Yamcs Maven Plugin

Release 1.2.8-SNAPSHOT

Yamcs
MISSION CONTROL

Space Applications Services, NV/SA
Leuvensesteenweg 325
1932 Sint-Stevens-Woluwe
Belgium
spaceapplications.com
yamcs.org

Aerospace Applications North America, Inc.
16850 Saturn Ln, Ste 100
Houston, TX 77058
United States of America
aerospaceapplications-na.com

Contents

1. About Yamcs Maven Plugin

This is a Maven plugin for developing a Yamcs application.

Yamcs is a Java-based open source mission control framework. Its functionalities can be extended with your own custom code.

Goals

Goal	Description
yamcs:run (page 3)	Run Yamcs as part of a Maven build.
yamcs:debug (page 4)	Run Yamcs in debug mode as part of a Maven build.
yamcs:bundle (page 5)	Bundle a Yamcs application into a single archive file.
yamcs:run-tool (page 6)	Run a Yamcs-related tool as part of a Maven build.
yamcs:detect (page 6)	Detect metadata for Yamcs plugins.

Usage

This plugin expects to find Yamcs configuration in `${project.basedir}/src/main/yamcs` in subfolders `etc` and `mdb`.

In the pom.xml add dependencies to the desired Yamcs modules. At least a dependency to yamcs-core is required. yamcs-web is another common dependency that makes Yamcs host a prebuilt copy of the Yamcs web interface:

```
<project>
  ...
  <packaging>jar</packaging>

  <properties>
    <yamcsVersion>5.5.5</yamcsVersion>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.yamcs</groupId>
      <artifactId>yamcs-core</artifactId>
      <version>${yamcsVersion}</version>
    </dependency>
    <dependency>
      <groupId>org.yamcs</groupId>
      <artifactId>yamcs-web</artifactId>
      <version>${yamcsVersion}</version>
    </dependency>
    ...
  </dependencies>
```

(continues on next page)

(continued from previous page)

```
  <build>
    <plugins>
      <plugin>
        <groupId>org.yamcs</groupId>
        <artifactId>yamcs-maven-plugin</artifactId>
        <version>1.2.8-SNAPSHOT</version>
      </plugin>
    </plugins>
  </build>

</project>
```

To run a Yamcs application:

```
mvn yamcs:run
```

Examples

- *Yamcs Plugin* (page 7)
- *Packaging Yamcs* (page 8)
- *Multi-Packaging Yamcs* (page 10)

2. Goals

2.1 yamcs:run

Runs Yamcs as part of a Maven build.

Attributes:

- Requires a Maven project to be executed.
- Requires dependency resolution of artifacts in scope: `test`.
- Invokes the execution of the lifecycle phase `process-classes` prior to executing itself.

Optional Parameters

args (list)
Arguments passed to the Yamcs executable. Add each argument in an <arg> subelement.

User property is: `yamcs.args`

configurationDirectory (file)
The directory that contains Yamcs configuration files. By convention this contains subfolders named `etc` and `mdb`.

Relative paths in yaml configuration files are resolved from this directory.

Default value is: `${basedir}/src/main/yamcs`

User property is: `yamcs.configurationDirectory`

directory (file)
The directory to create the runtime Yamcs server configuration under.

Default value is: `${project.build.directory}/yamcs`

User property is: `yamcs.directory`

jvmArgs (list)
JVM Arguments passed to the forked JVM that runs Yamcs. Add each argument in a <jvmArg> subelement.

User property is: `yamcs.jvmArgs`

skip (boolean)
Skip execution

Default value is: `false`

User property is: `yamcs.skip`

stopTimeout (long)
Time in milliseconds that a graceful stop of Yamcs is allowed to take. When this time has passed, Yamcs is stopped forcefully. A value < 0 causes the stop to be done async from the Maven JVM.

User property is: `yamcs.stopTimeout`

2.2 yamcs:debug

Runs Yamcs in debug mode as part of a Maven build.

Attributes:

- Requires a Maven project to be executed.
- Requires dependency resolution of artifacts in scope: `test`.
- Invokes the execution of the lifecycle phase `process-classes` prior to executing itself.

Optional Parameters

args (list)
Arguments passed to the Yamcs executable. Add each argument in an <arg> subelement.

User property is: `yamcs.args`

configurationDirectory (file)
The directory that contains Yamcs configuration files. By convention this contains subfolders named `etc` and `mdb`.

Relative paths in yaml configuration files are resolved from this directory.

Default value is: `${basedir}/src/main/yamcs`

User property is: `yamcs.configurationDirectory`

directory (file)
The directory to create the runtime Yamcs server configuration under.

Default value is: `${project.build.directory}/yamcs`

User property is: `yamcs.directory`

jvmArgs (list)
JVM Arguments passed to the forked JVM that runs Yamcs. Add each argument in a <jvmArg> subelement.

