Maven Repository

Overview

As a Maven repository, Artifactory is both a source for artifacts needed for a build, and a target to deploy artifacts generated in the build process. Maven is configured using a *settings.xml* file located under your Maven home directory (typically, this will be */user.home/.m2/settings.xml*). For more information on configuring Maven please refer to the Apache Maven Project Settings Reference.

The default values in this file configure Maven to work with a default set of repositories used to resolve artifacts and a default set of plugins.

To work with Artifactory you need to configure Maven to perform the following two steps:

1. Resolve artifacts through Artifactory
2. Deploy artifacts to repositories through Artifactory

Once your Maven build is configured, Artifactory also provides tight integration with commonly used CI servers (such as Jenkins, TeamCity or a Bamboo) through a set of plugins that you can freely install and use.

Page Contents

Read more

- Maven Artifactory Plugin

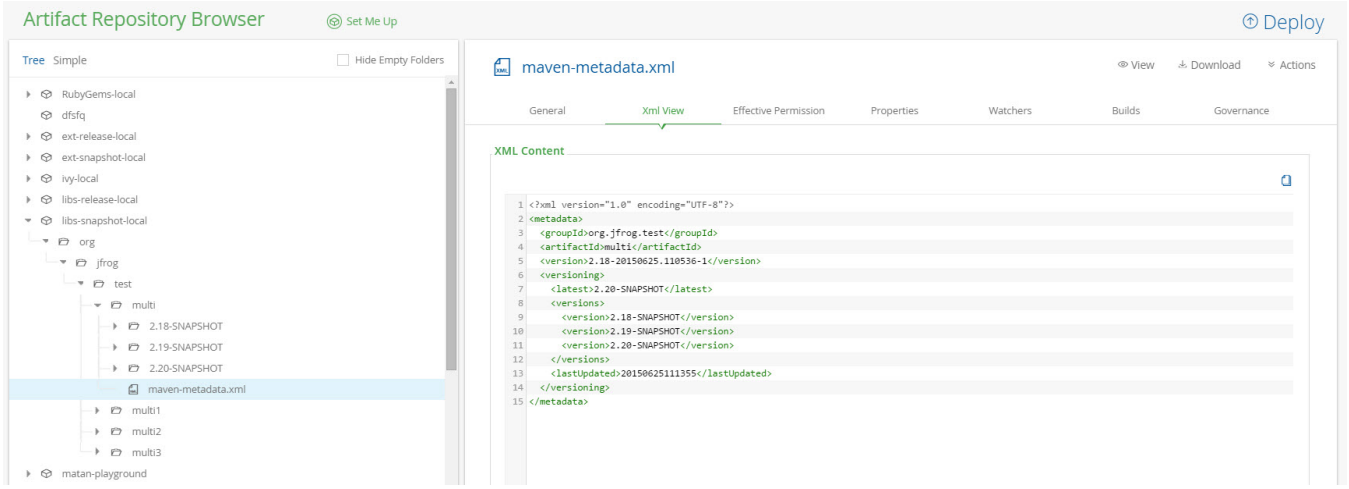
Integration Benefits

JFrog Artifactory and Maven Repositories

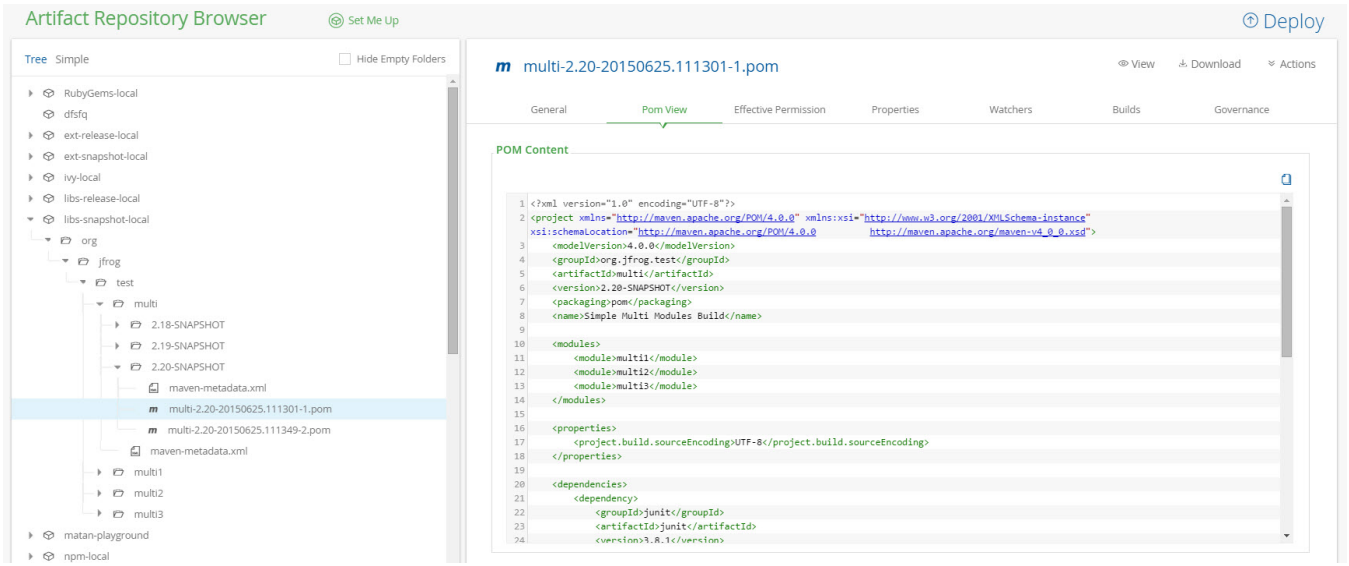
Viewing Maven Artifacts

If you select a Maven metadata file (maven-metadata.xml) or a POM file (pom.xml) in the Tree Browser, Artifactory provides corresponding tabs allowing you to view details on the selected item.

Maven Metadata View



POM View



Resolving Artifacts through Artifactory

