Repository Layouts

Overview

Together with the growing number of choices for build-tools and frameworks there are also many ways in which modules can be stored within a repository.

Initially, Artifactory supported the Maven layout conventions for dealing with modules (and relying on Maven-specific metadata). However, your build tool should be able to "talk" to the repository "naturally", so if you are using Ivy or Gradle, there is no need to configure them to use the Maven conventions in order to "fit in". Moreover, combining and chaining repositories that use different layouts should work out-of-the-box.

This is where the Repository Layouts Add-on comes into play!

Page Contents

The Freedom of Custom Layouts

With the Repository Layouts Add-on, Artifactory allows you to take full control over the layout used by each repository and uses layout definitions to identify module artifacts and descriptors. By using repository layouts, Artifactory offers these smart module management facilities for any build technology:

- Automatic snapshot/integration versions cleanup
- Deleting old versions
- Conversions between remote and local layouts
- Conversions between 2 local layouts when moving or copying
- Resolution conversions from a virtual repository to its underlying repositories (where the virtual repository has its own layout defined)

Bundled Layouts

Artifactory comes out-of-the-box with a number of default, predefined layouts requiring no additional configuration:

- **Maven 2/3**
- **Ivy** (default layout)
- **Gradle** (Wharf cache default layout)
- **Maven 1**

Support for repository layouts in Artifactory OSS

Layout configuration for conversion and resolution is available only to Artifactory Power Pack users. Users of the OSS version can only Configure their Repositories to use the default repository layouts bundled with Artifactory.

The OSS version only supports the automatic snapshot/integration version cleanup and deleting old version features.

Modules and Path Patterns used by Repository Layouts

Module Fields

To support smart module management, Artifactory must construct module information for stored files. Artifactory constructs this information based on path pattern information defined as part of the Repository Layout configuration (detailed below).

A module is comprised of various sub-elements or fields, which are typically expressed in the path of a stored artifact.

The module-sub elements recognized by Artifactory are listed below. At a minimum, there are three mandatory fields required for module identification:

- Organization
- Module
- Base Revision.

Field	Description	Example	Mandatory
Organization	A sequence of literals that identifies the artifact's organization	"`org.slf4j`"	✓
Module	A sequence of literals that identifies the artifact's module	"`slf4j-api`"	✓
Base Revision	A sequence of literals that identifies the base revision part of the artifact version, excluding any integration information	"`1.5.10`", or in case of an integration revision "`1.2-SNAPSHOT`" the base revision is "`1.2`"	✓
Folder Integration Revision	A sequence of literals that identifies the integration revision part used in folder names in the artifact's path, excluding the base revision	in case of an integration revision "`1.2-SNAPSHOT`" the folder integration revision is "`SNAPSHOT`"	✗
File Integration Revision	A sequence of literals that identifies the integration revision part in the artifact's file name, excluding the base revision	in case of an integration revision "`1.2-20110202.144533-3`" the file integration revision is "`20110202.144533-3`"	✗
Classifier	A sequence of literals that identifies the artifact's classifier	"`sources`"	✗
Extension	A sequence of literals that identifies the artifact's extension	"`zip`"	✗
Type	A sequence of literals that identifies the artifact's type. Typically used when the artifact's extension cannot be reused as the artifact's type	"`java-source`"	✗

Using Module Fields to Define Path Patterns

A path pattern is used in the configuration of a Repository Layout.

The pattern is similar to that of the Ivy pattern and is used to define a convention for artifact resolution and publication paths.

Artifactory uses path patterns to construct module information for stored files. This module information is then used to facilitate all the features mentioned above (version cleanup, cross-repo path conversions, etc.).

Path Pattern Tokens

A path pattern is constructed of tokens (explained below), path separators ('`/`'), optional parentheses ('`(`' and '`)`') and literals ('`.`', '`-`', etc.). Tokens are modeled after module fields, as presented above.

