Delivery Manager Tool - Information - How to Package Java applications using the Delivery Manager Tools

Purpose

This page advise you on how to package a Java Application based on Maven and Eclipse projects.

Applicable in CAST Version

Release	Yes/No
8.3.x	✅
8.2.x	✅
8.1.x	✅
8.0.x	✅

Applicable RDBMS

RDBMS	Yes/No
Oracle Server	✅
Microsoft SQL Server	✅
CSS3	✅
CSS2	✅

Details

1. **Qualify the application: maven, eclipse, both or another?**

   Currently, in CAST AIP, we're covering 2 main cases: maven and eclipse.
   The fact that the source code is containing maven (pom.xml) or eclipse (.project + .classpath) files is a good starting point to automate the configuration of the analysis but it is not enough to qualify an application. What we really need to know is what method is being used to build the source code and/or what method is being used by developers in the IDE to maintain the projects. Typically we can have old pom.xml with the source code that are not maintained anymore and the reference to use for the configuration is the eclipse files.

   a. What files can you trust ? Answering this question is really important as it'll have an impact on:

      i. The delivery process.
      ii. The selection. by default, before 7.0, the priority was the eclipse project. Since 7.1, the priority is given to maven projects as in the good usage of maven, the eclipse classpath is generated by the maven plugin in eclipse. In that case, it will not contain the information required to do a good configuration. You can identify this use case in the .classpath file with the following entry:

      ```
      <classpathentry kind="con" path="org.eclipse.m2e.MAVEN2_CLASSPATH_CONTAINER">
                                      <attributes>
                                                                      <attribute name="maven.
      pomderived" value="true"/>
                                      </attributes>
      </classpathentry>
      ```

   b. Are all projects based on the same mechanism?
      Typically, for some application like, the main 3 sub-folders ("al" + "ml" + "nmsjobs") are based on maven but the last one ("cim") contains a mix of different cases.

2. **Packaging for Maven**

a. Source code completeness: missing parent.

i. The Java discoverer does not create a project for each pom.xml. The DMT does not create a project for:

1. The core packaging value "pom": it's a pom parent.
2. The core packaging value "ear": it's a grouping of war. A warning is displayed in that case.
3. The non-core packaging values: a warning is displayed in that case.
4. When we want to use the maven discovery, the most important element to validate is to have all the pom parent files, including the top level pom (super pom). The main reason is that it will contain a big part of the configuration.

ii. In 8.2.3, when you don't have the pom parent, a warning message (**id="cast.dmt.discover.jee.maven.missingPomParent" format="Maven Java project: The parent artifact %PARENT% required by the artifact %ID% is missing. The parent artifact must be provided inside this package."**) is generated to help you to identify an incomplete delivery. In that case, you must clarify in what case you are:

1. If you have multiple "file system" package and the project and the parent are not in the same package then it's a known limitation. You must change the packaging in the DMT to regroup them.
2. If the pom parent is not stored with the source code (for some companies, they decide to share some common settings between applications – in that case, the top pom is available in the maven repository but it's not stored in the source code of the application), it's a know limitation. You must adapt your delivery process to include the pom parent in the package. One way to do it is to use the extraction from the file system and to add the corresponding maven files.
3. The source code delivered is not complete. In that case, you must request from the customer to deliver all the files.
4. To help you to identify them, you can also use the folder hierarchy:

   By default, the pom parent is in the parent folder.
   Otherwise the relativePath is specified in the parent section. It indicates the corresponding folder to search in :

```
<parent> <groupId>com.castsoftware</groupId> <artifactId>CAST-DeliveryServerParent<
/artifactId> <version>8.2.3</version> <relativePath>../CAST-DeliveryServerParent<
/relativePath> </parent>
```

iii. **What are the impacts of the missing pom parent ? Without all the pom parent, you'll mainly miss 3 groups of data:**

