

Artifactory

User Guide

1. Welcome to Artifactory	6
1.1 Installing Artifactory	9
1.1.1 System Requirements	12
1.1.2 Installing on Linux Solaris or Mac OS	13
1.1.3 Installing on Windows	20
1.1.4 Running with Docker	22
1.1.5 Changing the Default Storage	26
1.1.5.1 MySQL	29
1.1.5.2 Oracle	32
1.1.5.3 Microsoft SQL Server	33
1.1.5.4 PostgreSQL	37
1.1.6 Running Behind an HTTP Server	38
1.1.6.1 Apache HTTP Server	38
1.1.6.2 nginx	43
1.2 Upgrading Artifactory	45
1.3 Using Artifactory	50
1.3.1 General Information	52
1.3.2 Browsing Artifactory	53
1.3.3 Using WebDAV	58
1.3.4 Searching for Artifacts	59
1.3.5 Deploying Artifacts	66
1.3.6 Manipulating Artifacts	71
1.3.7 Updating Your Profile	77
1.3.8 Authentication	81
1.3.9 Artifactory REST API	82
1.3.9.1 Repository Configuration JSON	141
1.3.9.2 Security Configuration JSON	144
1.3.9.3 System Settings JSON	145
1.4 Configuring Artifactory	146
1.4.1 Configuring Repositories	151
1.4.1.1 Common Settings	154
1.4.1.2 Local Repositories	157
1.4.1.3 Remote Repositories	161
1.4.1.3.1 Managing Proxies	166
1.4.1.3.2 Advanced Settings	168

1.4.1.4 Smart Remote Repositories	173
1.4.1.5 Virtual Repositories	176
1.4.2 Configuring Security	181
1.4.2.1 Managing Users	183
1.4.2.2 Managing Permissions	189
1.4.2.3 Centrally Secure Passwords	192
1.4.2.4 Master Key Encryption	194
1.4.2.5 Managing Security with LDAP	196
1.4.2.6 Managing Security with Active Directory	199
1.4.2.7 Access Log	206
1.4.3 Mail Server Configuration	207
1.4.4 Configuration Files	209
1.4.5 Exposing Maven Indexes	213
1.4.6 Clustering Artifactory	215
1.5 System Monitoring and Maintenance	217
1.5.1 System Information	218
1.5.2 Monitoring Storage	218
1.5.3 Artifactory Log Files	221
1.5.4 Artifactory JMX MBeans	226
1.5.5 Regular Maintenance Operations	228
1.5.6 Managing Backups	232
1.5.7 Importing and Exporting	235
1.5.8 Managing Disk Space Usage	241
1.5.9 Getting Support	244
1.6 Artifactory High Availability	246
1.6.1 Installation and Setup	250
1.6.2 Managing the HA Cluster	257
1.6.3 Troubleshooting HA	259
1.7 Bintray Integration	262
1.7.1 Bintray info panel	262
1.7.2 Push to Bintray	263
1.7.3 Deploying Snapshots to oss.jfrog.org	272
1.8 Artifactory Pro	277
1.8.1 Artifactory Comparison Matrix	282
1.8.2 Build Integration	285
1.8.2.1 Jenkins (Hudson) Artifactory Plug-in	296
1.8.2.2 TeamCity Artifactory Plug-in	298
1.8.2.2.1 TeamCity Artifactory Plugin - Release Management ..	310
1.8.2.3 Bamboo Artifactory Plug-in	316
1.8.2.3.1 Bamboo Artifactory Plugin - Release Management ..	332
1.8.2.4 MSBuild Artifactory Plugin	339
1.8.3 Docker Repositories	348
1.8.3.1 Using Docker V1	368

1.8.4 Repository Replication	377
1.8.5 Artifactory Query Language	385
1.8.6 S3 Object Storage	402
1.8.7 User Plugins	406
1.8.8 NuGet Repositories	427
1.8.8.1 Microsoft Symbol Server	436
1.8.9 Npm Repositories	440
1.8.10 Bower Repositories	449
1.8.11 Git LFS Repositories	457
1.8.12 Vagrant Repositories	461
1.8.13 SBT Repositories	467
1.8.14 LDAP Groups	470
1.8.15 Atlassian Crowd Integration	472
1.8.16 Single Sign-on	475
1.8.17 SAML SSO Integration	477
1.8.18 OAuth Integration	481
1.8.19 VCS Repositories	491
1.8.20 YUM Repositories	499
1.8.21 Debian Repositories	510
1.8.22 RubyGems Repositories	517
1.8.23 PyPI Repositories	525
1.8.24 Properties	532
1.8.24.1 Using Properties in Deployment and Resolution	533
1.8.25 Smart Searches	535
1.8.26 Repository Layouts	538
1.8.27 License Control	547
1.8.28 Black Duck Code Center Integration	554
1.8.29 Filtered Resources	559
1.8.30 P2 Repositories	561
1.8.31 Watches	568
1.8.32 WebStart and Jar Signing	568
1.9 Working with Maven	571
1.9.1 Maven Artifactory Plugin	577
1.10 Working with Gradle	582
1.10.1 Gradle Artifactory Plugin	585
1.11 Working with Ivy	592
1.12 Troubleshooting	595
1.13 End of Life	596
1.14 Release Notes	598
1.14.1 Artifactory 4.3	599
1.14.2 Artifactory 4.2	600
1.14.2.1 Artifactory 4.2.1	601
1.14.2.2 Artifactory 4.2.2	602

1.14.3 Artifactory 4.1	602
1.14.3.1 Artifactory 4.1.2	603
1.14.3.2 Artifactory 4.1.3	603
1.14.4 Artifactory 4.0	604
1.14.4.1 Artifactory 4.0.1	605
1.14.4.2 Artifactory 4.0.2	606
1.14.5 Artifactory 3.9	606
1.14.5.1 Artifactory 3.9.1	607
1.14.5.2 Artifactory 3.9.2	608
1.14.6 Artifactory 3.8	608
1.14.7 Artifactory 3.7	609
1.14.8 Artifactory 3.6	610
1.14.9 Artifactory 3.5	611
1.14.9.1 Artifactory 3.5.3	612
1.14.9.2 Artifactory 3.5.2	613
1.14.9.3 Artifactory 3.5.1	613
1.14.10 Artifactory 3.4	614
1.14.10.1 Artifactory 3.4.2	615
1.14.10.2 Artifactory 3.4.1	616
1.14.11 Artifactory 3.3	616
1.14.11.1 Artifactory 3.3.1	617
1.14.11.2 Artifactory 3.3.0.1	618
1.14.12 Artifactory 3.2	619
1.14.12.1 Artifactory 3.2.1	619
1.14.12.2 Artifactory 3.2.2	620
1.14.13 Artifactory 3.1	621
1.14.13.1 Artifactory 3.1.1	622
1.14.14 Artifactory 3.0	623
1.14.14.1 Artifactory 3.0.1	624
1.14.14.2 Artifactory 3.0.2	624
1.14.14.3 Artifactory 3.0.3	625
1.14.14.4 Artifactory 3.0.4	625
1.14.15 Artifactory 2.x	626
1.14.15.1 1.3.0-RC2	626
1.14.15.2 1.3.0-RC1	629
1.14.15.3 1.3.0-beta-6	633
1.14.15.4 1.3.0-beta-5	637
1.14.15.5 1.3.0-beta-4	642
1.14.15.6 1.3.0-beta-3	647
1.14.15.7 Artifactory 2.1.1	651
1.14.15.8 Artifactory 2.2.4	651
1.14.15.9 Artifactory 2.2.5	652
1.14.15.10 Artifactory 2.3.3	652

1.14.15.11 Artifactory 2.3.4	653
1.14.15.12 Artifactory 2.3.4.1	653
1.14.15.13 Artifactory 2.1.2	654
1.14.15.14 Artifactory 2.1.3	654
1.14.15.15 Artifactory 2.2.0	655
1.14.15.16 Artifactory 2.2.1	655
1.14.15.17 Artifactory 2.2.2	655
1.14.15.18 Artifactory 2.2.3	656
1.14.15.19 Artifactory 2.3.0	656
1.14.15.20 Artifactory 2.3.1	657
1.14.15.21 Artifactory 2.3.2	657
1.14.15.22 Artifactory 2.4.0	658
1.14.15.23 Artifactory 2.4.1	658
1.14.15.24 Artifactory 2.4.2	659
1.14.15.25 Artifactory 2.5.0	659
1.14.15.26 Artifactory 2.5.1	659
1.14.15.27 Artifactory 2.5.1.1	660
1.14.15.28 Artifactory 2.5.2	660
1.14.15.29 Artifactory 2.6.0	660
1.14.15.30 Artifactory 2.6.1	661
1.14.15.31 Artifactory 2.6.2	661
1.14.15.32 Artifactory 2.6.3	662
1.14.15.33 Artifactory 2.6.4	662
1.14.15.34 Artifactory 2.6.5	662
1.14.15.35 Artifactory 2.6.6	663
1.14.15.36 Artifactory 2.6.7	663
1.14.16 Pivotal Cloud Foundry JFrog Artifactory Tile Release Notes	663

Welcome to Artifactory

Welcome to the JFrog Artifactory User Guide!

JFrog Artifactory is the only Universal Repository Manager supporting all major packaging formats, build tools and CI servers.

The screenshot shows the JFrog Artifactory web interface. On the left, there's a sidebar with links for Home, Artifactory, Builds, and Admin. The main area is titled "Artifact Repository Browser". It shows a tree view of artifacts under "Tree Simple" and a "Compress Empty Folders" option. A specific artifact, "artifactory-constants-2.6.4.pom", is selected. The right side displays detailed information about this artifact, including its name, repository path, module ID, deployed by, size, date, last modified, license, and download count. Below this is a "Package Information" section with a description and latest version. At the bottom, there's a "Dependency Declaration" section and a "Virtual Repository Associations" section. The overall interface is clean and modern, designed for managing software dependencies and artifacts.

This user guide is for Artifactory 4.0.x and higher.

Click this link to download the latest PDF version of the Artifactory User Guide:

Note that the online version may be more up-to-date.

If you are using Artifactory 3.x.y, please refer to the Artifactory 3 User Guide.

Which Artifactory Do You Need?

Artifactory comes in three flavors:

Artifactory oss	Offers powerful enterprise feature and fine-grained permission control behind a sleek and easy-to-use UI.	Download
------------------------	---	-----------------

Artifactory Pro	<p>Exposes a set of professional add-ons, on top of those already available to you from Artifactory Open Source, opening up a whole world of features that empower you to manage your binaries and integrate with industry standard tools in your development and deployment ecosystem.</p>	Download
Artifactory Cloud	<p>JFrog's SaaS-based solution for managing your artifacts and binary repositories in the cloud with the full power of Artifactory Pro behind you with 24/7 SLA-based support.</p>	Register

To see which version of Artifactory best suits your needs, please see the [Artifactory Features Matrix](#).

How is this Guide Organized?

Installing and Upgrading Artifactory	Learn how to install and upgrade Artifactory on all supported platforms, including detailed system requirements and pre-requisites.
Using Artifactory	Learn how to use Artifactory on a day-to-day basis including creating repositories, deploying, copying and moving artifacts and more.
Configuring Artifactory	Learn how to configure repositories, users, permissions and more.
Artifactory REST API	A detailed specification of Artifactory's extensive REST API letting you automate any process in your development ecosystem.
System Monitoring and Maintenance	Learn how to keep your system free of clutter and operating smoothly.

Artifactory High Availability	Learn how to configure and use Artifactory in a High Availability configuration providing the most stable and secure binary repository available to enterprise users today.
Bintray Integration	Learn how to integrate with JFrog Bintray to completely automate your software development pipeline all the way to distribution.
Artifactory Pro	Learn about all the add-ons that let Artifactory work seamlessly with packaging formats such as Docker, NuGet, Vagrant, RubyGems and more, as well as with all major CI servers and build tools
Release Notes	Learn about the changes that came with each release of Artifactory.

Page Contents

- [Welcome to the JFrog Artifactory User Guide!](#)
- [Which Artifactory Do You Need?](#)
- [How is this Guide Organized?](#)

Quick Links



Recently Updated

- [RubyGems Repositories](#)
6 minutes ago • updated by [Rami Honig](#) • [view change](#)
- [Updating Your Profile](#)
24 minutes ago • updated by [Rami Honig](#) • [view change](#)
- [Searching for Artifacts](#)
29 minutes ago • updated by [Rami Honig](#) • [view change](#)
- [Npm Repositories](#)
about an hour ago • updated by [Rami Honig](#) • [view change](#)
- [Bower Repositories](#)
about an hour ago • updated by [Rami Honig](#) • [view change](#)
- [Release Notes](#)
yesterday at 4:59 PM • updated by [Rami Honig](#) • [view change](#)
- [Artifactory 4.3](#)

- 📄 yesterday at 4:58 PM • updated by Rami Honig • [view change](#)
- 📄 [Artifactory 4.1.3](#)
yesterday at 3:17 PM • updated by Rami Honig • [view change](#)
- 📄 [Artifactory 4.1.2](#)
yesterday at 3:17 PM • updated by Rami Honig • [view change](#)
- 📄 [Artifactory 4.2.2](#)
yesterday at 3:17 PM • updated by Rami Honig • [view change](#)
- 📄 [Artifactory 4.2.1](#)
yesterday at 3:16 PM • updated by Rami Honig • [view change](#)
- 📄 [Artifactory 4.1](#)
yesterday at 3:16 PM • updated by Rami Honig • [view change](#)
- 📄 [Artifactory 4.2](#)
yesterday at 3:16 PM • updated by Rami Honig • [view change](#)
- 📄 [S3 Object Storage](#)
yesterday at 2:54 PM • updated by Rami Honig • [view change](#)
- 📄 [Getting Support](#)
yesterday at 2:18 PM • updated by Rami Honig • [view change](#)

Installing Artifactory

Overview

This section provides a guide on the different ways you can install and configure Artifactory.

System Requirements

Before you install Artifactory please refer to [System Requirements](#) for information on supported platforms, supported browsers and other requirements.

Installation

For detailed instructions, visit one of the following platform-specific pages:

- [Installing on Linux, Solaris or Mac OS](#)
- [Installing on Windows](#)
- [Running with Docker](#)

Changing the Default Database

A default installation of Artifactory works with a Derby Database, however for better performance and to reuse existing infrastructures you can configure Artifactory to work with alternative storage solutions.

For details please refer to [Changing the Default Storage](#).

Directory Structure

After installing Artifactory, the `$ARTIFACTORY_HOME` directory will contain the following directory structure

(the `$ARTIFACTORY_HOME` directory location depends on your installation type):

File/Folder	Description
<code>logs</code>	Artifactory log files (general, access, request etc.)
<code>etc</code>	Configuration files
<code>etc/plugins</code>	Custom Groovy user plugins.
<code>etc/security</code>	Global security related files (configuring global encryption key, PGP signing key etc.).
<code>etc/ui</code>	Manually uploaded custom UI logos.
<code>data/derby</code>	The Derby database (only present when using Derby).
<code>data/filestore</code>	The checksum based storage of binaries when using the default filesystem storage.
<code>data/tmp/work</code>	Directory to save temporary files which Artifactory generates.
<code>data/tmp/artifactory-uploads</code>	Directory to save files uploaded using the Web UI.
<code>bin</code>	Artifactory startup/shutdown scripts.

<i>tomcat</i>	The default tomcat directory bundled with Artifactory.
<i>tomcat/work</i>	The tmp directory tomcat and the JVM uses (Tomcat automatically assigns it to a java system environment variable as <code>java.io.tmpdir</code>)
<i>tomcat/logs</i>	Additional Tomcat log files
<i>misc</i>	Configuration files used as examples for different databases and servlet containers.
<i>backup</i>	The default backup directory Artifactory uses for system wide and repository backup.
<i>webapps</i>	Contains the Artifactory WAR file used by the bundled tomcat distribution.

Page Contents

- Overview
- System Requirements
- Installation
 - Changing the Default Database
 - Directory Structure
 - HTTP Server

Read more

- System Requirements
- Installing on Linux Solaris or Mac OS
- Installing on Windows
- Running with Docker
- Changing the Default Storage

- Running Behind an HTTP Server

HTTP Server

You can run Artifactory with one of the supported HTTP servers set up as a front end. For details please refer to [Running Behind an HTTP Server](#).

System Requirements

Supported Platforms

Artifactory has been tested and verified on Linux, Windows (Vista and higher), Solaris and Mac OS X. You should be able to run Artifactory on other platforms, but these have not been tested.

JDK

You must run Artifactory with **JDK 8** and above, preferably JDK 8 update 45 and above.

You can download the latest JDK from the [Oracle Java SE Download Site](#).

JAVA_HOME and JRE_HOME

Make sure your JAVA_HOME environment variable correctly points to your JDK 8 installation.

If you also have JRE_HOME defined in your system, this will take precedence over JAVA_HOME and therefore you need to either point JRE_HOME to your JDK 8 installation, or remove the JRE_HOME definition.

Page Contents

- [Supported Platforms](#)
- [JDK](#)
- [JVM Memory Allocation](#)
- [Browsers](#)
- [Recommended Hardware](#)
- [High Availability Configuration](#)
- [Database Requirements](#)
- [Servlet Containers](#)

JVM Memory Allocation

While not a strict requirement, we recommend that you modify the JVM memory parameters used to run Artifactory.

You should reserve at least 512MB for Artifactory, and the recommended values for JVM parameters are:

Recommended JVM parameters

```
-server -Xms2g -Xmx4g -Xss256k -XX:+UseG1GC
```

Repository size and number of concurrent users

The larger your repository or number of concurrent users, the larger you need to make the -Xms and -Xmx values accordingly

Browsers

Artifactory has been tested with the latest versions of Google Chrome, Firefox, Internet Explorer and Safari.

Recommended Hardware

The following table provides hardware recommendations for a single server machine:

Number of developers	OS/JVM	Processor	*Memory (RAM) for JVM Heap	Storage
1 - 20	64 bit	4 cores	4GB	Fast disk with free space that is at least 3 times the total size of stored artifacts
20 - 100	64 bit	4 cores	8GB	Fast disk with free space that is at least 3 times the total size of stored artifacts
100 - 200	64 bit	8 cores (16 cores recommended)	12GB	Fast disk with free space that is at least 3 times the total size of stored artifacts (backup SAN recommended)
200+	64 bit	Contact JFrog support for a recommended setup.		

*Memory (RAM) for JVM Heap

This specifies the amount of memory that Artifactory requires from the JVM heap. The server machine should have enough additional memory to run the operating system and any other processes running on the machine.

Build machine

For the purposes of this table, a build machine is considered equivalent to 10 developers

High Availability Configuration

If you are running Artifactory in a High Availability configuration, to maintain high system performance in case of single or multiple server crash, we recommend following the [recommended hardware](#) guidelines above for each of the HA server instances. For more details, please refer to [Artifactory High Availability](#).

Database Requirements

To avoid network latency issues when reading and writing artifacts data, we strongly recommend creating the database either on the same machine on which Artifactory is running (database engine and storage) or on a fast network storage. This recommendation is critical when using `fileDb` (whereby files are served from database BLOBs) and the file system cache is small.

For supported databases and more details, please refer to [Changing the Default Storage](#).

Servlet Containers

Artifactory should be run with its bundled Tomcat 8 servlet container.

Installing on Linux Solaris or Mac OS

Overview

This page describes how to install Artifactory on Linux, Solaris or Mac OS.

The procedure for all these platforms is identical, so for the sake of clarity the rest of this page will refer to Linux only.

You can install Artifactory on your Linux system in one of the following ways:

- [Manual Installation](#)
- [Service Installation](#)
- [RPM or Debian Installation](#)
- [As a Docker Image](#)

Running as root to install Artifactory as a service or RPM distribution

To install Artifactory as a service or RPM distribution you must have root privileges.

To run as root either execute the following command:

```
su -
```

or precede all commands with `sudo` (e.g. `sudo service artifactory start`)

If you are unable to get root privileges please contact your system administrator.

Page Contents

- [Overview](#)
- [Requirements](#)
 - Setting `JAVA_HOME`
 - Setting Java Memory Parameters
- [Manual Installation](#)
 - Installing Artifactory
 - Running Artifactory
- [Service Installation](#)
 - Installing Artifactory
 - Running Artifactory
- [RPM or Debian Installation](#)
 - Managed Files and Folders
 - Installing Artifactory
 - Running Artifactory
 - Additional Configuration for MySQL
 - Backup and Recover
- [Running with Docker](#)
- [The `ARTIFACTORY_HOME` Folder](#)
- [Accessing Artifactory](#)

Requirements

Setting `JAVA_HOME`

As mentioned in the section on [System Requirements](#), make sure that your `JAVA_HOME` environment variable is correctly set to your JDK installation.

Setting Java Memory Parameters

While not a strict requirement, it is recommended to modify the JVM memory parameters used to run Artifactory.

If you can reserve at least 512MB for Artifactory, the recommended values for JVM parameters are:

Recommended JVM parameters

```
-server -Xms512m -Xmx2g -Xss256k -XX:+UseG1GC
```

Repository size and number of concurrent users

The larger your repository or number of concurrent users, the larger you need to make the `-Xms` and `-Xmx` values accordingly

Manual Installation

Installing Artifactory

To install Artifactory manually, simply unzip the Artifactory download file to a location on your file system. This will be your `$ARTIFACTORY_HOME` location.

No further action is needed.

To modify your **JVM parameters** modify the `JAVA_OPTIONS` in `$ARTIFACTORY_HOME/bin/artifactory.default`

Running Artifactory

You can run Artifactory manually to see its behavior by directly executing:

```
$ARTIFACTORY_HOME/bin/artifactory.sh
```

The console is locked on the Artifactory process and you can stop it cleanly with `Ctrl+C`.

To directly run Artifactory as a daemon process, using the environment variables of the shell you are currently in, execute the following script:

```
$ARTIFACTORY_HOME/bin/artifactoryctl start
```

Startup time

Depending on your system performance it may take Artifactory several seconds to start up. If you try to access Artifactory through your browser while it is starting up, within a few seconds it will provide a notification that it is in the startup process.

Using the same script, you can check if Artifactory is running and display its process id, or stop it using:

Checking if Artifactory is running or stopping it

```
$ARTIFACTORY_HOME/bin/artifactoryctl check | stop
```

To run the Artifactory UI see [Accessing Artifactory](#).

Service Installation

Artifactory is packaged as a zip file with a bundled Tomcat, and a complete install script that can be used to install it as a Linux service, running under a custom user.

Permissions

When running Artifactory as a service, the installation script creates a user called `Artifactory` which must have run and execute permissions on the installation directory.

Therefore it is recommended to extract the Artifactory download file into a directory that gives run and execute permissions to all users such as `/opt`

Installing Artifactory

To install Artifactory as a service, browse to your `$ARTIFACTORY_HOME/bin` directory and execute the following command as root:

Running the installation script as root

```
$ARTIFACTORY_HOME/bin/installService.sh [USER [GROUP]]
```

The following table describes the sequence of commands performed by the install script:

<code>User creation</code>	Creates a default user named <code>artifactory</code> (<code>\$ARTIFACTORY_USER</code>). You can change the default user by editing the <code>\$ARTIFACTORY_USER</code> value in <code>/etc/opt/jfrog/artifactory/default</code> . You can also optionally run the install script to start the Artifactory service under a different user by passing in the user name as the first parameter. If you include the user name, you may also optionally include a group.
<code>etc config</code>	Creates the folder <code>/etc/opt/jfrog/artifactory</code> , copies the configuration files there and creates a soft link in <code>\$ARTIFACTORY_HOME/etc</code>
<code>etc default</code>	Creates the file <code>/etc/opt/jfrog/artifactory/default</code> containing the main environment variables needed for Artifactory to run: <code>JAVA_HOME</code> , <code>ARTIFACTORY_USER</code> , <code>ARTIFACTORY_HOME</code> , <code>JAVA_OPTIONS</code> , ... The <code>/etc/opt/jfrog/artifactory/default</code> is included at the top of <code>artifactoryctl</code> and can include any settings. To modify your JVM parameters modify <code>JAVA_OPTIONS</code> in <code>/etc/opt/jfrog/artifactory/default</code>
<code>init.d</code>	Copies the service script file <code>artifactory</code> to <code>/etc/init.d/artifactory</code>

Logs folder	Creates the folder <code>\$ARTIFACTORY_HOME/logs</code> , makes it writable for the user <code>ARTIFACTORY_USER</code> and creates a soft link <code>\$ARTIFACTORY_HOME/logs</code> .
	The <code>\$ARTIFACTORY_HOME/tomcat/logs</code> folder is linked to <code>\$ARTIFACTORY_HOME/logs/catalina</code> .
Backup folder	Creates the folder <code>\$ARTIFACTORY_HOME/backup</code> , so you must create a link if you want this folder to point to a different place (such as <code>/var/backup/artifactory</code> for example). The script makes <code>\$ARTIFACTORY_HOME/backup</code> writable for the user <code>ARTIFACTORY_USER</code> .
Data folder	Creates the folder <code>\$ARTIFACTORY_HOME/data</code> , so you must create a link if you want this folder to point to somewhere else. The script makes it writable for the user <code>ARTIFACTORY_USER</code> .
chkconfig calls	The script calls <code>add</code> and <code>list</code> (you can see the output), and then activates the Artifactory service

Running Artifactory

To start or stop Artifactory as a service you must be running as root and can use the following command:

```
service artifactory start | stop
```

Checking the status of the Artifactory service

Once Artifactory is correctly installed, you can check if it is running with:

```
service artifactory check
```

If Artifactory is running, you should see its pid.

If Artifactory is not running you will see a list of environment variables used by the service.

You can also check the Artifactory log with:

```
tail -f $ARTIFACTORY_HOME/logs/artifactory.log
```

When installing from an RPM or Debian distribution, Artifactory is generally started as `root` and will `su` internally to the `ARTIFACTORY_USER` user.

Security

For reasons of security, it is not recommended to leave the `ARTIFACTORY_USER` variable undefined with Artifactory running as the current user, especially if the current user is `root`.

To run the Artifactory UI see [Accessing Artifactory](#).

RPM or Debian Installation

Artifactory can also be installed from an RPM or Debian distribution on Red Hat compatible Linux distributions.

The installation package creates a dedicated user, installs a stripped-down distribution of the Apache Tomcat container configured for Artifactory (on port 8081), and registers this Tomcat as a service (but does not start it immediately).

This package effectively replaces the different setup scripts included with the Artifactory Zip distribution.

Managed Files and Folders

When installed from an RPM distribution, Artifactory retains the FHS (Filesystem Hierarchy Standard) format:

<i>File/Folder</i>	<i>Location</i>	<i>Ownership</i>
Artifactory home	<i>/var/opt/jfrog/artifactory</i>	artifactory
Artifactory etc	<i>/etc/opt/jfrog/artifactory</i>	artifactory
Artifactory logs	<i>/var/opt/jfrog/artifactory/logs</i>	artifactory
Artifactory env variables	<i>/etc/opt/jfrog/artifactory/default</i>	artifactory
Tomcat home	<i>/opt/jfrog/artifactory/tomcat</i>	artifactory (root for sub dirs)
Artifactory startup script	<i>/etc/init.d/artifactory</i>	root
Artifactory binary	<i>/opt/jfrog/artifactory</i>	root

Installing Artifactory

To install Artifactory from an RPM or Debian distribution **you must be running as root** and can use the corresponding command below:

▼ [Installing from an RPM distribution...](#)

```
rpm -ivh jfrog-artifactory-<oss | pro>-<version>.rpm
```

▼ [Installing from a Debian distribution...](#)

```
dpkg -i jfrog-artifactory-<oss | pro>-<version>.deb
```

Make sure to modify your JVM parameters by modifying JAVA_OPTIONS in */etc/opt/jfrog/artifactory/default* as appropriate for your installation.

Running Artifactory

To start or stop Artifactory **you must be running as root** and can use the following command:

```
service artifactory start | stop
```

Checking the status of the Artifactory service

Once Artifactory is correctly installed, you can check if it is running with:

```
service artifactory check
```

If Artifactory is running, you should see its pid.

If Artifactory is not running you will see a list of environment variables used by the service.

You can also check the Artifactory log with:

```
tail -f $ARTIFACTORY_HOME/logs/artifactory.log
```

When installing from an RPM distribution, Artifactory is generally started as `root` and will `su` internally to the `$ARTIFACTORY_USER` user.

Security

For reasons of security, it is not recommended to leave the `$ARTIFACTORY_USER` variable undefined with Artifactory running as the current user, especially if the current user is `root`.

Additional Configuration for MySQL

Since MySQL is a commonly used storage solution for RPM installations, Artifactory provides a small command line tool to assist you in [Changing the Default Storage](#) from Derby to MySQL if preferred.

The CLI tool is located in `/opt/jfrog/artifactory/bin/configure_mysql.sh` and provides you with the option to run the MySQL configuration manually after running the RPM installation process.

Prerequisite

MySQL version 5.5 or higher must be pre-installed and running for the command line tool to work

To run the Artifactory UI see [Accessing Artifactory](#).

Backup and Recover

When uninstalling an RPM distribution of Artifactory, it will save the `$ARTIFACTORY_HOME` folder and create a backup folder at `/var/opt/jfrog/` while preserving symbolic links to remote filestores.

After installing a new instance of Artifactory, you can recover the configuration and filestore from this backup by running the script `$ARTIFACTORY_BINARY/bin/recover.backup.sh`.

Working with an external database

This process does not back up an external database, but rather its definitions in Artifactory. Therefore, when working with an external database, a manual dump should be performed before uninstalling the RPM, and then imported when starting the new installation.

Installing/Upgrading on a new machine

The Backup and Recover described above will only work if you are re-installing the RPM on the same machine. If you are installing or upgrading the RPM on a new machine you will need to use Import as described in the section on [Upgrading Artifactory](#).

Running with Docker

From Version 3.6, Artifactory may be pulled as a Docker Image. For full details, please refer to [Running with Docker](#).

The `ARTIFACTORY_HOME` Folder

It is important to know where your Artifactory home folder is located, since this folder stores your configurations and important repository data.

When Artifactory runs for the first time, it sets up a default configuration and creates all necessary files and folders under the `ARTIFACTORY_HOME` folder.

The default location of `ARTIFACTORY_HOME` is `{user.home}/.artifactory`.

To run Artifactory with the home folder set to a different location on the file system (particularly when installing Artifactory on a production server), either:

- Start the Tomcat virtual machine with `-Dartifactory.home=<your preferred Artifactory home folder location>`
- or -
- Set an `ARTIFACTORY_HOME` environment variable pointing to your preferred location before running the installation.

Artifactory creates the home folder on startup if it does not already exist.

Permissions on the Artifactory Home Folder

Make sure that the user running the Tomcat has write permissions on the Artifactory home folder.

Accessing Artifactory

Artifactory can be accessed using the following URL:

`http://SERVER_DOMAIN:8081/artifactory`.

For example, if you are testing on your local machine you would use: `http://localhost:8081/artifactory`

Installing on Windows

Overview

There are three ways to install Artifactory on your Windows system:

- Manual Installation
- Service Installation
- As a Docker Image

Unzip the Artifactory download file to a location on your file system.

This will be your `%ARTIFACTORY_HOME%` location.

Define this location as an environment variable called `ARTIFACTORY_HOME`.

Page Contents

- Overview
- Requirements
 - Setting `JAVA_HOME`
 - Setting Java Memory Parameters
- Manual Installation
- Service Installation
 - Running Artifactory
- Running with Docker
- Accessing Artifactory

Requirements

Setting JAVA_HOME

As mentioned in the section on [System Requirements](#), make sure that your `JAVA_HOME` environment variable is correctly set to your JDK installation.

Setting Java Memory Parameters

While not a strict requirement, it is recommended to modify the JVM memory parameters used to run Artifactory.

This is done by modifying the `JAVA_OPTIONS` variable in `artifactory.bat`, for a [manual installation](#), or the `JOPTS` variable in `installService.bat` when running Artifactory as a [service](#).

For your changes to take effect you need to stop Artifactory and then rerun the modified file.

If you can reserve at least 512MB for Artifactory, the recommended values for JVM parameters are:

Recommended JVM parameters

```
-server -Xms512m -Xmx2g -Xss256k -XX:PermSize=128m -XX:MaxPermSize=256m -XX:+UseG1GC
```

In JDK 8 the `PermSize` and `MaxPermSize` parameters have been deprecated. If you continue to use these parameters you may see warnings in the Artifactory startup sequence, but these may be safely ignored.

Repository size and number of concurrent users

The larger your repository or number of concurrent users, the larger you need to make the `-Xms` and `-Xmx` values accordingly

Manual Installation

Browse to `%ARTIFACTORY_HOME%\bin` and execute the file `artifactory.bat`. This script searches for the Java executable and runs Artifactory's main class.

Security settings

Depending on the security settings under Windows, you might need to run `artifactory.bat` using 'Run as administrator'

To test your installation see [Accessing Artifactory](#).

Service Installation

Artifactory makes use of the [Apache Commons Procrun](#) components allowing you to install the application as a Windows Service.

To run Artifactory as a Windows service, browse to `%ARTIFACTORY_HOME%\bin`, and execute the file `InstallService.bat`.

By editing `InstallService.bat`, you can modify default properties such as `JOPTS` and the log directory.

For your changes to take effect you need to stop the currently running Artifactory service and run `InstallService.bat` again once you have completed your modifications.

To test your installation see [Accessing Artifactory](#).

Security

Windows 8 implements strict User Account Control (UAC). You must either disable UAC or right-click on `cmd.exe` and select "Run as administrator" in order to run this script.

Running on 32 bit Windows

If you are running a 32 bit version of Windows you need to do the following:

- Download the latest version of the [Apache Commons Daemon](#).
- Take `prunsrv.exe` from the downloaded archive and rename it to `artifactory-service.exe`
- Replace the current `artifactory-service.exe` found in your `%ARTIFACTORY_HOME%/bin` directory

Running Artifactory

After installing Artifactory you need to start the service.

To start or stop Artifactory as a service you can use the following command in a **Command Prompt** window:

Starting and stopping the Artifactory service

```
sc start|stop Artifactory
```

Checking the status of the Artifactory service

Once Artifactory is correctly installed, you can check if it is running with:

```
sc query Artifactory
```

You can also use any standard text editor to view the artifactory log data found in `$ARTIFACTORY_HOME/logs/artifactory.log`

Running with Docker

From Version 3.6, Artifactory may be pulled as a Docker Image. To run Artifactory in a Docker container on a Windows system, you first need to install [boot2docker](#).

For full details, please refer to [Running with Docker](#).

Accessing Artifactory

Artifactory can be accessed using the following URL:

http://SERVER_DOMAIN:8081/artifactory.

For example, if you are testing on your local machine you would use: <http://localhost:8081/artifactory>

Running with Docker

Artifactory as a Docker Image

Artifactory can be [pulled from Bintray](#) as a Docker image and run as a container. To do this, you clearly need to have Docker properly installed and configured on your system. For details about installing and using Docker, please refer to the [Docker documentation](#).

If you are running on a Windows or Mac system, you also first need to install [boot2docker](#).

The Artifactory Docker Image comes in three varieties:

1. Artifactory OSS

2. Artifactory Pro
3. Artifactory Pro Registry

Artifactory Docker Image Contents

The **Artifactory OSS** and **Artifactory Pro** Docker images contain the following components:

- CentOS v6.6
- OpenJDK
- An RPM installation for Artifactory.

The **Artifactory Pro Registry** also contains the above components, and in addition:

- Nginx with a self-signed certificate

As a result, the Artifactory Pro Registry version is a fully-fledged Docker registry supporting both the Docker V1 and Docker V2 APIs.

Using Artifactory as a secure private Docker registry

You may also create Docker repositories in Artifactory to make it your secure, private Docker registry. For more details, please refer to [Docker Repositories](#).

Page contents

- Artifactory as a Docker Image
 - Artifactory Docker Image Contents
- Pulling the Artifactory Docker Image
- Running Artifactory
 - Running Artifactory OSS or Pro
 - Accessing Artifactory OSS or Pro
 - Running Artifactory Pro Registry
 - Accessing Artifactory Pro Registry
- Upgrading Artifactory
 - RPM Upgrade
 - Using a Mounted Volume
- Screencast

Pulling the Artifactory Docker Image

Artifactory may be pulled from Bintray as a Docker image by executing the corresponding Docker command:

```
For Artifactory Pro
docker pull jfrog-docker-reg2.bintray.io/jfrog/artifactory-pro:<version>

For Artifactory OSS
docker pull jfrog-docker-reg2.bintray.io/jfrog/artifactory-oss:<version>

For Artifactory Pro Registry
docker pull jfrog-docker-reg2.bintray.io/jfrog/artifactory-registry:<version>
```

<version>: specifies the version. Use `latest` to indicate that the latest available version should be pulled.

Pulling Artifactory as a Docker image, running Artifactory as a container and using Artifactory as a private Docker registry is demonstrated in the [screencast](#) below.

Running Artifactory

Artifactory is included as an RPM service and is automatically started when running the Docker container.

NOTE: The volumes (-v) switches listed below are not working in artifactory v4.0.0 through v4.0.2. You will need to either run without volumes or

run the image once, stop it, then commit the container and run that image rather than the current one.

Running Artifactory OSS or Pro

To run Artifactory OSS or Artifactory Pro, execute the following Docker command:

```
docker run --name <container name> -p 8081:8081 <volume mounts>
jfrog-docker-reg2.bintray.io/jfrog/artifactory-<oss|pro|registry>:<version>
```

<container name>: A logical name for the running container

<version>: The version you want to run. Use `latest` to indicate that the latest available version should be run.

<volume mounts>: These are optional, but we highly recommended the following export and volume mounts for easy access:

export ARTIFACTORY_HOME=/var/opt/jfrog/artifactory

- -v \$ARTIFACTORY_HOME/data
- -v \$ARTIFACTORY_HOME/logs
- -v \$ARTIFACTORY_HOME/backup
- -v \$ARTIFACTORY_HOME/etc

Accessing Artifactory OSS or Pro

You can access Artifactory at:

- <http://localhost:8081/artifactory>.

Running with boot2docker

If you are running with `boot2docker` you need to use `DOCKER_HOST` IP instead of `localhost`. A custom host may be configured in the Nginx configuration running in the container under `/etc/nginx/conf.d/default.conf`

Running Artifactory Pro Registry

To run Artifactory Pro Registry, execute the following Docker command:

```
docker run -d --name <container name> -p 80:80 -p 8081:8081 -p 443:443 -p
5000-5002:5000-5002 <volume mounts>
jfrog-docker-reg2.bintray.io/jfrog/artifactory-registry:<version>
```

<container name>: A logical name for the running container

<version>: The version you want to run. Use `latest` to indicate that the latest available version should be run.

<volume mounts>: These are optional but we highly recommended the following export and volume mounts for easy access:

export ARTIFACTORY_HOME=/var/opt/jfrog/artifactory

- -v \$ARTIFACTORY_HOME/data
- -v \$ARTIFACTORY_HOME/logs
- -v \$ARTIFACTORY_HOME/backup
- -v \$ARTIFACTORY_HOME/etc

Additional Information Port Bindings

The port bindings used correspond to those used in the pre-configured nginx included with the Docker image and allow you to run

Artifactory with the https mappings.

Memory Parameters

Note that you can pass memory parameters to the container using `-e RUNTIME_OPTS="-Xms512m -Xmx4g"`. This is optional.

Working in the Docker container

You can work in the Docker container using:

```
docker exec -it <container name> /bin/bash
```

Accessing Artifactory Pro Registry

When running Artifactory Pro with the port bindings described above, you can access Artifactory at the following URLs:

- `http://localhost/artifactory`
- `http://localhost:8081/artifactory`
- `https://localhost:5000/v2` (This is mapped to `http://localhost:8081/artifactory/api/docker/docker-remote/v2`)
- `https://localhost:5001/v1` (This is mapped to `http://localhost:8081/artifactory/api/docker/docker-prod-local/v1`)
- `https://localhost:5002/v1` (This is mapped to `http://localhost:8081/artifactory/api/docker/docker-dev-local/v1`)
- `https://localhost:5001/v2` (This is mapped to `http://localhost:8081/artifactory/api/docker/docker-prod-local/v2`)
- `https://localhost:5002/v2` (This is mapped to `http://localhost:8081/artifactory/api/docker/docker-dev-local/v2`)

Running with boot2docker

If you are running with `boot2docker` you need to use `DOCKER_HOST` IP instead of `localhost`. A custom host may be configured in the Nginx configuration running in the container under `/etc/nginx/conf.d/default.conf`

Upgrading Artifactory

There are two ways to upgrade Artifactory when running with Docker:

1. An RPM upgrade
2. Using a Mounted Volume

RPM Upgrade

Artifactory runs from an RPM package within a Docker container. You can upgrade Artifactory by logging in to your Docker container and following the instructions in [Running as an RPM Installation](#).

Using a Mounted Volume

You can upgrade Artifactory by switching versions while riding on top of your container volumes as follows:

- Stop your current installation of Artifactory using the following command:
`docker stop <container name>`
- Run the new version of Artifactory using the following command:
`docker run -d --name <new container name> --volumes-from=<container name> -p 8081:8081 jfrog-docker-reg2.bintray.io/jfrog/artifactory-<oss|pro|registry>:<new version>`
- Once you have established that the new version is running correctly, you can remove the old version as follows:
`docker rm <container name>`

Here is an example showing these three steps when upgrading Artifactory OSS from version 4.0.0 (whose container is called `artifactory-4.0.0`) to version 4.1.0 (whose container is called `artifactory-4.1.0`):

```
docker stop artifactory-4.0.0
docker run -d --name artifactory-4.1.0 --volumes-from=artifactory-4.0.0 -p
8081:8081 jfrog-docker-reg2.bintray.io/jfrog/artifactory-oss:4.1.0
docker rm artifactory-4.0.0
```

Screencast

View the screencast below to see how you can download Artifactory as a Docker image and run it in a container within one minute.

Coming Soon...

Changing the Default Storage

Overview

Artifactory comes with a built-in Derby database that can be reliably used to store data for production-level repositories up to hundreds of gigabytes in size.

However, Artifactory's storage layer supports pluggable storage implementations allowing you to change the default storage and use other popular databases.

Artifactory currently supports the following databases:

- Derby (The default embedded database)
- MySQL v5.5 and above with InnoDB
- Oracle version 10g and above
- Microsoft SQL Server 2008 and above
- PostgreSQL v9.2 and above

Modes of Operation

Artifactory supports two modes of operation:

- Metadata in the database and binaries stored on the file system (This is the default and recommended configuration).
- Metadata and binaries stored as BLOBs in the database

"Once-And-Only-Once" Content Storage

Artifactory stores binary files only once.

When a new file about to be stored in Artifactory is found to have the same checksum as a previously stored file (i.e. it is identical to a file that has already been stored), Artifactory does not store the new file content, but rather creates a link to the existing file in the metadata of the newly deployed file.

This principle applies regardless of which repository and path artifacts are deployed - you can deploy the same file to many different coordinates, and as long as identical content is found in the storage, it is reused.

Page Contents

- Overview
 - Modes of Operation
 - "Once-And-Only-Once" Content Storage
- Before You Start
 - Backup Your Current Installation
 - Remove the Old Data Folder

- Setup the New Storage
 - The Bundled Storage Configurations

Read more

- MySQL
- Oracle
- Microsoft SQL Server
- PostgreSQL

Before You Start

Preprocessing

Changing the storage type does not automatically transfer your data to the new storage. Please follow the steps below to backup your data so that you can restore it after the change.

Backup Your Current Installation

When changing the storage type for an existing installation you must import your Artifactory content and configuration from a backup.

Make sure to backup your current Artifactory system before updating to a new storage type.

Remove the Old Data Folder

If you have previously run Artifactory with a different storage type you must remove (or move) the existing `$ARTIFACTORY_HOME/data` folder.

If you do not do this, Artifactory continues to use some of the previous storage definitions and will fail to start up producing a `NotFoundException` in several places during startup sequence. Removing (or emptying) the `$ARTIFACTORY_HOME/data` folder will avoid these errors.

Setup the New Storage

To setup your new storage you need to create a database instance, create an Artifactory user for the database, install the appropriate JDBC driver, copy the relevant database configuration file, and configure the corresponding `storage.properties` file.

This is fully detailed in the specific documentation page for each of the supported databases listed in the [Overview](#) section.

The Bundled Storage Configurations

For each of the supported databases you can find the corresponding properties file inside `$ARTIFACTORY_HOME/misc/db`.

Each file contains the mandatory parameters and definitions that should be configured to work with your database as follows:

<code>binary.provider.type</code>	<p>filesystem (default)</p> <p>This means that metadata is stored in the database, but binaries are stored in the file system. The default location is under <code>\$ARTIFACTORY_HOME/data/filestore</code> however this can be modified.</p> <p>fullDb</p> <p>All the metadata and the binaries are stored as BLOBS in the database.</p> <p>cachedFS</p> <p>Works the same way as <i>filesystem</i> but also has a binary LRU (Least Recently Used) cache for upload/download requests. Improves performance of instances with high IOPS (I/O Operations) or slow NFS access.</p> <p>S3</p> <p>This is the setting used for S3 Object Storage.</p>
<code>pool.max.active</code>	The maximum number of pooled database connections (default: 100).
<code>pool.max.idle</code>	The maximum number of pooled idle database connections (default: 10).
<code>binary.provider.cache maxSize</code>	If <code>binary.provider.type</code> is set to <code>fullDb</code> this value specifies the maximum cache size (in bytes) to allocate on the system for caching BLOBS.
<code>binary.provider.filesystem.dir</code>	If <code>binary.provider.type</code> is set to <code>filesystem</code> this value specifies the location of the binaries (default: <code>\$ARTIFACTORY_HOME/data/filestore</code>).
<code>binary.provider.cache.dir</code>	The location of the cache. This should be set to your <code>\$ARTIFACTORY_HOME</code> directory directly (not on the NFS).

Backing up `$ARTIFACTORY_HOME/data/filestore`

If `binary.provider.type` is set to `filesystem`, then raw Artifactory data must be backed up.

To do this, the `$ARTIFACTORY_HOME/data/filestore` folder should be backed-up in parallel with a database dump since both are required. The database must be dumped first.

This does not impact Artifactory's own backup system which is storage-agnostic.

Accessing a Remote Database

To avoid network latency issues when reading and writing artifacts data, we highly recommend that you create the database either on the same machine on which Artifactory is running or on a fast SAN disk.

This is critical when `binary.provider.type` is set to `fullDb` (whereby files are served from database BLOBS) and the file system cache is small.

MySQL

Overview

By using MySQL you can benefit from features in the MySQL infrastructure such as backup, restore and high availability.

For Artifactory to run with MySQL you must create a dedicated MySQL database instance and then configure Artifactory to use it as described in the following sections.

Before You Continue

Before proceeding with the steps below, please make sure you have read and followed the steps described in [Changing the Default Storage](#).

Page Contents

- [Overview](#)
- [Creating the Artifactory MySQL Database](#)
- [Increasing MySQL Default Packet Size](#)
- [Tuning MySQL for Artifactory](#)
- [Configuring Artifactory to use MySQL](#)

Creating the Artifactory MySQL Database

Supported MySQL Versions

Artifactory supports MySQL v5.5 and above with InnoDB engine which is the default provided.

Artifactory provides a script that will execute the SQL commands you need to create your MySQL database.

The script can be found in `$ARTIFACTORY_HOME/misc/mysql/createdb.sql` and is displayed below.

You should review the script and modify it as needed to correspond to your specific environment.

createdb.sql Script

```
CREATE DATABASE artdb CHARACTER SET utf8 COLLATE utf8_bin;
GRANT ALL ON artdb.* TO 'artifactory'@'localhost' IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
```

Selecting a Case-Sensitive Collation

While MySQL Database Server is not case-sensitive by default, it is important to select a case-sensitive collation because Artifactory is case-sensitive.

For example, in the `createdb.sql` script above COLLATE is set to "utf8_bin".

Artifactory privileges

We recommend providing Artifactory with full privileges on the database

Increasing MySQL Default Packet Size

Since some data files (builds, configurations etc.) are stored in MySQL, it is extremely important to increase the maximum allowed packet size used by MySQL to avoid errors related to large packets.

(Please refer to MySQL documentation: [Packet Too Large](#)).

It is recommended to change this in the `/etc/my.cnf` file as follows:

Modifying `/etc/my.cnf`

```
# The MySQL server
[mysqld]
.
.
# The maximum size of the binary payload the server can handle
max_allowed_packet=8M
.
```

`/etc/my.cnf` Absolute Path

If `/etc/my.cnf` does not already exist it should be created under the absolute path and not under `$ARTIFACTORY_HOME`

Restart required

After modifying the maximum allowed packed size you need to restart MySQL

Tuning MySQL for Artifactory

When using Artifactory with MySQL it is recommended to use the InnoDB engine with the following tuning parameters configured in the `/etc/my.cnf` file:

Tuning Parameters for MySQL

```
# The MySQL server
[mysqld]
.

# By default innodb engine use one file for all databases and tables. We recommend
changing this to one file per table.
# NOTE: This will take effect only if Artifactory tables are not created yet! Need to
be set and MySQL restarted before starting Artifactory for the first time.
innodb_file_per_table

# Theses are tuning parameters that can be set to control and increase the memory
buffer sizes.
innodb_buffer_pool_size=1536M
tmp_table_size=512M
max_heap_table_size=512M

# Theses control the innodb log files size and can be changed only when MySQL is down
and MySQL will not start if there are some innodb log files left in the datadir.
# So, changing theses means removing the old innodb log files before start.
innodb_log_file_size=256M
innodb_log_buffer_size=4M
.
```

Restart required

After tuning, you need to restart MySQL

Configuring Artifactory to use MySQL

1. Copy `$ARTIFACTORY_HOME/misc/db/mysql.properties` to `$ARTIFACTORY_HOME/etc/storage.properties` (If you do not have this file you can take it from the standalone zip distribution or directly from the JFrog domain). For a full explanation on the contents of this file please refer to [The Bundled Storage Configurations](#).
2. Adjust the connection definitions in the `$ARTIFACTORY_HOME/etc/storage.properties` file to match the attributes of the Artifactory database you created.
You must configure the database URL and username/password to use. The schema and tables are created first time Artifactory is run using the new database.
3. Download the MySQL JDBC driver (available from the [MySQL website](#)) and copy the `mysql-connector-java-<version>.jar` file into the server's shared lib directory.
For example `$TOMCAT_HOME/lib` when installed as a service or `$ARTIFACTORY_HOME/tomcat/lib` in the standalone version.

Permissions

Make sure your driver has the same permissions as the rest of the files in the shared lib directory.

4. Start Artifactory.

Storing BLOBs inside MySQL

The suggested (and recommended) configuration stores all artifact information in MySQL while the artifact binary data is stored on the file system (under `$ARTIFACTORY_HOME/data/filestore`).

While it is **not recommended**, it is possible to store BLOBs inside MySQL provided that the typical BLOB size is relatively small.

Storing large BLOBs in MySQL can cause memory issues because MySQL buffers BLOBs rather than streaming them (please refer to [MySQL Bug #15089](#)) and may result in `OutOfMemory` errors with large binaries depending on your JVM heap size.

To store BLOBs in MySQL, in the `storage.properties` file set `binary.provider.type=fullDb` and change `max_allowed_`

packet to be higher than the maximum artifact size you intend to store in Artifactory.

Oracle

Overview

By using Oracle you can benefit from features in Oracle infrastructure such as backup, restore and high availability.

For Artifactory to run with Oracle you must create a dedicated Oracle database instance and then configure Artifactory to use it as described in the following sections.

Before You Continue

Before proceeding with the steps below, please make sure you have read and followed the steps described in [Changing the Default Storage](#).

Page Contents

- [Overview](#)
- [Creating the Artifactory Oracle Database](#)
- [Configuring Artifactory to use Oracle](#)

Creating the Artifactory Oracle Database

Supported Oracle Versions

Artifactory supports Oracle v10g and above.

You can choose between two configurations to set up your Oracle Database

1. **DB-Filesystem**
This configuration stores metadata in Oracle Database and artifact binary data is stored on the file system (under `$ARTIFACTORY_HOME/data/filestore`). This option has the advantage of being very lightweight on the Oracle database.
2. **Full DB**
This configuration stores both metadata and BLOBs in Oracle Database. This option requires minimal maintenance and allows you to rely solely on Oracle for failover and backup procedures, since all data is in the database.
When using this option, make sure you have created a table space big enough to accommodate your binaries.

Artifactory privileges

Artifactory creates all tables automatically first time it is run. When performing a software upgrade Artifactory may have to alter tables and indices, so make sure you grant the configured connection the appropriate user permissions to perform such actions.

Recommendation

With both of the above options (Full DB and DB-Filesystem), it is recommended to create a dedicated table space and use AL32UTF8 encoding.

Reclaiming BLOB space

For efficiency, Artifactory uses a checksum to ensure that only one copy of any binary data is stored, however, you may want to reclaim deleted BLOB space from time to time by shrinking the BLOB table space as follows:

Reclaiming Deleted BLOB Space

```
{schema}.binary_blobs modify lob (data) (shrink space cascade);
```

Configuring Artifactory to use Oracle

1. Copy `$ARTIFACTORY_HOME/misc/db/oracle.properties` to `$ARTIFACTORY_HOME/etc/storage.properties` (If you do not have this file you can take it from the standalone zip distribution or directly from the JFrog domain). For a full explanation on the contents of this file please refer to [The Bundled Storage Configurations](#).
2. Adjust the connection definitions in the `$ARTIFACTORY_HOME/etc/storage.properties` file to match the attributes of the Artifactory database you created.
You must configure the database URL and username/password to use. The schema and tables are created first time Artifactory is run using the new database.
3. Download the JDBC driver corresponding to your Oracle version from the [JDBC/UCP Download Page](#) and copy the `ojdbc6.jar` file into the server's shared lib directory.
For example `$TOMCAT_HOME/lib` when installed as a service or `$ARTIFACTORY_HOME/tomcat/lib` in the standalone version.

