Use the Maven Distribution Archetype

The Maven Distribution Archetype is a Maven Archetype that creates a new Maven Project that creates Aspire distributions or AppBundles.

What is a Maven Archetype?

It is a special Maven Project that can create new Maven Projects. In basic terms, it automatically sets up all directories and project files for your new distribution.

Why not just create a distribution? Why create a Maven Project to create distributions?

In production systems there is never just one distribution. Most production systems have lots of different distributions:

- Distributions for multiple machines in a large distributed system
- A test or staging area
- Development distributions
- Distributions for individual developer machines

Using Maven profiles (see Introduction to Maven Profiles), it is possible to use a single Maven project to build all of these different types of distributions.

The Maven project built by the aspire-distribution-archetype is the starter project for this process. It can be used to create a single type of distribution, and then modified with Maven profiles to create distributions for all systems in your customer's development, test, and production environments.

On this page:

Related pages:

- Maven profiles
- Maven assembly plugin

Structure of a Distribution Project

Distribution projects have the following structure:

- **{distribution-project-directory}**

  This is the directory created by the Maven distribution archetype.

  - **distribution-files**

    Holds a "project template" which is used to build your distribution. This directory contains all of the files which make up your distribution, except those which are stored in the Maven repository (those are downloaded as needed when the project is built). Note well: If you need to change your distribution, you must change the files in the distribution-files directory. Otherwise, your changes will be overwritten when you next build your project.
  - **pom.xml**

    Holds the Maven instructions for building the project. This is the file which is used by the "mvn" command.
  - **distribution.xml**

    Holds detailed instructions on how to put your project together. "distribution.xml" specifies what files are copied from the distribution-files directory, and where they should go in your final project. It also specifies what files are downloaded from Maven repositories and where they should go.
  - **appBundle.xml**

    Specifies instructions on how to build your project as an Application Bundle JAR. Application Bundles can be deployed to Maven repositories and then loaded into any other instances of Aspire. See AppBundles (Aspire 2) for more information.

Before You Begin

Be sure to set up your installation of Maven to connect to the Search Technologies Maven repository.

See Connecting to the Search Technologies Maven Repository for more information.

Run the Aspire Distribution Archetype

Execute the following command in some directory: (yes, I know it's a long one)

Execute this command exactly as specified. Maven will then prompt you for all of the parameters it needs (see below).

For Version 3.1:

```
 mvn archetype:generate -DarchetypeGroupId=com.searchtechnologies.aspire -DarchetypeArtifactId=aspire-distribution-archetype -DarchetypeVersion=3.1 -DrepositoryId=stPublic
```

Note: If the above doesn't work, try adding the following to the "mvn" command:

```
-DarchetypeRepository=http://repository.searchtechnologies.com/artifactory/public
```

Parameters:

Maven will prompt you for several input parameters:

Parameter	Description
groupId	The Maven group-id for your new distribution. Typically something like "com.searchtechnologies.aspire" or "com.your-customer-name-here". Should be unique for your organization across all organizations in the world.
artifactId	The Maven artifact-id for this project. This is the main Maven identifier for this distribution project within the GroupID. Typically something like "aspire-customer-distribution".
version	The version number for your new distribution.
mavenPassword	This will be the password that you specified when you registered for downloading Aspire. It will be used by Aspire to communicate with Maven.
mavenUser	This will be the username that you specified when you registered for downloading Aspire.

Once you have entered all of the parameters, you will be prompted to verify your selections. Enter 'Y', and then Maven will create your new stage for you.

What you have created is a project for building Aspire distributions. You can modify this project as needed for your particular customer's needs.

Run the Distribution Project

After the distribution archetype is complete, it will come with a project file, a distribution descriptor, and a bunch of (mostly) empty directories.

You will need to execute Maven within your new distribution project to actually build the distribution, as follows:

```
  cd {distribution-name}
  mvn clean package assembly:assembly
```

or (for recent versions of Aspire)

```
 cd {distribution-name}
 mvn clean package
```

Where {distribution-name} should be replaced with the distribution's "artifactId" that you entered above. See here for a description of the messages you might see when building the distribution.

If you ever need to create a new distribution with fresh JARs from the repository, you would re-execute "mvn -U clean package assembly:assembly" to get all of the latest code (for the revision numbers that are have specified in your pom.xml).

You should now have a ready-to-go Aspire distribution. See Launch Control for instructions on how to start up Aspire and try your new distribution.

Modify Your Distribution

Add New or Updated System Configuration Files

New Application files should be added to the "distribution-files/config" directory within your Maven distribution project.

These files will be automatically added to new distributions built with the Distribution project.

Important Note: *DON'T FORGET TO UPDATE YOUR distribution-files/config BEFORE REBUILDING YOUR DISTRIBUTION.*

If you do forget, you may accidentally delete all of your configuration file changes when the target directory is removed during a "mvn clean package".

Change the Settings File

The settings.xml file is located in the "distribution-files/config" directory. It can be edited as needed.

Add New Components

With the new Maven repository feature in version 0.4 and above, there is no longer any need to modify the distribution.xml file to add new components. Instead, new components can simply be added to the Aspire System configuration file and they will be automatically downloaded as needed (into your machine's Maven local repository) and loaded into Aspire.