To configure Maven to resolve artifacts through Artifactory you need to modify the *settings.xml*. You can generate one automatically, or modify it manually.

Automatically Generating Settings

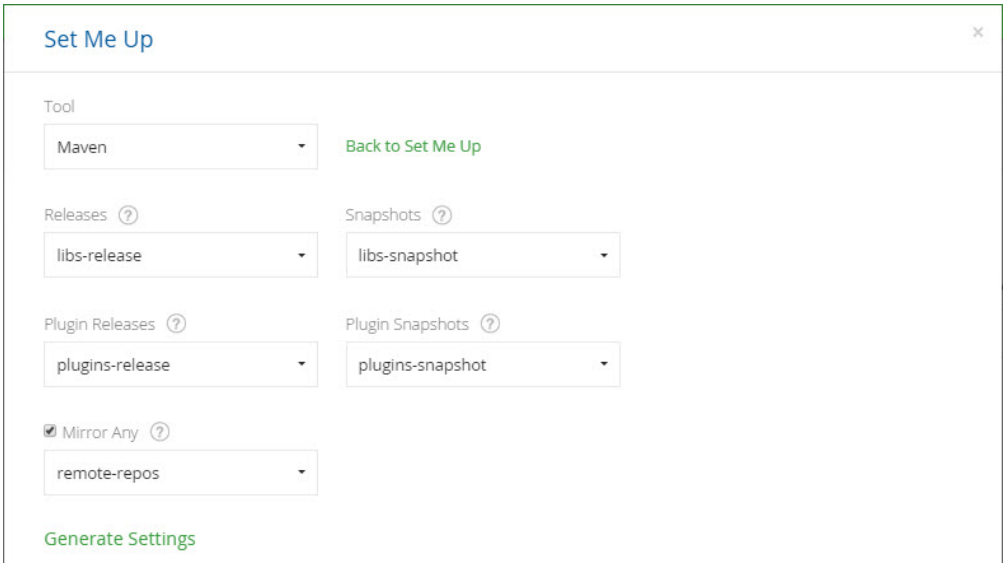
To make it easy for you to configure Maven to work with Artifactory, Artifactory can automatically generate a *settings.xml* file which you can save under your Maven home directory.

The definitions in the generated *settings.xml* file override the default **central** and **snapshot** repositories of Maven.

In the **Artifact Repository Browser** of the **Artifacts** module, select **Set Me Up.** In the **Set Me Up** dialog, set **Maven** in the **Tool** field and click "Generate Maven Settings". You can now specify the repositories you want to configure for Maven.

Releases	The repository from which to resolve releases
Snapshots	The repository from which to resolve snapshots
Plugin Releases	The repository from which to resolve plugin releases

Plugin Snapshots	The repository from which to resolve plugin snapshots
Mirror Any	When set, you can select a repository that should mirror any other repository. For more details please refer to Additional **Mirror Any** Setup



Once you have configured the settings for Maven you can click "Generate Settings" to generate and save the *settings.xml* file.

Provisioning Dynamic Settings for Users

You can deploy and provision a dynamic settings template for your users.