Permissions

Make sure your driver has the same permissions as the rest of the files in the shared lib directory.

4. Start Artifactory.

Microsoft SQL Server

Overview

By using MicrosoftSQLyou can benefit from features in the Microsoft SQL Server infrastructure such as backup,restoreand high availability.

Optimizing Artifactory when running with MS SQL Server

When running Artifactory with Microsoft SQL Server you may create the Artifactory schema on an existing server used for other applications, however for optimal performance, we recommend creating a dedicated Microsoft SQL Server database instance and then configure Artifactory to use it as described in the following sections.

Before You Continue

Before proceeding with the steps below, please make sure you have read and followed the steps described in [Changing the Default Storage](#).

Page Contents

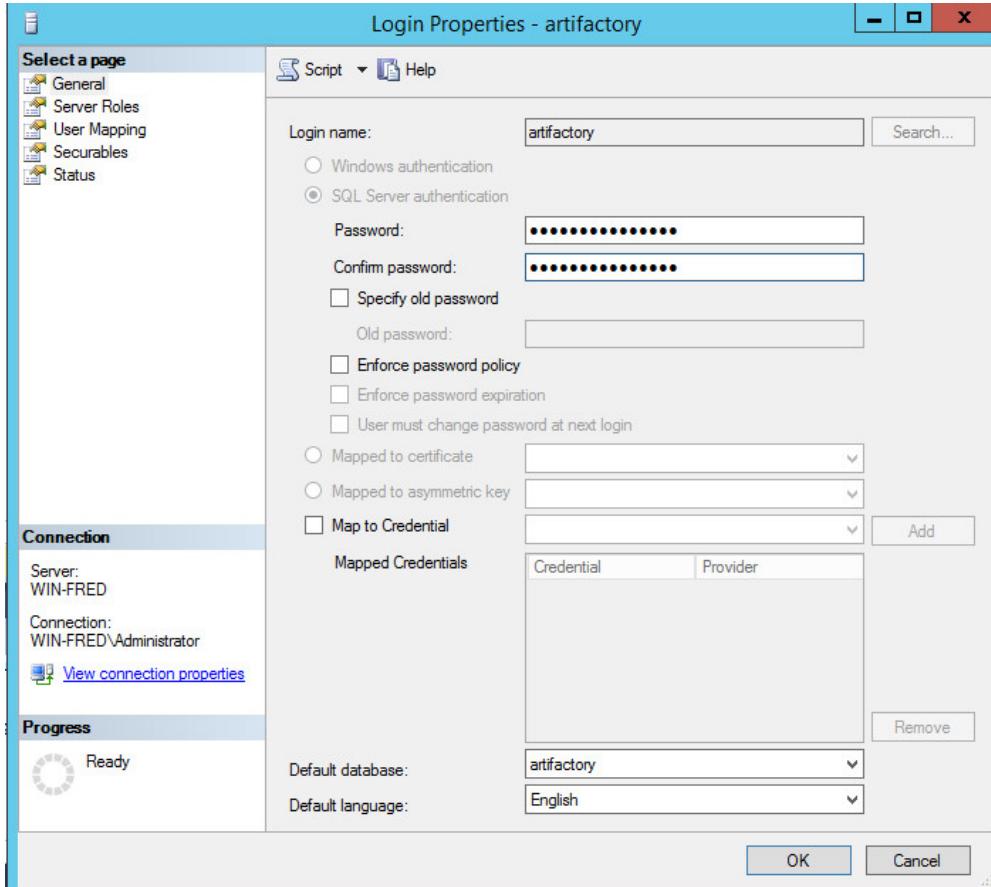
- Overview
- Creating the Artifactory Microsoft SQL Server Database
- Configuring Artifactory to use Microsoft SQL Server

Creating the Artifactory Microsoft SQL Server Database

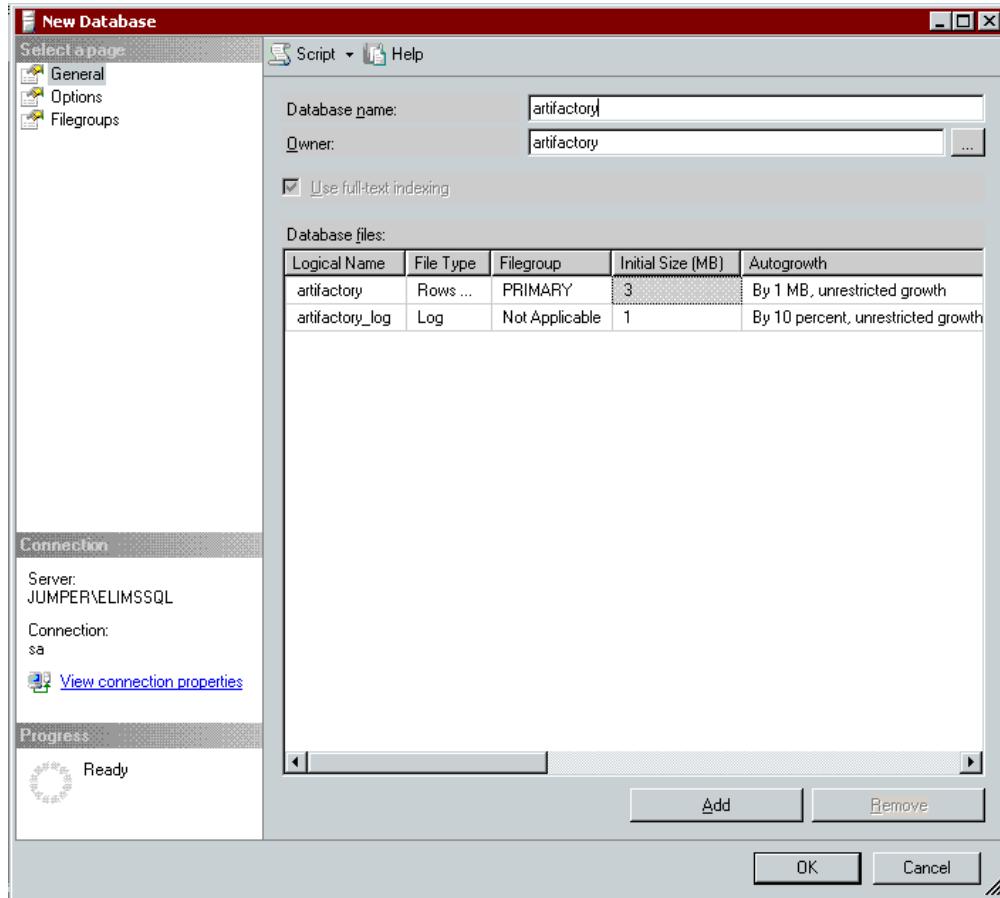
Supported Microsoft SQL Server Versions

Artifactory supports Microsoft SQL Server 2008 and above.

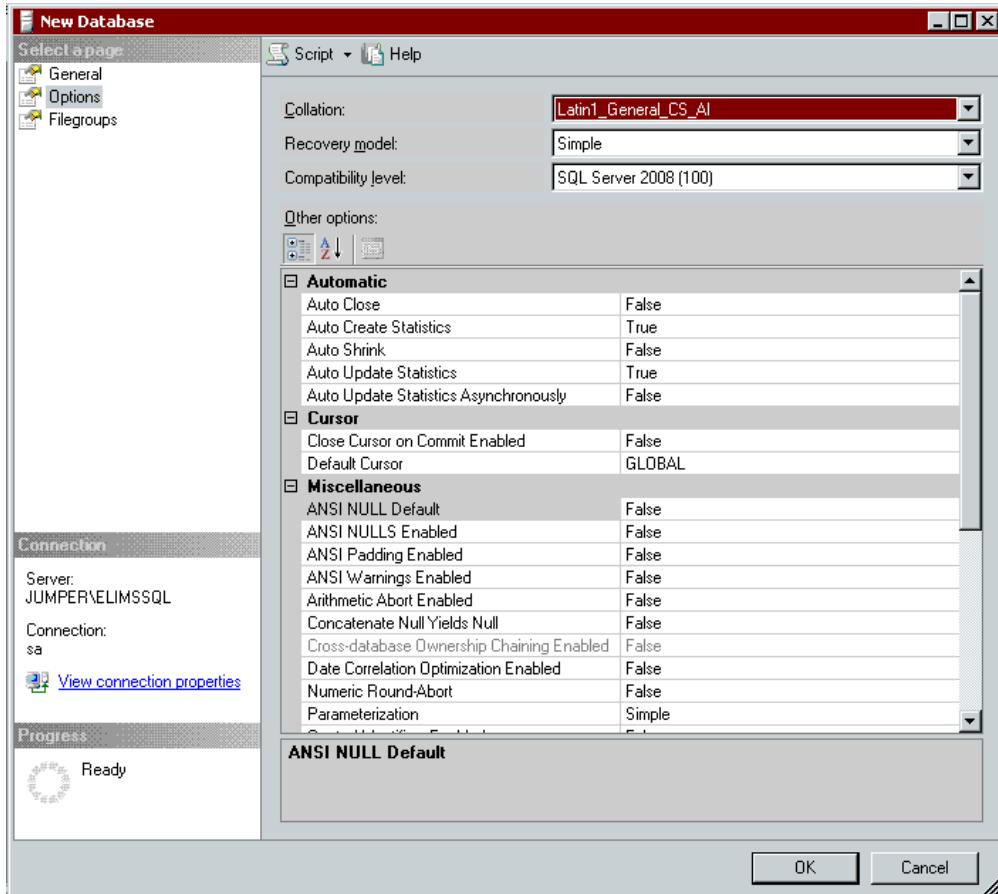
1. Create a new user for Artifactory:
In **Microsoft SQL Server Management Studio**, open the **Object Explorer**, right click on **Security** and select **New | Login....**
2. Create user "artifactory" and set its password.



3. Create the Artifactory database:
In Microsoft SQL Server Management Studio, open the **Object Explorer**, right click on **Databases** and select **New database...**
4. In the **New Database** dialog window, select **General** in the **Select a page:** navigation pane.
Set **Database name** to "artifactory" and **Owner** to "artifactory" (the user name you created in step 2).



5. Select the **Options** page and set **Collation** to "Latin1_General_CS_AI".
Then click **OK** to confirm.



Selecting a Case-sensitive Collation

While Microsoft SQL Database Server is not case-sensitive by default, it is important to select a case-sensitive collation because Artifactory is case-sensitive.

Configuring Artifactory to use Microsoft SQL Server

1. Copy `$ARTIFACTORY_HOME/misc/db/mssql.properties` to `$ARTIFACTORY_HOME/etc/storage.properties` (If you do not have this file you can take it from the standalone zip distribution or directly from the JFrog domain). For a full explanation on the contents of this file please refer to [The Bundled Storage Configurations](#).
2. Adjust the connection definitions in the `$ARTIFACTORY_HOME/etc/storage.properties` file to match the attributes of the Artifactory database you created.
You must configure the database URL and username/password to use. The schema and tables are created first time Artifactory is run using the new database.
For example:

Configuring the Database URL and user/password

```
url=jdbc:sqlserver://hostname:1433;databaseName=dbname;sendStringParametersAsUnicode=false;applicationName=Artifactory Binary Repository
```

Where `hostname` is your database address, `1433` is your database port (if not the default 1433), `dbname` is the name of the database you created in the previous step.

`sendStringParameterAsUnicode`

Make sure not to overwrite `sendStringParametersAsUnicode=false` since this is critical for appropriate and efficient use of the database indices.

3. Download and extract the Microsoft JDBC Driver and copy the `sqljdbc4.jar` file into the server's shared lib directory.
For example `$TOMCAT_HOME/lib` when installed as a service or `$ARTIFACTORY_HOME/tomcat/lib` in the standalone version.

Permissions

Make sure your driver has the same permissions as the rest of the files in the shared lib directory.

4. Start Artifactory.

PostgreSQL

Overview

By using PostgreSQL you can benefit from features in PostgreSQL infrastructure such as backup, restore and high availability.

For Artifactory to run with PostgreSQL you must create a dedicated PostgreSQL database instance and then configure Artifactory to use it as described in the following sections.

Before You Continue

Before proceeding with the steps below, please make sure you have read and followed the steps described in [Changing the Default Storage](#).

Page Contents

- [Overview](#)
- [Creating the Artifactory PostgreSQL Database](#)
- [Configuring Artifactory to use PostgreSQL](#)

Creating the Artifactory PostgreSQL Database

Supported PostgreSQL Versions

Artifactory supports PostgreSQL 9.2 and above using driver version 9.2-1002.jdbc4 and above.

The commands below create artifactory user and database with appropriate permissions.

Use the commands below to create an Artifactory user and database with appropriate permissions. Modify the relevant values to match your specific environment:

Creating an Artifactory User and Database

```
CREATE USER artifactory WITH PASSWORD 'password';
CREATE DATABASE artifactory WITH OWNER=artifactory ENCODING='UTF8';
GRANT ALL PRIVILEGES ON DATABASE artifactory TO artifactory;
```

Artifactory privileges

We recommend providing Artifactory with full privileges on the database.

Configuring Artifactory to use PostgreSQL

1. Copy `$ARTIFACTORY_HOME/misc/db/postgresql.properties` to `$ARTIFACTORY_HOME/etc/storage.properties` (If you do not have this file you can take it from the standalone zip distribution or directly from the JFrog domain). For a full explanation on the contents of this file please refer to [The Bundled Storage Configurations](#).
2. Adjust the connection definitions in the `$ARTIFACTORY_HOME/etc/storage.properties` file to match the attributes of the Artifactory database you created.
You must configure the database URL and username/password to use. The schema and tables are created first time Artifactory is run using the new database.
3. Download the JDBC driver corresponding to your PostgreSQL version from the [PostgreSQL JDBC Driver Download site](#) and copy the `postgresql-9.x-xxx.jdbc4.jar` file into the server's shared lib directory.
For example `$TOMCAT_HOME/lib` when installed as a service or `$ARTIFACTORY_HOME/tomcat/lib` in the standalone version.

Permissions

Make sure your driver has the same permissions as the rest of the files in the shared lib directory.

4. Start Artifactory.

Storing BLOBs inside PostgreSQL is not recommended

The above recommended configuration keeps all artifact information in PostgreSQL while storing the artifact binary data on the file system (under `$ARTIFACTORY_HOME/data/filestore`).

While it is possible, to store BLOBs inside PostgreSQL **we do not recommend it**. This is important because the PostgreSQL driver doesn't support streaming BLOBs with unknown length to the database. Therefore, Artifactory will temporarily save deployed files to the filesystem and only then save the BLOB to the database.

Running Behind an HTTP Server

Artifactory can run behind an HTTP server. Examples of two popular servers are:

1. Apache HTTP Server
2. nginx

Apache HTTP Server

Setting Up Apache HTTP Server

You can set up Apache HTTP Server as a front end to Artifactory using either the HTTP or AJP protocol.

```
Client -----> HTTPD -----> Artifactory  
          HTTP           HTTP/AJP
```

Using AJP

The AJP protocol offers optimized low-level binary communication between the servlet container and Apache with additional support for smart-routing and load balancing.

The configuration is flexible and can be used either with `mod_proxy_ajp`, or with `mod_jk`.

The example below shows how to configure Apache using `mod_proxy_ajp` which is distributed by default, however you need to install and then enable as follows:

Enabling mod_proxy_ajp

```
sudo a2enmod mod_proxy_ajp
```

Configuring Apache With mod_proxy_ajp Installed

The sample virtual host below refers to Apache as a reverse proxy to Tomcat, where Tomcat runs with the AJP connector on port 8019:

Page Contents

- Setting Up Apache HTTP Server
 - Using AJP
 - Configuring Apache With mod_proxy_ajp Installed
 - Configuring Your Tomcat
 - Configuring Apache With a Custom Artifactory Path
 - Using an HTTP Proxy
 - Configuring Apache With mod_proxy_ajp Installed
- Setting Up Apache HTTPS
 - Using AJP
 - Using an HTTP Proxy
 - Configuring Apache With mod_proxy_ajp Installed and Tomcat
 - Configuring a Custom URL Base in Artifactory
- Setting Up a Reverse SSL Proxy for SSO
 - Components and Versions
 - Modifying Your Webserver Configuration File
 - Configuring Artifactory for SSO

Configuring Apache with mod_ajp

```
<VirtualHost *:80>
    ServerAdmin your@email.address.com
    DocumentRoot "/srv/www/httpd/htdocs"
    ServerName artifactory.yourdomain.com
    ErrorLog "logs/artifactory-error_log"
    ProxyPreserveHost on
    ProxyPass /artifactory ajp://<yourdomain>:8019/artifactory
</VirtualHost>
```

Reset Your Cookies

When changing the Artifactory context path in Apache make sure to reset your browser's host and session cookies.

Having a stale context path value cached by cookies can lead to inconsistent issue with the user interface such as Not authorized to instantiate class errors when switching between tabs.

Configuring Your Tomcat

If you are using a dedicated Tomcat rather than the one that is bundled with the Artifactory download zip file, you must configure the AJP connector located, by default, under `$CATALINA_HOME/conf/server.xml`:

Configuring a Dedicated Tomcat

```
<Connector port="8019" protocol="AJP/1.3"
maxThreads="500" minSpareThreads="20"
enableLookups="false"
backlog="100"/>
```

Please refer to [Apache Tomcat Configuration Reference](#) for more configuration options.

Configuring Apache With a Custom Artifactory Path

You can configure Apache using the same setup as above but here the goal is to have `http://artifactory.yourdomain.com/repository/` as the root URL for Artifactory as follows:

Configuring Apache With Your Custom Artifactory Path

```
<VirtualHost *:80>
    ServerAdmin your@email.address.com
    DocumentRoot "/srv/www/httpd/htdocs"
    ServerName artifactory.yourdomain.com
    ErrorLog "logs/artifactory-error_log"
    ProxyPreserveHost on
    ProxyPass /repository ajp://<yourdomain>:8019/artifactory
    ProxyPassReverse /repository http://artifactory.yourdomain.com/artifactory
    ProxyPassReverseCookiePath /artifactory /repository
</VirtualHost>
```

Using an HTTP Proxy

When running Artifactory with Tomcat, we recommend that you set up Apache to proxy Artifactory via HTTP.

You must configure redirects correctly using the PassReverse directive, and also set the base URL in Artifactory itself so that the UI links show up correctly.

Configuring Apache With mod_proxy_ajp Installed

The sample virtual host assumes that the Tomcat HTTP connector runs on port 8081.

Ensuring HTTP Redirect Works Correctly

For HTTP redirects to work, you must set a PassReverse directive on Apache, otherwise the underlying container base URL is passed in redirects

In the example below it is set to `http://artifactory.yourdomain.com/artifactory/`.

Setting a PassReverse Directive on Apache

```
<VirtualHost *:80>
    ServerAdmin your@email.address.com
    DocumentRoot "/srv/www/httpd/htdocs"
    ServerName artifactory.yourdomain.com
    ErrorLog "logs/artifactory-error_log"
    ProxyPreserveHost on
    ProxyPass /artifactory http://<yourdomain>:8081/artifactory
    ProxyPassReverse /artifactory http://artifactory.yourdomain.com/artifactory
</VirtualHost>
```

Setting Up Apache HTTPS

You can set up Apache with SSL (HTTPS) as a front end to Artifactory using either the HTTP or AJP protocol.

```
Client -----> HTTPD -----> Artifactory
      HTTPS           HTTP/AJP
```

Using AJP

If you are not running Artifactory with Tomcat, then it is recommended to use AJP since it provides the servlet container with all the information about the correct base URL and requires no configuration in Artifactory.

Using an HTTP Proxy

Configuring Apache With `mod_proxy_ajp` Installed and Tomcat

The Apache and Tomcat sample configuration is as described in the section on Apache HTTP Server above under [Using AJP](#).

Configuring a Custom URL Base in Artifactory

When using an HTTP proxy, the links produced by Artifactory, as well as certain redirects contain the wrong port and use the `http` instead of `https`.

Therefore, you must configure a custom base URL as follows:

1. On the **Admin** tab select **Configuration | General** `Custom URL Base` field.
2. Set the **Custom URL Base** field to the value used to contact Artifactory on Apache
For example: `https://artifactory.yourdomain.com/artifactory`

Please refer to [General Configuration](#) for more details about configuring the base URL.

Setting Up a Reverse SSL Proxy for SSO

You may set up a reverse SSL proxy on your webserver in order to run Artifactory supporting SSO.

To do this, you need to have the right `components` installed, [modify your webserver configuration file](#), and then [configure Artifactory for SSO](#).

When correctly set up, you should be able to login to Artifactory with your Windows credentials and stay logged in between sessions.

Components and Versions

The instructions below have been tested to work with Kerberos/NTLM SSO

working with Artifactory using the following components.

- IBM Websphere 8.5.5 running on Windows 8 using the [IBM Websphere Java 7 JDK Package](#).
- Artifactory v3.3.0.1 or later must be installed on the Websphere instance. For details please refer to [Running Artifactory on IBM WebSphere](#).
- The `mod_auth_sspi` Apache module.

Modifying Your Webserver Configuration File

Once you have the right components and versions installed, you need to add the following lines to your `[HTTP_SERVER_HOME]/conf/httpd.conf` file:

httpd.conf file

```
<VirtualHost *:80>
ServerName yourhostname
DocumentRoot "C:/IBM/Installation
Manager/eclipse/plugins/org.apache.ant_1.8.3.v20120321-1730"
ProxyPreserveHost on
ProxyPass /artifactory http://yourhostname:9080/artifactory
ProxyPassReverse /artifactory http://yourhostname:9080/artifactory
</VirtualHost>

<Location /artifactory>
AuthName "Artifactory Realm"
AuthType SSPI
SSPIAuth On
SSPIAuthoritative On
require valid-user
RewriteEngine On
RewriteCond %{REMOTE_USER} (.+)
RewriteRule . - [E=RU:%1]
RequestHeader set REMOTE_USER %{RU}e
</Location>
```

Then you need to enable the following modules in your `httpd.conf` file:

Modules to enable

```
LoadModule sspi_auth_module modules/mod_auth_sspi.so
LoadModule headers_module modules/mod_headers.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule rewrite_module modules/mod_rewrite.so
```

Configuring Artifactory for SSO

In the Artifactory Admin module, go to Security | HTTP SSO and set the Artifactory is Proxied by a Secured HTTP Server option.

HTTP SSO

The screenshot shows the 'HTTP SSO' configuration page in Artifactory. It includes fields for 'Remote User Request Variable' (set to 'REMOTE_USER') and 'Auto Create Artifactory Users'. There are 'Reset' and 'Save' buttons at the bottom.

Artifactory is Proxied by a Secure HTTP Server

Remote User Request Variable
REMOTE_USER

Auto Create Artifactory Users

Reset **Save**

Finally, in the **General Settings** make sure that the **Server Name** and **Custom URL Base** fields are correctly filled in.

nginx

Setting Up nginx Server

You can use Artifactory behind an nginx server

```
Client -----> NGNIX -----> Artifactory
      HTTP/S           HTTP/S
```

When setting up nginx as a front end to Artifactory it is recommended to use HTTP or HTTPS.

Using HTTP or HTTPS

You must set the base URL in Artifactory itself so that the links in the user interface appear correctly.

In the example below, the configuration assumes that the Tomcat HTTP connector runs on port 8081.

Page Contents

- [Setting Up nginx Server](#)
 - [Using HTTP or HTTPS](#)
 - [Configuring Your Tomcat](#)
 - [Configuring a Custom URL Base in Artifactory](#)

Configuring nginx to use HTTP or HTTPS

```
server {
    listen *:80 ;
    server_name artifactory.yourdomain.com;
    client_max_body_size 2048M;
    access_log /var/log/nginx/artifactory.yourdomain.com.access.log;

    location /artifactory {
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://localhost:8081;
        proxy_pass_header Server;
        proxy_read_timeout 90;
    }
}

server {
    listen 443;
    server_name artifactory.yourdomain.com;

    access_log /var/log/nginx/artifactory.yourdomain.com.access.log;
    error_log /var/log/nginx/artifactory.yourdomain.com.error.log;

    ssl on;
    ssl_certificate /etc/nginx/ssl/site-name.crt;
    ssl_certificate_key /etc/nginx/ssl/site-name.key;

    ssl_session_timeout 5m;

    ssl_protocols SSLv3 TLSv1;
    ssl_ciphers ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv3:+EXP;
    ssl_prefer_server_ciphers on;

    location /artifactory {
        proxy_redirect off;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Ssl on;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_pass http://localhost:8081;
        proxy_pass_header Server;
        proxy_read_timeout 90;
    }
}
```

Internal Proxies

Regular expression (using `java.util.regex`) that a proxy's IP address must match to be considered an internal proxy. Internal proxies that appear in the `remoteIpHeader` are trusted and do not appear in the `proxiesHeader` value.

If not specified, the default value of `10\.\d{1,3}\.\d{1,3}\.\d{1,3}|192\.168\.\d{1,3}\.\d{1,3}|169\.254\.\d{1,3}\.\d{1,3}|127\.\d{1,3}\.\d{1,3}\.\d{1,3}` is used.

Configuring Your Tomcat

On Tomcat you must configure the HTTP connector as follows:

Configuring the HTTP connector

```
<Connector port="8081" protocol="HTTP/1.1"
maxThreads="500" minSpareThreads="20"
enableLookups="false" disableUploadTimeout="true"
backlog="100"/>
```

HTTP connector location

By default, the HTTP Connector can be found in `$CATALINA_HOME/conf/server.xml`

Also, the following should be added to `$TOMCAT_HOME/conf/Catalina/localhost/artifactory.xml`:

Modifying artifactory.xml

```
<Valve className="org.apache.catalina.valves.RemoteIpValve"
protocolHeader="x-forwarded-proto"/>
```

Configuring a Custom URL Base in Artifactory

When using an HTTP proxy, the links produced by Artifactory, as well as certain redirects contain the wrong port and use the `http` instead of `https`.

Therefore, you must configure a custom base URL as follows:

1. On the **Admin** tab select **Configuration | General** *Custom URL Base* field.
2. Set the **Custom URL Base** field to the value used to contact Artifactory on Apache
For example: <https://artifactory.yourdomain.com/artifactory>

Please refer to [General Configuration](#) for more details about configuring the base URL.

Upgrading Artifactory

Overview

The procedure to upgrade Artifactory depends on the version you currently have installed.

Please check the instructions below according to your current version.

Going Pro?

Even if you're just switching your current version of Artifactory OSS to the same version of Artifactory Pro, you still need to follow the same upgrade instructions specified on this page.

The one exception is for an [rpm installation](#). Since you are installing an rpm with the same version number, you need to add `--force` to the end of the command.

Upgrading Artifactory HA?

If you are upgrading an Artifactory HA cluster, please refer to [Upgrading Artifactory HA](#).

Page Contents

- Overview
- Upgrading from v3.x or v4.x to the Latest Version
 - Running as a Standalone Installation
 - Running as an RPM Installation
 - Upgrading Using YUM
 - Running in a Docker Container
- Single Package Type Repositories
 - Migrating to Single Package Type Repositories
 - Fixing Multiple Package Type Repositories
- Upgrading When Using External Servlet Containers
- Upgrading from Any Version Below v3.0
- Watch the Screencast

Before you upgrade

We strongly recommend that you do a complete [System Export](#) before commencing your upgrade procedure. If at any time you decide to roll back to your current version, you can use the export to reproduce your current system in its entirety.

Upgrading from v3.x or v4.x to the Latest Version

Upgrading from version 3.x or 4.x to the latest version is a simple procedure whether you are running as a standalone installation, an RPM installation or in a Docker container.

Before you proceed

Before proceeding, there are a few points you need to address:

1. JDK Version

From version 4.0, Artifactory requires JDK 8. If your current version is v3.x, before you upgrade to Artifactory 4.x, please make sure you install JDK 8 and update your JAVA_HOME environment variable to point to your JDK 8 installation. For more details, please refer to [System Requirements](#).

2. Repositories with Multiple Package Types

From version 4.0, Artifactory will only index, and work with corresponding clients for single package type repositories. If your current version is 3.x and the installation includes repositories that support multiple package types, you need to migrate them to single package type repositories. You may do so before upgrading or after. For more details please refer to [Single Package Type Repositories](#).

3. Servlet Containers

From version 4.0, Artifactory only supports the internal servlet container that comes with it out-of-the-box. If you are currently using an external servlet container, please first read about using [Upgrading When Using External Servlet Containers](#).

4. 'Slash' character encoding for NPM builds

Handling of 'slash' character encoding for NPM has been moved from the *artifactory.system.properties* file to the *catalina.properties* file of your Tomcat. For details, please refer to [Npm Scope Packages](#).

Running as a Standalone Installation

1. Unzip the Artifactory distribution archive.
2. If the `$ARTIFACTORY_HOME/tomcat/conf/server.xml` has been modified keep it in a temporary location.
3. If Artifactory is configured to work with a database that is not Derby, keep the `$ARTIFACTORY_HOME/tomcat/lib/<JDBC>` driver in a temporary location.
4. Remove the following files and folders from your `$ARTIFACTORY_HOME` folder:
 - `webapps/artifactory.war`
 - `tomcat`
 - `bin`
5. Replace the removed files and folders with the corresponding ones from the new unzipped version.
6. Any files that were stored in temporary locations should now be returned to their original location under the new installation.
7. If you installed Artifactory as a service, you now need to run the service
 - a. For a Linux service, browse to `$ARTIFACTORY_HOME/bin` and execute the following command **as root**: `$ARTIFACTORY_HOME/bin/installService.sh [USER [GROUP]]`
 - b. For Windows service, browse to `%ARTIFACTORY_HOME%\bin` and run `InstallService.bat`.

misc Folder

The *misc* folder contains configuration files for specialized environments such as when running Artifactory as a Standalone Installation or on [IBM Websphere](#) (which was supported for versions of Artifactory below 4.0).

Although these files are not required for runtime, it is recommended to replace this folder too.

Running as an RPM Installation

Make sure you are upgrading from v3.6 or above

When running as an RPM installation, you can only upgrade to v4.x if your current version is 3.6 or above. If necessary, first upgrade your current version to 3.6, and then upgrade to 4.x.

If you try to upgrade a version below 3.6 using `rpm --force` you may end up deleting all of your data.

1. Log in as root (or use `sudo su -`).
2. Execute the following command:

```
rpm -U jfrog-artifactory-<oss|pro>-4.y.z.rpm
```

If you are just switching from Artifactory OSS to Pro with the same version number, don't forget to append the command with `--force --nodeps`.

Upgrading Using YUM

An easy way to upgrade Artifactory from version 3.x or 4.x to the latest version is to use YUM with the Bintray Artifactory repository. The code snippets below show how to do this depending on whether your current version is below 3.6, or 3.6 and above.

▼ If your current version is 3.6 and above:

For Artifactory Pro

```
curl https://bintray.com/jfrog/artifactory-pro-rpms/rpm -o bintray-jfrog-artifactory-pro-rpms.repo && sudo mv bintray-jfrog-artifactory-pro-rpms.repo /etc/yum.repos.d
```

```
yum install jfrog-artifactory-pro
```

For Artifactory OSS

```
curl https://bintray.com/jfrog/artifactory-rpms/rpm -o bintray-jfrog-artifactory-rpms.repo && sudo mv bintray-jfrog-artifactory-rpms.repo /etc/yum.repos.d
```

```
yum install jfrog-artifactory-oss
```

▼ If your current version is below 3.6:

For Artifactory Pro

```
curl https://bintray.com/jfrog/artifactory-pro-rpms/rpm -o bintray-jfrog-artifactory-pro-rpms.repo && sudo mv bintray-jfrog-artifactory-pro-rpms.repo /etc/yum.repos.d  
yum upgrade artifactory  
yum install jfrog-artifactory-pro
```

For Artifactory OSS

```
curl https://bintray.com/jfrog/artifactory-rpms/rpm -o bintray-jfrog-artifactory-rpms.repo && sudo mv bintray-jfrog-artifactory-rpms.repo /etc/yum.repos.d  
yum upgrade artifactory  
yum install jfrog-artifactory-oss
```

Running in a Docker Container

Artifactory runs as an RPM service within a Docker container. To upgrade Artifactory, follow the instructions for [Running as an RPM Installation](#).

Single Package Type Repositories

Single Package Type

To work with version 4.x, you need to ensure that your repositories only contain artifacts with the same package type. A script to check this can be found on the [JFrog GitHub](#).

In version 3.x Artifactory supported repositories with multiple package types. You were able to upload packages with different types to the same repository and Artifactory would calculate the metadata for those packages. Nevertheless, maintaining a single package type per repository was always a best practice that optimized performance and produced a more organized repository structure in your system. From version 4.0, you need to specify a single **Package Type** for a repository when you create it. Artifactory will only calculate metadata for, and be recognized by the corresponding client software for artifacts of the **Package Type** specified for that repository. (Artifactory will not prevent you from uploading packages of a different type, however, it will not calculate metadata for those packages, and the client for the different package types will not recognize the repository).

If you currently have repositories that are configured to support multiple package types, you need to migrate them to single package type repositories, however, you may do so either before or after running the upgrade procedure.

To migrate your repositories before upgrading, please refer to [Migrating to Single Package Type Repositories](#).

If you prefer to migrate your repositories after upgrading, or have already upgraded, please refer to [Fixing Multiple Package Type Repositories](#).

Generic repositories

In version 4.x, if you need a repository to hold packages of several different types, you may specify its package type to be **Generic**. Artifactory does not calculate metadata for Generic repositories, and effectively, they behave like a simple file system to store packages.

Migrating to Single Package Type Repositories

To migrate a repository with multiple package types to single package type repositories, execute the following steps:

1. Change the configuration of the original repository so it supports only one package type.
2. For each additional packaging type needed, create a new repository with the corresponding package type
3. Use the REST API or the UI to move packages from the original repository to the new one(s) created until all repositories only contain packages of the same type.
When using the REST API, make sure to include the `suppressLayouts=1` query parameter in order to prevent artifact path transformations.

Npm Repositories

If you move data to an Npm repository, make sure to include the `.npm` folder. This will preserve extra information that may have been stored when deploying packages using the npm client.

Fixing Multiple Package Type Repositories

If you upgraded without migrating to single package type repositories, then Artifactory will start normally, however, repositories containing multiple package types will be **randomly** assigned one of single package from the original repository and output corresponding messages to the `$ARTIFACTORY_HOME/logs/artifactory.log` file.

For example, if `libs-release-local` contained three different package types: RubyGems, Npm and NuGet, after upgrading, your `$ARTIFACTORY_HOME/logs/artifactory.log` may contain messages similar to the ones below:

```
2015-06-28 10:10:47,656 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:42)
Converting repositories to a single package type
2015-06-28 10:10:47,663 [art-init] [ERROR] (o.a.v.c.v.SingleRepoTypeConverter:155)
Disabling package 'Gems' for repo 'libs-release-local' since only one packaging type
is allowed!
2015-06-28 10:10:47,664 [art-init] [ERROR] (o.a.v.c.v.SingleRepoTypeConverter:155)
Disabling package 'Npm' for repo 'libs-release-local' since only one packaging type is
allowed!
2015-06-28 10:10:47,664 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:128)
Setting repository 'libs-release-local' to type NuGet
2015-06-28 10:10:47,664 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:128)
Setting repository 'libs-snapshot-local' to type Maven
2015-06-28 10:10:47,664 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:128)
Setting repository 'plugins-release-local' to type Maven
2015-06-28 10:10:47,664 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:128)
Setting repository 'plugins-snapshot-local' to type Maven
2015-06-28 10:10:47,665 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:128)
Setting repository 'ext-release-local' to type Maven
2015-06-28 10:10:47,665 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:128)
Setting repository 'ext-snapshot-local' to type Maven
2015-06-28 10:10:47,666 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:128)
Setting repository 'jcenter' to type Maven
2015-06-28 10:10:47,666 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:128)
Setting repository 'remote-repo' to type Maven
2015-06-28 10:10:47,668 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:128)
Setting repository 'libs-snapshot' to type Maven
2015-06-28 10:10:47,668 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:128)
Setting repository 'p2' to type P2
2015-06-28 10:10:47,668 [art-init] [INFO ] (o.a.v.c.v.SingleRepoTypeConverter:56)
Finished Converting repositories to a single package type
```

In this example, Artifactory set the **Package Type** to NuGet.

To fix this condition, you can simply follow steps described above in [Migrating to Single Package Type Repositories](#), or after you upgrade, use the `packageType` utility found on the [JFrog Github](#) for 4.x migrations.

Upgrading When Using External Servlet Containers

From version 4.0 Artifactory only supports the internal servlet container it comes with out-of-the-box. Other external servlet containers (such as Websphere, JBoss etc.) as well as an external Tomcat are no longer supported.

If your current 3.x installation is running in an external servlet container, the upgrade process to version 4.x involves the following basic steps:

1. Install Artifactory 4.x in a new location but don't start it up

Install Artifactory 4.x in a new location using the normal process described in [Installing Artifactory](#).

Make sure to complete the steps below before you start up the new installation.

\$ARTIFACTORY_HOME location

\$ARTIFACTORY_HOME is location in which you unzipped the Artifactory installation file.

2. When using an external database

Copy the JDBC driver to your new installation and place it in `$ARTIFACTORY_HOME/tomcat/lib`.

3. Copy or symlink the old data from \$ARTIFACTORY_HOME to the new \$ARTIFACTORY_HOME directory

Copy or symlink the `data`, `etc` and `backup` directories from your old `$ARTIFACTORY_HOME` to the new `$ARTIFACTORY_HOME`.

4. Shut down your current installation

Perform an orderly shutdown of your current installation.

5. Start up Artifactory 4.x

Upgrading from Any Version Below v3.0

To upgrade from a version prior to 3.0, you first need to upgrade to version 3.9.x as described in [Upgrading Artifactory in the Artifactory 3 documentation](#).

Interim versions

Depending on your current version, upgrading to version 3.9.x may require you to first upgrade to an interim version.

Watch the Screencast

Using Artifactory

Overview

Artifactory provides you with the features you need to manage your binary repositories, both through an intuitive UI and with an extensive set of APIs.

The Artifactory UI is divided into four main modules:

Home

The **Home** module provides general information such as the current user, up time and version, as well as a list of add-ons that are available according to the license you have acquired.

Artifacts

The **Artifacts** module is used to browse the repositories in your system and search for artifacts using a number of different methods. While browsing through repositories, Artifactory displays useful information and provides you with tools to perform essential actions to manage your repositories.

Build

The **Build** module displays the **Build Browser** where you can view all the Ci server projects that output their builds to Artifactory. Here you can drill down to view detailed build information and compare one build to another.

Admin

The **Admin** tab is only available to users who are defined as administrators in the system, and is used to perform a variety of administration and maintenance activities such as:

- Configuring different entities such as repositories, software licenses, proxies, mail servers and more

- Managing different aspects of system security such as user definitions, groups, permissions, LDAP integration and more
- Managing backup and indexing of repositories
- Managing import and export of repositories or of the entire system
- Accessing system information and scheduling different cleanup operations

In addition to the feature set offered by the UI, Artifactory provides an extensive REST API to facilitate integration with build automation and continuous integration systems.

Page Contents

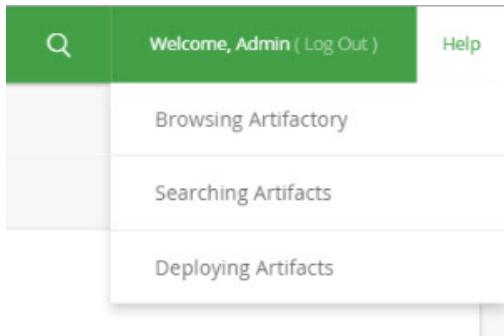
- [Overview](#)
- [Help Menu](#)
- [Set Me Up](#)

Read more

- [General Information](#)
- [Browsing Artifactory](#)
- [Using WebDAV](#)
- [Searching for Artifacts](#)
- [Deploying Artifacts](#)
- [Manipulating Artifacts](#)
- [Updating Your Profile](#)
- [Authentication](#)
- [Artifactory REST API](#)

Help Menu

On every screen of Artifactory, you can click the **Help** menu to display a list of help topics relevant for the current screen. Selecting any of the help topics opens a new browser tab displaying the corresponding Artifactory documentation.



Set Me Up

Artifactory's **Set Me Up** feature provides you with the commands you can use to deploy and resolve artifacts to and from specific repositories. Simply select any item in the [Tree Browser](#) or [Simple Browser](#) and click **Set Me Up**.

Set Me Up

X

Tool

Vagrant

Repository

vagrant-local

Deploy

To deploy Vagrant boxes to this Artifactory repository using an explicit URL with Matrix Parameters use:

```
1 curl -i -u<USERNAME>:<API_KEY> -T <PATH_TO_FILE> "http://10.100.1.110:8081/artifactory/vagrant-local/{vagrantBoxName.box};box_name={name};box_provider={provider};box_version={version}"
```



Resolve

To provision a Vagrant box, all you need is to construct it's name in the following manner.

```
1 vagrant box add "http://10.100.1.110:8081/artifactory/api/vagrant/vagrant-local/{boxName}"
```



Regardless of what was selected when you pop up this dialog, you can select a **Tool** and a specific **Repository** (the list of repositories displayed corresponds to the tool you selected), and Artifactory will provide the relevant commands according to your selection.

General Information

The **Home** module provides general information on your installation of Artifactory.

- The total number of artifacts currently being managed by the system.
- The currently logged in user
- Current version number and latest release available
- System up time
- Convenient link for you to follow JFrog on Twitter

In addition, you can view a list of add-ons that are available according to the license you have acquired.

The screenshot shows the Artifactory homepage. At the top, it displays "Artifactory is happily serving 3717 artifacts." and "You are running Artifactory version 4.x-SNAPSHOT (latest release is 3.9.2)". The sidebar includes links for Home, Artifacts, Builds, and Admin. On the right, there's a "JFrog News" section with a link to "Artifactory Query Language (AQL)" and a Twitter follow button. Below the news is a grid of icons representing various ecosystem integrations, each with a status indicator (Available or Not Available). The integrations listed include RubyGems, PyPI, Build Integration, Docker, Black Duck Integration, Bower (beta), Debian, Filtered Resources, Git LFS, Artifactory HA, LDAP Groups, 3rd Party License Control, NPM, NuGet, P2 Support, Replication, User Plugins, Advanced REST, Smart Searches, Crowd & SSO Integration, and Vagrant.

Browsing Artifactory

Overview

The **Artifacts** module in Artifactory displays the **Artifact Repository Browser** that provides two ways to browse through repositories:

- **Tree Browser:** Displays the repository as a tree
- **Simple Browser:** Focuses on the currently selected item and displays the level below it in the repository hierarchy as a flat list

Both browsers adhere to the security rules defined in the system, and only allow you to perform actions authorized by the system security policies.

To switch between browsing modes, simply select the corresponding link at the top of the **Artifact Repository Browser**.

The screenshot shows the "Artifact Repository Browser" interface in "Tree" mode. The top navigation bar has tabs for "Tree" (which is highlighted with a red border), "Simple", and a search icon. There is also a checkbox for "Compress Empty Folders". The main area displays a hierarchical tree view of repositories. Under the root node, there are two branches: "alonn-local" and "bower-local".

Both the Tree Browser and Simple Browser have features to help you navigate them and search for repositories:

- **Repository type icon:** Each repository is displayed with a distinct icon that represents its type

(local, cache, remote and virtual).

- **Search in the browser:** You can search for a specific repository in both browsers by clicking on the filter icon.
- **Keyboard navigation:** While in the browser, type the name of the repository you are searching for and Artifactory will navigate you to that repository.
- **Filters:** Click the filter icon to filter the repositories displayed in the browser to only display the types that interest you. You can also simply type the filter expression while on the browser.

Page Contents

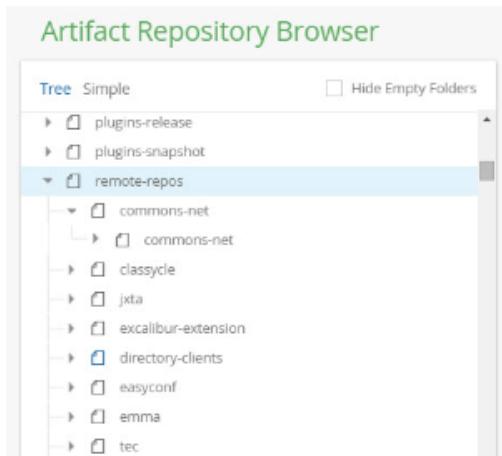
- Overview
- Tree Browsing
- Simple Browsing
- Filters
- Information Tabs
- List Browsing
- Remote Browsing
- WebDAV Browsing

Tree Browsing

The Tree Browser lets you drill down through the repository hierarchy and displays full information for each level within it. For any repository, folder or artifact selected in the tree, a tabbed panel displays detailed data views and a variety of actions that can be performed on the selected item. The information tabs available are context sensitive and depend on the item selected. For example, if you select an npm package, an Npm Info tab displays information specific to Npm packages. Similarly for NuGet, RubyGems and any other supported package formats.

Collapse All

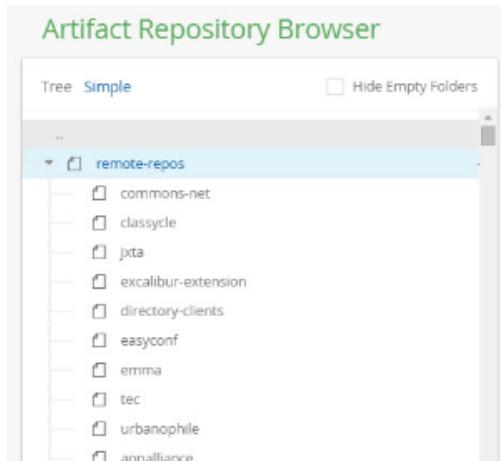
Click on the "Tree" link at the top of the tree browser to collapse all open nodes in the tree.



Simple Browsing

The simple browser lets you browse repositories using simple path-driven URLs, which are the same URLs used by build tools such as Maven. It provides lightweight, view-only browsing.

A unique feature of this browsing mode is that you can also [view remote repositories](#) (when enabled for the repository), and virtual repositories to see the true location of each folder and artifact



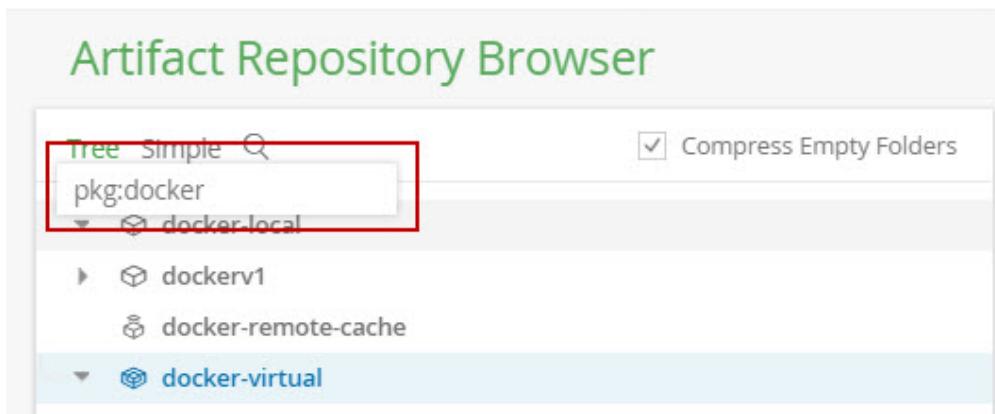
Filters

You can apply a filter to both the Tree Browser and the Simple Browser, either by clicking the search icon, or just typing out the filter expression while on the browser.

By repository name: To filter by repository name, just type the name you want to filter by

By package type: To only display repositories for a particular package type, type `pkg:<package type>`. For example, typing `pkg:docker` will filter out any repositories that are not Docker repositories

By repository type: To only display particular repository types, type `repo:<repository type>`. For example, typing `repo:local` will filter out any repositories that are not local repositories. The options are `local`, `cached`, `remote` and `virtual`.



Information Tabs

Selecting any item in the Tree or Simple browser displays tabs which provide information regarding the metadata associated with the selected item:

General	Information including download statistics such as the total number of downloads, time stamp of last download and the last user who downloaded. Checksums displays SHA1 and MD5 checksums automatically; SHA2 checksum are calculated on demand.
----------------	--

Effective Permission	The permissions (Delete/Overwrite, Deploy/Cache, Annotate, Read) that each user or group regarding the selected item. Permissions for groups are as specifically assigned to them. Permissions for individual users are the union of permissions specifically assigned as well as those inherited by virtue of the user being included in groups.
Properties	The list of properties annotating the selected item.
Watchers	The list of users watching this item.
Builds	The list of builds that either produce or use the selected item.
Governance	Information on the selected item retrieved from Black Duck Code Center.
Xml/Pom View	XML and POM files also display a tab through which you can view the file contents.

 check-1.0-sources.jar Download Actions

General	Effective Permissions	Properties	Watchers	Builds	Governance
Licenses:	Not Found Add Scan Delete Search Archive License File <small>(?)</small>				
Downloaded:	1				
Last Downloaded By:	admin				
Last Downloaded:	20-07-15 18:38:24 UTC				
Remote Downloaded:	0				
<input type="checkbox"/> Filtered <small>(?)</small>					

Package Information

Name:	aopalliance:aopalliance (?)
Description:	AOP Alliance
Latest Version:	1.0

Virtual Repository Associations

 libs-snapshot	 plugins-snapshot  qwerty
---	---

Checksums

SHA-2:	Calculate
SHA-1:	0235ba8b489512805ac13a8f9ea77a1ca5ebe3e8 (Uploaded: Identical)
MD5:	04177054e180d09e3998808efa0401c7 (Uploaded: Identical)

List Browsing

List Browsing lets you browse the contents of a repository outside of the Artifactory UI. It provides a highly responsive, read-only view and is similar to a directory listing provided by HTTP servers.

To use List Browsing, click the icon to the right of a repository name in the Simple Browser (indicated in red in the screenshot above).

Creating public views with List Browsing

List browsing can be used to provide public views of a repository by mounting it on a well-known path prefix such as `list` (see example below).

This allows the system administrator to create a virtual host that only exposes Artifactory's List Browsing feature to public users (while maintaining write and other advanced privileges), but limiting access to the intensive UI and REST API features for internal use only.

```
http://host:port/artifactory/list/repo-path
```

Index of download.eclipse.org/

Name	Last modified	Size
------	---------------	------

.index/	14-Jul-2015 15:19	-
forums/->	-	-
projects/->	-	-
releases/	30-Apr-2015 11:31	-
technology/	30-Apr-2015 11:31	-
donate->	-	-
downloads->	-	-
IRC->	-	-
IRC->	-	-

Artifactory/4.x-SNAPSHOT Server at 10.100.1.110 Port 8081

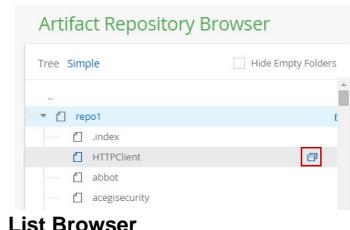
Remote Browsing

Artifactory remote browsing capabilities let you navigate the contents of the remote repository even if the artifacts have not been cached locally.

To enable remote browsing, you need to set the **List Remote Folder Items** checkbox in the remote repository configuration. Once this is set you can navigate that repository using the Simple or List Browser.

In the Simple Browser, an item that is not cached is indicated by an icon on its right when you hover over the item. In the List Browser, an item that is not cached is indicated by an arrow.

Simple Browser



Index of repo1/acegisecurity

Name	Last modified	Size
.. /		
acegi-security/	30-Apr-2015 11:31	-
acegi-security-adapters/ ->	-	-
acegi-security-cas/ ->	-	-
acegi-security-catalina/ ->	-	-
acegi-security-catalina-common/ ->	-	-
acegi-security-catalina-server/ ->	-	-
acegi-security-domain/ ->	-	-

Initial responsiveness of remote repositories

Initial remote browsing may be slow, especially when browsing a virtual repository containing multiple remote repositories. However, browsing speeds up since remote content is cached for browsing according to the [Retrieval Cache Period](#) defined in the remote repository configuration panel.

WebDAV Browsing

Artifactory fully supports browsing with WebDAV. For full details please refer to [Using WebDAV](#).

Using WebDAV

Overview

Artifactory supports WebDAV shares. A local or cached repository may be mounted as a secure WebDAV share, and made accessible from any WebDAV-supporting file manager, by referencing the URL of the target repository as follows:

```
http://host:port/artifactory/repo-path
```

When trying to deploy a file through WebDAV where file locking is enabled, the Artifactory log may display the following message:

"Received unsupported request method: lock".

In some cases this can be solved by disabling file locking before mounting the repository and is done differently for each WebDAV client. For example, for davfs2 file locking is disabled as follows:

```
echo "use_locks 0" >> /etc/davfs2/davfs2.conf
```

Note that while for some clients file locking is disabled by default, it is not necessarily possible to disable file locking in all clients.

- Authentication for davfs2 Clients

Authentication for davfs2 Clients

Davfs2 does not use preemptive authentication. Therefore, in order to authenticate using the user credentials, the client must be authenticated using two requests. The first request is sent without credentials, and receives a 401 challenge in response. Then, a second request is sent, this time with the credentials.

