branches and targets. These branches and targets might be for the various supported platforms, product versions, trunk and development branches. Coverity Analysis runs over each code base to produce a snapshot of each code base. A snapshot consists of the results of running Coverity Analysis once over a code base. The snapshot includes both the issue information and the version of the source code in which the defects were found and is committed to Coverity Connect after the analysis process is completed.

As your developers continue to modify their code bases, it is useful to provide them with on-going data about the creation of new issues and the elimination of existing ones. Administrators can define streams for each specific code base that they wish to analyze. A stream is a sequence of snapshots over a specified code base. Each time Coverity Analysis runs, the analysis results are grouped with previous results that are made up of the same code base and configuration. Streams capture issue information and trends over time. For more information, see the *Coverity Platform 2020.12 User and Administrator Guide*).

For specific implementation details (including how to configure compilers, integrate with the build, enable checkers and so forth), see the *Coverity Analysis 2020.12 User and Administrator Guide*.

**Note**

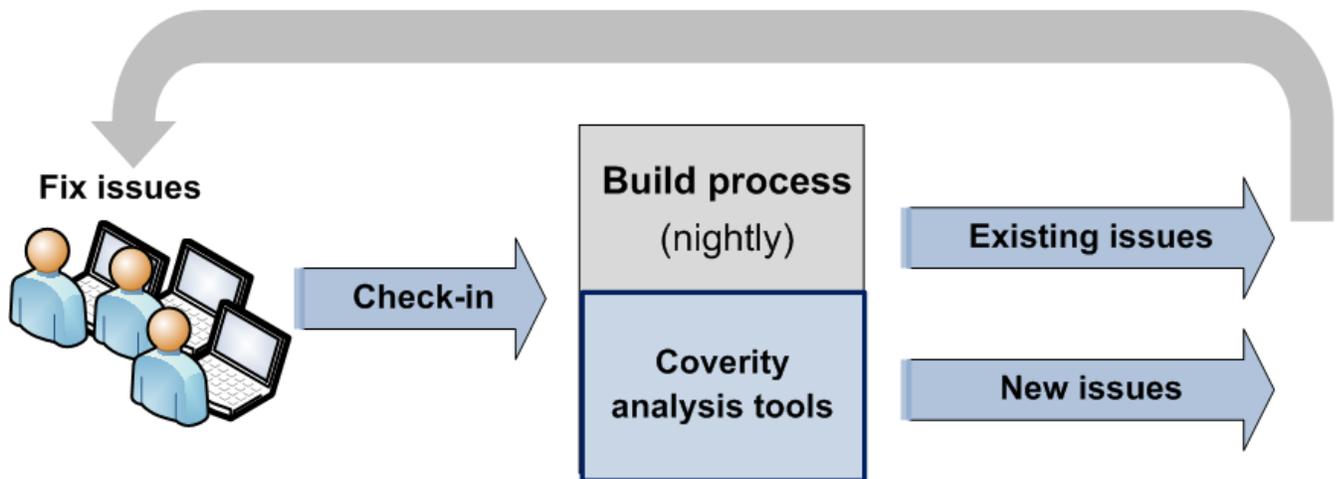
Hardware considerations for the analysis tools are generally established by the needs of your organization's build server (performance, size, and so forth.) However, there are some sizing options to consider. For more information, see Section 6.2, "Coverity Analysis hardware".

### 5.3.1. Central build deployment model

Build engineers typically write scripts that automatically run Coverity Analysis on the source repository at some scheduled interval (typically nightly). They can also allow developers to subscribe to automatically

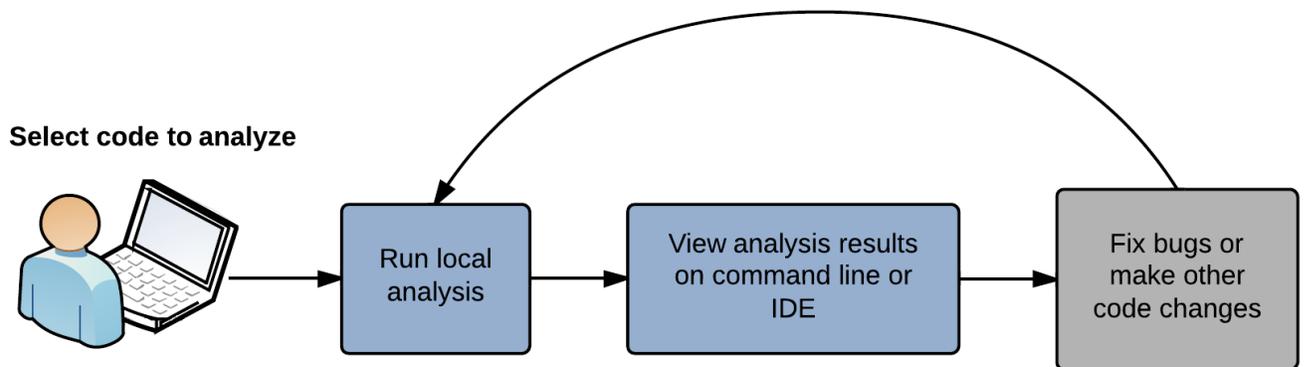
receive email notifications of new issues in their source files. Developers triage and annotate the defects within Coverity Connect or Coverity Desktop. The central build model introduces the least amount of change to the development process and provides a strict separation between developer and build engineer tasks. A developer interacts with Coverity Analysis by adding or modifying source files in the code repository and viewing issue results in Coverity Connect. A build engineer writes scripts to check out the source from the repository, build it, initiate an analysis, and store the results in a Coverity Connect database. Optionally, the build engineer can automatically notify developers of new issues in their source code. The build engineer can integrate Coverity Analysis with the build process to automatically provide Coverity Analysis consumers with fresh snapshots each morning or at another desired interval.

### Assign issues to developers



### 5.3.2. Desktop deployment model (local analysis)

This deployment model, which adds to the central Coverity Analysis build model, allows developers to analyze code locally, and fix defects before checking their source code changes into the repository, which means fewer issues reported in the central Coverity Connect database. Desktop developers also have the ability to focus the analysis on a specific set of source files, which speeds up the analysis considerably, and work with issues directly within their editor or IDE.



It is suggested that the desktop model be deployed individually alongside a central build and analysis. The central analysis should run periodically (perhaps nightly) in order to maintain current analysis data for the full code base. Once this is in place, developers can use Desktop Analysis to get rapid, accurate feedback from Coverity Analysis, drawing on summary data from the central build.

Coverity provides the Desktop Analysis tool, which can be used in one of three ways: from the command line, with a text editor, or with Coverity Desktop, a plug-in to the Eclipse, Wind River WorkBench, QNX Momentics, or Visual Studio IDEs. Coverity Desktop allows you to perform an analysis of your code, then view and triage any issues within the IDE. For more information about Desktop Analysis, see the *Coverity Desktop Analysis 2020.12: User Guide* [↗](#).

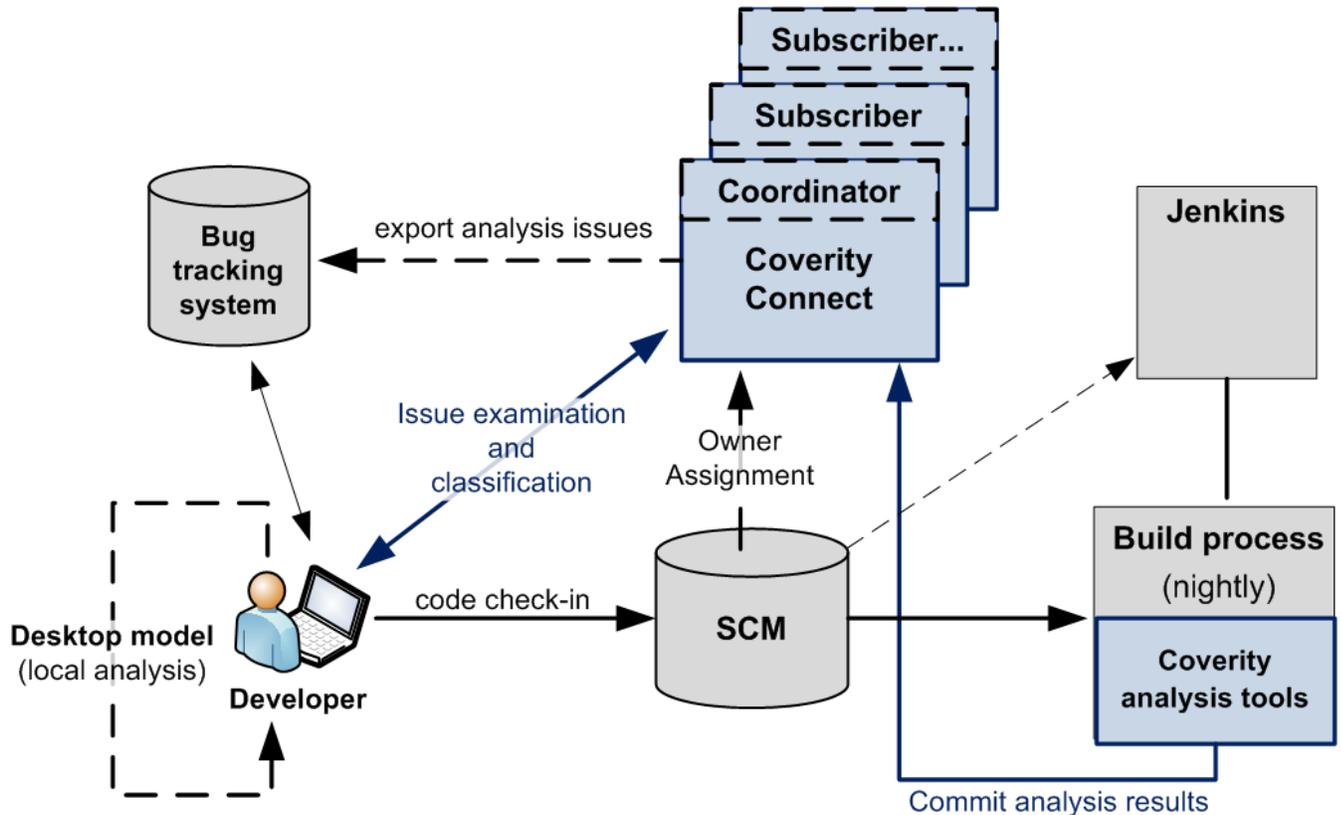
The plugins for each IDE are designed to store various configuration settings in a file, `coverity.conf`. The file can be stored in a directory that is part of source control, so that developers working on a common code base will automatically receive the same settings in some parts of their IDE.

### 5.4. Coverity Connect deployment options

Coverity Connect receives the commit and issue data that was discovered by the Coverity analysis tools. The discovered issues can then be assigned to your organization's developers. The developers, in turn, can view the issues in the source code, classify the issues, and so forth. There are many more powerful features in Coverity Connect; that you can implement to help you locate and fix issues, as well as tracking and charting your organization's projects. For more information about Coverity Connect features, see the *Coverity Platform 2020.12 User and Administrator Guide* [↗](#).

#### **Note**

Coverity Analysis supports SNI (Server Name Indication) for all Coverity client tools. This means that you can place Coverity Connect behind a reverse proxy that serves multiple domains.



There are two primary methods in which to deploy Coverity Connect:

- As a stand-alone deployment.
- As a clustered environment.

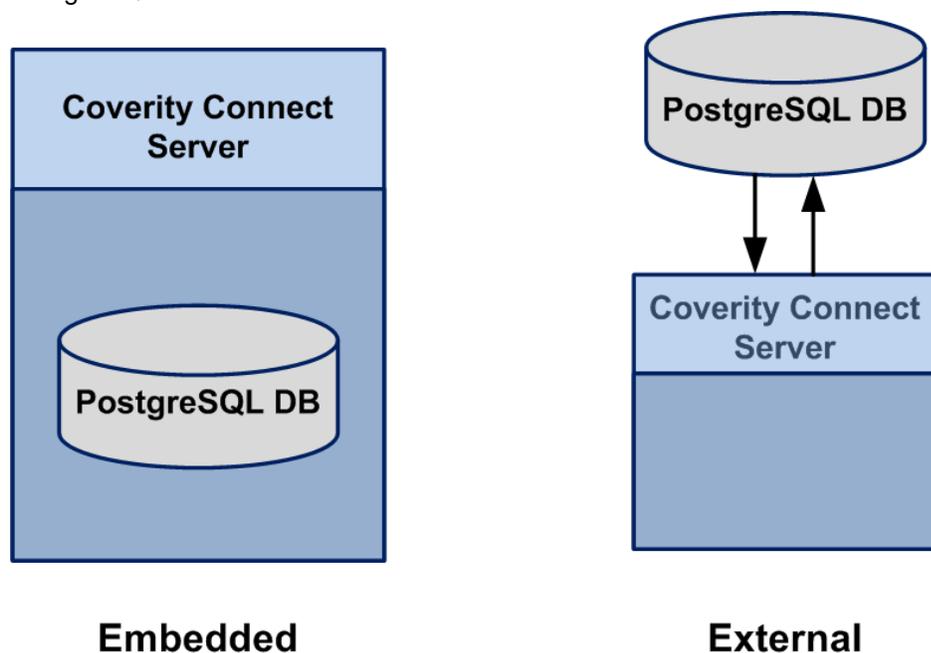
Note that in the diagram above, the clustered environment is represented by the coordinator and subscriber instances. At least one (stand alone or otherwise) Coverity Connect instance is required.

#### 5.4.1. Coverity Connect stand-alone deployments

In this model, Coverity Connect is deployed as a standalone application. Users within your organization will log in to and access a central instance of Coverity Connect. There can be multiple instances of Coverity Connect in your organization, but issue data, classification states (triage), and trending metrics are not shared among the individual instances (for information about deploying Coverity Connect as a shared environment, see Section 5.4.2, "Coverity Connect clustered deployment model"). Coverity Connect comprises two main components:

- The Coverity Connect application server that hosts the application and its associated tools and features.
- The database, which is a PostgreSQL database that contains all of the analysis and defect data, as well as user and system components.

In a standalone environment, Coverity Connect can be installed either with an embedded or external PostgreSQL database:



**Coverity Connect server with an embedded database.** Installs the Coverity Connect application server and the PostgreSQL database at the same time and on the same machine. This option is generally used for its ease of use. It is the default option during installation and is useful for getting the system up and running quickly. It might also be a good alternative for organizations that do not have a dedicated Database Administrator.

**Coverity Connect server with an external database.** Allows you to install the Coverity Connect application server in one location, and associate it with a separately installed PostgreSQL database. The benefits to using an external database are:

- You can maintain and scale the PostgreSQL database separately. For example, you have finer control of database system resources, such as kernel parameters and file system options.
- You can associate the Coverity Connect application server to an existing PostgreSQL database.

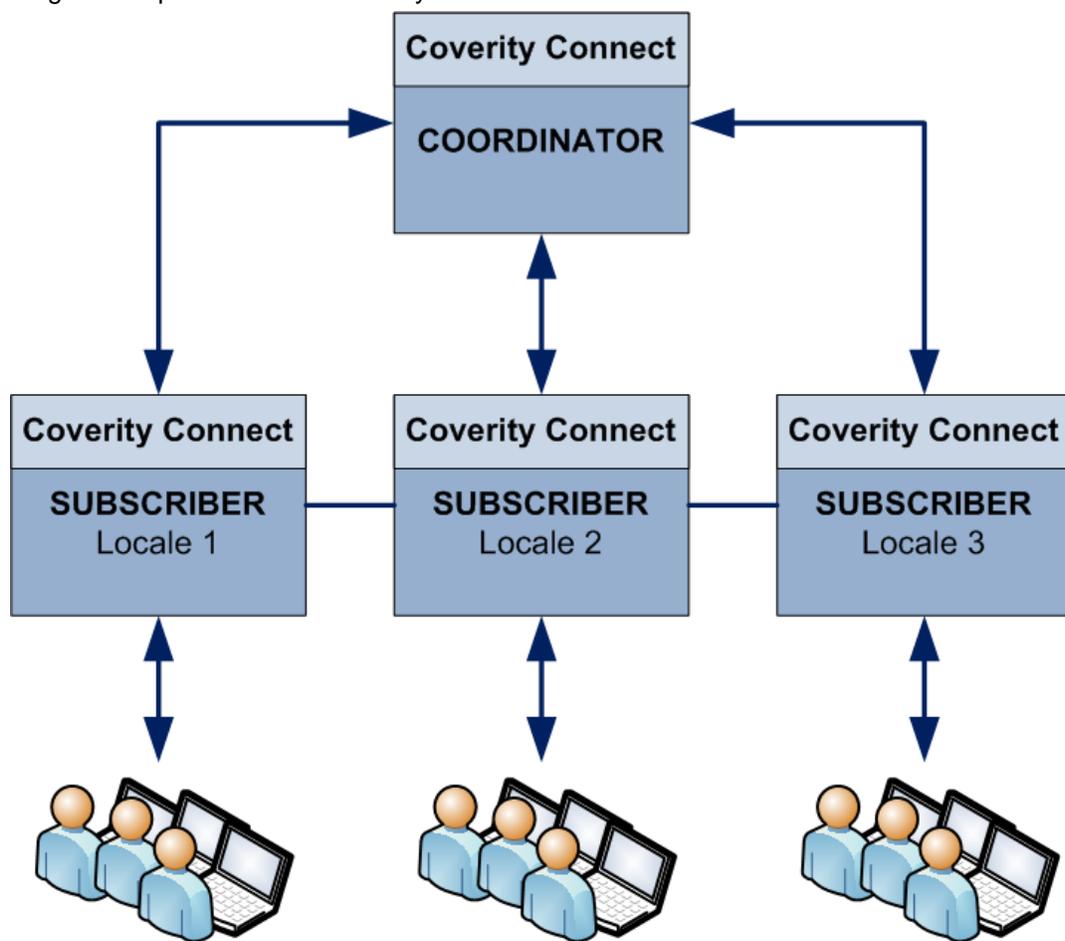
When you install Coverity Connect, you are prompted to choose a Production or Demo performance tuning option. The installer will inform you if your current system settings are not properly configured. If this occurs, you will have to adjust your system settings in order for Coverity Connect to run. For more information, see Section 1.1, "Installing Coverity Connect".

Both of the standalone deployment options are subject to system limits. So, it is recommended that you determine the load on your system using the deployment checklist and then reference your results with the Coverity Connect deployment limits table. If you think that your system might exceed these limits, you might to consider deploying Coverity Connect in a clustered environment.

### 5.4.2. Coverity Connect clustered deployment model

The Coverity Connect clustered deployment model allows you to deploy clusters of Coverity Connect instances on which centrally managed data is synchronized in an enterprise. Importantly, developer-set triage states 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.

When a developer updates an issue through the Coordinator or through a Subscriber, the update propagates to other members of the cluster. In this way, the Coordinator is responsible for synchronizing triage data updates across Coverity Connect Subscribers.



The benefits of using a clustered environment include the following:

- You can distribute the commit load over the subscribers or coordinator and you can choose specific hardware for the clustered components (for example, clustered components that will accept larger commit loads might have a different hardware configuration than those with lesser commits.)

- The number of Coverity Connect users can be distributed across the environment so as to not limit performance. For example, if the number of users on the system exceeds the numbers recommended in the recommended maximum limits for a stand-alone system, you can set up a clustered environment to offset the performance load.
- If you have users in separate geographic locations, you can set up subscribers for any number of locales and still share issue information.
- Clustered components can be set with either embedded or external databases.