AppBundles

If you're creating an AppBundles from this distribution, see Build and Deploy AppBundles for details

Create New Distribution Profiles

You can take advantage of Maven profiles to have a different builds for each environment where you need to run Aspire (dev, qa, production, staging).

You could have, for instance, a different set of files and a different set of configuration settings for each environment, and with a single instruction compile the correct distribution type.

For example, you may want to have different user names and password for your development and production environment. In this case, you could have two "settings.xml" files, each one with the proper configuration. When building the distribution, you specify the profile ("qa", "dev"), and the correct "settings.xml" file will be copied to target/distribution-name, ready to be run.

Steps are:

1. Add a new Maven profile in the pom.xml file of your distribution project. Make sure the profile has a profile/id element.

Example:

```
  <profile>
     <id>test-profile</id>
     <activation>
       <activeByDefault>false</activeByDefault>
     </activation>
     <properties>
       <distribution.file>test.xml</distribution.file>
       <distribution.useId>false</distribution.useId>
     </properties>
  </profile>
```

Notice that the ID of this profile is "test-profile". This will be needed later when building your distribution for test profile (other IDs may be "qa-distribution", "dev-distribution").

Also, notice that the new profile is not active by default (<activeByDefault>false</activeByDefault). This will let you have more than one profile and select the correct one when building the distribuotion project.

2. Create a new file test.xml on the root of your distribution project. This file is a Maven Assembly Descriptor file.

This file controls what files or folders are used to compile your distribution.

Example:

```
<assembly>
 <id>distribution</id>
 <formats>
   <format>dir</format>
 </formats>
 <includeBaseDirectory>false</includeBaseDirectory>
 <dependencySets>
   <dependencySet>
     <outputFileNameMapping>${artifact.artifactId}-${artifact.baseVersion}${dashClassifier?}.${artifact.
extension}</outputFileNameMapping>
     <unpack>false</unpack>
     <useTransitiveDependencies>false</useTransitiveDependencies>
```

```
    <outputDirectory>bundles/aspire</outputDirectory>
    <includes>
      <include>com.searchtechnologies.aspire:*</include>
    </includes>
    <excludes>
      <exclude>${artifact.groupId}:${artifact.artifactId}:jar:${artifact.baseVersion}</exclude>
    </excludes>
  </dependencySet>
  <dependencySet>
    <outputFileNameMapping>felix.jar</outputFileNameMapping>
    <unpack>false</unpack>
    <useTransitiveDependencies>false</useTransitiveDependencies>
    <outputDirectory>bin</outputDirectory>
    <includes>
      <include>org.apache.felix:org.apache.felix.main</include>
    </includes>
  </dependencySet>
  <dependencySet>
    <outputFileNameMapping>${artifact.artifactId}${dashClassifier?}.${artifact.extension}<
/outputFileNameMapping>
    <unpack>false</unpack>
    <useTransitiveDependencies>false</useTransitiveDependencies>
    <outputDirectory>bundles/system</outputDirectory>
    <includes>
      <include>org.apache.felix:*</include>
      <include>commons-fileupload:*</include>
      <include>commons-io:*</include>
      <include>org.apache.geronimo.bundles:*</include>
    </includes>
    <excludes>
      <exclude>org.apache.felix:org.apache.felix.main</exclude>
    </excludes>
  </dependencySet>
 </dependencySets>
 <fileSets>
  <fileSet>
    <directory>distribution-files/bin</directory>
    <outputDirectory>bin</outputDirectory>
  </fileSet>
  <fileSet>
    <directory>distribution-files/bundles/system</directory>
    <outputDirectory>bundles/system</outputDirectory>
  </fileSet>
  <fileSet>
    <directory>distribution-files/config</directory>
    <outputDirectory>config</outputDirectory>
  </fileSet>
  <fileSet>
    <directory>distribution-files/data</directory>
    <outputDirectory>data</outputDirectory>
  </fileSet>
  <fileSet>
    <directory>distribution-files/lib</directory>
    <outputDirectory>lib</outputDirectory>
  </fileSet>
  <fileSet>
    <directory>distribution-files/log</directory>
    <outputDirectory>log</outputDirectory>
  </fileSet>
  <fileSet>
    <directory>distribution-files/resources</directory>
    <outputDirectory>resources</outputDirectory>
  </fileSet>
  <fileSet>
    <directory>distribution-files/web</directory>
    <outputDirectory>web</outputDirectory>
  </fileSet>
 </fileSets>
</assembly>
```

3. Modify text.xml as needed.

For example, if you want to add a new folder to the compiled distribution:

```
  <fileSet>
    <directory>distribution-files/mynewfolder</directory>
    <outputDirectory>mynewfolder</outputDirectory>
  </fileSet>
```

Or if you want to get a different config folder (assuming you have "dev-config" and "qa-config" folders on the directory "distribution-files"):

```
 <fileSet>
     <directory>distribution-files/qa-config</directory>
     <outputDirectory>config</outputDirectory>
</fileSet>
```

You can have as many profiles as you need, and as many Assembly descriptor files. You may reuse the same descriptor for multiple profiles, according to your needs.

More information can be found on Maven Documentation:

- Maven Profiles
- Maven Assembly Plugin