Anonymous access with Artifactory

Artifactory may be configured to allow anonymous access and it will therefore accept requests without authentication.

In this case, Artifactory will not respond with a 401 challenge and you will get file access with anonymous user permissions which may be less than your own user permissions.

To access your repository through Artifactory with your full user permissions you need add an authorization header to the client configuration.

This way, the requests sent to Artifactory will be authenticated and there is no need to receive a 401 challenge and respond with a second request.

Thus you are given anonymous access to Artifactory, and yet can still authenticate with your own credentials.

This can be done as follows:

1. Encode your username and password credentials in base64 using the following Groovy script:

Groovy script

```
Basic ${"username:password".bytes.encodeBase64() }
```

2. Edit the file `/etc/davfs2/davfs2.conf` or `~/.davfs2/davfs2.conf` and add the encoded credentials to the authorization header as follows:

Adding authorization header

```
add_header Authorization "Basic c2hheTpwYXNzd29yZA=="
```

Searching for Artifacts

Overview

Artifactory provides several ways for you to search for artifacts in the **Artifacts** module:

- **Quick Search:** Search by artifact file name.
- **Archive Search:** Search for files that reside within archives (e.g. within a jar file).
- **GAVC Search:** Search for artifacts according to their GAVC specification.
- **Property Search:** Search for artifacts based on names and values of properties assigned to them.
- **Checksum Search:** Search for artifacts based on their checksum value.

- **Remote Search:** Search for artifacts in Bintray's JCenter repository.

Additional advanced search features are available through the [REST API](#).

Search in Artifactory provides true real-time results that always reflect the current state of the repository with no need to periodically rebuild indexes. Therefore, search results will immediately show any artifacts deployed, and will not show any artifacts removed. The * and ? wildcards are supported in your search term to help you narrow down your search. After conducting a search, you can hover over any result item for available actions such as:

Download	Download the artifact
Show in Tree	Displays the artifact within the Tree Browser where you can view its full details
Delete	Delete the artifact

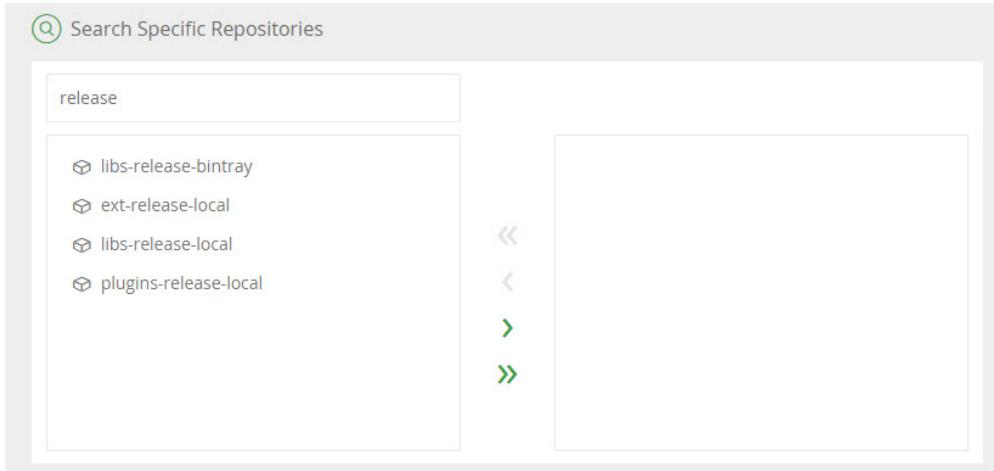
Page Contents

- Overview
- General
- Quick Search
- Archive Search
- Package Search
- Property Search
- Checksum Search
- Remote Search
- Search Results Stash

General

The different search features are available in the **Artifacts** module above the **Artifact Repository Browser**. To start a search, simply select the search method you want to use.

Each search method offers a set of input fields corresponding to the type of search you have selected. In addition, you can always narrow down your search by clicking **Search Specific Repositories**. In the table that is displayed, you can select the specific repositories that your search should be limited to. Use the **Filter** field to help display repositories you are looking for.

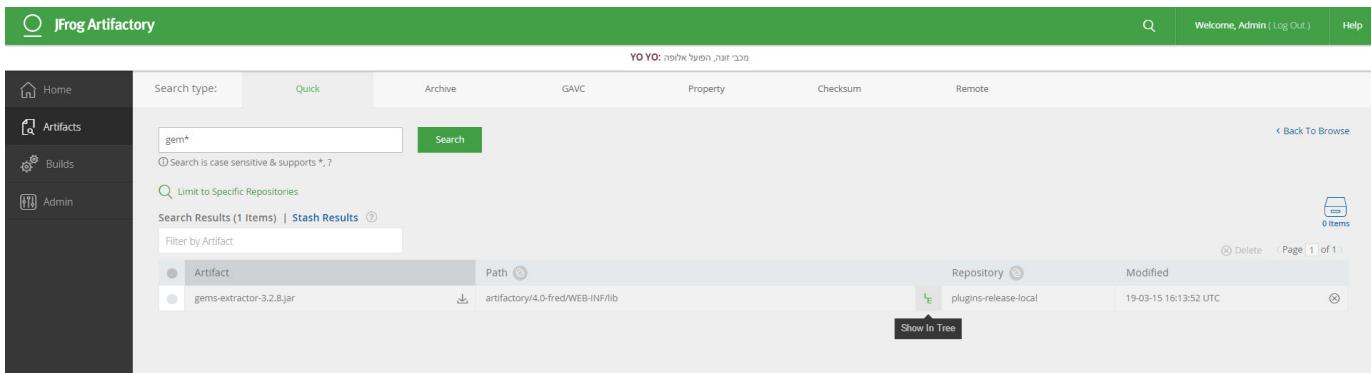
A screenshot of the 'Search Specific Repositories' interface. At the top is a search bar with the placeholder 'release'. Below it is a tree view of repositories under the 'release' category, including 'libs-release-bintray', 'ext-release-local', 'libs-release-local', and 'plugins-release-local'. To the right of the tree view are four navigation arrows: double-left, left, right, and double-right.

Case Sensitive

For all searches, the search term is case-sensitive.

Quick Search

Using Quick Search you can search for artifacts by name. Select **Quick**, enter your search term and then click the "Search" button.

A screenshot of the JFrog Artifactory UI. The top navigation bar shows 'JFrog Artifactory' and 'Welcome, Admin (Log Out)'. The main search bar has 'gem*' entered. Below the search bar, there are filter options: 'Search is case sensitive & supports *, ?' and 'Limit to Specific Repositories'. The search results table shows one item: 'Search Results (1 Items) | Stash Results'. The table columns are 'Artifact', 'Path', 'Repository', and 'Modified'. The result listed is 'gem-extractor-3.2.8.jar' located at 'artifactory/4.0-fred/WEB-INF/lib' in the 'plugins-release-local' repository, modified on '19-03-15 16:13:52 UTC'. A 'Show In Tree' button is visible below the table.

For readability, you can limit the number of results displayed by setting the following two parameters in the `$ARTIFACTORY_HOME/etc/artifactory.system.properties` file:

Configuring the number of search results

```
## Maximum number of results to return when searching through the UI
#artifactory.search.maxResults=500

## The backend limit of maximum results to return from sql queries issued by users.
## Should be higher than maxResults.
#artifactory.search.userQueryLimit=1000
```

You can run a Quick Search from any screen

You can also run a Quick Search from any screen using the search field in the top-right corner of the screen.



Archive Search

Archive search performs a search for all files that match your search parameters in an archive. Typical examples are a zip or jar file, however, all file types defined in the [MIME types](#) configuration are supported. You can specify the following parameters for your search:

Path	Allows you to specify a path filter for the search.
Name	The term to search for within the file name
Search class resources only	When checked, only class resources are searched. Any other file type is filtered out of the search
Exclude Inner Classes	When checked, inner classes are excluded from the search

The example below shows the results of searching for `.class` files (only) with a name that includes "active".

The screenshot shows the JFrog Artifactory web interface. The top navigation bar includes the logo, 'Welcome Admin (Log Out)', and 'Help'. The left sidebar has links for Home, Artifacts, Builds, and Admin. The main search area has a 'Search type' dropdown set to 'Archive', a search input field containing '*active*', and a checkbox for 'Search Class Resources Only' which is checked. Below the search bar is a table titled 'Search Results (4 items)' with columns for Name, Artifact, Artifact Path, Repository, and Modified. The results are:

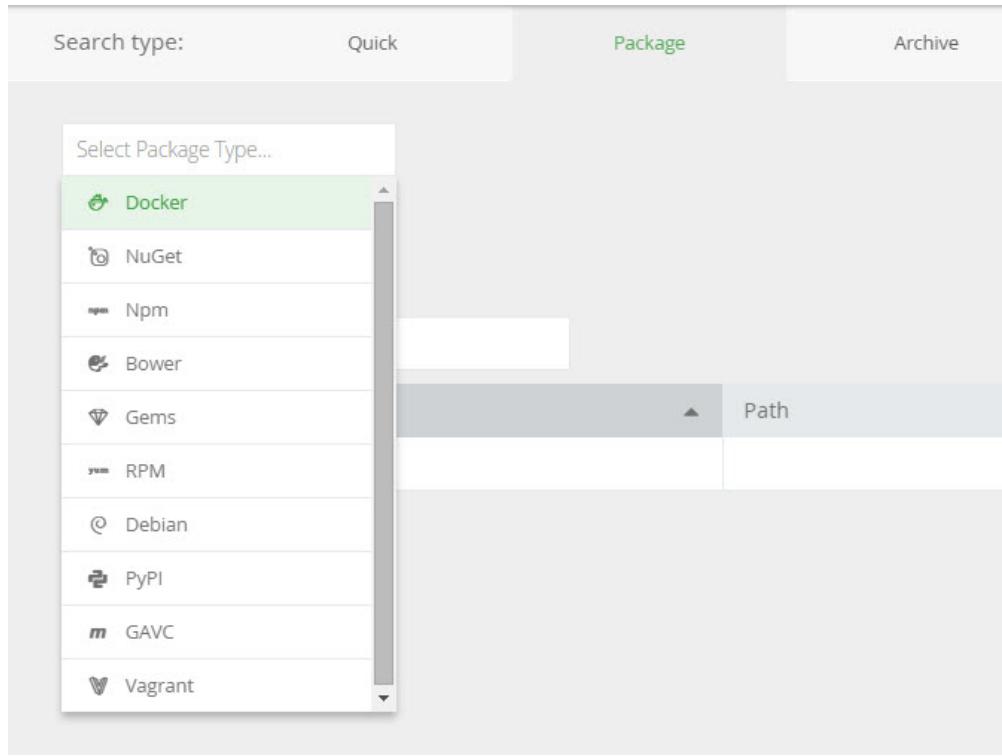
Name	Artifact	Artifact Path	Repository	Modified
com/atlassian/crowd/exception/inactiveAccountException.class	atlassian-crowd-rest-client-1.2.jar	artifactory/4.0-fred/WEB-INF/lib/	plugins-release-local	25-02-15 14:14:16 UTC
com/google/common/collect/ComparisonChain\$inactiveComparisonChain.class	guava-18.0.jar	artifactory/4.0-fred/WEB-INF/lib/	plugins-release-local	09-03-15 12:20:24 UTC
com/google/common/collect/ComparisonChain\$inactiveComparisonChain.class	guava-r08.jar	com/google/guava/guava/r08/	gradle-libs-cache	31-01-11 21:47:14 UTC
org/springframework/security/authentication/event/interactiveAuthenticationS...	spring-security-core-3.2.0.RELEASE.jar	artifactory/4.0-fred/WEB-INF/lib/	plugins-release-local	19-02-15 21:42:00 UTC

View the source file

You can hover over a class file and select **View** to view the corresponding source file if it's available.

Package Search

Package search enables you to run a search based on a specific packaging type. For each type, you can specify search parameters based on the relevant metadata for the selected package type. For example, **Docker** search is suitable for searching through Docker repositories.



The following table displays the parameters you may use for each package type:

Search type	Search parameters
<i>Docker</i>	Full Image Namespace, Image Tag, Image Digest
<i>NuGet</i>	Package ID, Version
<i>Npm</i>	Package name, Version, Scope
<i>Bower</i>	Package name, Version
<i>Gems</i>	Package name, Version
<i>RPM</i>	Package name, Version, Architecture, Release
<i>Debian</i>	File name (without the <code>.deb</code> extension), Distribution, Component, Architecture
<i>PyPI</i>	Package name, Version

GAVC	Group ID, Artifact ID, Version, Classifier
Vagrant	Box Name, Version, Provider

All these search fields support the "?" and "*" wildcard characters.

Limit search to specific repositories

When limiting search to specific repositories, Artifactory will only let you select repositories with the corresponding package type. Package search depends on those repositories having the correct layout. Searching through repositories with the wrong layout will have unpredictable and unreliable results.

The example below shows the results of a search for the latest tag of a Docker image with "ubuntu" in its name.

The screenshot shows the Artifactory search interface. The search type is set to 'Package'. The search query is 'ubuntu' with the tag 'latest'. The results table shows one item: an 'Image' named 'ubuntu' with tag 'latest' from repository 'docker-local' last modified on 'Sun Jul 26 13:02:00 UTC 2015'.

Image	Tag	Repository	Modified
ubuntu	latest	docker-local	Sun Jul 26 13:02:00 UTC 2015

Under the hood

Package search is based on standard properties that Artifactory attaches to packages according to their type. For example, when searching for NuGet packages, Artifactory is actually matching the search terms to the values for the `nuget.id` and `nuget.version` properties that should be attached to every NuGet package.

Package search as an AQL query

For most package formats, package search is implemented as an AQL query. Click the "Show AQL" button to view the AQL query used in the search. You may also click the "Copy" icon in the AQL code snippet to copy the query to your clipboard.

Limitation

Package search does not currently work on remote repository caches for RubyGems and Debian repositories.

Property Search

Artifactory Pro allows you to search for artifacts or folders based on **Properties** assigned to them, whether they are standard properties assigned by Artifactory, or custom properties which you can freely assign yourself.

To define your search parameters, in the **Property Name** field, enter the name of the property to search for, or select one from the list provided.

Then, in the **Property Value** field, set the value you are searching for in the specified property and select **Add**.

You can repeat this process to specify any number of properties and values for your search.

Wildcards can be used in the Property Value field

You can use the "?" or "*" wildcards in the **Property Value** field.

Combining properties and values

Properties are combined using the AND operator.

Different values assigned to a specific property are also combined using the AND operator. This means that only artifacts that meet all the search criteria specified will be found.

The example below shows a search for artifacts that have a build.name property beginning with "upload" (specified as upload*), and a build.number property with a value of 23

The screenshot shows the JFrog Artifactory web interface. In the top navigation bar, the user is logged in as 'Admin'. The main search bar has 'Name *' set to 'upload*' and 'Value' set to '23'. The 'Property' tab is selected. The search results table shows 9 items, all of which have 'build.name' starting with 'upload' and 'build.number' set to 23. The table includes columns for Item, Type, Path, Repository, and Modified.

Item	Type	Path	Repository	Modified
multi-2.33-20150716.145139-1.pom		org/frog/test/multi/2.33-SNAPSHOT	ext-snapshot-local	16-07-15 14:51:54 UTC
multi1-2.33-20150716.145139-1-sourcesjar		org/frog/test/multi1/2.33-SNAPSHOT	ext-snapshot-local	16-07-15 14:51:56 UTC
multi1-2.33-20150716.145139-1-tests.jar		org/frog/test/multi1/2.33-SNAPSHOT	ext-snapshot-local	16-07-15 14:51:56 UTC
multi1-2.33-20150716.145139-1.jar		org/frog/test/multi1/2.33-SNAPSHOT	ext-snapshot-local	16-07-15 14:51:55 UTC
multi1-2.33-20150716.145139-1.pom		org/frog/test/multi1/2.33-SNAPSHOT	ext-snapshot-local	16-07-15 14:51:55 UTC

Checksum Search

Artifactory allows you to search for artifacts based on MD5, SHA1 or SHA2 checksum value.

This can be especially useful if you want to identify an artifact whose name has been changed.

Wildcard characters are not supported in Checksum Search, so the term entered in the search field must be valid MD5 or SHA1 value.

The example below shows a search for an artifact whose MD5 checksum value is ed03551bf07e32184164528037654368 .

The screenshot shows the JFrog Artifactory web interface. In the top navigation bar, the user is logged in as 'Admin'. The search bar contains the MD5 checksum value '077d56067c16bf2b24bc7a566380e1e76646028'. The 'Checksum' tab is selected. The search results table shows 1 item, which has the specified MD5 checksum. The table includes columns for Artifact, Path, Repository, and Modified.

Artifact	Path	Repository	Modified
testdisk-7.0.mac_intel.tar.bz2	[root]	libs-release-local	25-05-15 14:22:23 UTC

Remote Search

Bintray is JFrog's social platform for sharing software libraries. Using this free cloud-based service, you can publish, download and share your binaries with the developer community.

For more details, please refer to the [JFrog website Bintray page](#).

Artifactory provides a direct connection to Bintray's JCenter repository which contains a comprehensive collection of popular Apache Maven packages.

To search for packages on Bintray, select **Search By: Remote** and enter the name of the package you are looking for.

Search Results (50 Results)

Name	Path	Package	Released	Cached
bifrost-bintray-plugin-0.1.jar.asc	de/envia/bifrost/bifrost-bintray-plugin/0.1.1/bifrost-bintray-plugin-0.1.1...	bifrost-bintray-plugin	25-04-15 11:44:19 UTC	
gradle-bintray-plugin-0.1.pom	com/jfrog/bintray/gradle/gradle-bintray-plugin/0.1/gradle-bintray-plugin-0...	gradle-bintray-plugin	29-06-13 21:12:09 UTC	
gradle-bintray-plugin-0.3-sources.jar	com/jfrog/bintray/gradle/gradle-bintray-plugin/0.3/gradle-bintray-plugin-0...	gradle-bintray-plugin	16-08-13 00:04:45 UTC	
gradle-bintray-plugin-0.3.jar	com/jfrog/bintray/gradle/gradle-bintray-plugin/0.3/gradle-bintray-plugin-0.3...	gradle-bintray-plugin	16-08-13 00:04:44 UTC	
gradle-bintray-plugin-0.4-javadoc.jar	com/jfrog/bintray/gradle/gradle-bintray-plugin/0.4/gradle-bintray-plugin-0.4...	gradle-bintray-plugin	23-06-14 22:20:44 UTC	
gradle-bintray-plugin-0.4.pom	com/jfrog/bintray/gradle/gradle-bintray-plugin/0.4/gradle-bintray-plugin-0.4...	gradle-bintray-plugin	23-06-14 22:20:48 UTC	

You can hover over any search result and click **Show in Bintray** to display the selected artifact in Bintray.

Search Results Stash

Requires Artifactory Pro

This feature is available with an Artifactory Pro license

Artifactory maintains a stash where you can save search results. This provides easy access to artifacts found without having to run the search again and also provides a convenient way to perform bulk operations on the result set.

For details, please refer to [Saving Search Results in the Stash](#).

Deploying Artifacts

Overview

You can deploy artifacts into a local repository of Artifactory from the **Artifacts** module by clicking **Deploy** to display the **Deploy** dialog. Artifacts can be deployed individually or in multiples.

Artifact Repository Browser

Set Me Up **Deploy**

Using Import to "deploy" a whole repository

If you want to "deploy" a whole repository, you should actually import it using the [Import Repository](#) feature in the **Admin** tab under **Import & Export | Repositories**.

You can also deploy Artifacts to a virtual repository using the REST API.

Page Contents

- Overview
- Deploying a Single Artifact
 - Deploying Maven Artifacts
 - Deploying with Properties
- Deploying Multiple Files
- Deploying an Artifact Bundle
- Deploying to a Virtual Repository
- Failed Uploads

Deploying a Single Artifact

To deploy a single artifact, simply fill in the fields in the Deploy dialog and click "Deploy".

The screenshot shows the 'Deploy' dialog box. At the top, it says 'Target Repository' with a dropdown menu showing 'libs-release-local'. Below that, 'Repository Type: Maven' is selected. Under 'Type', 'Single' is highlighted in green. In the center, there's a progress bar with the text 'Uploading jQuery.2.1.2.nupkg' and a close button. To the left, 'Target Path' is set to 'org/jfrog/testjQuery.2.1.2.nupkg'. At the bottom, there's a checkbox labeled 'Deploy as Maven Artifact' and a large green 'Deploy' button.

If you are deploying a Maven artifact, you may need to configure additional attributes as described in the next section.

Suggested Target Path

Artifactory will suggest a **Target Path** based on the details of your artifact (this works for both Maven and Ivy). For example, if a JAR artifact has an embedded POM under its internal META-INF directory, this information is used.

Deploying Maven Artifacts

If you are deploying an artifact that conforms to the Maven repository layout, you should set **Deploy as Maven Artifact** to expose fields that specify the corresponding Maven attributes - **GroupId**, **ArtifactId**, **Version**, **Classifier** and **Type**.

The fields are automatically filled in according to the artifact name, however you can edit them and your changes will also be reflected in the **Target Path**.

If your target repository does not include a POM, set **Also Deploy Jar's Internal POM/Generate Default POM**, to use the POM within the artifact you are deploying, or generate a default POM respectively.

Take care when editing the POM manually

If you are editing the POM manually, be very careful to keep it in a valid state.

Deploy

Target Repository
ext-release-local

Repository Type: Debian

Type: **Single** Multi

Uploading slf4j-api-1.7.5.jar

Target Path
org.slf4j/slf4j-api/1.7.5/slf4j-api-1.7.5.jar

Deploy as Maven Artifact

Maven Artifact

GroupId org.slf4j	ArtifactId slf4j-api
----------------------	-------------------------

Deploy

Deploy

Maven Artifact

GroupId org.slf4j	ArtifactId slf4j-api
Version 1.7.5	Classifier
Type jar	

Deploy Jar's Internal POM/Generate Default POM

```

1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
4
5   <parent>
6     <groupId>org.slf4j</groupId>
7     <artifactId>slf4j-parent</artifactId>
8     <version>1.7.5</version>
9   </parent>

```

Deploy

Deploying with Properties

Properties can be attached to the uploaded file by specifying them on the **Target Path**.

First, unset the **Deploy as Maven Artifact** check box, if necessary.

Then, in the **TargetPath** field, add the properties delimited from the path and from each other by semicolons. Each property is a key=value pair

For example, to upload an artifact with the property **qa** set to "passed", and **build.number** set to "102", use the following **Target Path**:

```
dir1/file.zip;qa=passed;build.number=102
```

Deploying Multiple Files

To deploy multiple files together, simple set the deploy **Type** to **Multi**, fill in the rest of the fields in the dialog and click "Deploy".

The screenshot shows the 'Deploy' dialog box. At the top, it says 'Target Repository' with 'libs-release-local' selected. Below that, it says 'Repository Type: Maven'. Under 'Type', 'Multi' is selected. A central area has a green 'Drop files or Select files' button and two file entries: 'artifactory-papi-2.6.4-javadoc.jar' and 'slf4j-api-1.7.5.jar', each with a delete icon. At the bottom, 'Target Path' is set to 'org/jfrog/test'. In the bottom right corner is a large green 'Deploy' button.

Deploying an Artifact Bundle

An artifact bundle is deployed as a set of artifacts packaged in an archive with one of the following supported extensions: zip, tar, tar.gz, tgz.

When you specify that an artifact should be deployed as a bundle, Artifactory will extract the archive contents when you deploy it.

File structure within the archive

Artifacts should be packaged within the archive in the same file structure with which they should be deployed to the target repository.

To deploy an artifact bundle, in the **Deploy** dialog, first upload the archive file you want to deploy.

Check the **Deploy as Bundle Artifact** checkbox and click **Deploy**.

Deploy

x

Target Repository

Package Type: Maven

Type: **Single** | Multi



Target Path [?](#)

Deploy as Bundle Artifact

Deploy

Deploying to a Virtual Repository

From version 4.2, Artifactory supports deploying artifacts to a virtual repository.

To enable this, you first need to designate one of the local repositories that is aggregated by the virtual repository as a deployment target. This can be done through the UI by setting the **Default Deployment Repository** in the **Basic Settings** of the **Edit Repository** screen.

Edit docker-virtual Repository

Basic Advanced

Docker

General

Repository Key *

Repository Layout

Public Description

Internal Description

Include Patterns

Exclude Patterns

Docker Settings

Force Authentication [?](#)

Repositories

Available Repositories

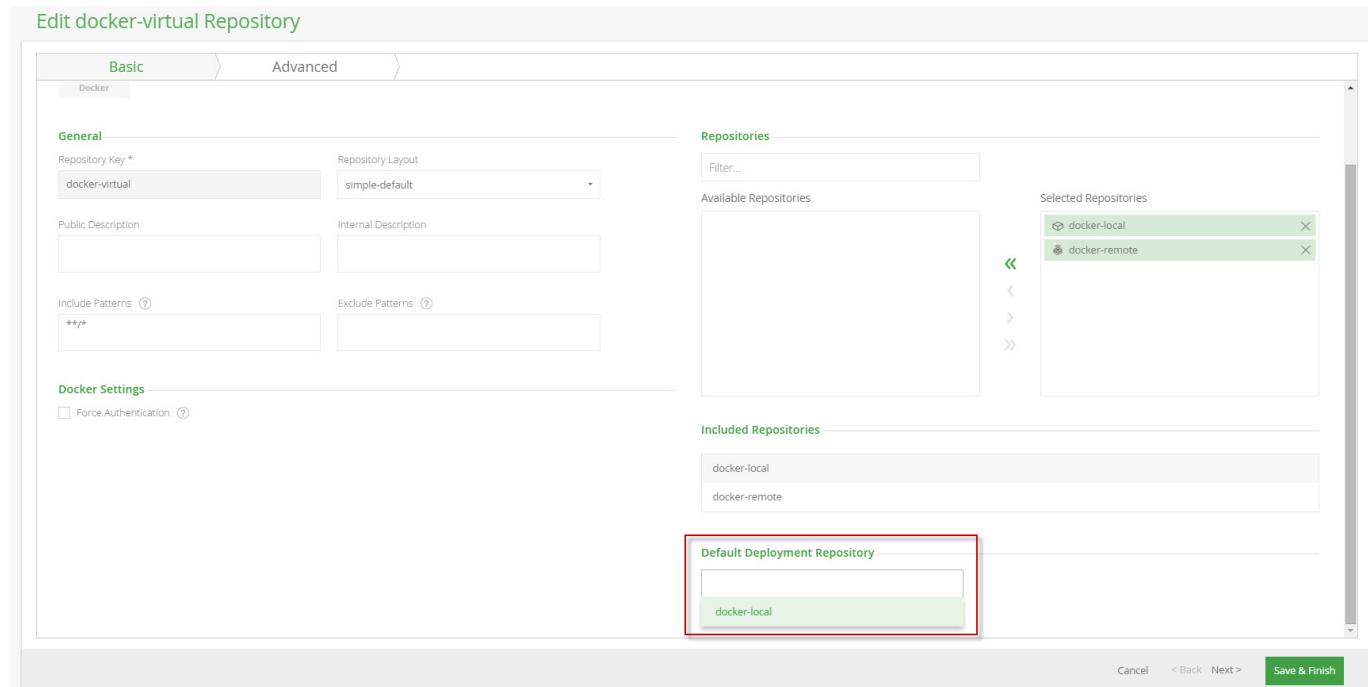
Selected Repositories

« < > »

Included Repositories

Default Deployment Repository

Cancel < Back Next > Save & Finish



You can also set the Default Deployment Repository using the `defaultDeploymentRepo` parameter of the [Virtual Repository Configuration JSON](#) used in the [Create or Replace Repository Configuration](#) and [Update Repository Configuration](#) REST API endpoints. Once the deployment target is configured, you may deploy artifacts to it using any packaging format client configured to work with Artifactory. For example, `docker push`, `npm publish`, `NuGet push`, `gem push` etc.

You can also use Artifactory's REST API to [deploy an artifact](#) and use the virtual repository key in the path to deploy.

Failed Uploads

The most common reasons for a rejected deployment are:

- Lack of permissions
- A conflict with the target repository's includes/excludes patterns
- A conflict with the target repository's snapshots/releases handling policy.

Manipulating Artifacts

Overview

Artifactory supports move, copy and deletion of artifacts to keep your repositories consistent and coherent. When an artifact is moved, copied or deleted, Artifactory immediately and automatically updates the corresponding metadata descriptors (such as `maven-metadata.xml`, RubyGems, Npm and more) to reflect the change and keep your repositories consistent with the package clients.

In addition, as a convenience feature, Artifactory provides a simple way to do a complete [version cleanup](#).

Page Contents

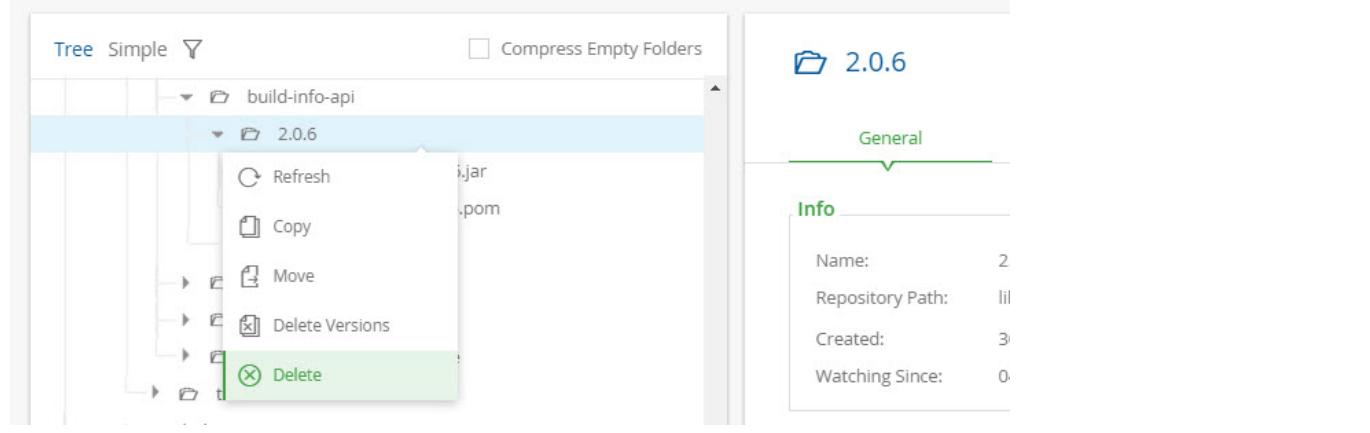
- Overview
- Deleting a Single Item
- Deleting a Version
- Moving and Copying Artifacts
 - Simulating a Move or Copy
- Downloading a Folder
 - Configuring Folder Download

Deleting a Single Item

To delete a single artifact or directory, select the item in the Tree Browser, and click **Delete** from the **Actions** menu or the right-click menu.

Once the item is deleted, the corresponding metadata file is updated to reflect the change so the item will not be found in a search.

Artifact Repository Browser



Deleting a Version

It is common for a repository to accumulate many different artifacts deployed under the same group (or path prefix) and the same version. This is especially true for snapshot deployments of multi-module projects, where all deployed artifacts use the same version. To delete a version by individually deleting its constituent artifacts can be tedious, and would normally be managed by writing a dedicated script. Artifactory lets you select one of the artifacts in a version and then delete all artifacts with the same version tag in a single click while keeping the corresponding metadata descriptors up to date.

To delete a version, right-click a folder in the Tree Browser and select **Delete Versions...**

Artifactory drills down into the selected folder and returns with a list of all the groups and versions that can be deleted.

The screenshot shows the 'Delete Versions' dialog box overlaid on the 'Artifact Repository Browser'. The browser on the left lists various local repositories like 'libs-release-local', 'matan-playground', etc. The dialog box on the right contains a table with columns: Group ID, Version, and Directories Count. It lists several entries, including 'org.jfrog.buildinfo' at version 2.0.12 (1 directory), 'org.jfrog.test' at version 2.1 (4 directories), 'org.jfrog.buildinfo' at version 2.0.6 (3 directories), 'org.jfrog.test' at version 1.0 (1 directory), 'org.jfrog.test' at version 2.1.8 (4 directories), 'org.jfrog.test' at version 2.1.7 (4 directories), and 'org.artifactory' at version 2.6.4 (1 directory). There is a 'Filter by Group ID or Version' input field at the top of the dialog, and 'Cancel' and 'Delete Selected' buttons at the bottom right.

Group ID	Version	Directories Count
org.jfrog.buildinfo	2.0.12	1
org.jfrog.test	2.1	4
org.jfrog.buildinfo	2.0.6	3
org.jfrog.test	1.0	1
org.jfrog.test	2.1.8	4
org.jfrog.test	2.1.7	4
org.artifactory	2.6.4	1

Select the versions you want to clean up and click **Delete Selected**

Limit to number of versions displayed

To avoid an excessively long search, Artifactory only displays the different version numbers assigned to the first 1000 artifacts found in the selected level of the repository hierarchy. If you do not see the version number you wish to delete, filter the artifacts displayed in the **Delete Versions** dialog by Group ID or Version number.

Moving and Copying Artifacts

To move or copy an artifact or folder, select it in the Tree Browser and then click **Move...** or **Copy...** from the **Actions** menu or from the right-click menu.

Artifact Repository Browser

The screenshot shows the 'Artifact Repository Browser' interface. On the left, there's a tree view of repositories: 'libs-snapshot-local' (expanded to show '.index', 'org' (expanded to show 'jfrog' which contains 'test'), 'maven', 'npm', 'nuget', 'p2-local' (expanded to show 'e...', 'Delete Versions', 'Delete'), and 'plugins-release-local'. A context menu is open over the 'test' folder under 'jfrog'. The menu items include 'Refresh', 'Copy', 'Move', 'Watch', 'Delete Versions', and 'Delete'. At the top right, there's a checkbox for 'Compress Empty Folders'. On the right, a detailed view of the 'test' folder is shown. It has tabs for 'General', 'Effective Permission', and 'Properties', with 'General' selected. Under 'Info', it shows: Name: test, Repository Path: libs-snapshot-local/org/jfrog/test, Created: 25-06-15 11:05:46 UTC (23d 21h 42m 40s ago), Deployed by: admin. Below that is a section for 'Virtual Repository Associations' with icons for 'libs-snapshot', 'p2', 'test-p2', and 'virt-dima'.

Artifactory will display a list of repositories from which you need to select your **Target Repository** for the operation.

The screenshot shows the 'Copy test' dialog. It has a 'Target Repository' dropdown set to 'ext-snapshot-local'. There's a checked checkbox for 'Copy to a custom path' with a field below containing 'libs-snapshot-local/org/jfrog/example'. At the bottom are 'Dry Run' and 'Copy' buttons.

After selecting your **Target Repository**, you can specify a custom **Target Path** if you want your source artifacts moved to a different location the **Target Repository**.

Copy operations are executed using Unix conventions

Copy operations are executed using Unix conventions. For example, copying `org/jfrog/1` from a source repository to `org/jfrog/1` in a target repository will result in the contents of the source being copied to `org/jfrog/1/1`.

To achieve the same path in the target repository, copy the source into one folder up in the hierarchy. In the above example, that would mean copying source `org/jfrog/1` into target `org/jfrog`.

If you leave the Target Path empty, the source will be copied into the target repository's root folder.

Custom target path suppresses cross-layout translation

If you are copying or moving your source artifacts to a repository with a different layout, specifying a **Custom Target Path** suppresses cross-layout translation. This means that your client may NOT be able to resolve the artifacts, even in cases of a same-layout operation.

Once you have selected your **Target Repository** (and **Custom Target Path** if needed), Click **Move...** or **Copy...** to complete the operation.

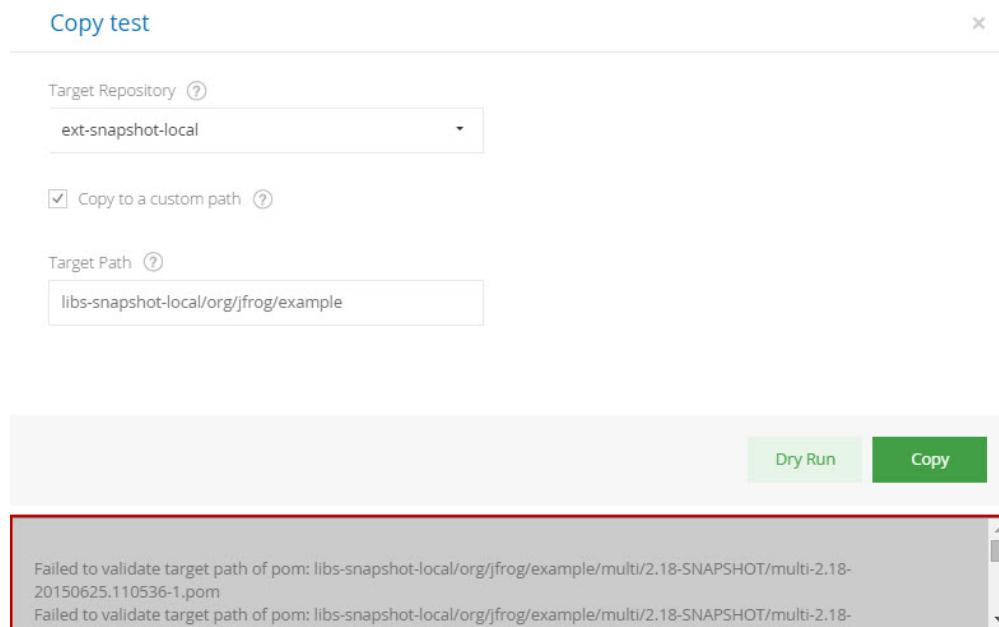
All metadata is updated to reflect the operation once completed.

Simulating a Move or Copy

Note that an operation may fail or generate warnings for several reasons. For example, the target repository may not accept all the items due to its own specific policies, or you may not have the required permissions to perform the operation.

Before actually doing an operation, we recommend that you check if it will succeed without errors or warnings by clicking **Dry Run**.

Artifactory will run a simulation and display any errors and warnings that would appear when doing the actual operation.



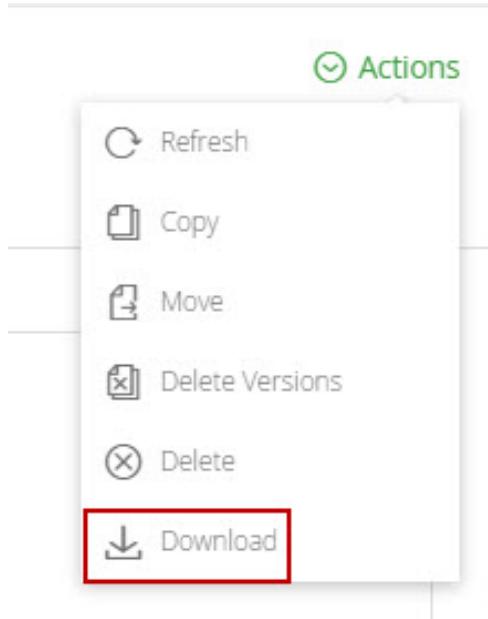
Permissions required

To successfully complete a move, you need to have **DELETE** permission on your source repository, and **DEPLOY** permission on your target repository

Downloading a Folder

Artifactory allows the download of a complete folder that is selected in the [Tree Browser](#) or [Simple Browser](#).

This ability is configurable by an Artifactory administrator, and if allowed, when a folder is selected the **Download** function is available in the **Actions** menu.



When you select folder download, Artifactory will display a dialog allowing you to select the **Archive Type**. Currently, zip, tar, tar.gz and tgz are supported.

Download Folder

Folder Name: org
Files Count: 47
Total Size: 7.85 MB

Archive Type

zip

Download

You can also download a folder using the [Rest API](#).

Configuring Folder Download

An Artifactory administrator can enable complete folder download in the **Admin** module under **Configuration | General**.

Folder Download Settings

Enable Folder Download

Max Size [?](#)

1024

Max Number of Files [?](#)

5000

Max Parallel Folder Downloads [?](#)

10

Max Size	The maximum size (MB) of artifacts that can be downloaded in a folder download.
Max Number of Files	The maximum number of artifacts that may be downloaded from the selected folder and all its sub-folders.
Max Parallel Folder Downloads	The maximum number of concurrent folder downloads allowed.

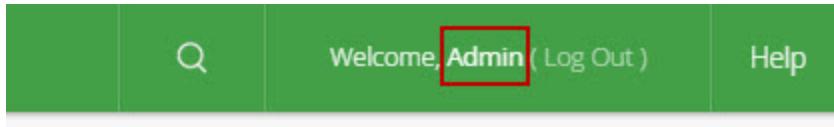
Updating Your Profile

Overview

Your profile page is used to manage the following aspects of your user profile:

- API Key
- Password
- Email
- Bintray Settings
- Binding OAuth Accounts

To display your profile page, click your login name on the top right-hand corner of the screen.



Unlocking Your Profile

To edit your profile, you first need to unlock it by entering your current password and clicking **Unlock**.

Once unlocked, you can modify all the elements of your user profile.

Page Contents

- Overview
- [Unlocking Your Profile](#)
- [Changing your Personal Settings](#)
 - API Key
 - Creating an API Key
 - Revoking or Regenerating an API Key
 - REST API
 - Changing Your Password and Email
 - Password Reminder
- Bintray Settings
- Binding OAuth Accounts

Saving your changes

Be sure to click **Update** to save any changes to your profile.

Changing your Personal Settings

Personal settings include your Artifactory API Key, password and email address.

You are not able to change your password if Artifactory is configured to use external authentication such as LDAP.

API Key

Artifactory allows authentication for REST API calls using the `X-Api-Key` header with which you can specify an API key as an alternative to your username and password.

Creating an API Key

To create an API Key, once you have unlocked your profile, click the "Generate" button next to the **API Key** field.



Revoking or Regenerating an API Key

Once an API Key is created, it displayed, masked, in the corresponding field. Click the "View" icon to see the API Key in clear-text, or the "Copy" icon to copy the API Key to the clipboard.

To revoke the current API Key, click "Revoke API Key". Note that any REST API calls using the current API key for authentication will no longer be valid.

You may revoke the current API Key and create a new one in a single action by clicking "Regenerate". Any REST API calls using the current API key for authentication will no longer be valid, until you replace the API Key with the new one you just generated.



REST API

The following REST API endpoints are available with regard to API Keys:

<i>Endpoint</i>	<i>Description</i>
<i>Create API Key</i>	Create an API key for the current user.
<i>Get API Key</i>	Get the current user's own API key.
<i>Revoke API Key</i>	Revokes the current user's API key.
<i>Revoke User API Key</i>	Revokes the API key of another user (requires Admin privileges).
<i>Revoke All API Keys</i>	Revokes all API keys currently defined in the system (requires Admin privileges).

Changing Your Password and Email

Once your profile is unlocked, Artifactory displays your password in an encrypted format that can be used whenever you need to provide your password in an environment that is not secure. For example, when making REST API calls over HTTP.

The encrypted password is initially masked, but you may click the "View" icon to view the encrypted password in clear-text. You may also click the "Copy" icon to copy the encrypted password to the clipboard.

To change your Artifactory password, enter your new password and verify it.

You can also modify your email address.

Personal Settings

New Password

Retype Password

Email Address *

Authentication Settings

API Key

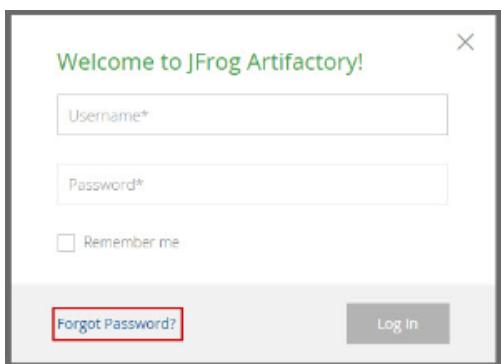
Encrypted Password

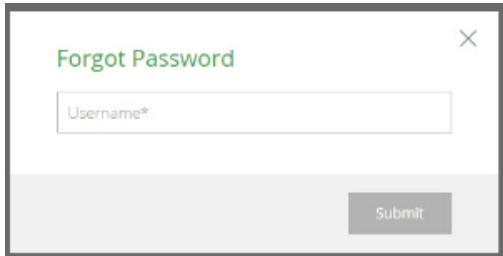
(X) Revoke API Key

For more information about using secured passwords with your profile, please refer to [Centrally Secure Passwords](#).

Password Reminder

If you forget your password, on the Artifactory Login dialog, select **Forgot Password**, and enter your username in the following dialog that is displayed.





When you click **Submit**, Artifactory will send a message to the email address configured for your user account, with a link you can click on to reset your password.

Bintray Settings

Bintray is JFrog's platform to store and distribute your software libraries. For more details please refer to the [JFrog Bintray Page](#).

Upon installation, Artifactory has Bintray's JCenter Java repository defined as a remote repository.

You may freely read from JCenter, and other Bintray repositories, however to upload an artifact, or perform other operations on Bintray through Artifactory such as search, you must have a Bintray account and provide your credentials through your profile page.

To provide your Bintray credentials, enter your **Bintray Username** and the **Bintray API Key** you received when registering to Bintray into the specified fields in Artifactory.

To verify that your credentials are valid you can click **Test**.

Bintray Settings

Bintray Username

Bintray API Key

 (@)

[Register to Bintray...](#)

Test

Binding OAuth Accounts

From version 4.2, Artifactory is integrated with OAuth allowing you to log in through your account with one of the configured OAuth providers. To do so, you need to bind your OAuth account to your Artifactory user by clicking **Click to bind** next to the corresponding OAuth provider. For more details, please refer to [OAuth Integration](#).

OAuth User Binding



Git Enterprise

[Click to bind](#)



Google-JFrog

[Click to bind](#)



OpenID

[Click to bind](#)

Authentication

Overview

The following sections describe all the means of authentication available in Artifactory.

Basic Authentication

Artifactory provides a detailed and flexible permission-based system to control users' access to different features and artifacts.

For more details, please refer to [Configuring Security](#).

LDAP

Artifactory supports authenticating users against an LDAP server out-of-the-box. When LDAP authentication is active, Artifactory first attempts to authenticate the user against the LDAP server. If LDAP authentication fails, Artifactory tries to authenticate via its internal database. For every LDAP authenticated user Artifactory creates a new user in the internal database (provided that the user does not already exist), and automatically assigns that user to the default groups.

For more details, please refer to [Managing Security with LDAP](#).

Page Contents

- [Overview](#)
- [Basic Authentication](#)
- [LDAP](#)
- [Active Directory](#)
- [Single Sign-On](#)
- [SAML](#)
- [OAuth](#)
- [Atlassian Crowd Integration](#)
- [Custom Authentication with User Plugins](#)

Active Directory

Artifactory supports integration with an Active Directory server to authenticate users and synchronize groups. When authentication using Active Directory is configured and active, Artifactory first attempts to authenticate the user against the Active Directory server. If the authentication fails, Artifactory tries to authenticate via its internal database. For every externally authenticated user configured in your Active Directory server, Artifactory creates a new user in the internal database (provided the user does not already exist), and automatically assigns that user to the default groups.

For more details, please refer to [Managing Security with Active Directory](#).

Single Sign-On

The Single Sign-on (SSO) Add-on allows you to reuse existing HTTP-based SSO infrastructures with Artifactory, such as the SSO modules offered by Apache HTTPD. Artifactory's authentication will work with commonly available SSO solutions, such as native NTLM, Kerberos, etc... SSO works by letting Artifactory know what trusted information it should look for in the HTTP request, assuming that this request has already been authenticated by the SSO infrastructure, which sits in front of Artifactory.

For more details, please refer to [Single Sign-on](#).

SAML

SAML is an XML standard that allows you to exchange user authentication and authorization information between web domains. JFrog's Artifactory offers a SAML-based Single Sign-On service allowing federated Artifactory partners (identity providers) full control over the authorization process. Using SAML, Artifactory acts as service provider which receives users authentication information from external identity providers. In such case Artifactory is no longer responsible to authenticate the user although it still has to redirect the login request to the identity provider and verify the integrity of the identity provider's response.

For more details, please refer to [SAML SSO Integration](#).

OAuth

OAuth integration allows you to delegate authentication requests to external providers and let users login to Artifactory using their accounts with those providers. Currently, Google, OpenID Connect, GitHub Enterprise and Cloud Foundry UAA are supported.

For more details, please refer to [OAuth Integration](#).

Atlassian Crowd Integration

The Atlassian Crowd Integration allows you to delegate authentication requests to Atlassian Crowd, use authenticated Crowd users and have Artifactory participate in a transparent SSO environment managed by Crowd. In addition, Atlassian Crowd Integration allows the use of JIRA User Server as an authentication server, but without support of SSO.

For more details, please refer to [Atlassian Crowd Integration](#).

Custom Authentication with User Plugins

You can use User Plugins to implement custom authentication policies.

For more details, please refer to [Management of Security Realms](#).

Artifactory REST API

WADL

Artifactory exposes its REST API through an auto-generated WADL file (courtesy of the [Jersey REST framework](#)).

This provides a convenient and up-to-date self-descriptive API and can be used by various tools/frameworks to automate the creation of REST calls.

The WADL file is available at the following URL:

<http://server:port/artifactory/api/application.wadl>

REST Resources

The sections below provide a comprehensive listing of the REST resources exposed by Artifactory.

For details on handling errors please refer to [ERROR RESPONSES](#) below.

Usage of REST resources is subject to security restrictions applicable to each individual resource.

BUILDS

All Builds

Description: Provides information on all builds

Since: 2.2.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/build

Produces: application/vnd.org.jfrog.build.Builds+json

Sample Output:

```
GET /api/build
{
  "uri": "http://localhost:8080/artifactory/api/build"
  "builds" : [
    {
      "uri" : "/my-build",
      "lastStarted" : ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)
    },
    {
      "uri" : "/jackrabbit",
      "lastStarted" : ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)
    }
  ]
}
```

Build Runs

Description: Build Runs

Since: 2.2.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/build/{buildName}

Produces: application/vnd.org.jfrog.build.BuildsByName+json

Sample Output:

```
GET /api/build/my-build
{
  "uri":
  "http://localhost:8080/artifactory/api/build/my-build"
  "buildsNumbers" : [
    {
      "uri" : "/51",
      "started" : ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)
    },
    {
      "uri" : "/52",
      "started" : ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)
    }
  ]
}
```

Build Upload

Description: Upload Build

Security: Requires a privileged user (can be anonymous)

Notes: All build modules must have the `build.name` and `build.number` properties set as well as the correct SHA1 and MD5 in order to be properly linked in the build info.