1. Value of a variable. The symptom is a name containing a variable (like ${xxx}), either:
   a. In the list of discovered maven projects, the project name
   b. In the list of alerts, the alert
   c. You need to search for the <properties> block in the current pom.xml or in its parents
2. Version of a dependency. The symptom is an alert without a version.
   a. In the current pom, we have a set of dependency defined in the <dependencies> block. But we also need the <dependencyManagement> block that defines the common version of the dependency to use inside the child.
   b. List of all dependencies. There is no symptom in the DMT (except the warning about the missing parent) when a dependency is defined in one of the parent which is not available. The information is missing and you'll see a symptom during the analysis.
   c. This is the main reason why it is not recommended to start looking at the other information before having all the parents.

iv. **Supported packaging values. Only some values for packaging are supported. The list of supported is:**

1. Default maven
2. jar
3. war
4. ejb
5. ejb-client
6. some specific ones that we know are compatible with the default :
   a. sonar-plugin
   b. maven-plugin
   c. bundle
7. For the others, you'll get an information message telling you that this project is ignored. For example, the packaging "content-package" is ignored:

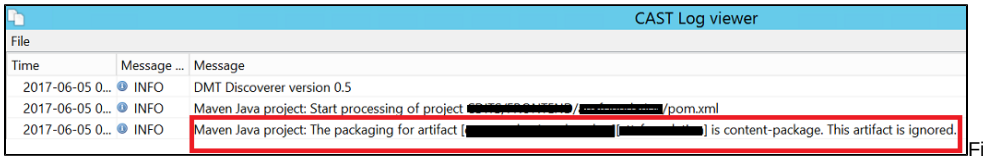

Figure 1

b. **Alerts for Maven**

i. There are different types of alerts under DMT based on different technologies. For maven, you will have :

1. Missing projects: this is the main case to check. Go to the paragraph Missing dependencies: incomplete delivery or third party
2. Missing source folder: you can't do anything for that.
3. Ambiguous project reference: initially you should not have this alert. If you add multiple packages for maven repositories, you can have new alerts if the artifact has been found multiple times. Refer to DMT packaging process for maven.

a. There are 2 types of ambiguity when it comes to maven projects. viz: External & Internal jars.
b. External jars one is explained below. To remediate missing project alerts, we would have added more than one URL's for Maven http repository. Hence, there is a possibility of packaging same artifact in both packages, which will result in ambiguity.

i. The only workaround currently we can think of is to turn one package into a maven file system and remove the artifact in double and re-package entire version. Please go through screenshot below

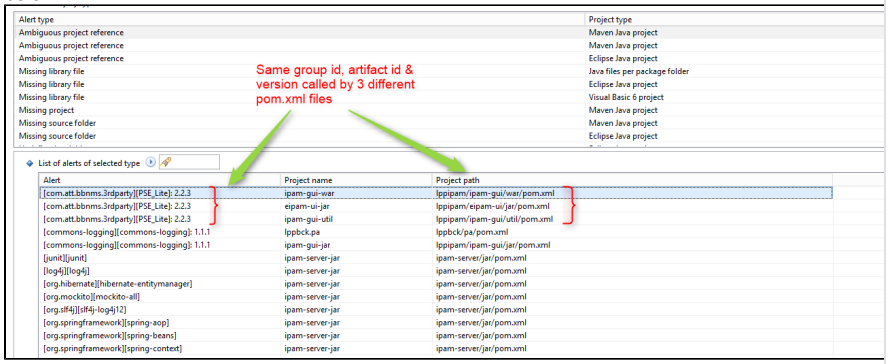

Figure 2

4. Undefined variable

c. **Missing dependencies: incomplete delivery or third party.**

i. Each maven project is an artifact. It is identified by a groupId + artifactId [+ packaging] + version. But we do not know if it is an artifact that should be delivered as part of the source code of the application or a third party. The only clue is the value of the groupId which is generally clear enough.
ii. The "internal" dependencies can be delivered inside the same package or in another "file system" package. If you immediately start to connect to a maven repository, the corresponding compiled artifact will be extracted and you will not identify these missing "internal" dependencies.
iii. In all cases for maven, you must create one or multiple package "Automated extraction of required jar files".