Path patterns can be defined for every artifact in the repository or you can define a separate path patterns for descriptor-type artifacts (such as, a `.pom` or an `ivy.xml` file).

The following tokens are available:

`[org]`	Represents the ***Organization*** field where the levels are separated by dots ('`.`'), a-la Ivy. For example: "`org.slf4j`"
`[orgPath]`	Represents the ***Organization*** field where the levels are separated by path separators ('`/`'), a la Maven. For example: "`org/slf4j`"
`[baseRev]`	Represents the ***Base Revision*** field

`[module]`	Represents the ***Module*** field
`[folderItegRev]`	Represents the ***Folder Integration Revision*** field
`[fileItegRev]`	Represents the ***File Integration Revision*** field
`[classifier]`	Represents the ***Classifier*** field
`[ext]`	Represents the ***Extension*** field
`[type]`	Represents the ***Type*** field
`[customTokenName<customTokenRegex>]`	A custom token. Can be used to create a new type of token when the provided defaults are insufficient. For example, `[myIntegRev<ITEG-(?:[0-9]+)>]` creates a new custom token named `myIntegRev` that matches the word `ITEG` followed by a dash and a minimum of one digit.

Custom tokens

When using custom tokens in the repository layout, make sure that the layout begins with **[orgPath]/[module].** If the [module] token is missing in the layout, some REST API calls will not work.

Multiple Custom Tokens

Artifactory supports any number of custom tokens, but when provided with multiple custom tokens of the same key, Artifactory only takes into account the regular expression of the first occurrence while substituting the rest with a repetition expression (even if each occurrence has a different regular expression value.

For example:

```
[custom1<.+>]/[custom1<.*>]/[custom1<[0-9]+>]
```

Translates to:

```
<custom1>.+/\1/\1
```

Optional parts

To specify tokens or a sequence of tokens and literals as optional in the path pattern, surround the sequence with the optional parentheses '(' and ')' literals.

For example, the pattern "`[module](-[classifier])`" matches both "`bobs-tools-sources`" and "`bobs-tools`", and the pattern "`[baseRev](-[fileItegRev])`" matches both "`1.2-SNAPSHOT`" and "`1.2`".

Artifact Path Patterns

An artifact path pattern represents the typical structure that all module artifacts are expected to be stored in.

For example,

- To represent a normal Maven artifact path: "`org/eclipse/jetty/jetty-ajp/7.0.2.v20100331/jetty-ajp-7.0.2.v20100331.jar`"

  Use the artifact path pattern:

  ```
  [orgPath]/[module]/[baseRev](-[folderItegRev])/[module]-[baseRev](-[fileItegRev])(-[classifier]).[ext]
  ```

- To represent a default Ivy artifact path: "`org.eclipse.jetty/jetty-ajp/7.0.2.v20100331/jars/jetty-ajp-7.0.2.v20100331.jar`"

  Use the artifact path pattern:

```
[org]/[module]/[baseRev](-[folderItegRev])/[type]s/[module]-[baseRev](-[fileItegRev])(-[classifier]).
[ext]
```

Descriptor Path Patterns

A descriptor path pattern is used to recognize descriptor files (like `.pom` or `ivy.xml` files).

Using a specific descriptor path pattern is optional. When not used, Artifactory constructs module information for descriptor files using the artifact path pattern.

Even though descriptor paths patterns are optional, usage of them is ***highly recommended*** in cases of distinctive descriptors, such as Ivy `ivy_-1.0.xml` and Maven `bobs-tools-1.0.pom`.