**Note**

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.

For more information about how the data is synchronized and to set up a clustered environment, see the *Coverity Platform 2020.12 User and Administrator Guide* [↗](#).

### 5.4.3. Recommended maximum limits in Coverity Connect

Coverity recommends the following boundary limits to ensure that Coverity Connect runs properly and does not experience performance degradation. It is important to estimate for these limits, the results may affect the way you deploy Coverity Connect and the hardware you choose for the deployment. To estimate the limits in your organization, see the Section 5.1, “Deployment checklist”.

See the Glossary for basic definitions of the items described below.

You should not exceed any of the following:

**Table 5.2. Coverity Connect maximum settings**

System item	Maximum number
Streams	1000
Projects	1000
Triage stores	1000 - This number should be much smaller than the number of streams (1 is ideal).
Users	20,000
User Groups	100
Component maps	100
Components per component map	100
Defects per source file	100
Number of lines per source file	10,000
Database size	600 GB
Custom RBAC roles	20

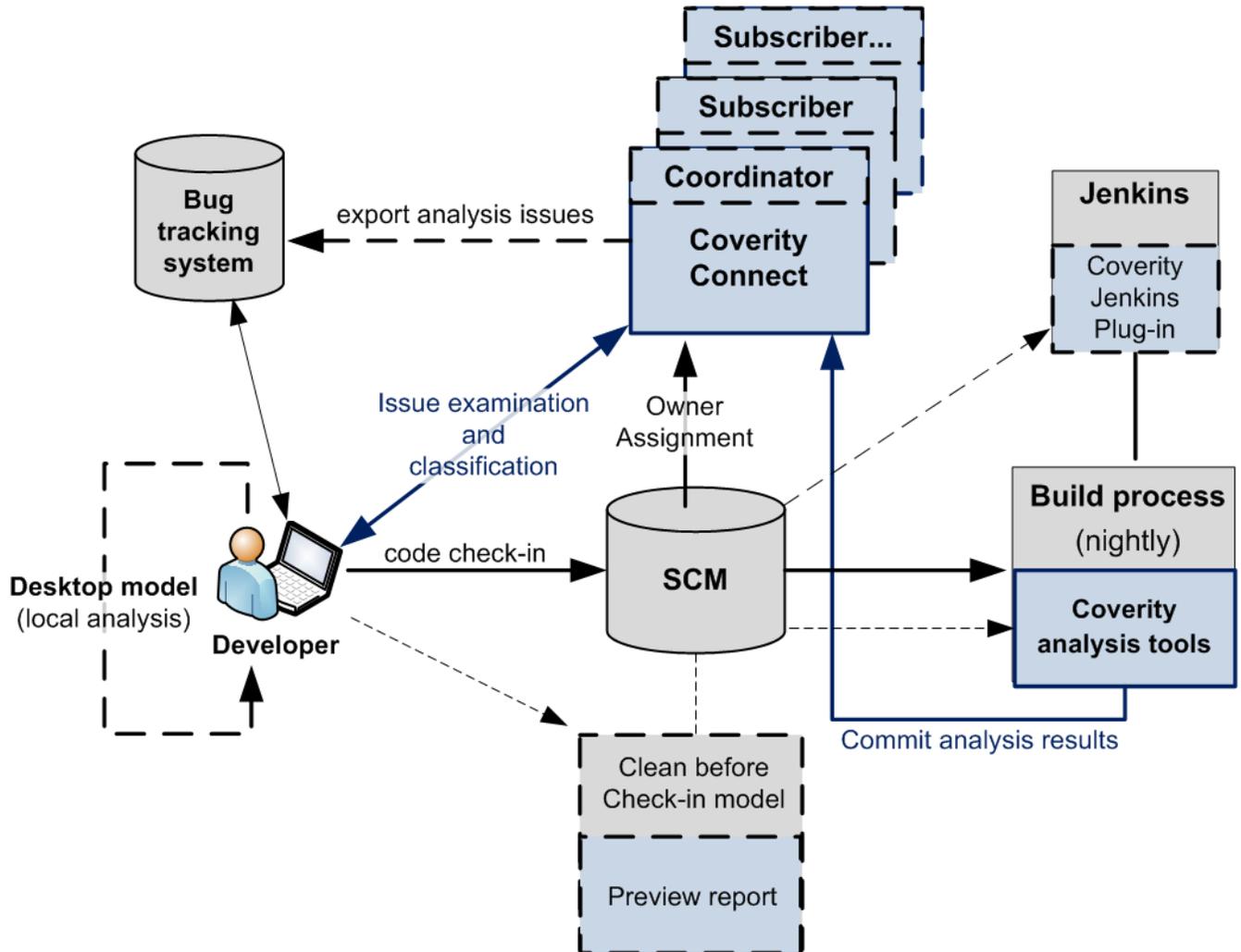
When working with the desktop deployment model, the following deployment memory requirements should support only up to the listed number of users:

- 8GB (medium installer settings) - 100 users
- 16GB - 500 users
- 32GB - 1000 users

This guidance assumes that each Desktop Analysis client performs 20 operations per day and that there are periodic commits against the Coverity Connect server which consist of a full analysis of the code base. The same stream is compared (using the latest snapshot) against each individual `cov-run-desktop` operation.

### **5.5. Other deployment models and features**

The following diagram represents additional (and optimal) deployment considerations that can add value to the developers and administrator of your system.



The flow is as follows:

1. The developer fixes issues after being notified by Coverity Connect
2. The developer checks the fixes into the build.
3. Coverity analysis tools run on the code base.
4. Before the results are committed to Coverity Connect, the build administrator creates the Preview Report to make sure that the code is clean before checking it in.
5. After the code is checked in, results of a passing or failing build is reported in Jenkins.
6. An issue report is exported in XML format and integrated into the organization's third party plug-in.

### 5.5.1. Clean before check-in

The clean before check-in model is a way to verify that your code is "clean" before it is checked in to your source control management (SCM) system. Before you commit the defects you can run a command line executable to produce a Commit Preview Report. This report (in JSON format) shows the current state of the issues on your system. Using this report, you can determine if you want to proceed with the commit.

The definition of "clean" is a policy that is determined by your organization; it is not necessarily defined by Coverity. (Although, Coverity does offer the Deployment Maturity Model -- a Professional Services program to bring your deployment to be "Coverity Clean." See Section 9.1.3, "The Coverity deployment maturity model".) So, these clean policies will most likely differ from organization to organization. For example, your organization may only determine the code to be clean if there are no New issues were found after a given analysis.

### 5.5.2. Coverity Jenkins plug-in

The Coverity plug-in for Jenkins makes it easy to add invocations of Coverity tools to an existing automated build environment. The Jenkins plug-in adds value to the system by:

- Invoking the Coverity Analysis tools during your build
- Failing the build if defects are found matching certain criteria
- Reporting found defects after the build

To get the plug-in, go to <https://plugins.jenkins.io/synopsys-coverity>.

### 5.5.3. Export issues

Coverity Connect allows you to export an Coverity Analysis issue into a file that can be imported into a third-party bug tracking system. This is accomplished by enabling the Export button in the Triage pane in the Coverity Connect user interface. To enable the Export button, you must create and install a utility program to automatically handle the exported file.

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

---

# Chapter 6. Hardware and network recommendations and requirements

## Table of Contents

6.1. Coverity Connect hardware .....	75
6.2. Coverity Analysis hardware .....	76
6.3. Coverity Connect Network Connectivity Requirements .....	79

### 6.1. Coverity Connect hardware

The following table lists the recommended hardware configurations for Coverity Platform deployments. Note that these are recommendations and not necessarily requirements.

The hardware recommendations take into account the following Coverity Connect deployment options:

- Coverity Connect with an embedded database. This deployment includes the PostgreSQL database and the Coverity Connect application installed on the same server and is a common Coverity Platform deployment.
- Coverity Connect with an external database. In this deployment scenario, the Coverity Connect server uses an external PostgreSQL database. For more information, see Section 1.3.1, “Using an external PostgreSQL database with Coverity Connect”.

**Table 6.1. Minimum hardware deployment recommendations**

Server	CPU	RAM	Storage size	Storage device	Notes
Embedded database	Intel Xeon E6 26xx v3 Haswell series or AMD equivalent	32 GB + DDR3 1066Mhz	512GB+	SSD <i>or</i> HDD	SSD: Enabling of TRIM recommended
External database			240GB+		HDD: 7200 rpm recommended

 **Note**

- The minimum number of CPU cores is 8 with a minimum CPU speed of 2.0 Ghz.
- RAM should be 32 GB+ as a starting point. For existing Coverity Connect databases, it is recommended that the amount of RAM be at least 25% of the database size. The database size can be found in Coverity Connect by navigating to Help → About... → Database Size.
- Recommendations based on database size are a rough estimate.
- Performance depends on a variety of factors (commits, web access, web services traffic, etc.) and can't be calculated by a formula. In addition, the performance SLA might vary.
- Requirements vary over time as database growth and usage patterns change.

- You should monitor resource usage (Java and PostgreSQL processes) and modify performance and resource allocations as necessary.
- Here is an example configuration for a typical production performance host running Coverity Connect with an embedded database as of 2018: 24 cores, 128GB RAM, 2TB HDD, 1TB SSD
- Virtualized environments are not recommended for optimal performance.
- If you deploy to a VM, ensure that the VM is at least on par with the recommended minimum configuration.
- Troubleshooting performance problems on a VM is difficult because of the lack of visibility into the VM host's resource usage. CPU cores and RAM can be limited on an underprovisioned VM host.
- Sharing and available IOPS should be on par with the SSD/HDD.
- Storage virtualization is not recommended. Such systems might have problems achieving low-latency I/O performance.
- Most RAID controllers do not support TRIM on RAID volumes.
- TRIM is required to maintain performance and longevity of the SSD.
- RAID configurations with parity (such as RAID 5) are sub-optimal for database I/O.

### 6.1.1. Database size guidelines

The amount of RAM available limits the size of the Coverity Connect database. It is recommended that the amount of RAM be at least 25% of the database size.

In the minimum hardware configuration, there is a minimum of 32GB of RAM. If the JVM heap setting is 75% of system memory, or 24GB, then the database size can reach approximately 96GB before there are any performance problems. You should periodically check the size of the Coverity Connect database and consider provisioning more resources for the server if the database size increases to more than four times the amount of available RAM.

The database size can be found in Coverity Connect by navigating to Help → About... → Database Size.

## 6.2. Coverity Analysis hardware

Coverity Analysis has certain minimum requirements for memory size. Though the speed of the analysis can increase many times through the use CPU parallelism and extra memory, it is important to note the following constraints:

- The speed of the analysis depends on the analysis configuration.
- There are points of rapidly diminishing return beyond which neither additional CPU parallelism nor additional memory will increase the speed significantly.

## 6.2.1. Minimum requirements

When you start an analysis, Coverity Analysis will automatically compute how many analysis workers to use based on the amount of memory the analysis requires and how much memory is available. It will issue a warning if there is not enough memory.

You can use this section to understand memory requirements without starting an analysis.

### CPU

There are no CPU minimums.

### Memory

1.5 GiB [p. 78 ] minimum. However, running a JavaScript, TypeScript, PHP, Python, Swift, Web application security, or Android application security analysis requires additional memory. See the detailed requirements for the different types of analyses below.

The analysis might use more memory when it detects sufficient hardware.

Tuning the analysis with `cov-analyze` options such as `--max-mem` and `--jobs` affects the required minimum. For information about parallel analysis, see Section 6.2.1.1, “Memory requirements for parallel analysis”.

- When running *only quality* checkers (*not including* JavaScript, TypeScript, PHP, Python, or Swift analysis):

```
1.0 GiB + (0.5 GiB * number of analysis workers)
```

For example, for an analysis that is *not running* any Web application security checkers and not performing JavaScript, TypeScript, PHP, Python, or Swift analysis:

To run six analysis workers, you need 4 GiB ( $= 1 + (0.5 * 6)$ ) of free memory. For eight workers, you need 5 GiB ( $= 1 + (0.5 * 8)$ ).

- When running Java, C#, and Visual Basic Web application security checkers or Android security checkers (with or without the quality checkers), use the following formula to help determine the minimum:

```
1.0 GiB + (0.5 GiB * number of analysis workers) + (3 GiB per million LOC)
```

Minimum memory requirements:

1.5GiB (only if result  $\leq$  1.5 GiB), else the result of this formula.

For example, to run six analysis workers when using a Web application security checker on a 500,000 line code base (0.5 million LOC), you need 5.5 GiB ( $= 1.0 \text{ GiB} + 0.5 \text{ GiB} * 6 + 0.5 \text{ million LOC} * 3 \text{ GiB} = 1 \text{ GiB} + 3 \text{ GiB} + 1.5 \text{ GiB}$ ). However, if your result  $\leq$  1.5 GiB, you still need 1.5 GiB.

- JavaScript and TypeScript checkers:

Separate requirements for the file capture and analysis steps of the workflow apply.

- For JavaScript or TypeScript filesystem capture with `cov-build`: 2GiB
- Analysis of JavaScript or TypeScript with *only quality* checkers (not including Web application security checkers such as DOM\_XSS and SQLI):

```
1.0 GiB + (0.5 GiB * number of analysis workers) + (3 GiB per million LOC)
```

- Analysis of JavaScript or TypeScript, including Web application security checkers such as DOM\_XSS and SQLI:

```
1.0 GiB + (2.0 GiB * number of analysis workers) + (3 GiB per million LOC)
```

- PHP, Python, Ruby, and Swift checkers:

- For filesystem capture with `cov-build`: 1.5GiB

- For analysis:

```
1.5 GiB + (2.0 GiB * number of analysis workers) + (3 GiB per million LOC)
```



#### Note

1 GiB is  $2^{30}$  bytes (approximately 1.074 GB).

### 6.2.1.1. Memory requirements for parallel analysis

The analysis can use hardware CPU parallelism (multiprocessor, multi-core, and multi-thread), assuming adequate additional memory is available, but increases in speed depend on a number of factors:

- Whether you are running Web application security analysis or Android application security analysis: Parallel analysis increases the speed of quality analysis only. It does not affect the speed of the Web application security analysis or the Android application analysis.
- In general, the analysis runs faster with more threads, but the scalability of that speed increase depends on the kind of analysis, the code language(s), and other properties of the code base. The analysis of C code typically parallelizes best, followed by C++, followed by C# and Java quality analysis, followed by Android security analysis and Web application security analysis. Android and Web application security analysis is largely unparallelized.

Using available CPU parallelism requires sufficient memory. The number of analysis worker processes can be up to the number of logical CPUs on the system (the number of processes that can run simultaneously on the hardware), but each worker requires more memory to run. You can use the following formula to estimate the minimum amount of memory that you are likely to need (noting that more memory is safer, but less might work, especially for code bases smaller than 1 million LOC). For detailed memory requirements, see Section 6.2.1, “Minimum requirements”.

### 6.2.2. Disks

Analysis of large codebases (MLOC) is especially sensitive to the random access speed of disks, though extra memory can mitigate that issue somewhat by serving as OS disk buffers. Solid state disks (SSDs)

show a clear advantage over mechanical fixed disks ("hard drives"), and the IOPS rating of an SSD is the best predictor of its performance with Coverity analysis (higher is better).

### 6.2.3. Operating system considerations

Coverity tools run notably slower on Windows than on other operating systems, such as Linux.

### 6.2.4. Shared machines

When the analysis runs simultaneously with other programs and makes a significant use of machine resources, the primary concern is memory. With adequate physical memory for active processes and adequate virtual memory (swap space) for inactive ones, sharing CPU time among other processes does not pose problems. If adequate memory is unavailable, you can reduce the memory usage by configuring the analysis to use only part of available CPU parallelism.

You can think of the memory as "free physical memory" requirements, though when running the analysis alone under a basic user environment, that is close enough to "total physical memory".

### 6.2.5. Virtual machines

The analysis is particularly sensitive to the overheads of running under a virtual machine. Better virtual machine technologies significantly lower the observed overheads of worse technologies, but the fastest analysis is achieved by running directly on the hardware.

## 6.3. Coverity Connect Network Connectivity Requirements

**Table 6.2. Coverity Connect Network Connectivity Requirements**

Port name	Default port number	Protocol	Notes
HTTPS port	8443	<b>HTTPS</b> – Also hosts the Commit protocol via WebSocket protocol.	Secured web service port for API and GUI clients. See the <i>Coverity Platform 2020.12 User and Administrator Guide</i> <a href="#">🔗</a> for information on how to enable this port and set its certificates.
HTTP port	8080	<b>HTTP</b> – Also hosts the Commit protocol via WebSocket protocol.	Non-secured web service port for API and GUI clients.   <b>Warning</b>  Appropriate only for demonstration purposes because it's unencrypted. Credentials and sensitive data are visible in transit. See

Hardware and network recommendations and requirements

Port name	Default port number	Protocol	Notes
			the <i>Coverity Platform 2020.12 User and Administrator Guide</i> <a href="#">🔗</a> for information on how to configure and disable this port.
Commit port	9090	<b>Commit</b> – Proprietary protocol used for uploading analysis results. This protocol is deprecated.	The Commit port must be reachable by clients if the Commit protocol is still being used in your environment to upload analysis results. The Commit protocol shares TLS certificates with the HTTPS protocol. See the <i>Coverity Platform 2020.12 User and Administrator Guide</i> <a href="#">🔗</a> for information on how to configure these certificates.
		<b>Remote Config</b> – Proprietary protocol used for communication between Coverity Connect instances in a Coverity Connect cluster.	The Commit port must also be reachable by the other Coverity Connect instances in a Coverity Connect cluster environment. The Remote Config protocol uses a dedicated, private certificate store. See the <i>Coverity Platform 2020.12 User and Administrator Guide</i> <a href="#">🔗</a> for information on how to configure this certificate store.
Control port	8005	<b>TCP</b>	Receives a message from <code>cov-stop-im</code> and <code>cov-im-ctl</code> commands that tells Coverity Connect to shut down. The Control port is used only on the loopback interface. It should not be externally exposed.

---

## Chapter 7. Supported platforms

### Table of Contents

7.1. Coverity product components: Supported versions and compatibility .....	81
7.2. Coverity Connect, Policy Manager, and Reports platforms .....	82
7.3. Coverity Analysis .....	84
7.4. Coverity Test Advisor SCM and platform support .....	91
7.5. Architecture Analysis .....	98
7.6. Coverity Desktop .....	99

Coverity provides technical assistance explaining, configuring, debugging, and providing bug fixes and work-arounds based on service level agreements on all supported platforms listed in this part. Refer to the maintenance service terms for more details.

#### Important

Support for this version of Coverity will be discontinued 18 months after the next major release.

### 7.1. Coverity product components: Supported versions and compatibility

Coverity product components include: Coverity Connect, Coverity Analysis, Dynamic Analysis, Test Advisor, and Coverity Desktop plug-ins.

#### 7.1.1. Supported versions of Coverity product components

The following table lists currently supported versions of Coverity product components:

**Table 7.1. Coverity product component version support**

Coverity Connect	Coverity Analysis, Dynamic Analysis, Test Advisor	Coverity Desktop
2019.03	2019.03	Unsupported
2019.06	2019.06	
2019.09	2019.09	
2019.12	2019.12	
2020.03	2020.03	2019.03
2020.06	2020.06	2020.06
2020.09	2020.09	2020.09

#### 7.1.2. Compatibility between Coverity product components

Coverity product components are compatible in the following ways:

## Supported platforms

- Any supported version of Coverity Connect is compatible with any supported version of Coverity Analysis.
- Coverity Analysis, Dynamic Analysis, Test Advisor, and Coverity Desktop plug-ins must all be the same supported version.
- Coverity Connect is compatible with the current version of Coverity Desktop plugins and the previous two quarterly releases.