Once downloaded, settings are generated according to your own logic and can automatically include user authentication information.

For more details, please refer to the Provisioning Build Tool Settings under Filtered Resources.

Manually Overriding the Built-in Repositories

To override the built-in **central** and **snapshot** repositories of Maven, you need to ensure that Artifactory is correctly configured so that no request is ever sent directly to them.

Using the automatically generated file as a template

You can use the automatically generated *settings.xml* file as an example when defining the repositories to use for resolving artifacts.

To do so, you need to insert the following into your parent POM or *settings.xml* (under an active profile):

```
<repositories>
    <repository>
        <id>central</id>
        <url>http://[host]:[port]/artifactory/libs-release</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </repository>
    <repository>
        <id>snapshots</id>
        <url>http://[host]:[port]/artifactory/libs-snapshot</url>
        <releases>
            <enabled>false</enabled>
        </releases>
    </repository>
</repositories>
<pluginRepositories>
    <pluginRepository>
        <id>central</id>
        <url>http://[host]:[port]/artifactory/plugins-release</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </pluginRepository>
    <pluginRepository>
        <id>snapshots</id>
        <url>http://[host]:[port]/artifactory/plugins-snapshot</url>
        <releases>
            <enabled>false</enabled>
        </releases>
    </pluginRepository>
</pluginRepositories>
```

Using the Default Global Repository

You can configure Maven to run with the Default Global Repository so that any request for an artifact will go through Artifactory which will search through all of the local and remote repositories defined in the system.

We recommend that you fine tune Artifactory to search through a more specific set of repositories by defining a dedicated virtual (or local) repository, and configure Maven to use that to resolve artifacts instead.

Additional Mirror Any Setup

In addition to overriding built-in Maven repositories, you can use the **Mirror Any** setting to redirect all requests to a Maven repository through Artifactory, including those defined inside POMs of plug-ins and third party dependencies. (While it does not adhere to best practices, it is not uncommon for POMs to reference Maven repositories directly). This ensures no unexpected requests directly to Maven are introduced by such POMs.

You can either check **Mirror Any** in the **Maven Settings** screen when generating your *settings.xml* file, or you can manually insert the following:

```
<mirrors>
    <mirror>
      <id>artifactory</id>
      <mirrorOf>*</mirrorOf>
      <url>http://[host]:[port]/artifactory/[virtual repository]</url>
      <name>Artifactory</name>
    </mirror>
</mirrors>
```

Care when using "Mirror Any"

While this is a convenient way to ensure Maven only accesses repositories through Artifactory, it defines a coarse proxying rule that does not differentiate between releases and snapshots and relies on the single specified repository to do this resolution.

Using Mirrors

For more information on using mirrors please refer to Using Mirrors for Repositories in the Apache Maven documentation.

Configuring Authentication

Artifactory requires user authentication in three cases:

- Anonymous access has been disabled by unchecking the global **Allow Anonymous Access** setting.
- You want to restrict access to repositories to a limited set of users
- When deploying builds (while theoretically possible, it is uncommon to allow anonymous access to deployment repositories)

Authentication is configured in Maven using `<server>` elements in the `settings.xml` file.

Each <repository> and <mirror> element specified in the file must have a corresponding <server> element with a matching <id> that specifies the username and password.

The sample snippet below emphasizes that the `<repository>` element with `id=central` has a corresponding `<server>` element with `id=central`.

Similarly, the `<repository>` element with `id=snapshots` has a corresponding `<server>` element with `id=snapshots`.

The same would hold for <mirror> elements that require authentication.

In both cases the username is `admin` and the password is encrypted.

```
...

<servers>

        <server>

                <id>central</id>
                <username>admin</username>
                <password>\{DESede\}kFposSPUydYZf89Sy/o4wA==</password>

        </server>

        <server>

                <id>snapshots</id>
                <username>admin</username>
                <password>\{DESede\}kFposSPUydYZf89Sy/o4wA==</password>

        </server>


</servers>

<profiles>

        <profile>

                <repositories>

                        <repository>

                                <id>central</id>
                                <snapshots>
                                <enabled>false</enabled>
                                </snapshots>
                                <name>libs-release</name>
                                <url>http://localhost:8081/artifactory/libs-release</url>

                        </repository>
                        <repository>

                                <id>snapshots</id>
                                <snapshots />
                                <name>libs-snapshot</name>
                                <url>http://localhost:8081/artifactory/libs-snapshot</url>

                        </repository>

                </repositories>

        </profile>

</profiles>

...
```