For example,

- To represent a normal Maven descriptor path: "`org/eclipse/jetty/jetty-ajp/7.0.2.v20100331/jetty-ajp-7.0.2.v20100331.pom`"

  Use the descriptor path pattern:

  ```
  [orgPath]/[module]/[baseRev](-[folderItegRev])/[module]-[baseRev](-[fileItegRev])(-[classifier]).pom
  ```

- To represent a default Gradle descriptor path: "`org.eclipse.jetty/jetty-ajp/ivy-7.0.2.v20100331.xml`"

  Use the descriptor path pattern:

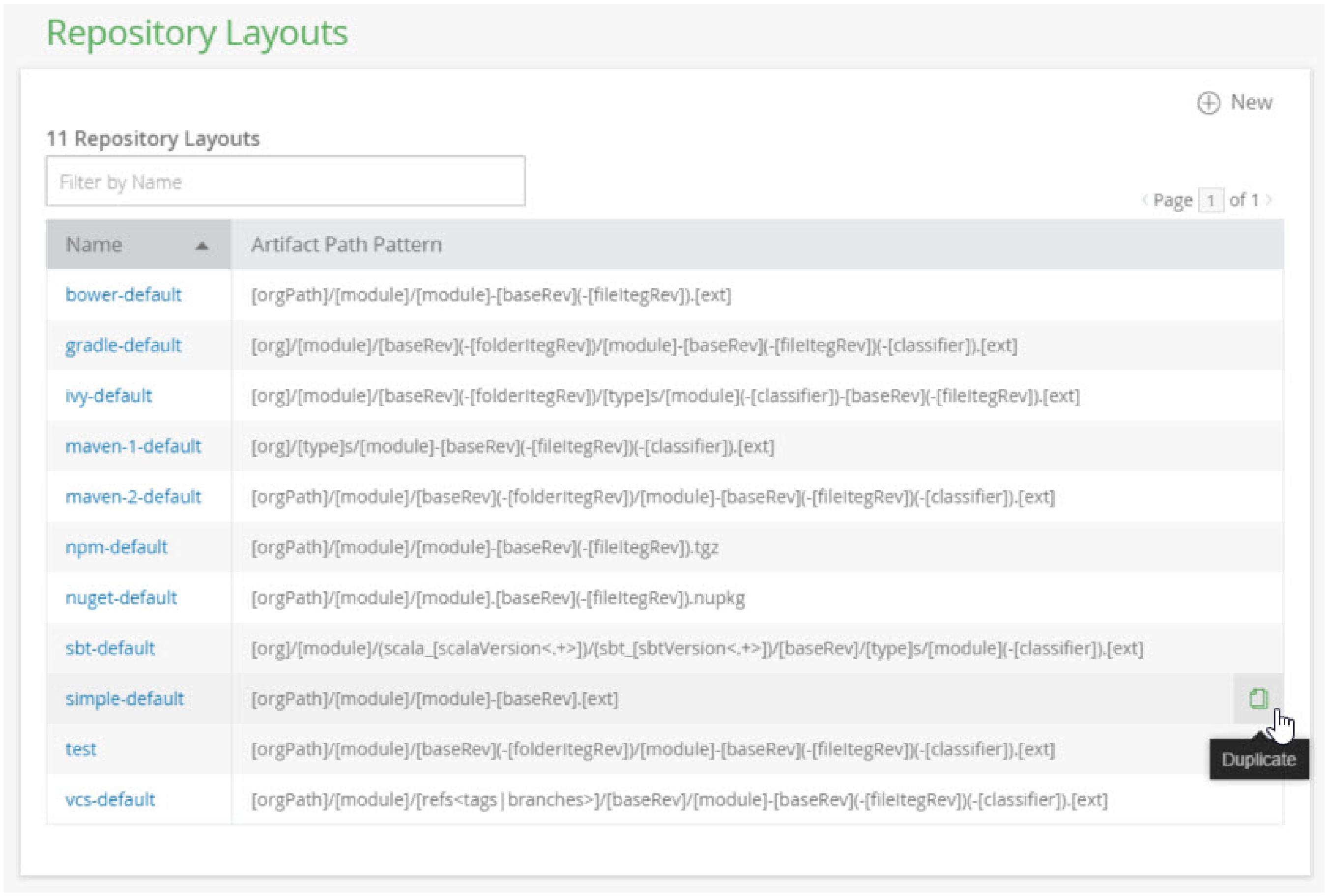
  ```
  [org]/[module]/ivy-[baseRev](-[fileItegRev]).xml
  ```

Configuration

Repository layouts are configured on the global level of your Artifactory instance, so that any layout can be shared and reused across any number of repositories.