 **Note**

If your Coverity Connect version is older than your Coverity Analysis version, you might see issue descriptions in English rather than your preferred language because localized versions of new issue descriptions are available only in the latest version of Coverity Connect.

## 7.2. Coverity Connect, Policy Manager, and Reports platforms

Coverity Connect, Coverity Policy Manager, and Coverity Reports support the following server platforms and browsers.

**Table 7.2. Coverity Connect, Coverity Policy Manager, and Coverity Reports server platform support**

Host OS	Host OS version	32- or 64-bit	Hardware architecture	Notes
Windows	Windows workstation releases: Windows 8.1 or higher.	64-bit	x86_64	
	Windows server releases: Windows Server 2012 or higher.			
Linux	Linux Kernel v2.6.32 or higher, GTK2+ or higher, glibc 2.12 (glibc 2.14 for Coverity Reports) or higher.			Operations using HTTPS connections to Coverity Connect might hang if the Coverity Connect instance is hosted on an Ubuntu server of version 11.10 or earlier.

**Table 7.3. Coverity Connect, Coverity Policy Manager, and Coverity Reports browser support**

Browser	Version	Notes
Internet Explorer	11	

## Supported platforms

---

Browser	Version	Notes
Microsoft Edge	Windows 10-supported versions	
Firefox	Mozilla-supported versions	Coverity supports only the Firefox and Chrome versions that are under maintenance, and deprecates all end-of-life versions.
Google Chrome	Google-supported versions	
Safari	8 and later	Support for versions of Safari that Apple no longer supports is deprecated, and will be removed from a future release.

The Coverity Desktop plug-ins for Eclipse, Microsoft Visual Studio, and other supported IDEs require the same version for the Coverity Analysis and Coverity Platform installations for which Coverity Desktop is configured to use. If you use Coverity Desktop, you must upgrade all Coverity products together to the same version.

### 7.2.1. Coverity Connect software requirements

Coverity Connect requires the following software on the server-side operating system:

- On Windows:
  - kernel32.dll
  - powrprof.dll
  - versionhelpers.h
- On Linux:
  - glibc
  - udev

Coverity Connect requires the following software on the client side:

- A supported browser. See Table 7.3, “Coverity Connect, Coverity Policy Manager, and Coverity Reports browser support”.
- JavaScript enabled.
- Display resolution of at least 1024 x 768 pixels recommended.

Coverity Connect supports TLS v1.2. Client tools that invoke Coverity Connect should also be TLS v1.2-compliant. For example, OpenSSL requires version 1.0.1 or newer, and cURL requires version 7.3.4.0 or newer.

The Coverity Connect server is bundled with the following software:

- Apache Tomcat 8.5

- PostgreSQL 10.12 (the embedded database)
- OpenJDK 11.0.6

External PostgreSQL database:

- In addition to the embedded database, Coverity Connect supports the following external database:  
PostgreSQL 9.5.0–10.9
- The configuration, deployment, and management of external databases should be handled by experienced PostgreSQL DBAs

## 7.3. Coverity Analysis

### 7.3.1. Supported platforms for Coverity Analysis

This section describes Coverity Analysis support per language and per platform. The tables in this section list all platforms and versions supported by Coverity Analysis. Specific language support is broken out per platform.

Virtual machine (VM) implementations of the supported platforms are also supported if you use FlexNet licensing or a default license that is not node-locked to a particular host machine (see the section "Supported platforms for Extend SDK and FLEXnet Licensing" in this chapter for more information).

The notes in the first table pertain to all platforms.

**Table 7.4. Language support notes**

Language	Notes
C/C++	Compliance analysis is available with all C/C++ analysis platforms except AIX and NetBSD.
CUDA	Compliance and C++ analysis is available with all CUDA analysis platforms.
Java	Coverity supports the execution of Web application security analysis and SpotBugs analysis (through Coverity Analysis for Java). All Java analysis requires Oracle Java SE Runtime Environment 8 (JRE-8) platform support.  Coverity does not support the Oracle JRockit JDK.  For information about invoking a particular Java compiler with cov-build, see the table Java supported compilers for Coverity Analysis in this chapter.
JavaScript, TypeScript	Coverity supports the execution of JSHint analysis (through Coverity Analysis for JavaScript) on platforms supported by Node.js 14.12.0.

## Supported platforms

Language	Notes
Kotlin	Coverity supports the execution of Detekt analysis (through Coverity Analysis for Kotlin) and requires Oracle Java SE Runtime Environment 8 (JRE 8) platform support.
Objective-C/Objective-C++	For details on the Clang compiler see the section Supported Compilers: Coverity Analysis for C/C++.

**Table 7.5. AIX: Coverity Analysis platform support**

Platform version	C/C++	Notes
AIX 7.1 on PowerPC	✓	Only manually integrated build capture is supported because the cov-build command is not available, and it is necessary to run cov-analyze on a fully supported platform. For details, see the references to AIX in the Coverity Analysis 2018.09 User and Administrator Guide.

**Table 7.6. FreeBSD: Coverity Analysis platform support**

Platform version	C/C++ analysis	Notes
FreeBSD 8.4 (64-bit) on amd64 (x86_64)	✓	Deprecation Notice: Support for FreeBSD 8.4 is deprecated as of 2019.09 and will be removed in a future release.
FreeBSD 11.3 (32-bit) on i386 (x86)		Deprecation Notice: Support for FreeBSD 11.3 is deprecated as of 2020.12 and will be removed in a future release.
FreeBSD 11.3 (64-bit) on amd64 (x86_64)		
FreeBSD 11.4 (32-bit) on i386 (x86)		
FreeBSD 11.4 (64-bit) on amd64 (x86_64)		
FreeBSD 12.0 (32-bit) on i386 (x86)		Deprecation Notice: Support for FreeBSD 12.0 is deprecated as of 2020.12 and will be removed in a future release.
FreeBSD 12.0 (64-bit) on amd64 (x86_64)		
FreeBSD 12.1 (32-bit) on i386 (x86)		
FreeBSD 12.1 (64-bit) on amd64 (x86_64)		

Supported platforms

**Table 7.7. Linux: Coverity Analysis platform support**

Platform version	C/C++ analysis	C# analysis	Go analysis	Java analysis	JavaScript, Kotlin, PHP, Python, Ruby, Scala, TypeScript analysis	Objective-C/C++ analysis	CUDA analysis	Fortran analysis	Notes
Linux Kernel 2.6.32+ (32-bit) with glibc 2.12–2.27 (32-bit) on x86	✓					✓			Deprecation notice: Support for glibc versions 2.12-2.16 is deprecated as of 2020.12 and will be removed in a future release.
Linux Kernel 2.6.32+ (64-bit) with glibc 2.12–2.27 (64-bit) on x86_64		✓	✓	✓	✓		✓	✓	

**Table 7.8. macOS: Coverity Analysis platform support**

Platform version	C/C++ analysis	C# analysis	Go analysis	Java analysis	JavaScript, Kotlin, PHP, Python, Ruby, Scala, TypeScript analysis	Objective-C/C++ analysis	Swift analysis	Notes
macOS 10.13	✓	✓	✓	✓	✓	✓	✓	C# support on macOS is limited to Unity 2018.3 projects using

Supported platforms

Platform version	C/C++ analysis	C# analysis	Go analysis	Java analysis	JavaScript, Kotlin, PHP, Python, Ruby, Scala, TypeScript analysis	Objective-C/C++ analysis	Swift analysis	Notes
								the Unity Roslyn compiler.  Deprecation notice: Support for macOS 10.13 has been deprecated as of 2020.06.
macOS 10.14								C# support on macOS is limited to Unity 2018.3 projects using the Unity Roslyn compiler.
macOS 10.15								

**Table 7.9. NetBSD: Coverity Analysis platform support**

Platform version	C/C++ analysis	Notes
NetBSD 7.0 32-bit on x86	✓	NetBSD 32-bit support covers the x86 processor only, not other 32-bit processors.
NetBSD 7.0 64-bit on x86_64		
NetBSD 7.1 32-bit on x86		Because of a NetBSD bug, on 64-bit versions of NetBSD, Coverity Analysis will fail while attempting to capture any 32-bit build (or mixed 32 and 64-bit build). To avoid this, you must bypass
NetBSD 7.1 64-bit on x86_64		
NetBSD 7.2 32-bit on x86		
NetBSD 7.2 64-bit on x86_64		
NetBSD 8.0 32-bit on x86		

Supported platforms

Platform version	C/C++ analysis	Notes
NetBSD 8.0 64-bit on x86_64		cov-build and use cov-translate directly.
NetBSD 8.1 32-bit on x86		
NetBSD 8.1 64-bit on x86_64		
NetBSD 9.0 32-bit on x86		
NetBSD 9.0 64-bit on x86_64		

**Table 7.10. Solaris: Coverity Analysis platform support**

Platform version	C/C++ analysis	Notes
Solaris 11 (64-bit) on x86_64	✓	
Solaris 11 (64-bit) on SPARC		

**Table 7.11. Windows: Coverity Analysis platform support**

Platform version	C/C++ analysis	Go analysis	Java analysis	C#/ Visual Basic analysis	JavaScript, Kotlin, PHP, Python, Ruby, Scala, TypeScript analysis	Objective C/C++ analysis	CUDA analysis	Fortran analysis	Notes
Windows 32-bit workstation releases Windows 8.1 and later	✓					✓			
Windows 32-bit server releases Windows Server 2012 R2 and later									
Windows 64-bit workstation releases Windows 8.1 and later		✓	✓	✓	✓		✓	✓	Coverity Analysis for C# and Visual Basic supports

## Supported platforms

Platform version	C/C++ analysis	Go analysis	Java analysis	C#/ Visual Basic analysis	JavaScript, Kotlin, PHP, Python, Ruby, Scala, TypeScript analysis	Objective C/C++ analysis	CUDA analysis	Fortran analysis	Notes
Windows 64-bit server releases Windows Server 2012 R2 and later									analysis of programs compiled by the Visual C# compiler (csc.exe) and Visual Basic compiler (vbc.exe) from .NET Framework versions 3.5 SP1 and 4.5.2–4.8.

### 7.3.2. Supported platforms for Extend SDK and FLEXnet Licensing

This section describes Extend SDK and FLEXnet Licensing support per platform. The following table lists all platforms and versions on which Extend SDK and/or FLEXnet Licensing is supported.

Virtual machine (VM) implementations of the supported platforms are also supported if you use FlexNet licensing or a default license that is not node-locked to a particular host machine.

**Table 7.12. Extend SDK and FLEXnet licensing platform support**

OS	Platform	Extend SDK	FLEXnet licensing	Notes
Linux	Linux Kernel 2.6.32+ (32-bit) with glibc 2.14-2.27 (32-bit) on x86	✓	✓	Deprecation notice: Support for glibc versions 2.14-2.16 is deprecated as of 2020.12 and will be removed in a future release.
	Linux Kernel 2.6.32+ (64-bit) with			

## Supported platforms

OS	Platform	Extend SDK	FLEXnet licensing	Notes
	glibc 2.14-2.27 (64-bit) on x86_64			
macOS	10.13		✓	Deprecation notice: Support for macOS 10.13 has been deprecated as of 2020.06.
	10.14			
	10.15			
Solaris	11 (64-bit) on x86_64	✓	✓	Extend SDK requires the libiconv library, and you must configure the system dynamic loader (ld.so.1) to locate it.
	11 (64-bit) on SPARC			
Windows	Windows 32-bit workstation releases Windows 8.1 and later	✓	✓	
	Windows 32-bit server releases Windows Server 2012 R2 and later			
	Windows 64-bit workstation releases 8.1 and later			
	Windows 64-bit server releases Windows Server 2012 R2 and later			

### 7.3.3. Supported platforms for Coverity Wizard

Coverity Wizard supports the following platforms:

**Table 7.13. Coverity Wizard supported platforms**

Host OS	Version	Notes
Windows	64-bit Windows workstation releases: 8.1 and later	

## Supported platforms

Host OS	Version	Notes
	64-bit Windows server releases: Windows Server 2012 R2 and later	
Linux	64-bit Linux that can run OpenJRE 1.8 and Eclipse 4.4	
macOS	10.13-10.15	Mac OS X and macOS can access Coverity Wizard only for use with Coverity Analysis. Coverity Wizard for Test Advisor is not supported on any Mac platform.  Deprecation Notice: Support for MacOS 10.13 is deprecated as of 2020.06 and will be removed in a future release.

### 7.4. Coverity Test Advisor SCM and platform support

The following tables list the platforms, compilers, and Source Control Management (SCM) systems that are supported for C/C++, Java, and C# test environments.

The minimum system requirements are the same that are defined for Coverity Analysis. For more information, see the Coverity Analysis.

Browser requirements are the same that are defined for Coverity Connect.

#### 7.4.1. Test Advisor supported compilers and platforms

Compiler and platform support varies by programming language.

##### 7.4.1.1. Test Advisor for C/C++ supported compilers and platforms

**Table 7.14. Test Advisor for C/C++ supported compilers and platforms**

Supported coverage tools	Supported compilers	Supported OS	Supported processor architecture	Notes
gcov	GNU GCC, G++ version 3.4.6 to 6.x	Linux	N/A	Deprecation notice Support for gcov is deprecated as of 2020.12 and will be removed in a future release.
Bullseye	Bullseye-supported compilers	Linux Windows	N/A	Deprecation notice: Support

Supported platforms

Supported coverage tools	Supported compilers	Supported OS	Supported processor architecture	Notes
		VxWorks		<p>for Bullseye is deprecated as of 2020.12 and will be removed in a future release.</p> <p>Test Advisor support using Bullseye requires a licensed installation of the BullseyeCoverage tool. The minimum supported version of BullseyeCoverage is 8.7.53. For more information about BullseyeCoverage implementation, see the Test Advisor 2018.09 User and Administrator Guide.</p> <p>Bullseye supports Wind River VxWorks for Real Time Processes (RTPs), so Test Advisor can be used in this situation. Test Advisor also supports CoverageScope for VxWorks Kernel Modules. For more information, see Section 1A in the Test Advisor 2018.09 User</p>

Supported platforms

Supported coverage tools	Supported compilers	Supported OS	Supported processor architecture	Notes
				<p>and Administrator Guide.</p> <p>Test Advisor supports Bullseye-supported compilers, but only those compilers that are officially supported by BOTH Bullseye and Coverity Analysis:</p> <ol style="list-style-type: none"> <li>1. Locate the particular compiler as supported by Bullseye on <a href="http://www.bullseye.com/platform.html">http://www.bullseye.com/platform.html</a>.</li> <li>2. Verify that the compiler is supported by Coverity Analysis.</li> <li>3. If the compiler appears in both support tables, the compiler is supported. If the compiler ONLY appears in the Bullseye support table, then the compiler is not supported by Test Advisor.</li> </ol>
Function Coverage Instrumentation	GNU GCC, G++ version 3.4.6 or later	Linux	ARM, Intel-compatible	Deprecation notice: Support for Function Coverage Instrumentation is deprecated as of 2020.12 and will be

Supported platforms

Supported coverage tools	Supported compilers	Supported OS	Supported processor architecture	Notes
				<p>removed in a future release.</p> <p>The Coverity Function Coverage run-time library must be recompiled for ARM and deployed to the target device in order to collect coverage data. See the Coverity Runtime Library Development Guide for more information.</p>
	Visual Studio 2005 C++ or later	Windows	Intel-compatible	<p>Deprecation notice: Support for Function Coverage Instrumentation is deprecated as of 2020.12 and will be removed in a future release.</p> <p>The compiler version can be determined by running the compiler (cl) on the command line, which returns detailed information.</p> <p>Managed C++ and Common Language Runtime (CLR) are not supported. Compilations with switches beginning with "/CLR" will be skipped.</p>

### 7.4.1.2. Coverity Test Advisor for Java supported compilers and platforms

Table 7.15. Test Advisor for Java supported compilers and platforms

Supported coverage tools	Supported compilers	Supported OS	Notes
Cobertura	Oracle JDK v1.7	Linux	<p>Deprecation notice: Support for Cobertura is deprecated as of 2020.12 and will be removed in a future release.</p> <p>Test Advisor works only with the shipped version of Cobertura (1.9.4.1).</p> <p>Test Advisor works only with the shipped version of JaCoCo (0.7.1).</p> <p>Only JDK 1.8 is supported on Windows 10.</p> <p>Test Advisor ships with the Cobertura and JaCoCo open source Java test coverage tools.</p>
		Windows	
JaCoCo	Oracle JDK v1.7 and v1.8	Linux	<p>Deprecation notice: Support for JaCoCo is deprecated as of 2020.12 and will be removed in a future release.</p> <p>Test Advisor works only with the shipped version of Cobertura (1.9.4.1).</p> <p>Test Advisor works only with the shipped version of JaCoCo (0.7.1).</p> <p>Only JDK 1.8 is supported on Windows 10.</p>
		Windows	

## Supported platforms

Supported coverage tools	Supported compilers	Supported OS	Notes
			Test Advisor ships with the Cobertura and JaCoCo open source Java test coverage tools.

### 7.4.1.3. Coverity Test Advisor for C# supported compilers and platforms

**Table 7.16. Test Advisor for C# supported compilers and platforms**

Supported compilers	Supported OS	32- or 64-bit	Notes
Microsoft C# compiler	Windows	32- or 64-bit	<p>Deprecation notice: Support for Microsoft C# compiler is deprecated as of 2020.12 and will be removed in a future release.</p> <p>Test Advisor ships with the OpenCover open source C# test coverage tool.</p>

### 7.4.2. Test Advisor supported SCM systems

SCM system support is identical for C/C++, C#, and Java.

**Table 7.17. Test Advisor supported SCM systems for C/C++, C#, and Java**

Supported SCM systems	SCM version	Supported OS	Notes
Accurev	7.0–7.4	Linux and Windows	
ClearCase	9.0.0		
CVS	1.11.17–1.12.13		
Git	2.2–2.26		
Mercurial	3.3–5.5		
Perforce	2017.1-2020.1		
Plastic	6.0.16.975–6.0.16.x (where x is greater than 975)		

## Supported platforms

Supported SCM systems	SCM version	Supported OS	Notes
SVN	1.10-1.14		
Team Foundation Server,  Azure DevOps Server	TFS 2012	Windows	Deprecation notice: Support for TFS 2012 is deprecated as of 2020.03 and will be removed in a future release.  Depends on TFS 2012 powertools.
	TFS 2012 Update 1		Deprecation notice: Support for TFS 2012 update 1 is deprecated as of 2020.03 and will be removed in a future release.  Depends on TFS 2012 update 1 powertools.
	TFS 2012 Update 2		Deprecation notice: Support for TFS 2012 update 2 is deprecated as of 2020.03 and will be removed in a future release.  Depends on TFS 2012 update 2 powertools.
	TFS 2013		Depends on TFS 2013 powertools.
	TFS 2015		Depends on TFS 2015 powertools.
	TFS 2017		
	TFS 2018		
	ADS 2019		



### Note

Team Foundation Version Control is the only SCM system that Coverity supports for Team Foundation Server and Azure DevOps Server. Additionally your workspace must be created by the `tf.exe` tool. Workspaces created by tools that use TFS SDK for Java, for example Team Explorer Everywhere (TEE), are not supported.