Artifactory encrypts passwords for safe and secure access to Maven repositories

To avoid having to use cleartext passwords, Artifactory encrypts the password in the settings.xml file that is generated. For example, in the above sample snippet we can see that the admin user name is specified in cleartext, but the password is encrypted:

```
<username>admin</username>
<password>\{DESede\}kFposSPUydYZf89Sy/o4wA==</password>
```

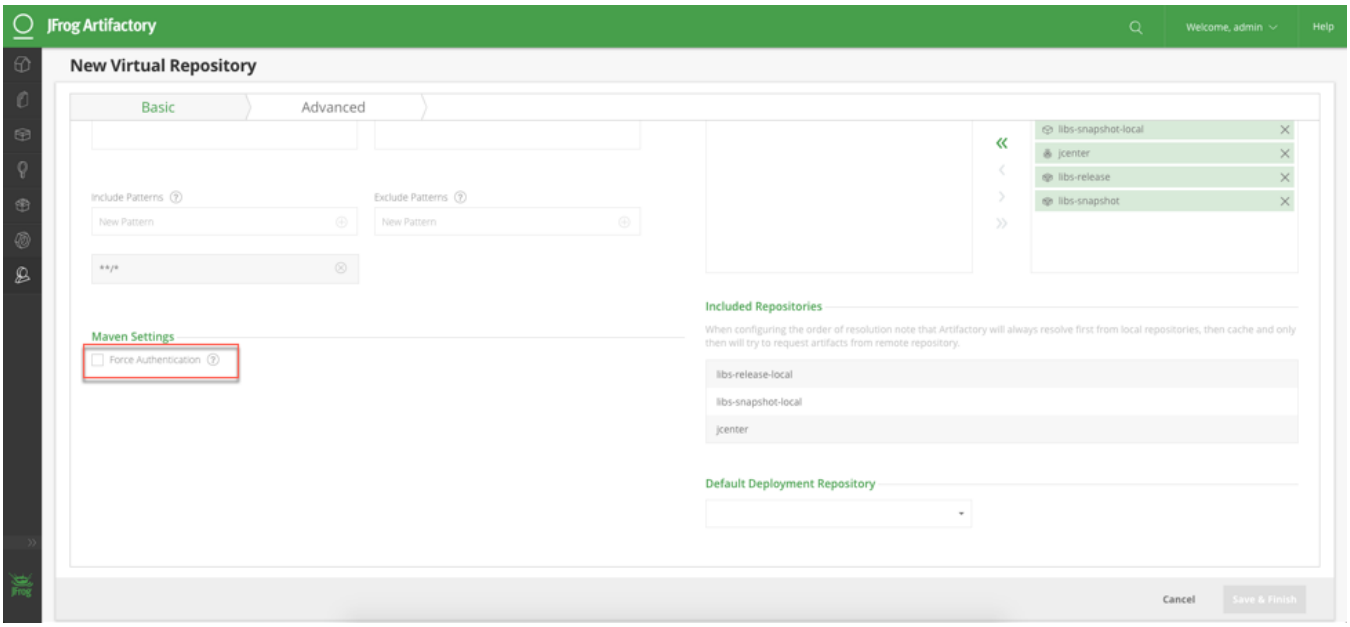
Synchronizing authentication details for repositories with the same URL

If you have repository definitions (either for deployment or download) that use *the same URL*, Maven takes the authentication details (from the corresponding server definition) of the first repository encountered and uses it for the life-time of the running build for all repositories with the same URL. This may cause authentication to fail (producing 401 errors for downloads or deployment) if you are using different authentication details for the respective repositories. This is inherent Maven behavior and can only be solved by using the same authentication details for all repository definitions with the same URL in your *settings.xml*.

Forcing Authentication on Virtual Maven Repositories

Artifactory supports Maven repositories with Allow Anonymous Access enabled by default and will not query the Maven client for authentication parameters.
If you want to enforce authentication you need to explicitly instruct Artifactory to request authentication parameters.

In the New or Edit Repository dialog of the Maven virtual repositories, select the **Force Authentication** check box.



