Sonatype CLM Enforcement Points - IDE

Contents

1 Introduction **1**

2 Installing Sonatype CLM for Eclipse **2**

3 Configuring Sonatype CLM for Eclipse **5**

4 Using the Component Info View **9**

4.1 Overview 9

4.2 Filtering the Component List 14

4.3 Searching for Component Usages 15

4.4 Inspecting Component Details 15

5 Migrating to Different Component Versions **17**

6 Using Sonatype CLM with Other IDEs **21**

6.1 Maven Plugin Setup 21

6.2 IntelliJ IDEA 22

6.3 Netbeans IDE 24

List of Figures

2.1 Eclipse Dialog to Install New Software with Sonatype CLM for Eclipse 3

3.1 Activating the Component Info View of Sonatype CLM for Eclipse 6

3.2 Warning after initial installation 6

3.3 Sonatype CLM for Eclipse Configuration Dialog 7

4.1 Example Component Info View 9

4.2 Details for a Component in the Component Info View 11

4.3 Properties of a Component for a Version Range 13

4.4 Filter Dialog for the Component Info View 14

4.5 Example Component Details Display 16

5.1 Migrating to a Newer Component Version 18

5.2 Applying a Dependency Version Upgrade 19

5.3 Selecting Dependency Version or Property Upgrade 19

5.4 Applying a Property Upgrade 20

6.1 Creating a Maven Run Configuration for a CLM Evaluation in IntelliJ 23

6.2 Maven Projects View with the CLM Evalulation Run Configuration in IntelliJ 23

6.3 CLM Maven Plugin Output in the Run Console in IntelliJ 24

6.4 Project View with the `pom.xml` in Netbeans 25

6.5 Maven Goal Setup for a CLM Evaluation in Netbeans 25

6.6 CLM Maven Plugin Output in the Output Window in Netbeans 26

Chapter 1

Introduction

This guide shows you how to install and use Sonatype CLM for IDE to analyze the components used by your software development project and take action to resolve any issues you discover.

Chapter 2

Installing Sonatype CLM for Eclipse

Often only called Eclipse, the Eclipse IDE is a very powerful, open source IDE written mostly in Java and managed by the Eclipse Foundation. It can be used for development in a number of languages, and is the most widely used IDE for Java development. It features a powerful plug-in system that allows you to customize the IDE, with features that support a large number software development-related tasks including localization options, version control systems, and myriad of other tasks.

Sonatype CLM for Eclipse requires Eclipse 3.7 or higher. In addition it requires the Maven integration for Eclipse m2e to be installed. Most Eclipse download bundles related to Java development include this integration. If your Eclipse version does not have m2e installed, you need to install it before installing Sonatype CLM for Eclipse following the instructions on the m2e site.

Sonatype CLM for Eclipse can be installed by adding a new software repository. Navigate to the *Help* menu and select *Install New Software*. Press the *Add* button in the dialog displayed in Figure 2.1 and create a new repository with the *Location* set to the URL for Sonatype CLM for Eclipse releases from URL for the Sonatype CLM for Eclipse repository and a *Name* of your choice. Once you press *OK* a list of available releases is downloaded and an entry for the latest version of Sonatype CLM for Eclipse is displayed. Uncheck the item *Show only the latest versions of available software*, if you need to install an older release. Figure 2.1 shows a list of releases available.

Warning
This guide assumes an installation of the currently released version of the Sonatype CLM for IDE plugin and the compatible Sonatype CLM Server. Before picking a version please verify compatibility.