## 7.5. Architecture Analysis

Architecture Analysis supports the following platforms:

**Table 7.18. Architecture Analysis for C/C++ and C# platform support**

Host OS	Host OS or kernel version	Host CPU	32- or 64-bit	Notes
Windows	Windows Server 2012 R2 and later	x86_64	64-bit	Deprecation Notice: Support for Windows is deprecated as of 2020.06 and will be removed in a future release.
Linux	v2.6.32+, glibc 2.14–2.27	x86	32- or 64-bit	Deprecation Notice: Support for Linux is deprecated as of 2020.06 and will be removed in a future release.
macOS	v10.13–10.14	x86	32- or 64-bit	Deprecation Notice: Support for macOS is deprecated as of 2020.06 and will be removed in a future release.

**Table 7.19. Architecture Analysis for Java platform support**

Host OS	Host OS or kernel version	Host CPU	32- or 64-bit	Notes
Windows	Windows Server 2012 R2 and later	x86_64	64-bit	Deprecation Notice: Support for Windows is deprecated as of 2020.06 and will be removed in a future release.
Linux	v2.6.32+, glibc 2.14–2.27	x86	32- or 64-bit	Deprecation Notice: Support for Linux is deprecated as of 2020.06 and will be removed in a future release.
macOS	v10.13–10.14	x86	32- or 64-bit	Deprecation Notice: Support for macOS

## Supported platforms

Host OS	Host OS or kernel version	Host CPU	32- or 64-bit	Notes
				is deprecated as of 2020.06 and will be removed in a future release.

## 7.6. Coverity Desktop

### 7.6.1. Version Compatibility with Coverity Connect and Coverity Analysis.

The Coverity Desktop plug-ins require an active connection with a current version of Coverity Connect in order to enable all analysis options. A Coverity Desktop plug-in is compatible with any Coverity Connect instance of the same version, or the two most recent versions through the next major release (represented by the first digit in the version number). For example, Coverity Desktop version 2019.09 will be supported by all Coverity Connect versions more recent than 2019.09 and prior to 2020.12.

 **Note**

Please note that the behavior described above is available starting with Coverity Desktop version 7.7. Coverity Desktop versions 7.6 and earlier must be connected to *matching* versions of Coverity Connect.

The Coverity Desktop plug-ins require access to Coverity Analysis installed on the same machine. Coverity Desktop and Coverity Analysis versions must match. Hotfix versions are considered as matching the GA version (for example, 2019.09-x matches 2019.09).

### 7.6.2. IDE and Java version support

Coverity Desktop supports the following IDE and Java versions.

**Table 7.20. Supported IDE and Java versions**

IDE	IDE version	Java version	Notes
Visual Studio	2015–2019		
Android Studio	3.0–4.1	Embedded JRE	
Eclipse	Eclipse 4.8–4.17	Oracle Java 8–14	Support for Eclipse 4.6 has been dropped as of 2019.12 for the Eclipse specific plugin. Other Eclipse variant IDEs such as IBMRTC and DS-5 which are based on Eclipse 4.6 are still supported. We will only support issues from Eclipse 4.6 if they come
IBM Rational Team Concert (bundled with Eclipse)	Eclipse 4.6, 4.8–4.17		
ARM Development Studio 5 (bundled with Eclipse)			

## Supported platforms

IDE	IDE version	Java version	Notes
			<p>from one of these two variant IDEs.</p> <p>Deprecation Notice: Support for Java 14 is deprecated as of 2020.12 and will be removed in a future release.</p>
WindRiver WorkBench	4.0		<p>The Java Version column indicates the Java version required for the IDE to run. For information regarding the JDK version required for Java code analysis, refer to Supported compilers: Coverity Analysis (Quality analysis) for Java in this chapter.</p> <p>Deprecation Notice: Support for Java 14 is deprecated as of 2020.12 and will be removed in a future release.</p>
QNX Momentics	7.0		
IntelliJ IDEA	2017.2–2020.2.1		<p>The Java Version column indicates the Java version required for the IDE to run. For information regarding the JDK version required for Java code analysis, refer to Supported compilers: Coverity Analysis (Quality analysis) for Java in this chapter.</p> <p>Deprecation Notice: Support for Java 14 is deprecated as of 2020.12 and will be removed in a future release.</p>
CLion			
RubyMine			
WebStorm			
PyCharm			
PHPStorm			

## Supported platforms

IDE	IDE version	Java version	Notes
			Deprecation Notice: Support for IntelliJ 2017.2 based IDEs are deprecated as of 2020.12 and will be removed in a future release.

 **Note**

The "Java Version" column indicates the Java version required for the IDE to run. For information regarding the JDK version required for Java code analysis, refer to "Supported compilers: Coverity Analysis (Quality analysis) for Java" in this chapter.

### 7.6.3. Coverity Desktop for Eclipse on supported platforms

Coverity Desktop for Eclipse is supported on the following platforms.

**Table 7.21. Eclipse Plug-in platform support**

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
Windows	Windows workstation releases: Windows 8.1 and higher.  Windows server releases: Windows Server 2012 R2 and higher.	x86_64	Eclipse 4.8–4.17	C/C++, Java, JavaScript, PHP, Python, Ruby, and Scala	
Linux	Linux Kernel 2.6.32+, glibc 2.14–2.27				
macOS	macOS 10.13-10.15				Deprecation Notice: Support for MacOS 10.13 is deprecated as of 2020.06 and will be removed in a future release.

#### 7.6.4. Coverity Desktop for IBM Rational Team Concert (RTC) on supported platforms

Coverity Desktop for IBM Rational Team Concert (RTC) version 6.0.6.1+ is supported on the following platforms.

**Table 7.22. IBM RTC Plug-in platform support**

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
Windows	Windows workstation releases: Windows 8.1 and higher.  Windows server releases: Windows Server 2012 R2 and higher.	x86_64	Eclipse 4.6, 4.8–4.17	C/C++, Java, JavaScript, PHP, Python, and Ruby	Support for Eclipse 4.6 has been dropped as of 2019.12 for the Eclipse specific plugin. Other Eclipse variant IDEs such as IBMRTC and DS-5 which are based on Eclipse 4.6 are still supported. We will only support issues from Eclipse 4.6 if they come from one of these two variant IDEs.
Linux	Linux Kernel 2.6.32+, glibc 2.14–2.27				

#### 7.6.5. Coverity Desktop for ARM Development Studio 5 (DS-5) on supported platforms

Coverity Desktop for ARM Development Studio 5 (DS-5) version 5.22+ is supported on the following platforms.

**Table 7.23. DS-5 Plug-in platform support**

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
Windows	Windows workstation releases: Windows 8.1 and higher.	x86_64	Eclipse 4.6, 4.8–4.17	C/C++	Support for Eclipse 4.6 has been dropped as of 2019.12 for the

## Supported platforms

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
	Windows server releases: Windows Server 2012 R2 and higher.				Eclipse specific plugin. Other Eclipse variant IDEs such as IBMRTC and DS-5 which are based on Eclipse 4.6 are still supported. We will only support issues from Eclipse 4.6 if they come from one of these two variant IDEs.
Linux	Linux Kernel 2.6.32+, glibc 2.14–2.27				

### 7.6.6. Coverity Desktop for Wind River Workbench on supported platforms

Coverity Desktop for Wind River Workbench is supported on the following platforms.

**Table 7.24. Wind River Workbench Plug-in platform support**

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
Windows	Windows workstation releases: Windows 8.1 and higher.  Windows server releases: Windows Server 2012 R2 and higher.	x86_64	WindRiver WorkBench 4.0	C/C++, Java	
Linux	Linux Kernel 2.6.32+, glibc 2.14–2.27				

### 7.6.7. Coverity Desktop for QNX Momentics IDE on supported platforms

Coverity Desktop for QNX Momentics IDE is supported on the following platforms.

**Table 7.25. QNX Momentics Plug-in platform support**

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
Windows	Windows workstation releases: Windows 8.1 and higher.  Windows server releases: Windows Server 2012 R2 and higher.	x86_64	QNX Momentics 7.0	C/C++, Java, JavaScript, PHP, Python, and Ruby	
Linux	Linux Kernel 2.6.32+, glibc 2.14–2.27				

**7.6.8. Coverity Desktop for Microsoft Visual Studio IDE on supported platforms**

Coverity Desktop for Microsoft Visual Studio IDE is supported on the following platforms:

**Table 7.26. Coverity Desktop for Microsoft Visual Studio IDE platform**

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
Windows	Windows workstation releases: Windows 8.1 and higher.	x86_64	Visual Studio 2015–2019	C/C++, Java, JavaScript, PHP, Python, and Ruby	

**7.6.9. Coverity Desktop for IntelliJ IDEA on supported platforms**

Coverity Desktop for IntelliJ IDEA is supported on the following platforms.

**Table 7.27. IntelliJ IDEA Plug-in platform support**

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
Windows	Windows workstation releases: Windows 8.1 and higher.	x86_64	IntelliJ 2017.2–2020.2.1	C/C++, Java, JavaScript, PHP, Python, Ruby, and Scala	Deprecation Notice: Support for IntelliJ 2017.2 based IDEs are deprecated as of 2020.12

## Supported platforms

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
	Windows server releases: Windows Server 2012 R2 and higher.				and will be removed in a future release.
Linux	Linux Kernel 2.6.32+, glibc 2.14–2.27		IntelliJ 2017.2–2020.1		
macOS	macOS 10.13-10.15		IntelliJ 2017.2–2020.2.1		<p>Deprecation Notice: Support for MacOS 10.13 is deprecated as of 2020.06 and will be removed in a future release.</p> <p>Deprecation Notice: Support for IntelliJ 2017.2 based IDEs are deprecated as of 2020.12 and will be removed in a future release.</p>

### 7.6.10. Coverity Desktop for Android Studio on supported platforms

Coverity Desktop for Android Studio is supported on the following platforms.

**Table 7.28. Android Studio Plug-in platform support**

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
Windows	Windows workstation releases: Windows 8.1 and higher.  Windows server releases:	x86_64	Android Studio 3.0–4.1	Java	

## Supported platforms

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
	Windows Server 2012 R2 and higher.				
Linux	Linux Kernel 2.6.32+, glibc 2.14–2.27				
macOS	macOS 10.13-10.15				Deprecation Notice: Support for MacOS 10.13 is deprecated as of 2020.06 and will be removed in a future release.

### 7.6.11. Coverity Desktop for other IntelliJ-based IDEs on supported platforms

Coverity Desktop for other IntelliJ-based IDEs (such as RubyMine, WebStorm, PyCharm, PhpStorm, and CLion) is supported for the following platforms.

**Table 7.29. IntelliJ-based IDE Plug-in platform support**

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
Windows	Windows workstation releases: Windows 8.1 and higher.  Windows server releases: Windows Server 2012 R2 and higher.	x86_64	CLion 2017.2–2020.2.1  PhpStorm 2017.2–2020.2.1  PyCharm 2017.2–2020.2.1  RubyMine 2017.2–2020.2.1	C/C++, Java, JavaScript, PHP, Python, and Ruby	Deprecation Notice: Support for IntelliJ 2017.2 based IDEs are deprecated as of 2020.12 and will be removed in a future release.
Linux	Linux Kernel 2.6.32+, glibc 2.14–2.27				
macOS	macOS 10.13-10.15		WebStorm 2017.2–2020.2.1		Deprecation Notice: Support for IntelliJ 2017.2 based IDEs are

---

Supported platforms

---

Host OS	Host OS or kernel version	Host CPU	Vendor IDE and version	Supported languages	Notes
					deprecated as of 2020.12 and will be removed in a future release.  Deprecation Notice: Support for MacOS 10.13 is deprecated as of 2020.06 and will be removed in a future release.

---

## Chapter 8. Supported languages, compilers, and frameworks for Coverity Analysis

### Table of Contents

8.1. C/C++ compilers .....	108
8.2. C# compilers .....	126
8.3. CUDA compilers .....	126
8.4. Go compilers .....	127
8.5. Java compilers .....	128
8.6. Kotlin compilers .....	128
8.7. Scala compilers .....	129
8.8. Swift compilers .....	129
8.9. Visual Basic compilers .....	130
8.10. Third-party compilers .....	130
8.11. Frameworks .....	130

### 8.1. C/C++ compilers

Coverity provides, and supports the Compiler Integration Toolkit (CIT) (CIT). This toolkit is used to integrate compilers with Coverity Analysis for C/C++. Coverity acknowledges that not all compilers can be integrated using CIT. Coverity provides compiler integrations out of the box for many popular compilers (listed below).

Coverity provides commercially reasonable efforts to support documented compiler features driven by market need. Coverity does not generally support undocumented or unintended language features. To properly analyze files that use undocumented, unintended, or non-standard features, customers might need to customize their configuration or change their code.

Non-CIT issues with compilers that have not been integrated by Coverity will be handled on a case by case basis but are generally considered "unsupported". Since Coverity will not be able to validate these issues, we will require very detailed and precise problem reports to begin the investigation.

Unless otherwise specified, Coverity does not support new or changed language features specified in any of the following:

- Unpublished C and C++ language standards.
- C and C++ technical reports (TRs) or technical specifications (TSs).

Examples of such specifications include ISO/IEC TR 18037 (Programming languages—C—Extensions to support embedded processors), ISO/IEC TS 19217:2015 (Information technology—Programming languages—C++ Extensions for concepts), and ISO/IEC TS 21544:2018 (Programming languages—Extensions to C++ for modules).

Coverity 2012.12 provides provisional support for the C++20 language standard. Provisional support means that some functionality, such as the parsing of C++20-specific code, defect presentation, and syntax highlighting, might not work properly, and that the Coverity Analysis engine might not include new

features for analysis. Support for ISO/IEC JTC1/SC22/WG21 proposals are detailed in Table 8.2, “C++20 core language features”. Descriptions of support for the major language features are as follows:

- Concepts – Support for most use cases, including defining concepts and applying constraints.
- Coroutines – Complete parsing support. Coroutines are currently excluded for the purposes of analysis.
- Modules – Not supported.
- Three-way Comparison – Supported.

Coverity provides the following compiler integrations:

**Table 8.1. Coverity Analysis for C/C++ compiler integrations**

Compiler	Version	Host OS	Target	Notes
Analog Devices Compilers	8.5.0.0–8.12.0.0	Windows	Blackfin	ISO/IEC TR 18037 fixed point extensions are supported for C (not C++) code on Blackfin and SHARC.  Deprecation Notice: Support for Analog Devices Compiler on Blackfin less then version 8.12.0.0 is deprecated as of 2020.12 and will be removed in a future release.
	8.0.0.8–8.12.0.0		SHARC	
	7.3.0.5		TigerSHARC	
ARM C and C++	5.0–6.13.1	Windows/Linux	ARM	Deprecation Notice: Support for ARM C/C++ 5.04 for Nintendo 3DS is deprecated as of 2020.12 and will be removed in a future release.
	5.04	Windows	Nintendo 3DS	
Borland C++	CodeGear C++ 5.93	Windows	x86	The VCL library, which is shipped with Borland compilers, is not supported.
	Borland C++ 5.5.1			
CEVA compilers	17.0	Linux	CEVA-X	

Supported languages, compilers, and frameworks for Coverity Analysis

Compiler	Version	Host OS	Target	Notes
	b753	Windows	CEVA-TL4CC	
	b480		CEVA-XC12	
	10.3–16.1		CEVA-XC321, CEVA-XC323, CEVA-XC4210, CEVA-XC4500	
	b453		Windows/Linux	
Clang	Android NDK Clang 3.1–3.4 (NDK revisions r8c-r9d)	Windows, Linux, macOS, FreeBSD	ARM, Hexagon, MIPS, x86, x86_64	<p>Clang compilers have various use limitations with Coverity products, which are listed in the Coverity Analysis User and Administrator Guide.</p> <p>Coverity Analysis supports the MISRA C:2004 and MISRA C:2012 compliance standards for Clang compilers.</p> <p>MISRA C:2012 rule 1.1 ensures that the analyzed program is compliant with the C standard. The Clang compiler uses different parse warnings and error messages than cov-emit uses. You might encounter minor discrepancies between the enforcement of rule 1.1 by the Clang compiler and cov-emit.</p>
	Kyoto Microcomputer Clang 6.0			
	LLVM Clang 3.7–11.0			
	Qualcomm Hexagon 8.0.10			
	Rynda Clang			

Supported languages, compilers, and frameworks for Coverity Analysis

Compiler	Version	Host OS	Target	Notes
				<p>Host OS support for Linux is limited to x86 and x86_64. Linux on Itanium is not supported.</p> <p>Clang 3.3 with Android NDK r9d is not supported, please use clang 3.4 instead.</p> <p>Clang compilers are supported on FreeBSD only on version 11.3 or later.</p> <p>Deprecation Notice: Support for LLVM Clang 3.7 is deprecated as of 2020.12 and will be removed in a future release.</p>
clang-cl	7.0–9.0	Windows	x86_64	
Cosmic C Cross Compilers	cx6808 4.5.10–4.6.3	Windows	Freescal 68HC08 and HCS08	
	cx6812 4.6i		Freescal 68HC12 and HCS12	
	cxS12x 4.7.7–4.8.9		Freescal HCS12X	
	cx332 4.1l		Freescal MC68332	
	cxstm8 4.3.7		STM8 and STLUX family	
	cxXgate 4.2.4		Freescal XGATE co-processor	
	cx6805 4.2d		Motorola 68HC05	
	cx6811 4.1t		Motorola 68HC11	
	cx6816 4.1r		Motorola 68HC16	
CrossWorks	3.1.1	Windows	MSP430	

Supported languages, compilers, and frameworks for Coverity Analysis

Compiler	Version	Host OS	Target	Notes
	4.0.1		ARM	ISO/IEC TR 18037 fixed point extensions are supported for C (not C++) code on ARM.
Freescale Codewarrior	3.0	Windows	ARM	Codewarrior compilers are supported only for command-line builds. Coverity Analysis does not support the Codewarrior IDE.
	6.4		ColdFire	
	3.0		DS	
			DSC56800E	
	4.0		DSi	
	3.0.5		E68k	
	4.2		EPPC 5xx	
	5.0.32		HC12	
	5.0.25		Microcontrollers (HC08)	
	4.2		MPC55xx	
	10.5		MSC815x	
	10.9		StarCore	
	4.3	Wii		
	3.0	Linux	MSC815x	Deprecation Notice: Support for Freescale Codewarrior 10.5 for MSC815x is deprecated as of 2020.12 and will be removed in a future release.
GNU GCC and G++	FSF GCC 4.0–10.1.0	FreeBSD, Linux, macOS, NetBSD, Solaris, Windows	ARM, Itanium, MIPS, PowerPC, SPARC, x86, x86_64	ISO/IEC TR 18037 fixed point extensions are supported for C (not C++) code.  GNU GCC compilers distributed with Apple Xcode are not supported.  Versions of any of these compilers that are modified to accept non-standard syntax are not supported.