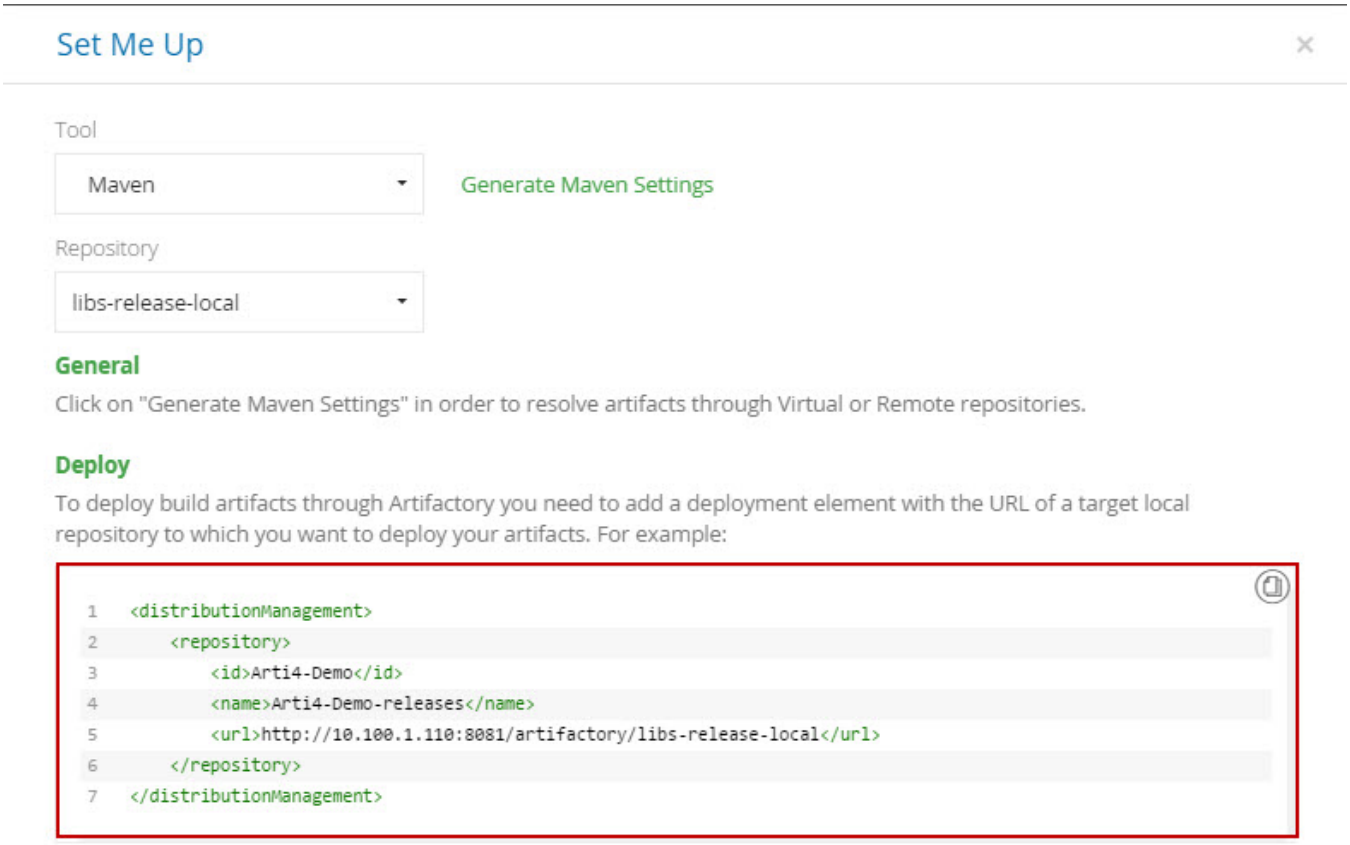
When 'Force Authentication' is selected, Artifactory will request authentication parameters from the Maven client before trying to access this repository.

Deploying Artifacts Through Artifactory

Setting Up Distribution Management

To deploy build artifacts through Artifactory you must add a deployment element with the URL of a target local repository to which you want to deploy your artifacts.

To make this easier, Artifactory displays a code snippet that you can use as your deployment element. In the **Artifacts** module **Tree Browser** select the repository you want to deploy to and click **Set Me UP.** The code snippet is displayed under **Deploy.**



Remember that you can not deploy build artifacts to remote, so you should not use them in a deployment element.

Setting Up Security in Maven Settings

When deploying your Maven builds through Artifactory, you must ensure that any `<repository>` element in your distribution settings has a corresponding `<server>` element in the *`settings.xml`* file with a valid username and password as described in Configuring Authentication above. For the example displayed above, the Maven client expects to find a `<server>` element in the *`settings.xml`* with *`<id>artifactory</id>`* specified.

Anonymous access to distribution repository

If anonymous access to your distribution repository is allowed then there is no need to configure authentication. However, while it is technically possible, this is not good practice and is therefore an unlikely scenario

Watch the Screencast