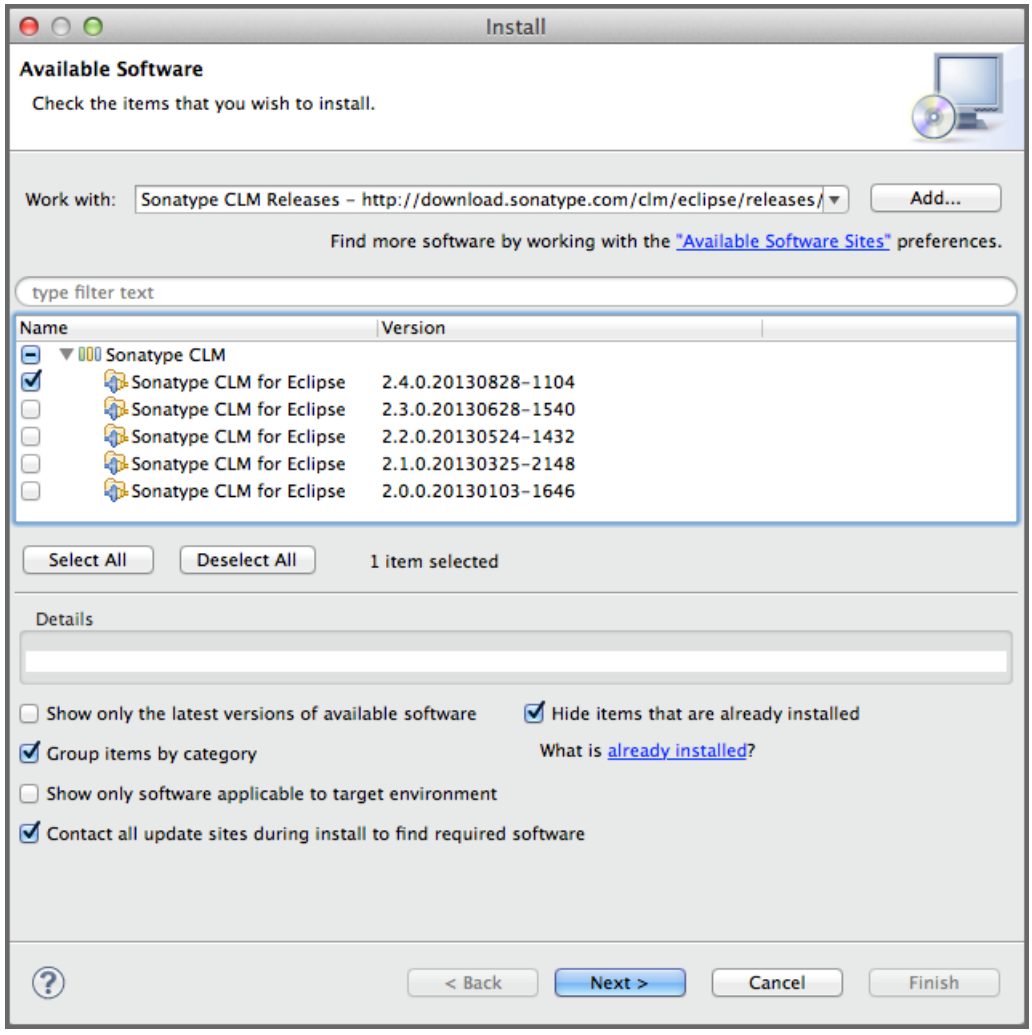

Figure 2.1: Eclipse Dialog to Install New Software with Sonatype CLM for Eclipse

URL for the Sonatype CLM for Eclipse repository

```
http://download.sonatype.com/clm/eclipse/releases/
```

Select the version of Sonatype CLM for Eclipse you would like to install and press *Next*>, proceed through accepting the end user license agreement and restart Eclipse to complete the installation.

Chapter 3

Configuring Sonatype CLM for Eclipse

After successful installation of Sonatype CLM for Eclipse, you will be able to choose to show the Sonatype CLM view displayed in Figure 3.1.

To access this view:

1. Choose the *Window* menu and select *Other* in the *Show View* submenu.
2. Locate the *Sonatype CLM* section with *Component Info* as shown in Figure 3.1.
3. Select it and press *OK* and the view will appear in your IDE.

Tip
By typing "Compo" in the filter input, Component Info is automatically highlighted.

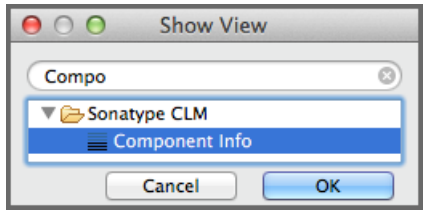

Figure 3.1: Activating the Component Info View of Sonatype CLM for Eclipse

Once the view is displayed, a warning will appear. This is because the you need to point Eclipse at your Sonatype CLM Server.

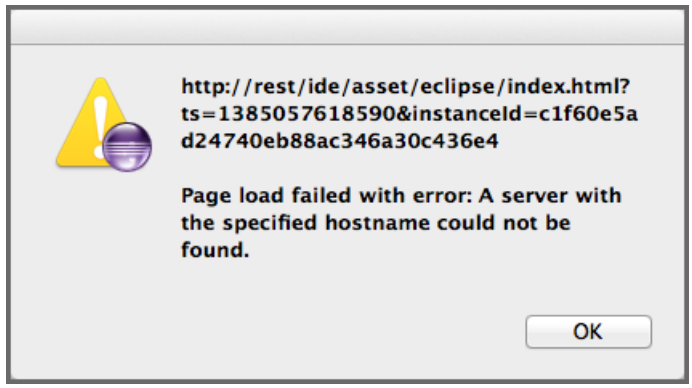

Figure 3.2: Warning after initial installation

To configure the Sonatype CLM for Eclipse plugin, simply press the *Configure* button in the top right-hand side of the component view.

Once in Sonatype CLM for Eclipse Configuration area, there are a number of parameters you will need to complete before you can review data from Sonatype CLM. These are covered below.

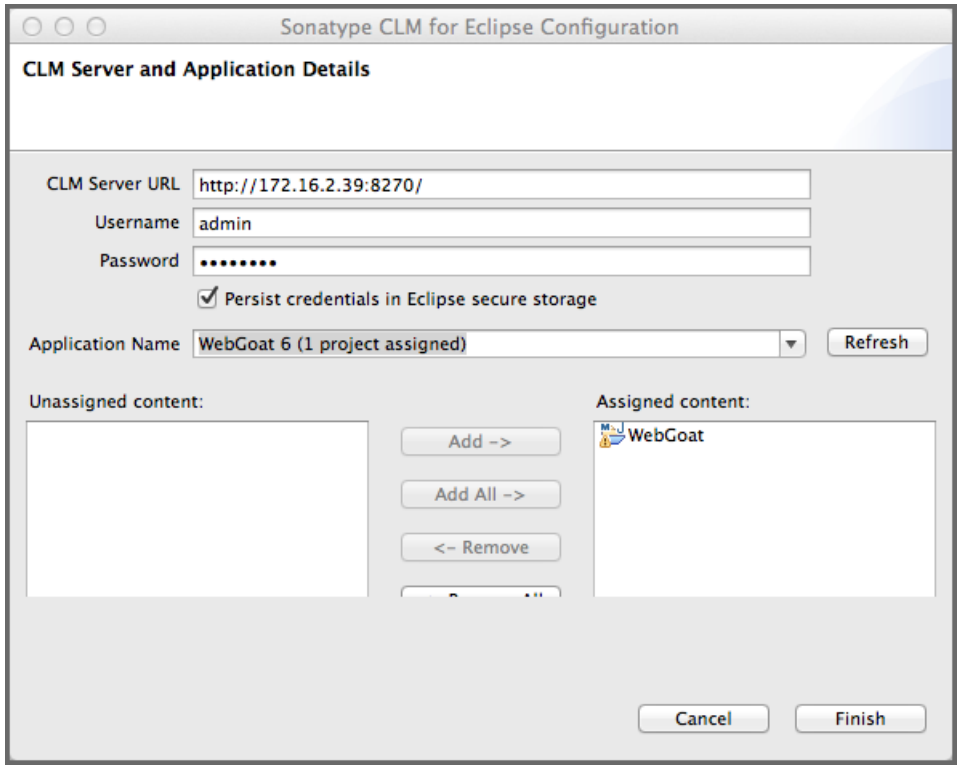

Figure 3.3: Sonatype CLM for Eclipse Configuration Dialog

CLM Server URL

The *CLM Server URL* input field has to be configured with the URL of your Sonatype CLM server.