Supported languages, compilers, and frameworks for Coverity Analysis

Compiler	Version	Host OS	Target	Notes
				<p>Only GNU GCC and G++ 4.2.1 are supported on FreeBSD 8.4.</p> <p>Deprecation Notice: Support for GNU GCC and G++ 4.2.1 is deprecated on FreeBSD 8.4 as of 2019.09 and will be removed in a future release.</p>
	Kyoto Microcomputer gcc 6.4.0	Windows	ARM	<p>ISO/IEC TR 18037 fixed point extensions are supported for C (not C++) code.</p> <p>GNU GCC compilers distributed with Apple Xcode are not supported.</p> <p>Versions of any of these compilers that are modified to accept non-standard syntax are not supported.</p> <p>Only GNU GCC and G++ 4.2.1 are supported on FreeBSD 8.4.</p> <p>Deprecation Notice: Support for GNU GCC and G++ 4.2.1 is deprecated on FreeBSD 8.4 as of 2019.09 and will be removed in a future release.</p>

Supported languages, compilers, and frameworks for Coverity Analysis

Compiler	Version	Host OS	Target	Notes
				Deprecation Notice: Support for GNU GCC and G++ 3.0–3.4.6 is deprecated as of 2020.06 and will be removed in a future release.
Green Hills Optimizing C and C++/EC++	4.0.7	Linux	PowerPC	
	4.2.3	Solaris	PPC, MPC83xx, PowerQUICC II PRO	
	4.2.3–2018.1.4	Windows	ARM	
	4.0.2–2012.1		PowerPC	
	5.3		Wii U	
	2015.1–2018.5.5		V850	
	2019.5.5		RH850	
	2012.1		68K/ColdFire	
	2015.1.4–2018.1.4		Linux	
HighTec compiler	4.9.3.0	Windows	Tricore	
IAR Embedded Workbench C/C++	4.40A–8.11	Windows	ARM	
	4.30–6.10		Atmel AVR	
	4.1		Atmel AVR32	
	1.13C–2.30		Dallas/Maxim MAXQ	
	3.2		Freescale HCS12	
	4.10A–5.30		MSP430	
	3.1		National CR16C	
	1.3		Renesas RH850	
	2.10B–2.30		Renesas H8	
	2.21		Renesas RL78	
	2.3		Renesas SuperH	
	3.21D–3.50		Renesas M16C	
	3.21A–3.30		Renesas M32C	
	3.5		Renesas R8C	
	1.4		Renesas R32C	
	4.1		Renesas V850	

Supported languages, compilers, and frameworks for Coverity Analysis

Compiler	Version	Host OS	Target	Notes
	4.12		Renesas RX	
	3.2		Samsung SAM8	
	2.2		STM8	
IBM XLC	13	AIX	Power, PPC	
Intel C++	17.0.0	Linux	x86	
	17.0.0–19.0.3	Windows		
Keil Compilers	RVCT 5.06 for uVision	Windows	ARM	
	8.12–9.52		C51	
	6.11–7.53		C166	
	4.53a–5.55		C251	
Marvell MSA	2.0, 3.1	Windows	MSA	
Microchip Compilers	XC8 1.42–2.20	Windows	8-bit PIC MCUs, 8-bit AVR MCUs	
	XC8 2.20	Linux		
	XC16 1.50	Windows/Linux	16-bit PIC MCUs	
	XC32 2.20	Linux	32-bit PIC MCUs	
Microsoft Visual C++	2013–2019	Windows	x86, x86_64, ARM	<p>Managed C++ and Common Language Runtime (CLR) are not supported. Compilations with switches beginning with "/CLR" will be skipped.</p> <p>Visual C++ 2013 and higher are the only supported compilers for `FxCop` and `cov-import-msvsca`.</p> <p>The compiler version can be determined by running the compiler (`cl`) on the command line, which returns detailed</p>

Supported languages, compilers, and frameworks for Coverity Analysis

Compiler	Version	Host OS	Target	Notes
				<p>information. For example:</p> <pre>&gt; cl</pre> <p>Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 14.00.50727.42 for 80x86</p> <p>Copyright (C) Microsoft Corporation. All rights reserved.</p> <p>usage: cl [ option... ] filename... [ /link linkoption... ]</p>
Panasonic compiler	5.4R3	Windows	MN103S, MN103L	
QNX C/C++	6.3.2–7.0.0	Windows/Linux	ARM	
			Intel XScale	
			MIPS	
			PPC	
			SH-4	
			x86	
Qualcomm Kalimba C compiler	2.06	Windows	x86_64	
Renesas C/C++ Compilers	1.02r01	Windows	R32C	<p>Deprecation Notice: Support for Renesas C/C++ M32R 5.01 is deprecated as of 2020.12 and will be removed in a future release.</p>
	1.03		RH850	
	1.0–3.01		RL78	
	2.72		RX	
	3.47		78k0r	
	5.01		CA850 (NEC V850)	
	5.41		M32R	
			M32C	
	6.02		M16C	
	H8			

Supported languages, compilers, and frameworks for Coverity Analysis

Compiler	Version	Host OS	Target	Notes
			H8S	
	9.01–9.04		SuperH	
	1.31		V850	
SONY PS4 SDK	1.000–6.000	Windows	PS4	Sony PS4 is supported only on Windows 64-bit systems.
Sun (Oracle) CC and cc	Studio8: CC and cc 5.5	Solaris	SPARC, x86	
	Studio9: CC and cc 5.6			
	Studio10: CC and cc 5.7			
	Studio11: CC and cc 5.8			
	Studio12: CC and cc 5.9			
	Studio12.1: CC and cc 5.10			
	Studio12.2: CC and cc 5.11			
	Studio12.4: CC and cc 5.13			
Synopsys MetaWare C and C++ Compilers (mcc/hcac binaries)	I-2013.12	Windows/Linux	ARC 600	ISO/IEC TR 18037 fixed point extensions are supported for C (not C++) code.
			ARC 700	
			ARC EM family	
			ARCompact	
			ARCtangent-A4	
Synopsys MetaWare C and C++ Compilers (ccac binary)	N-2017.09–Q-2019.12	Windows/Linux	ARC EM family	ISO/IEC TR 18037 fixed point extensions are supported for C and C++ code.
			ARC HS family	
TASKING	5.2r1	Windows	ARM Cortex	
	6.0r1–6.2r1		TriCore	

Supported languages, compilers, and frameworks for Coverity Analysis

Compiler	Version	Host OS	Target	Notes
				(not C++) code on TriCore.
Tensilica Xtensa xt-xcc and xt-xtc++ compilers	RA-2006.5	Windows/Linux	x86	
Texas Instruments Code Composer	5.1.0–7.6.0	Windows	TMS320C6x	TI compilers require an environment variable to be set in order for `cov-configure` to properly probe compiler behavior. The environment variable should point to the include directories, and is specific to the compiler (for example, `C6X_C_DIR` for the C6000 compiler).
	4.1.0–4.2.0		TMS320C54x	
	3.2.2–4.3.6		TMS320C55x	
	4.1.0–6.2.8		TMS320C2000	
	4.1.2–4.6.3		TMS470	
	17.3.0–18.12.5		ARM	
	1.2.0	Linux	C7000	
	5.1.0–8.3.1		TMS320C6x	
	4.1.0–6.2.8		TMS320C2000	
	15.12.3		ARM	
Wind River (formerly Diab) C/C++	5.0.x–5.9.x	Windows/Linux	ARM (XSCALE)	
			ColdFire	
			M32R	
			MC68k	
			M*CORE	
			MIPS	
			PPC	
			SH	
			SPARC	
			TriCore	
			x86	
	4.3g	Windows	PowerPC	
XBox One	17.0	Windows	x86_64	Deprecation Notice: Support for XBox One compiler is deprecated as of 2020.12 and will be

Compiler	Version	Host OS	Target	Notes
				removed in a future release.
Xcode (with Clang compiler)	Apple Clang 7.0 (Xcode 7.0–7.2)	macOS	ARM, x86, x86_64	The new build system introduced in Xcode 10 is not supported with Clang compilers. See the section "Building projects that use Xcode 10's new build system" in the "Coverity Analysis User and Administrator Guide" for details on how to work around this issue.  Deprecation Notice: Support for Apple Clang 7.0–7.2 is deprecated as of 2020.12 and will be removed in a future release.
	Apple Clang 7.3 (Xcode 7.3)			
	Apple Clang 8.0 (Xcode 8.0–8.2)			
	Apple Clang 8.1 (Xcode 8.3)			
	Apple Clang 9.0 (Xcode 9.0–9.2)			
	Apple Clang 9.1 (Xcode 9.3–9.4)			
	Apple Clang 10.0 (Xcode 10.0)			
	Apple Clang 11.0 (Xcode 11.0–11.4)			
Apple Clang 12.0 (Xcode 12.0–12.1)				



#### Note

The Platform Builder IDE is supported if the compiler that it is using is supported. Platform Builder is not a compiler.

### 8.1.1. C/C++ Language Support

The following table describes Coverity support for specific proposals incorporated into the C++20 language standard. Values that appear in the **Support level** column, have the following meanings:

- Yes – Fully supported.
- Partial – Partially supported. Most uses cases are expected to work, but there are known issues that customers might encounter.
- No – Not yet supported, either because it is not known whether issues exist, or because there are known issues that severely impair use of the product.

**Table 8.2. C++20 core language features**

<b>Paper</b>	<b>Feature</b>	<b>Support level</b>
P0734	Concepts: Wording Paper, C++ extensions for Concepts	Partial
P0857	Concepts: Fixing functionality gaps in constraints	No
P1084	Concepts: Today's return-type-requirements are insufficient	No
P1141	Concepts: Yet another approach for constrained declarations	No
P0848	Concepts: Conditionally trivial special member functions	No
P1616	Concepts: Using unconstrained template template parameters with constrained templates	No
P1452	Concepts: On the non-uniform semantics of return-type-requirements	No
P1972	Concepts: [US105] Check satisfaction of constraints for non-templates when forming pointer to function	No
P1980	Concepts: [CA096] Declaration matching for non-dependent requires-clauses	Partial
P2092	Concepts: Disambiguating Nested-Requirements	Partial
P2113	Concepts: [CA112] Proposed resolution for 2019 comment CA112	No
P2104	Concepts: Disallow changing concept values	No
P0912	Coroutines: Merge Coroutines TS into C++20 working draft	Partial
P0664	Coroutines: [US065] Apply Coroutines issue 24 from P0664R8	Partial
P1971	Coroutines: [US052] Non-executed return statements in coroutines	Partial

Supported languages, compilers, and frameworks for Coverity Analysis

<b>Paper</b>	<b>Feature</b>	<b>Support level</b>
P2107	Coroutines: [US064] Copy semantics of coroutine parameters	Yes
P1103	Modules: Merging Modules	No
P1766	Modules: Mitigating minor modules maladies	No
P1811	Modules: Relaxing redefinition restrictions for re-exportation robustness	No
P1703	Modules: Recognizing header unit imports requires full preprocessing	No
P1874	Modules: Dynamic Initialization Order of Non-Local Variables in Modules	No
P1979	Modules: [US086] Resolution to US086	No
P1779	Modules: ABI isolation for member functions	No
P1857	Modules: Modules Dependency Discovery	No
P2115	Modules: [US069] Merging of multiple definitions for unnamed unscoped enumerations	No
P1815	Modules: Translation-unit-local entities	No
P2103	Modules: [US033] Allow "import" inside linkage-specification	No
P2109	Modules: [US084] Disallow "export import foo" outside of module interface	No
P0515	Comparisons: Consistent comparison	Partial
P0905	Comparisons: Symmetry for spaceship	Yes
P1120	Comparisons: Consistency improvements for comparisons	No
P1185	Comparisons: <=> != ==	Yes

<b>Paper</b>	<b>Feature</b>	<b>Support level</b>
P1186	Comparisons: Synthesizing three-way comparison for specified comparison category	No
P1630	Comparisons: Spaceship needs a tune-up	No
P1946	Comparisons: Allow defaulting comparisons by value	No
P1959	Comparisons: Remove <code>std::weak_equality</code> and <code>std::strong_equality</code>	Yes
P2002	Comparisons: Fixes wording for defaulted comparison operator functions	No
P2085	Comparisons: Consistent defaulted comparisons	No
P1073	constexpr: Immediate functions	Partial
P1937	constexpr: Fixing inconsistencies between constexpr and consteval functions	Partial
P0595	constexpr: <code>std::is_constant_evaluated()</code>	Yes
P1064	constexpr: constexpr virtual functions	Partial
P1002	constexpr: constexpr try-catch blocks	Partial
P1327	constexpr: constexpr <code>dynamic_cast</code> and polymorphic typeid	Partial
P1330	constexpr: Changing the active member of a union inside constexpr	Partial
P1331	constexpr: Trivial default initialization in constexpr functions	Partial
P1668	constexpr: Unevaluated asm-declaration in constexpr functions	Partial
P0784	constexpr: constexpr container operations	No

Supported languages, compilers, and frameworks for Coverity Analysis

<b>Paper</b>	<b>Feature</b>	<b>Support level</b>
P0859	constexpr: Less eager instantiation of constexpr functions	Partial
P0479	[[likely]] and [[unlikely]]	Partial
P0840	[[no_unique_address]]	Partial
P1301	[[nodiscard]] with message	Partial
P1771	[[nodiscard]] for constructors	Partial
P0329	Designated Initialization	Yes
P0482	char8_t	Yes
P1041	Make char16_t/char32_t string literals be UTF-16/32	No
P1139	Stronger Unicode requirements	No
	Address wording issues related to ISO 10646	
P0409	Allow lambda capture [=, this]	No
P0306	Comma omission and comma deletion	No
P1042	__VA_OPT__ wording clarifications	No
P0428	template-parameter-list for generic lambdas	No
P0683	Default member initializers for bit-fields	No
P0702	Language support for Constructor Template Argument Deduction	No
P0704	Fixing const-qualified pointers to members	No
P0315	Lambdas in unevaluated contexts	No
P0614	init-statements for range-based for	Partial
P0624	Default constructible and assignable stateless lambda	No
P0641	const mismatch with defaulted copy ctor	No
P0692	Access checking on specializations	No
P0846	ADL and function templates that are not visible	No

Supported languages, compilers, and frameworks for Coverity Analysis

<b>Paper</b>	<b>Feature</b>	<b>Support level</b>
P0634	Make typename more optional	No
P0780	Pack expansion in lambda init-capture	No
P2095	Resolve lambda init-capture pack grammar [CWG2378]	No
P0722	Efficient sized delete for variable sized classes	No
P0732	Class types in non-type template parameters	No
P1907	Inconsistencies with non-type template parameters	No
P0528	Atomic Compare-and-Exchange with Padding Bits	No
P0892	Conditional explicit	No
P0941	Integrating feature-test macros	No
P1353	Missing feature-test macros	No
P1008	Prohibit aggregates with use declared constructors	No
P1094	Nested inline namespaces	No
P1236	Signed integers are two's complement	No
P0960	Parenthesized initialization of aggregates	No
P1975	Fixing the wording of parenthesized aggregate-initialization	No
P1091	Extending structured bindings to be more like variable declarations	No
P1381	Reference capture of structured bindings	No
P0388	Permit conversions to arrays of unknown bound	No
P1143	constinit	No
P1099	using enum	No
P1814	Wording for Class Template Argument Deduction for Alias Templates	No

<b>Paper</b>	<b>Feature</b>	<b>Support level</b>
P1816	Wording for class template argument deduction for aggregates	No
P2082	Fixing CTAD for aggregates	No
P0929	Checking for abstract class types	No
P1971	[US053] Mandate the return type for return_void and return_value to be void	No
P0735	Interaction of memory_order_consume with release sequences	No
P1208	Adopt source_location for C++20	No
P1358	[DR 2310] Type completeness and derived-to-base pointer conversions	No
P1971	Core Changes for NB Comments at the Nov 2019 (Belfast) meeting	No
P1908	Reserving Attribute Namespaces for Future Use	No
P0588	[DR] Simplifying implicit lambda capture	No
P0961	[DR] Relaxing the structured bindings customization point finding rules	No
P0962	[DR] Relaxing the range-for loop customization point finding rules	No
P0969	[DR] Allow structured bindings to accessible members	No
P1009	[DR] Array size deduction in new-expressions	No
P1286	[DR] Explicitly defaulted functions with different exception specifications	No
P1825	[DR] Implicit move for more local objects and rvalue references	No
P0593	[DR] Pseudo-dtors end object lifetimes	No
P1957	[DR] Converting from T* to bool should be considered narrowing	No

Paper	Feature	Support level
P0767	Deprecate POD	No
P0806	Deprecate implicit capture of this via [=]	No
P1161	Deprecate comma operator in subscripts	No
P1152	Deprecate some uses of volatile	No

## 8.2. C# compilers

**Table 8.3. Supported C# compilers for static analysis**

Compiler	Compiler version	Language version	Host OS	Notes
Visual Studio	2013, 2015, 2017, 2019	Up to C# 8	Windows 64-bit workstation releases Windows 8.1 and later  Windows 64-bit server releases Windows Server 2012 R2 and later	Visual Studio Express editions are not supported.  Coverity supports analysis of Windows RT applications.
.NET Core	2.1, 3.1		Windows 64-bit workstation releases Windows 8.1 and later  Windows 64-bit server releases Windows Server 2012 R2 and later  Linux 64-bit releases that support .NET Core 3.1	Coverity supports capturing projects only with the Unity Roslyn compiler.  The following KBs must be installed to avoid errors when capturing .NET Core projects or analyzing .NET web applications:  For Windows 8.1 or Windows Server 2012 R2: KB2999226
Unity	2018.3		macOS	

## 8.3. CUDA compilers

**Table 8.4. Supported compilers: Coverity Analysis for CUDA**

Compiler	Compiler version	Host OS	Host compiler	Notes
NVIDIA nvcc	10.1–10.2	Linux (64-bit)	GNU gcc	Coverity supports analysis of C, C++,

Compiler	Compiler version	Host OS	Host compiler	Notes
			Blackberry QNX C/C++ version 7.0	and CUDA source files compiled by the nvcc compiler.
		Windows (64-bit)	Microsoft Visual C++	

## 8.4. Go compilers

Table 8.5. Supported compilers: Coverity Analysis for Go

Compiler	Compiler version	Host OS	Notes
Go	1.13–1.14.x	Linux (64-bit) macOS Windows (64-bit)	<p>Coverity only supports projects that are built with the following commands: go build, go install, go run, and go test. Coverity does not support projects that are built by invoking either go tool compile or gccgo directly.</p> <p>Coverity does not directly recognize custom flags and arguments of go run or go test. In order for Coverity to recognize these custom flags and arguments, you must modify config/templates/go/go_switches.dat.</p> <p>The cov-emit-go command might have dependencies on external tools, depending on the Go code being compiled. Refer to the cov-emit-go command in the Coverity Command Reference for details.</p> <p>Deprecation Notice: Support for Go 1.13 is deprecated as of 2020.12 and will be</p>

Compiler	Compiler version	Host OS	Notes
			removed in a future release.

## 8.5. Java compilers

Coverity Analysis for Java supports the analysis of Java code that is built in two ways:

- [Recommended] Using `cov-build`. For information about this build mode, see the section "Build capture (for compiled languages)" in the *Coverity Analysis 2020.12 User and Administrator Guide*, and see the `cov-build` command documentation in the *Coverity 2020.12 Command Reference*.
- Using `buildless` or `filesystem` capture. For information about these build modes, see the section "The capture" in the *Coverity Analysis 2020.12 User and Administrator Guide*, and see the `cov-emit-java` command documentation in the *Coverity 2020.12 Command Reference*.