Usage: PUT /api/build/ -H "Content-Type: application/json" --upload-file build.json

Consumes: application/vnd.org.jfrog.build.BuildsByName+json

Example: curl -X PUT "<http://localhost:8080/artifactory/api/build>" -H "Content-Type: application/json"

--upload-file build.json

Sample input:

▼ [Click to view sample build.json](#)

```
{  
    "version" : "1.0.1",  
    "name" : "Maven2-3",  
    "number" : "9",  
    "type" : "MAVEN",  
    "buildAgent" : {  
        "name" : "Maven",  
        "version" : "3.0.5"  
    },  
    "agent" : {  
        "name" : "Jenkins",  
        "version" : "1.565.2"  
    },  
    "started" : "2015-03-20T11:01:38.445+0200",  
    "durationMillis" : 8926,  
    "artifactoryPrincipal" : "admin",  
    "url" : "http://localhost:8080/job/Maven2-3/9/",  
    "vcsRevision" :  
"83049487ecc61bef3dce798838e7a9457e174a5a",  
    "vcsUrl" : "https://github.com/aseftel/project-examples",  
    "licenseControl" : {  
        "runChecks" : false,  
        "includePublishedArtifacts" : false,  
        "autoDiscover" : true,  
        "scopesList" : "",  
        "licenseViolationsRecipientsList" : ""  
    },  
    "buildRetention" : {  
        "count" : -1,  
        "deleteBuildArtifacts" : true,  
        "buildNumbersNotToBeDiscarded" : [ ]  
    },  
    "modules" : [ {  
        "properties" : {  
            "project.build.sourceEncoding" : "UTF-8"  
        },  
        "id" : "org.jfrog.test:multi:2.19-SNAPSHOT",  
        "artifacts" : [ {  
            "type" : "pom",  
            "sha1" : "045b66ebbf8504002b626f592d087612aca36582",  
            "md5" : "c25542a353dab1089cd186465dc47a68",  
            "name" : "multi-2.19-SNAPSHOT.pom"  
        } ]  
    }, {  
        "properties" : {
```

```
        "project.build.sourceEncoding" : "UTF-8"
    },
    "id" : "org.jfrog.test:multi1:2.19-SNAPSHOT",
    "artifacts" : [ {
        "type" : "jar",
        "sha1" : "f4c5c9cb3091011ec2a895b3dedd7f10d847361c",
        "md5" : "d1fd850a3582efba41092c624e0b46b8",
        "name" : "multi1-2.19-SNAPSHOT.jar"
    }, {
        "type" : "pom",
        "sha1" : "2ddbf9824676f548d637726d3bcb494ba823090",
        "md5" : "a64aa7f305f63a85e63a0155ff0fb404",
        "name" : "multi1-2.19-SNAPSHOT.pom"
    },
    ...
    {
        "type" : "jar",
        "sha1" : "6fdd143a44cea3a2636660c5c266c95c27e50abc",
        "md5" : "12a1e438f4bef8c4b740fe848a1704a4",
        "id" : "org.slf4j:slf4j-simple:1.4.3",
        "scopes" : [ "compile" ]
    }, {
        "type" : "jar",
        "sha1" : "496e91f7df8a0417e00cecd8a840cdf0e5f2472c",
        "md5" : "76a412a37c9d18659d2daccdb1c24ff",
        "id" : "org.jenkins-ci.lib:dry-run-lib:0.1",
        "scopes" : [ "compile" ]
    }
], {
    "properties" : {
        "daversion" : "2.19-SNAPSHOT",
        "project.build.sourceEncoding" : "UTF-8"
    },
    "id" : "org.jfrog.test:multi2:2.19-SNAPSHOT",
    "artifacts" : [ {
        "type" : "txt",
        "name" : "multi2-2.19-SNAPSHOT.txt"
    }, {
        "type" : "java-source-jar",
        "sha1" : "49108b0c7db5fdb4efe3c29a5a9f54e806aecb62",
        "md5" : "0e2c5473cf2a9b694afb4a2e8da34b53",
        "name" : "multi2-2.19-SNAPSHOT-sources.jar"
    },
    ...
    {
        "type" : "jar",
        "sha1" : "476e89d290ae36dabb38ff22f75f264ae019d542",
        "md5" : "fa9b3df58ac040fffccff9310f261be80",
        "name" : "multi2-2.19-SNAPSHOT.jar"
    }, {
        "type" : "pom",
        "sha1" : "b719b90364e5ae38cda358072f61f821bdae5d5d",
        "md5" : "8d5060005235d75907baca4490cf60bf",
        "name" : "multi2-2.19-SNAPSHOT.pom"
    }
],
```

```
"dependencies" : [ {
    "type" : "jar",
    "sha1" : "19d4e90b43059058f6e056f794f0ea4030d60b86",
    "md5" : "dc95bcb84b09897b2b66d4684c040da",
    "id" : "xpp3:xpp3_min:1.1.4c",
    "scopes" : [ "provided" ]
}, {
    "type" : "jar",
    "sha1" : "e2d866af5518e81282838301b49a1bd2452619d3",
    "md5" : "e9e4b59c69305ba3698dd61c5dfc4fc8",
    "id" : "org.jvnet.hudson.plugins:perforce:1.3.7",
    "scopes" : [ "compile" ]
},
...
{
    "type" : "jar",
    "sha1" : "6fdd143a44cea3a2636660c5c266c95c27e50abc",
    "md5" : "12ale438f4bef8c4b740fe848a1704a4",
    "id" : "org.slf4j:slf4j-simple:1.4.3",
    "scopes" : [ "compile" ]
}, {
    "type" : "jar",
    "sha1" : "496e91f7df8a0417e00cecd8a40cdf0e5f2472c",
    "md5" : "76a412a37c9d18659d2daccdb1c24ff",
    "id" : "org.jenkins-ci.lib:dry-run-lib:0.1",
    "scopes" : [ "compile" ]
} ],
{
    "properties" : {
        "project.build.sourceEncoding" : "UTF-8"
    },
    "id" : "org.jfrog.test:multi3:2.19-SNAPSHOT",
    "artifacts" : [ {
        "type" : "java-source-jar",
        "sha1" : "3cd104785167ac37ef999431f308ffef10810348",
        "md5" : "c683276f8dda97078ae8eb5e26bb3ee5",
        "name" : "multi3-2.19-SNAPSHOT-sources.jar"
    }, {
        "type" : "war",
        "sha1" : "34aeebeb805b23922d9d05507404533518cf81e4",
        "md5" : "55af06a2175cfb23cc6dc3931475b57c",
        "name" : "multi3-2.19-SNAPSHOT.war"
    },
    ...
},
{
    "type" : "jar",
    "sha1" : "496e91f7df8a0417e00cecd8a40cdf0e5f2472c",
    "md5" : "76a412a37c9d18659d2daccdb1c24ff",
    "id" : "org.jenkins-ci.lib:dry-run-lib:0.1",
    "scopes" : [ "compile" ]
}, {
    "type" : "jar",
    "sha1" : "7e9978fdb754bce5fc5161133e7734ecb683036",
    "md5" : "7df83e09e41d742cc5fb20d16b80729c",
    "id" : "hsqldb:hsqldb:1.8.0.10",
}
```

```
        "scopes" : [ "runtime" ]
    } ]
}
},
"governance" : {
    "blackDuckProperties" : {
        "runChecks" : false,
        "includePublishedArtifacts" : false,
        "autoCreateMissingComponentRequests" : false,
```

```
        "autoDiscardStaleComponentRequests" : false
    }
}
```

Build Info

Description: Build Info

Since: 2.2.0

Security: Requires a privileged user with deploy permissions (can be anonymous)

Usage: GET /api/build/{buildName}/{buildNumber}

Produces: application/vnd.org.jfrog.build.BuildInfo+json

Sample Output:

```
GET /api/build/my-build/51
{
  "uri": "http://localhost:8080/artifactory/api/build/my-build/51",
  "buildInfo" : {
    ...
  }
}
```

Builds Diff

Description: Compare a build artifacts/dependencies/environment with an older build to see what has changed (new artifacts added, old dependencies deleted etc).

Since: 2.6.6

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/build/{buildName}/{buildNumber}?diff={OlderbuildNumber}

Produces: application/vnd.org.jfrog.build.BuildsDiff+json

Sample Output:

```
GET /api/build/my-build/51?diff=50
{
  "artifacts": {
    "updated": [],
    "unchanged": [],
    "removed": [],
    "new": []
  }, "dependencies": {
    "updated": [],
    "unchanged": [],
    "removed": [],
    "new": []
  }, "properties": {
    "updated": [],
    "unchanged": [],
    "removed": [],
    "new": []
  }
}
```

Page contents

- [WADL](#)
- [REST Resources](#)
 - [BUILDs](#)
 - [All Builds](#)
 - [Build Runs](#)
 - [Build Upload](#)
 - [Build Info](#)
 - [Builds Diff](#)
 - [Build Promotion](#)
 - [Delete Builds](#)
 - [Build Rename](#)
 - [Push Build to Bintray](#)
 - [ARTIFACTS & STORAGE](#)
 - [Folder Info](#)
 - [File Info](#)
 - [Get Storage Summary Info](#)
 - [Item Last Modified](#)
 - [File Statistics](#)
 - [Item Properties](#)
 - [Set Item Properties](#)
 - [Delete Item Properties](#)
 - [Set Item SHA256 Checksum](#)
 - [Retrieve Artifact](#)
 - [Retrieve Latest Artifact](#)
 - [Retrieve Build Artifacts Archive](#)
 - [Retrieve Folder or Repository Archive](#)
 - [Trace Artifact Retrieval](#)
 - [Archive Entry Download](#)
 - [Create Directory](#)
 - [Deploy Artifact](#)
 - [Deploy Artifact by Checksum](#)
 - [Deploy Artifacts from Archive](#)
 - [Push a Set of Artifacts to Bintray](#)
 - [Push Docker Tag to Bintray](#)
 - [File Compliance Info](#)
 - [Delete Item](#)
 - [Copy Item](#)
 - [Move Item](#)
 - [Get Repository Replication Configuration](#)
 - [Set Repository Replication Configuration](#)
 - [Update Repository Replication Configuration](#)

- Delete Repository Replication Configuration
- Scheduled Replication Status
- Pull/Push Replication
- Create or Replace Local Multi-push Replication
- Update Local Multi-push Replication
- Delete Local Multi-push Replication
- Artifact Sync Download
- Folder Sync (Deprecated)
- File List
- **SEARCHES**
 - Artifactory Query Language (AQL)
 - Artifact Search (Quick Search)
 - Archive Entry Search (Class Search)
 - GAVC Search
 - Property Search
 - Checksum Search
 - Bad Checksum Search
 - Artifacts Not Downloaded Since
 - Artifacts With Date in Date Range
 - Artifacts Created in Date Range
 - Pattern Search
 - Builds for Dependency
 - License Search
 - Artifact Version Search
 - Artifact Latest Version Search Based on Layout
 - Artifact Latest Version Search Based on Properties
 - Build Artifacts Search
- **SECURITY**
 - Get Users
 - Get User Details
 - Get User Encrypted Password
 - Create or Replace User
 - Update User
 - Delete User
 - Create API Key
 - Get API Key
 - Revoke API Key
 - Revoke User API Key
 - Revoke All API Keys
 - Get Groups
 - Get Group Details
 - Create or Replace Group
 - Update Group
 - Delete Group
 - Get Permission Targets
 - Get Permission Target Details
 - Create or Replace Permission Target
 - Delete Permission Target
 - Effective Item Permissions
 - Security Configuration
 - Save Security Configuration
 - Activate Master Key Encryption
 - Deactivate Master Key Encryption
 - Set GPG Public Key
 - Get GPG Public Key
 - Set GPG Private Key
 - Set GPG Pass Phrase
- **REPOSITORIES**
 - Get Repositories
 - Repository Configuration
 - Create or Replace Repository Configuration
 - Update Repository Configuration
 - Delete Repository
 - Remote Repository Configuration
 - Calculate YUM Repository Metadata
 - Calculate NuGet Repository Metadata
 - Calculate Npm Repository Metadata
 - Calculate Maven Index
 - Calculate Maven Metadata
 - Calculate Debian Repository Metadata
- **SYSTEM & CONFIGURATION**
 - System Info
 - System Health Ping
 - General Configuration
 - Save General Configuration

- Update Custom URL Base
- License Information
- Install License
- Version and Add-ons information
- **PLUGINS**
 - Execute Plugin Code
 - Retrieve All Available Plugin Info
 - Retrieve Plugin Info Of A Certain Type
 - Retrieve Build Staging Strategy
 - Execute Build Promotion
 - Reload Plugins
- **IMPORT & EXPORT**
 - Import Repository Content
 - Import System Settings Example
 - Full System Import
 - Export System Settings Example
 - Export System
- **ERROR RESPONSES**

Read More

- [Repository Configuration JSON](#)
- [Security Configuration JSON](#)
- [System Settings JSON](#)

Build Promotion

Description: Change the status of a build, optionally moving or copying the build's artifacts and its dependencies to a target repository and setting properties on promoted artifacts.

All artifacts from all scopes are included by default while dependencies are not. Scopes are additive (or).

Since: 2.3.3

Notes: Requires Artifactory Pro

Security: Requires a privileged user (can be anonymous)

Usage: POST /api/build/promote/{buildName}/{buildNumber}

Consumes: application/vnd.org.jfrog.artifactory.build.PromotionRequest+json

```

POST /api/build/promote/my-build/51
{
  "status": "staged" // new build status (any string)
  "comment" : "Tested on all target platforms." // An optional comment describing the reason for promotion. Default: ""
  "ciUser": "builder" // The user that invoked promotion from the CI server
  "timestamp" : ISO8601 // the time the promotion command was received by Artifactory
  "dryRun" : false // run without executing any operation in Artifactory, but get the results to check if the operation can succeed. Default: false
  "targetRepo" : "libs-release-local" // optional repository to move or copy the build's artifacts and/or dependencies
  "copy": false // whether to copy instead of move, when a target repository is specified. Default: false
  "artifacts" : true // whether to move/copy the build's artifacts. Default: true
  "dependencies" : true // whether to move/copy the build's dependencies. Default: false.
  "scopes" : [ "compile", "runtime" ] // an array of dependency scopes to include when "dependencies" is true
  "properties": { // a list of properties to attach to the build's artifacts (regardless if "targetRepo" is used).
    "components": [ "c1", "c3", "c14" ],
    "release-name": [ "fb3-ga" ]
  }
  "failFast": true // fail and abort the operation upon receiving an error. Default: true
}

```

Produces: application/vnd.org.jfrog.artifactory.build.PromotionResult+json
Sample Output:

```
{
  "messages" : [
    {
      "level": "error",
      "message": "The repository has denied...."
    },...
  ]
}
```

Delete Builds

Description: Removes builds stored in Artifactory. Useful for cleaning up old build info data.
If the artifacts parameter is evaluated as 1 (0/false by default), build artifacts are also removed.
If the deleteAll parameter is evaluated as 1 (0/false by default), the whole build is removed.
Since: 2.3.0; artifact removal since 2.3.3;
Notes: Requires Artifactory Pro
Security: Requires an admin user
Usage: DELETE /api/build/{buildName}[?buildNumbers=n1[,n2]][&artifacts=0/1][&deleteAll=0/1]
Produces: application/text
Sample Usage:

```

DELETE /api/build/my-build?buildNumbers=51,52,55&artifacts=1

The following builds has been deleted successfully: 'my-build#51', 'my-build#52',
'my-build#55'.

DELETE /api/build/my-build

All 'my-build' builds have been deleted successfully.

```

Build Rename

Description: Renames a build stored in Artifactory. Typically used to keep the build info in sync with a renamed build on the CI server.

Since: 2.2.5

Notes: Requires Artifactory Pro

Security: Requires a valid user with deploy permissions

Usage: POST /api/build/rename/{buildName}?to=newBuildName

Produces: application/text

Sample Usage:

```

POST /api/build/rename/myJobName?to=myNewJobName

Build renaming of 'myJobName' to 'myNewJobName' was successfully started.

```

Push Build to Bintray

Description: Push a build to Bintray as a version.

Uses a descriptor file (that must have 'bintray-info' in its filename and a .json extension) that is included with the build artifacts. For more details, please refer to [Pushing a Build](#).

Signing a version is controlled by the `gpgSign` parameter in the descriptor file, and the `gpgSign` parameter passed to this command. **The value passed to this command always takes precedence over the value in the descriptor file.**

If you also want a passphrase to be applied to your signature, specify `gpgPassphrase=<passphrase>`.

You may omit the descriptor file by passing 6 override parameters (see below). If you wish to use the descriptor file you should pass an empty json string instead.

Since: 3.5.0

Security: Requires a valid user with deploy permissions and Bintray credentials defined (for more details, please refer to [Entering your Bintray credentials](#)).

Usage: POST /api/build/pushToBintray/{build.name}/{build.number}?gpgPassphrase=<passphrase>[&gpgSign=true\false]

Consumes: application/vnd.org.jfrog.artifactory.build.BintrayDescriptorOverrideParams+json

Sample Input:

```

POST /api/build/pushToBintray/testBuild/1?gpgPassphrase=password&gpgSign=true
{
    "subject": "myUser",
    "repoName": "test",
    "packageName": "overridePkg",
    "versionName": "overrideVer",
    "licenses": ["MIT"],
    "vcs_url": "https://github.com/bintray/bintray-client-java"
}

```

Produces: application/vnd.org.jfrog.artifactory.bintray.BintrayPushResponse+json

Sample Output:

```
{"Message": "Pushing build to Bintray finished successfully."}
```

ARTIFACTS & STORAGE

Folder Info

Description: Folder Info

For virtual use, the virtual repository returns the unified children. Supported by local, local-cached and virtual repositories.

Since: 2.2.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/storage/{repoKey}/{folder-path}

Produces: application/vnd.org.jfrog.artifactory.storage.FolderInfo+json

Sample Output:

```
GET /api/storage/libs-release-local/org/acme
{
  "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme",
  "repo": "libs-release-local",
  "path": "/org/acme",
  "created": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ),
  "createdBy": "userY",
  "lastModified": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ),
  "modifiedBy": "userX",
  "lastUpdated": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ),
  "children": [
    {
      "uri" : "/child1",
      "folder" : "true"
    },
    {
      "uri" : "/child2",
      "folder" : "false"
    }
  ]
}
```

File Info

Description: File Info

For virtual use the virtual repository returns the resolved file. Supported by local, local-cached and virtual repositories.

Since: 2.2.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/storage/{repoKey}/{filePath}

Produces: application/vnd.org.jfrog.artifactory.storage.FileInfo+json

Sample Output:

```

GET /api/storage/libs-release-local/org/acme/lib/ver/lib-ver.pom
{
  "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver/lib-ver.pom",
  "downloadUri": "http://localhost:8080/artifactory/libs-release-local/org/acme/lib/ver/lib-ver.pom",
  "repo": "libs-release-local",
  "path": "/org/acme/lib/ver/lib-ver.pom",
  "remoteUrl": "http://some-remote-repo/mvn/org/acme/lib/ver/lib-ver.pom",
  "created": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ),
  "createdBy": "userY",
  "lastModified": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ),
  "modifiedBy": "userX",
  "lastUpdated": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ),
  "size": "1024", //bytes
  "mimeType": "application/pom+xml",
  "checksums": {
    "md5" : string,
    "sha1" : string,
    "sha256" : string
  },
  "originalChecksums": {
    "md5" : string,
    "sha1" : string,
    "sha256" : string
  }
}

```

Get Storage Summary Info

Description: Returns storage summary information regarding binaries, file store and repositories.
Since: 4.2.0
Security: Requires a privileged user (Admin only)
Usage: GET /api/storageinfo
Produces: application/json
Sample Output:

```

GET /api/storageinfo
{
  "binariesSummary": {
    "binariesCount": "125,726",
    "binariesSize": "3.48 GB",
    "artifactsSize": "59.77 GB",
    "optimization": "5.82%",
    "itemsCount": "2,176,580",
    "artifactsCount": "2,084,408"
  },
  "fileStoreSummary": {

```

```
"storageType": "filesystem",
"storageDirectory": "/home/.../artifactory/devenv/.artifactory/repositories",
"totalSpace": "204.28 GB",
"usedSpace": "32.22 GB (15.77%)",
"freeSpace": "172.06 GB (84.23%)"
},
"repositoriesSummaryList": [
{
    "repoKey": "plugins-release",
    "repoType": "VIRTUAL",
    "foldersCount": 0,
    "filesCount": 0,
    "usedSpace": "0 bytes",
    "itemsCount": 0,
    "packageType": "Maven",
    "percentage": "0%"
},
{
    "repoKey": "repo",
    "repoType": "VIRTUAL",
    "foldersCount": 0,
    "filesCount": 0,
    "usedSpace": "0 bytes",
    "itemsCount": 0,
    "packageType": "Generic",
    "percentage": "0%"
},
...
{
    "repoKey": "TOTAL",
    "repoType": "NA",
    "foldersCount": 92172,
    "filesCount": 2084408,
    "usedSpace": "59.77 GB",
    "itemsCount": 2176580
}
]
```

Item Last Modified

Description: Retrieve the last modified item at the given path. If the given path is a folder, the latest last modified item is searched for recursively. Supported by local and local-cached repositories.

Since: 2.2.5

Notes: Requires Artifactory Pro

Security: Requires a valid user with deploy permissions

Usage: GET /api/storage/{repoKey}/{item-path}?lastModified

Produces: application/vnd.org.jfrog.artifactory.storage.ItemLastModified+json

Sample Output:

```
GET /api/storage/libs-release-local/org/acme?lastModified
{
  "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/foo/1.0-SNAPSHOT/foo-1.0-SNAPSHOT.pom",
  "lastModified": ISO8601
}
```

File Statistics

Description: Item statistics record the number of times an item was downloaded, last download date and last downloader. Supported by local and local-cached repositories.

Since: 3.1.0

Security: Requires read privileges

Usage: GET /api/storage/{repoKey}/{item-path}?stats

Produces: application/vnd.org.jfrog.storage.StatsInfo+json

Sample Output:

```
GET /api/storage/libs-release-local/org/acme/foo/1.0/foo-1.0.jar?stats
{
  "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/foo/1.0/foo-1.0.jar",
  "lastDownloaded": Timestamp (ms),
  "downloadCount": 1337,
  "lastDownloadedBy": "user1"
}
```

Item Properties

Description: Item Properties. Optionally return only the properties requested. Supported by local and local-cached repositories.

Since: 2.2.1

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/storage/{repoKey}/{itemPath}?properties[=x[,y]]

Produces: application/vnd.org.jfrog.artifactory.storage.ItemProperties+json

Sample Output:

```
GET /api/storage/libs-release-local/org/acme?properties\[=x[,y]\]
{
  "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme"
  "properties": {
    "p1": ["v1", "v2", "v3"],
    "p2": ["v4", "v5", "v6"]
  }
}
```

Set Item Properties

Description: Attach properties to an item (file or folder). When a folder is used property attachment is recursive by default. In order to supply special characters (comma (,), backslash (\), pipe(|), equals(=)) as key/value you must add a backslash (\) before them. For example: ..?properties=a=1\=1 will attach key a with 1=1 as value
Supported by local and local-cached repositories.

Notes: Requires Artifactory Pro

Since: 2.3.0

Security: Requires a privileged user (can be anonymous)

Usage: PUT /api/storage/{repoKey}{itemPath}?properties=p1=v1[,v2][|p2=v3][&recursive=1]

Sample Usage:

```
PUT
/api/storage/libs-release-local/ch/qos/logback/logback-classic/0.9.9?properties=os=win
,linux|qa=done&recursive=1
```

Delete Item Properties

Description: Deletes the specified properties from an item (file or folder). When a folder is used property removal is recursive by default. Supported by local and local-cached repositories.

Notes: Requires Artifactory Pro

Since: 2.3.2

Security: Requires a privileged user (can be anonymous)

Usage: DELETE /api/storage/{repoKey}{itemPath}?properties=p1[,p2][&recursive=1]

Sample Usage:

```
DELETE
/api/storage/libs-release-local/ch/qos/logback/logback-classic/0.9.9?properties=os,qa&
recursive=1
```

Set Item SHA256 Checksum

Description: Calculates an artifact's SHA256 checksum and attaches it as a property (with key "sha256"). If the artifact is a folder, then recursively calculates the SHA256 of each item in the folder and attaches the property to each item.

Since: 4.2.1

Security: Requires an admin user

Consumes: application/json

Usage: POST /api/checksum/sha256

Sample Usage:

```
POST /api/checksum/sha256
{
  "repoKey": "ext-snapshot-local",
  "path": "artifactory-powerpack-3.9.3/bin/"
}
```

Retrieve Artifact

Description: Retrieves an artifact from the specified destination.
You can also use [Property-based Resolution](#) as part of retrieving artifacts.
Security: Requires a user with 'read' permission (can be anonymous)
Usage: GET /repo-key/path/to/artifact.ext
Sample Usage:

```
GET
http://localhost:8080/artifactory/libs-release-local/ch/qos/logback/logback-classic/0.
9.9/logback-classic-0.9.9.jar
```

Retrieve Latest Artifact

Description: Retrieves the youngest (last uploaded) artifact version from the specified destination.
[Latest Maven Release/Integration](#): Specify SNAPSHOT or [RELEASE] for the version in the requested path to get the latest Maven integration or release artifact.
[Latest Non-Maven Release/Integration](#): Specify [INTEGRATION] or [RELEASE] for the version in the requested path (replacing the [folderItegRev] and [fileItegRev] as defined by the repository's [layout](#)) to get the latest integration version or latest release version artifact accordingly.
Integration and release tokens cannot be mixed together. Only local, cache and virtual repositories will be used.
You can also use [property-based resolution](#) as part of retrieving artifacts.
Notes: Requires Artifactory Pro.
Since: Latest Maven: 2.6.0; Latest non-Maven: 2.6.2
Security: Requires a user with 'read' permission (can be anonymous)
Usage: GET /repo-key/path/to/artifact.ext
Sample Usage:

Download the latest Maven unique snapshot artifact:

```
GET
http://localhost:8080/artifactory/libs-release-local/ch/qos/logback/logback-classic/0.
9.9-SNAPSHOT/logback-classic-0.9.9-SNAPSHOT.jar
```

Download the latest release artifact:

```
GET http://localhost:8080/artifactory/ivy-local/org/acme/[RELEASE]/acme-[RELEASE].jar
```

Download the latest integration artifact:

```
GET
http://localhost:8080/artifactory/ivy-local/org/acme/1.0-[INTEGRATION]/acme-1.0-[INTEGRATION].jar
```

Retrieve Build Artifacts Archive

Description: Retrieves an archive file (supports zip/tar/tar.gz/tgz) containing all the artifacts related to a specific build, you can optionally provide mappings to filter the results, the mappings support [regexp capturing groups](#) which enables you to dynamically construct the target path inside the result archive file.

Notes: Requires Artifactory Pro

Since: 2.6.5

Security: Requires a privileged user (can be anonymous)

Usage: POST /api/archive/buildArtifacts

Consumes: application/vnd.org.jfrog.artifactory.build.BuildArtifactsRequest+json

Produces: application/zip (for zip archive type), application/x-tar (for tar archive type), application/x-gzip (for tar.gz/tgz archive type)

Sample Usage:

```
POST /api/archive/buildArtifacts
{
  +"buildName": "build-name" // The build name for search by
  +"buildNumber": "15" // The build number to search by, can be LATEST to search for
  the latest build number
  -"buildStatus": "Released" // Optionally search by latest build status (e.g:
"Released")
  -"repos": ["libs-release-local,ext-release-local"] // Optionally refine search for
  specific repos, omit to search within all repositories
  +"archiveType": "tar/zip/tar.gz/tgz" // The archive file type to return
  -"mappings": [ // Optionally refine the search by providing a list of regexp patterns
  to search by
    {
      "input": "(.+)/(.+)-sources.jar",
      "output": "$1/sources/$2.jar" // Optionally provide different path of the found
  artifacts inside the result archive, supports regexp groups tokens
    },
    {
      "input": "(.+)-release.zip"
    }
  ]
}
```

Retrieve Folder or Repository Archive

Description: Retrieves an archive file (supports zip/tar/tar.gz/tgz) containing all the artifacts that reside under the specified path (folder or repository root). Requires [Enable Folder Download](#) to be set.

Notes: Requires Artifactory Pro

Since: 4.1.0

Security: Requires a privileged user with read permissions on the path.

Usage: GET /api/archive/download/{repoKey}/{path}?archiveType={archiveType}

Produces: */*

Sample Usage:

```
GET /api/archive/download/my-local-repo/folder?archiveType=zip
{Stream containing contents of path my-local-repo/folder}

GET /api/archive/download/my-local-repo?archiveType=zip
{Stream containing contents of repo my-local-repo}
```

Trace Artifact Retrieval

Description: Simulates an artifact retrieval request from the specified location and returns verbose output about the resolution process.

This API is useful for debugging artifact retrieval issues.
Security: As applied to standard artifact retrieval by the requesting user.
Since: 2.6.0
Usage: GET /repo-key/path/to/artifact.ext?trace
Produces: text/plain
Sample Output:

```
GET
http://localhost:8080/artifactory/libs-release-local/jmock/jmock/1.0.1/jmock-1.0.1.jar
?trace

Request ID: 51c808f6
Repo Path ID: libs-release-local:jmock/jmock/1.0.1/jmock-1.0.1.jar
Method Name: GET
User: resolver
Time: 2012-05-06T10:49:09.528+03:00
Thread: pool-1-thread-31
Steps:
2012-05-06T10:49:09.587+03:00 Received request
2012-05-06T10:49:09.588+03:00 Request source = 0:0:0:0:0:0:1, Last modified =
01-01-70 01:59:59 IST, If modified since = -1, Thread name = pool-1-thread-31
2012-05-06T10:49:09.697+03:00 Retrieving info
2012-05-06T10:49:09.723+03:00 Identified resource as a file
...
2012-05-06T10:49:09.788+03:00 Responding with selected content handle
2012-05-06T10:49:09.807+03:00 Request succeeded
```

Archive Entry Download

Description: Retrieves an archived resource from the specified archive destination.
Security: Requires a user with 'read' permission (can be anonymous)
Usage: GET /repo-key/path/to/artifact.jar!*/path/to/archived/resource (**NOTE!** the '!' between the archive file name and the archive entry path)
Sample Output:

```
GET
http://localhost:8080/artifactory/repol-cache/commons-lang/commons-lang/2.6/commons-la
ng-2.6.jar!/META-INF/LICENSE.txt
```

Create Directory

Description: Create new directory at the specified destination.
Notes: You can also attach properties as part of creating directories.
Security: Requires a user with 'deploy' permissions (can be anonymous)
Usage: PUT /repo-key/path/to/directory/
Produces: application/vnd.org.jfrog.artifactory.storage.ItemCreated+json
Sample Usage:

```

PUT /libs-release-local/path/to/directory/
{
  "uri": "http://localhost:8080/artifactory/libs-release-local/path/to/directory",
  "repo": "libs-release-local",
  "path": "/path/to/directory",
  "created": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ),
  "createdBy": "userY",
  "children" : [ ]
}

```

Deploy Artifact

Description: Deploy an artifact to the specified destination.

Notes: You can also attach properties as part of deploying artifacts.

Security: Requires a user with 'deploy' permissions (can be anonymous)

Usage: PUT /repo-key/path/to/artifact.ext

Produces: application/vnd.org.jfrog.artifactory.storage.ItemCreated+json

Sample Usage:

```

PUT /libs-release-local/my/jar/1.0/jar-1.0.jar
{
  "uri": "http://localhost:8080/artifactory/libs-release-local/my/jar/1.0/jar-1.0.jar",
  "downloadUri":
    "http://localhost:8080/artifactory/libs-release-local/my/jar/1.0/jar-1.0.jar",
  "repo": "libs-release-local",
  "path": "/my/jar/1.0/jar-1.0.jar",
  "created": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ),
  "createdBy": "userY",
  "size": "1024", //bytes
  "mimeType": "application/java-archive",
  "checksums":
  {
    "md5" : string,
    "sha1" : string
  },
  "originalChecksums":{
    "md5" : string,
    "sha1" : string
  }
}

```

Deploy Artifact by Checksum

Description: Deploy an artifact to the specified destination by checking if the artifact content already exists in Artifactory.

If Artifactory already contains a user readable artifact with the same checksum the artifact content is copied over to the new location and return a response without requiring content transfer.

Otherwise, a 404 error is returned to indicate that content upload is expected in order to deploy the artifact.

Notes: You can also attach properties as part of deploying artifacts.

Security: Requires a user with 'deploy' permissions (can be anonymous)

Usage: PUT /repo-key/path/to/artifact.ext

Headers: X-Checksum-Deploy: true, X-Checksum-Sha1: sha1Value, X-Checksum-Sha256: sha256Value

Produces: application/vnd.org.jfrog.artifactory.storage.ItemCreated+json

Since: 2.5.1

Sample Output:

```

PUT /libs-release-local/my/jar/1.0/jar-1.0.jar
{
  "uri": "http://localhost:8080/artifactory/libs-release-local/my/jar/1.0/jar-1.0.jar",
  "downloadUri":
    "http://localhost:8080/artifactory/libs-release-local/my/jar/1.0/jar-1.0.jar",
  "repo": "libs-release-local",
  "path": "/my/jar/1.0/jar-1.0.jar",
  "created": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ),
  "createdBy": "userY",
  "size": 1024, //bytes
  "mimeType": "application/java-archive",
  "checksums":
  {
    "md5" : string,
    "sha1" : string
  },
  "originalChecksums":{
    "md5" : string,
    "sha1" : string
  }
}

```

Deploy Artifacts from Archive

Description: Deploys an archive containing multiple artifacts and extracts it at the specified destination maintaining the archive's file structure. Deployment is performed in a single HTTP request and only the extracted content is deployed, not the archive file itself. Supported archive types are: zip; tar; tar.gz; and tgz. **NOTE!** that deployment of compressed archives (unlike tar) may incur considerable CPU overhead.

Notes: Requires Artifactory Pro

Security: Requires a user with 'deploy' permissions (can be anonymous)

Usage: PUT path1/to/repo-key/ /path2/to/archive.zip

Headers: X-Explode-Archive: true

Produces: text/plain

Since: 2.6.3

Sample Usage:

```
PUT /libs-release-local/ /Users/user/Desktop/archive.zip
```

Push a Set of Artifacts to Bintray

Description: Push a set of artifacts to Bintray as a version.

Uses a descriptor file (that must have 'bintray-info' in its filename and a .json extension) that was deployed to artifactory, the call accepts the full path to the descriptor as a parameter.

For more details, please refer to [Pushing a Set of Files](#).

Signing a version is controlled by the *gpgSign* parameter in the descriptor file, and the *gpgSign* parameter passed to this command. **The value passed to this command always takes precedence over the value in the descriptor file.**

If you also want a passphrase to be applied to your signature, specify *gpgPassphrase=<passphrase>*.

Security: Requires a valid user with deploy permissions and Bintray credentials defined (for more details, please refer to [Entering your Bintray credentials](#)).

Usage: POST /api/bintray/push/?descriptor=pathToDescriptorFile[&gpgPassphrase=passphrase][&gpgSign=true\false]

Since: 3.5.0

Produces: application/vnd.org.jfrog.artifactory.bintray.BintrayPushResponse+json

Sample Output:

```
{"Message": "Pushing build to Bintray finished successfully."}
```

Push Docker Tag to Bintray

Description: Push Docker tag to Bintray

Calculation can be synchronous (the default) or asynchronous.

Security: Requires a valid user with deploy permissions and Bintray credentials defined (for more details, please refer to [Entering your Bintray credentials](#)).

Usage: POST /api/bintray/docker/push/{repoKey}

Since: 3.6.0

Produces: text/plain

Sample Output:

```
POST api/bintray/docker/push/docker-local
{
  "dockerImage": "jfrog/ubuntu:latest", // The docker image to push, use ':' for
  // specific tag or leave blank for 'latest'
  "bintraySubject": "shayy", // The Bintray Subject
  "bintrayRepo": "containers", // The Bintray Subject's repository
  "async": false // Optionally execute the push asynchronously. Default: false
}
```

File Compliance Info

Description: Get compliance info for a given artifact path. The result includes license and vulnerabilities, if any. Supported by local and local-cached repositories.

Notes: Requires Artifactory Pro, requires Black Duck addon enabled.

Since: 3.0.0

Security: Requires an authenticated user.

Usage: GET: /api/compliance/{repoKey}/{item-path}

Produces: application/json

Sample output:

```
GET:
/api/compliance/libs-release-local/ch/qos/logback/logback-classic/0.9.9/logback-classic-0.9.9.jar
{
  "licenses" : [ {"name":"LGPL v3", "url": "http://"}, {"name":"APL v2", "url":
  "http://"}... ],
  "vulnerabilities" : [ {"name":"CVE-13427", "url": "http://"}, {"name":"CVE-1041",
  "url": "http://"}... ]
}
```

Delete Item

Description: Deletes a file or a folder from the specified destination.

Security: Requires a user with 'delete' permission (can be anonymous)

Usage: DELETE /repo-key/path/to/file-or-folder

Sample Usage:

```
DELETE  
http://localhost:8080/artifactory/libs-release-local/ch/qos/logback/logback-classic/0.  
9.9
```

Copy Item

Description: Copy an artifact or a folder to the specified destination. Supported by local repositories only.

Optionally suppress cross-layout module path translation during copy.

You can test the copy using a dry run.

Copy item behaves similarly to a standard file system and supports renames. If the target path does not exist, the source item is copied and optionally renamed. Otherwise, if the target exists and it is a directory, the source is copied and placed under the target directory.

Notes: Requires Artifactory Pro

Security: Requires a privileged user (can be anonymous)

Usage: POST /api/copy/{srcRepoKey}/{srcFilePath}?to=/targetRepoKey/{targetFilePath}[&dry=1][&suppressLayouts=0/1][&failFast=0/1]

Produces: application/vnd.org.jfrog.artifactory.storage.CopyOrMoveResult+json

Since: 2.2.2

Sample Output:

```
POST /api/copy/libs-release-local/org/acme?to=/ext-releases-local/org/acme-new&dry=1  
{  
  "messages" : [  
    {  
      "level": "error",  
      "message": "The repository has denied...."  
    },...  
  ]  
}
```

Move Item

Description: Moves an artifact or a folder to the specified destination. Supported by local repositories only.

Optionally suppress cross-layout module path translation during move.

You can test the move using dry run.

Move item behaves similarly to a standard file system and supports renames. If the target path does not exist, the source item is moved and optionally renamed. Otherwise, if the target exists and it is a directory, the source is moved and placed under the target directory.

Notes: Requires Artifactory Pro

Security: Requires a privileged user (can be anonymous)

Usage: POST /api/move/{srcRepoKey}/{srcFilePath}?to=/targetRepoKey/{targetFilePath}[&dry=1][&suppressLayouts=0/1][&failFast=0/1]

Produces: application/vnd.org.jfrog.artifactory.storage.CopyOrMoveResult+json

Since: 2.2.2

Sample Output:

```
POST /api/move/libs-release-local/org/acme?to=/ext-releases-local/org/acme-new&dry=1  
{  
  "messages" : [  
    {  
      "level": "error",  
      "message": "The repository has denied...."  
    },...  
  ]  
}
```

Get Repository Replication Configuration

Description: Returns the replication configuration for the given repository key, if found. Supported by local and remote repositories.

Notes: Requires Artifactory Pro

Security: Requires a privileged user

Usage: GET /api/replications/{repoKey}

Produces: application/vnd.org.jfrog.artifactory.replications.ReplicationConfigRequest+json

Since: 3.1.1

Sample Usage:

```
GET /api/replications/libs-release-local
{
  "url" : "http://localhost:8081/artifactory/remote-repo",
  "socketTimeoutMillis" : 15000,
  "username" : "admin",
  "password" : "password",
  "enableEventReplication" : false,
  "enabled" : true,
  "cronExp" : "0 0 12 * * ?",
  "syncDeletes" : true,
  "syncProperties" : true,
  "repoKey" : "libs-release-local"
}
```

Set Repository Replication Configuration

Description: Add or replace replication configuration for given repository key. Supported by local and remote repositories.

Notes: Requires Artifactory Pro

Security: Requires a privileged user

Usage: PUT /api/replications/{repoKey}

Consumes: application/vnd.org.jfrog.artifactory.replications.ReplicationConfigRequest+json

Since: 3.1.1

Sample Usage:

```
PUT /api/replications/libs-release-local
```

Update Repository Replication Configuration

Description: Update existing replication configuration for given repository key, if found. Supported by local and remote repositories.

Notes: Requires Artifactory Pro

Security: Requires a privileged user

Usage: POST /api/replications/{repoKey}

Consumes: full or partial application/vnd.org.jfrog.artifactory.replications.ReplicationConfigRequest+json

Since: 3.1.1

Sample Usage:

```
POST /api/replications/libs-release-local
```

Delete Repository Replication Configuration

Description: Delete existing replication configuration for given repository key. Supported by local and local-cached repositories.

Notes: Requires Artifactory Pro

Security: Requires a privileged user

Usage: DELETE /api/replications/{repoKey}

Since: 3.1.1

Sample Usage:

```
DELETE /api/replications/libs-release-local
```

Scheduled Replication Status

Description: Returns the status of scheduled cron-based replication jobs define via the Artifactory UI on repositories. Supported by local, local-cached and remote repositories.

Notes: Requires Artifactory Pro

Security: Requires a user with 'read' permission (can be anonymous)

Usage: GET /api/replication/{repoKey}

Produces: application/vnd.org.jfrog.artifactory.replication.ReplicationStatus+json

Since: 2.4.2

Sample Usage:

```
GET /api/replication/remote-libs
{
  "status": never_run|incomplete(running or interrupted)|error|warn|ok|inconsistent
  "lastCompleted": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ) or null if never completed
}
```

Pull/Push Replication

Description: Schedules immediate content replication between two Artifactory instances. Replication can include properties and can optionally delete local items if they do not exist in the source repository.

This API completes the exiting [cron-based](#) replication exposed via the Artifactory UI and allows for on-demand execution.

Pull Replication - pulls content from a remote Artifactory repository to a local cache of the remote repository.

Push Replication - pushes content from a local repository into a remote Artifactory local repository.

Supported by local, local-cached and remote repositories.

Notes: Requires Artifactory Pro

Security: Requires a privileged user (can be anonymous) For non-admin users will replicate at max the number of files as defined by the `artifactory.search.userQueryLimit` system property.

Usage: POST /api/replication/{srcRepoKey}/{srcPath}

Consumes: application/vnd.org.jfrog.artifactory.replication.ReplicationRequest+json

Since: 2.4.0

Sample Usage:

```
POST /api/replication/libs-release-local/com/acme
{
  - "properties": true, //Sync item properties (true by default)
  - "delete": true, //Sync deletions (false by default)
  //The following is only applicable for push replication
  + "url" : "https://repo.nmiy.org/repo-key", // The remote repository URL
  + "username": "replicator", //The name of a user with deploy permissions on the
  remote repository
  + "password": "****", //The remote repository password
  - "proxy": "org-prox", //A name of an Artifactory-configured proxy to use for remote
  requests
}
```

+ = mandatory; - = optional

Create or Replace Local Multi-push Replication

Description:Creates or replaces a local multi-push replication configuration. Supported by local and local-cached repositories.

Notes: Requires an enterprise license

Security: Requires an admin user.

Usage: PUT /api/replications/multiple

Consumes: application/vnd.org.jfrog.artifactory.replications.MultipleReplicationConfigRequest+json

Since: 3.7

Sample Usage:

```
PUT /api/replications/multiple/libs-release-local
{
  "cronExp": "0 0/9 14 * * ?",
  "enableEventReplication": true,
  "replications": [
    {
      "+ url": "http://localhost:8081/artifactory/repo-k",
      "+ socketTimeoutMillis": 15000,
      "+ username": "admin",
      "+ password": "password",
      "- enableEventReplication": true,
      - "enabled": true,
      - "syncDeletes": false,
      - "syncProperties": true,
      - "repoKey": "libs-release-local"
    },
    {
      "+ url": "http://localhost:8081/artifactory/repo-v",
      "+ socketTimeoutMillis": 15000,
      "+ username": "admin",
      "+ password": "password",
      - "enableEventReplication": true,
      - "enabled": true,
      - "syncDeletes": false,
      - "syncProperties": true,
      - "repoKey": "libs-release-local"
    }
  ]
}
```

+ =mandatory; - =optional

Update Local Multi-push Replication

Description:Updates a local multi-push replication configuration. Supported by local and local-cached repositories.

Notes: Requires an enterprise license

Security: Requires an admin user.

Usage: POST /api/replications/multiple

Consumes: application/vnd.org.jfrog.artifactory.replications.MultipleReplicationConfigRequest+json

Since: 3.7

Sample Usage:

```

POST /api/replications/multiple/libs-release-local
{
  "cronExp": "0 0/9 14 * * ?",
  "enableEventReplication": true,
  "replications": [
    {
      "url": "http://localhost:8081/artifactory/repo-k",
      "socketTimeoutMillis": 15000,
      "username": "admin",
      "password": "password",
      "enableEventReplication": true,
      "enabled": true,
      "syncDeletes": false,
      "syncProperties": true,
      "repoKey": "libs-release-local"
    },
    {
      "url": "http://localhost:8081/artifactory/repo-v",
      "socketTimeoutMillis": 15000,
      "username": "admin",
      "password": "password",
      "enableEventReplication": true,
      "enabled": true,
      "syncDeletes": false,
      "syncProperties": true,
      "repoKey": "libs-release-local"
    }
  ]
}

```

+mandatory; -=optional

Delete Local Multi-push Replication

Description: Deletes a local multi-push replication configuration. Supported by local and local-cached repositories.

Notes: Requires an enterprise license

Security: Requires an admin user.

Usage: DELETE /api/replications/{repoKey}?url={replicatedURL}

Produces: application/vnd.org.jfrog.artifactory.replications.ReplicationConfigRequest+json, application/vnd.org.jfrog.artifactory.replications.MultipleReplicationConfigRequest+json

Since: 3.7

Sample Usage:

```

DELETE
/api/replications/libs-release-local?url=http://10.0.0.1/artifactory/libs-release-local

```

Artifact Sync Download

Description: Downloads an artifact with or without returning the actual content to the client. When tracking the progress marks are printed (by default every 1024 bytes). This is extremely useful if you want to trigger downloads on a remote Artifactory server,

for example to force eager cache population of large artifacts, but want to avoid the bandwidth consumption involved in transferring the artifacts to the triggering client. If no content parameter is specified the file content is downloaded to the client.

Notes: This API requires Artifactory Pro.

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/download/{repoKey}/{filePath}[?content=none/progress][&mark=numOfBytesToPrintANewProgressMark]

Produces: application/octet-stream, text/plain (depending on content type)

Since: 2.2.2

Sample Output:

```
GET /api/download/my-remote/org/acme/1.0/acme-1.0.jar?content=progress&mark=512
.....
.....
.....
Completed: 150/340 bytes
```

Folder Sync (Deprecated)

Description: Triggers a no-content download of artifacts from a remote Artifactory repository for all artifacts under the specified remote folder. Can optionally delete local files if they do not exist in the remote folder,

overwrite local files only if they are older than remote files or never overwrite local files.

The default is not to delete any local files and to overwrite older local files with remote ones. By default progress marks of the sync are displayed. The default timeout for the remote file list is 15000 milliseconds (15 seconds).

Notes: This API is **deprecated**. Requires Artifactory Pro

Security: Requires a privileged user (can be anonymous) For non-admin users will replicate at max the number of files as defined by the `artifactory.search.userQueryLimit` system property.

Usage: GET

/api-sync/{remoteRepositoryKey}/{folderPath}[?progress=showProgress][&mark=numOfBytesToPrintANewProgressMark][&delete=deleteExistingFiles][&overwrite=never/force][&timeout=fileListTimeoutInMillis]

Produces: text/plain

Since: 2.2.4

Sample Output:

```
GET /api-sync/my-remote/org/acme/1.0?progress=1&delete=1
.....
.....
.....
.....
Completed: 970/1702 bytes
.....
.....
Completed: 1702/1702 bytes
Completed with 0 errors and 2 warnings (please check the server log for more details).
```

File List

Description: Get a flat (the default) or deep listing of the files and folders (not included by default) within a folder.

For deep listing you can specify an optional depth to limit the results.

Optionally include a map of metadata timestamp values as part of the result (only properties are displayed in since 3.0.0).
folder inclusion since 2.3.2; checksum inclusion since: 2.3.3; include folder root path since: 2.5.2. Supported by all types of repositories.

Since: 2.2.4

Notes: Requires Artifactory Pro

Security: Requires a non-anonymous privileged user.

Usage: GET /api/storage/{repoKey}/{folder-path}?list[&deep=0/1][&depth=n][&listFolders=0/1][&mdTimestamps=0/1][&includeRootPath=0/1]

Produces: application/vnd.org.jfrog.artifactory.storage.FileList+json

Sample Output:

```
GET /api/storage/libs-release-local/org/acme?list&deep=1&listFolders=1&mdTimestamps=1
{
  "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme",
  "created": ISO8601,
  "files" : [
    {
      "uri": "/archived",
      "size": "-1",
      "lastModified": ISO8601,
      "folder": "true"
    },
    {
      "uri": "/doc.txt",
      "size": "253207", //bytes
      "lastModified": ISO8601,
      "folder": "false",
      "sh1": sha1Checksum,
      "mdTimestamps": { "properties" : lastModified (ISO8601) }
    },
    {
      "uri": "/archived/doc1.txt",
      "size": "253100", //bytes
      "lastModified": ISO8601,
      "folder": "false",
      "sh1": sha1Checksum,
      "mdTimestamps": { "properties" : lastModified (ISO8601) }
    },...
  ]
}
```

SEARCHES

All searches return limited results for anonymous users (same limits as in the user interface).

Artifactory Query Language (AQL)

Description: Flexible and high performance search using [Artifactory Query Language \(AQL\)](#).

Since: 3.5.0

Security: Requires an authenticated user

Usage: POST /api/search/aql

Consumes: text/plain

Sample Usage:

```
POST /api/search/aql
items.find(
{
  "repo": {"$eq": "jcenter"}
}
)
```

Produces: application/json

Sample Output:

```
{
  "results" : [
    {
      "repo" : "libs-release-local",
      "path" : "org/jfrog/artifactory",
      "name" : "artifactory.war",
      "type" : "item type",
      "size" : "75500000",
      "created" : "2015-01-01T10:10;10",
      "created_by" : "Jfrog",
      "modified" : "2015-01-01T10:10;10",
      "modified_by" : "Jfrog",
      "updated" : "2015-01-01T10:10;10"
    }
  ],
  "range" : {
    "start_pos" : 0,
    "end_pos" : 1,
    "total" : 1
  }
}
```

Artifact Search (Quick Search)

Description: Artifact search by part of file name.

Searches return file info URIs. Can limit search to specific repositories (local or caches).

Since: 2.2.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/search/artifact?name=name[&repos=x[y]]

Headers (Optional): X-Result-Detail: info (To add all extra information of the found artifact), X-Result-Detail: properties (to get the properties of the found artifact), X-Result-Detail: info, properties (for both).

Produces: application/vnd.org.jfrog.artifactory.search.ArtifactSearchResult+json

Sample Output:

```

GET /api/search/artifact?name=lib&repos=libs-release-local
{
"results": [
{
    "uri":
"http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver/lib
-ver.pom"
}, {
    "uri":
"http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver2/li
b-ver2.pom"
}
]
}

```

Archive Entry Search (Class Search)

Description: Search archive entries for classes or any other jar resources.

Can limit search to specific repositories (local or caches).

Since: 2.2.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/search/archive?name=[archiveEntryName][&repos=x[y]]

Produces: application/vnd.org.jfrog.artifactory.search.ArchiveEntrySearchResult+json

Sample Output:

```

GET
/api/search/archive?name=*Logger.class&repos=third-party-releases-local,repol-cache
{
"results" :[
{
    "entry":
"org/apache/jackrabbit/core/query/lucene/AbstractIndex.LoggingPrintStream.class",
    "archiveUris": [

"http://localhost:8080/artifactory/api/storage/third-party-releases-local/org/apache/j
ackrabbit/
                jackrabbit-core/1.2.3/jackrabbit-core-1.2.3.jar",

"http://localhost:8080/artifactory/api/storage/third-party-releases-local/org/apache/j
ackrabbit/
                jackrabbit-core/1.3.1/jackrabbit-core-1.3.1.jar"
]
}, {
    "entry": "org/codehaus/plexus/logging/AbstractLogger.class",
    "archiveUris": [

"http://localhost:8080/artifactory/api/storage/repol-cache/org/codehaus/plexus/plexus-
container-default/
                1.0-alpha-9-stable-1/plexus-container-default-1.0-alpha-9-stable-1.jar"
]
}
]
}

```

GAVC Search

Description: Search by Maven coordinates: GroupId, ArtifactId, Version & Classifier.

Search must contain at least one argument. Can limit search to specific repositories (local or caches).

Since: 2.2.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/search/gavc?[g=groupId][&a=artifactId][&v=version][&c=classifier][&repos=x[y]]

Headers (Optional): X-Result-Detail: info (To add all extra information of the found artifact), X-Result-Detail: properties (to get the properties of the found artifact), X-Result-Detail: info, properties (for both).

Produces: application/vnd.org.jfrog.artifactory.search.GavcSearchResult+json

Sample Output:

```
GET /api/search/gavc?g=org.acme&a=artifact&v=1.0&c=sources&repos=libs-release-local
{
  "results": [
    {
      "uri":
      "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/artifact/1.0/artifact-1.0-sources.jar"
    },
    {
      "uri":
      "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/artifactB/1.0/artifactB-1.0-sources.jar"
    }
  ]
}
```

Property Search

Description: Search by properties.

If no value is specified for a property - assume '*'. Can limit search to specific repositories (local or caches).

Since: 2.2.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/search/prop?[p1=v1,v2][&p2=v3][&repos=x[y]]

Headers (Optional): X-Result-Detail: info (To add all extra information of the found artifact), X-Result-Detail: properties (to get the properties of the found artifact), X-Result-Detail: info, properties (for both).

Produces: application/vnd.org.jfrog.artifactory.search.MetadataSearchResult+json

Sample Output:

```
GET /api/search/prop?p1=v1,v2&p2=v3&repos=libs-release-local
{
  "results" : [
    {
      "uri":
      "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver/lib-ver.pom"
    },
    {
      "uri":
      "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver2/lib-ver2.pom"
    }
  ]
}
```

Checksum Search

Description: Artifact search by checksum (md5, sha1, or sha256)

Searches return file info URIs. Can limit search to specific repositories (local or caches).

Notes: Requires Artifactory Pro

Since: 2.3.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/search/checksum?md5=md5sum?sha1=sha1sum?sha256=sha256sum[&repos=x[,y]]

Headers (Optional): X-Result-Detail: info (To add all extra information of the found artifact), X-Result-Detail: properties (to get the properties of the found artifact), X-Result-Detail: info, properties (for both).

Produces: application/vnd.org.jfrog.artifactory.search.ChecksumSearchResult+json

Sample Output:

```
GET  
/api/search/checksum?sha256=9a7fb65f15e00aa2a22c1917d0dafd4374fee8daf0966a4d94cd37a0b9  
acafb9&repos=libs-release-local  
{  
  "results": [  
    {  
      "uri":  
        "http://localhost:8080/artifactory/api/storage/libs-release-local/org/jfrog/build-info  
-api/1.3.1/build-info-api-1.3.1.jar"  
    }  
  ]  
}
```

Bad Checksum Search

Description: Find all artifacts that have a bad or missing client checksum values (md5 or sha1)

Searches return file info uris. Can limit search to specific repositories (local, caches or virtuals).

Notes: Requires Artifactory Pro

Since: 2.3.4

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/search/badChecksum?type=md5|sha1[&repos=x[,y]]

Produces: application/vnd.org.jfrog.artifactory.search.BadChecksumSearchResult+json

Sample Output:

```
GET /api/search/badChecksum?type=md5&repos=libs-release-local  
{  
  "results": [  
    {  
      "uri":  
        "http://localhost:8080/artifactory/api/storage/libs-release-local/org/jfrog/build-info  
-api/1.3.1/build-info-api-1.3.1.jar"  
      "serverMd5": "4040c7c184620af0a0a8a3682a75eb7"  
      "clientMd5": "4040c7c184620af0a0a8a3682a75e44" //On missing checksum this  
element will be an empty string  
    }  
  ]  
}
```

Artifacts Not Downloaded Since

Description: Retrieve all artifacts not downloaded since the specified Java epoch in **milliseconds**.