User property is: `yamcs.jvmArgs`

jvmDebugPort (int)
Port for debugging

Default value is: `7896`

User property is: `yamcs.jvm.debug.port`

jvmDebugSuspend (boolean)
Suspend when debugging

User property is: `yamcs.jvm.debug.suspend`

skip (boolean)
Skip execution

Default value is: `false`

User property is: `yamcs.skip`

stopTimeout (long)
Time in milliseconds that a graceful stop of Yamcs is allowed to take. When this time has passed, Yamcs is stopped forcefully. A value < 0 causes the stop to be done async from the Maven JVM.

User property is: `yamcs.stopTimeout`

2.3 yamcs:bundle

Bundle a Yamcs application into a single archive file.

Attributes:

- Requires a Maven project to be executed.
- Requires dependency resolution of artifacts in scope: `compile+runtime`.
- Invokes the execution of the lifecycle phase `package`.

Optional Parameters

attach (boolean)
Controls whether this mojo attaches the resulting bundle to the Maven project.

Default value is: `true`

User property is: `yamcs.attach`

classifier (string)
Classifier to add to the generated bundle.

Default value is: `bundle`

configurationDirectory (file)
The directory that contains Yamcs configuration files. By convention this contains subfolders named `etc` and `mdb`.

Relative paths in yaml configuration files are resolved from this directory.

Default value is: `${basedir}/src/main/yamcs`

User property is: `yamcs.configurationDirectory`

includeDefaultWrappers (boolean)
Whether `yamcs` and `yamcsadmin` wrapper scripts should be included in the bundle.

Default value is: `true`

User property is: `yamcs.includeDefaultWrappers`

includeConfiguration (boolean)
Whether this module's configuration directory (default location: `src/main/yamcs`) should be included in the bundle

Default value is: `true`

User property is: `yamcs.includeConfiguration`

formats (list)
Specifies the formats of the bundle. Multiple formats can be supplied. Each format is specified by supplying one of the following values in a <format> subelement:

- zip - Creates a ZIP file format
- tar - Creates a TAR format
- tar.gz or tgz - Creates a gzip'd TAR format
- tar.bz2 or tbz2 - Creates a bzip'd TAR format
- tar.snappy - Creates a snappy'd TAR format
- tar.xz or txz - Creates a xz'd TAR format

If unspecified the behavior is equivalent to:

```
<formats>
  <format>tar.gz</format>
</formats>
```

skip (boolean)
Skip execution

Default value is: `false`

User property is: `yamcs.skip`

2.4 yamcs:run-tool

Runs a Yamcs-related tool as part of a Maven build.

Attributes:

- Requires a Maven project to be executed.
- Requires dependency resolution of artifacts in scope: `test`.
- Invokes the execution of the lifecycle phase `process-classes` prior to executing itself.

Optional Parameters

args (list)
Arguments passed to the Yamcs executable. Add each argument in an <arg> subelement.

User property is: `yamcs.args`

tool (string)
Class name of the tool to execute.

User property is: `yamcs.tool`

directory (file)
The directory where Yamcs is installed.

Default value is: `${project.build.directory}/yamcs`

User property is: `yamcs.directory`

2.5 yamcs:detect

Finds implementations of `org.yamcs.Plugin` in the current project and generates metadata for Yamcs.

Attributes:

- Requires a Maven project to be executed.
- Requires dependency resolution of artifacts in scope: `compile`.
- Binds by default to the lifecycle phase `process-classes`.

Optional Parameters

None

3. Examples

3.1 Yamcs Plugin

Writing a Yamcs plugin is just like writing any other jar. Declare your dependencies to the desired Yamcs artifacts, and define the Java version that you want to comply with. Official Yamcs plugins strive to remain compatible with Java 8 language features for the foreseeable future, but you are free to use more recent Java version in your project if you can.

To prototype your plugin in a local Yamcs application, add the `yamcs-maven-plugin` to the plugins section. Once you have specified a valid configuration in `src/main/yamcs/`, you can get your copy of Yamcs running with:

```
mvn yamcs:run
```

To package your Yamcs plugin, simply do `mvn package`. The resulting jar artifact can be dropped in the `lib/` or `lib/ext/` folder of any compatible Yamcs server.

For optimal integration we recommend adding an execution of the *yamcs:detect* (page 6) mojo as shown below. It will allow Yamcs to find metadata on your plugin and will give your plugin the opportunity to hook into the lifecycle of Yamcs.

```
<project>
  ...
  <packaging>jar</packaging>

  <properties>
    <yamcsVersion>5.5.5</yamcsVersion>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.yamcs</groupId>
      <artifactId>yamcs-core</artifactId>
      <version>${yamcsVersion}</version>
    </dependency>
    <dependency>
      <groupId>org.yamcs</groupId>
      <artifactId>yamcs-web</artifactId>
      <version>${yamcsVersion}</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
```

(continues on next page)