**Table 8.6. Supported Java compilers for static analysis**

Host OS	Compiler	Compiler version	Notes
Linux/Windows (64-bit)	OpenJDK	1.8, 11, 14	Deprecation Notice: Support for Open JDK 14 has been deprecated as of 2020.12 and will be removed in a future release.
macOS/Linux/Windows (64-bit)	Sun/Oracle JDK	1.7–1.8, 11, 14	JDK 1.7 is not supported on Windows 10.  Deprecation Notice: Support for Oracle JDK 14 has been deprecated as of 2020.12 and will be removed in a future release.
Linux/Windows (64-bit)	IBM JDK	7, 7.1, 8	

## 8.6. Kotlin compilers

Coverity Analysis for Kotlin supports the analysis of Kotlin code that is built using the `cov-build` command. For information about this build mode, see the section "Build capture (for compiled languages)" in the *Coverity Analysis 2020.12 User and Administrator Guide*, and see the `cov-build` command documentation in the *Coverity 2020.12 Command Reference*.

**Table 8.7. Supported compilers: Coverity Analysis for Kotlin**

Compiler	Compiler version	Host OS	Notes
Kotlin	1.3–1.3.71	Linux (64-bit)	Coverity only supports Kotlin projects that

Compiler	Compiler version	Host OS	Notes
		macOS	are targeted to JVM or Android, not other platforms. For multiplatform projects, Coverity only captures Kotlin source files that are targeted to the supported platforms.
		Windows (64-bit)	

## 8.7. Scala compilers

Table 8.8. Supported compilers: Coverity Analysis for Scala

Compiler	Compiler version	Host OS
Scala	2.12.x	Linux (64-bit)
		macOS
		Windows (64-bit)

## 8.8. Swift compilers

Table 8.9. Supported compilers: Coverity Analysis for Swift

Compiler	IDE	Minimum Host OS	Notes
Swift 5.3.x	Xcode 12.0.x	macOS 10.15.4	<p>Cross-compiler using Mac Catalyst is not supported.</p> <p>Coverity Analysis supports Swift compiler invocations via xcodebuild. Swift Package Manager is not supported.</p> <p>When using xcodebuild - UseModernBuildSystem=NO option must be set to emulate legacy capture behavior.</p>

## 8.9. Visual Basic compilers

**Table 8.10. Supported compilers: Coverity Analysis for Visual Basic**

Compiler	Compiler version	Language version	Host OS	Notes
Visual Studio	2013–2019	Up to Visual Basic 16	Windows 64-bit workstation releases Windows 8.1 and later	Visual Studio Express editions are not supported.  The following KBs must be installed to avoid errors when capturing .NET Core projects or analyzing .NET web applications:  For Windows 8.1 or Windows Server 2012 R2: KB2999226
.NET Core	3.1		Windows 64-bit server releases Windows Server 2012 R2 and later	

## 8.10. Third-party compilers

Coverity Analysis tools can be run with certain third party owned and supported compilers. These third party providers are solely responsible for supporting the compilers in question, and any issues with misconfiguration should be fielded by the their support teams directly.

The following third party compilers are supported:

**Table 8.11. Third-party supported compilers**

Compiler	Version	CIT configuration
MPLAB XC8	1.36	Included in compiler distribution
MPLAB XC16	1.26	
MPLAB XC32	1.42	

## 8.11. Frameworks

Coverity Analysis explicitly supports the following frameworks, libraries, APIs, and other technologies (referred to hereafter as simply frameworks). Coverity Analysis can successfully analyze most frameworks, even if they are not explicitly supported. If your framework is not listed in the following table, don't be overly concerned; just run an analysis and check your results.

**Table 8.12. Frameworks supported by Coverity**

Language	Supported Frameworks
Java	Android SDK

Supported languages, compilers, and frameworks for Coverity Analysis

Language	Supported Frameworks
	Apache Shiro Axis DWR Enterprise Java Beans (EJBs) GWT Hibernate iBatis Java Persistence API (JPA) javax.websocket JAX RS JAX WS JEE JSF/Facelets JSP and JSP Standard Tag Library (JSTL) ReactiveX (RxJava, Reactor) Restlet Spring boot Spring framework Struts Terasoluna Tiles Vert.x WSXmlRpc
C#	ASP.NET ASMX Web Services ASP.NET Core ASP.NET Core MVC ASP.NET Web API ASP.NET MVC ASP.NET Web Forms Razor templates WCF Services
JavaScript/TypeScript	----- Client-Side ----- Angular

Language	Supported Frameworks
	AngularJS
	Backbone
	Bootstrap
	Cordova
	Ember
	HTML5 DOM APIs / Ajax
	JQuery
	Mithril
	React/preact
	socket.io
	Vue
	----- Server-Side -----
	Angular server-side rendering (Express and Hapi engines)
	Express
	Fastify
	Hapi
	Koa
	Mean.io
	Node
	Restify
	SAP XS Classic and Advanced
	Passport
	React server-side rendering (next.js)
	Vue server-side rendering
	----- Template Engines -----
	Consolidate
	doT.js
	EJS
	Haml
	Handlebars
	Hogan

Supported languages, compilers, and frameworks for Coverity Analysis

Language	Supported Frameworks
	Jade koa-views Lodash (templating) Marko Mustache Nunjucks Pug Swig Twig Underscore (templating) Vision  ----- Major Libraries ----- Axios Google Cloud APIs (Storage) Mongoose / MongoDB request Sequelize Underscore / Lodash
PHP	Symphony
Python	Django Flask
Ruby	Ruby on Rails
Kotlin	Android SDK
Go	Echo Gin

---

# Chapter 9. Coverity Connect system tuning, maintenance, and monitoring

## Table of Contents

9.1. Post-installation: what's next? .....	134
9.2. Coverity Connect system and database tuning .....	136
9.3. Monitoring and diagnosing Coverity Connect performance .....	139

This part provides important information about tuning your system and database to ensure that your system is achieving a high level of performance. In addition, this part provides guidelines for maintaining the size of the Coverity Connect database, as well as tools that you can use to diagnose problems in case the performance of your system seems sub-optimal.

### 9.1. Post-installation: what's next?

This section provides a list of tasks for you to perform to bring your entire deployment into production. This section does not provide the "how to" information for each task, but references the documentation that contains the information.

#### 9.1.1. cov-analysis configuration and set-up

The tasks mentioned below are described in full in the *Coverity Analysis 2020.12 User and Administrator Guide* [↗](#).

##### *Generate a compiler configuration*

Before running your first code analysis, you typically generate a configuration of your native compiler.

##### *Enable analysis checkers*

Coverity runs checkers that detect specific types of issues in your source code. For example, the RESOURCE\_LEAK checker finds many types of resource leaks from variables that go out of scope while "owning" a resource, such as freshly allocated memory. Checkers are classified by language and grouped by the types of problems that they detect.

##### *Create custom models to improve analysis results*

A custom model is a piece of source code that is written by a developer to replace the actual implementation of a function or method. Custom models can lead to a more accurate analysis by helping Coverity find more issues and eliminate false positive results.

#### 9.1.2. cov-platform system configuration and set-up

Except for performance tuning and monitoring, the tasks mentioned below are described in full in the *Coverity Platform 2020.12 User and Administrator Guide* [↗](#).

##### *Configuring sign-in options*

Controls user access to Coverity Connect.

##### *Setting up SSL*

Configure Coverity Connect to encrypt communications using SSL.

#### *Connecting to an LDAP database*

The LDAP configuration feature for Coverity Connect allows site administrators to enter this information only once, and all LDAP-compliant applications can use this central resource.

#### *Configuring email notification*

You can enable Coverity Connect to send email to notify developers of new issues or issues that changed triage states.

#### *Creating and/or importing users*

Create or import users into your system. You can also create and organize users in groups.

#### *Setting up role-based access control*

Role-based access control (RBAC) is a feature that restricts system access to authorized Coverity Connect users.

#### *Creating streams and projects*

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.

#### *Creating data stores*

A triage store is a repository for the current and historical triage values of CIDs. 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.

#### *Configuring components*

The 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, and limit access to code and issues

#### *Configuring and managing databases*

Database size can be optimized for performance, and clean-up processes can be scheduled to remove unneeded information. In addition, databases can be backed up and restored. Procedures are provided for backing up and restoring embedded databases in both stand-alone and clustered deployments.

#### *Tuning the embedded database*

See Section 9.2, "Coverity Connect system and database tuning".

#### *Tuning the Coverity Connect server*

See Section 9.2, "Coverity Connect system and database tuning".

#### *Monitoring and diagnosing performance*

See Section 9.3, "Monitoring and diagnosing Coverity Connect performance".

### **9.1.3. The Coverity deployment maturity model**

The deployment maturity model provided by Coverity consulting services is a program that takes your deployment's level of maturity into consideration and works with you to craft a roadmap towards successful adoption and integration of development testing into your development life cycle. Beginning

with a comprehensive understanding of your business goals, use cases and technical environment, Coverity Professional Services will create a deployment plan based on a level-by-level assessment that aligns with your organizational objectives.

## 9.2. Coverity Connect system and database tuning

This chapter provides a series of recommendations that you can implement in your Coverity Connect deployment to help the system run more efficiently and more reliably. This section includes the following:

- Embedded PostgreSQL tuning parameters
- JVM tuning options (Java heap settings)

For more information about the PostgreSQL settings mentioned in this chapter, or for information about tuning your *external* database, see the PostgreSQL documentation at [https://wiki.postgresql.org/wiki/Tuning\\_Your\\_PostgreSQL\\_Server](https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server).

Tuning Coverity Connect and its related components is an iterative process that can vary for different deployments. Each deployment is unique and might require further customization to suit specific requirements. It is recommended to keep detailed records of your tuning and configuration settings in case you need to revert your changes.



### Note

Note that after you make any tuning changes, you will need to restart the Coverity Connect application and database. For more information, see Section 1.4.4, “Stopping and starting Coverity Connect”.

### 9.2.1. PostgreSQL database tuning: embedded database

When you run the 2020.12 Coverity Connect installer, you select a database performance level for a Production or Demo system. The selected level applies changes to both the PostgreSQL database configuration that is embedded with Coverity Connect and the JVM. These changes balance the memory utilization so that approximately 75% of RAM is allocated to the JVM and the remaining 25% is allocated to PostgreSQL and the operating system cache. The allocation of memory in this fashion is to avoid disk swapping by either the application or the database. This new memory allocation strategy biases memory towards the JVM to allow the application to scale under heavier loads without impacting database performance.

You can see the settings stored for your environment and installation by executing the `cov-admin-db tune --read` command.

#### 9.2.1.1. Modifying the database configuration file

To change your PostgreSQL database configuration, you can edit the `postgresql.conf`. This file is in the following location, if you chose the default location during the installation or upgrade:

- `<install_dir_cc>/database/` (Unix systems)
- `<install_dir_cc>\database\` (Windows systems)

If you update the database configuration, you must use the instructions in PostgreSQL Performance Tuning [or](#) follow the direction of Coverity Support. Incorrect configuration updates can prevent Coverity Connect from starting or cause Coverity Connect to perform poorly. Note that you can recover from this error by restoring the configuration file to its original state.

**Table 9.1. postgres.conf settings**

Parameter	Setting and notes
shared_buffers	512MB. (Due to the way PostgreSQL is implemented in Windows, the maximum value for shared_buffers is 512MB. On Windows, PostgreSQL relies heavily on OS caching. This limits the ability to tune memory on this platform and impacts performance negatively.
work_mem	This amount is allocated per session for operations such as sorts (Coverity Connect usually spawns ~30) threshold for disk-based sorts. If it calls for more space, the sort will happen on disk (which is much slower).
effective_io_concurrency	Depends on storage device/system. (16 assuming that you are restoring onto an OWC drive. Allocates approximately 4000 I/Ops per process. Otherwise, it should be equal to the number of drives in your RAID array or 1 for a single drive).
wal_buffers	16MB (for the write-ahead-log. Only requires enough space for logging, not queries).
checkpoint_segments	128 (more segments means less log writing contention) .
track_counts	on/true (stats required for vacuum).
effective_cache_size	1GB (embedded DB) the threshold amount the postgresQL uses to determine if queries can be managed in memory. Does not count toward actual mem usage.
autovacuum	Automates the execution of VACUUM and ANALYZE commands. When set to on, this option checks for tables that have had a large number of inserted, updated, or deleted tuples. These checks use the statistics collection facility; therefore, autovacuum cannot be used unless track_counts <a href="#">is</a> set to true. In the default configuration, autovacuuming is enabled and the related configuration parameters are appropriately set.
autovacuum_max_workers	4 (multiply by maintenance_work_mem to figure out the RAM footprint imposed by concurrent autovacuum processes) counts to the total number of connections (max_connections). large values may skew require PostgreSQL shared_buffers requirement. May require retuning the kernel.shmmax and kernel.shmall parameters based on guidance provided in the pgctl.log.

You must stop and restart Coverity Connect after you change these parameters. Use `cov-im-ctl stop` to stop Coverity Connect and `cov-im-ctl start` to restart it.

### 9.2.1.2. Windows limitations

Limitations for PostgreSQL on Windows include the following:

- `shared_buffer` cannot exceed 1GB.

- `effective_io` cannot be used to improve performance. Enabling this feature under Windows will prevent PostgreSQL from starting.

### 9.2.1.3. Maintaining and backing up the database

After you have deployed the Coverity products and have brought your system into production, proper maintenance and backup must be performed. For more information, see the *Coverity Platform 2020.12 User and Administrator Guide*.

## 9.2.2. JVM settings

This section includes tuning recommendation for the Java Virtual Machine. JVM parameters for Coverity Connect must be placed in the `<install_dir_cc>/config/system.properties` files in the `java_opts_post` attribute. Do not use quotes. This allows the default parameters to be overridden.

To display your current JVM options, use the following command:

```
java -server -XX:+UnlockDiagnosticVMOptions -XX:+PrintFlagsFinal -version
```

During installation, Coverity Platform sets the following JVM parameters, depending on the performance tuning options you chose:

For Production installation tunings:

```
-Xms512m
-Xmx<75% of the Total System Memory>
```

### 9.2.2.1. Required JVM settings

**Table 9.2. Required JVM settings**

Parameter	Setting and notes
<code>-server</code>	Explicitly set the server runtime (as opposed to the client runtime). There are no additional arguments.
<code>-Xms</code>	Set the minimum heap size. Example usage: <code>-Xms512m</code>
<code>-Xmx</code>	Set the maximum heap size. Example usage: <code>-Xmx8g</code>

### 9.2.2.2. Optional JVM settings

The following parameters are optional, but are recommended for optimization:

**Table 9.3. Optional JVM settings**

Parameter	Settings and notes
<code>-xx:+UseCompressedOops</code>	Enable (+) the use of 32-bit ordinary object pointers in 64-bit JREs. Enabling this parameter can provide faster and more efficient 32-bit addressing for

Parameter	Settings and notes
	OOPs at the expense of heap size. Note that the heap will be limited to a maximum of 32Gb.
<code>-XX: -UseGCOverheadLimit</code>	Disable (-) the garbage collection overhead limit. When this parameter is disabled, <code>OutOfMemoryExceptions</code> are suppressed when more than 98% of the total time is spent in garbage collection and less than 2% of the heap is recovered. Note that this might allow the application to hang when the heap is under-provisioned.

### 9.3. Monitoring and diagnosing Coverity Connect performance

This section provides information about tools that you can use to monitor the performance of your system. It is highly recommended that you establish a monitoring schedule or policy within your organization so that you can continually record performance numbers. This way, you can track system performance trends over time to assess the health of the system.

Additionally, this section provides tools that will help you diagnose issues if you suspect that there is a degradation in performance.

#### 9.3.1. Linux monitoring and diagnosis tools

You can use the following commands to monitor and diagnose the performance of your Coverity Connect deployment on Linux:

##### CPU Core Count

The following command will help you monitor CPU usage. Note that this does not count hyperthreading (even if it is enabled):

```
# cat /proc/cpuinfo | egrep "core id|physical id" | tr -d "\n" | sed s/physical/\nphysical/g | grep -v ^$ | sort | uniq | wc -l
```

##### Total RAM

The following command will help you monitor the total RAM usage:

```
# cat /proc/meminfo | grep MemTotal
```

##### Shared Memory

The following commands allow you to view the RAM that is shared between applications on your system.

```
# sysctl -a | grep kernel.shmmax
# sysctl -a | grep kernel.shmall
```

#### 9.3.1.1. Diagnosing performance problems

You can use the Linux/Unix `top` command as a preliminary diagnosis tool for performance-related problems. The following table lists the tools that you can use to diagnose performance problems with `top`.

**Table 9.4. top command options**

Option	Description
<code>iowait</code>	<code>iowait</code> (or just <code>wa</code> ) displays the percentage of time that the CPU (or CPUs) were idle during which the system had an outstanding disk I/O request. Sustained values higher than 5-10% can indicate that an IO bottleneck exists. This command might display misleading values and should be used with other fields to support/deny a particular diagnosis path.
SWAP	SWAP contains statistics about swap space. The SWAP field can be turned on to show swap size for each process. High values coupled with high <code>iowait</code> give a strong indication of thrashing. Processes with high SWAP values need to be tuned in order to decrease the swap space.
Mem	Contains statistics on memory usage. Usage values nearing the physical memory in the machine, coupled with a large swap space, indicate the need for additional RAM. It is recommended that the amount of RAM be at least 25% of the database size.  The database size can be found in Coverity Connect by navigating to Help → About... → Database Size.

For usage information, see the <http://linux.die.net/man/1/top> .

### 9.3.1.2. Diagnosing I/O problems

If an I/O bottleneck is presumed to exist in the system, `iostat` can assist in finding it. The following table lists the recommended options that you can use to diagnose performance problems with `iostat`. Run `iostat` with `-x` to see all fields.

**Table 9.5. top command options**

Option	Description
<code>svctm</code> and <code>await</code>	Displays the number of milliseconds spent servicing requests. High values indicate that the device might be overloaded. Spikes in <code>svctm</code> and <code>await</code> are usually normal and only drives that show persistently slow service times should raise alarm. You can try running <code>iostat -d 1 -x &gt; iostatout.txt</code> to look at persistent data. This will measure I/O every second for as long as it is left to run.

Option	Description
%util	A value of 100% means the device is saturated with requests.
avgqu-sz	Indicates the average queue size. High values along with high svctm, await and %util values near 100% likely means the device is overloaded with read/write requests.

For usage information, see <http://linux.die.net/man/1/iostat> .

### 9.3.2. Windows monitoring and diagnosis tools

You can use the following commands to monitor and diagnose the performance of your Coverity Connect deployment on Windows:

#### CPU Core Count

The following command will help you monitor the CPU usage.

```
# systeminfo | findStr "Processors(s) "
```

#### Total RAM

The following command will help you monitor the total RAM usage:

```
# systeminfo | findStr "Total Physical Memory"
```

#### 9.3.2.1. Using the Performance Monitor to diagnose Coverity Connect issues

The Performance Monitor for Windows shows persistent values for various system statistics including CPU, RAM and I/O Counters. The following example shows an overview of how to use the Windows Performance Monitor. Note that this is just an example. For usage information, see the Performance Monitor documentation at [http://technet.microsoft.com/en-us/library/dd744567\(v=WS.10\).aspx](http://technet.microsoft.com/en-us/library/dd744567(v=WS.10).aspx) .

