### Maven POM Configuration

This section describes the configuration of [pom.xml](https://github.com/logpresso/logpresso-app-examples/blob/main/logpresso-sample-app/pom.xml). Please refer to the source for full details.

#### Artifact packaging

<groupId>com.logpresso.sonar</groupId><artifactId>logpresso-sample-app</artifactId><version>1.0.2306.0</version><packaging>bundle</packaging><name>Logpresso Sample App</name>

Always specify the group ID of your Logpresso app as com.logpresso.sonar. We recommend defining the artifact ID in the form logpresso-VENDOR-MODEL. The version is defined in the form Major.Minor.YYMM.REV. This is because it is easier to estimate the compatible Logpresso platforms if the app version indicates when it was deployed.

Finally, it's important to note that the packaging is specified as bundle. The maven-bundle-plugin plugin, described below, does the bundle packaging.

#### Properties

<properties> <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding> <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding> <jdk.version>11</jdk.version></properties>

Specify the file encoding as UTF-8 so that the operating system locale does not affect the build, and specify the JDK version as 11. The jdk.version variable is used below.

#### Maven repositories

<repositories> <repository> <id>logpresso-repo</id> <name>Logpresso Maven Repository</name> <url>https://maven.logpresso.com/</url> </repository></repositories><pluginRepositories> <pluginRepository> <id>logpresso-plugin-repo</id> <name>Logpresso Maven Repository</name> <url>https://maven.logpresso.com/</url> </pluginRepository></pluginRepositories>

Logpresso provides the libraries needed for third-party development in the <https://maven.logpresso.com> repository. Add the <pluginRepository> setting to download the iPOJO plugin needed for bytecode manipulation during the build phase, and the <repository> setting to download the app API and iPOJO library.

#### Compiler options

Specify the source code and bytecode versions, whether to include debug information, and whether to optimize when compiling Java.

<plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <version>3.8.1</version> <configuration> <encoding>UTF-8</encoding> <source>${jdk.version}</source> <target>${jdk.version}</target> <debug>true</debug> <optimize>true</optimize> <showDeprecations>true</showDeprecations> </configuration></plugin>

The maven-bundle-plugin plugin creates an OSGi bundle by adding an OSGi manifest to a JAR file. An OSGi bundle is a JAR file with an OSGi manifest appended to the META-INF/MANIFEST.MF file.

<plugin> <groupId>org.apache.felix</groupId> <artifactId>maven-bundle-plugin</artifactId> <version>5.1.4</version> <extensions>true</extensions> <configuration> <instructions> <Bundle-SymbolicName>com.logpresso.sonar.sample</Bundle-SymbolicName> <Export-Package> com.logpresso.sonar.sample;version=${project.version}, </Export-Package> <Import-Package> org.json;version="1.1.0", org.araqne.codec;version="2.2", org.araqne.log.api;version="3.13.0", org.logpresso.api.profile;version="1.1.0", org.logpresso.api.profile.query;version="1.1.0", org.araqne.logdb;version="3.10.0", org.araqne.msgbus;version="1.12.0", com.logpresso.sonar.api.\*;resolution:=optional, \* </Import-Package> <Private-Package> com.logpresso.sonar.sample.impl, com.logpresso.sonar.sample.query, </Private-Package> </instructions> </configuration></plugin>

You need to look at the four items in <instruction>:

* Bundle-SymbolicName: Defines the bundle identifier. Write it according to Java package naming conventions.
* Export-Package: Enter a list of packages to be exposed to other OSGi bundles. Typically, you would only expose packages that contain Java interfaces. Use the ${project.version} macro to specify the version of the package you are releasing.
* Import-Package: Enter a list of packages to import from another OSGi bundle. Specify the minimum version by defining ;version="VERSION" after the package name. If you don't specify a version, the build plugin uses the version of the maven artifact defined in <dependencies>, so the range of compatible package versions may not be as wide as intended and bundle dependencies may not be resolved.
* Private-Package: Enter a list of internal packages that you do not want to make public to other OSGi bundles. Typically, packages containing implementations should not be made public; only interfaces should be made public externally, so that implementation details can be easily changed at any time.

The Import-Package predefines the versions of the packages required for the Logpresso app API calls. The set of package versions that can be imported may change depending on the sonar-app-api library version (currently 4.0.2306.0).

<plugin> <groupId>org.apache.felix</groupId> <artifactId>maven-ipojo-plugin</artifactId> <version>1.12.1.asm8</version> <executions> <execution> <goals> <goal>ipojo-bundle</goal> </goals> </execution> </executions></plugin>

iPOJO is a framework that makes it possible to develop OSGi components declaratively. Because OSGi assumes runtime plugins, features that each bundle depends on can be installed or removed at any time. This makes developing using only the OSGi interface very complicated to implement, as you need to be notified of every state change, cascading to disable or enable features. iPOJO enables you to develop OSGi services declaratively by defining a component lifecycle model and adding simple annotations.

iPOJO manipulator recognizes iPOJO annotations during the build phase and performs the automatic conversion by inserting the bytecode. Therefore, maven-ipojo-plugin must be called during the build phase as shown above.

#### Dependencies

<dependencies> <dependency> <groupId>junit</groupId> <artifactId>junit</artifactId> <version>4.8.1</version> <scope>test</scope> </dependency> <dependency> <groupId>org.apache.felix</groupId> <artifactId>org.apache.felix.ipojo.annotations</artifactId> <version>1.12.1.asm8</version> </dependency> <dependency> <groupId>org.slf4j</groupId> <artifactId>slf4j-api</artifactId> <version>1.7.12</version> </dependency> <dependency> <groupId>com.logpresso.sonar</groupId> <artifactId>sonar-app-api</artifactId> <version>4.0.2306.0</version> </dependency></dependencies>

The Logpresso app uses the following libraries:

* junit: Used for writing unit tests. Unit test classes are placed under the src/test/java directory.
* org.apache.felix.ipojo.annotations: Provides iPOJO component annotations.
* slf4j-api: Provides logging interface. Logs are written to the log/araqne.log file, and the log level can be adjusted at runtime.
* sonar-app-api: Provides the Logpresso app API. The version of this library refers to the compatible Logpresso platform version.