(continued from previous page)

```
      <plugin>
        <groupId>org.yamcs</groupId>
        <artifactId>yamcs-maven-plugin</artifactId>
        <version>1.2.8-SNAPSHOT</version>
        <executions>
          <execution>
            <goals>
              <goal>detect</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

3.2 Packaging Yamcs

Running through Maven is useful for development and for creating prototypes, but it is not recommended for production environments.

This plugin includes a `bundle` goal, which supports two packaging approaches:

- Bundle everything (Yamcs + Yamcs Plugins + Your Project) in one single distribution
- Bundle only your project

Bundling everything together is convenient, whereas the split approach allows you to make use of official Yamcs distributions.

In both cases the `bundle` goal of the yamcs-maven-plugin binds to the Maven `package` lifecycle phase. This makes Maven generate a Yamcs application with the command `mvn package`.

The resulting artifact can be used as input to platform-specific packaging tools, for example to create an RPM or DEB package.

Note: The `bundle` goal supports only a limited set of options. If you require to have more control over the layout and contents of your package, use other Maven plugins such as maven-assembly-plugin[1].

3.2.1 All-in-one

This example bundles Yamcs together with your extensions and configurations in one integrated distribution.

```
<project>
  ...
  <packaging>jar</packaging>

  <properties>
    <yamcsVersion>5.5.5</yamcsVersion>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.yamcs</groupId>
      <artifactId>yamcs-core</artifactId>
      <version>${yamcsVersion}</version>
    </dependency>
    <dependency>
```

(continues on next page)

[1] https://maven.apache.org/plugins/maven-assembly-plugin/

(continued from previous page)

```
      <groupId>org.yamcs</groupId>
      <artifactId>yamcs-web</artifactId>
      <version>${yamcsVersion}</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.yamcs</groupId>
        <artifactId>yamcs-maven-plugin</artifactId>
        <version>1.2.8-SNAPSHOT</version>
        <executions>
          <execution>
            <id>bundle-yamcs</id>
            <phase>package</phase>
            <goals>
              <goal>bundle</goal>
            </goals>
            <configuration>
              <formats>
                <format>tar.gz</format>
                <format>zip</format>
              </formats>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

3.2.2 Project Only

This example bundles only your extensions and configurations. The generated package can be extracted into an existing Yamcs installation directory.

Set the Maven scope of standard Yamcs dependencies to `provided`. This way they can be used during compilation, while the `bundle` goal will ignore them.

Set also `includeDefaultWrappers` to `false` to prevent the `yamcs` and `yamcsadmin` shell scripts from being added to your package. These are already included in official Yamcs core builds.

```
<project>
  ...
  <packaging>jar</packaging>

  <properties>
    <yamcsVersion>5.5.5</yamcsVersion>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.yamcs</groupId>
      <artifactId>yamcs-core</artifactId>
      <version>${yamcsVersion}</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.yamcs</groupId>
      <artifactId>yamcs-web</artifactId>
      <version>${yamcsVersion}</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
```

(continues on next page)

(continued from previous page)

```
  <build>
    <plugins>
      <plugin>
        <groupId>org.yamcs</groupId>
        <artifactId>yamcs-maven-plugin</artifactId>
        <version>1.2.8-SNAPSHOT</version>
        <executions>
          <execution>
            <id>bundle-yamcs</id>
            <phase>package</phase>
            <goals>
              <goal>bundle</goal>
            </goals>
            <configuration>
              <includeDefaultWrappers>false</includeDefaultWrappers>
              <formats>
                <format>tar.gz</format>
                <format>zip</format>
              </formats>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

3.3 Multi-Packaging Yamcs

Multiple Yamcs applications can be packaged from a single Maven project by defining multiple executions of the Yamcs Maven Plugin. Each execution must have a separate execution id. You should also specify different `classifier` properties in the configuration block of each execution. The classifier is used in the naming of the generated bundles. Without it, the two executions would overwrite each others outputs.

If you need different configurations of Yamcs for each server, then look into overriding the `configurationDirectory` (default is `src/main/yamcs/`).

```
<project>
  ...
  <artifactId>myproject</artifactId>
  <packaging>jar</packaging>

  <properties>
    <yamcsVersion>5.5.5</yamcsVersion>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.yamcs</groupId>
      <artifactId>yamcs-core</artifactId>
      <version>${yamcsVersion}</version>
    </dependency>
    <dependency>
      <groupId>org.yamcs</groupId>
      <artifactId>yamcs-web</artifactId>
      <version>${yamcsVersion}</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.yamcs</groupId>
        <artifactId>yamcs-maven-plugin</artifactId>
        <version>1.2.8-SNAPSHOT</version>
```

(continues on next page)

(continued from previous page)

```
        <executions>
          <execution>
            <id>bundle-yamcs1</id>
            <phase>package</phase>
            <goals>
              <goal>bundle</goal>
            </goals>
            <configuration>
              <classifier>ops</classifier>
            </configuration>
          </execution>
          <execution>
            <id>bundle-yamcs2</id>
            <phase>package</phase>
            <goals>
              <goal>bundle</goal>
            </goals>
            <configuration>
              <classifier>sim</classifier>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

This will generate two bundles:

```
target/
|-- myproject-1.0.0-SNAPSHOT-ops.tar.gz
|-- myproject-1.0.0-SNAPSHOT-sim.tar.gz
```