Username and Password

This is the username and password your Sonatype CLM Administrator has assigned you. In many cases this will simply be your single sign on credentials (e.g. LDAP), though it may also be a unique username. Again, your administrator will advise you of this login information.

Note

Selecting the option to persist credentials in Eclipse secure storage will reuse your credentials after a restart. If this is not selected you will need to reenter your credentials after a restart.

Application Name

The *Application Name* is the application which has been configured in the CLM server for you. This should match the common name you associate with the application. If you don't see a name you recognize, contact your Sonatype CLM Administrator.

Note

The drop down will display a list of all available applications after pressing the *Refresh* button.

Assigned vs. Unassigned Content

After selecting an application name that represents a collection of policies configured in your CLM server, you can determine the Eclipse projects that should be analyzed. The list on the left titled *Unassigned content* contains all projects in your current Eclipse workspace that have not been assigned to a Sonatype CLM Application. Select a project from that list and add it to the *Assigned content* list on the right by clicking the *Add* button. This will add the project to the component analysis via the CLM server. In order to perform an analysis, the project needs to be open. To select multiple projects use the Shift and Control keys, and then click the *Add* button. The *Add All*, *Remove* and *Remove All* buttons help you to control the projects to analyze for different analysis sessions.

Note

Projects can, at most, be assigned to a single application.

With a finished selection of the projects you want to analyze, press the *Finish* button and wait for the component list to be displayed in the view. Chapter 4 documents how to inspect the results of the analysis and further features available from this information.

Tip

Only open projects will be taken into account as part of the component analysis.

Chapter 4

Using the Component Info View

4.1 Overview

Once configured and the component analysis is completed a component view will look similar to the example displayed in Figure 4.1. It's important to note, that the list of components will reflect an analysis of everything on the build path. For Maven projects, we exclude optional dependencies and dependencies in test, system and provided scope.

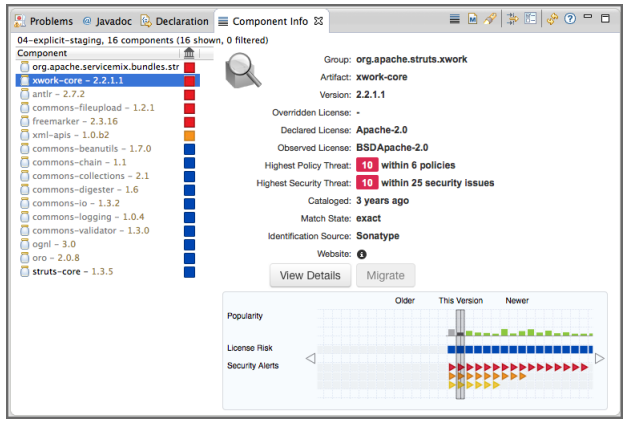

Figure 4.1: Example Component Info View

The top left-hand corner of the Sonatype CLM for Eclipse *Component Info* view displays either the number of projects currently being examined in the view, or the name of the specific project. Next to that, the number of components found, and the number of components shown in the list is displayed.

The top right-hand corner provides a number of buttons to access the following features of Sonatype CLM for Eclipse:

Open Component Details
Opens another window with more details about the selected component including policy violations, license analysis and security issues.

Open POM
Opens the Maven pom.xml file of the selected component from the list in the Maven POM Editor.

Locate Declarations
Starts a search, that displays all usages of a selected component in the projects currently examined as documented in Section 4.3.

Filter
Brings up the filter selection, that lets you narrow down the number of components visible in the view as documented in. Section 4.2.

Configure
Activates the configuration dialog for the component analysis.

Refresh
Refreshes the component list and analysis results.

Show information about the plugin
Displays the Sonatype CLM for Eclipse support pages in an external browser.

Minimize
Minimize the view.

Maximize
Maximize the view.

The left-hand side of the view contains the list of components found in the project and identified by their artifact identifier and version number. A color indicator beside the components signals potential policy

violations. The right-hand side of the view displays the details of the selected component from the list on the left.

Tip
You may notice some components are black or gray. This indicates components you have included (black) in your application, versus components that are included via a transitive dependency (gray).

By clicking on the list header on the left, the list can be ordered by the threat level of the policy a component has violated. In cases where there is no violation, the threat is simply light blue.

When you select a specific component in the list, the details, various properties, and a visualization of the different versions is displayed to the right of the list.

Tip
Depending on your screen size, the visual display may be shown below the component list. Try adjusting your screen size, or adjusting the panel.

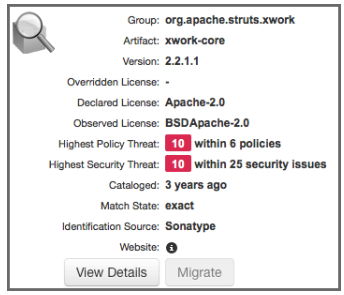

Figure 4.2: Details for a Component in the Component Info View

The details of a specific component as displayed in Figure 4.2 include properties about the component and provide access to further features:

Group
The Maven `groupId` the component was published with. In many cases this is equivalent with the reverse domain name of the organization responsible for the deployment or running the project.