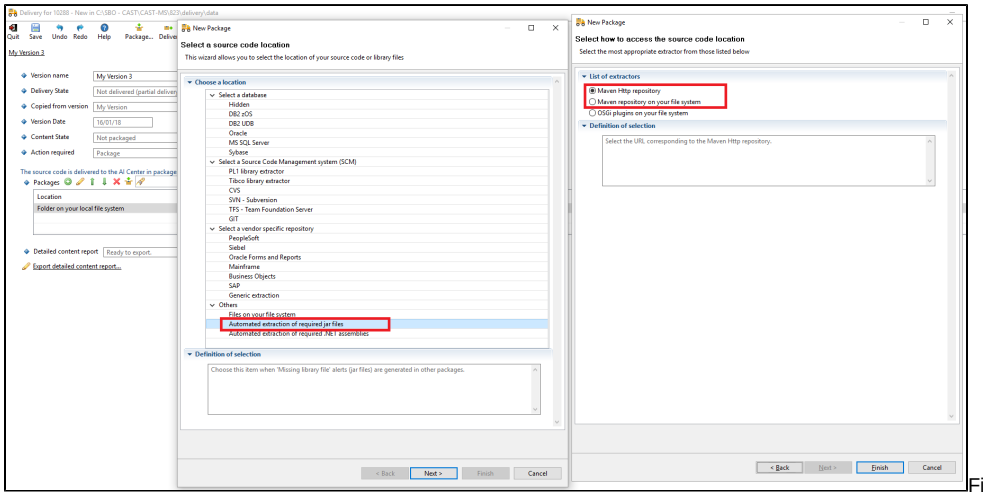

Figure 3

d. **Tips and Tricks**

i. You can split into different packages per top pom which, most of the time, will correspond to the main sub-folders when any. The advantage is that you'll identify project of the same name (like "util") in different context. It's useful for the configuration in CMS.

3. **Packaging for Eclipse**

a. There are different types of alerts under DMT based on different technologies. Few of them are as below for J2EE technology.

i. Ambiguous project reference
ii. Missing library files
iii. Missing projects (eclipse/Maven)
iv. Missing source folder
v. Library file not found in chosen root
vi. Undefined variable
vii. Unreachable library file

Let us find out in detail one-by-one with specific cases like below. Let us take an example of "**Undefined Variable**" for now.