Layout Configuration

Layout configuration is available to administrator users in the **Admin** module under **Repositories | Layouts**.

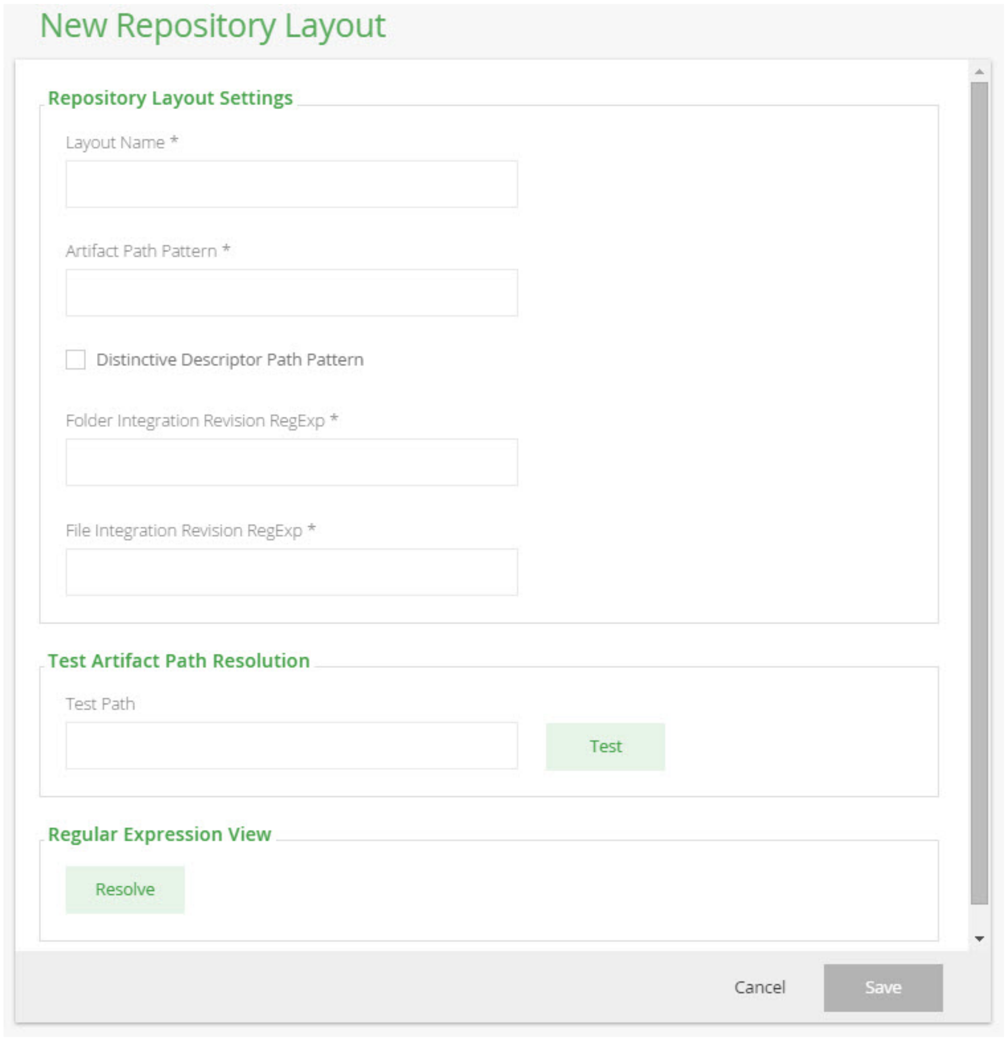


Repository Layouts

11 Repository Layouts

Name	Artifact Path Pattern
bower-default	[orgPath]/[module]/[module]-[baseRev](-[fileItegRev]).[ext]
gradle-default	[org]/[module]/[baseRev](-[folderItegRev])/[module]-[baseRev](-[fileItegRev])(-[classifier]).[ext]
ivy-default	[org]/[module]/[baseRev](-[folderItegRev])/[type]s/[module](-[classifier])-[baseRev](-[fileItegRev]).[ext]
maven-1-default	[org]/[type]s/[module]-[baseRev](-[fileItegRev])(-[classifier]).[ext]
maven-2-default	[orgPath]/[module]/[baseRev](-[folderItegRev])/[module]-[baseRev](-[fileItegRev])(-[classifier]).[ext]
npm-default	[orgPath]/[module]/[module]-[baseRev](-[fileItegRev]).tgz
nuget-default	[orgPath]/[module]/[module].[baseRev](-[fileItegRev]).nupkg
sbt-default	[org]/[module]/(scala_[scalaVersion<.+>])/(sbt_[sbtVersion<.+>])/[baseRev]/[type]s/[module](-[classifier]).[ext]
simple-default	[orgPath]/[module]/[module]-[baseRev].[ext]
test	[orgPath]/[module]/[baseRev](-[folderItegRev])/[module]-[baseRev](-[fileItegRev])(-[classifier]).[ext]
vcs-default	[orgPath]/[module]/[refs<tags|branches>]/[baseRev]/[module]-[baseRev](-[fileItegRev])(-[classifier]).[ext]

Additional layouts can be created from scratch by clicking "New" or by duplicating an existing layout.


New Repository Layout
Repository Layout Settings
Layout Name *
Artifact Path Pattern *
Distinctive Descriptor Path Pattern
Folder Integration Revision RegExp *
File Integration Revision RegExp *
Test Artifact Path Resolution
Test Path
Test
Regular Expression View
Resolve
Cancel
Save

Testing Layouts

Once you have finished configuring your layout, you can test it on an artifact path, and see how Artifactory would build module information from the path, using the layout definitions.

Path Patterns