Artifact
The Maven `artifactId` of the component acts as a short and ideally descriptive name.

Version
The Maven `version` of the component. A version string ending in `-SNAPSHOT` signifies a transient, in development version, any other version is a release version.

Overridden License
The value of a license override configured in your Sonatype CLM server.

Declared License
The software license declared by the developer of the project, which in some cases, is identified during research by Sonatype, or directly from the Maven POM file.

Observed License
The licenses found by the Sonatype CLM server in a source code analysis.

Highest Policy Threat
The highest threat level policy that has been violated, as well as the total number of violations.

Highest Security Threat
The highest security threat level as well as the number of issues found with the respective level.

Patch Available
This is a future feature that will provide details in instances where a patch is available. Patches will be provided and verified by Sonatype.

Cataloged
The age of the component in the Central Repository.

Identification Source
The catalog in which a component identification match was found. This includes either a match made by Sonatype (e.g. the catalog of the Central Repository), or a match made manually (i.e. through the Sonatype CLM claiming process).

Website
If available, an information icon providing a link to the project is displayed.

View Details
Press this button to display the details view for the selected component as detailed in Section 4.4.

Migrate
Press this button to start a project refactoring that allows you to change all usages of the current component to a different version as documented in Chapter 5.

Custom Metadata
This is a future feature that will allow you to display all custom metadata tags assigned to the component.

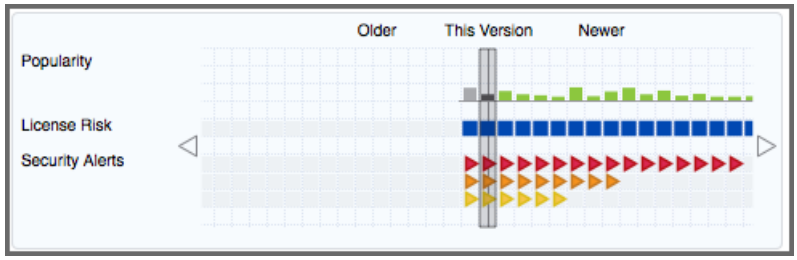

Figure 4.3: Properties of a Component for a Version Range

The visualization chart displayed in Figure 4.2 shows a number of properties for different, available versions of the selected component. Older versions are displayed on the left and newer versions on the right. Click on any section in the visualization, and all information for that particular version will be highlighted, with the specific version number at the bottom. In addition, the details for that version of the component will display in the left-hand list of properties. Arrows to the left and right of the visualization allow you to view the full range of available versions.

The properties displayed include:

Popularity
: the relative popularity of a version as compared to all other component versions

License Conflict
: displays an indicator, if the observed licenses in the component are creating a legal conflict, e.g. GPL V2 and Apache V2 are not compatible for distribution of one component

License Risk
: the risk posed based on what has been set within the license threat groups. While defaults are available, these are configurable via the Sonatype CLM Server.

Security Alerts
: indicators for the severity of security alerts affecting the component version

You will likely notice a number of colors within the visualization chart. The value for each of these colors is as follows:

For Popularity

- Grey for any versions older than the current version.

- Green for newer, but within the same major version of the component.
- Blue for newer component versions, but with a greater major version than the current component.

For License and Security

- Blue - no security or license risk
- Yellow - minor security or license risk
- Orange - medium security or license risk
- Red - severe security or license risk

4.2 Filtering the Component List

The list of components found in the analysis and displayed in the component info view can be configured by pressing the *Filter* button. The filter dialog, displayed in Figure 4.4, allows you to narrow down the components shown.

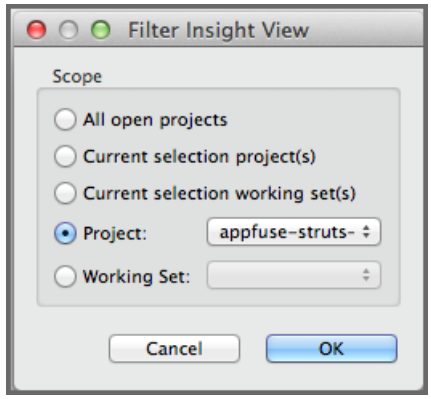

Figure 4.4: Filter Dialog for the Component Info View

The *Scope* setting determines, which projects' components are displayed in the list:

All open projects
include all the components, from all open projects

Current selection project(s)
include the components from the project currently selected in the package explorer

Current selection working set(s)
include the components from all the projects in the working set currently selected in the package explorer

Project
include the components from the project selected in the drop down

Working Set
include the components from all the projects in the working set selected in the drop down

4.3 Searching for Component Usages

Once you have selected a specific component in the list on the left of the component info view, Sonatype CLM can determine in which projects the component is used. After pressing the *Locate Declarations* button, and once the search has completed, you will see the results in a tree view of projects and project pom.xml files, all displayed in the *Search* window.

Inspecting this list can help you assess the impact of a potential upgrade to a new component version. Further detail is documented in Chapter 5. Looking at the found projects, you can potentially remove wrong declarations, determine side effects from transitive dependencies, or find out why this component shows up as dependency at all.

4.4 Inspecting Component Details

Press the *Open Component Details* button in order to access the details about policy violations, license analysis and security issues for a specific component selected in the list. Figure 4.5 displays an example details view.