Optionally include only artifacts created before the specified `createdBefore` date, otherwise only artifacts created before `notUsedSince` are returned.

Can limit search to specific repositories (local or caches).

Since: 2.2.4

Security: Requires a privileged non-anonymous user.

Usage: GET /api/search/usage?notUsedSince=javaEpochMillis[&createdBefore=javaEpochMillis][&repos=x[,y]]

Produces: application/vnd.org.jfrog.artifactory.search.ArtifactUsageResult+json

Sample Output:

```
GET /api/search/usage?notUsedSince=long&createdBefore=long&repos=libs-release-local
{
  "results" : [
    {
      "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver/lib-ver.jar",
      "lastDownloaded": ISO8601
    },{
      "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver2/lib-ver2.jar",
      "lastDownloaded": ISO8601
    }
  ]
}
```

Artifacts With Date in Date Range

Description: Get all artifacts with specified dates within the given range. Search can be limited to specific repositories (local or caches).

Since: 3.2.1

Security: Requires a privileged non-anonymous user.

Usage: GET /api/search/dates?[from=fromVal][&to=toVal][&repos=x[,y]][&dateFields=c[,d]]

Parameters: The `from` and `to` parameters can be either a long value for the java epoch (**milliseconds** since the epoch), or an ISO8601 string value. `from` is mandatory. If `to` is not provided, `now()` will be used instead, and if either are omitted, `400 bad request` is returned.

The `dateFields` parameter is a comma separated list of date fields that specify which fields the `from` and `to` values should be applied to. The date fields supported are: `created`, `lastModified`, `lastDownloaded`. It is a mandatory field and it also dictates which fields will be added to the JSON returned.

If ANY of the specified date fields of an artifact is within the specified range, the artifact will be returned.

Produces: application/vnd.org.jfrog.artifactory.search.ArtifactResult+json

Sample Output:

```

GET
/api/search/dates?dateFields=created,lastModified,lastDownloaded&from=long&to=long&repos=libs-release-local
{
"results" : [
  {
    "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver/lib-ver.jar",
    "created": ISO8601,
    "lastModified": ISO8601,
    "lastDownloaded": ISO8601
  },
  {
    "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver2/lib-ver2.jar",
    "created": ISO8601,
    "lastModified": ISO8601,
    "lastDownloaded": ISO8601
  }
]
}

```

Artifacts Created in Date Range

Description: Get All Artifacts Created in Date Range

If 'to' is not specified use now(). Can limit search to specific repositories (local or caches).

Since: 2.2.0

Security: Requires a privileged non-anonymous user.

Usage: GET /api/search/creation?from=javaEpochMillis[&to=javaEpochMillis][&repos=x[,y]]

Produces: application/vnd.org.jfrog.artifactory.search.ArtifactCreationResult+json

Sample Output:

```

GET /api/search/creation?from=long&to=long&repos=libs-release-local
{
"results" : [
  {
    "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver/lib-ver.jar",
    "created": ISO8601
  },
  {
    "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver2/lib-ver2.jar",
    "created": ISO8601
  }
]
}

```

Pattern Search

Description: Get all artifacts matching the given Ant path pattern

Since: 2.2.4

Notes: Requires Artifactory Pro. Pattern "***" is not supported to avoid overloading search results.

Security: Requires a privileged non-anonymous user.

Usage: GET /api/search/pattern?pattern=repo-key:this/is/a/*pattern*.war

Produces: application/vnd.org.jfrog.artifactory.search.PatternResultFileSet+json

Sample Output:

```
GET /api/search/pattern?pattern=libs-release-local:killer/*/ninja/**.jar
{
    "repositoryUri" : "http://localhost:8080/artifactory/libs-release-local",
    "sourcePattern" : "libs-release-local:killer/*/ninja/**.jar",
    "files" : [
        "killer/coding/ninja/1.0/monkey-1.0.jar",
        "killer/salty/ninja/1.5-SNAPSHOT/pickle-1.5-SNAPSHOT.jar"
    ]
}
```

Builds for Dependency

Description: Find all the builds an artifact is a dependency of (where the artifact is included in the build-info dependencies)

Notes: Requires Artifactory Pro

Since: 2.3.4

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/search/dependency?sha1=sha1Checksum

Produces: application/vnd.org.jfrog.artifactory.search.DependencyBuilds+json

Sample Output:

```
GET /api/search/dependency?sha1=451a3c5f8cfa44c5d805379e760b5c512c7d250b
{
    "results" : [
        {
            "uri": "http://localhost:8080/artifactory/api/build/my-build/50"
        },
        {
            "uri": "http://localhost:8080/artifactory/api/build/my-build/51"
        }
    ]
}
```

License Search

Description: Search for artifacts with specified statuses.

To search by specific license values use Property Search with the 'artifactory.licenses' property.

Default parameter values when unspecified: unapproved=1, unknown=1, notfound=0, neutral=0, approved=0, autofind=0.

When the autofind parameter is specified Artifactory will try to automatically find new license information and return it as part of the result in the found field. Please note that this can affect the speed of the search quite dramatically.

Can limit search to specific repositories (local or caches).

Since: 2.3.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: GET /api/search/license[?unapproved=1][&unknown=1][¬found=0][&neutral=0][&approved=0][&autofind=0][&repos=x,y]]

Produces: application/vnd.org.jfrog.artifactory.search.LicenseResult+json

Sample Output:

```

GET
/api/search/license?approved=1&unknown=1&autofind=1&repos=libs-release-local,staging
{
"results" : [
  {
    "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver/lib-ver.jar",
    "license": "lgplv2",
    "found": "lgplv2",
    "status": "approved"
  },
  {
    "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme/lib/ver/lib-ver.jar",
    "license": "cddlrv1",
    "found": "gplv3",
    "status": "neutral"
  },
  {
    "uri": "http://localhost:8080/artifactory/api/storage/staging/org/acme/lib/ver2/lib-ver2.jar",
    "license": "gplv3",
    "found": "gplv3",
    "status": "unapproved"
  }
]
}

```

Artifact Version Search

Description: Search for all available artifact versions by GroupId and ArtifactId in local, remote or virtual repositories.

Search can be limited to specific repositories (local, remote and virtual) by settings the `repos` parameter.

Release/integration versions: Unless the `version` parameter is specified, both release and integration versions are returned. When `version` is specified, e.g. `1.0-SNAPSHOT`, result includes only integration versions.

Integration versions are determined by the [repository layout](#) of the repositories searched. For integration search to work the repository layout requires an 'Artifact Path Pattern' that contains the `baseRev` token and then the `fileItegRev` token with only literals between them.

Remote searches: By default only local and cache repositories are used. When specifying `remote=1`, Artifactory searches for versions on remote repositories. **NOTE!** that this can dramatically slow down the search.

For Maven repositories the remote `maven-metadata.xml` is consulted. For non-maven layouts, remote file listing runs for all remote repositories that have the 'List Remote Folder Items' checkbox enabled.

Filtering results (Artifactory 3.0.2+): The `version` parameter can accept the `*` and/or `?` wildcards which will then filter the final result to match only those who match the given version pattern.

Since: 2.6.0

Notes: Requires Artifactory Pro

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/search/versions?[g=groupId][&a=artifactId][&v=version][&remote=0/1][&repos=x,y]]

Produces: application/vnd.org.jfrog.artifactory.search.ArtifactVersionsResult+json

Sample Output:

```

GET /api/search/versions?g=org.acme&a=artifact&repos=libs-release-local
{
  "results": [
    {
      "version": "1.2",
      "integration": false
    },
    {
      "version": "1.0-SNAPSHOT",
      "integration": true
    },
    {
      "version": "1.0",
      "integration": false
    }
  ]
}

```

Artifact Latest Version Search Based on Layout

Description: Search for the latest artifact version by groupId and artifactId, based on the layout defined in the repository. Search can be limited to specific repositories (local, remote and virtual) by setting the `repos` parameter.

Latest release vs. latest integration: Unless the `version` parameter is specified, the search returns the latest artifact release version. When `version` is specified, e.g. `1.0-SNAPSHOT`, the result is the latest integration version. Integration versions are determined by the [repository layout](#) of the repositories searched. For integration search to work the repository layout requires an "Artifact Path Pattern" that contains the `baseRev` token and then the `fileIntegRev` token with only literals between them.

Remote searches: By default only local and cache repositories will be used. When specifying `remote=1`, Artifactory searches for versions on remote repositories. **NOTE!** that this can dramatically slow down the search.

For Maven repositories the remote `maven-metadata.xml` will be consulted. For non-Maven layouts, remote file listing runs for all remote repositories that have the 'List Remote Folder Items' checkbox enabled.

Filtering results (Artifactory 3.0.2+): The `version` parameter can accept the `*` and/or `?` wildcards which will then filter the final result to match only those who match the given version pattern.

Artifact path pattern: The `[org]` and `[module]` fields must be specified in the [artifact path pattern](#) of the repository layout for this call to work.

Since: 2.6.0

Notes: Requires Artifactory Pro

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/search/latestVersion?[g=groupId][&a=artifactId][&v=version][&remote=1][&repos=x,y]

Produces: text/plain

Sample Output:

```

GET
/api/search/latestVersion?g=org.acme&a=artifact&v=1.0-SNAPSHOT&repos=libs-snapshot-local

1.0-201203131455-2

```

Artifact Latest Version Search Based on Properties

Description: Search for artifacts with the latest value in the "version" property. Only artifacts with a "version" property expressly defined will be returned. Results can be filtered by specifying additional properties.

{repo}: Specify a repository to search through or replace with "`_any`" to search through all repositories

{path}: Specify a path to search through or replace with "`_any`" to search through all paths

listFiles=0 (default): Artifactory will only retrieve the latest version

listFiles=1: Artifactory will retrieve the latest version and the corresponding files

You may specify filters to restrict the set of artifacts that are searched by adding any properties to your search URL

Notes: Requires Artifactory Pro

Since: 3.1.1

Security: Requires an authenticated user (not anonymous) to use the api and read permission to the repository of each artifact.

Usage: GET /api/versions/{repo}/{path}?[listFiles=0/1]&[{<property key>}=<property value>]&[{<property key>}=<property value>]

Consumes: json

Examples:

```
Return the latest version and corresponding artifacts by searching for through all repositories whose path starts with a/b and are annotated with the properties os=win and license=GPL.
```

```
GET /api/versions/_any/a/b?os=win&license=GPL&listFiles=1
{
    "version" : "1.1.2",
    "artifacts" : [ {
        "uri" : "http://...."
    }]
}
```

```
Return the latest version (without the corresponding artifacts) by searching through all repositories whose path starts with a/b and are annotated with the properties os=win and license=GPL.
```

```
Return only the version.
```

```
GET /api/versions/_any/a/b?os=win&license=GPL
{
    "version" : "1.1.2",
    "artifacts" : []
}
```

Build Artifacts Search

Description: Find all the artifacts related to a specific build.

Notes: Requires Artifactory Pro

Since: 2.6.5

Security: Requires a privileged user (can be anonymous)

Usage: POST /api/search/buildArtifacts

Consumes: application/vnd.org.jfrog.artifactory.search.BuildArtifactsRequest+json

Sample Usage:

```
POST /api/search/buildArtifacts
{
    +"buildName": "build-name" // The build name for search by
    +"buildNumber": "15" // The build number to search by, can be LATEST to search for
    the latest build number
    -"buildStatus": "Released" // Optionally search by latest build status (e.g:
    "Released")
    -"repos": ["libs-release-local,ext-release-local"] // Optionally refine search for
    specific repos, omit to search within all repositories
    -"mappings": [ // Optionally refine the search by providing a list of regexp patterns
    to search by
        {
            "input": "(.+)-sources.jar"
        },
        {
            "input": "(.+)-javadoc.jar"
        }
    ]
}
```

Produces: application/vnd.org.jfrog.artifactory.search.BuildArtifactsSearchResult+json

Sample Output:

```
POST /api/search/buildArtifacts
{
  "results" : [
    {
      "downloadUri":
      "http://localhost:8080/artifactory/libs-release-local/org/acme/lib/ver/lib-sources.jar"
    },
    {
      "downloadUri":
      "http://localhost:8080/artifactory/ext-release-local/org/acme/lib/ver/lib-ver-javadoc.jar"
    }
  ]
}
```

SECURITY

Get Users

Description: Get the users list

Since: 2.4.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: GET /api/security/users

Produces: application/vnd.org.jfrog.artifactory.security.Users+json

Sample Output:

```
GET /api/security/users
[
  {
    "name": "davids"
    "uri" : "http://localhost:8080/artifactory/api/security/users/davids"
  }, {
    "name": "danl"
    "uri" : "http://localhost:8080/artifactory/api/security/users/danl"
  }
]
```

Get User Details

Description: Get the details of an Artifactory user

Since: 2.4.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: GET /api/security/users/{userName}

Produces: application/vnd.org.jfrog.artifactory.security.User+json

Sample Output:

```
GET /api/security/users/davids
{
  user.json
}
```

Get User Encrypted Password

Description: Get the encrypted password of the authenticated requestor

Since: 3.3.0

Security: Requires a privileged user

Usage: GET /api/security/encryptedPassword

Produces: plain/text

Sample Output:

```
GET /api/security/encryptedPassword

AP5v2zs9ga7CJNZb74u3arAKE5B
```

Create or Replace User

Description: Creates a new user in Artifactory or replaces an existing user

Since: 2.4.0

Notes: Requires Artifactory Pro

Missing values will be set to the default values as defined by the consumed type.

Security: Requires an admin user

Usage: PUT /api/security/users/{userName}

Consumes: application/vnd.org.jfrog.artifactory.security.User+json

Sample Usage:

```
PUT /api/security/users/davids
{
  user.json
}
```

Update User

Description: Updates an exiting user in Artifactory with the provided user details.

Since: 2.4.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: POST /api/security/users/{userName}

Consumes: application/vnd.org.jfrog.artifactory.security.User+json

Sample Usage:

```
POST /api/security/users/davids
{
  user.json
}
```

Delete User

Description: Removes an Artifactory user.

Since: 2.4.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: DELETE /api/security/users/{userName}

Produces: application/text

Sample Usage:

```
DELETE /api/security/users/davids

User 'davids' has been removed successfully.
```

Create API Key

Description: Create an API key for the current user

Since: 4.3.0

Usage: POST /api/apiKey/auth

Produces: application/json

Sample usage:

```
POST /api/apiKey/auth
{
    "apiKey": "3OluposOtVFyCMrT+cXmCAScmVMPsSYXkWIjiyDCXsY="
}
```

Get API Key

Description: Get the current user's own API key

Since: 4.3.0

Usage: GET /api/apiKey/auth

Produces: application/json

Sample usage:

```
GET /api/apiKey/auth
{
    "apiKey": "3OluposOtVFyCMrT+cXmCAScmVMPsSYXkWIjiyDCXsY="
}
```

Revoke API Key

Description: Revokes the current user's API key

Since: 4.3.0

Usage: DELETE /api/apiKey/auth

Produces: application/json

Headers: X-Api-Key:<current-api-key>

Revoke User API Key

Description: Revokes the API key of another user

Since: 4.3.0

Security: Requires a privileged user (Admin only)

Usage: DELETE /api/apiKey/auth/{username}

Produces: application/json

Headers: X-Api-Key:<current-api-key>

Revoke All API Keys

Description: Revokes all API keys currently defined in the system

Since: 4.3.0
Security: Requires a privileged user (Admin only)
Usage: DELETE /api/apiKey?deleteAll={0/1}
Produces: application/json
Headers: X-Api-Key:<current-api-key>

Get Groups

Description: Get the groups list
Since: 2.4.0
Notes: Requires Artifactory Pro
Security: Requires an admin user
Usage: GET /api/security/groups
Produces: application/vnd.org.jfrog.artifactory.security.Groups+json
Sample Output:

```
GET /api/security/groups
[
  {
    "name": "readers"
    "uri" : "http://localhost:8080/artifactory/api/security/groups/readers"
  }, {
    "name": "tech-leads"
    "uri" : "http://localhost:8080/artifactory/api/security/groups/tech-leads"
  }
]
```

Get Group Details

Description: Get the details of an Artifactory Group
Since: 2.4.0
Notes: Requires Artifactory Pro
Security: Requires an admin user
Usage: GET /api/security/groups/{groupName}
Produces: application/vnd.org.jfrog.artifactory.security.Group+json
Sample Output:

```
GET /api/security/groups/dev-leads
{
  group.json
}
```

Create or Replace Group

Description: Creates a new group in Artifactory or replaces an existing group
Since: 2.4.0
Notes: Requires Artifactory Pro
Missing values will be set to the default values as defined by the consumed type.
Security: Requires an admin user
Usage: PUT /api/security/groups/{groupName}
Consumes: application/vnd.org.jfrog.artifactory.security.Group+json
Sample Usage:

```
PUT /api/security/groups/dev-leads
{
  group.json
}
```

Update Group

Description: Updates an existing group in Artifactory with the provided group details.

Since: 2.4.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: POST /api/security/groups/{groupName}

Consumes: application/vnd.org.jfrog.artifactory.security.Group+json

Sample Usage:

```
POST /api/security/groups/dev-leads
{
  group.json
}
```

Delete Group

Description: Removes an Artifactory group.

Since: 2.4.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: DELETE /api/security/groups/{groupName}

Produces: application/text

Sample Usage:

```
DELETE /api/security/groups/dev-leads
Group 'dev-leads' has been removed successfully.
```

Get Permission Targets

Description: Get the permission targets list

Since: 2.4.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: GET /api/security/permissions

Produces: application/vnd.org.jfrog.artifactory.security.PermissionTargets+json

Sample Output:

```
GET /api/security/permissions
[
  {
    "name": "readSourceArtifacts"
    "uri" :
"http://localhost:8080/artifactory/api/security/permissions/readSourceArtifacts"
  },
  {
    "name": "populateCaches"
    "uri" :
"http://localhost:8080/artifactory/api/security/permissions/populateCaches"
  }
]
```

Get Permission Target Details

Description: Get the details of an Artifactory Permission Target

Since: 2.4.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: GET /api/security/permissions/{permissionTargetName}

Produces: application/vnd.org.jfrog.artifactory.security.PermissionTarget+json

Sample Output:

```
GET /api/security/permissions/populateCaches
{
  permission-target.json
}
```

Create or Replace Permission Target

Description: Creates a new permission target in Artifactory or replaces an existing permission target

Since: 2.4.0

Notes: Requires Artifactory Pro

Missing values will be set to the default values as defined by the consumed type.

Security: Requires an admin user

Usage: PUT /api/security/permissions/{permissionTargetName}

Consumes: application/vnd.org.jfrog.artifactory.security.PermissionTarget+json

Sample Usage:

```
PUT /api/security/permissions/populateCaches
{
  permission-target.json
}
```

Delete Permission Target

Description: Deletes an Artifactory permission target.

Since: 2.4.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: DELETE /api/security/permissions/{permissionTargetName}

Produces: application/text

Sample usage:

```
DELETE /api/security/permissions/populateCaches

Permission Target 'remoteCachePopulation' has been removed successfully.
```

Effective Item Permissions

Description: Returns a list of effective permissions for the specified item (file or folder).

Only users and groups with some permissions on the item are returned. Supported by local and local-cached repositories.

Permissions are returned according to the following conventions:

m=admin; d=delete; w=deploy; n=annotate; r=read

Notes: Requires Artifactory Pro

Since: 2.3.4

Security: Requires a valid admin or local admin user.

Usage: GET /api/storage/{repoKey}/{itemPath}?permissions

Produces: application/vnd.org.jfrog.artifactory.storage.ItemPermissions+json

Sample Output:

```
GET /api/storage/libs-release-local/org/acme?permissions
{
  "uri": "http://localhost:8080/artifactory/api/storage/libs-release-local/org/acme"
  " principals": {
    "users" : {
      "bob": ["r", "w", "m"],
      "alice" : ["d", "w", "n", "r"]
    },
    "groups" : {
      "dev-leads" : ["m", "r", "n"],
      "readers" : ["r"]
    }
  }
}
```

Security Configuration

Description: Retrieve the security configuration (security.xml).

Since: 2.2.0

Notes: This is an advanced feature - make sure the new configuration is really what you wanted before saving.

Security: Requires a valid admin user

Usage: GET /api/system/security

Produces: application/xml

Sample Output:

```
GET /api/system/security

<security.xml/>
```

Save Security Configuration

Description: Save the security configuration (security.xml). Requires the security.xml file from the same version.

Since: 2.2.0

Notes: This API is **deprecated**.

Security: Requires a valid admin user

Usage: POST /api/system/security

Consumes: application/xml
Sample Usage:

```
POST /api/system/security

<security.xml/>
```

Activate Master Key Encryption

Description: Creates a new master key and activates master key encryption.

Since: 3.2.2

Notes: This is an advanced feature intended for administrators

Security: Requires a valid admin user

Usage: POST /api/system/encrypt

Produces: text/plain

Sample Usage:

```
POST /api/system/encrypt

DONE
```

Deactivate Master Key Encryption

Description: Removes the current master key and deactivates master key encryption.

Since: 3.2.2

Notes: This is an advanced feature intended for administrators

Security: Requires a valid admin user

Usage: POST /api/system/decrypt

Produces: text/plain

Sample Usage:

```
POST /api/system/decrypt

DONE
```

Set GPG Public Key

Description: Sets the public key that Artifactory provides to Debian clients to verify packages

Security: Requires a valid admin user

Usage: PUT /api/gpg/key/public

Produces: text/plain

Since: 3.3

Sample Usage:

```
PUT /api/gpg/key/public
```

Get GPG Public Key

Description: Gets the public key that Artifactory provides to Debian clients to verify packages

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/gpg/key/public

Produces: text/plain

Since: 3.3

Sample Usage:

```
GET /api/gpg/key/public
```

Set GPG Private Key

Description: Sets the private key that Artifactory will use to sign Debian packages

Security: Requires a valid admin user

Usage: PUT /api/gpg/key/private

Produces: text/plain

Since: 3.3

Sample Usage:

```
PUT /api/gpg/key/private
```

Set GPG Pass Phrase

Description: Sets the pass phrase required signing Debian packages using the private key

Security: Requires a valid admin user

Usage: PUT /api/gpg/key/passphrase

Headers: -H X-GPG-PASSPHRASE:passphrase

Produces: text/plain

Since: 3.3

Sample Usage:

```
PUT /api/gpg/key/passphrase
```

REPOSITORIES

Get Repositories

Description: Returns a list of minimal repository details for all repositories of the specified type.

Since: 2.2.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/repositories[?type=repositoryType (local|remote|virtual)]

Produces: application/vnd.org.jfrog.artifactory.repositories.RepositoryDetailsList+json

Sample Output:

```

GET /api/repositories
[
  {
    "key" : "libs-releases-local",
    "type" : "LOCAL",
    "description" : "Local repository for in-house libraries",
    "url" : "http://localhost:8080/artifactory/libs-releases-local"
  }, {
    "key" : "libs-snapshots-local",
    "type" : "LOCAL",
    "description" : "Local repository for in-house snapshots",
    "url" : "http://localhost:8080/artifactory/libs-snapshots-local"
  }
]

```

Repository Configuration

Description: Retrieves the current configuration of a repository. Supported by local, remote and virtual repositories.

Since: 2.3.0

Notes: Requires Artifactory Pro

Security: Requires a valid user for a shared remote repository and admin user for anything else. Shared remote repository data is sanitized for security when a non-admin user is used.

Usage: GET /api/repositories/{repoKey}

Produces: application/vnd.org.jfrog.artifactory.repositories.LocalRepositoryConfiguration+json, application/vnd.org.jfrog.artifactory.repositories.RemoteRepositoryConfiguration+json, application/vnd.org.jfrog.artifactory.repositories.VirtualRepositoryConfiguration+json

Sample Output:

```

GET /api/repositories/libs-release-local
{
  repository-config.json
}

```

Create or Replace Repository Configuration

Description: Creates a new repository in Artifactory with the provided configuration or replaces the configuration of an existing repository. Supported by local, remote and virtual repositories.

A position may be specified using the `pos` parameter. If the map size is shorter than `pos` the repository is the last one (the default behavior).

Since: 2.3.0

Notes: Requires Artifactory Pro

An existing repository with the same key are removed from the configuration and its content is removed!

Missing values are set to the default values as defined by the consumed type spec.

Security: Requires an admin user

Usage: PUT /api/repositories/{repoKey}[?pos=position]

Consumes: application/vnd.org.jfrog.artifactory.repositories.LocalRepositoryConfiguration+json, application/vnd.org.jfrog.artifactory.repositories.RemoteRepositoryConfiguration+json, application/vnd.org.jfrog.artifactory.repositories.VirtualRepositoryConfiguration+json

Sample Usage:

```

PUT /api/repositories/libs-release-local?pos=2
{
  repository-config.json
}

```

Update Repository Configuration

Description: Updates an exiting repository configuration in Artifactory with the provided configuration elements. Supported by local, remote and virtual repositories.

Since: 2.3.0

Notes: Requires Artifactory Pro

The class of a repository (the `rclass` attribute cannot be updated).

Security: Requires an admin user

Usage: POST /api/repositories/{repoKey}

Consumes: application/vnd.org.jfrog.artifactory.repositories.LocalRepositoryConfiguration+json, application/vnd.org.jfrog.artifactory.repositories.RemoteRepositoryConfiguration+json, application/vnd.org.jfrog.artifactory.repositories.VirtualRepositoryConfiguration+json

Sample Usage:

```
POST /api/repositories/libs-release-local
{
  repository-config.json
}
```

Delete Repository

Description: Removes a repository configuration together with the whole repository content. Supported by local, remote and virtual repositories.

Since: 2.3.0

Notes: Requires Artifactory Pro

Security: Requires an admin user

Usage: DELETE /api/repositories/{repoKey}

Produces: application/text

Sample Usage:

```
DELETE /api/repositories/libs-release-local

Repository 'libs-release-local' and all its content have been removed successfully.
```

Remote Repository Configuration

Description: Repository Configuration (Deprecated)

Gets the shared configuration of a remote repository.

Since: 2.2.0

Notes: This API is **deprecated**. Use the Get Repository Configuration API instead.

Security: Requires a valid user for a shared remote repository and admin user for anything else. Shared remote repository data will be sanitized for security when non-admin user is used.

Usage: GET /api/repositories/{remoteRepoName}/configuration

Produces: application/vnd.org.jfrog.artifactory.repositories.SharedRemoteRepositoryConfiguration+json

Sample Output:

```
GET /api/repositories/remote-repo/configuration
{
  repository-config.json
}
```

Calculate YUM Repository Metadata

Description: Calculates/recalculates the YUM metdata for this repository, based on the RPM package currently hosted in the repository. Supported by local repositories only.

Calculation can be synchronous (the default) or asynchronous.
Please see the [YUM integration](#) documentation for more details.

Notes: Requires Artifactory Pro. Immediate calculation requests cannot be called on repositories with automatic asynchronous calculations enabled.

Security: Requires a valid admin user

Usage: POST /api/yum/{repoKey}[?async=0/1]

Produces: application/text

Since: 2.3.5

Sample Output:

```
POST /api/yum/libs-release-local?async=1

YUM metadata calculation for repository 'libs-release-local' accepted.
```

Calculate NuGet Repository Metadata

Description: Recalculates all the NuGet packages for this repository (local/cache/virtual), and re-annotate the NuGet properties for each NuGet package according to it's internal nuspec file.

Please see the [NuGet integration](#) documentation for more details.

Supported by local, local-cache, remote and virtual repositories.

Notes: Requires Artifactory Pro.

Security: Requires a valid admin user

Usage: POST /api/nuget/{repoKey}/reindex

Produces: application/text

Since: 3.0.3

Sample Output:

```
POST /api/nuget/nuget-local/reindex

NuGet reindex calculation for repository 'nuget-local' accepted.
```

Calculate Npm Repository Metadata

Description: Recalculates the npm search index for this repository (local/virtual). Please see the [Npm integration](#) documentation for more details. Supported by local and virtual repositories.

Notes: Requires Artifactory Pro.

Security: Requires a valid admin user

Usage: POST /api/npm/{repoKey}/reindex

Produces: application/text

Since: 3.2.0

Sample Output:

```
POST /api/npm/npm-local/reindex

Recalculating index for npm repository npm-local scheduled to run
```

Calculate Maven Index

Description: Calculates/caches a Maven index for the specified repositories.

For a virtual repository specify all underlying repositories that you want the aggregated index to include.

Calculation can be forced, which for remote repositories will cause downloading of a remote index even if a locally cached index has not yet expired; and index recalculation based on the cache on any failure to download the remote index, including communication errors (the default behavior is to only use the cache when a remote index cannot be found and returns a 404). Forcing has no effect on local repositories index calculation.

Please see the [Exposing Maven Indexes](#) documentation for more details.

Notes: Requires Artifactory Pro.

Security: Requires a valid admin user
Usage: POST /api/maven[?repos=x[,y]][&force=0/1]
Produces: application/text
Since: 2.5.0
Sample Output:

```
POST /api/maven?repos=libs-release-local,ext-release-local&force=1

Maven index refresh for repositories '[libs-release-local, ext-release-local]' has
been accepted.
```

Calculate Maven Metadata

Description: Calculates Maven metadata on the specified path (local repositories only).
Security: Requires a valid user with deploy permissions
Usage: POST /api/maven/calculateMetadata/{repoKey}/{folder-path}
Produces: application/text
Since: 3.0.2
Sample Output:

```
POST /api/maven/calculateMetadata/libs-release-local/org/acme
OK
```

Calculate Debian Repository Metadata

Description: Calculates/recalculates the Packages and Release metadata for this repository,based on the Debian packages in it. Calculation can be synchronous (the default) or asynchronous. Please refer to [Debian Repositories](#) for more details. Supported by local repositories only.
Notes: Requires Artifactory Pro.
Security: Requires a valid admin user
Usage: POST /api/deb/reindex/{repoKey} [?async=0/1]
Headers (Optional): -H X-GPG-PASSPHRASE:passphrase
Produces: application/text
Since: 3.3
Sample Output:

```
POST /api/deb/reindex/debian-local

Recalculating index for Debian repository debian-local scheduled to run.
```

SYSTEM & CONFIGURATION

System Info

Description: System Info
Get general system information.
Since: 2.2.0
Security: Requires a valid admin user
Usage: GET /api/system
Produces: text/plain
Sample Output:

```
GET /api/system  
  
system info output text
```

System Health Ping

Description: Get a simple status response about the state of Artifactory
Returns 200 code with an 'OK' text if Artifactory is working properly, if not will return an HTTP error code with a reason.
Since: 2.3.0
Security: Requires a valid user (can be anonymous)
Usage: GET /api/system/ping
Produces: text/plain
Sample Output:

```
GET /api/system/ping  
OK
```

General Configuration

Description: Get the general configuration (artifactory.config.xml).
Since: 2.2.0
Security: Requires a valid admin user
Usage: GET /api/system/configuration
Produces: application/xml (http://www.jfrog.org/xsd/artifactory-v1_4_5.xsd)
Sample Output:

```
GET /api/system/configuration  
  
<artifactory.config.xml/>
```

Save General Configuration

Description: Save the general configuration (artifactory.config.xml).
Since: 2.2.0
Notes: This is an advanced feature - make sure the new configuration is really what you wanted before saving.
Security: Requires a valid admin user
Usage: POST /api/system/configuration
Consumes: application/xml (http://www.jfrog.org/xsd/artifactory-v1_4_5.xsd)
Sample Usage:

```
POST /api/system/configuration  
  
<artifactory.config.xml/>
```

Update Custom URL Base

Description: Changes the Custom URL base
Since: 3.9.0
Security: Requires a valid admin user
Usage: PUT /api/system/configuration/baseUrl

Example: curl -X PUT "http://localhost:8081/artifactory/api/system/configuration/baseUrl" -d 'https://mycompany.com:444/artifactory' -uadmin:password -H "Content-type: text/plain"

Sample Output:

```
URL base has been successfully updated to "https://mycompany.com:444/artifactory".
```

License Information

Description: Retrieve information about the currently installed license.

Since: 3.3.0

Security: Requires a valid admin user

Usage: GET /api/system/license

Produces: application/json

Sample Output:

```
GET /api/system/license
{
  "type" : "Commercial",
  "validThrough" : "May 15, 2014",
  "licensedTo" : "JFrog inc."
}
```

Install License

Description: Install new license key or change the current one.

Since: 3.3.0

Security: Requires a valid admin user

Usage: POST /api/system/license

Produces: application/json

Consumes: application/json ({ "licenseKey": "your supplied license key ..." })

Sample Output:

```
POST /api/system/license
{
  "status" : 200,
  "message" : "The license has been successfully installed."
}
```

Version and Add-ons information

Description: Retrieve information about the current Artifactory version, revision, and currently installed Add-ons

Since: 2.2.2

Security: Requires a valid user (can be anonymous)

Usage: GET /api/system/version

Produces: application/vnd.org.jfrog.artifactory.system.Version+json

Sample Output:

```
GET /api/system/version
{
  "version" : "2.2.2",
  "revision" : "10427",
  "addons" : [ "build", "ldap", "properties", "rest", "search", "sso", "watch",
"webstart" ]
}
```

PLUGINS

Execute Plugin Code

Description: Executes a named execution closure found in the executions section of a [user plugin](#).

Execution can take parameters and be synchronous (the default) or asynchronous.

Since: 2.3.1

Notes: Requires Artifactory Pro

Security: Requires an authenticated user (the plugin can control which users/groups are allowed to trigger it)

Usage: POST /api/plugins/execute/{executionName}?[params=p1=v1[,v2][|p2=v3][&async=1]]

Produces: text/plain

Sample Output:

```
POST /api/plugins/execute/cleanup?params=suffix=SNAPSHOT|types=jar,war,zip&async=1
OK
```

Retrieve All Available Plugin Info

Description: Retrieves all available [user plugin](#) information (subject to the permissions of the provided credentials).

Since: 2.5.2

Notes: Requires Artifactory Pro

Security: Requires an authenticated user.

Usage: GET /api/plugins

Produces: application/json

Sample Output:

```

GET /api/plugins
{
  "executions": [
    {
      "name": "execution1",
      "version": "version",
      "description": "description",
      "users": ["user1"],
      "groups": ["group1", "group2"],
      "params": {}
    }
  ],
  "staging": [
    {
      "name": "strategy1",
      "version": "1.0",
      "description": "desc",
      "params": {"key1": "val1"}
    }
  ]
}

```

Retrieve Plugin Info Of A Certain Type

Description: Retrieves all available [user plugin](#) information (subject to the permissions of the provided credentials) of the specified type.

Since: 2.5.2

Notes: Requires Artifactory Pro

Security: Requires an authenticated user.

Usage: GET /api/plugins/{pluginType}

Produces: application/json

Sample Output:

```

GET /api/plugins/staging
{
  "staging": [
    {
      "name": "strategy1",
      "version": "1.0",
      "description": "desc",
      "params": {"key1": "val1"}
    }
  ]
}

```

Retrieve Build Staging Strategy

Description: Retrieves a build staging strategy defined by a [user plugin](#).

Since: 2.5.2

Notes: Requires Artifactory Pro

Security: Requires an authenticated user.

Usage: GET /api/plugins/build/staging/{strategyName}?buildName={buildName}&[params=p1=v1[,v2][|p2=v3]]

Produces: application/vnd.org.jfrog.plugins.BuildStagingStrategy

Sample Output:

```

GET /api/plugins/build/staging/strategy1?buildName=build1?params=types=jar,war,zip
{
    "defaultModuleVersion": {
        "moduleId": "moduleId",
        "nextRelease": "nextRelease",
        "nextDevelopment": "nextDevelopment"
    },
    "vcsConfig": {
        "useReleaseBranch": true,
        "releaseBranchName": "branchName",
        "createTag": true,
        "tagUrlOrName": "tagUrl",
        "tagComment": "comment",
        "nextDevelopmentVersionComment": "comment"
    },
    "promotionConfig": {
        "targetRepository": "repoKey",
        "comment": "comment",
        "status": "statusName"
    }
}

```

Execute Build Promotion

Description: Executes a named promotion closure found in the `promotions` section of a [user plugin](#).

Since: 2.5.2

Notes: Requires Artifactory Pro

Security: Requires an authenticated user.

Usage: POST /api/plugins/build/promote/{promotionName}/{buildName}/{buildNumber}?[params=p1=v1[,v2][|p2=v3]]

Produces: text/plain

Sample Output:

```

POST /api/plugins/build/promote/promotion1/build1/3?params=types=jar,war,zip
OK

```

Reload Plugins

Description: Reloads user plugins if there are modifications since the last user plugins reload. Works regardless of the automatic user plugins refresh interval.

Since: 2.9.0

Notes: Requires Artifactory Pro

Security: Requires a valid admin user

Usage: POST /api/plugins/reload

Produces: text/plain

Sample Output:

```

POST /api/plugins/reload
Successfully loaded: myplugin1.groovy, myplugin2.groovy

```

IMPORT & EXPORT

Import Repository Content

Description: Import one or more repositories.

Since: 2.2.2

Security: Requires a valid admin user

Usage: POST: /api/import/repositories

Requests Params:

path - The base path to import from (may contain a single repo or multiple repos with named sub folders)

repo - Empty/null repo -> all

metadata - Include metadata - default 1

verbose - Verbose - default 0

Produces: text/plain

Sample Output:

```
POST: /api/import/repositories?path=pathToRepos&verbose=1
```

Import System Settings Example

Description: Returned default Import Settings JSON.

Since: 2.4.0

Security: Requires a valid admin user

Usage: GET: /api/import/system

Produces: application/vnd.org.jfrog.artifactory.system.ImportSettings+json

Sample Usage:

```
GET /api/import/system
{
  "importPath" : "/import/path",
  "includeMetadata" : true,
  "verbose" : false,
  "failOnError" : true,
  "failIfEmpty" : true
}
```

Full System Import

Description: Import full system from a server local Artifactory export directory.

Since: 2.4.0

Security: Requires a valid admin user

Usage: POST: /api/import/system

Consumes: application/vnd.org.jfrog.artifactory.system.ImportSettings+json

Produces: text/plain

Sample Usage:

```
POST /api/import/system
{
  import-settings.json
}
```

Export System Settings Example

Description: Returned default Export Settings JSON.
Since: 2.4.0
Security: Requires a valid admin user
Usage: GET: /api/export/system
Produces: application/vnd.org.jfrog.artifactory.system.ExportSettings+json
Sample Usage:

```
GET /api/export/system
{
  "exportPath" : "/export/path",
  "includeMetadata" : true,
  "createArchive" : false,
  "bypassFiltering" : false,
  "verbose" : false,
  "failOnError" : true,
  "failIfEmpty" : true,
  "m2" : false,
  "incremental" : false,
  "excludeContent" : false
}
```

Export System

Description: Export full system to a server local directory.
Since: 2.4.0
Security: Requires a valid admin user
Usage: POST: /api/export/system
Consumes: application/vnd.org.jfrog.artifactory.system.ExportSettings+json, application/json
Produces: text/plain
Sample Usage:

```
POST /api/export/system{ export-settings.json }
```

ERROR RESPONSES

In case of an error, Artifactory will return an error response in JSON format. The response contains the HTTP status code and error message. For example, a badly formatted API call would return the "404, File not found" response below:

```
{
  "errors" : [ {
    "status" : 404,
    "message" : "File not found."
  } ]
}
```

Security: Requires a valid user with deploy permissions and Bintray credentials defined (for more details, please refer to [Entering your Bintray credentials](#)).

Repository Configuration JSON

Repository Configuration JSON

Legend

+	Mandatory element in create/replace queries
-	Optional element in create/replace queries
(default)	The default value when unspecified in create/replace queries

application/vnd.org.jfrog.artifactory.repositories.LocalRepositoryConfiguration+json

```
{
  - "key": "local-repol",
  + "rclass" : "local",
  + "packageType": "maven" | "gradle" | "ivy" | "sbt" | "nuget" | "gems" | "npm" |
"bower" | "debian" | "pypi" | "docker" | "vagrant" | "gitlfs" | "yum" | "generic"
  - "description": "The local repository public description",
  - "notes": "Some internal notes",
  - "includesPattern": "**/*" (default),
  - "excludesPattern": "" (default),
  - "repoLayoutRef" : "maven-2-default",
  - "debianTrivialLayout" : false,
  - "checksumPolicyType": "client-checksums" (default) | "server-generated-checksums"
  - "handleReleases": true (default),
  - "handleSnapshots": true (default),
  - "maxUniqueSnapshots": 0 (default),
  - "snapshotVersionBehavior": "unique" | "non-unique" (default) | "deployer",
  - "suppressPomConsistencyChecks": false (default),
  - "blacklisted": false (default),
  - "propertySets": ["ps1", "ps2"],
  - "archiveBrowsingEnabled" : false,
  - "calculateYumMetadata" : false,
  - "yumRootDepth" : 0
}
```

application/vnd.org.jfrog.artifactory.repositories.RemoteRepositoryConfiguration+json

```
{
  - "key": "remote-repol",
  + "rclass" : "remote",
  + "packageType": "maven" | "gradle" | "ivy" | "sbt" | "nuget" | "gems" | "npm" |
"bower" | "debian" | "pypi" | "docker" | "yum" | "vcs" | "p2" | "generic"
  + "url" : "http://host:port/some-repo",
  - "username": "remote-repo-user",
  - "password": "pass",
  - "proxy": "proxy1",
  - "description": "The remote repository public description",
  - "notes": "Some internal notes",
  - "includesPattern": "**/*" (default),
  - "excludesPattern": "" (default),
  - "remoteRepoChecksumPolicyType": "generate-if-absent" (default) | "fail" |
"ignore-and-generate" | "pass-thru",
  - "handleReleases": true (default),
  - "handleSnapshots": true (default),
  - "maxUniqueSnapshots": 0 (default),
  - "suppressPomConsistencyChecks": false (default),
  - "hardFail": false (default),
  - "offline": false (default),
  - "blacklistedOut": false (default),
  - "storeArtifactsLocally": true (default),
  - "socketTimeoutMillis": 15000 (default),
  - "localAddress": "212.150.139.167",
  - "retrievalCachePeriodSecs": 43200 (default),
  - "failedRetrievalCachePeriodSecs": 30 (default),
  - "missedRetrievalCachePeriodSecs": 7200 (default),
  - "unusedArtifactsCleanupEnabled": false (default),
  - "unusedArtifactsCleanupPeriodHours": 0 (default),
  - "fetchJarsEagerly": false (default),
  - "fetchSourcesEagerly": false (default),
  - "shareConfiguration": false (default),
  - "synchronizeProperties": false (default),
  - "propertySets": ["ps1", "ps2"]
  - "allowAnyHostAuth": false (default),
  - "enableCookieManagement": false (default)
  - "bowerRegistryUrl": "https://bower.herokuapp.com" (default)
  - "vcsType": "GIT" (default)
  - "vcsGitProvider": "GITHUB" (default) | "BITBUCKET" | "STASH" | "ARTIFACTORY" |
"CUSTOM"
  - "vcsGitDownloadUrl": "" (default)
}
```

application/vnd.org.jfrog.artifactory.repositories.VirtualRepositoryConfiguration+json

```
{
  - "key": "virtual-repo1",
  + "rclass" : "virtual",
  + "packageType": "maven" | "gradle" | "ivy" | "sbt" | "nuget" | "gems" | "npm" |
"bower" | "pypi" | "p2" | "generic"
  - "repositories": ["local-rep1", "local-rep2", "remote-rep1", "virtual-rep2"]
  - "description": "The virtual repository public description",
  - "notes": "Some internal notes",
  - "includesPattern": "**/*" (default),
  - "excludesPattern": "" (default),
  - "debianTrivialLayout" : false
  - "artifactoryRequestsCanRetrieveRemoteArtifacts": false,
  - "keyPair": "keypair1",
  - "pomRepositoryReferencesCleanupPolicy": "discard_active_reference" (default) |
"discard_any_reference" | "nothing"
  - "defaultDeploymentRepo": "local-repo1"
}
```

Security Configuration JSON

Security Configuration JSON

Legend

+	Mandatory element in create/replace queries
-	Optional element in create/replace queries
!	Read-only element
(default)	The default value when unspecified in create/replace queries

application/vnd.org.jfrog.artifactory.security.User+json

```
{
  - "name": "davids",
  + "email" : "davids@jfrog.com",
  + "password": "****" (write-only, never returned),
  - "admin": false (default),
  - "profileUpdatable": true (default),
  - "internalPasswordDisabled": false (default),
  ! "lastLoggedIn": ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ),
  ! "realm": "Internal",
  - "groups" : [ "deployers", "users" ]
}
```

application/vnd.org.jfrog.artifactory.security.Group+json

```
{
  - "name": "dev-leads",
  - "description" : "The development leads group",
  - "autoJoin" : false (default),
  - "realm": "Realm name (e.g. ARTIFACTORY, CROWD)",
  - "realmAttributes": "Realm attributes for use by LDAP"
}
```

application/vnd.org.jfrog.artifactory.security.PermissionTarget+json

Permissions are set/returned according to the following conventions:
m=admin; d=delete; w=deploy; n=annotate; r=read

```
{
  - "name": "populateCaches",
  - "includesPattern": "/**" (default),
  - "excludesPattern": "" (default),
  - "repositories": ["local-rep1", "local-rep2", "remote-rep1", "virtual-rep2"],
  - "principals": {
      "users" : {
          "bob": ["r", "w", "m"],
          "alice" : ["d", "w", "n", "r"]
      },
      "groups" : {
          "dev-leads" : ["m", "r", "n"],
          "readers" : ["r"]
      }
  }
}
```

System Settings JSON

System Settings JSON

Legend

+	Mandatory element
-	Optional element
(default)	The default value when unspecified

application/vnd.org.jfrog.artifactory.system.ImportSettings+json

```
{
  + "importPath" : "/import/path" (A path to a directory on the local file system of Artifactory server),
  - "includeMetadata" : true (default),
  - "verbose" : false (default),
  - "failOnError" : true (default),
  - "failIfEmpty" : true (default)
}
```

application/vnd.org.jfrog.artifactory.system.ExportSettings+json

```
{
  + "exportPath" : "/export/path" (A path to a directory on the local file system of Artifactory server),
  - "includeMetadata" : true (default),
  - "createArchive" : false (default),
  - "bypassFiltering" : false (default),
  - "verbose" : false (default),
  - "failOnError" : true (default),
  - "failIfEmpty" : true (default),
  - "m2" : false (default),
  - "incremental" : false (default),
  - "excludeContent" : false (default)
}
```

application/vnd.org.jfrog.artifactory.system.Version+json

```
{
  "version" : "2.2.2",
  "revision" : "10427",
  "addons" : [ "build", "ldap", "properties", "rest", ... ] (list of active addons)
}
```

Configuring Artifactory

Overview

You can access the General Configuration settings of Artifactory in the **Admin** tab under **Configuration | General**.

Saving changes

Any changes you make must be saved in order for them to take effect.

General Settings

The fields under General Settings allow you to set up various global parameters in Artifactory. The ? icon next to each field provides a detailed description of the field. Fields marked with a red star are mandatory.

Page Contents

- Overview
- General Settings

- Folder Download Settings
- Look and Feel Settings (Branding)
- Custom Message

Read More

- Configuring Repositories
- Configuring Security
- Mail Server Configuration
- Configuration Files
- Exposing Maven Indexes
- Clustering Artifactory

General Configuration

General Settings

Server Name [?](#)

Custom URL Base [?](#)

File Upload Max Size * [?](#)

100

Bintray Max Files Upload *

0

Date Format * [?](#)

dd-MM-yy HH:mm:ss z

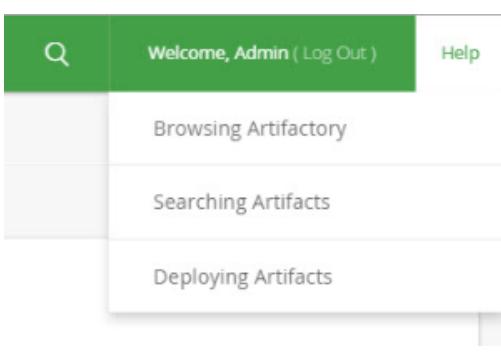
Global Offline Mode [?](#)

Enable Help Component

The General Settings fields are as follows:

Server Name (mandatory)	The name of the server to be displayed on the title of each page.
-------------------------	---

The name of the server to be displayed on the title of each page.

Custom URL Base	<p>By default, URLs generated in Artifactory use the context URL returned by your servlet container as a base.</p> <p>A custom URL base is useful when Artifactory is running behind a proxy. In this case the base for URLs generated in Artifactory for links and redirect responses must be specified manually.</p> <p>Another reason to change the base URL would be to have non-request-based operations in Artifactory use the correct address, for example in generated emails.</p> <p>This may also be modified using the Update Custom URL Base REST API.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>Overriding the Custom URL Base</p> <p>You can override the Custom URL Base by adding an <code>X-Artifactory-Override-Base-Url</code> HTTP header to your request. This can be useful if you need Artifactory to answer to more than one host name.</p> </div>
File Upload Max Size	Maximum size allowed for files uploaded via the web interface.
Date Format (mandatory)	The date format for displaying dates in the web interface.
Global Offline Mode	When checked, Artifactory will behave as if it is not connected to an external network (such as the internet), and therefore, will not query remote repositories (regardless of the offline status of any specific remote repository).
Enable Help Component	<p>If set, Artifactory will display context sensitive help topics in the top right corner of the UI.</p> 

Folder Download Settings

Folder Download Settings

Enable Folder Download

Max Size ?
1024

Max Number of Files ?
5000

Max Parallel Folder Downloads ?
10

Enable Folder Download	Must be set to enable folder download
Max Size	The maximum size (in MB) of a folder that may be downloaded
Max Number of Files	The maximum number artifacts that may be downloaded under one folder
Max Parallel Folder Downloads	The maximum number of folder download requests that may be run concurrently

Look and Feel Settings (Branding)

In the Look and Feel Settings you can configure Artifactory to present your company logo in the top left corner of the screen and customize the footer text.

You may either upload an image to be used locally, or reference a remote image URL.

Look and Feel Settings

Upload from: [File](#) [Url](#)

Logo File

Drop file here or [Select File](#) [Clear](#)

artifactory By JFrog

File	If selected, upload your company logo image file.
	<p>Logo file upload The uploaded logo file is copied into the <code>ARTIFACTORY_HOME/etc/ui</code> folder.</p>
URL	If selected, specify the URL from which Artifactory should display your company logo

Custom Message

You can display a **Custom Message** at the top of Artifactory screens.

To configure the custom message, in the **Admin** module, select **Configuration | General**.

<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Show Only in Home Page
Title	Title Color
Upgrade Notice	#43A047
Message	
<pre>We have upgraded to Artifactory 4. Make sure all your repositories are [http://www.jfrog.com/confluence/display/RTF/Upgrading+Artifactory#UpgradingArtifactory-SinglePackageTypeRepositories,single package type].</pre>	

(i) To embed a link inside the message use : [http://example.com, text]

Enabled	Toggles the custom message on and off.
Show Only in Home Page	When set, the custom message will only be displayed on the Artifactory Home Page .
Title	The title for the message.
Title Color	Click on the colored rectangle to select a color, or enter the color in Hex format.
Message	

Configuring Repositories

Overview

Artifactory hosts three types of repository:

- [Local](#)
- [Remote](#)
- [Virtual](#)

Local and remote repositories are true physical repositories, while a virtual repository is actually an aggregation of them used to create controlled domains for search and resolution of artifacts.

To configure repositories, in the **Admin** module, select **Repositories**.

Repositories can be created, deleted, edited, ordered and aggregated.

Single Package Type

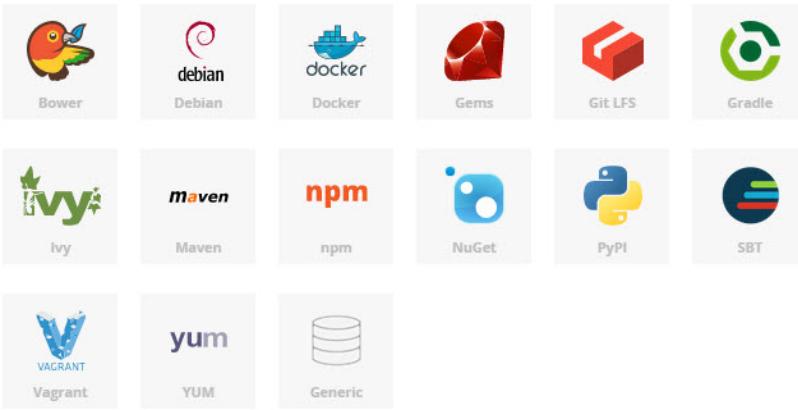
When creating any repository, you must specify its package type; this is a fundamental characteristic of the repository and can not be changed later. Once the repository type is set, Artifactory will index artifacts and calculate the corresponding metadata for every package uploaded which optimizes performance when resolving artifacts. Note that virtual repositories can only include repositories of the same type.

Wrong Package Type

While Artifactory will not prevent you from uploading a package of the wrong type to a repository, we strongly recommend maintaining consistency between the repository type and packages you upload.

If you do upload packages of the wrong type to a repository, Artifactory will not index the package or update the metadata for the repository.

Select Package Type



Page Contents

- Overview
- Single Package Type
 - Generic Repositories
- Local Repositories
- Remote Repositories
- Virtual Repositories
 - The Default Virtual Repository
 - Virtual Resolution Order
- General Resolution Order

Read More

- Common Settings
- Local Repositories
- Remote Repositories
- Smart Remote Repositories
- Virtual Repositories

Generic Repositories

You may define a repository as **Generic** in which case it has no particular type, and you may upload packages of any type. Generic repositories do not maintain separate package indexes. For using a client associated with a specific package type (e.g. yum, gem) you should create a matching repository.