1. Open the Performance Monitor.
2. Click *Monitoring Tools / Performance Monitor* in the Console Tree (the menu on the left).
3. Click the green plus sign on the top bar of the Performance Monitor screen.
4. Add the counters you want to measure from the list and then click OK.
5. On the right side, select More Actions → New → Data Collector Set. Follow the prompts to create the logs.
6. Click Data Collector Sets → User Defined → <name of the set you just created> in the Console Tree.
7. Click the **Play** button on the top toolbar.
8. Attempt the task which is under-performing or failing to complete (commit, upgrade, viewing issues, triaging, and so forth).
9. Press the **Stop** button.

10. You can now view the data by clicking the **view latest report** button (a small green log book icon) on the top toolbar.

Use the following descriptions to diagnose the performance issue:

**Table 9.6. Defect event processing**

Option	Description
<b>CPU Counters</b>	
Processor \ % Interrupt Time	Measures the time the processor spends receiving and servicing hardware interruptions during specific sample intervals. This counter indicates a possible hardware issue if the value is greater than 15%.
System \ Processor Queue Length	This indicates the number of threads in the processor queue. The server doesn't have enough processor power if the value is more than two times the number of CPUs for an extended period of time.
<b>I/O Counters</b>	
LogicalDisk \ % Free Space	Measures the percentage of free space on the selected logical disk drive. Take note if this falls below 15%, as you risk running out of free space for the OS to store critical files. One solution is to add more disk space.
PhysicalDisk \ Avg. Disk Queue Length	Indicates how many I/O operations are waiting for the hard drive to become available. If the value is larger than the two times the number of spindles, that means the disk itself may be the bottleneck. This is of particular interest when considering a RAID array as a single disk that may become overloaded as a result of the configuration.
PhysicalDisk \ % Idle Time	Measures the percentage of time the disk was idle during the sample interval. If this counter falls below 20 percent for an extended period of time, the disk system is saturated. Replacing the current disk system with a faster disk system may be warranted. Note that this alone is not a very strong indicator of an existing problem.
PhysicalDisk \ Avg. Disk Sec/Read	Measures the average time, in seconds, to read data from the disk. If the number is larger than 10 milliseconds (ms), that means the disk system is experiencing latency when reading from the disk. It is recommended to replace the current disk system with a faster disk system if possible.
PhysicalDisk \ Avg. Disk Sec/Write	Measures the average time, in seconds, it takes to write data to the disk. If the number is larger than

Option	Description
	10 ms, the disk system is experiencing latency when writing to the disk. It is recommended to replace the current disk system with a faster disk system if possible.
<b>Memory Counters</b>	
Memory \ % Committed Bytes in Use	Measures the ratio of Committed Bytes to the Commit Limit—in other words, the amount of virtual memory in use. Ideally this should be zero or very small. This indicates insufficient memory if the number is greater than 80%. It probably indicates thrashing. Check the Pool Paged Bytes and Available Mbytes (see below).
Memory \ Available Mbytes	Measures the amount of physical memory, in megabytes, available for running processes. If this value is less than 5% of the total physical RAM, that means there is insufficient memory, which can increase paging activity. To resolve this problem, you should add more memory.
Memory \ Pool Paged Bytes	Measures the amount of physical memory, in megabytes, available for running processes. If this value is less than 5% of the total physical RAM, that means there is insufficient memory, and that can increase paging activity. To resolve this problem, you should add more memory.
Memory \ Pages per Second	Measures the rate at which pages are read from or written to disk to resolve hard page faults. If the value is greater than 1,000, as a result of excessive paging, tuning the JVM and PostgreSQL settings is likely required.
<b>Network Counters</b>	
Network Interface \ Bytes Total/Sec	Measures the rate at which bytes are sent and received over each network adapter, including framing characters. The network is saturated if you discover that more than 70 percent of the interface is consumed. For example, for a 100-Mbps NIC, the interface consumed is 8.7MB/Sec (100Mbps = (100Mb)*(1MB/8Mb)/Sec = 12.5MB/Sec => 12.5MB/Sec*(0.7) = 8.7MB/Sec).
Network Interface \ Output Queue Length	Measures the length of the output packet queue, in packets. There is network saturation if the value is more than 2 for an extended period of time.

### 9.3.3. System performance reference

The following numbers demonstrate a typical system with very good performance. These numbers are not meant for you to compare to your system, because any number of variables can add or subtract from your build, commit, or analysis times. This is simply provided as an example.

The hardware for this system is as follows:

- Supermicro X9DR3-F/X9DR3-F, BIOS 2.0 01/30/2013
- Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz stepping 07 (8-core single CPU)
- 64GB RAM
- OWC Mercury EXTREME Pro 6G SSD (SATA III mode)

For system configuration settings, the system uses the recommended JVM and PostgreSQL default settings. The memory is optimized for 8GB of total RAM.

#### 9.3.3.1. Build

```
13:13:15 cov-emit phase: 1047 seconds.
```

#### 9.3.3.2. Analysis

```
13:32:28 Analysis summary report:
13:32:28 -----
13:32:28 Files analyzed                : 4425
13:32:28 Total LoC input to cov-analyze  : 2858216
13:32:28 Functions analyzed              : 84963
13:32:28 Paths analyzed                  : 6174509
13:32:28 Time taken by analysis          : 00:19:13
13:32:28 Defect occurrences found        : 4539 Total
13:32:28                                1  ALLOC_FREE_MISMATCH
13:32:28                                50 ARRAY_VS_SINGLETON
13:32:28                                1  ASSERT_SIDE_EFFECT
13:32:28                                154 ATOMICITY
13:32:28                                3  BAD_COMPARE
13:32:28                                9  BAD_FREE
13:32:28                                2  BAD_SIZEOF
13:32:28                                21 BUFFER_SIZE
13:32:28                                48 BUFFER_SIZE_WARNING
13:32:28                                251 CHECKED_RETURN
13:32:28                                100 CONSTANT_EXPRESSION_RESULT
13:32:28                                16 COPY_PASTE_ERROR
13:32:28                                280 DEADCODE
13:32:28                                30 DIVIDE_BY_ZERO
13:32:28                                51 EVALUATION_ORDER
13:32:28                                426 FORWARD_NULL
13:32:28                                15 INCOMPATIBLE_CAST
13:32:28                                6  INFINITE_LOOP
13:32:28                                73 INTEGER_OVERFLOW
```

```

13:32:28 151 LOCK
13:32:28 255 MISSING_BREAK
13:32:28 242 MISSING_LOCK
13:32:28 9 MIXED_ENUMS
13:32:28 25 NEGATIVE_RETURNS
13:32:28 20 NESTING_INDENT_MISMATCH
13:32:28 406 NO_EFFECT
13:32:28 103 NULL_RETURNS
13:32:28 23 ORDER_REVERSAL
13:32:28 28 OVERFLOW_BEFORE_WIDEN
13:32:28 317 OVERRUN
13:32:28 2 PASS_BY_VALUE
13:32:28 10 PW.ASSIGN_WHERE_COMPARE_MEANT
13:32:28 1 PW.BOOLEAN_CONTROLLING_EXPR_IS_CONSTANT
13:32:28 6 PW.BRANCH_PAST_INITIALIZATION
13:32:28 12 PW.CONVERSION_TO_POINTER_ADDS_BITS
13:32:28 8 PW.INCLUDE_RECURSION
13:32:28 2 PW.MISSING_INITIALIZER_ON_CONST
13:32:28 52 PW.PARAMETER_HIDDEN
13:32:28 5 PW.PARAM_SET_BUT_NOT_USED
13:32:28 54 PW.POINTER_CONVERSION_LOSES_BITS
13:32:28 17 PW.SIGNED_ONE_BIT_FIELD
13:32:28 2 PW.SWITCH_SELECTOR_EXPR_IS_CONSTANT
13:32:28 2 PW.USELESS_TYPE_QUALIFIERS
13:32:28 135 RESOURCE_LEAK
13:32:28 2 RETURN_LOCAL
13:32:28 168 REVERSE_INULL
13:32:28 13 REVERSE_NEGATIVE
13:32:28 61 SIGN_EXTENSION
13:32:28 9 SIZECHECK
13:32:28 20 SIZEOF_MISMATCH
13:32:28 1 STACK_USE
13:32:28 5 STRAY_SEMICOLON
13:32:28 8 STRING_NULL
13:32:28 1 STRING_SIZE
13:32:28 3 SWAPPED_ARGUMENTS
13:32:28 355 TAINTED_SCALAR
13:32:28 4 TAINTED_STRING
13:32:28 7 TOCTOU
13:32:28 199 UNINIT
13:32:28 32 UNREACHABLE
13:32:28 114 UNUSED_VALUE
13:32:28 85 USE_AFTER_FREE
13:32:28 14 VARARGS
13:32:28 cov-analyze: 1153 seconds.

```

### 9.3.3.3. Commit

```

13:32:30 2013-05-09 20:32:30 UTC - Committing 6597 file descriptions...
13:32:38 2013-05-09 20:32:38 UTC - Committing 6583 source files...
13:33:37 2013-05-09 20:32:30 UTC - Calculating 6597 cross-reference bundles...
13:33:37 2013-05-09 20:33:37 UTC - Committing 6597 cross-reference bundles...

```

## Coverity Connect system tuning, maintenance, and monitoring

---

```
13:34:42 2013-05-09 20:34:42 UTC - Committing 84963 functions...
13:36:30 2013-05-09 20:36:30 UTC - Committing 4539 defect occurrences...
13:38:44 2013-05-09 20:38:44 UTC - Committing 4 output files...
13:39:07 New snapshot ID 10031 added.
13:39:07 Elapsed time: 00:06:38
```

---

## Appendix A. Coverity Dynamic Analysis for Java

### Table of Contents

A.1. Supported compilers ..... 147

### A.1. Supported compilers

Compiler support varies based on whether you use the `cov-build` command. Coverity Dynamic Analysis supports the analysis of Java code that is built in two ways:

- [Recommended] Using `cov-build`. For information about this build mode, see the section "Build capture (for compiled languages)" in the *Coverity Analysis 2020.12 User and Administrator Guide*, and see the `cov-build` command documentation in the *Coverity 2020.12 Command Reference*.
- Using buildless or filesystem capture. For information about these build modes, see the section "The capture" in the *Coverity Analysis 2020.12 User and Administrator Guide*, and see the `cov-emit-java` command documentation in the *Coverity 2020.12 Command Reference*.

**Table A.1. Supported Java compilers for dynamic analysis**

Host OS	Compiler	Compiler version	Notes
macOS/Linux/Windows (64-bit)	Sun/Oracle JDK	1.7–1.8	Deprecation Notice: Support for all operating systems is deprecated as of 2020.06 and will be removed in a future release.  Coverity does not support Oracle JRockit JDK  JDK 1.7 is not supported on Windows 10.

Refer to the following list for details:

- Using Coverity Dynamic Analysis requires you to either run `cov-build` or use the "Alternative analysis process flow", as described in the *Coverity Analysis 2020.12 User and Administrator Guide* and the `cov-emit-java` command documentation in the *Coverity 2020.12 Command Reference*. For the list of supported compilers, see the section "Supported compilers: Coverity Analysis for Java".
- Coverity Dynamic Analysis Broker runs on the same platforms that support Coverity Analysis for Java (see the section "Supported platforms for Coverity Analysis").
- Coverity Dynamic Analysis Agent is supported on v1.7 Oracle HotSpot JVM.

---

## Appendix B. Coverity Glossary

### Table of Contents

Glossary .....	148
----------------	-----

### Glossary

#### A

Abstract Syntax Tree (AST)	A tree-shaped data structure that represents the structure of concrete input syntax (from source code).
action	In Coverity Connect, a customizable attribute used to triage a CID. Default values are Undecided, Fix Required, Fix Submitted, Modeling Required, and Ignore. Alternative custom values are possible.
Acyclic Path Count	<p>The number of execution paths in a function, with loops counted one time at most. The following assumptions are also made:</p> <ul style="list-style-type: none"><li>• <code>continue</code> breaks out of a loop.</li><li>• <code>while</code> and <code>for</code> loops are executed exactly 0 or 1 time.</li><li>• <code>do...while</code> loops are executed exactly once.</li><li>• <code>goto</code> statements which go to an earlier source location are treated as an exit.</li></ul> <p><i>Acyclic (Statement-only) Path Count</i> adds the following assumptions:</p> <ul style="list-style-type: none"><li>• Paths within expressions are not counted.</li><li>• Multiple case labels at the same statement are counted as a single case.</li></ul>
advanced triage	<p>In Coverity Connect, streams that are associated with the same always share the same triage data and history. For example, if Stream A and Stream B are associated with Triage Store 1, and both streams contain CID 123, the streams will share the triage values (such as a shared <i>Bug</i> classification or a <i>Fix Required</i> action) for that CID, regardless of whether the streams belong to the same project.</p> <p>Advanced triage allows you to select one or more triage stores to update when triaging a CID in a Coverity Connect project. Triage store selection is possible only if the following conditions are true:</p> <ul style="list-style-type: none"><li>• Some streams in the project are associated with one triage store (for example, TS1), and other streams in the project are associated with</li></ul>

another triage store (for example, TS2). In this case, some streams that are associated with TS1 must contain the CID that you are triaging, and some streams that are associated with TS2 must contain that CID.

- You have permission to triage issues in more than one of these triage stores.

In some cases, advanced triage can result in CIDs with issue attributes that are in the Various state in Coverity Connect.

See also, triage.

analysis annotation

A marker in the source code. An analysis annotation is not executable, but modifies the behavior of Coverity Analysis in some way.

Analysis annotations can suppress false positives, indicate sensitive data, and enhance function models.

Each language has its own analysis annotation syntax and set of capabilities. These are not the same as the syntax or capabilities available to the other languages that support annotations.

- For C/C++, an analysis annotation is a comment with special formatting. See code-line annotation and function annotation.
- For C# and Visual Basic, an analysis annotation uses the native C# attribute syntax.
- For Java, an analysis annotation uses the native Java annotation syntax.

Other languages do not support annotations.

annotation

See analysis annotation.

## C

call graph

A graph in which functions are nodes, and the edges are the calls between the functions.

category

See issue category.

checker

A program that traverses paths in your source code to find specific issues in it. Examples of checkers include RACE\_CONDITION, RESOURCE\_LEAK, and INFINITE\_LOOP. For details about checkers, see *Coverity 2020.12 Checker Reference*.

checker category

See issue category.

churn	A measure of change in defect reporting between two Coverity Analysis releases that are separated by one minor release, for example, 6.5.0 and 6.6.0.
CID (Coverity identifier)	See Coverity identifier (CID).
classification	A category that is assigned to a software issue in the database. Built-in classification values are Unclassified, Pending, False Positive, Intentional, and Bug. For Test Advisor issues, classifications include Untested, No Test Needed, and Tested Elsewhere. Issues that are classified as Unclassified, Pending, and Bug are regarded as software issues for the purpose of defect density calculations.
code-line annotation	<p>For C/C++, an analysis annotation that applies to a particular line of code. When it encounters a code-line annotation, the analysis engine skips the defect report that the following line of code would otherwise trigger.</p> <p>By default, an ignored defect is classified as <code>Intentional</code>. See "Models and Annotations in C/C++" in the <i>Coverity Checker Reference</i>.</p> <p>See also function annotation.</p>
code base	A set of related source files.
code coverage	The amount of code that is tested as a percentage of the total amount of code. Code coverage is measured different ways: line coverage, path coverage, statement coverage, decision coverage, condition coverage, and others.
component	A named grouping of source code files. Components allow developers to view only issues in the source files for which they are responsible, for example. In Coverity Connect, these files are specified by a Posix regular expression. See also, component map.
component map	Describes how to map source code files, and the issues contained in the source files, into components.
control flow graph	A graph in which blocks of code without any jumps or jump targets are nodes, and the directed edges are the jumps in the control flow between the blocks. The entry block is where control enters the graph, and the exit block is where the control flow leaves.
Coverity identifier (CID)	An identification number assigned to a software issue. A snapshot contains issue <i>instances</i> (or occurrences), which take place on a specific code path in a specific version of a file. Issue instances, both within a snapshot and across snapshots (even in different streams), are grouped together according to similarity, with the intent that two issues are "similar" if the same source code change would fix them both. These groups of similar issues are given a numeric identifier, the CID. Coverity

Connect associates triage data, such as classification, action, and severity, with the CID (rather than with an individual issue).

CWE (Common Weakness Enumeration)

A community-developed list of software weaknesses, each of which is assigned a number (for example, see CWE-476 at <http://cwe.mitre.org/data/definitions/476.html>). Coverity associates many categories of defects (such as "Null pointer dereferences") with a CWE number.

Coverity Connect

A Web application that allows developers and managers to identify, manage, and fix issues found by Coverity analysis and third-party tools.

## D

data directory

The directory that contains the Coverity Connect database. After analysis, the `cov-commit-defects` command stores defects in this directory. You can use Coverity Connect to view the defects in this directory. See also `intermediate` directory.

deadcode

Code that cannot possibly be executed regardless of what input values are provided to the program.

defect

See `issue`.

deterministic

A characteristic of a function or algorithm that, when given the same input, will always give the same output.

dismissed issue

Issue marked by developers as *Intentional* or *False Positive* in the Triage pane. When such issues are no longer present in the latest snapshot of the code base, they are identified as *absent dismissed*.

domain

A combination of the language that is being analyzed and the type of analysis, either static or dynamic.

dynamic analysis

Analysis of software code by executing the compiled program. See also `static analysis`.

dynamic analysis agent

A JVM agent for Dynamic Analysis that instruments your program to gather runtime evidence of defects.

dynamic analysis stream

A sequential collection of snapshots, which each contain all of the issues that Dynamic Analysis reports during a single invocation of the Dynamic Analysis broker.

## E

event

In Coverity Connect, a software issue is composed of one or more events found by the analysis. Events are useful in illuminating the context of the issue. See also `issue`.

## F

false negative	A defect in the source code that is not found by Coverity Analysis.
false path pruning (FPP)	A technique to ensure that defects are only detected on feasible paths. For example, if a particular path through a method ensures that a given condition is known to be true, then the <code>else</code> branch of an <code>if</code> statement which tests that condition cannot be reached on that path. Any defects found in the <code>else</code> branch would be impossible because they are “on a false path”. Such defects are suppressed by a false path pruner.
false positive	A potential defect that is identified by Coverity Analysis, but that you decide is not a defect. In Coverity Connect, you can dismiss such issues as false positives. In C or C++ source, you might also use code-line annotations to identify such issues as intentional during the source code analysis phase, prior to sending analysis results to Coverity Connect.
fixed issue	Issue from the previous snapshot that is not in the latest snapshot.
fixpoint	The Extend SDK engine notices that the second and subsequent paths through the loop are not significantly different from the first iteration, and stops analyzing the loop. This condition is called a fixpoint of the loop.
flow-insensitive analysis	A checker that is stateless. The abstract syntax trees are not visited in any particular order.
function annotation	For C/C++, an analysis annotation that applies to the definition of a particular function. The annotation either suppresses or enhances the effect of that function's model. See "Models and Annotations in C/C++" in the <i>Coverity Checker Reference</i> .  See also code-line annotation.
function model	A model of a function that is not in the code base that enhances the intermediate representation of the code base that Coverity Analysis uses to more accurately analyze defects.
<b>I</b>	
impact	Term that is intended to indicate the likely urgency of fixing the issue, primarily considering its consequences for software quality and security, but also taking into account the accuracy of the checker. Impact is necessarily probabilistic and subjective, so one should not rely exclusively on it for prioritization.
inspected issue	Issue that has been triaged or fixed by developers.
intermediate directory	A directory that is specified with the <code>--dir</code> option to many commands. The main function of this directory is to write build and analysis results

before they are committed to the Coverity Connect database as a snapshot. Other more specialized commands that support the `--dir` option also write data to or read data from this directory.