Policy Violations

Policy	Constraint	Summary
Security-High	CVSS >=7 and <10	Found 2 Security Vulnerabilities with Severity >= 7
		Found 2 Security Vulnerabilities with Severity < 10
		Found 2 Security Vulnerabilities with Status OPEN
Architecture-Quality	Old	Age was 7 years, 5 months and 24 days

License Analysis

Threat Level	Declared License(s)	Observed License(s)
Liberal	BSD-3-Clause	BSD

Security Issues

Threat Level	Problem Code	Status	Summary
9	CVE-2007-4575	Open	HSQLDB before 1.8.0.9, as used in OpenOffice.org (OOo) 2 before 2.3.1, allows user-assisted remote attackers to execute arbitrary Java code via crafted database documents, related to "exposing static java methods."
	OSVDB-40548	Open	OpenOffice.org (OOo) HSQLDB Database Document Handling Unspecified Arbitrary Java Code Execution

Figure 4.5: Example Component Details Display

The *Policy Violations* section in the top contains a list of all the policies that have been violated by the component.

The *License Analysis* section contains the *Threat Levels* posed by the licenses declared for each component, as well as those that have been observed in the source code.

The *Security Issues* section below contains the list of issues found. They are sorted from higher to lower risk, with each issue detailed by a *Threat Level* ranging from 0 to 10, or potentially with the value *Unscored* and a descriptive text in the *Summary* column. In addition, links to the security vulnerability database entry are added as links in the *Problem Code* column.

Chapter 5

Migrating to Different Component Versions

If you determine that a component upgrade is required to avoid a security or license issue or a policy violation, after reviewing your component usage, Sonatype CLM for Eclipse can be used to assist you in the necessary refactoring.

NOTE
This feature relies on the project being a Maven project.

The first step to start the migration is to select a newer version for the component in the visualization chart. An example is displayed in Figure 5.1.

Figure 5.1: Migrating to a Newer Component Version

Once you have selected a different version than the one currently used, the *Migrate* button will become active. Pressing the button opens a dialog that assists you in the migration to the newer component. The complexity of this task varies considerably from project setup to project setup. The migration wizard is able to detect circumstances such as the component being a transitive dependency or versions managed in a property.

The simplest flow is when a dependency version can be applied, and the result is a single dialog like the one displayed in Figure 5.2.

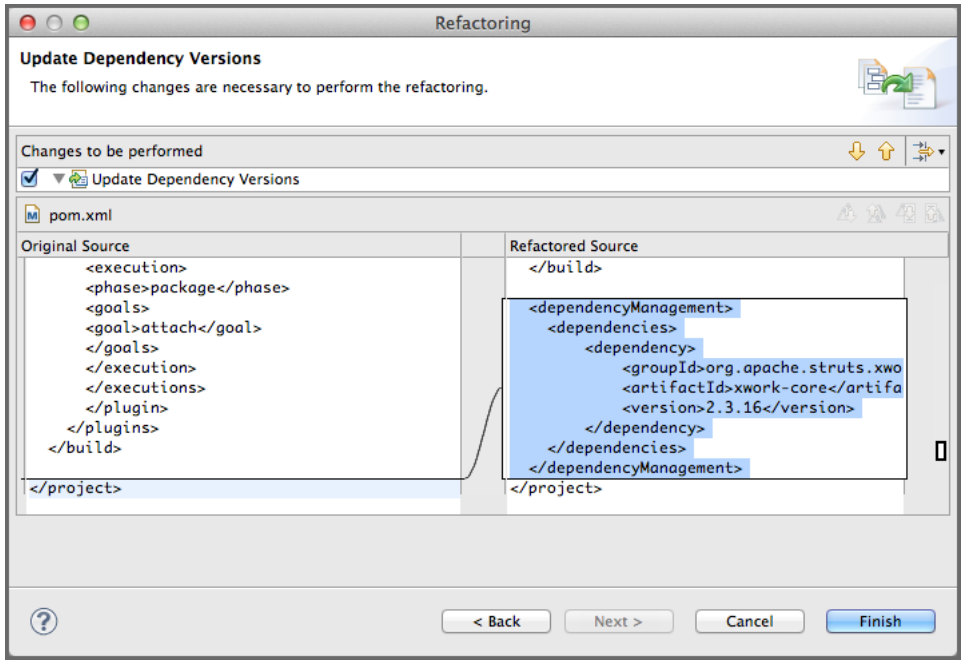

Figure 5.2: Applying a Dependency Version Upgrade

If the version is managed in a property, the initial screen from Figure 5.3 allows you to select if you want to continue with a property upgrade, or perform a replacing version upgrade.

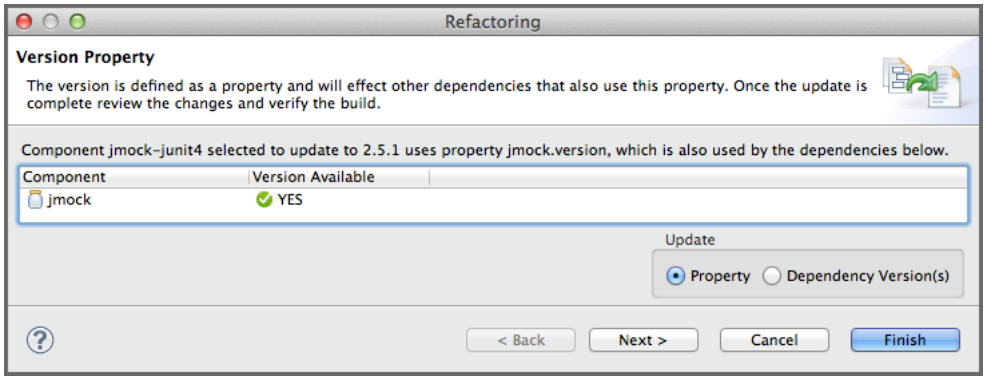

Figure 5.3: Selecting Dependency Version or Property Upgrade

Once you have selected to perform a property upgrade, you will be able to apply it in the next screen, *Refactoring*, visible in Figure 5.4.