Local Repositories

Local repositories are physical, locally-managed repositories into which you can deploy artifacts.

Artifacts in a local repository can be accessed directly using the following URL:

`http://<host>:<port>/artifactory/<local-repository-name>/<artifact-path>`

Artifactory is deployed with a number of pre-configured local repositories which can be used for internal and external releases, snapshots and plugins.

For full details on configuring local repositories, please refer to [Local Repositories](#).

Remote Repositories

A remote repository serves as a caching proxy for a repository managed at a remote URL (which may itself be another Artifactory remote

repository).

Artifacts are stored and updated in remote repositories according to various configuration parameters that control the caching and proxying behavior. You can remove artifacts from a remote repository cache but you cannot actually deploy a new artifact into a remote repository.

Artifacts in a remote repository can be accessed directly using the following URL:

```
http://<host>:<port>/artifactory/<remote-repository-name>/<artifact-path>
```

This URL will fetch a remote artifact to the cache if it has not yet been stored.

In some cases it is useful to directly access artifacts that are already stored in the cache (for example to avoid remote update checks).

To directly access artifacts that are already stored in the cache you can use the following URL:

```
http://<host>:<port>/artifactory/<remote-repository-name>-cache/<artifact-path>
```

Artifactory is deployed with a number of pre-configured, remote repositories which are in common use. Of course you can change these according to the needs of your organization.

Proxy vs. Mirror

A remote repository acts as a **proxy** not as a mirror. Artifacts are not pre-fetched to a remote repository cache. They are only fetched and stored *on demand* when requested by a client.

Therefore, a remote repository should not contain any artifacts in its cache immediately after creation. Artifacts will only be fetched to the cache once clients start working with the remote repository and issuing requests.

For full details on configuring remote repositories please refer to [Remote Repositories](#).

Virtual Repositories

A virtual repository (or "repository group") aggregates several repositories with the same package type under a common URL. The repository is virtual in that you can resolve and retrieve artifacts from it but you cannot deploy artifacts to it.

Generic Virtual Repositories

By their nature, Virtual Repositories whose package type has been specified as **Generic** can aggregate repositories of any type, however generic virtual repositories do not maintain any metadata

The Default Virtual Repository

By default, Artifactory uses a global virtual repository available at:

```
http://<host>:<port>/artifactory/repo
```

This repository contains all local and remote repositories.

Virtual Resolution Order

When an artifact is requested from a virtual repository, the order in which repositories are searched or resolved is local repositories first, then remote repository caches, and finally remote repositories themselves.

Within each of these, the order by which repositories are queried is determined by the order in which they are listed in the configuration as described in [General Resolution Order](#) below.

For a virtual repository, you can see the effective search and resolution order in the **Included Repositories** list view in the **Basic** settings tab. This is particularly helpful when nesting virtual repositories. For more details on configuring a virtual repository please refer to [Virtual Repositories](#).

General Resolution Order

You can set the order in which repositories of each type (local, remote and virtual) are searched and resolved by simply ordering them accordingly within the corresponding section of the **Configure Repositories** page. To set the order you need to add the repositories to the list of selected repositories in the order in which they should be searched to resolve artifacts.

The order in which repositories are searched is also affected by additional factors such as security privileges, include/exclude patterns and policies for handling snapshots and releases.

Common Settings

Overview

Several of the settings are common for local, remote and virtual repositories. These are found in the **Basic Settings** tab of the corresponding **New/Edit** screen under the **General** section. Additional settings may be found in the type-specific section according to the package types specified for the repository.

Common Basic Settings

New Local Repository

Basic Advanced Replications

Package Type *

Maven

Maven

Repository Key *

General

Repository Layout: maven-2-default

Public Description: Internal Description:

Include Patterns: Exclude Patterns:

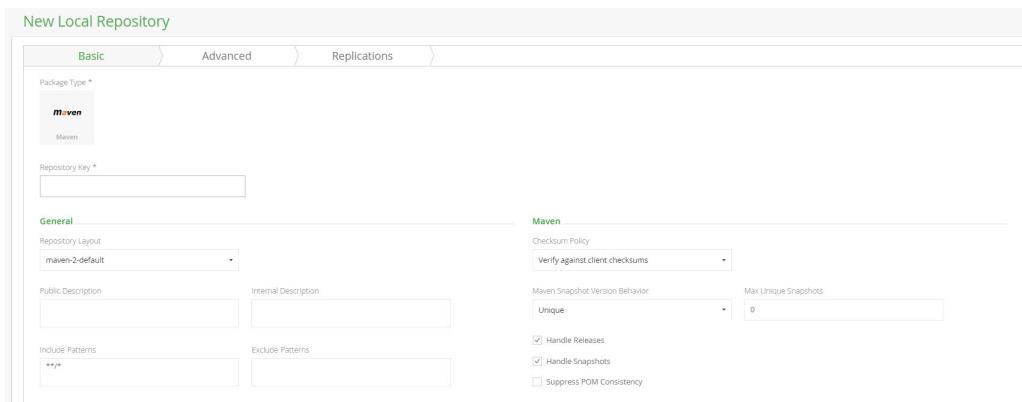
Maven

Checksum Policy: Verify against client checksums

Maven Snapshot Version Behavior: Unique

Max Unique Snapshots: 0

Handle Releases
 Handle Snapshots
 Suppress POM Consistency



Package Type	The Package Type must be specified when the repository is created, and once set, cannot be changed.
Repository Key	The Repository Key is a mandatory identifier for the repository and must be unique within an Artifactory instance. It cannot begin with a number or contain spaces or special characters. For local repositories we recommend using a "-local" suffix (e.g. "libs-release-local").
Repository Layout	Sets the layout that the repository should use for storing and identifying modules. Artifactory will suggest a layout that corresponds to the package type defined, and index packages uploaded and calculate metadata accordingly.
Public Description	A free text field that describes the content and purpose of the repository.
Internal Description	A free text field to add additional notes about the repository. These are only visible to the Artifactory administrator.

Include and Exclude Patterns

The **Include Patterns** and the **Exclude Patterns** fields provide a way to filter out specific repositories when trying to resolve the location of different artifacts.

In each field you can specify a comma-separated list of Ant-like patterns to filter in and filter out artifact queries. Filtering works by subtracting the excluded patterns (default is none) from the included patterns (default is all).

Example:

Consider that the Include Patterns and Exclude Patterns for a repository are as follows:

```
Include Patterns:  
org/apache/**,com/acme/**  
Exclude Patterns: com/acme/exp-project/**
```

In this case, Artifactory will search the repository for *org/apache/maven/parent/1/1.pom* and *com/acme/project-x/core/1.0/nit-1.0.jar* but not for **com/acme/exp-project/core/1.1/san-1.1.jar** because *com/acme/exp-project/** is specified as an Exclude pattern.

Page Contents

- Overview
- Common Basic Settings
 - Avoiding Security Risks with an Exclude Pattern
 - Avoiding Performance Issues with an Include Pattern
- Local and Remote Repositories

Avoiding Security Risks with an Exclude Pattern

Any proprietary artifacts you deploy to Artifactory are stored within local repositories so that they are available for secured and authorized internal use.

Anyone searching for one of your internal artifacts by name will extract it through Artifactory from the local repository.

However, consider what happens if a request for an internal artifact is inadvertently directed *outside* of the organization.

Two examples of how this could happen are:

- there is a simple typo in the requested artifact name
- the developer has requested a snapshot with a version number that does not exist.

In this case, since Artifactory does find the requested artifact in a local repository, it continues to search through the remote repositories defined in the system. Artifactory will, in fact, search through *all* the remote repositories defined in your system before returning "Not found".

This presents a security risk since any request made on a remote repository may be logged **exposing all details of the query including the full artifact name which may include sensitive business information**.

Best practices using an excludes pattern for remote repositories to avoid security risks

To avoid exposing sensitive business information as described above, we strongly recommend the following best practices:

- The list of remote repositories used in an organization should be managed under a single virtual repository to which all requests are directed
- All internal artifacts should be specified in the **Exclude Pattern** field of the virtual repository (or alternatively, of each remote repository) using wildcard characters to encapsulate the widest possible specification of internal artifacts.

Avoiding Performance Issues with an Include Pattern

In a typical scenario, Artifactory will reference large all-purpose repositories such as [JCenter](#) or Maven Central for resolving artifacts.

In addition, Artifactory may reference any number of additional repositories which may host a more specialized and specific set of artifacts.

If Artifactory receives a request for a deterministic set of artifacts (e.g. a specific version of an artifact), then it searches through the different repositories according to its resolution order until the artifact is found.

However, if Artifactory receives a request for a non-deterministic set of artifacts (e.g. all versions of `maven-metadata.xml`) then it must search through *all* of the repositories it references until it can provide a complete response.

In most cases, the majority of artifacts downloaded by an organization will come from one of the large all-purpose repositories, but in non-deterministic requests **performance is downgraded because Artifactory continues to search through all the specialized repositories** before it can return a response.

Best practices using an includes pattern for remote repositories to avoid needless and wasteful search

To avoid performing needless and wasteful search when responding to non-deterministic requests we strongly recommend that all specialized repositories be configured with an appropriate **Include Pattern** specifying only the set of artifacts that the organization might need.

In this case, non-deterministic requests for artifacts that are typically found in general purpose repositories will skip over the specialized repositories thereby improving performance.

Local and Remote Repositories

In addition to the settings above, Local and Remote repositories share the following settings in the type-specific section for relevant package types.

Maven Snapshot Version Behavior	
<input type="button" value="Unique"/>	
<input checked="" type="checkbox"/> Handle Releases	
<input checked="" type="checkbox"/> Handle Snapshots	
<input type="checkbox"/> Suppress POM Consistency	
Max Unique Snapshots	Specifies the maximum number of unique snapshots of the same artifact that should be stored. Once this number is reached and a new snapshot is uploaded, the oldest stored snapshot is removed automatically. A value of 0 (default) indicates that there is no limit on the number of unique snapshots.

Handle Releases	If set, Artifactory allows you to deploy release artifacts into this repository.
Handle Snapshots	If set, Artifactory allows you to deploy snapshot artifacts into this repository.

Local Repositories

Overview

To configure a local repository, in the **Admin** module, go to **Repositories | Local** and click it to display the **Edit Repository** screen.

Common Basic Settings

The following are fully described in the [Common Settings](#) page.

- Package Type
- Repository Key
- Repository Layout
- Public Description

[Internal Description](#)

- Includes and Excludes Pattern

Page Contents

- Overview
- Common Basic Settings
- Additional Basic Settings
 - Maven, Gradle, Ivy and SBT Repositories
 - Other Repository Types
- Advanced Settings
- Pre-defined Local Repositories

Additional Basic Settings

Repositories may have additional **Basic** settings depending on the **Package Type**.

Maven, Gradle, Ivy and SBT Repositories

Maven, Gradle, Ivy and SBT repositories share the same additional **Basic** settings.

Maven

Checksum Policy

Select Checksum Policy...

Maven Snapshot Version Behavior

Select Maven Snapshot Version Behavior...

Max Unique Snapshots

Handle Releases

Handle Snapshots

Suppress POM Consistency

Checksum Policy	<p>Checking the Checksum effectively verifies the integrity of a deployed resource. The Checksum Policy determines how Artifactory behaves when a client checksum for a deployed resource is missing or conflicts with the locally calculated checksum.</p> <p>There are two options:</p> <ol style="list-style-type: none">Verify against client checksums (default) - If a client has not sent a valid checksum for a deployed artifact then Artifactory will return a 404 (not found) error to a client trying to access that checksum. If the client has sent a checksum, but it conflicts with the one calculated on the server then Artifactory will return a 409 (conflict) error until a valid checksum is deployed.Trust server generated checksums - Artifactory will not verify checksums sent by clients and will trust the server's locally calculated checksums. An uploaded artifact is immediately available for use, but integrity might be compromised.
------------------------	---

**Maven
Snapshot
Version
Behavior**

Artifactory supports centralized control of how snapshots are deployed into a repository, regardless of end user-specific settings. This can be used to guarantee a standardized format for deployed snapshots within your organization. There are three options:

1. **Unique:** Uses a unique, time-based version number
2. **Nonunique:** Uses the default self-overriding naming pattern: *artifactoryID-version-SNAPSHOT.type*
3. **Deployer:** Uses the format sent by the deployer as is.

Maven 3 Only Supports Unique Snapshots

Maven 3 has dropped support for resolving and deploying non-unique snapshots. Therefore, if you have a snapshot repository using non-unique snapshots, we recommend that you change your Maven snapshot policy to 'Unique' and remove any previously deployed snapshots from this repository.

The unique snapshot name generated by the Maven client on deployment cannot help in identifying the source control changes from which the snapshot was built and has no relation to the time sources were checked out. Therefore, we recommend that the artifact itself should embed the revision/tag (as part of its name or internally) for clear and visible revision tracking. Artifactory allows you to tag artifacts with the revision number as part of its [Build Integration](#) support.

**Max Unique
Snapshots**

Specifies the maximum number of unique snapshots of the same artifact that should be stored. Once this number is reached and a new snapshot is uploaded, the oldest stored snapshot is removed automatically.

A value of 0 (default) indicates that there is no limit on the number of unique snapshots.

**Handle
Releases**

If set, Artifactory allows you to deploy release artifacts into this repository.

**Handle
Snapshots**

If set, Artifactory allows you to deploy snapshot artifacts into this repository.

Suppress POM Consistency

When deploying an artifact to a repository, Artifactory verifies that the value set for `groupId:artifactId:version` in the POM is consistent with the deployed path.

If there is a conflict between these then Artifactory will reject the deployment. You can disable this behavior by setting this checkbox.

Other Repository Types

For other type-specific repository configuration, please refer to the corresponding repository page under [Artifactory Pro](#).

Advanced Settings

New Local Repository

Basic Advanced Replications

Select Property Sets

Filter...

Available

Selected

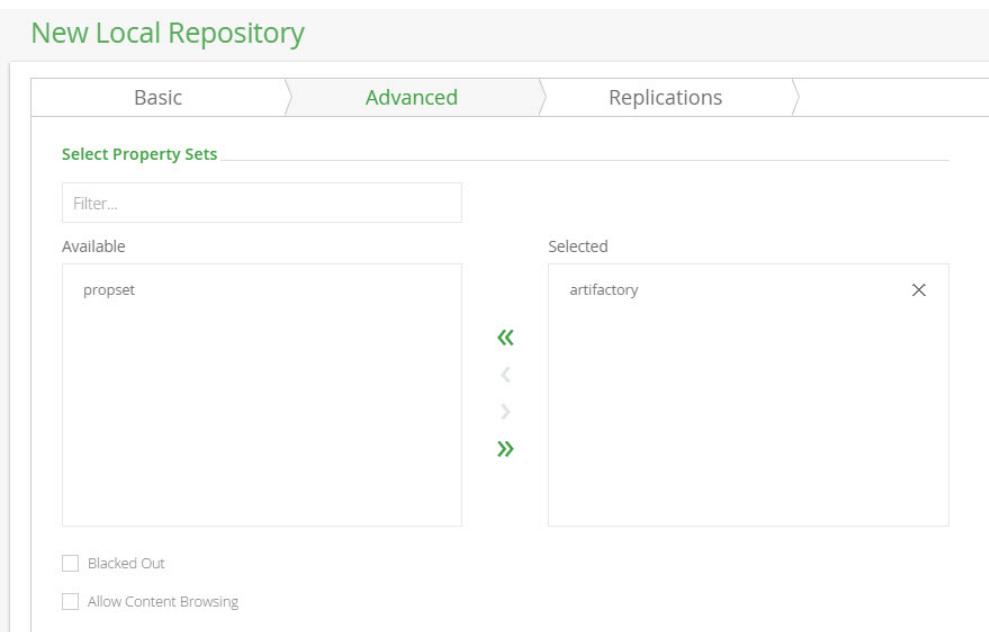
propset

artifactory

« < > »

Blocked Out

Allow Content Browsing



Select Property Sets

Defines the property sets that will be available for artifacts stored in this repository.

Blocked Out

If set, Artifactory ignores this repository when trying to resolve artifacts. The repository is also not available for download or deployment of artifacts.

<p>Allow Content Browsing</p>	<p>If set, allows you to view file contents (e.g., Javadoc browsing, HTML files) directly from Artifactory.</p> <div style="border: 2px solid red; padding: 10px; margin-top: 10px;"> <p>Security</p> <p>When content browsing is allowed we recommend strict content moderation to ensure that any uploaded content does not compromise security (for example, cross-site scripting attacks)</p> </div>
--------------------------------------	---

Pre-defined Local Repositories

Artifactory comes with a set of pre-defined local repositories, which reflect best practices in binary repository management as follows:

libs-release-local	Your code releases
libs-snapshot-local	Your code snapshots
ext-release-local	Manually deployed 3rd party libs (releases)
ext-snapshot-local	Manually deployed 3rd party libs (snapshots)
plugins-release-local	Your and 3rd party plugins (releases)
plugins-snapshot-local	Your and 3rd party plugins (snapshots)

Remote Repositories

Overview

To configure a remote repository, in the **Admin** module, go to **Repositories | Remote** and click it to display the **Edit Repository** screen.

Common Basic Settings

The following are fully described in the [Common Settings](#) page.

- Package Type
- Repository Key
- Public Description

[Internal Notes](#)

- Includes and Excludes Pattern

Additional Basic Settings

URL	The URL for the remote repository. Currently only HTTP and HTTPS URLs are supported.
Offline	If set, this repository will be considered offline and no attempts will be made to fetch artifacts from it. For more details, please refer to Single Repository Offline below.

Page Contents

- Overview
- Common Basic Settings
- Additional Basic Settings
- Type-Specific Basic Settings
 - Maven, Gradle, Ivy and SBT Repositories
 - Other Repository Types
- Handling Offline Scenarios
 - Single Repository Offline
 - Global Offline Mode

Read More

- Managing Proxies
- Advanced Settings

Type-Specific Basic Settings

Repositories may have additional **Basic** settings depending on the **Package Type**.

Maven, Gradle, Ivy and SBT Repositories

Maven Settings

Checksum Policy

Generate if absent	▼
--------------------	---

Max Unique Snapshots

0

- Eagerly Fetch Jars
- Suppress POM Consistency
- Eagerly Fetch Sources
- List Remote Folder Items
- Handle Releases
- Handle Snapshots

Checksum Policy	<p>Checking the Checksum effectively verifies the integrity of a deployed resource. The Checksum Policy determines how Artifactory behaves when a client checksum for a remote resource is missing or conflicts with the locally calculated checksum.</p> <p>There are four options:</p> <ol style="list-style-type: none"> Generate if absent (default): Artifactory attempts to retrieve the the remote checksum, If it is not found, Artifactory will automatically generate one and fetch the artifact. If the remote checksum does not match the locally calculated checksum, the artifact will not be cached and the download will fail. Fail: If the remote checksum does not mach the locally calculated checksum, or is not found, the artifact will not be cached and the download will fail. Ignore and generate: Artifactory ignores the remote checksum and only uses the locally generated one. As a result, remote artifact retrieval never fails, however integrity of the retrieved artifact may be compromised. Ignore and Pass-thru: Artifactory stores and passes through all remote checksums (even if they do not match the locally generated one). If a remote checksum is not found, Artifactory generates one locally. As a result, remote resource retrieval never fails, however integrity of the retrieved artifact may be compromised, and client side checksum validation (as performed by Maven, for example) will fail.
Max Unique Snapshots	Please refer to Max Unique Snapshots under Local Repositories.
Eagerly Fetch Jars	When set, if a POM is requested, Artifactory attempts to fetch the corresponding jar in the background. This will accelerate first access time to the jar when it is subsequently requested.
Suppress POM Consistency	<p>By default, Artifactory keeps your repositories healthy by refusing POMs with incorrect coordinates (path). If the <code>groupId:artifactId:version</code> information inside the POM does not match the deployed path, Artifactory rejects the deployment with a "409 Conflict" error.</p> <p>You can disable this behavior by setting the Suppress POM Consistency checkbox.</p>
Eagerly Fetch Sources	When set, if a binaries jar is requested, Artifactory attempts to fetch the corresponding source jar in the background. This will accelerate first access time to the source jar when it is subsequently requested.

List Remote Folder Items	When set, lists the items of remote folders in simple and list browsing. Required for dynamic resolution depending on remote folder content information, such as remote Ivy version lookups. The remote content is cached according to the value of Retrieval Cache Period .
Handle Releases	Please refer to Handle Releases under Local Repositories.
Handle Snapshots	Please refer to Handle Snapshots under Local Repositories.

Other Repository Types

For other type-specific repository configuration, please refer to the specific repository page under [Artifactory Pro](#).

Handling Offline Scenarios

Artifactory supports offline repository management at two levels:

- [Single Repository](#): One or more specific remote repositories need to be offline.
- [Global](#): The whole organization is disconnected from remote repositories

Single Repository Offline

If a remote repository goes offline for any reason, Artifactory can be configured to ignore it by setting the [Offline](#) checkbox. In this case, only artifacts from this repository that are already present in the cache are used. No further attempt will be made to fetch remote artifacts.

Global Offline Mode

This is common in organizations that require a separate, secured network and are disconnected from the rest of the world (for example, military or financial institutions).

In this case, remote repositories serve as caches only and do not proxy remote artifacts.

You can enable Global Offline Mode by setting the corresponding checkbox in the [Admin](#) tab under [Configuration | General](#).

General Settings

Server Name

Custom URL Base

File Upload Max Size *

Date Format *

Global Offline Mode

Managing Proxies

Overview

In corporate environments it is often required to go through a proxy server to access remote resources.

Artifactory supports several types of network proxy including NTLMv2.

Page Contents

- Overview
- Defining Proxies

Defining Proxies

To create a new proxy definition, in the **Admin** module go to **Configuration | Proxies** and click the "New" button.

Fields that are not required by the proxy may be left blank (for example, if you are not using authentication credentials or with an NTLM proxy you may leave the **Username** and **Password** fields blank).

New Proxy

Proxy Settings

Proxy Key *

System Default

Host *

Port *

User Name

Password

NT Host

NT Domain

Redirecting Proxy Target Hosts

Proxy Key	The unique ID of the proxy.
System Default	When set, this proxy will be the default proxy for new remote repositories and for internal HTTP requests. When you set this checkbox, Artifactory displays a confirmation message and offers to apply the proxy setting also to existing remote repository configurations.
Host	The name of the proxy host.
Port	The proxy port number.
Username	The proxy username when authentication credentials are required.
Password	The proxy password when authentication credentials are required.
NT Host	The computer name of the machine (the machine connecting to the NTLM proxy).

<i>NT Domain</i>	The proxy domain/realm name.
<i>Redirecting Proxy target Hosts</i>	An optional list of newline or comma separated host names to which this proxy may redirect requests. The credentials defined for the proxy are reused by requests redirected to all of these hosts.

Using proxies

Artifactory only accesses a remote repository through a proxy if one is selected in the **Network** section of the **Advanced** settings for a remote repository.

Whether this has been set manually, or by setting a **System Default** proxy as described [above](#), you can override this by removing the **Proxy** setting for any specific repository.

In this case, Artifactory will access the specific repository without going through a proxy.

Advanced Settings

Overview

The advanced settings for a remote repository configure network access behavior, cache management and several other parameters related to remote repository access.

To access the advanced settings, in the **Edit Remote Repository** screen select the **Advanced** tab.

Remote Credentials

Remote Credentials

Username	Password
<input type="text"/>	<input type="password"/>

<i>Username</i>	The username that should be used for HTTP authentication when accessing this remote proxy.
<i>Password</i>	The password that should be used for HTTP authentication when accessing this remote proxy.

Page Contents

- Overview
- Remote Credentials
- Network Settings
- Cache Settings
 - Zapping Caches
- Select Property Sets
- Other Settings

Network Settings

Network

Proxy

Local Address

Socket Timeout

Query Params

Lenient host authentication

Cookie management

Proxy	If your organization requires you to go through a proxy to access a remote repository, this parameter lets you select the corresponding Proxy Key . For more details on setting up proxies in Artifactory please refer to Managing Proxies .
Local Address	When working on multi-homed systems, this parameter lets you specify which specific interface (IP address) should be used to access the remote repository. This can be used to ensure that access to the remote repository is not blocked by firewalls or other organizational security systems.
Socket Timeout	The time that Artifactory waits (for both a socket and a connection) before giving up on an attempt to retrieve an artifact from a remote repository. Upon reaching the specified Socket Timeout Artifactory registers the repository as "assumed offline" for the period of time specified in Assumed Offline Limit .
Query Params	A custom set of parameters that should automatically be included in all HTTP requests to this remote repository. For example, <code>param1=value1&param2=value2&param3=value3</code>
Lenient Host Authentication	When set, allows using the repository credentials on any host to which the original request is redirected.

Cookie Management	When set, the repository will allow cookie management to work with servers that require them.
--------------------------	---

Using Oracle Maven Repository

To use [Oracle Maven Repository](#):

- Set your Oracle credentials in **Username** and **Password** of the **Remote Credentials**
- Set **Lenient Host Authentication**
- Set **Enable Cookie Management**.

Cache Settings

Artifactory stores artifacts retrieved from a remote repository in a local cache. The **Cache Settings** specify how to manage cached artifacts.

Caching Maven artifacts

Caching for Maven artifacts is only applicable to snapshots since it is assumed that releases never change.

Cache

Unused Artifacts Cleanup Period

0

Retrieval Cache Period

7200

Assumed Offline Period

300

Missed Retrieval Cache Period

43200

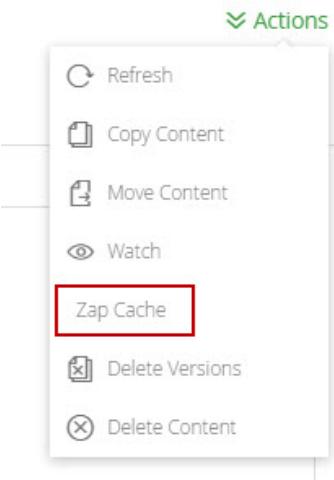
Unused Artifacts Cleanup Period	Many cached artifacts in Artifactory remote repository storage are actually unused by any current projects in the organization. This parameter specifies how long an unused artifact will be stored before it is removed. A value of 0 means that the artifact is stored indefinitely.
	<p>Removing unused artifacts</p> <p>Unused artifacts are not removed immediately upon reaching this timeout, but only on the next invocation of cleanup.</p> <p>For more details please refer Cleanup Unused Cached Artifacts in Regular Maintenance Operations.</p>

<i>Retrieval Cache Period</i>	Defines how long before Artifactory checks for a newer version of a requested artifact in a remote repository. A value of 0 means that Artifactory will always check for a newer version. This setting refers to artifacts that may change (e.g. snapshots, metadata files etc.). Note that most artifacts that are downloaded do not change (e.g. release versions), therefore this setting has no effect on them.
<i>Assumed Offline Period</i>	In case of a connection error, this parameter specifies how long Artifactory should wait before attempting an online check in order to reset the offline status. A value of 0 means that the repository is never assumed offline and Artifactory will always attempt to make the connection when demanded.
<i>Missed Retrieval Cache Period</i>	If a remote repository is missing a requested artifact, Artifactory will return a "404 Not found" error. This response is cached for the period of time specified by this parameter. During that time, Artifactory will not issue new requests for the same artifact. A value of 0 means that the response is not cached and Artifactory will always issue a new request when demanded.

Zapping Caches

"Zapping" a cache means forcing the Retrieval Cache Period and Missed Retrieval Cache Period to time out. To "zap" a cache, in the **Artifacts** module **Tree** browser,

Select the repository cache you wish to "zap" and click **Zap caches** in the right-click menu or **Actions** drop-down menu.



Select Property Sets

Defines the property sets that will be available for artifacts stored in this repository.

Other Settings

Others

- Blacked out
- Allow content browsing
- Store artifacts locally
- Synchronize properties
- Share configuration

Blacked out	If set, Artifactory ignores this repository when trying to resolve artifacts. The repository is also not available for download or deployment of artifacts.
Allow content browsing	When set, allows Artifactory users to browse the internal contents of archives (for example, browsing specific Javadoc files from within a Javadoc archive). Care When archive browsing is allowed, strict content moderation should be employed to ensure malicious users do not upload content that may compromise security (e.g. cross-site scripting attacks)
Store artifacts locally	When set, Artifactory artifacts from this repository will be cached locally. If not set, direct repository-to-client streaming is used. When might you use direct repository-to-client streaming? If your organization has multiple servers connected over a high speed LAN, you may have one instance of Artifactory caching data on a central storage facility with additional instances of Artifactory running on other servers. In this case, it makes sense for the additional instances of Artifactory to act as satellite pass-through servers rather than have them duplicate the cached data within their own environments.
Synchronize properties	When set, synchronizes properties of artifacts retrieved from a remote instance of Artifactory.

Smart Remote Repositories

Overview

A smart remote repository is a remote repository that proxies a repository from another instance of Artifactory. In addition to the usual benefits of [remote repositories](#), smart remote repositories offer several additional benefits:

Reported download statistics

Artifactory maintains download statistics for repositories so you are able to evaluate if artifacts are still being used and manage your cleanup policies. When you proxy a repository in another instance of Artifactory, and cache an artifact downloaded from the other instance, the distant Artifactory is not aware if users on your end continue to use the artifact (downloading it from your local cache), and may end up cleaning up the original artifact. An Artifactory Smart Remote Repository lets you notify the distant instance whenever a cached artifact is downloaded, so it can update an internal counter for remote downloads.

Download statistics may vary between Artifactory instances

Downloads are only reported through the proxy chain from the time this option is set, so the actual download statistics reported for an artifact may be different in the local Artifactory instance compared the numbers reported in the remote Artifactory instance.

Synchronized properties

When you proxy a repository in another instance of Artifactory and cache an artifact downloaded from it, you may not be aware of changes that may have been made to the original artifact's properties if they are done after you cache it. By synchronizing properties, any changes to artifact properties in the remote instance are propagated to your cached instance of the artifact.

Remote repository browsing

You can browse the contents of the repository in the remote Artifactory instance for all package types, even if none have been cached in your instance of Artifactory.

Delete indication

When viewing a cached artifact, Artifactory will indicate if the original artifact in the remote instance has been deleted. This gives you an opportunity to copy the artifact over from your remote repository cache to a local repository in case you need to maintain access to it.

Page Contents

- Overview
- Configuration

- Remote List Browsing

Configuration

To create a Smart Remote Repository and set the repository **URL** to point to a repository in another instance of Artifactory.

New Remote Repository

Once you have finished entering the URL and move to another field, Artifactory automatically detects that the remote URL is on another instance of Artifactory and displays a dialog where you can configure the behavior of your smart remote repository.

Note also that the package type icon is overlaid with an Artifactory logo to indicate a smart remote repository.

Artifactory Remote Repository Detected

Since the remote repository at the URL you specified is hosted by another instance of Artifactory, you may configure some additional capabilities for your repository:

Sync Statistics

If set, download statistics for the artifact at the remote Artifactory instance will be updated each time a cached item is downloaded from your repository.

Sync Properties

If set, properties for artifacts that have been cached in this repository will be updated if they are modified in the artifact hosted at the remote Artifactory instance.

List Remote Folder Items

If set, Artifactory lets you navigate the contents of the repository at the remote Artifactory instance, for all package types, even if the artifacts have not been cached in this repository.

• Delete Detection:

When viewing a cached item in this repository, you will see an indication if it has been deleted from the repository at the remote Artifactory instance.

These settings are also available in the Edit Repository screen.

OK

Report Statistics	If set, Artifactory will notify the remote instance whenever an artifact in the Smart Remote Repository is downloaded locally so the it can update its download counter. Note that if this option is not set, there may be a discrepancy between the number of artifacts reported to have been downloaded in the different Artifactory instances of the proxy chain.
Sync Properties	If set, properties for artifacts that have been cached in this repository will be updated if they are modified in the artifact hosted at the remote Artifactory instance. The trigger to synchronize the properties is download of the artifact from the remote repository cache of the local Artifactory instance.
List Remote Folder Items	If set, enables Remote List Browsing .
Delete Detection	This is not a setting, but a notification that for smart remote repositories, Artifactory displays an indication on cached items if they have been deleted from the corresponding repository in the remote Artifactory instance.

You can modify these settings at any time from the [Edit Repository](#) screen.

Edit smart-remote Repository

Basic Advanced Replications

Package Type * **Maven**

Repository Key * **smart-remote**

URL * <http://10.0.0.111:8080/artifactory/libs-release-local/>

Test

General

Repository Layout	Remote Layout Mapping
maven-2-default	maven-2-default
Public Description	Internal Description

Smart Remote Repository

Sync Statistics [?](#)

Sync Properties [?](#)

List Remote Folder Items [?](#)

Remote List Browsing

When **List Remote Folder Items** is checked for a repository, Artifactory lets you navigate the contents of the repository at the remote Artifactory instance, for all package types, even if the artifacts have not been cached in the repository defined in your instance of Artifactory.

Virtual Repositories

Overview

To simplify access to different repositories, Artifactory allows you to define a virtual repository which is a collection of local, remote and other virtual repositories accessed through a single logical URL.

A virtual repository hides the access details of the underlying repositories letting users work with a single, well-known URL. The underlying participating repositories and their access rules may be changed without requiring any client-side changes.

Page Contents

- [Overview](#)
- [Basic Settings](#)
 - [Nesting](#)
 - [Using Includes and Excludes Patterns](#)
- [Deploying to a Virtual Repository](#)
- [Advanced Settings](#)
 - [Maven, Gradle, Ivy and SBT Repositories](#)
- [Pre-defined Repositories](#)

Basic Settings

The following are fully described in the [Common Settings](#) page.

- [Package Type](#)
- [Repository Key](#)
- [Repository Layout](#)
- [Public Description](#)
- [Internal Description](#)
- [Includes and Excludes Pattern](#)

New Virtual Repository

Basic

Advanced

Package Type *



Maven

Repository Key *

General

Repository Layout

Public Description

Internal Description

Include Pattern

Exclude Pattern

In addition, in the **Repositories** section of the **Basic** settings screen you select the **Available Repositories** you want to include in the new virtual repository and move them to the **Selected Repositories** list.

This list can be re-ordered by dragging and dropping within the **Selected Repositories** list.

Repositories

Filter...

Available Repositories

- libs-release
- libs-snapshot
- p2
- p2-virtual
- plugins-release
- plugins-snapshot
- test-p2
- vri-terra-cotta

Selected Repositories

- libs-release-local
- ext-snapshot-local
- remote-repos



Included Repositories

- libs-release-local
- ext-snapshot-local
- java.net.m1
- repo1
- gradle-libs

The **Included Repositories** section displays the effective list of actual repositories included in this virtual repository. If any of the available

repositories you have selected are themselves virtual repositories, then the **Included Repositories** section will display the local and remote repositories included within them. The **Included Repository** list is automatically updated in case any of the nested virtual repositories change.

The search/resolution order when requesting artifacts from a virtual repository is always:

1. Local repositories
2. Remote repository caches
3. Remote repositories themselves.

The order within these categories is controlled by the order they are presented in the **Selected Repositories** list.

Nesting

Nesting is a unique feature in Artifactory and facilitates more flexibility in using virtual repositories.

You should take care not to create an "infinite loop" of nested repositories. Artifactory analyzes the internal composition of virtual repositories and will issue a warning if the virtual repository can not be resolved due to invalid nesting.

Using Includes and Excludes Patterns

The ability to define and **Includes Pattern** and an **Excludes Pattern** for virtual repositories (especially when nesting is used) provides a powerful tool you can use to manage artifact requests in your organization.

For example, your organization may have its own artifacts which are hosted both internally in a local repository, but also in a remote repository. For optimal performance, you would want these artifacts to be accessed from the local repository rather than from the remote one. To enforce this policy, you can define a virtual repository called "remote-repos" which includes the full set of remote repositories accessed by your organization, and then specify an Excludes Pattern with your organization's groupId. in this way, any attempt to access your internal artifact from a remote repository would be rejected.

Consider another example in which you wish to define a virtual repository for your developers, however you wish to keep certain artifacts hidden from them. This could be achieved by defining an **Excludes Pattern** based on groupId, source or version.

Deploying to a Virtual Repository

From version 4.2, Artifactory supports deploying artifacts to a virtual repository. For example you can now use `docker push`, `npm publish`, `NuGet push`, `gem push` Artifactory's REST API and more to deploy packages to a virtual repository.

For more details, please refer to [Deploying Artifacts](#).

Advanced Settings

New Virtual Repository

Basic

Advanced

Artifactory Requests Can Retrieve Remote Artifacts

Cleanup Repository References in POMs

Discard active references

Key-Pair

No key-pairs are currently configured. You can add new key-pairs [here](#).

Artifactory Requests Can Retrieve Remote Artifacts

An Artifactory instance may request artifacts from a virtual repository in another Artifactory instance. This checkbox specifies whether the virtual repository should search through remote repositories when trying to resolve an artifact requested by another Artifactory instance. For example, you can use this feature when Artifactory is deployed in a mesh (grid) architecture, and you do not want all remote instances of Artifactory to act as proxies for other Artifactory instances.

Maven, Gradle, Ivy and SBT Repositories

In addition to the above checkbox, these repository types offer the following **Advanced** settings:

Cleanup Repository References in POMs

Public POMs may include direct references to external repositories. If either of the below code samples are present in the POM, Maven dynamically adds an external repository URL to the build which circumvents Artifactory.

```
<project><repositories><repository>  
or  
<project><pluginRepositories><pluginRepository>
```

A client side solution for this is to use `mirrorOf`. For details please refer to [Additional "Mirror-any" Setup](#).

This setting gives you the ability to ensure Artifactory is the sole provider of Artifacts in your system by automatically cleaning up the POM file. The three values available for this setting are:

Discard Active References	Removes repository elements that are declared directly under project or under a profile in the same POM that is <code>activeBy Default</code>
Discard Any References	Removes all repository elements regardless of whether they are included in an active profile or not
Nothing	Does not remove any repository elements declared in the POM

Key Pair

A named key-pair to use for automatically signing artifacts.

Please refer to [WebStart and Jar Signing](#).

Pre-defined Repositories

Artifactory comes with a set of pre-defined virtual repositories, which reflect binary management best practices as follows.

remote-repos	Aggregation of all remote repositories
lib-releases	libs-releases-local, ext-releases and remote-repos
plugins-releases	plugins-releases-local, ext-releases and remote-repos

libs-snapshots	libs-snapshots-local, ext-snapshots-local, remote-repos
plugins-snapshots	plugins-snapshots-local, ext-snapshots-local, remote-repos

Configuring Security

Overview

Artifactory's security model offers protection at several levels. It allows you to do the following:

- Assign role-based or user-based permissions to areas in your repositories (called Permission Targets)
- Allow sub-administrators for Permission Targets
- Configure LDAP out-of-the-box
- Prevent clear text in Maven's `settings.xml` file
- Inspect security definitions for a single artifact or folder and more.

Artifactory's security is based on Spring Security and can be extended and customized.

This section explains the strong security aspects and controls offered by Artifactory:

Page Contents

- Overview
- General Configuration
 - Allow Anonymous Access
 - Prevent Anonymous Access to Build Related Info
 - Hide Existence of Unauthorized Resources
 - Managing API Keys
 - Password Encryption Policy

Read More

- Managing Users
- Managing Permissions
- Centrally Secure Passwords
- Master Key Encryption
- Managing Security with LDAP
- Managing Security with Active Directory
- Access Log

General Configuration

Artifactory provides several system-wide settings to control access to different resources. These are found under **Security | General** in the **Administration** tab.

Security General Configuration

Security General Settings

- Allow Anonymous Access
- Prevent Anonymous Access to Build Related Info
- Hide Existence of Unauthorized Resources

Password Encryption Policy

SUPPORTED

Allow Anonymous Access

Artifactory provides a detailed and flexible permission-based system to control users' access to different features and artifacts.

However, Artifactory also supports the concept of "Anonymous Access" which controls the features and artifacts available to a user who has not logged in.

This is done through an "Anonymous User" which comes built-in to Artifactory with a default set of permissions.

Anonymous access may be switched on (default) or off using the **Allow Anonymous Access** setting under **Security General Settings** in the **Administration** module.

You can modify the set of permissions assigned to the "Anonymous User" just like you would for any other user, and this requires that **Allow Anonymous Access** is enabled.

Prevent Anonymous Access to Build Related Info

This setting gives you more control over anonymous access, and allows you to prevent anonymous users from accessing the **Build** module where all information related to builds is found, even when anonymous access is enabled.

Hide Existence of Unauthorized Resources

When a user tries to access a resource for which he is not authorized, Artifactory default behavior is to indicate that the resource exists but is protected.

For example, an anonymous request will result in a request for authentication (401), and a request by an unauthorized authenticated user will simply be denied.

You can configure Artifactory to return a 404 - Not Found response in these cases by setting **Hide Existence of Unauthorized Resources** under **Security | General** in the **Administration** module.

Managing API Keys

As an admin user, you can revoke any specific user's current API key, or revoke all the API keys currently defined in the system under **Security | General** in the **Administration** module.

API Keys Management

Revoke User API Key

User Name

 Revoke

 **Revoke API Keys For All Users**

To remove a specific user's API key, enter the username in the corresponding field and click "Revoke".

To revoke all API keys in the system, click "Remove API Keys for All Users!"

Once you revoke a user's API key, any REST API calls using that API key will no longer work. The user will have to create new API key and update any scripts that use it.

Password Encryption Policy

Artifactory provides a unique solution to support encrypted passwords through the **Password Encryption Policy** setting as follows:

Supported	Artifactory can receive requests with an encrypted password but will also accept requests with a non-encrypted password (default)
Required	Artifactory requires an encrypted password for every authenticated request
Unsupported	Artifactory will reject requests with encrypted password

For more details on why Artifactory allows you to enforce password encryption please refer to Centrally Secure Passwords.

Managing Users

Overview

You can manage access to repositories by defining users, assigning them to groups and setting up roles and permissions which can be applied to both users and groups.

Creating and Editing Users

To manage users who can access repositories in your system, in the **Admin** module, select **Security | Users**.

Users Management						
Filter by name or email		Actions				
Name	Email	Realm	Related Groups	Related Permissions	Admin	Last Login
adam	[REDACTED]	internal	-	-	false	2015/05/20
admin	[REDACTED]	internal	-	-	true	2015/07/14
anonymous	[REDACTED]		1 readers	2 Anything, Any Remote	false	
elig	[REDACTED]	ldap Check external status	2 deployers, readers	2 testpermission, Anything	false	2012/08/08
elitest	[REDACTED]		2 readers, managers-il	2 testpermission, Anything	true	
jbaruch	[REDACTED]		3 deployers, readers, qa-il	2 testpermission, Anything	true	
jenkins	[REDACTED]	internal	1 readers	2 Anything, CI	false	
matank	[REDACTED]		1 deployers	2 testpermission, Anything	false	
royz	[REDACTED]	internal	-	-	true	2015/07/14
yossi	[REDACTED]	internal	1 deployers	2 testpermission, Anything	true	2015/07/12

Page Contents

- Overview
- Creating and Editing Users
 - Administrator Users
 - The Anonymous User
- Creating and Editing Groups
 - Default Groups
- User Management
 - Setting Groups for a User
 - Setting Users for a Group
- Recreating the Default Admin User
 - Obtaining a Security Configuration File
 - Resetting the Admin Password
 - Replacing the Security Configuration File
- Disabling Remember Me at Login

Create a new user by clicking **New** at the top of the users table.

Only administrators can create users

To create users you must be an administrator (unless you are using external authentication such as LDAP)

Add New User

User Name *

Email Address *

Password *

Retype Password *

Can Update Profile

Admin

Disable Internal Password

Cancel Save

In the **New User** (or **Edit User**) dialog you can set the **User Name**, **Email Address** and **Password** for the user as well as the following parameters:

Can Update Profile	When set, this user can update his profile details (except for the password). Only an administrator can update the password. There may be cases in which you want to leave this unset to prevent users from updating their profile. For example, a departmental user with a single password shared between all department members.
Admin	When set, this user is an administrator with all the ensuing privileges. For more details please refer to Administrator Users .
Disable Internal Password	When set, disables the fallback of using an internal password when external authentication (such as LDAP) is enabled.

Artifactory stores passwords as hashes or encrypted hashes.

Administrator Users

An administrator user is to Artifactory as a "root" is to UNIX systems. Administrators are not subject to any security restrictions, and we therefore recommend to create a minimum number of administrators in your system.

You can control which permission-targets administrators have access to thereby assigning responsibility for a specific repository path. For details please refer to [Managing Permissions](#).

The Default Admin Account

The default user name and password for the built-in administrator user are: **admin/password**. You should change the password after first log in. If you forget the admin account password, you can recover it. Please refer to [Recreating the Default Admin User](#).

The Anonymous User

Artifactory supports the concept of anonymous users and installs with a pre-defined `anonymous` user to which you can assign permissions just like for any other user.

Anonymous access can be controlled under [Security General Configuration](#). Set **Allow Anonymous Access** to activate the anonymous user. The anonymous user must be activated before you can fine tune its permissions.

When anonymous access is activated, anonymous requests can download cached artifacts and populate caches, regardless of other permissions defined.

Creating and Editing Groups

A group represents a role in Artifactory and is used with RBAC (Role-Based Access Control) rules.

To manage groups, in the **Admin** module select **Security | Groups**.

Groups Management				
<input type="text" value="Filter by Group Name"/>				
Group Name	Related Permissions	External	Auto Join	New
readers	1 Anything	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
jfrogusers	-	<input type="checkbox"/>	<input type="checkbox"/>	
deployers	2 testpermission, Anything	<input type="checkbox"/>	<input type="checkbox"/>	
dnsupdateproxy	-	<input type="checkbox"/>	<input type="checkbox"/>	
pre-windows 2000 compatible access	-	<input type="checkbox"/>	<input type="checkbox"/>	
hyper-v administrators	-	<input type="checkbox"/>	<input type="checkbox"/>	
qa-il	1 testpermission	<input type="checkbox"/>	<input type="checkbox"/>	
managers-il	-	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

Create a new group by clicking **New** at the top of the groups table.

Add New Group

Group Name *

Description

Automatically Join New Users to this Group

You must assign a unique name to each group and can add an optional description

Default Groups

When creating (or editing) a group you can set **Automatically Join New Users to this Group**.

When this parameter is set, any new users defined in the system are automatically assigned to this group.

This is particularly useful if users are defined automatically and you want them to be assigned to certain groups. For example, when using external authentication such as LDAP, users are automatically created on successful login and you can use this parameter to assign these users to particular groups by default.

User Management

There are two ways to manage users' assignment to groups:

1. Setting the groups for a user
2. Setting the users for a group

Setting permissions

In both cases, you can assign corresponding permissions to the user or group respectively on the same screen. For more details please refer to [Managing Permissions](#).

Setting Groups for a User

You can assign and remove a user from groups by editing the user's details.

In the **Admin** module, under **Security | Users**, from the list of users, select the user you wish to assign to or remove from groups.

In the **User Groups** section of the form, you can set which groups the user should be assigned to.

The screenshot shows the 'Edit jbaruch' user profile page. At the top, there are fields for User Name (jbaruch), Email Address (jbaruch@jfrog.com), Password, Retype Password, and Password Strength. Below these are checkboxes for Can Update Profile (checked), Admin (checked), and Disable Internal Password (unchecked). The 'User Groups' section contains a list of available groups on the left ('jfrogiusers', 'dnsupdateproxy', 'pre-windows 2000 compatible access', 'hyper-v administrators', 'managers-il') and a list of assigned groups on the right ('deployers', 'readers', 'qa-il'). Between the two lists are four arrows: '<', '<', '>', and '>'. A 'Clear All' button is located at the bottom right of the group lists.

Setting Users for a Group

You can assign and remove a users from a group by editing the group's details.

In the **Admin** module, under **Security | Groups**, from the list of groups, select the group you wish modify.

In the **Users** section of the form, you can set which users should be assigned to the group.

Recreating the Default Admin User

If you are unable to obtain administrator access, you will need to recreate the default administrator user in order to be able to manage users of your system using the following steps::

1. Obtain a security configuration file
2. Reset the admin password
3. Correctly place the security configuration file

Obtaining a Security Configuration File

The security configuration file is called `security.xml`.

If your instance of Artifactory is configured to perform [backups](#) automatically, you can find it in the root backup folder.

If Artifactory is **not** configured to perform backups automatically you need to force creation of a new `security.xml` file as follows:

- Remove the file `$ARTIFACTORY_HOME/data/.deleteForSecurityMarker` and restart Artifactory .
- Make sure that Artifactory completes the startup sequence without interruption
- The security configuration file with the current time stamp can be found in `$ARTIFACTORY_HOME/etc/security.<time stamp>.xml`

Resetting the Admin Password

Reset the admin password as follows:

- Make a copy of the `security.xml` file you obtained in the previous section
- In the copy, edit the admin's password field and enter the password hash code (according to your version of Artifactory) as follows:

Admin password hash code

```
For version 3.x: <password>1f70548d73baca61aab8660733c7de81</password>
For version 2.x: <password>5f4dcc3b5aa765d61d8327deb882cf99</password>
```

Replacing the Security Configuration File

- Place the modified security configuration file under `$ARTIFACTORY_HOME/etc`

- Rename the file to `security.import.xml`
- Restart Artifactory

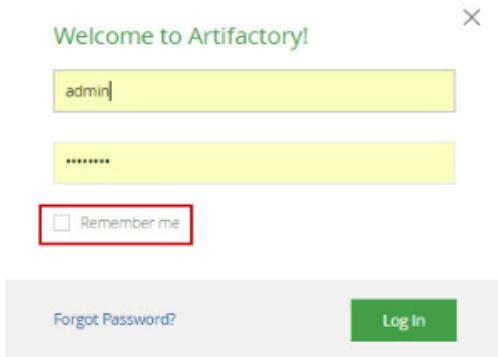
Once Artifactory has completed its startup sequence you will be able to login using the default admin user credentials:

User	admin
Password	password

Disabling Remember Me at Login

The Artifactory login screen includes a **Remember Me** checkbox. If the user sets this checkbox when logging in, Artifactory will store a cookie in the browser for a period of 2 weeks allowing the user to be logged in automatically when starting up Artifactory.

Once the cookie expires, the user will have to log in again.



An Artifactory administrator can disable this feature and force all users to enter their credentials at every login. To do so simply add the following property to `$ARTIFACTORY_HOME/etc/artifactory.system.properties` and restart Artifactory:

```
artifactory.security.disableRememberMe=true
```

Managing Permissions

Overview

Artifactory allows you to control access to repositories via **Permission Targets**.

A permission target is comprised of a set of physical repositories (i.e. local or remote repositories - but not virtual ones), and a set of users or groups with a corresponding set of permissions defining how they can access the specified repositories. Include and Exclude patterns give you finer control over access to a specific set of artifacts within the repositories of the permission target.

For example, you can create a permission target that allows user "Builder" and group "Deployers" to read from and deploy artifacts to the `libs-releases` repository. Using the Include Pattern and Exclude Pattern settings you could implement finer control over specific artifacts within that repository if so desired.

To manage permissions, in the **Admin** module go to **Security | Permissions**.

Permissions Management			
Permission Target Name	Repositories	Groups	Number of Users
Any Remote	9 download.eclipse.org,gradle-libs,java.net.java.net.m...		1 anonymous
jbaruch	25 RubyGems-local,dfsfq-ext-release-local,ext-snapsh...		
testpermission	23 RubyGems-local,download.eclipse.org-ext-release-...	2 qa-il,deployers	1 elitest
Anything	25 RubyGems-local,dfsfq-ext-release-local,ext-snapsh...	2 deployers,readers	1 anonymous
CI	16 RubyGems-local,dfsfq-ext-release-local,ext-snapsh...		1 jenkins

Page Contents

- Overview
- Creating a Permission Target
 - Permission Target Managers
 - Preventing Overwriting Deployments
- Examining Permissions
 - By Repository
 - By User or Group

Creating a Permission Target

To create a **Permission Target**, in the **Permissions Management** page click "New" to display the **New Permission Target** dialog.

New Permissions Target

Name *

Repositories Groups Users

Include Pattern: `**`

Exclude Pattern: `**/*-.sources.*`

filter...

Available Repositories	Selected Repositories
download.eclipse.org	X
gradle-libs	X
java.net	X
java.net.m1	X
localhost	X
npm	X
nuget	X
repo1	X

Any Local Repository

Any Remote Repository

Cancel < Back Next > **Save & Finish**

Name

You must provide a unique name for each **Permission Target**.

Repositories

Select the repositories to which this **Permission Target** applies. You can use the **Any Local Repository** or **Any Remote Repository** check boxes as a convenience.

Include and Exclude Patterns

Using an "Ant-like" script, you can specify any number of Include or Exclude Patterns as a comma-separated list in the corresponding entry field.

In the example above, source files have been excluded from the **Permission Target** named "Not sources" using the appropriate **Exclude Pattern**.

User and Group Permissions

Using the corresponding tabs, you can set the permissions granted to a user or a group. Double-click the user or group you want to modify to add it to the list of **Principals**, and then check the permissions you wish to grant.