These are used to define the artifact path pattern and the descriptor path pattern (optional), as explained above.

Use patterns in the directory part of the path

To achieve best path matching results, it is highly recommended that artifact and descriptor patterns also contain the mandatory tokens (`[org]` or `[orgPath]`, `[module]` and `[baseRev]`) within the directory structure itself.
For example, Gradle's artifact path pattern:

```
[org]/[module]/[baseRev](-[folderItegRev])/[module]-[baseRev](-[fileItegRev])(-[classifier]).[ext]
```

Regular Expressions for File and Folder Integration Revision

These fields should contain regular expressions that exactly match and describe the integration revision (excluding the base revision) formats as they are expected in the artifact's file name and path-structure folder name.

Avoid using capturing group syntax in regexp

Regular expressions entered in these fields are wrapped and treated as a single capturing group.

Refrain from introducing any capturing groups within the expressions. Failure to do so may result in unexpected behavior and compromise the accuracy of the matcher.

Folder integration revision regular expression examples:

- Maven's folder integration revision is simply the constant `-SNAPSHOT`appended to the base revision ("1.2-SNAPSHOT"), so the regular expression is

```
SNAPSHOT
```

- Ivy's default folder integration revision is usually equal to the file integration revision and is normally a 14 digit timestamp ("5.1-20101202161531"), so the regular expression can be

```
\d{14}
```

File integration revision regular expression examples:

- Maven's file integration revision can be either the `-SNAPSHOT`constant ("1.2-SNAPSHOT") or a timestamp, where the date and time are separated by a dot ('.'), with an addition of a dash ('-') and a build-number ("2.3-20110108.100922-2"), so the regular expression should be able to fit them both

```
SNAPSHOT|(?:(?:\d{8}.\d{6})-(?:\d+))
```