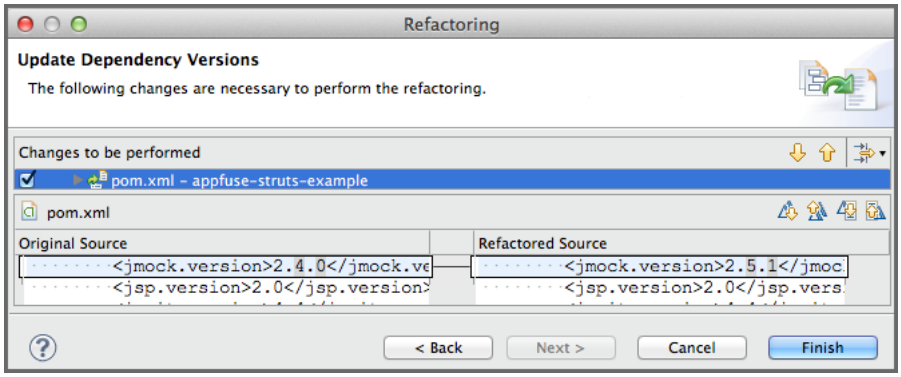

Figure 5.4: Applying a Property Upgrade

The *Refactoring* screen features navigation tools allowing you to view all potential changes in the dialog, and step through them one by one before deciding to continue.

After you have completed the refactoring of your project files, you should perform a full build, as well as a thorough test, to determine that you can proceed with the new version in your development.

Typically smaller version changes will have a higher chance of working without any major refactorings, or adaptations, of your code base and projects, while larger version changes potentially give you more new features or bug fixes.

Your release cycle, customer demands, productions issues and other influencing factors, will determine your version upgrade choices. You might decide a multi-step approach, where you do a small version upgrade immediately to resolve current issues and then work on the larger upgrade subsequently to get the benefits of using a newer version. Or, you might be okay with doing an upgrade to the latest available version straight away. Potentially, a combination of approaches in different branches of your source code management system is used to figure out the best way of going forward with the upgrade.

Sonatype CLM for Eclipse and other tools of the Sonatype CLM suite can assist you through the process of upgrading, as well as monitoring, the applications after upgrade completion.

Chapter 6

Using Sonatype CLM with Other IDEs

While the integration with Eclipse offered by Sonatype CLM for IDE is the most powerful tooling for developers available, user of other popular integrated development environments are not left without support. All common Java IDEs have powerful integration with Apache Maven and therefore can be used together with CLM Maven Plugin for project evaluation against your Sonatype CLM server.

This chapter showcases the integration with IntelliJ IDEA from JetBrains and NetbeansIDE from Oracle.

6.1 Maven Plugin Setup

In our example setup for the usage with other IDE's we are going to add a plugin configuration for the CLM Maven Plugin into the `pom.xml` file of the project we want to analyze as documented in Example Configuration of the CLM Maven Plugin. This configuration defines `serverUrl` of the CLM server to be contacted for the evaluation, the `applicationId` used to identify the application in the CLM server to evaluate against and the `stage` configuration to use for the evaluation.

Example Configuration of the CLM Maven Plugin

```
<build>
  <pluginManagement>
    <plugins>
      <plugin>
        <groupId>com.sonatype.clm</groupId>
```

```
        <artifactId>clm-maven-plugin</artifactId>
        <version>2.1.1</version>
        <configuration>
          <serverUrl>http://localhost:8070</serverUrl>
          <applicationId>test</applicationId>
          <stage>develop</stage>
        </configuration>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
```

With this configuration in place a user can kick off an evaluation with the command line `mvn package clm:evaluate`.

This will result in an output detailing the components to be analyzed, any policy violations and a link to the resulting report in the Sonatype CLM server.

Note
To speed the build up you can skip the test compilation and execution by passing `-Dmaven.test.skip=true` on the command line invocation, since it is not needed for the CLM evaluation.

6.2 IntelliJ IDEA

IntelliJ IDEA supports Maven projects natively and you can simply open a project in the IDE by opening the `pom.xml` file.

Once your project is opened and you have added the plugin configuration for the CLM Maven Plugin from Example Configuration of the CLM Maven Plugin, you can create a configuration to run the desired Maven command.

Select *Edit Configurations* from the *Run* menu, press the + button and select *Maven* to add a new configuration. Enter the command line and other desired details as displayed in Figure 6.1