New Permissions Target

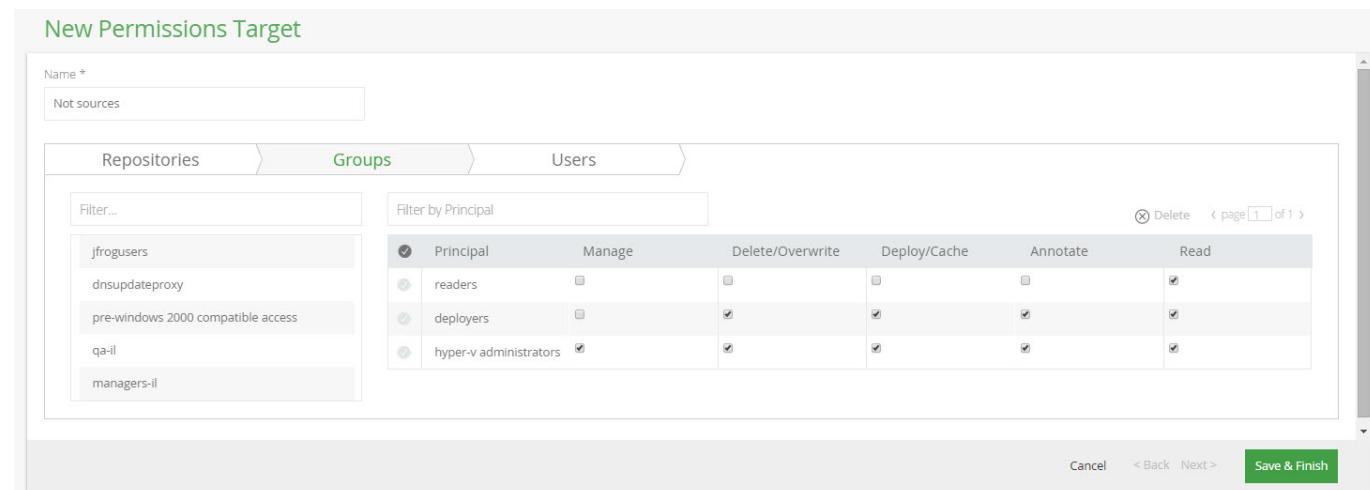
Name * Not sources

Repositories > Groups > Users

Filter... Filter by Principal

Principal	Manage	Delete/Overwrite	Deploy/Cache	Annotate	Read
readers	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
deployers	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
hyper-v administrators	<input checked="" type="checkbox"/>				

Cancel < Back Next > Save & Finish



The available permissions are as follows:

Manage	Allows changing the permission settings for other users on this permission target
Delete	Allows deletion or overwriting of artifacts
Deploy	Allows deploying artifacts and deploying to caches (i.e. populating caches with remote artifacts)
Annotate	Allows annotating artifacts and folders with metadata and properties
Read	Allows reading and downloading of artifacts

Multiple Permissions

Permissions are additive and must be explicitly granted. If a checkbox is not set for a user, then that user does not have the corresponding permission.

Permission Target Managers

By assigning the **Manage** permission to a user, you may designate them as the "Permission Target Manager". These users may assign and modify permissions granted to other users and groups for this **Permission Target**. In the Artifactory UI these users have access to the specific

users they are allowed to manage. This can be useful on a multi-team site since you can delegate the responsibility of managing specific repositories to different team members.

Preventing Overwriting Deployments

You can prevent a user or group from overwriting a deployed release or unique snapshot by not granting the **Delete** permission. Non-unique snapshots can always be overwritten (provided the **Deploy** permission is granted).

Examining Permissions

You can examine permissions in the context of repositories, users or groups.

By Repository

In the **Artifacts** module, select repository you want to view in the **Artifact Repository Browser** and then select the **Effective Permissions** tab to see the permissions granted to users or groups for this repository.

The screenshot shows the 'Artifact Repository Browser' interface. On the left, a tree view lists repositories: RubyGems-local, ext-release-local (selected), ext-snapshot-local, ivy-ivy, ivy-local, libs-release-bintray, libs-release-local, libs-snapshot-local, matan-playground, npm-local, nuget-local, p2-local, plugins-release-local, plugins-snapshot-local, poms-repo, rpm-local, and zanbel-maven-tester. The right side shows the 'ext-release-local' repository details. The 'Effective Permission' tab is selected, displaying a table of permissions for various principals:

Principal	Type	Delete/Overwrite	Deploy/Cache	Annotate	Read
admin	user	✓	✓	✓	✓
anonymous	user	✓	✓	✓	✓
deployers	group	✓	✓	✓	✓
elij	user	✓	✓	✓	✓
elitest	user	✓	✓	✓	✓
jbaruch	user	✓	✓	✓	✓
jenkins	user	✓	✓	✓	✓
matank	user	✓	✓	✓	✓
qa-il	group	✓	✓	✓	✓
readers	group				✓
royz	user	✓	✓	✓	✓
yossi	user	✓	✓	✓	✓

By User or Group

For any user or Group, you can view the list of Permission Targets that it is associated with (whether directly or through membership in a group).

For users, In the **Admin** module, under **Security | Users**, select the user you wish to examine. The **User Permissions** are displayed at the bottom of the user's page.

The screenshot shows the 'User Permissions' table. It has columns: Permission Target, Inherited From, Repositories, Manage, Delete/Overwrite, Deploy/Cache, Annotate, and Read. Two rows are shown:

Permission Target	Inherited From	Repositories	Manage	Delete/Overwrite	Deploy/Cache	Annotate	Read
testpermission	deployers	23 RubyGems-local, d...	true	true	true	true	true
Anything	readers	1 ANY	false	false	false	false	true

You can similarly view Group permissions in the **Admin** module under **Security | Groups**.

Centrally Secure Passwords

Overview

Some tools use cleartext passwords, which can pose a security risk. The security risk is even greater if you use LDAP or other external authentication, since you expose your SSO password in cleartext and that password is likely to be used for other services, not just Artifactory.

For example, Maven uses cleartext passwords in the `settings.xml` file by default.

Using Maven's built-in support for encrypted passwords and generating passwords on the client side does not overcome the security risks for the following reasons:

1. The login password is decrypted on the client side and ends up as cleartext in memory, and then transmitted over the wire (unless forcing SSL too).
2. The master password used for decryption is stored in clear text on the file system.
3. Password encryption is left to the good will of the end-user and there is no way to centrally mandate it.

Artifactory provides a unique solution to this problem by generating encrypted passwords for users based on secret keys stored in Artifactory. You can ensure users' shared passwords are never stored or transmitted as clear text.

Page Contents

- Overview
- Using Your Secure Password

You can set a central policy for using or accepting encrypted passwords in the **Admin** module under **Security | General** by setting the **Password Encryption Policy** field.

Security General Configuration

<input checked="" type="checkbox"/> Allow Anonymous Access	<input checked="" type="checkbox"/> Prevent Anonymous Access to Build Related Info
<input type="checkbox"/> Hide Existence of Unauthorized Resources	
Password Encryption Policy	
SUPPORTED	

The behavior according to the **Password Encryption Policy** setting is as follows:

Supported	Artifactory can receive requests with encrypted password (default).
Required	Artifactory requires an encrypted password for every authenticated request.
Unsupported	Artifactory will reject requests with encrypted password.

Using Your Secure Password

To secure your password:

1. Open your profile page (click on your login name on the upper-right corner), type-in your password in the **Current Password** field and click **Unlock**.
2. Your encrypted password is displayed in the **API Key** field. Click the corresponding icons next to this field to view the API Key openly or copy it to the clipboard.

User Profile: admin

Current Password

.....

Unlock

Insert the password and press the Unlock button to edit the profile.

Personal Settings

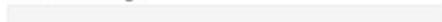
API Key

.....



New Password

Password Strength



Retype Password

Email Address*

.....@.....com

Different encryption mechanisms

The encryption mechanisms of the Oracle and IBM JDKs are not identical. Switching from one to another will make your encrypted password obsolete

IBM JDK Encryption Restrictions

Some of the IBM JRE/JDK are shipped with a restriction on the encryption key size (mostly for countries outside the US); This restriction can be officially removed by downloading unrestricted policy files from IBM and overriding the existing ones:

1. Register and download the unrestricted JCE policy files from the IBM website.
2. Select the correct zip that matches your JAVA version.
3. The downloaded zip file contains 2 jar files - `local_policy.jar` and `US_export_policy.jar`. Backup the existing files in `$IBM_JDK_HOME/jre/lib/security` and extract the jars from the zip file to this location
4. Restart Artifactory

Master Key Encryption

Overview

The [global Artifactory configuration file](#) stores the various passwords that are needed in order to interface with your organizations systems and external repositories. For example, Artifactory may need your LDAP server password.

In order to keep these passwords secure, you can choose to store them in an encrypted format. In this case, Artifactory will generate a **Master Encryption Key** which will be used to encrypt these passwords for storage and display, and to decrypt them when you need to access the corresponding resources.

IBM JDK Encryption Restrictions

Users of the IBM JDK should read about IBM JDK encryption restrictions described in [Using Your Secure Password](#).

Page Contents

- Overview

- Encrypting Passwords
- Decrypting Passwords
- Exporting and Importing the Master Key
 - Master Key File Location

Encrypting Passwords

When Master Key Encryption is activated all current passwords in the global configuration file are encrypted, and any new passwords, or updates will also be encrypted automatically.

The default Artifactory configuration does not encrypt passwords. An Artifactory administrator can activate encryption by either using the [REST API](#), or through the Artifactory UI in the **Admin** module under **Security | General**.

Security General Configuration

The screenshot shows the 'Security General Configuration' page. It includes sections for 'Allow Anonymous Access' (checkbox checked), 'Prevent Anonymous Access to Build Related Info' (checkbox uncheckable), 'Hide Existence of Unauthorized Resources' (checkbox uncheckable), 'Password Encryption Policy' (dropdown set to 'SUPPORTED'), and a 'Configuration Passwords Encryption' section where it says 'All passwords in your configuration are currently visible in plain text.' A green 'Encrypt' button is present. At the bottom right are 'Cancel' and 'Save' buttons.

Once Master Key Encryption is activated, subsequent activation using the REST API are ignored.

Decrypting Passwords

An Artifactory administrator can deactivate encryption, and decrypt any currently encrypted passwords by either using the [REST API](#), or through the Artifactory UI in the **Admin** module under **Security | General**.

When you select **Decrypt**, all passwords in the global configuration file are decrypted, the configuration is reloaded and the current Master Key is removed.

Any new passwords entered, or passwords updated will not be encrypted.

Exporting and Importing the Master Key

If the Master Key is in its default location under the `$ARTIFACTORY_HOME/etc` folder, it will be exported during a [system backup](#) or full [system export](#).

Correspondingly, if a Master Key was exported, and you now perform a full system import, the key will be copied to the default location and the Master Key Encryption feature will be activated. i.e. the Master Key will be used to encrypt and decrypt the imported configuration.

Master Key File Location

By default, the Master Key file is located under `$ARTIFACTORY_HOME/etc/security/artifactory.key`.

You may wish to exercise more stringent security so that the master key file is in a more secure location.

In this case you can change the file location by modifying the `artifactory.security.master.key` property in the `artifactory.system.properties` file.

For example,

Modifying the default master key file location

```
artifactory.security.master.key=<other location>/artifactory.key
```

If you use a partial path, then it will be interpreted as relative to the `$ARTIFACTORY_HOME/etc` folder.

If you change the Master Key file location, it will not be exported automatically. It is up to the administrator to back it up along with the export, and restore it manually on an import.

Managing Security with LDAP

Introduction

Artifactory supports authenticating users against an LDAP server out-of-the-box.

When LDAP authentication is active, Artifactory first attempts to authenticate the user against the LDAP server. If LDAP authentication fails, Artifactory tries to authenticate via its internal database.

For every LDAP authenticated user Artifactory creates a new user in the internal database (provided the user does not already exist), and automatically assigns that user to the default groups.

Managing Permissions for LDAP Groups

Artifactory can synchronize your LDAP groups and leverage your existing organizational structure when managing group-based permissions. LDAP groups in Artifactory use super-fast caching and support Static, Dynamic and Hierarchical mapping strategies.

Powerful management is accomplished with multiple, switchable LDAP settings and visual feedback about the up-to-date status of groups and users coming from LDAP.

The LDAP Groups feature is bundled as one of the Add-ons included in Artifactory Pro.

For full details on how to synchronize your LDAP Groups with Artifactory, please refer to [LDAP Groups](#).

Using Active Directory?

If you are using Active Directory to authenticate users, please refer to [Managing Security with Active Directory](#).

Page Contents

- Introduction
- Configuration
 - Cached Authentication
- Avoiding Clear Text Passwords
- Preventing Authentication Fallback to the Local Artifactory Realm
- Using LDAPS (Secure LDAP)

Configuration

To configure LDAP authentication, in the **Admin** module go to **Security | LDAP** and click **New**.

New LDAP Setting

Enabled

Settings Name *

LDAP URL *

Auto Create Artifactory Users

User DN Pattern

Email Attribute

Search Filter

Search Base

Manager DN

Manager Password

Sub-tree Search

Test LDAP Connection

Test User Name

Test Password

Test Connection

Create

Cancel

The configuration parameters for LDAP connection settings are as follows:

Settings Name	The unique ID of the LDAP setting.
Enabled	When set, these settings are enabled.
LDAP URL	<p>Location of the LDAP server in the following format: <i>ldap://myserver:myport/dc=sampledomain,dc=com</i>.</p> <p>The URL should include the base DN used to search for and/or authenticate users.</p>

User DN Pattern	<p>A DN pattern used to log users directly in to the LDAP database. This pattern is used to create a DN string for "direct" user authentication, and is relative to the base DN in the LDAP URL.</p> <p>The pattern argument {0} is replaced with the username at runtime. This only works if anonymous binding is allowed and a direct user DN can be used (which is not the default case for Active Directory).</p> <p>For example:</p> <pre>uid={0},ou=People</pre>
Auto Create Artifactory Users	When set, Artifactory will automatically create new users for those who have logged in using LDAP, and assign them to the default groups.
Email Attribute	An attribute that can be used to map a user's email to a user created automatically by Artifactory.
Search Filter	<p>A filter expression used to search for the user DN that is used in LDAP authentication.</p> <p>This is an LDAP search filter (as defined in 'RFC 2254') with optional arguments. In this case, the <code>username</code> is the only argument, denoted by '<code>{0}</code>'.</p> <p>Possible examples are:</p> <ul style="list-style-type: none"> <code>uid={0}</code> - this would search for a username match on the uid attribute. <p>Authentication using LDAP is performed from the DN found if successful.</p>
Search Base	The Context name in which to search relative to the base DN in the LDAP URL. This parameter is optional.
Manager DN	The full DN of a user with permissions that allow querying the LDAP server. When working with LDAP Groups, the user should have permissions for any extra group attributes such as <code>memberOf</code> .
Manager Password	The password of the user binding to the LDAP server when using "search" authentication.
Search Sub Tree	When set, enables deep search through the sub-tree of the LDAP URL + Search Base. True by default.

Cached Authentication

You can configure Artifactory to cache data about authentication against external systems such as LDAP. This means that the first time a user needs to be authenticated, Artifactory will query the external system for the user's permissions, group settings etc.

The information received from the external system is cached for a period of time which you can configure in the `$ARTIFACTORY_HOME/etc/artifactory.system.properties` file by setting the `artifactory.security.authentication.cache.idleTimeSecs` property.

This means that once a user is authenticated, while the authentication data is cached, Artifactory will use the cached data rather than querying the external system, so authentication is much faster

By default this is set to 300sec.

Avoiding Clear Text Passwords

Storing your LDAP password in clear text in `settings.xml` on your disk is a big security threat, since this password is very sensitive and is used in SSO to other resources in the domain.

When using LDAP, we strongly recommend, using [Artifactory's Encrypted Passwords](#) in your local settings.

Preventing Authentication Fallback to the Local Artifactory Realm

In some cases, as an administrator you may want to require users to authenticate themselves through LDAP with their LDAP password. However, if a user already has an internal account with a password in Artifactory, Artifactory can fallback to use his internal password if LDAP authentication fails.

You can prevent this fallback authentication by ensuring that the **Disable Internal Password** checkbox in the [Edit User](#) dialog is set.

Using LDAPS (Secure LDAP)

To use LDAPS with a valid certificate from a CA trusted by Java, all you need to do is use a secure LDAP URL in your settings, e.g. `ldaps://secure_ldap_host:636/dc=sampledomain,dc=com`.

If you want to use LDAPS with a non-trusted (self-signed) certificate, please follow the steps below (thanks to Marc Schoechlin for providing this information):

1. Download the CA of the ssl secured ldap server
`openssl s_client -connect the ldap.server.net:636 -showcerts < /dev/null > server.ca`
2. Identify the CA certificate and keep only the ascii-text between BEGIN/END CERTIFICATE marker
3. Identify the standard `cacerts` file of your Java installation
4. Create a custom `cacerts` file by copying the `cacerts` file to the Artifactory configuration dir, e.g.
`cp /usr/lib64/jvm/java-1_6_0_ibm-1.6.0/jre/lib/security/cacerts /etc/opt/jfrog/artifactory/`
5. Import the CA certificate into the **customized** `cacerts` file
`keytool -import -alias myca -keystore /etc/opt/jfrog/artifactory/cacerts -trustcacerts -file server.ca`
=> Password: changeit
=> Agree to add the certificate
6. Change permissions for the artifactory user
`chown 755 /etc/opt/jfrog/artifactory/cacerts`
`chown artifactory:users /etc/opt/jfrog/artifactory/cacerts`
7. Modify the defaults of the Artifactory JVM to use the custom `cacerts` file
`echo "export JAVA_OPTIONS=\"$JAVA_OPTIONS -Djavax.net.ssl.trustStore=/etc/opt/jfrog/artifactory/cacerts\" >> /etc/opt/jfrog/artifactory/default`
8. Restart Artifactory

Managing Security with Active Directory

Introduction

Artifactory supports integration with an Active Directory server to authenticate users and synchronize groups.

When authentication using Active Directory is configured and active, Artifactory first attempts to authenticate the user against the Active Directory server. If the authentication fails, Artifactory tries to authenticate via its internal database.

For every externally authenticated user configured in your Active Directory server, Artifactory creates a new user in the internal database (provided the user does not already exist), and automatically assigns that user

to the default groups.

Page Contents

- Introduction
- Working With Active Directory
- Importing Active Directory Groups
 - Support for Nested Groups
- Using Secure Active Directory

Working With Active Directory

We will describe how to configure Artifactory to work with Active Directory using an example.

Consider an Active Directory server that must support the following conditions:

- Users are located in two geographically separated sites. Some are in the US (designated as "us"), while others are in Israel (designated as "il").
- Each site defines users and groups in different places in the Active Directory tree as displayed below.

The screenshot shows the Active Directory Administrative Center interface. The left navigation pane shows a tree structure of Active Directory objects, including 'Active Directory Admin...', 'jfrog (local)', 'frogs', 'il', 'us', and various user and group entries under 'il' and 'us'. The main pane displays a table for the 'frogs' organizational unit, with two entries: 'il' (Organizational Unit, Israel Group) and 'us' (Organizational Unit, USA Group). A search bar at the top is set to 'frogs (2)'. On the right, a 'Tasks' sidebar provides options like 'New', 'Delete', 'Move...', 'Search under this node', and 'Properties' for both the 'frogs' container and the selected 'il' object. At the bottom, a 'WINDOWS POWERSHELL HISTORY' window is visible.

Name	Type	Description
il	Organizational Unit	Israel Group
us	Organizational Unit	USA Group

To configure Active Directory authentication, in the **Admin** module, go to **Security | LDAP** and click **New**.

New LDAP Settings

LDAP Settings

Enabled

Settings Name *

frogs-il

LDAP URL *

ldap://win2012.jfrog.local:389/dc=jfrog,dc=local

Auto Create Artifactory Users

User DN Pattern

Email Attribute

mail

Search Filter

sAMAccountName={0}

Search Base

ou=il,ou=frogs | ou=us,ou=frogs

Manager DN

cn=Administrator,cn=Users,dc=jfrog,dc=local

Manager Password

Sub-tree Search

Test LDAP Connection

Test User Name

Test Password

Test Connection

Cancel

Create

The configuration parameters are as follows:

Settings Name	The unique ID of the Active Directory setting.
Enabled	When set, these settings are enabled.
Active Directory URL	Location of the Active Directory server LDAP access point in the following format: <i>ldap://myserver:myport/dc=sampledomain,dc=com</i> . The URL may include the base DN used to search for and/or authenticate users. If not specified, the Search Base field is required.

User DN Pattern	A DN pattern used to log users directly in to the LDAP database. For Active Directory, we recommend leaving this field blank since this only works if anonymous binding is allowed and a direct user DN can be used, which is not the default case in Active Directory.
Auto Create Artifactory Users	When set, Artifactory will automatically create new users for those who have logged in using Active Directory. Any newly created users will be associated to the default groups.
Email Attribute	An attribute that can be used to map a user's email to a user created automatically by Artifactory. This corresponds to the mail field in Active Directory.
Search Filter	A filter expression used to search for the user DN that is used in Active Directory authentication. This is an LDAP search filter (as defined in 'RFC 2254') with optional arguments. In this case, the <code>username</code> is the only argument, denoted by ' <code>{0}</code> '. For Active Directory the corresponding field should be <code>sAMAccountName={0}</code> .
Search Base	The Context name in which to search relative to the base DN in the Active Directory URL. This parameter is optional, but if possible, we highly recommend that you set it to prevent long searches on the Active Directory tree. Leaving this field blank will significantly slow down the Active Directory integration. The configuration in the example below indicates that search should only be performed under "frogs/il" or "frogs/us". This improves search performance since Artifactory will not search outside the scope of the "frogs" entry.
Manager DN	The full DN of a user with permissions that allow querying the Active Directory server. When working with LDAP Groups, the user should have permissions for any extra group attributes such as memberOf .
Manager Password	The password of the user binding to the Active Directory server when using "search" authentication.
Search Sub Tree	When set, enables deep search through the sub-tree of the Active Directory URL + Search Base. True by default.

Importing Active Directory Groups

Active Directory groups can be imported using either a **Static** mapping strategy or a **Dynamic** one (Active Directory works for both).

The only difference is in the attribute defined on the corresponding Active Directory entry:

- The Static mapping strategy defines a "member" multi-value attribute on the **group** entry containing user DNs of the group members
- The "Dynamic" configuration defines a "memberOf" multi-value attribute on the **user** entry containing group DNs of the groups the user is a member of.

Active Directory supports both configurations, so you can choose the one which fits your organization's structure.

New LDAP Group Setting

LDAP Group Settings

Settings Name *	LDAP Setting
allfrogs	frogs
Mapping Strategy: Static Dynamic Hierarchy	
Group Member Attribute *	Group Name Attribute *
member	cn
Description Attribute *	Filter *
description	(objectClass=group)
Search Base	<input checked="" type="checkbox"/> Sub-tree Search
ou=frogs	

Synchronize LDAP Groups

Search Group by Username (leave blank for *)	Import	< page 1 of 1 >
dev		
Group Name	Description	Sync ...
dev-il	R&D Israel	<input checked="" type="checkbox"/>
dev-us	R&D USA	<input checked="" type="checkbox"/>

Cancel **Create**

The screenshot displays a configuration interface for importing Active Directory groups via LDAP. The top section, 'LDAP Group Settings', allows defining the local group name ('allfrogs'), the LDAP base ('frogs'), and the mapping strategy ('Static'). It also specifies attributes for members ('member') and names ('cn'), and includes a search base ('ou=frogs') and a sub-tree search option. The bottom section, 'Synchronize LDAP Groups', shows a list of groups from the LDAP source ('dev'). Each group has a sync checkbox, with both 'dev-il' and 'dev-us' checked. The interface includes standard navigation and save/cancel buttons.

New LDAP Group Setting

LDAP Group Settings

Settings Name *	LDAP Setting
allfrogs	frogs
Mapping Strategy: Static Dynamic Hierarchy	
Group Member Attribute *	Group Name Attribute *
memberOf	cn
Description Attribute *	Filter *
description	(objectClass=group)
Search Base	<input checked="" type="checkbox"/> Sub-tree Search
ou=frogs	

Synchronize LDAP Groups

Search Group by Username (leave blank for *)		Import	< page 1 of 1 >
dev			
Group Name	Description	Sync ...	
dev-il	R&D Israel	<input checked="" type="checkbox"/>	
dev-us	R&D USA	<input checked="" type="checkbox"/>	

Cancel **Create**

Support for Nested Groups

Artifactory supports synchronization with Active Directory "Nested Groups".

Microsoft provides a unique OID for rule chain matching as part of the search filter syntax, so when executing an LDAP Query to Active Directory using this OID, Active Directory returns a list of all the groups that a user's main group membership inherits from.

The screenshot below shows the following example:

Mapping Strategy: Static

Group Membership Attribute: member:1.2.840.113556.1.4.1941:

Group Name Attribute: cn

Filter: (objectClass=group)

New LDAP Group Setting

LDAP Group Settings

Settings Name * [?](#)

LDAP Setting [?](#)

Mapping Strategy: [Static](#) | [Dynamic](#) | [Hierarchy](#)

Group Member Attribute * [?](#)

Group Name Attribute * [?](#)

Description Attribute * [?](#)

Filter * [?](#)

Search Base [?](#)

 Sub-tree Search

Synchronize LDAP Groups



46 Records

Import < Page 1 of 2 >

<input checked="" type="radio"/> Group Name	Description	Sync S...
<input checked="" type="radio"/> enterprise read-only domain con...	Members of this group are Read-Only Domain Controllers in the enterp...	<input checked="" type="checkbox"/>
<input checked="" type="radio"/> winrmremotewmiusers_	Members of this group can access WMI resources over management pr...	<input checked="" type="checkbox"/>
<input checked="" type="radio"/> cert publishers	Members of this group are permitted to publish certificates to the direct...	

[Cancel](#)

[Create](#)

Using Secure Active Directory

To use Secure Active Directory with a valid certificate from a CA trusted by Java, all you need to do is use a secure Active Directory URL in your settings, e.g. `ldaps://secure_ldap_host:636/dc=sampledomain,dc=com`.

If you want to use Secure Active Directory with a non-trusted (self-signed) certificate, please follow the steps below (thanks to Marc Schoechlin for providing this information):

1. Download the CA of the ssl secured Active Directory server
`openssl s_client -connect the.ldap.server.net:636 -showcerts < /dev/null > server.ca`
2. Identify the CA certificate and keep only the ascii-text between BEGIN/END CERTIFICATE marker
3. Identify the standard cacerts file of your Java installation
4. Create a custom cacerts file by copying the cacerts file to the Artifactory configuration dir, e.g.
`cp /usr/lib64/jvm/java-1_6_0-ibm-1.6.0/jre/lib/security/cacerts /etc/opt/jfrog/artifactory/`
5. Import the CA certificate into the **customized** cacerts file
`keytool -import -alias myca -keystore /etc/opt/jfrog/artifactory/cacerts -trustcacerts -file server.`

```

ca
=> Password: changeit
=> Agree to add the certificate
6. Change permissions for the artifactory user
    chown 755 /etc/opt/jfrog/artifactory/cacerts
    chown artifactory:users /etc/opt/jfrog/artifactory/cacerts
7. Modify the defaults of the Artifactory JVM to use the custom cacerts file
    echo "export JAVA_OPTIONS=\"\$JAVA_OPTIONS -Djavax.net.ssl.trustStore=/etc/opt/jfrog/artifactory/cacerts\\"" >> /etc/opt/jfrog/artifactory/default
8. Restart Artifactory

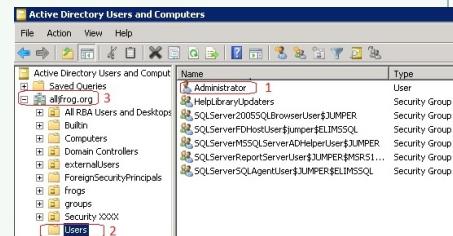
```

Manager DN

To construct the Manager DN string according to your Active Directory server, navigate to a user with administrator privileges (e.g. Administrator (1)), and then construct the Manager DN in reverse order (2,3) from the User, up the folder hierarchy.

For example, in this simple configuration, the Manager DN here should be
cn=Administrator,cn=Users,dc=alljfrog,dc=org

*Notice that the domain (3) is split in reverse order to
dc=alljfrog,dc=org*



Access Log

The Artifactory access.log

Artifactory maintains an access log containing all security-related events, their source IP and context. Events include information on accept/reject of logins, and download, browsing and deployment of artifacts.

The access log is located at `$ARTIFACTORY_HOME/logs/access.log`.

You can also view and download the access log from the Artifactory UI. In the **Admin** module go to **Advanced | System Logs**.

System Logs

artifactory.log

File last modified: Tue Jul 14 19:00:01 IOT 2015 View last updated: Tue Jul 14 19:02:23 IOT 2015

Download (17.0 MB)

```
istruz npmjs.org/raw-body 49, 10 KB at 2370.29 KB/sec
2015-07-14 15:42:23, 469 ([http://nlu-0081-exec-7] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/websocket-driver 34.68 KB
2015-07-14 15:42:23, 472 ([http://nlu-0081-exec-10] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/lodash._htmlscraper 19.85 KB
2015-07-14 15:42:23, 475 ([http://nlu-0081-exec-1] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/lodash._htmlscraper 19.85 KB at 2711.42 KB/sec
2015-07-14 15:42:23, 484 ([http://nlu-0081-exec-10] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/lodash._htmlscraper 19.85 KB at 1461.61 KB/sec
2015-07-14 15:42:23, 706 ([http://nlu-0081-exec-2] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/fatwarem 40.88 KB
2015-07-14 15:42:23, 709 ([http://nlu-0081-exec-1] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/fatwarem 40.88 KB at 4024.01 KB/sec
2015-07-14 15:42:23, 804 ([http://nlu-0081-exec-6] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/pullstream 17.78 KB
2015-07-14 15:42:23, 819 ([http://nlu-0081-exec-1] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/semimedium 10.29 KB
2015-07-14 15:42:24, 008 ([http://nlu-0081-exec-1] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/semimedium 10.29 KB at 1200.75 KB/sec
2015-07-14 15:42:24, 182 ([http://nlu-0081-exec-4] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/mach-stream 9.09 KB
2015-07-14 15:42:24, 198 ([http://nlu-0081-exec-4] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/mach-stream 9.09 KB at 888.42 KB/sec
2015-07-14 15:42:24, 261 ([http://nlu-0081-exec-5] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/argparse 30.39 KB
2015-07-14 15:42:24, 263 ([http://nlu-0081-exec-5] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/argparse 30.39 KB at 21.84 KB/sec
2015-07-14 15:42:25, 159 ([http://nlu-0081-exec-5] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/binary 38 KB at 1940.80 KB/sec
2015-07-14 15:42:25, 168 ([http://nlu-0081-exec-6] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/websocket-driver/-/websocket-driver-0.6.1.tgz 17.57 KB
2015-07-14 15:42:25, 204 ([http://nlu-0081-exec-4] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/find-index 7.38 KB at 782.94 KB/sec
2015-07-14 15:42:25, 208 ([http://nlu-0081-exec-1] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/argparse 30.39 KB
2015-07-14 15:42:25, 260 ([http://nlu-0081-exec-9] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/argparse 30.39 KB at 2846.02 KB/sec
2015-07-14 15:42:26, 660 ([http://nlu-0081-exec-9] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/avicons@2aviconfont 46.93 KB
2015-07-14 15:42:26, 679 ([http://nlu-0081-exec-9] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/avicons@2aviconfont 46.93 KB at 2489.23 KB/sec
2015-07-14 15:42:27, 274 ([http://nlu-0081-exec-7] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/ultron 3.84 KB
2015-07-14 15:42:27, 277 ([http://nlu-0081-exec-7] [INFO] ) [o.a.r.HttpRepo :389] - npm downloaded http://registry.npmjs.org/ultron 3.84 KB at 884.95 KB/sec
2015-07-14 15:42:27, 289 ([http://nlu-0081-exec-8] [INFO] ) [o.a.r.HttpRepo :389] - npm downloaded http://registry.npmjs.org/globule 20.65 KB
2015-07-14 15:42:27, 319 ([http://nlu-0081-exec-8] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/globule 20.65 KB at 1246.72 KB/sec
2015-07-14 15:42:27, 652 ([http://nlu-0081-exec-2] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/options 7.14 KB
2015-07-14 15:42:27, 655 ([http://nlu-0081-exec-2] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/options 7.14 KB at 861.84 KB/sec
2015-07-14 15:42:27, 988 ([http://nlu-0081-exec-9] [INFO] ) [o.a.r.HttpRepo :389] - npm downloading http://registry.npmjs.org/nan 107.02 KB
2015-07-14 15:42:27, 995 ([http://nlu-0081-exec-9] [INFO] ) [o.a.r.HttpRepo :371] - npm downloaded http://registry.npmjs.org/nan 107.02 KB at 6780.72 KB/sec
2015-07-14 15:42:28, 539 ([http://nlu-0081-exec-1] [INFO] ) [o.a.r.HttpRepo :389] - npm downloaded http://registry.npmjs.org/angr 23.09 KB
... ... ...
```

Page Contents

- The Artifactory access.log
- Watches

Watches

You can also choose to receive focused information about events for a specific repository section, using the **Watches Add-on**.

Mail Server Configuration

Overview

Artifactory supports sending mail to notify administrators and other users for significant events that happen in your system.

Some examples are:

- Watch notifications
- Alerts for backup warnings and errors
- License violation notifications

To enable mail notifications, you need to configure Artifactory with your mail server details as described below.

Page Contents

- Overview
- Setup

Setup

To access the mail server configuration, in the **Admin** module select **Configuration | Mail**.

Setup is straightforward and can be verified by sending a test message. Simply click "Send Test Mail" in the **Configure Mail** screen.

Admin

Back >

Configuration

General

Repositories

Repository Layouts

Artifactory Licenses

Black Duck

Property Sets

Proxies

Mail

Bintray

Register License

Security

Services

Import & Export

Configure Mail

Mail Server Settings

 Enable [?](#)

Host *

Port *

Username

Password

From

Subject Prefix

Artifactory URL

 Use TLS Use SSL

Test Message Recipient

[Send Test Mail](#)[Reset](#)[Save](#)

Enabled	When set, mail notifications are enabled
Host	The host name of the mail server
Port	The port of the mail server
Username	The username for authentication with the mail server
Password	The password for authentication with the mail server
From (optional)	The "from" address header to use in all outgoing mails.
Subject Prefix	A prefix to use for the subject of all outgoing mails

Artifactory URL (optional)	The Artifactory URL to use in all outgoing mails to denote links to Artifactory.
Use TLS	When set, uses Transport Layer Security when connecting to the mail server
Use SSL	When set, uses a secure connection to the mail server
Test Message Recipient	The email address of a recipient to receive a test message

Configuration Files

All Artifactory configuration files are located under the `$ARTIFACTORY_HOME/etc` folder.

On Linux, Solaris and MacOS `$ARTIFACTORY_HOME` is usually a soft link to `/etc/artifactory`.

Global Configuration Descriptor

The global Artifactory configuration file is used to provide a default set of configuration parameters.

The file is located in `$ARTIFACTORY_HOME/etc/artifactory.config.xml` and is loaded by Artifactory at initial startup. Once the file is loaded, Artifactory renames it to `artifactory.config.bootstrap.xml` and from that point on, the configuration is stored internally in Artifactory's storage. This ensures Artifactory's configuration and data are coherently stored in one place making it easier to back up and move Artifactory when using direct database backups. On every startup, Artifactory also writes its current configuration to `$ARTIFACTORY_HOME/etc/artifactory.config.startup.xml` as a backup.

At any time, the default configuration can be changed in the Artifactory UI **Admin** module.

There are two ways to directly modify the Global Configuration Descriptor:

1. Using the Artifactory UI
2. Using the REST API

Page Contents

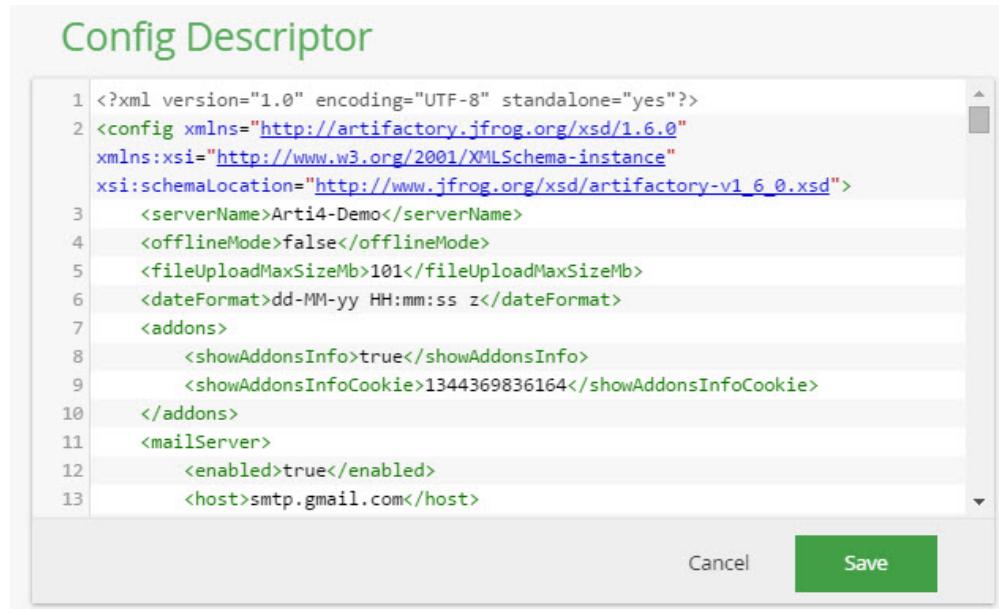
- Global Configuration Descriptor
 - Modifying Configuration Using the UI
 - Modifying Configuration Using the REST API
 - Bootstrapping the Global Configuration
- Security Configuration Descriptor
 - Modifying Security Using the UI
 - Modifying Security Using the REST API
 - Bootstrapping the Security Configuration
- Content Type/MIME Type
 - MIME Type Attributes
 - Setting Content-Type During Download
- System Properties
- Logging Configuration Files
- Storage Properties

Care

Direct modification of the global configuration descriptor is an advanced feature, and if done incorrectly may render Artifactory in an undefined and unusable state. We strongly recommend backing up the configuration before making any direct changes, and taking great care when doing so.

Modifying Configuration Using the UI

You can access the Global Configuration Descriptor in the **Admin** module under **Advanced | Config Descriptor**. There you can modify the file's contents directly or copy the contents from the entry field.



Modifying Configuration Using the REST API

You can retrieve or set the global configuration by sending a GET or POST request to `http://<host>:<port>/artifactory/api/system/configuration`. For example:

Retrieving and Setting the Global Configuration Descriptor

```
curl -u admin:password -X GET -H "Accept: application/xml"  
http://localhost:8080/artifactory/api/system/configuration  
curl -u admin:password -X POST -H "Content-type:application/xml" --data-binary  
@artifactory.config.xml http://localhost:8080/artifactory/api/system/configuration
```

Bootstrapping the Global Configuration

You can bootstrap Artifactory with a predefined global configuration by creating an `$ARTIFACTORY_HOME/etc/artifactory.config.import.xml` file containing the Artifactory configuration descriptor.

If Artifactory detects this file at startup, it uses the information in the file to override its global configuration. This is useful if you want to copy the configuration to another instance of Artifactory.

Security Configuration Descriptor

There are two ways to directly modify the Security Configuration Descriptor:

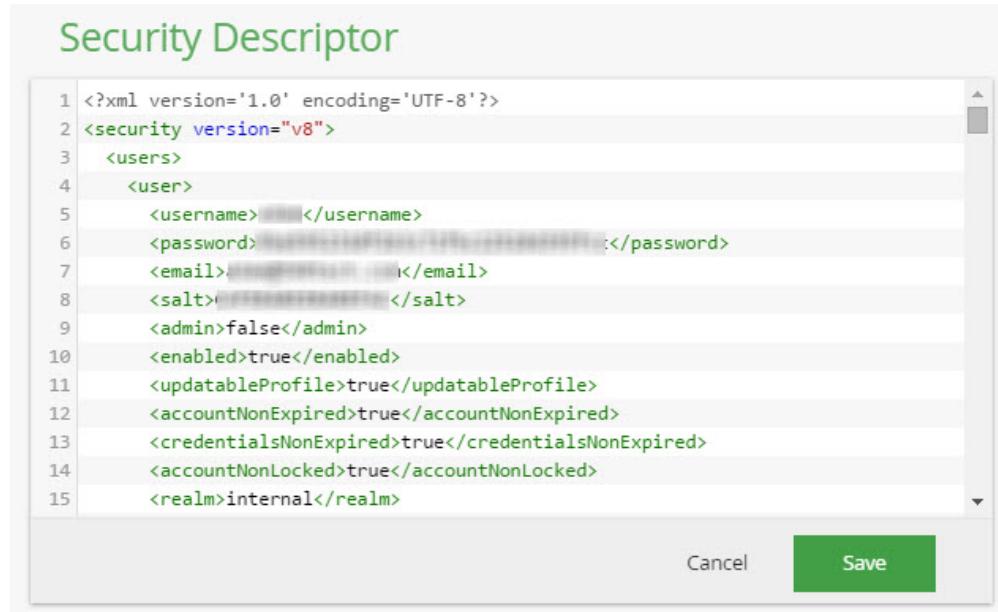
1. Using the Artifactory UI
2. Using the REST API

Care

Direct modification of the security descriptor is an advanced feature, and if done incorrectly may render Artifactory in an undefined and unusable state. We strongly recommend backing up the configuration before making any direct changes, and taking great care when doing so.

Modifying Security Using the UI

You can access the Security Configuration Descriptor in the **Admin** module under **Advanced | Security Descriptor**. There you can modify the file's contents directly or copy the contents from the entry field.



Modifying Security Using the REST API

You can retrieve or set the security configuration by sending a GET or POST request to `http://<host>:<port>/artifactory/api/system/security`. For example:

Modifying the Security Descriptor

```
curl -u admin:password -X GET -H "Accept: application/xml"
http://localhost:8080/artifactory/api/system/security
curl -u admin:password -X POST -H "Content-Type: application/xml" --data-binary
@artifactory.config.xml http://localhost:8080/artifactory/api/system/security
```

Admin privileges

You must supply a user with **Admin** privileges to modify the security descriptor through the REST API

Bootstrapping the Security Configuration

Artifactory stores all security information as part of its internal storage. You can bootstrap Artifactory with a predefined security configuration by creating an `$ARTIFACTORY_HOME/etc/security.import.xml` file containing the Artifactory exported security configuration information.

If Artifactory detects this file at startup, it uses the information in the file to override all security settings. This is useful if you want to copy the security configuration to another instance of Artifactory.

Content Type/MIME Type

Artifactory provides a flexible mechanism to manage content type/MIME Type. You can define system-wide MIME types for common usage, but you can also overwrite the MIME types for specific files as needed. The list of default MIME types can be found in `$ARTIFACTORY_HOME/etc/mimetypes.xml` and can be edited in order to add, remove or change MIME types. If a file has an extension that is not supported by any of the MIME types, or does not have an extension at all, Artifactory will use the default MIME type of `application/octet-stream`. To determine an artifact's MIME type, Artifactory compares its extension with those in the `mimetype.xml` file, and applies the MIME type of the first extension that matches.

MIME Type Attributes

Each MIME type may have the following attributes:

<code>type</code>	The MIME type unique name (mandatory)
<code>extensions</code>	A comma separated list of file extensions mapped to this MIME type (mandatory)
<code>index</code>	True if this MIME type should be indexed for archive searching (valid only for supported archive files)
<code>archive</code>	True if this MIME type is a browsable archive
<code>viewable</code>	True if this MIME type can be viewed as a text file inside Artifactory UI
<code>syntax</code>	The UI highlighter syntax to for this MIME type (only relevant if this is a viewable type)
<code>css</code>	The css class of a display icon for this mime type

Example of `mimetype.xml`

```
<mimetypes version="4">
    <mimetype type="text/plain" extensions="txt, properties, mf, asc" viewable="true" syntax="plain"/>
        <mimetype type="text/html" extensions="htm, html" viewable="true" syntax="xml"/>
        <mimetype type="text/css" extensions="css" viewable="true" syntax="css"/>
        <mimetype type="text/xsl" extensions="xsl" viewable="true" syntax="xml"/>
        <mimetype type="text/xslt" extensions="xslt" viewable="true" syntax="xml"/>
        <mimetype type="text/x-java-source" extensions="java" viewable="true" syntax="java"/>
        <mimetype type="text/x-javafx-source" extensions="fx" viewable="true" syntax="javafx"/>
</mimetypes>
```

For example, from the extensions parameter in the above `mimetypes.xml` file sample we can conclude that:

- `test.properties` is a `text/plain` MIME type
- `test.css` is a `text/css` MIME type
- `test.doc` is an `application/octet-stream` MIME type since `"doc"` is not included in any of the other MIME types).

Artifactory MIME Types

Some of the Mime-Types specified in `mimetypes.xml` (e.g. `application/x-checksum`) are used by Artifactory. Great care should be taken before changing these Mime-Types to ensure Artifactory continues to function correctly.

Setting Content-Type During Download

Using Artifactory, when downloading files you can override the `Content-Type` HTTP header by setting the `artifactory.content-type` property.

If the `artifactory.content-type` property is not explicitly set, Artifactory will use the default mechanism of matching the artifact name extension to the extensions in the `mimetypes.xml` file to apply the Content-Type.

This feature is only available with Artifactory Pro.

System Properties

Rather than configuring properties in the JVM runtime configuration of the hosting container, you can edit `$ARTIFACTORY_HOME/etc/artifactory.system.properties` file and restart Artifactory.

The Artifactory system properties are documented within this file.

Since these settings impact the entire container VM, we recommend using this feature primarily for specifying Artifactory-related properties only (such as changing the database used by Artifactory, etc.).

Setting properties in `artifactory.system.properties` is an advanced feature and is typically not required.

Do not confuse these setting with those in the `$ARTIFACTORY_HOME/data/artifactory.properties` file, which are for internal use.

Logging Configuration Files

Artifactory uses the Logback Framework to manage logging and lets you configure the verbosity of log files. For details please refer to [Configuring Log Verbosity](#)

Storage Properties

Artifactory provides you with a `storage.properties` file so that you can configure the specific storage solution used in your system. For details please refer to [Setup the New Storage](#)

Exposing Maven Indexes

Overview

Artifactory exposes Maven indexes for use by Maven integrations of common IDEs (for example, IntelliJ IDEA, NetBeans, Eclipse).

Indexes are fetched remotely from remote repositories that provide them and are calculated for local and

virtual repositories (note that many repositories do not provide indexes, or do not keep an updated index).

If Artifactory cannot find a remote index, it calculates one locally, based on the remote repository's previously cached artifacts.

Page Contents

- [Overview](#)
- [Usage](#)

Artifactory's search and indexing facilities are not related to Maven indexes

The indexing performed by Artifactory is secure, immediately effective and supports a larger variety of search options, including custom metadata searches.

Maven indexes only exist in Artifactory for the purpose IDE integrations. They are periodically calculated, contain a limited set of data and are non-secure by design.

Information about the content of a repository is exposed to anyone with access to the repository's index, regardless of any effective path permission you have in place. If this is a concern, do not expose an index for that repository.

Usage

To administer Maven indexes, in the **Admin** module select **Services | Maven Indexer**.

Artifactory provides you with controls to specify how frequently indexing is run and which repositories are included in the index calculation.

Maven Indexer Support

Enabled
Cron Expression*
0 23 5 * * ?

Next Indexing Time
Thu Jul 09 05:23:00 UTC 2015

Run Indexing Now

Included Repositories	Excluded Repositories
<input type="text" value="filter..."/>	<input type="button" value="Clear All"/>
<input type="button" value="<<"/>	<input type="button" value="x"/>
<input type="button" value="<"/>	<input type="button" value="x"/>
<input type="button" value=">"/>	<input type="button" value="x"/>
<input type="button" value=">>"/>	<input type="button" value="x"/>
gradle-libs	RubyGems-local
java.net.m1	dfsfq
repo1	ext-release-local
	ext-snapshot-local
	ivy-local
	libs-release-local

Enabled When set, indexing is enabled and will run according to the Cron Expression setting

Cancel **Save**

<i>Cron Expression</i>	A valid Cron expression that determines the frequency in which Maven indexes on the selected repositories will be recalculated
<i>Next Indexing Time</i>	Indicates the next scheduled indexing run
<i>Run Indexing Now</i>	Invokes indexing immediately
<i>Included Repositories</i>	Specifies the repositories that should be indexed on the next run
<i>Excluded Repositories</i>	Specifies the repositories that should not be indexed on the next run
<i>Clear All</i>	Removes all repositories from the Included Repositories list

Indexing is resource intensive

Calculating and indexing for a repository may be a resource intensive operation, especially for a large local repository or if the repository is a virtual one containing other underlying repositories.

Therefore, we recommend that you do not include repositories that do not require indexing for a periodic index calculation.

Clustering Artifactory

Active/Active Architecture

Artifactory HA is an Active/Active clustered installation of Artifactory that provides a full set of true High Availability features and is provided as a part of the Artifactory [Enterprise Value Pack](#).

For full details please refer to [Artifactory High Availability](#).

Active/Passive Architecture

Overview

Artifactory clustered Active/Passive architecture provides fast disaster recovery and can be implemented in one of the following two ways:

- Deployment on fault-tolerant storage (strongly recommended)
- Periodic cross-server data sync.

Deployment on Fault-tolerant Storage

Using a fault-tolerant disk mounted on another machine allows for a very short MTR (Mean Time to Recovery) in case the "active" server goes down. If Artifactory is deployed on a NAS or SAN the "passive" machine can immediately mount the storage, bootstrap Artifactory from it and start accepting requests in place of the originally "active" machine that has gone down.

Page Contents

- Active/Active Architecture
- Active/Passive Architecture
 - Overview
 - Deployment on Fault-tolerant Storage
 - Cross-server Data Synchronization
 - Syncronizing the Data and Configuration on Directories
 - Syncronizing the Database
 - Time Synchronization on the Standby Server

To set this up quickly and efficiently, we recommend using the built-in Virtual Machine Failover feature offered by virtualization software providers as follows:

1. Create a VM image that runs the Artifactory startup script and mounts the auxiliary storage.
2. The storage should contain the full Artifactory installation along with the data in a location defined as \$ARTIFACTORY_HOME.
3. Use the VM image on two Virtual Machines and have Artifactory running on one machine while the other machine is readily available as a failover target by the virtualization monitor.

Cross-server Data Synchronization

If deployment on fault-tolerant storage, as described in the previous section, is not possible (or if redundancy is required), fault-tolerance can be achieved by correctly replicating the data folder to a warm standby server.

The setup of an up-to-date passive replication server for the active Artifactory server requires database replication and synchronization of file system directories.

Synchronizing the Data and Configuration Directories

To synchronize the data and configuration directories you need to run `rsync` on `$ARTIFACTORY_HOME/data` and `$ARTIFACTORY_HOME/etc`.

This can be done by running the `rsync` command on `$ARTIFACTORY_HOME` while excluding the directories that are not required as follows:

```
rsync -vvah --del --progress --log-file=/home/replication/replication.log  
--exclude-from=rsync-excludes.txt \  
artifactory@active-artifactory-host:$ARTIFACTORY_HOME/ $ARTIFACTORY_HOME/
```

For the above example the `rsync-excludes.txt` file appears as follows:

```
/work/  
/data/tmp*/  
/data/cache/  
/logs/
```

rsync

The `rsync` should be executed from the passive stand-by server

Synchronizing the Database

Database Replication

Database replication must run **before** executing `rsync`.

The procedure to synchronize a database varies between the different database vendors. Please refer to the relevant documentation for your specific database.

For example, instructions on how to synchronize with MySQL can be found in the MySQL documentation for [How to Set Up Replication](#).

It is also possible to use a full dump/restore procedure on the database to synchronize the database and filestore state. In this case, we recommend that you perform the dump in a single routine along with `rsync` (in case of File System Storage Types).

Time Synchronization on the Standby Server

It is very important that the metadata stored in the database and the data stored on the file system are synchronized on the standby server.

A straightforward way to achieve this, is to make sure that the database synchronized is in a state that is **prior** to the file system (data/filestore) state.

This allows you to:

- Make a database dump before executing the file system sync,
- Activate database replication on demand just before executing `rsync`.

Since the sync operations are not atomic, there may be a gap between the data from `rsync` and data from database replication.

1. The snapshot time that Artifactory is set to is the database replication time.
2. Items synced to the file system which have no representation in the database can be purged by clicking on **Prune Unreferenced Data** in the **Admin** tab and then **Advanced | Maintenance** in the Artifactory configuration.

System Monitoring and Maintenance

Overview

Artifactory provides a set of tools that allow you to monitor and maintain your system to keep it running and responsive:

- **System Information** lets you examine the various properties and parameters of your system at runtime and is a valuable resource when investigating any issues that may arise.
- You can **monitor storage** to view the number of artifacts and physical files in your system as well as the amount of space that they occupy.
- **Log files** let you monitor all the activity that has occurred in your system
- **JMX Beans** let you monitor repositories, executor pools and storage
- You can configure regular, periodic **maintenance operations** to manage resource allocation and free up disk space
- You can define a regimen for **complete system backup**
- You can **import and export** data both at system level and repository level

Read More

- System Information
- Monitoring Storage
- Artifactory Log Files
- Artifactory JMX MBeans
- Regular Maintenance Operations
- Managing Backups
- Importing and Exporting
- Managing Disk Space Usage
- Getting Support

System Information

Overview

Artifactory can display different system information such as JVM runtime parameters, JVM arguments, memory usage and more.

This can be useful if you need to examine your system at runtime and is a valuable resource when investigating any issues that may arise.