The intermediate representation of the build is stored in `<intermediate_directory>/emit` directory, while the analysis results are stored in `<intermediate_directory>/output`. This directory can contain builds and analysis results for multiple languages.

See also `data` directory.

**intermediate representation** The output of the Coverity compiler, which Coverity Analysis uses to run its analysis and check for defects. The intermediate representation of the code is in the intermediate directory.

**interprocedural analysis** An analysis for defects based on the interaction between functions. Coverity Analysis uses call graphs to perform this type of analysis. See also `intraprocedural analysis`.

**intraprocedural analysis** An analysis for defects within a single procedure or function, as opposed to `interprocedural analysis`.

**issue** Coverity Connect displays three types of software issues: quality defects, potential security vulnerabilities, and test policy violations. Some checkers find both quality defects and potential security vulnerabilities, while others focus primarily on one type of issue or another. The Quality, Security, and Test Advisor dashboards in Coverity Connect provide high-level metrics on each type of issue.

Note that this glossary includes additional entries for the various types of issues, for example, an inspected issue, issue category, and so on.

**issue category** A string used to describe the nature of a software issue; sometimes called a "checker category" or simply a "category." The issue pertains to a subcategory of software issue that a checker can report within the context of a given domain.

Examples:

- `Memory - corruptions`
- `Incorrect expression`
- `Integer overflow Insecure data handling`

Impact tables in the *Coverity 2020.12 Checker Reference* list issues found by checkers according to their category and other associated checker properties.

## K

**killpath** For Coverity Analysis for C/C++, a path in a function that aborts program execution. See `<install_dir_sa>/library/generic/common/killpath.c` for the functions that are modeled in the system.

For Coverity Analysis for Java, and similarly for C# and Visual Basic, a modeling primitive used to indicate that execution terminates at this point, which prevents the analysis from continuing down this execution path. It can be used to model a native method that kills the process, like `System.exit`, or to specifically identify an execution path as invalid.

**kind** A string that indicates whether software issues found by a given checker pertain to SECURITY (for security issues), QUALITY (for quality issues), TEST (for issues with developer tests, which are found by Test Advisor), or QUALITY/SECURITY. Some checkers can report quality and security issues. The Coverity Connect UI can use this property to filter and display CIDs.

## L

**latest state** A CID's state in the latest snapshot merged with its state from previous snapshots starting with the snapshot in which its state was 'New'.

**local analysis** Interprocedural analysis on a subset of the code base with Coverity Desktop plugins, in contrast to one with Coverity Analysis, which usually takes place on a remote server.

**local effect** A string serving as a generic event message that explains why the checker reported a defect. The message is based on a subcategory of software issues that the checker can detect. Such strings appear in the Coverity Connect triage pane for a given CID.

Examples:

- May result in a security violation.
- There may be a null pointer exception, or else the comparison against null is unnecessary.

**long description** A string that provides an extended description of a software issue (compare with type). The long description appears in the Coverity Connect triage pane for a given CID. In Coverity Connect, this description is followed by a link to a corresponding CWE, if available.

Examples:

- The called function is unsafe for security related code.

- All paths that lead to this null pointer comparison already dereference the pointer earlier (CWE-476).

## M

model	<p>In Coverity Analysis of the code for a compiled language—such as C, C++, C#, Java, or Visual Basic—a model represents a function in the application source. Models are used for interprocedural analysis.</p> <p>Each model is created as each function is analyzed. The model is an abstraction of the function's behavior at execution time; for example, a model can show which arguments the function dereferences, and whether the function returns a null value.</p> <p>It is possible to write custom models for a code base. Custom models can help improve Coverity's ability to detect certain kinds of bugs. Custom models can also help reduce the incidence of false positives.</p>
modeling primitive	<p>A modeling primitive is used when writing custom models. Each modeling primitive is a function stub: It does not specify any executable code, but when it is used in a custom model it instructs Coverity Analysis how to analyze (or refrain from analyzing) the function being modeled.</p> <p>For example, the C/C++ checker CHECKED_RETURN is associated with the modeling primitive <code>_coverity_always_check_return_()</code>. This primitive tells CHECKED_RETURN to verify that the function being analyzed really does return a value.</p> <p>Some modeling primitives are generic, but most are specific to a particular checker or group of checkers. The set of available modeling primitives varies from language to language.</p>

## N

native build	The normal build process in a software development environment that does not involve Coverity products.
--------------	---

## O

outstanding issue	Issues that are uninspected and unresolved.
outstanding defects count	The sum of security and non-security defects count.
outstanding non-security defects count	The sum of non-security defects count.
outstanding security defects count.	The sum of security defects count.

**owner** User name of the user to whom an issue has been assigned in Coverity Connect. Coverity Connect identifies the owner of issues not yet assigned to a user as *Unassigned*.

## P

**postorder traversal** The recursive visiting of children of a given node in order, and then the visit to the node itself. Left sides of assignments are evaluated after the assignment because the left side becomes the value of the entire assignment expression.

**primitive** In the Java language, elemental data types such as strings and integers are known as *primitive types*. (In the C-language family, such types are typically known as *basic types*).

For the function stubs that can be used when constructing custom models, see modeling primitive.

**project** In Coverity Connect, a specified set of related streams that provide a comprehensive view of issues in a code base.

## R

**resolved issues** Issues that have been fixed or marked by developers as *Intentional* or *False Positive* through the Coverity Connect Triage pane.

**run** In Coverity releases 4.5.x or lower, a grouping of defects committed to the Coverity Connect. Each time defects are inserted into the Coverity Connect using the `cov-commit-defects` command, a new run is created, and the run ID is reported. See also snapshot

## S

**sanitize** To clean or validate tainted data to ensure that the data is valid. Sanitizing tainted data is an important aspect of secure coding practices to eliminate system crashes, corruption, escalation of privileges, or denial of service. See also tainted data.

**severity** In Coverity Connect, a customizable property that can be assigned to CIDs. Default values are Unspecified, Major, Moderate, and Minor. Severities are generally used to specify how critical a defect is.

**sink** Coverity Analysis for C/C++: Any operation or function that must be protected from tainted data. Examples are array subscripting, `system()`, `malloc()`.

Coverity Analysis for Java: Any operation or function that must be protected from tainted data. Examples are array subscripting and the JDBC API `Connection.execute`.

snapshot	<p>A copy of the state of a code base at a certain point during development. Snapshots help to isolate defects that developers introduce during development.</p> <p>Snapshots contain the results of an analysis. A snapshot includes both the issue information and the source code in which the issues were found. Coverity Connect allows you to delete a snapshot in case you committed faulty data, or if you committed data for testing purposes.</p>
snapshot scope	<p>Determines the snapshots from which the CID are listed using the <i>Show</i> and the optional <i>Compared To</i> fields. The show and compare scope is only configurable in the <i>Settings</i> menu in <i>Issues:By Snapshot</i> views and the snapshot information pane in the <i>Snapshots</i> view.</p>
source	<p>An entry point of untrusted data. Examples include environment variables, command line arguments, incoming network data, and source code.</p>
static analysis	<p>Analysis of software code without executing the compiled program. See also dynamic analysis.</p>
status	<p>Describes the state of an issue. Takes one of the following values: <i>New</i>, <i>Triaged</i>, <i>Dismissed</i>, <i>Absent</i> <i>Dismissed</i>, or <i>Fixed</i>.</p>
store	<p>A map from abstract syntax trees to integer values and a sequence of events. This map can be used to implement an abstract interpreter, used in flow-sensitive analysis.</p>
stream	<p>A sequential collection of snapshots. Streams can thereby provide information about software issues over time and at a particular points in development process.</p>

## T

tainted data	<p>Any data that comes to a program as input from a user. The program does not have control over the values of the input, and so before using this data, the program must sanitize the data to eliminate system crashes, corruption, escalation of privileges, or denial of service. See also sanitize.</p>
translation unit	<p>A translation unit is the smallest unit of code that can be compiled separately. What this unit is, depends primarily on the language: For example, a Java translation unit is a single source file, while a C or C++ translation unit is a source file plus all the other files (such as headers) that the source file includes.</p> <p>When Coverity tools capture code to analyze, the resulting intermediate directory contains a collection of translation units. This collection includes source files along with other files and information that form the</p>

context of the compilation. For example, in Java this context includes bytecode files in the class path; in C or C++ this context includes both preprocessor definitions and platform information about the compiler.

**triage** The process of setting the states of an issue in a particular stream, or of issues that occur in multiple streams. These user-defined states reflect items such as how severe the issue is, if it is an expected result (false positive), the action that should be taken for the issue, to whom the issue is assigned, and so forth. These details provide tracking information for your product. Coverity Connect provides a mechanism for you to update this information for individual and multiple issues that exist across one or more streams.

See also advanced triage.

**triage store** A repository for the current and historical triage values of CIDs. 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. Advanced triage allows you to select one or more triage stores to update when triaging a CID in a Coverity Connect project.

See also advanced triage.

**type** A string that typically provides a short description of the root cause or potential effect of a software issue. The description pertains to a subcategory of software issues that the checker can find within the scope of a given domain. Such strings appear at the top of the Coverity Connect triage pane, next to the CID that is associated with the issue. Compare with long description.

Examples:

```
The called function is unsafe for security related code
```

```
Dereference before null check
```

```
Out-of-bounds access
```

```
Evaluation order violation
```

Impact tables in the *Coverity 2020.12 Checker Reference* list issues found by checkers according to their type and other associated checker properties.

## U

**unified issue** An issue that is identical and present in multiple streams. Each instance of an identical, unified issue shares the same CID.

**uninspected issues** Issues that are as yet unclassified in Coverity Connect because they have not been triaged by developers.

unresolved issues

Defects are marked by developers as *Pending* or *Bug* through the Coverity Connect Triage pane. Coverity Connect sometimes refers to these issues as *Outstanding* issues.

## V

various

Coverity Connect uses the term *Various* in two cases:

- When a checker is categorized as both a quality and a security checker. For example, `USE_AFTER_FREE` and `UNINIT` are listed as such in the *Issue Kind* column of the View pane. For details, see the *Coverity 2020.12 Checker Reference*.
- When different instances of the same CID are triaged differently. Within the scope of a project, instances of a given CID that occur in separate streams can have different values for a given triage attribute if the streams are associated with different . For example, you might use advanced triage to classify a CID as a *Bug* in one triage store but retain the default *Unclassified* setting for the CID in another store. In such a case, the View pane of Coverity Connect identifies the project-wide classification of the CID as *Various*.

Note that if all streams share a single triage store, you will never encounter a CID in this triage state.

view

Saved searches for Coverity Connect data in a given project. Typically, these searches are filtered. Coverity Connect displays this output in data tables (located in the Coverity Connect View pane). The columns in these tables can include CIDs, files, snapshots, checker names, dates, and many other types of data.

---

## Appendix C. Legal Notice

### Table of Contents

C.1. Legal Notice .....	160
-------------------------	-----

### C.1. Legal Notice

The information contained in this document, and the Licensed Product provided by Synopsys, are the proprietary and confidential information of Synopsys, Inc. and its affiliates and licensors, and are supplied subject to, and may be used only by Synopsys customers in accordance with the terms and conditions of a license agreement previously accepted by Synopsys and that customer. Synopsys' current standard end user license terms and conditions are contained in the `cov_EULM` files located at `<install_dir>/doc/en/licenses/end_user_license`.

Portions of the product described in this documentation use third-party material. Notices, terms and conditions, and copyrights regarding third party material may be found in the `<install_dir>/doc/en/licenses` directory.

Customer acknowledges that the use of Synopsys Licensed Products may be enabled by authorization keys supplied by Synopsys for a limited licensed period. At the end of this period, the authorization key will expire. You agree not to take any action to work around or override these license restrictions or use the Licensed Products beyond the licensed period. Any attempt to do so will be considered an infringement of intellectual property rights that may be subject to legal action.

If Synopsys has authorized you, either in this documentation or pursuant to a separate mutually accepted license agreement, to distribute Java source that contains Synopsys annotations, then your distribution should include Synopsys' `analysis_install_dir/library/annotations.jar` to ensure a clean compilation. This `annotations.jar` file contains proprietary intellectual property owned by Synopsys. Synopsys customers with a valid license to Synopsys' Licensed Products are permitted to distribute this JAR file with source that has been analyzed by Synopsys' Licensed Products consistent with the terms of such valid license issued by Synopsys. Any authorized distribution must include the following copyright notice: **Copyright © 2020 Synopsys, Inc. All rights reserved worldwide.**

**U.S. GOVERNMENT RESTRICTED RIGHTS:** The Software and associated documentation are provided with Restricted Rights. Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in subparagraph (c)(1) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of Commercial Computer Software – Restricted Rights at 48 CFR 52.227-19, as applicable.

The Manufacturer is: Synopsys, Inc. 690 E. Middlefield Road, Mountain View, California 94043.

The Licensed Product known as Coverity is protected by multiple patents and patents pending, including U.S. Patent No. 7,340,726.

#### Trademark Statement

Coverity and the Coverity logo are trademarks or registered trademarks of Synopsys, Inc. in the U.S. and other countries. Synopsys' trademarks may be used publicly only with permission from

## Legal Notice

---

Synopsys. Fair use of Synopsys' trademarks in advertising and promotion of Synopsys' Licensed Products requires proper acknowledgement.

Microsoft, Visual Studio, and Visual C# are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Microsoft Research Detours Package, Version 3.0.

Copyright © Microsoft Corporation. All rights reserved.

Oracle and Java are registered trademarks of Oracle and/or affiliates. Other names may be trademarks of their respective owners.

"MISRA", "MISRA C" and the MISRA triangle logo are registered trademarks of MISRA Ltd, held on behalf of the MISRA Consortium. © MIRA Ltd, 1998 - 2013. All rights reserved. The name FindBugs and the FindBugs logo are trademarked by The University of Maryland.

Other names and brands may be claimed as the property of others.

This Licensed Product contains open source or community source software ("**Open Source Software**") provided under separate license terms (the "**Open Source License Terms**"), as described in the applicable license agreement under which this Licensed Product is licensed ("**Agreement**"). The applicable Open Source License Terms are identified in a directory named `licenses` provided with the delivery of this Licensed Product. For all Open Source Software subject to the terms of an LGPL license, Customer may contact Synopsys at `software-integrity-support@synopsys.com` and Synopsys will comply with the terms of the LGPL by delivering to Customer the applicable requested Open Source Software package, and any modifications to such Open Source Software package, in source format, under the applicable LGPL license. Any Open Source Software subject to the terms and conditions of the GPLv3 license as its Open Source License Terms that is provided with this Licensed Product is provided as a mere aggregation of GPL code with Synopsys' proprietary code, pursuant to Section 5 of GPLv3. Such Open Source Software is a self-contained program separate and apart from the Synopsys code that does not interact with the Synopsys proprietary code. Accordingly, the GPL code and the Synopsys proprietary code that make up this Licensed Product co-exist on the same media, but do not operate together. Customer may contact Synopsys at `software-integrity-support@synopsys.com` and Synopsys will comply with the terms of the GPL by delivering to Customer the applicable requested Open Source Software package in source code format, in accordance with the terms and conditions of the GPLv3 license. No Synopsys proprietary code that Synopsys chooses to provide to Customer will be provided in source code form; it will be provided in executable form only. Any Customer changes to the Licensed Product (including the Open Source Software) will void all Synopsys obligations under the Agreement, including but not limited to warranty, maintenance services and infringement indemnity obligations.

The Cobertura package, licensed under the GPLv2, has been modified as of release 7.0.3. The package is a self-contained program, separate and apart from Synopsys code that does not interact with the Synopsys proprietary code. The Cobertura package and the Synopsys proprietary code co-exist on the same media, but do not operate together. Customer may contact Synopsys at `software-integrity-support@synopsys.com` and Synopsys will comply with the terms of the GPL by delivering to Customer the applicable requested open source package in source format, under the GPLv2 license. Any Synopsys proprietary code that Synopsys chooses to provide to Customer upon its request will be provided in object form only. Any changes to the Licensed Product will void all

## Legal Notice

---

Coverity obligations under the Agreement, including but not limited to warranty, maintenance services and infringement indemnity obligations. If Customer does not have the modified Cobertura package, Synopsys recommends to use the JaCoCo package instead.

For information about using JaCoCo, see the description for `cov-build --java-coverage` in the *Command Reference*.

### LLVM/Clang subproject

Copyright © All rights reserved. Developed by: LLVM Team, University of Illinois at Urbana-Champaign (<http://llvm.org/>). Permission is hereby granted, free of charge, to any person obtaining a copy of LLVM/Clang and associated documentation files ("Clang"), to deal with Clang without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of Clang, and to permit persons to whom Clang is furnished to do so, subject to the following conditions: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimers. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimers in the documentation and/or other materials provided with the distribution. Neither the name of the University of Illinois at Urbana-Champaign, nor the names of its contributors may be used to endorse or promote products derived from Clang without specific prior written permission.

CLANG IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH CLANG OR THE USE OR OTHER DEALINGS WITH CLANG.

### Rackspace Threading Library (2.0)

Copyright © Rackspace, US Inc. All rights reserved. Licensed under the Apache License, Version 2.0 (the "License"); you may not use these files except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>.

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

### SIL Open Font Library subproject

Copyright © 2020 Synopsys Inc. All rights reserved worldwide. ([www.synopsys.com](http://www.synopsys.com)), with Reserved Font Name fa-gear, fa-info-circle, fa-question.

This Font Software is licensed under the SIL Open Font License, Version 1.1. This license is available with a FAQ at <http://scripts.sil.org/OFL>.

### Apache Software License, Version 1.1

Copyright © 1999-2003 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

## Legal Notice

---

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgement: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)."

Alternately, this acknowledgement may appear in the software itself, if and wherever such third-party acknowledgements normally appear.

4. The names "The Jakarta Project", "Commons", and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [apache@apache.org](mailto:apache@apache.org).
5. Products derived from this software may not be called "Apache" nor may "Apache" appear in their names without prior written permission of the Apache Group.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Apache License Version 2.0, January 2004 <http://www.apache.org/licenses/>  
Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at: <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Results of analysis from Coverity and Test Advisor represent the results of analysis as of the date and time that the analysis was conducted. The results represent an assessment of the errors, weaknesses and vulnerabilities that can be detected by the analysis, and do not state or infer that no other errors, weaknesses or vulnerabilities exist in the software analyzed. Synopsys does NOT guarantee that all errors, weakness or vulnerabilities will be discovered or detected or that such errors, weaknesses or vulnerabilities are discoverable or detectable.

SYNOPSYS AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, CONDITIONS AND REPRESENTATIONS, EXPRESS, IMPLIED OR STATUTORY, INCLUDING THOSE RELATED

## Legal Notice

---

TO MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, ACCURACY OR COMPLETENESS OF RESULTS, CONFORMANCE WITH DESCRIPTION, AND NON-INFRINGEMENT. SYNOPSIS AND ITS SUPPLIERS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES, CONDITIONS AND REPRESENTATIONS ARISING OUT OF COURSE OF DEALING, USAGE OR TRADE.