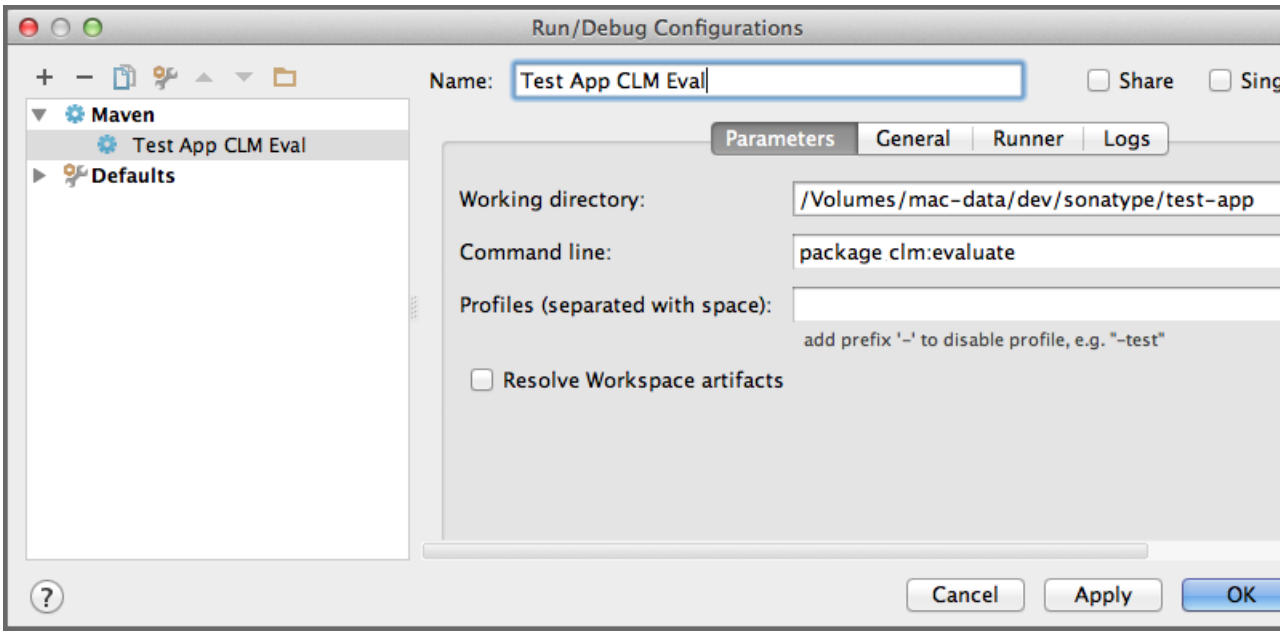

Figure 6.1: Creating a Maven Run Configuration for a CLM Evaluation in IntelliJ

After pressing *OK* in the dialog the new configuration will be available in the run configuration drop down as well the *Maven Projects* view. You can open the view using the *View* menu, selecting *Tools Window* and pressing *Maven Projects*. You will see the window appear in the IDE looking similar to Figure 6.2. It displays the run configuration selector with the green play button on the top as well as the Maven project with the CLM evaluation run configuration.

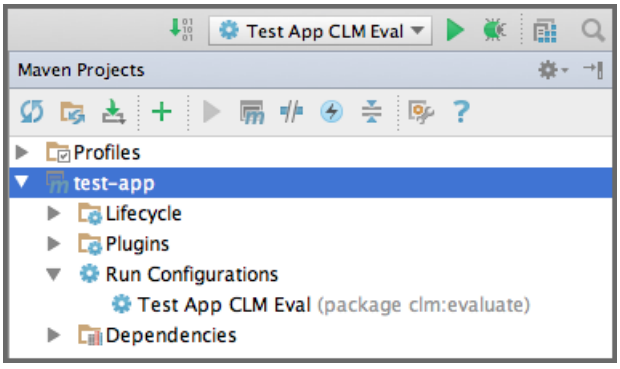

Figure 6.2: Maven Projects View with the CLM Evalulation Run Configuration in IntelliJ

You can press the green play button beside the run configuration, or the configuration entry itself in the *Maven Projects* window, to start a build. The build will run in an embedded console window in the IDE as displayed in Figure 6.3, and show all the output from the Maven build including any error messages and the link to the produced report in the Sonatype CLM server. Policy violations can be configured to result in a build failure.

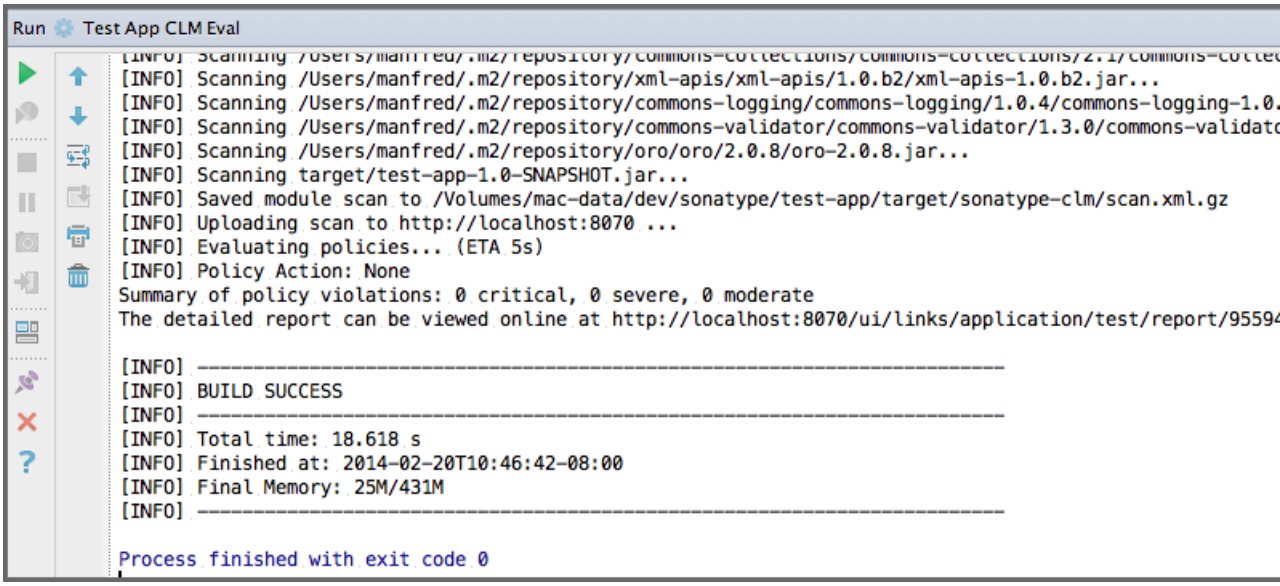

Figure 6.3: CLM Maven Plugin Output in the Run Console in IntelliJ

6.3 Netbeans IDE

Netbeans IDE supports Maven projects natively and you can simply open a project in the IDE by choosing *Open Project* from the *File* menu and navigating to the directory that contains your project.