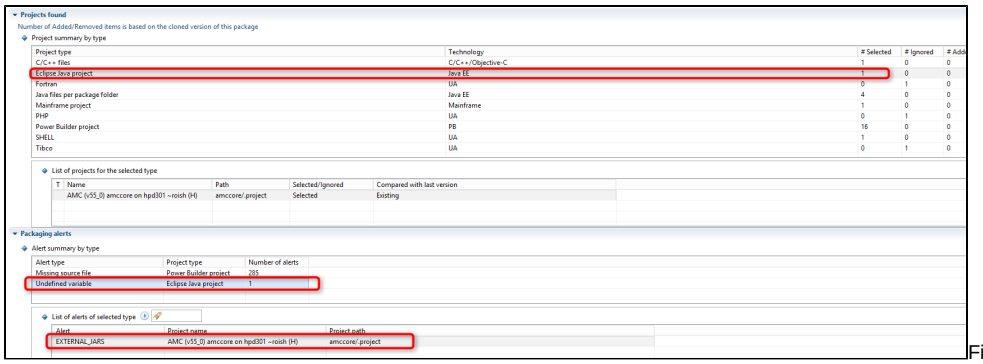

Figure 4

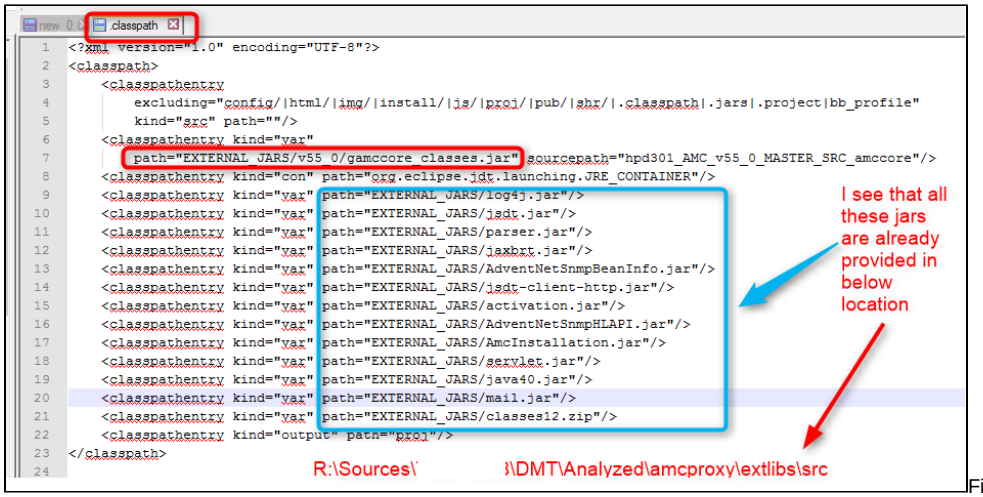

Figure 5

As the above screenshots are self explanatory about the issue, it is evident that we provide a value to variable as given in below screenshot to resolve the same.

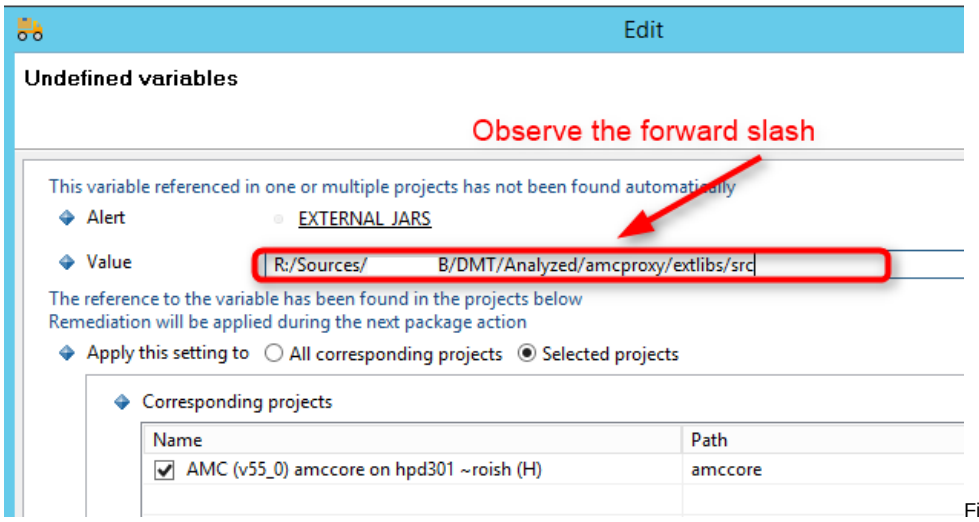

Figure 6

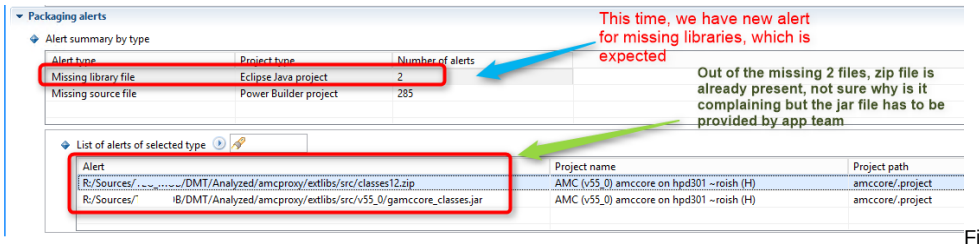

Figure 7

After adding the value, new alerts will pop-up for missing library, which is expected if some of the libraries are missing in the path provided (In screenshot, ignore missing **zip file** as of now. It is confirmed that current DMT won't recognize zip files. But the missing jar file has to be taken into account, they need to be available for the package)

Notes/comments

Related Pages