- Ivy's default file integration revision is is normally a 14 digit timestamp ("5.1-20101202161531") and usually equal to the folder integration revision, so the regular expression may be the same as suggested in the file's example

```
\d{14}
```

Repository Configuration

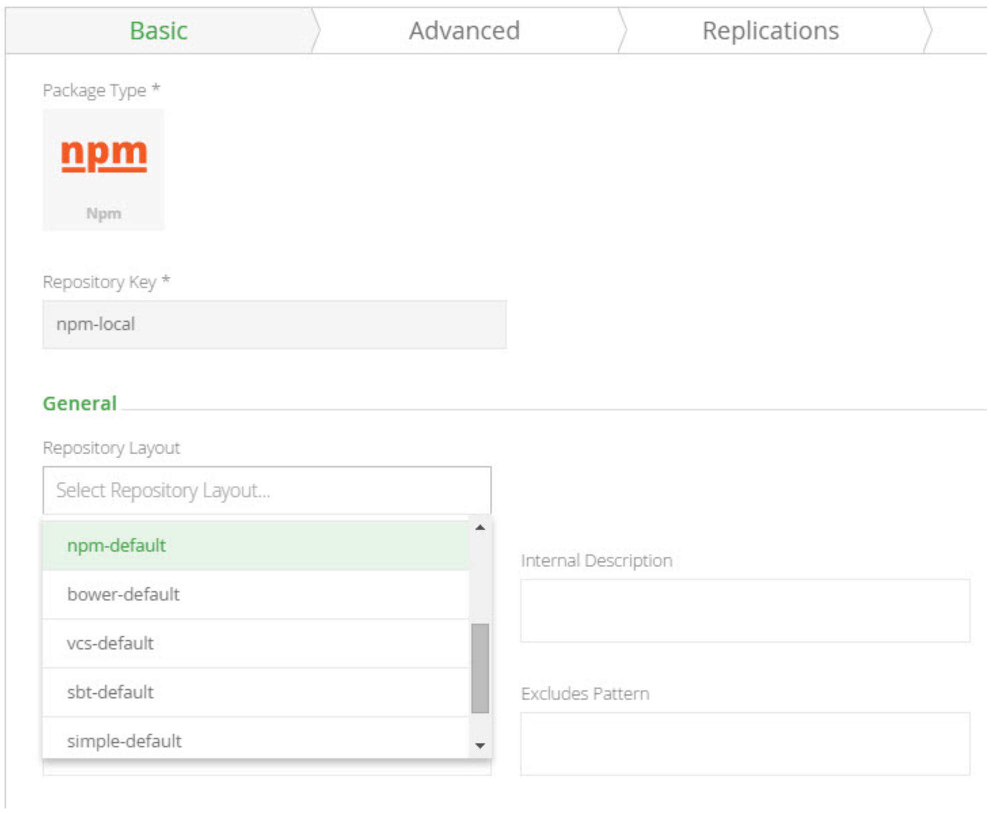
Before custom layouts

Repositories created prior to the introduction of custom repository layouts are automatically configured with the default Maven 2 layout.

Local Repository Configuration

Layouts are mandatory for local repositories, since they define the structure with which artifact are stored.

When you create a new repository, Artifactory will recommend the best layout according to the **Package Type** selected for the repository.