Once your project is opened, you can expand the *Project Files* section in the *Projects* window as displayed in Figure 6.4. Double-click on the `pom.xml` file and add the plugin configuration for the CLM Maven Plugin from Example Configuration of the CLM Maven Plugin.

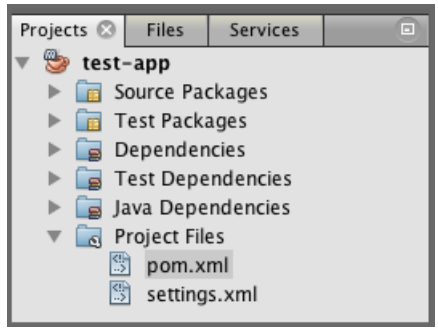

Figure 6.4: Project View with the `pom.xml` in Netbeans

If you right-click on the `pom.xml` file, you can choose *Run Maven* and *Goals*, to display the dialog displayed in Figure 6.5. Enter the configuration as displayed and don't forget to select *Remember as:* providing a name. This will allow you to simply start this defined configuration from the context *Run Maven* context menu of the `pom.xml` file.

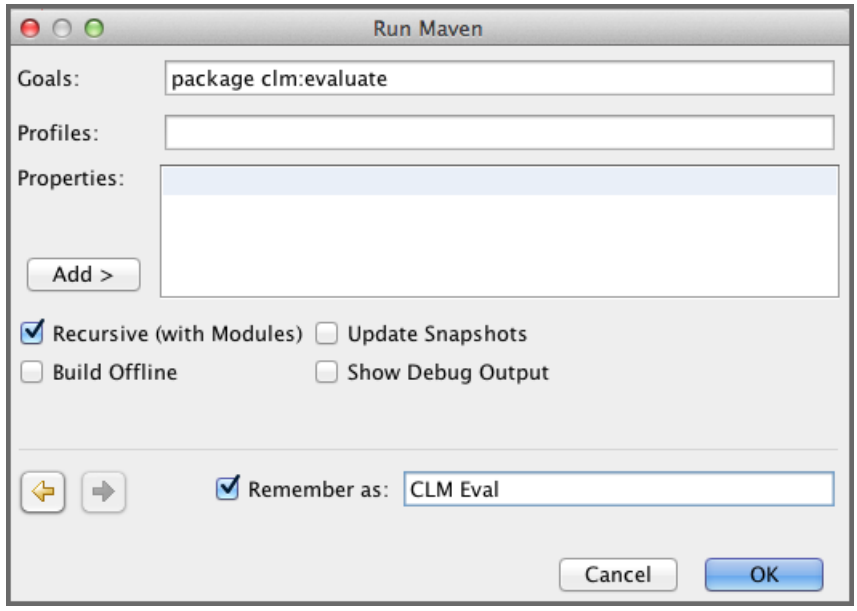

Figure 6.5: Maven Goal Setup for a CLM Evaluation in Netbeans

After pressing *OK* the defined Maven execution will start and display the output including any error messages and the link to the produced report in the Sonatype CLM server in the Output window displayed in Figure 6.6. Policy violations can be configured to result in a build failure.

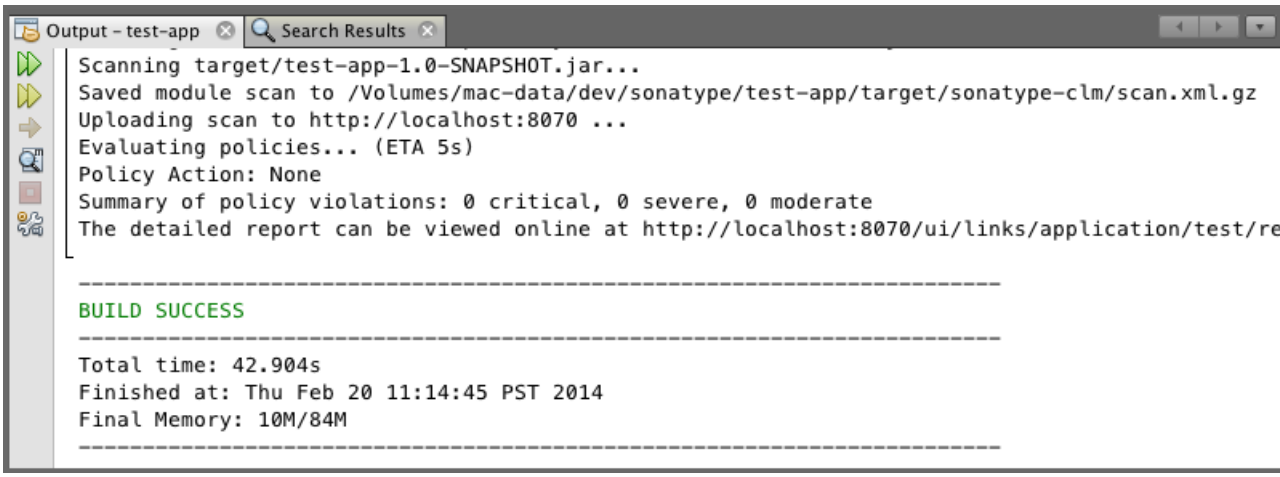

Figure 6.6: CLM Maven Plugin Output in the Output Window in Netbeans