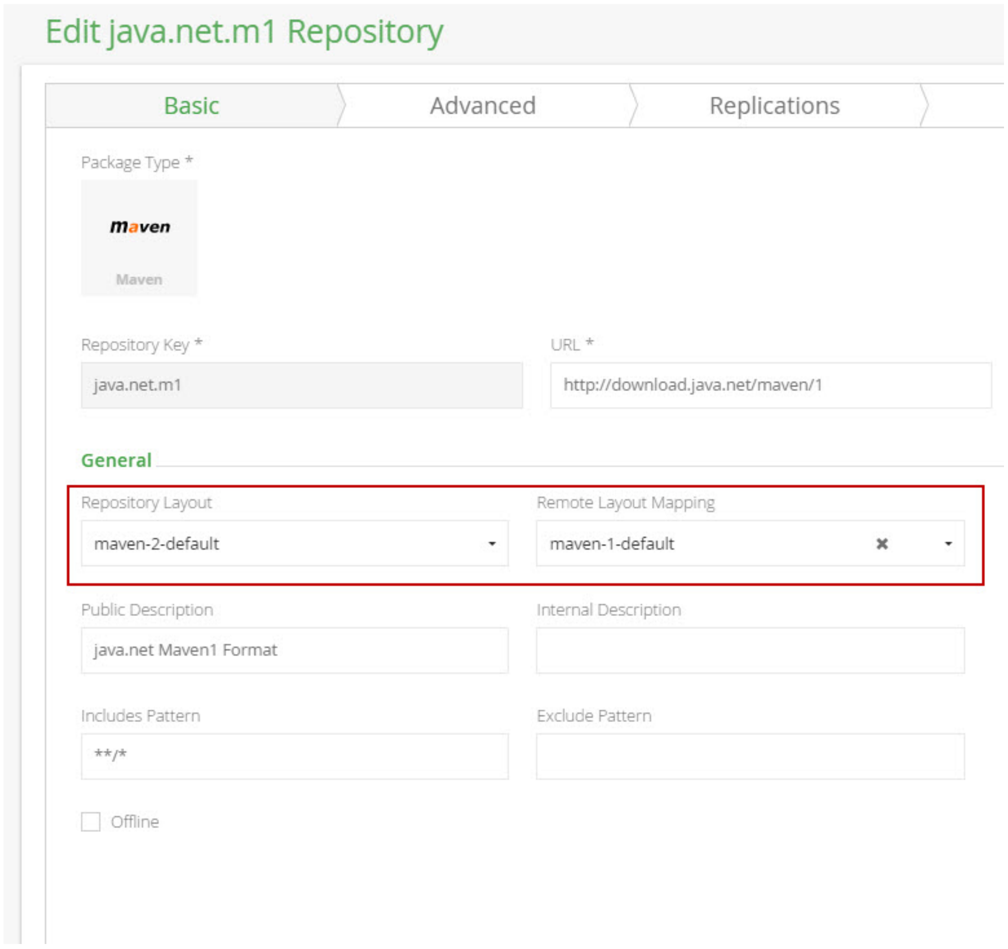


Remote Repository Configuration

Layouts are mandatory only for the remote repository cache configuration, however, you can also specify the layout of the remote repository itself.

If the remote repository itself uses a different layout than the one chosen for the cache, all requests the to the remote target are translated from the path of the cache layout to the path of the remote layout.

For example, the remote repository `http://download.java.net/maven/1` stores its artifacts according to the Maven 1 convention. You can configure the cache of this repository to use the Maven 2 layout, but set the **R emote Layout Mapping** to Maven 1. This way, the repository cache handles Maven 2 requests and artifact storage, while outgoing requests to the remote repository are translated to the Maven 1 convention.


Edit java.net.m1 Repository

Package Type *

Repository Key *: java.net.m1

URL *: http://download.java.net/maven/1

General

Repository Layout: maven-2-default

Remote Layout Mapping: maven-1-default

Public Description: java.net Maven1 Format

Internal Description:

Includes Pattern: **/*

Exclude Pattern:

Offline

Virtual Repository Configuration

You can also configure a layout for a virtual repository.

When configured, all resolution requests can be made according to the virtual repository layout. When trying to resolve requests to the virtual repository Artifactory attempts to translate the request path to the layout of each nested repository, according to the module information constructed from the virtual request.

In the following cases, the request path is not translated, and requests pass through to nested repositories with the original specified path:

- Module information cannot be constructed
- The virtual module information cannot be mapped to a nested repository (e.g., fields are missing on one of the sides)
- The virtual repository or the nested repository are not configured with a layout