To view Artifactory system information, in the **Admin** module, go to **Advanced | System Info**.

Page Contents

- Overview

System Info

Storage Info:

Database Type: derby
Storage Type: filesystem
Connection Url: jdbc:derby:/data/downloads/artifactory-powerpack-4.x-SNAPSHOT/data/derby;create=true

System Properties:

artifactory.running.mode:	PRO
artifactory.running.state:	Online
Wicket_HeaderRenderStrategy:	org.apache.wicket.markup.renderStrategy.ParentFirstHeaderRenderStrategy
artifactory.home:	/data/downloads/artifactory-powerpack-4.x-SNAPSHOT
artifactory.version:	4.x-SNAPSHOT
awt.toolkit:	sun.awt.X11.XToolkit
catalina.base:	/data/downloads/artifactory-powerpack-4.x-SNAPSHOT/tomcat
catalina.home:	/data/downloads/artifactory-powerpack-4.x-SNAPSHOT/tomcat
catalina.useNaming:	true
common.loader:	"\${catalina.base}/lib","\${catalina.base}/lib/*.jar","\${catalina.home}/lib","\${catalina.home}/lib/*.jar"
derby.language.logStatementText:	false
derby.module.mgmt.jmx:	org.apache.derby.impl.services.jmxnone.NoManagementService
derby.storage.pageCacheSize:	500
derby.stream.error.file:	/data/downloads/artifactory-powerpack-4.x-SNAPSHOT/logs/derby.log
derby.stream.error.logSeverityLevel:	0
file.encoding:	UTF8
file.encoding.pkg:	sun.io
file.separator:	/

Monitoring Storage

Overview

Artifactory allows you to monitor various statistics related to the amount of storage that repositories occupy in

your system. You can view the number of artifacts and physical files as well as the amount of space that they occupy. To monitor usage of storage in your system, in the **Admin** module, go to **Advanced | Storage Summary**.

Page Contents

- Overview
- Binaries
- File Store
- Repositories

Binaries

This section provides information on the number of files in your system and the amount of physical and virtual storage that they occupy.

Binaries

Binaries Count	3,422
Binaries Size	446.61 MB
Artifacts Size	533.10 MB
Optimization	83.78%
Items Count	6,754

Binaries Count	The total number of physical binaries stored in your system.
Binaries Size	The amount of physical storage occupied by the binaries in your system.
Artifacts Size	The amount of physical storage that would be occupied if each artifact was a physical binary (not just a link).
Optimization	The ratio of Binaries Size to Artifacts Size . This reflects how much the usage of storage in your system has been reduced by Artifactory
Items Count	The total number of items (both files and folders) in your system.

File Store

Your system is set up to store binaries as defined in your storage configuration file.

This section provides information on where your binaries are stored and the amount of storage space they are using.

File Store

Storage Type	filesystem
Storage Directory	/data/downloads/artifactory-powerpack-4.x-SNAPSHOT/data/filestore
Total Space	195.88 GB
Used Space	143.20 GB (73.11%)
Free Space	52.68 GB (26.89%)

Storage Type	The type of storage used (may be "filesystem" or "fullDb").
Storage Directory	If Storage Type is "filesystem" then this is the path to the physical file store. If Storage Type is "fullDb" then this is the path to the directory that caches binaries when they are extracted from the database.
Total Space	The total size of your file system.
Used Space	The amount of space used on your file system.
Free Space	The amount of free space left on your file system.

Repositories

The **Repositories** section provides detailed information about the storage used by each repository in your system.

Repo Key	Folders Count	Files Count	Used Space	Items Count	Percentage
plugins-release	0	0	0 bytes	0	0%
p2	0	0	0 bytes	0	0%
java.net.m1-cache	4	2	103.02 KB	6	0.02%
plugins-snapshot	0	0	0 bytes	0	0%
p2-local	23	59	64.76 MB	82	12.15%
npm-virtual	0	0	0 bytes	0	0%
localhost-cache	0	0	0 bytes	0	0%
rubygems-cache	3	60	4.00 MB	63	0.75%
libs-snapshot	0	0	0 bytes	0	0%
libs-snapshot-local	19	107	24.35 MB	126	4.57%

<i>Repo key</i>	The repository id.
<i>Folders Count</i>	The total number of folders in this repository.
<i>Files Count</i>	The total number of files in this repository.
<i>Used Space</i>	The amount of space used by this repository.
<i>Items Count</i>	The total number of items (folders and files) in this repository.
<i>Percentage</i>	The percentage of the total available space occupied by this repository.

Artifactory Log Files

Overview

Artifactory uses the [Logback Framework](#) to manage logging. Activity is logged according to type in four different log files which can be found under the `ARTIFACTORY_HOME/logs` folder.

The following log files are available:

<i>artifactory.log</i>	The main Artifactory log file containing data on Artifactory server activity
<i>access.log</i>	Security log containing important information about accepted and denied requests, configuration changes and password reset requests. The originating IP address for each event is also recorded
<i>request.log</i>	Generic http traffic information similar to the Apache HTTPd request log.
<i>import.export.log</i>	A log used for tracking the process of long-running import and export commands

- Configuring Log Verbosity
- Log File Structure
 - Request Log
 - Access Log
- Viewing Log Files from the UI
- Sending Artifactory Logs to Syslog

Tomcat/Servlet container-specific log files

When running Artifactory inside an existing servlet container, the container typically has its own log files.

These files normally contain additional information to that in `artifactory.log` or application bootstrapping-time information that is not found in the Artifactory logs.

In Tomcat, these files are `catalina.out` and `localhost.yyyy-mm-dd.log` respectively.

Configuring Log Verbosity

The verbosity of any logger in your system can be configured by entering or modifying the `level` value in the corresponding entry in the Logback configuration file `ARTIFACTORY_HOME/etc/logback.xml`.

For example:

Modifying the verbosity of a logger

```
<logger name="org.apache.wicket">
    <level value="error"/>
</logger>
```

Artifactory loads any changes made to the Logback configuration file within several seconds without requiring a restart.

Log File Structure

The Request and Access log files each display specific type of activity and as such have a consistent and specific file structure for maximum readability

Request Log

A request log file record has the following structure:

Date and Time stamp | Request time | Request type | IP | User name | Request method | Requested resource path | Protocol version | Response code | Request Content-Length

Note: If not provided by the client, 'Request Content-Length' is initialised as "-1".

Here is a typical example:

Request log file record sample

```
20140508154145|2632|REQUEST|86:12:14:192|admin|GET|/jcenter/org/iostreams/iostreams/0.2/iostreams-0.2.jar|HTTP/1.1|200|8296
```

Date and time stamp	The date and time the request was completed and entered into the log file. Format is [YYYYMMDDHHMMSS]
Request time	The time in ms taken for the request to be processed
Request type	DOWNLOAD for a download request UPLOAD for an upload request REQUEST for any other request
IP	The requesting user's IP address
User name	The requesting user's user name or "non_authenticated_user" when accessed anonymously
Request method	The HTTP request method. e.g. GET, PUT etc.
Requested resource path	Relative path to the requested resource
Protocol version	The HTTP protocol version
Response code	The HTTP response code
Size (bytes) of request or response	If request method is GET: Size of response If request method is PUT or POST: Size of request

Access Log

An access log file record has the following structure:

Date and Time stamp | Action response and type | Repository path (Optional) | Message (Optional) | User name | IP

Here is a typical example:

Access log file record
2014-05-08 15:52:27,456 [ACCEPTED DOWNLOAD] jcenter-cache:org/iostreams/iostreams/0.2/iostreams-0.2.jar for anonymous/86:12:14:192.

<i>Date and Time stamp</i>	The date and time that the entry was logged. Format is [YYYY-MM-DD HH:MM:SS, milliseconds]
<i>[Action response and type]</i>	The response (ACCEPTED/DENIED) and the action type (e.g. DOWNLOAD, UPLOAD etc.)
<i>Repository path (Optional)</i>	The repository that was accessed
<i>Message (Optional)</i>	An optional system message
<i>User name</i>	The accessing user's user name or "anonymous" when accessed anonymously
<i>IP</i>	The accessing user's IP address

Viewing Log Files from the UI

You can view or download any of the Artifactory log files from the UI.

In the **Admin** module, under **Advanced | System Logs**, select the file you want to view from the drop-list. The log tail view is automatically refreshed every few seconds.

System Logs

artifactory.log

File last modified: Wed Jul 08 16:09:12 IDT 2015 View last updated: Wed Jul 08 16:28:55 IDT 2015

[Download](#) (11.6 MB)

```
2015-07-08 13:09:12,230 [art-init] [INFO ] (o.a.s.ArtifactoryApplicationContext:221) - Initializing org.artifactory.storage.db.servers.service.ArtifactoryHeartbeatService
2015-07-08 13:09:12,231 [art-init] [INFO ] (o.a.s.ArtifactoryApplicationContext:221) - Initializing org.artifactory.p2.transformer.XmlStreamTransformerService
2015-07-08 13:09:12,257 [art-init] [INFO ] (o.a.s.ArtifactoryApplicationContext:221) - Initializing org.artifactory.security.interceptor.SecurityConfigurationChangesInterceptors
2015-07-08 13:09:12,260 [art-init] [INFO ] (o.a.s.ArtifactoryApplicationContext:221) - Initializing org.artifactory.storage.InternalStorageService
2015-07-08 13:09:12,263 [art-init] [INFO ] (o.a.s.ArtifactoryApplicationContext:221) - Initializing org.artifactory.search.InternalSearchService
2015-07-08 13:09:12,263 [art-init] [INFO ] (o.a.s.ArtifactoryApplicationContext:221) - Initializing org.artifactory.repo.interceptor.StorageAggregationInterceptors
2015-07-08 13:09:12,265 [art-init] [INFO ] (o.a.s.ArtifactoryApplicationContext:221) - Initializing org.artifactory.md.MetadataDefinitionService
2015-07-08 13:09:12,492 [art-init] [INFO ] (o.a.s.ArtifactoryApplicationContext:406) - Artifactory application context is ready.
2015-07-08 13:09:12,496 [art-init] [INFO ] (o.a.w.s.ArtifactoryContextConfigListener:225) -
#####
### Artifactory successfully started (14.670 seconds) #####
#####
```

To save system resources, do not leave the log view open in your browser unnecessarily.

Sending Artifactory Logs to Syslog

Some sites want to consolidate logs into the syslog facility. Switching artifactory to use syslog in addition to, or instead of the standard log files takes a quick edit of a couple of files. Artifactory currently uses the logback library for logging, so that's what needs to be configured.

First edit the \$ARTIFACTORY_HOME/etc/logback.xml file to send logs to the syslog facility. You need to add an appender to syslog:

```
<appender name="SYSLOG" class="ch.qos.logback.classic.net.SyslogAppender">
<syslogHost>localhost</syslogHost>
<facility>SYSLOG</facility>
<suffixPattern>[%thread] %logger %msg</suffixPattern>
</appender>
```

then you need to add this appender to the output, in the section:

```
<root>
<level value="info"/>
<appender-ref ref="CONSOLE"/>
<appender-ref ref="FILE"/>
</root>
```

add:

```
<appender-ref ref="SYSLOG"/>
```

before the </root> line.

Save the file, you will not need to restart artifactory for this to take effect.

Since logback is using internet sockets, you have to make sure your syslog facility accepts them. Modern linux distributions are using the rsyslog daemon for syslogging. Ensure that the configuration for internet domain sockets is enabled, either by editing /etc/rsyslog.conf and uncommenting:

```
# Provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514

# Provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 514
```

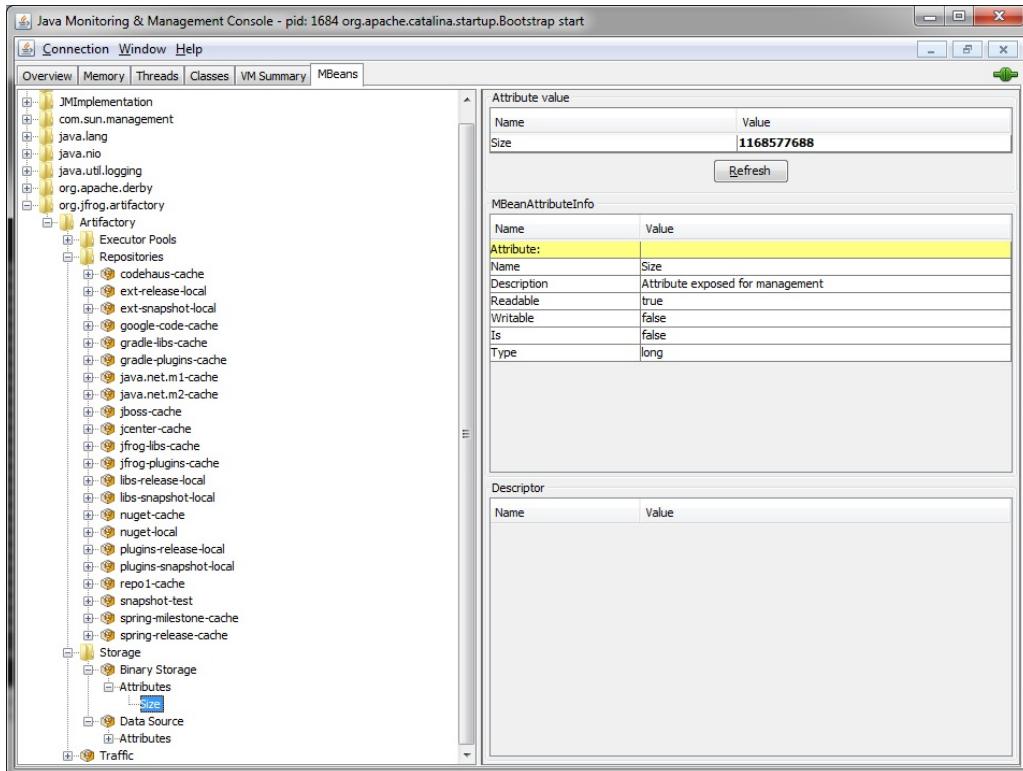
or placing it in a file under /etc/rsyslog.d ending in .conf

Rsyslog will need restarting with service rsyslog restart for this to take effect.

Artifactory JMX MBeans

Overview

Artifactory exposes MBeans under the `org.jfrog.artifactory` domain that let you monitor repositories, executor pools and storage.



Page Contents

- Overview
- Repositories
- Executor Pools

- Storage

Repositories

This section lists the available repositories under the current instance of Artifactory.
Read-only attributes are as follows:

<i>RepositoryKey</i>	Name of the repository
<i>ArtifactsCount</i>	Number of artifacts in the repository
<i>ArtifactsTotalSize</i>	Total size of all of the artifacts in the repository

Executor Pools

This section lists the executor pools in use by Artifactory.
Read-only attributes are as follows:

<i>TaskCount</i>	Total number of tasks that have ever been scheduled for execution
<i>CompletedTaskCount</i>	Total number of tasks that have been completed by the executors
<i>CorePoolSize</i>	Executor pool size
<i>MaximumPoolSize</i>	Maximum size of the executor pool
<i>ActiveCount</i>	Number of active executors

Storage

This section describes File System Binary Storage and Database Data Source read-only attributes.

There is only one File System Binary Storage read-only attribute:

<i>Size</i>	Total size of Artifactory storage in bytes
-------------	--

Database Data Source read-only attributes are:

MaxActive	Maximum number of active connections to the database
Url	Database URL
Idle	Number of idle database connections
ActiveConnectionsCount	Number of active database connections
MaxIdle	Maximum number of idle database connections allowed
MaxWait	Timeout in ms when attempting to get a free connection
MinIdle	Minimum number of idle database connections to maintain

Regular Maintenance Operations

Overview

Artifactory provides several facilities allowing you to maintain your system for optimal performance.

To configure your global system maintenance, in the **Admin** module select **Advanced | Maintenance**.

Artifactory Online Users

JFrog manages regular maintenance operations for all instances of Artifactory Online. If you are an Artifactory Online user, the features described on this page are monitored and optimally managed for you by JFrog Artifactory Online administrators.

Page Contents

- Overview
- Garbage Collection
- Storage Quota Limits
- Cleanup Unused Cached Artifacts
- Cleanup Virtual Repositories
- Storage

Garbage Collection

Garbage Collection

Cron Expression*	Next Garbage Collection Time
<input type="text" value="0 0 /4 * * ?"/>	Wed Jul 08 16:00:00 UTC 2015
<button>Run Now</button>	

Artifactory uses checksum-based storage to ensure that each binary file is only stored once.

When a new file is deployed, Artifactory checks if a binary with the same checksum already exists and if so, links the repository path to this binary. Upon deletion of a repository path, Artifactory does not delete the binary since it may be used by other paths. However, once all paths pointing to a binary are deleted, the file is actually no longer being used. To make sure your system does not become clogged with unused binaries, Artifactory periodically runs a "Garbage Collection" to identify unused ("deleted") binaries and dispose of them from the datastore. By default, this is set to run every 4 hours and is controlled by a [cron](#) expression.

For example, to run garbage collection every 12 hours you should specify the following expression:

```
0 0 /12 * * ?
```

Cron Expression	Specifies the frequency in which garbage collection should be run automatically
Next Garbage Collection Time	Indicates the next automatic run of garbage collection according to the specified Cron Expression
Run Storage Garbage Collection	Manually invokes garbage collection immediately

Garbage collection frequency

Garbage collection is a resource intensive operation. Running it too frequently may compromise system performance.

Storage Quota Limits

Storage Quota

Enable Quota Control

Storage Space Limit (Percentage)*

95

Storage Space Warning (Percentage)*

85

Artifactory lets you set a limit on how much of your entire system disk space storage may be used to ensure that your server file system capacity is never used up. This helps to keep your system reliable and available.

Once disk space used for storage reaches the specified limit, any attempt to deploy a binary is rejected by Artifactory with a status code of [413 Request Entity Too Large](#) and a "Datastore disk space is too high" error is displayed at the bottom of the [Maintenance](#) screen.

When using filesystem storage, the partition checked is the one containing the `$ARTIFACTORY_HOME/data/filestore` directory.
When using database blob storage, the partition checked is the one containing the `$ARTIFACTORY_HOME/data/cache` directory.

To help you avoid reaching your disk space quota, Artifactory also allows you to specify a warning level. Once the specified percentage of disk space is used, Artifactory will log a warning in the `$ARTIFACTORY_HOME/logs/artifactory.log` file and display a "Datastore disk space is too high" warning at the bottom of the **Maintenance** screen.

Enable Quota Control	When set, Artifactory will monitor disk space usage and issue warnings and errors according to the quotas specified in Storage Space Limit and Storage Space Warning
Storage Space Limit	The percentage of available disk space that may be used for storage before Artifactory rejects deployments and issues errors
Storage Space Warning	The percentage of available disk space that may be used for storage before Artifactory issues warnings

Cleanup Unused Cached Artifacts

Cleanup Unused Cached Artifacts

Cron Expression*

```
0 12 5 * * ?
```

Next Garbage Collection Time

Thu Jul 09 05:12:00 UTC 2015

Run Unused Cached Artifacts Cleanup

When configuring a remote repository, the [Keep Unused Artifacts](#) setting lets you specify how long a cached unused artifact from that repository should be kept before it is a candidate for cleanup. This setting does not immediately clean up the unused cached artifact, but merely marks it for clean up after the specified number of hours. The **Cleanup Unused Cached Artifacts** setting specifies when the cleanup operation should run, and only then unused, cached artifacts marked for cleanup are actually removed from the system.

The cleanup frequency is specified with a [cron](#) expression. For example, to run cleanup every 12 hours you should specify the following expression:

```
0 0 /12 * * ?
```

Cleanup Virtual Repositories

Cleanup Virtual Repositories

Cron Expression*

```
0 12 5 * * ?
```

Next Garbage Collection Time

```
Tue Jun 02 05:12:00 UTC 2015
```

Clean Virtual Repositories Now

Virtual repositories use an internal cache to store aggregated metadata such as POM files. The Cleanup Virtual Repositories operation deletes cached POM files that are older than 168 hours (one week)

The cleanup frequency is specified with a [cron](#) expression. For example, to run cleanup every 12 hours you should specify the following expression:

```
0 0 /12 * * ?
```

Storage

Storage

Compress the Internal Database

Prune Unreferenced Data

**Compress
the Internal
Database**

Derby database only

This feature is only relevant when using the internal Derby database

A Derby database may typically contain unused allocated space when a large amount of data is deleted from a table or its indices are updated. By default, Derby does not return unused space to the operating system. For example, once a page has been allocated to a table or index, it is not automatically returned to the operating system until the table or index is destroyed.

When you invoke this action, Artifactory reclaims unused and allocated space in a table and its indexes thereby compressing the internal database.

We recommend running this when Artifactory activity is low, since compression may not be able to complete when storage is busy (in which case the storage will not be affected).

**Prune
Unreferenced
Data**

Unreferenced binary files may occur due to running with wrong file system permissions on storage folders, or running out of storage space.

When you invoke this action, Artifactory removes unreferenced binary files and empty folders present in the filestore or cache folders.

Ensure complete shutdown

To avoid such errors, we recommend that you always allow Artifactory to shut down completely

Managing Backups

Complete System Backup

You can automatically and periodically backup the entire Artifactory system. The backup process creates a time-stamped directory in the target backup directory.

To define multiple backups, in the **Admin** module, select **Services | Backups**. Each backup may have its own schedule and repositories to either process or exclude.

Page Contents

- Complete System Backup

- Restoring a Backup

Backup content is stored in standard file system format and can be loaded into any repository, so that Artifactory never locks you out.

In the Backups page you may select an existing **Backup** to edit, or click "New" to create a new **Backup**

Edit backup-daily Backup

Backup Settings

Enabled

Backup Key *

Cron Expression * ⓘ

Next Backup Time

Thu Nov 12 02:00:00 UTC 2015

Server Path For Backup ⓘ

Advanced

Send Mail to Admins if there are Backup Errors ⓘ

Exclude Builds

Exclude new Repositories

Retention Period Hours ⓘ

Incremental

Excluded Repositories	Included Repositories
repo1	deb-local
debian-flat	debian-new
debian-local	vagrant-local
docker-local	github
dockerv1	nuget-remote
ext-release-local	simple-docker-remote
ext-snapshot-local	smart-remote
...	...

Back up to a Zip Archive (Slow and CPU intensive)

Backup Key

A unique logical name for this backup

Cron Expression	A valid Cron expression that you can use to control backup frequency. For example, to back up every 12 hours use a value of: 0 0 /12 * * ?
Next Time Backup	When the next backup is due to run
Server Path for Backup	<p>The directory to which local repository data should be backed up as files The default is <code>\$ARTIFACTORY_HOME/backup/[backup_key]</code></p> <p>Each run of this backup will create a new directory under this one with the time stamp as its name.</p>
Sent Mail to Admins of there are Backup Errors	If set, all Artifactory administrators will be notified by email if any problem is encountered during backup.
Exclude Builds	Exclude all builds from the backup.
Retention Period	<p>The number of hours to keep a backup before Artifactory will clean it up to free up disk space. Applicable only to non-incremental backups.</p> <div style="border: 1px solid red; padding: 10px;"> <p>Do not store any custom files under the target backup directory, since the automatic backup cleanup processes may delete them!</p> </div>
Incremental	<p>When set, this backup should be incremental. In this case, only changes from the previous run will be backed up, so the process is very fast.</p> <p>The backup directory name will be called <code>current</code> (as opposed to using the timestamp)</p> <p>The backup files can be used by any incremental file-system based backup utility (such as rsync).</p>
Backup to a Zip Archie (Slow and CPU Intensive)	If set, backups will be created within a Zip archive

Restoring a Backup

To restore a system backup you need perform a system import. For details please refer to [System Import and Export](#).

Importing and Exporting

Overview

Artifactory supports import and export of data at two levels:

- [System level](#)
- [Repository level](#)

At **system level**, Artifactory can export and import the whole Artifactory server: configuration, security information, stored data and metadata. The format used is identical to the [System Backup](#) format. This is useful when manually running backups and for migrating and restoring a complete Artifactory instance (as an alternative to using database level backup and restore).

At **repository level**, Artifactory can export and import data and metadata stored in a repository. This is useful when moving store data, including its metadata between repositories and for batch population of a repository.

Page Contents

- [Overview](#)
- [System Import and Export](#)
- [Repositories Import and Export](#)
 - [Export](#)
 - [Import](#)
 - [Import Layout](#)

System Import and Export

To access import and export of your entire system, in the **Admin** module, select **Import & Export | System**

System Import & Export

Export System

Target Export Dir*

 c:\work Exclude Content Exclude Metadata Exclude Builds Create .m2 Compatible Export Create a Zip Archive (Slow and CPU Intensive!) Output Verbose Log

Import System

This will wipe all existing Artifactory content - please make sure to back up first!

System Zip File or Directory*

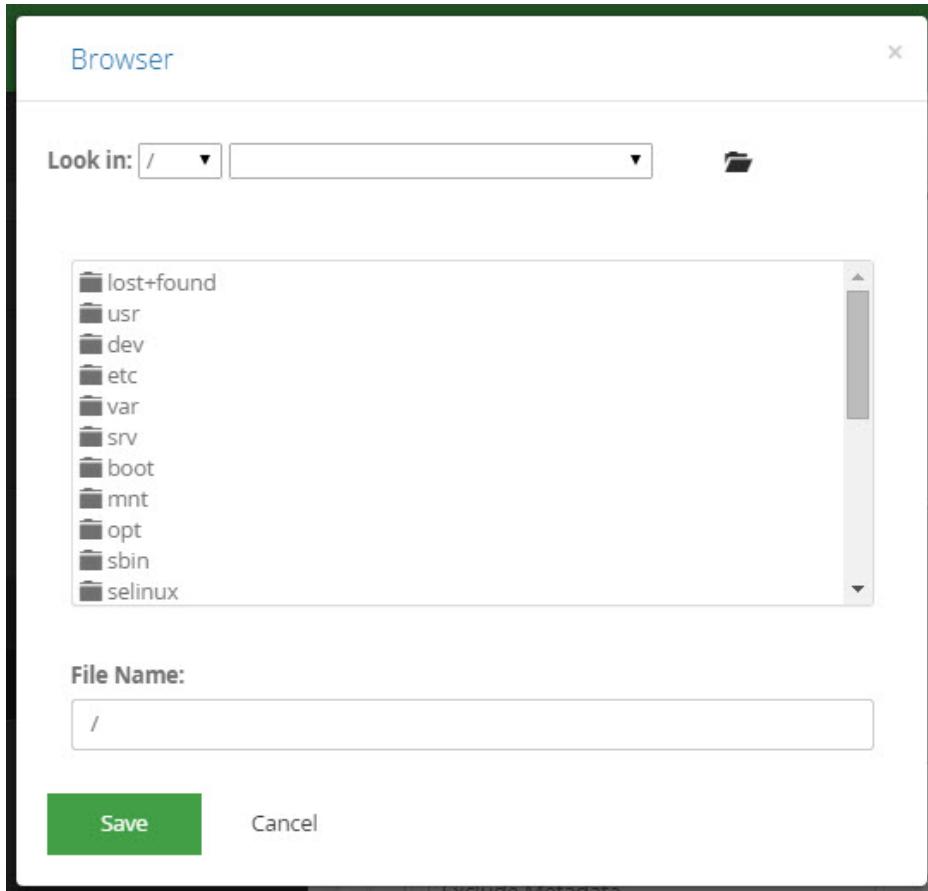
 c:\work\20150404.103948 Exclude Content Exclude Metadata Output Verbose Log

Target Export Dir	The target directory for the exported files. You may browse your file system to select the directory
Exclude Content	Export: When set, repository binaries are excluded from the export. Import: When set, binaries and metadata are excluded from the import. Only builds and configuration files are imported.

Exclude Metadata	When set, repository metadata are excluded from the import/export. (Maven 2 metadata is unaffected by this setting)
	<p>Docker repositories must have metadata</p> <p>For Docker repositories to work they must have their metadata intact. Therefore, if you have Docker repositories, make sure that Exclude Metadata is not checked when doing a system export or import.</p>
Exclude Builds	When set, all builds are excluded from the export
Create .m2 Compatible Export	When set, includes Maven 2 repository metadata and checksum files as part of the export
Create a Zip Archive (Slow and CPU Intensive!)	When set, creates and exports to a Zip archive
Output Verbose Log	When set, lowers the log level to "debug" and redirects the output from the standard log to the import-export log.
	<p>Monitoring the log</p> <p>You can monitor the log in the System Logs page.</p>

The source/target of the import/export operations are folders (Zip archives are not recommended) on the Artifactory server itself.

You can use the built-in server-side browsing inside Artifactory to select server-side source/target folders:



Importing or exporting a large amount of data may be time consuming. During the import/export operation you can browse away from the page and sample the [System Logs](#) to monitor progress.

Repositories Import and Export

To access import and export of repositories, in the **Admin** tab, select **Import & Export | Repositories**

Export

Export Repository to Path

Target Local Repository*

All Repositories

Export to Path*



Exclude Metadata

Create .m2 Compatible Export

Output Verbose Log

Export

When exporting, you need to specify the following parameters:

Source Local Repository	You can specify a single repository to export, or All Repositories
Export to Path	The export target directory on your server
Exclude Metadata	When set, repository metadata are excluded from the export.(Maven 2 metadata is unaffected by this setting)
Create .m2 Compatible Export	When set, includes Maven 2 repository metadata and checksum files as part of the export
Output Verbose Log	When set, lowers the log level to "debug" and redirects the output from the standard log to the import-export log. Monitoring the log You can monitor the log in the System Logs page.

Import

You can import repositories from a server side folder, or by zipping a repository and uploading it to Artifactory.

Import Repository from Path

Target Local Repository*

debian-local

Server Path For Import*

Exclude Metadata

Output Verbose Log

When importing, you need to specify the following parameters:

Target Local Repository	You can specify a single repository to import, or All Repositories . The repository layout should be different depending on your selection. Please refer to Import Layout
Server Path for Import	The import source directory on your server
Exclude Metadata	When set, repository metadata are excluded from the import
Output Verbose Log	When set, lowers the log level to "debug" and redirects the output from the standard log to the import-export log.

Monitoring the log

You can monitor the log in the [System Logs](#) page.

Don't exclude metadata for Docker

To work with a Docker repository, it must have its metadata intact. Therefore, when importing to/exporting from a Docker repository make sure that **Exclude Metadata** is not checked.

Importing into a Remote Repository Cache

You can take advantage of remote repositories you have already downloaded to your local environment, and import them directly into the cache of a remote repository.

For example, you can take your local Maven repository (usually located under `~/.m2`) and upload it into Artifactory so that all the artifacts you have already downloaded are now available on the server.

Import Layout

An imported repository needs to be formatted using a Maven 2 repository layout.

When importing a single repository, the file structure within the import folder (or zip file) should be as follows:

```
IMPORT_FOLDER/ZIP_FILE
|
| --LIB_DIR_1
```

When importing all repositories, the file structure within the import folder should be as follows:

```
IMPORT_FOLDER/ZIP_FILE
|
| --REPOSITORY_NAME_DIR_1
| |
| | --LIB_DIR_1
```

When importing all repositories, you need to ensure that the names of the directories representing the repositories in the archive match the names of the target repositories in Artifactory.

Managing Disk Space Usage

Overview

Artifactory includes features to help you manage the amount of disk space used by your system. This is done by providing alerts, limiting the amount of space allocated for the output of automatic procedures, and by cleaning up unused artifacts in a controlled manner.

Garbage Collection

When an Artifactory user "deletes" a file, what is actually deleted is the reference from the Artifactory database to the physical file. Before actually deleting a file Artifactory must scan the system to ensure that there are no other users referencing the file. Scanning the system is very CPU intensive, and locks files while the scan is in process, and this may stress the development environment. Therefore this can be scheduled to run periodically as a "Garbage Collection" process during times when demands on the system are low.

This is done in the Artifactory UI **Admin** module under **Advanced | Regular Maintenance Operations**, where you can schedule an automatic run of Garbage Collection with a Cron expression. You can also invoke an immediate run by clicking "Run Storage Garbage Collection".

Garbage Collection

Cron Expression*	Next Garbage Collection Time
<input type="text" value="0 0 /4 * * ?"/>	Wed Jul 08 16:00:00 UTC 2015
Run Now	

Page Contents

- Overview
 - Garbage Collection
 - Storage Quota
- Limiting the Number of Snapshots
- Deleting Unused Cached Artifacts
- Deleting Complete Versions
- User Plugins
- Manual Cleanup with the REST API
- Discarding old builds with Jenkins Artifactory plugin

Storage Quota

To avoid running out of disk space Artifactory allows you to limit the storage space allocated for your repositories.

In the **Admin** module, under **Advanced | Maintenance**, set **Enable Quota Control**, and specify **Storage Space Limit** to specify the percentage of disk space that you allocate for your repositories. An attempt to store binaries above the allocated storage percentage will fail with an error. Yo

You may also set **Storage Space Warning** to specify at what percentage of disk space usage to receive a warning from Artifactory.

Storage Quota

Enable Quota Control

Storage Space Limit (Percentage)*

95

Storage Space Warning (Percentage)*

85

Limiting the Number of Snapshots

Working with snapshots is a standard development practice, however depending on the number of snapshots that are saved, this can use up large quantities of disk space.

To specify the maximum number of snapshots that may be stored, select the **Repositories** module and click the repository whose settings you want to edit.

In the **Basic** settings, check **Handle Snapshots** and then set the **Max Unique Snapshots** field. This value is zero by default, which means that all snapshots are saved.

Maven Settings

Checksum Policy

Verify against client checksums

Maven Snapshot Version Behavior

Unique

Max Unique Snapshots

5

Handle Releases

Handle Snapshots

Suppress POM Consistency

To avoid issues of concurrency, Artifactory requires that you store a minimum of 2 unique snapshots, however you can control the maximum number of snapshots that are stored.

Redundant snapshots are not deleted immediately

Every 5 minutes, Artifactory runs a background process that checks the value of **Max Unique Snapshots** and deletes the oldest snapshots beyond that number. For example, if you currently have 7 snapshots and set **Max Unique Snapshots** to 5 then next time the background process runs, it will delete the 2 oldest snapshots.

Deleting Unused Cached Artifacts

When working with [remote repositories](#), to optimize performance, Artifactory locally caches and aggregates snapshots of remote artifacts that are being used. However, if at some point, these artifacts are no longer used, Artifactory can identify and remove them.

You can control how long an unused artifact will remain cached before it is eligible for cleanup. In the [Edit Repository](#) screen under **Advanced Settings**, specify the number of hours in the **Unused Artifacts Cleanup Period** field.

By default this value is set to zero which means that artifacts from the corresponding repository are never removed from the cache.

Cache

Unused Artifacts Cleanup Period

0

Retrieval Cache Period

43200

Assumed Offline Period

300

Missed Retrieval Cache Period

7200

Cleaning up unused cached artifacts can be scheduled to run automatically during times when demands on the system are low using a Cron expression in the **Admin** module under **Advanced | Maintenance**. You can also invoke an immediate run by clicking "Run Unused Cached Artifacts Cleanup"

Cleanup Unused Cached Artifacts

Cron Expression*

0 12 5 * * ?

Next Garbage Collection Time

Thu Jul 09 05:12:00 UTC 2015

Run Unused Cached Artifacts Cleanup

Recommended Frequency for Deleting Unused Cached Artifacts

Deleting unused cached artifacts is a resource-intensive operation, so to avoid concurrency and performance issues it is recommended to do it no more than once or twice a day, and preferably during "quiet time" such as outside of regular working hours.

Deleting Complete Versions

Artifactory supports a complete manual deletion of an installed version. This is fully described in [Deleting a Version](#).

User Plugins

Artifactory supports cleanup by allowing you to write custom [User Plugins](#) which you can develop to meet your own specific cleanup requirements.

JFrog provides a number of [cleanup scripts on GitHub](#) which you can use as provided or modify to suit your own needs. For example the following `artifactCleanup` plugin deletes artifacts that have not been downloaded for a specified number of months.

Manual Cleanup with the REST API

Using the Artifactory [REST API](#), you may write scripts to implement virtually any custom cleanup logic. This provides you with an extensive and flexible set of customization capabilities as provided by the REST API.

Examples:

- Use the REST API as described [Artifacts Not Downloaded Since](#), to identify artifacts that have not been downloaded since a specific Java epoch, and then remove them.
- Use the REST API as described in [Artifacts Created in Date Range](#) to identify artifacts created within a specific date range and then remove them.

Discarding old builds with Jenkins Artifactory plugin

When using Jenkins for continuous integration, you can configure a policy to discard old builds that are stored in Artifactory along with their artifacts.

For more details please refer to the [Artifactory Plugin](#) page of the Jenkins Wiki Documentation.

Getting Support

Overview

JFrog provides SLA based support for Silver, Gold and Enterprise licensing tiers. If you have purchased one of these tiers you may contact JFrog support through the JFrog Support Portal. In most cases, JFrog support will require some initial information about your system and relevant log files. In order to expedite handling of your issue, Artifactory lets you generate all the initially required information in the **Admin** module **Support Zone** screen. When opening a support ticket, you can attach the information bundle to expedite handling of your issue.

Artifactory OSS and Pro users

If you are running Artifactory on an OSS or Pro license, and therefore do not have access to JFrog Support Portal, you may visit [JFrog website support page](#) to access the Artifactory Community Forum (for OSS) or to submit a bug report (for Pro).

Availability

Support Zone is only available for Artifactory on-prem installations.

Page Contents

- [Overview](#)
- [Requesting Support](#)
- [Collecting an Information Bundle](#)
- [Previously Created Bundles](#)

Requesting Support

To request support, create an [information bundle](#) with the relevant information, and then login to [JFrog Support Portal](#) where you can open a support ticket and attach the information bundle.

What should I include?

Unless you are sure about the information JFrog support will need in order to address your issue, we recommend providing all items in the information bundle you upload.

Collecting an Information Bundle

The support zone provides a variety of options to select what information is included in the bundle you provide JFrog support.

Support Zone

Information To Collect

The support info package is not sent to JFrog support directly. Once you completed the download log in to JFrog ticket portal and open a relevant ticket.

System Info

Security Descriptor

Config Descriptor

Configuration Files

Storage Summary

Scrub Passwords and Private Information (?)

Thread Dump

Number of Thread Dumps: Interval (Milliseconds):

System Logs

Date Span:

Start Date: End Date:

Previously Created Bundles

Thu Nov 12 2015 11:48:38 GMT+0200 (Jerusalem Standard Time) (Latest)

Create

System info	If checked, provides information about your system including storage, system properties, JVM information and plugin status. For details please refer to System Information .
Security descriptor	If checked, provides information about how you have security configured in Artifactory. For details please refer to Security Configuration Descriptor .
Config descriptor	If checked, provider your Artifactory config descriptor which includes detailed information on how Artifactory and its repositories are configured. For details please refer to Global Configuration Descriptor .
Configuration files	If checked, provides configuration files that affect Artifactory's functionality.
Storage summary	If checked, provides information about your system's storage including binaries, file store, and repositories. For details, please refer to Monitoring Storage .
Scrub passwords and private information	If checked, passwords and private information such as email addresses are removed from all items in the information bundle.

<i>Thread dump</i>	If checked, Artifactory will create a thread dump for all running threads. By default a single thread dump is created, however, to get a picture of how data may change over time, you can request several thread dumps separated by a specified time interval with the Number of Thread Dumps and Interval fields.
<i>System logs</i>	If checked, system logs are included in the information bundle. You may specify the time span for which system logs should be included. <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> Date range Date range considers files according to the time stamp present in the file name, not by its contents. </div>

Once you have checked all the information items you wish to include in your information bundle, click "Create" to create the bundle.

Artifactory HA

When creating an information bundle for an Artifactory HA installation, the bundle is created by the specific HA node that happens to handle the "Create" request.

Resource intensive operations

Note that creating a **Thread dump** and **System logs** may be resource intensive operations and may create large information bundles.

Previously Created Bundles

Every information bundle you create is stored in Artifactory and is available for download under **Previously Created Bundles**.

Artifactory High Availability

Overview

From version 3.1, Artifactory supports a High Availability network configuration with a cluster of 2 or more, active/active, read/write Artifactory servers on the same Local Area Network (LAN).

This presents several benefits to your organization and is included with [Artifactory Pro Enterprise Value Pack](#).

Benefits

Maximize Uptime

Artifactory HA redundant network architecture means that there is no single-point-of-failure, and your system can continue to operate as long as at least one of the Artifactory nodes is operational. This maximizes your uptime and can take it to levels of up to "five nines" availability.

Manage Heavy Loads

By using a redundant array of Artifactory server nodes in the network, your system can accommodate

larger load bursts with no compromise to performance. With horizontal server scalability, you can easily increase your capacity to meet any load requirements as your organization grows.

Minimize Maintenance Downtime

By using an architecture with multiple Artifactory servers, Artifactory HA lets you perform most maintenance tasks with no system downtime.

Page Contents

- Overview
- Benefits
 - Maximize Uptime
 - Manage Heavy Loads
 - Minimize Maintenance Downtime
- Architecture
 - Network Topology
 - Load Balancer
 - Artifactory Server Cluster
 - Local Area Network
 - Shared Network File System
 - Database
- Watch the Screencast

Read more

- Installation and Setup
- Managing the HA Cluster
- Troubleshooting HA

Architecture

Artifactory HA architecture presents a Load Balancer connected to a cluster of two or more Artifactory servers that share a common database and Network File System. The Artifactory cluster nodes must be connected through a fast internal LAN in order to support high system performance as well as to stay synchronized and notify each other of actions performed in the system instantaneously. One of the Artifactory cluster nodes is configured to be a "master" node. Its roles are to execute cluster-wide tasks such as cleaning up unreferenced binaries.

JFrog support team is available to help you configure the Artifactory cluster nodes. It is up to your organization's IT staff to configure your load balancer, database and network file system.

Network Topology

Load Balancer

The load balancer is the entry point to your Artifactory HA installation and optimally distributes requests to the artifactory server nodes in your system.

Your load balancer must support session affinity (sticky sessions) and it is the responsibility of your organization to manage and configure it correctly.

The code samples below show some basic examples of load balancer configurations:

▼ Apache load balancer configuration example...

```

First install the following modules:
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_balancer_module
modules/mod_proxy_balancer.so
LoadModule proxy_http_module
modules/mod_proxy_http.so

Then configure as follows:
<VirtualHost *:80>
    ServerAdmin admin@test.com
    ServerName apache-ha-test
    ServerAlias apache-ha-test.jfrog.local

    Header add Set-Cookie
    "ROUTEID=. %{BALANCER_WORKER_ROUTE}e;
    path=/artifactory/" env=BALANCER_ROUTE_CHANGED
    <Proxy balancer://tomcats>

        # Artifactory server #1
        BalancerMember http://10.0.0.32:8081
        route=art1

        # Artifactory server #2
        BalancerMember http://10.0.0.33:8081
        route=art2

    ProxySet lbmethod=byrequests
    ProxySet stickysession=ROUTEID
</Proxy>

    ProxyPreserveHost on
    ProxyPass /balancer-manager !
    ProxyPass / balancer://tomcats/

    RewriteEngine On
    RewriteRule ^/$ /artifactory
[R,L]

<Location /balancer-manager>
    SetHandler balancer-manager
    Order deny,allow
    Allow from 10.0.0 192.168.0
</Location>

    LogLevel warn
    ErrorLog
    /var/log/httpd/apache-ha-test.error.log
    CustomLog
    /var/log/httpd/apache-ha-test.access.log combined
</VirtualHost>
```

▼ nginx load balancer configuration example...

```

http {
...
...
...
upstream artifactory {
    ip_hash;                      # for stickiness by IP

    server IP_SERVER_1:8081;
    server IP_SERVER_2:8081;
}

server {
    listen 80;
    server_name YOUR_SERVER_NAME;
...
...
...
rewrite ^/$
http://$host/artifactory/webapp/login.html;
location / {
    proxy_pass http://artifactory;
}
}
}

```

More details are available on the [nginx website](#).

Artifactory Server Cluster

Each Artifactory server in the cluster receives requests routed to it by the load balancer. All servers share a common database and NFS mount, and communicate with each other to ensure that they are synchronized on all transactions.

Local Area Network

To ensure good performance and synchronization of the system, all the components of your Artifactory HA installation must be installed on the same high-speed LAN.

In theory, Artifactory HA could work over a Wide Area Network (WAN), however in practice, network latency makes it impractical to achieve the performance required for high availability systems.

Shared Network File System

Artifactory HA requires a shared file system to store cluster-wide configuration and binary files. Artifactory HA requires that your shared file system **concurrent requests**
and
file locking

Currently, the only shared file system that has been certified to work with Artifactory HA and is supported is **NFS (versions 3 and 4)**

Mounting the NFS from Artifactory HA nodes

When mounting the NFS on the client side, make sure to add the following option for the `mount`

command:

```
lookupcache=none
```

This ensures that nodes in your HA cluster will immediately see any changes to the NFS made by other nodes.

Database



Artifactory HA requires an external database, and currently supports MySQL, Oracle, MS SQL and PostgreSQL. For details on how to configure a database, see [Changing the Default Storage](#).

Since Artifactory HA contains multiple Artifactory cluster nodes, your database must be powerful enough to serve all of them.

If you are replicating your database you must ensure that at any given point in time all nodes see a consistent view of the data. Artifactory HA supports eventual consistency, and write-behind database synchronization is not supported.



Installation and Setup

Overview

This page describes how to set up a set of Artifactory nodes as an Artifactory HA system.

Each of the HA components is configured individually and a common setup file is used to bring together all of the components in the system as a whole.

Requirements

Version

Artifactory HA is supported from Artifactory 3.1 and above. If you are running a previous version of Artifactory, you first need to upgrade as described in [Upgrading Artifactory](#).

All nodes within the same Artifactory HA installation must be running the same Artifactory version and the same JVM version.

Licensing

Artifactory HA is provided as a feature of the [Artifactory Pro Enterprise Value Pack](#) with licenses for a set number of cluster nodes.

When setting up Artifactory HA you need to install a different license on each of the Artifactory nodes in the cluster.

If you have more cluster nodes than the number of licenses provided you may purchase additional Artifactory HA licenses.

Hardware

Artifactory HA requires the following hardware:

- Load balancer with session affinity (sticky session)
- NFS (Network File System)
- External database server with a single URL to the database

Network

- All the Artifactory HA components (Artifactory cluster nodes, database server, NFS server and load balancer) must be within the same fast LAN
- All the HA nodes must communicate with each other through dedicated TCP ports

Database

Artifactory HA requires an external database and currently supports Oracle, MySQL, MS SQL and PostgreSQL. For details on how to configure any of these databases please refer to [Changing the Default Storage](#).

Page Contents

- Overview
- Requirements
- Home Directories
- Configuring Artifactory HA
 - Configuring the Cluster
 - Installing a Cluster Node
 - New installation
 - Upgrading a current installation of Artifactory Pro to Artifactory HA
 - Configuring a Cluster Node
- Testing Your HA Configuration
- Upgrading Artifactory HA
 - Upgrading from Any Version below 3.5
 - Upgrading from Version 3.5+

Home Directories

When setting up Artifactory HA you need to configure the `$ARTIFACTORY_HOME` directory separately for each of the Artifactory cluster nodes in

your system, and a common `$CLUSTER_HOME` directory that is found on the NFS.

The general layout of these directories is as follows:

```
/- $ARTIFACTORY_HOME
  |- etc/
    |- ha-node.properties
    |- logback.xml
    |- artifactory.lic
  |- data/
    |- tmp/
    |- artifactory.properties
  |- logs/
  |- bin/
  |- misc/
  |- webapps/
  |- tomcat/
    |- lib/
    |- <jdbc driver>

/- $CLUSTER_HOME
  |- ha-etc/
    |- cluster.properties
    |- storage.properties
    |- -
  artifactory.system.properties
    |- mimetypes.xml
    |- ui/
    |- plugins/
  |- ha-data/
    |- filestore/
    |- tmp/
    |- artifactory.properties
  |- ha-backup/
```

artifactory.system.properties and ha-node.properties

Note that the `artifactory.system.properties` file under `$ARTIFACTORY_HOME/etc` folder should be replaced with an `ha-node.properties` file. The `artifactory.system.properties` is moved to the `$CLUSTER_HOME/ha-etc` folder.

Privileges

Each of the Artifactory cluster nodes must have full write privileges on the `$CLUSTER_HOME` directory tree.

Configuring Artifactory HA

For Artifactory HA to operate properly you need to ensure that the cluster is configured correctly as a shared system, and that each specific server is configured correctly as a node in the system.

To ensure correct configuration of your system at each step, we recommend the following procedure:

1. **Configure the cluster**

Then, for each cluster node in your system do the following:

2. **Install the cluster node**

3. **Configure the cluster node**

4. **Test your HA configuration**

Configuring the Cluster

You need to create a shared `$CLUSTER_HOME` directory on your NFS storage which is visible and writable to all the Artifactory cluster nodes in your system.

The contents of this shared directory are as follows:

ha-etc	Shared configuration files for the cluster, including the backend storage and information on all the Artifactory cluster nodes in your system
ha-data	Shared data for the cluster. Among other things, this directory contains the filestore when using Artifactory in db-filesystem mode.
ha-backup	Shared backup directory for automatic backups performed from one of the Artifactory cluster nodes.

You need to manually create the following two files for the shared cluster configuration:

<code>\$CLUSTER_HOME/ha-etc/cluster.properties</code>	Configuration parameters that are shared by all of the Artifactory cluster nodes
<code>\$CLUSTER_HOME/ha-etc/storage.properties</code>	<p>Identical to the <code>storage.properties</code> file used in a regular Artifactory installation.</p> <p>This file replaces the <code>storage.properties</code> file in each individual cluster node since each node reads the properties from this file in the common location under <code>\$CLUSTER_HOME</code>.</p> <p>The file is created in the right location when you configure the first Artifactory cluster node in your system and is fully described below.</p>

The `cluster.properties` file contains the following property:

<code>security.token=<your_selected_token></code>	An ASCII string token that you select, which is used to send secured messages between the servers. This can be any string you choose (like a password).
---	---

For example, a `cluster.properties` file for an Artifactory HA installation could be:

```
security.token=76b07383dcda344979681e01efa5ac50
```

Uniqueness

Make sure that each cluster has a unique security token.

Installing a Cluster Node

As mentioned in the [Requirements](#) section above Artifactory HA is supported from Artifactory 3.1 and above.

Whether you are performing a new installation or upgrading a current one, we recommend that you have all of your Artifactory cluster nodes installed and fully functional with Artifactory 3.1 or above as separate servers before configuring them to be a part of the HA cluster.

We recommend that you complete the configuration of each Artifactory cluster node, and its integration into your HA cluster as described below before going on to the next node.

New installation

For a new installation, simply install Artifactory 3.1 as described in [Installing Artifactory](#).

For the first node that you install you need to move the `storage.properties` file from your server's `$ARTIFACTORY_HOME/etc/` directory to your `$CLUSTER_HOME/ha-etc/` directory.

Upgrading a current installation of Artifactory Pro to Artifactory HA

1. Verify that the Artifactory server on the current installation you are upgrading is shut down
2. To upgrade your current installation to Artifactory 3.1 and above, please refer to [Upgrading Artifactory](#)
3. Copy `$ARTIFACTORY_HOME/etc/` and `$ARTIFACTORY_HOME/data/` from our current installation to the corresponding locations under `$CLUSTER_HOME`.
You only need to do this once, so for any subsequent servers that you upgrade to configure into your Artifactory HA system, this step can be omitted.

JDBC driver

You should also verify that your database JDBC driver is correctly located in `$ARTIFACTORY_HOME/tomcat/lib/` for each Artifactory cluster node.

Configuring a Cluster Node

1. Shut down the Artifactory cluster node.
2. Create an `$ARTIFACTORY_HOME/etc/ha-node.properties` file and populate it with the following parameters (you can use `$ARTIFACTORY_HOME/mischa/ha-node.properties.template` as a template to define your `ha-node.properties` file):

node.id	Unique descriptive name of this server. Uniqueness Make sure that each node has an id that is unique on your whole network.
cluster.home	The location of <code>\$CLUSTER_HOME</code> that you set up on your NFS.

context.url	The context url that should be used to communicate with this server within the cluster. <div style="border: 1px solid #ccc; padding: 10px; background-color: #fff;"><p>Use an explicit IP address The host must be explicitly defined as an IP address and not as a host name.</p></div>
membership.port	The port that should be used to communicate with this server within the cluster. If not specified, Artifactory will allocate a port automatically, however we recommend to set this to a fixed value to ensure that the port allocated is open to all of your organizations security systems such as firewalls etc.
primary	(Optional, true false) Indicates if this is the primary server. There must be one (and only one) server configured in the cluster to be the primary server.

For example, an `ha-node.properties` file for a server called `art1` connected to a mounted drive with `$CLUSTER_HOME` at `/mnt/shared/artifactory/clusterhome`, communicating with the other nodes through port 10001, and configured as the "primary" would be as follows:

```
node.id=art1
cluster.home=/mnt/shared/artifactory/clusterhome
context.url=http://10.0.0.121:8081/artifactory
membership.port=10001
primary=true
```

ha-node.properties file permissions

On Linux, once the `ha-node.properties` file is created, the Artifactory user should be set as its owner and its permissions should be set to `644 (-rw-r--r--)`

3. Set a valid Artifactory HA license in your `$ARTIFACTORY_HOME/etc/artifactory.lic` file. If this file exists (from a previous installation of Artifactory Pro), then simply replace your Pro license with the Artifactory HA license, otherwise create the file and populate it with your HA license.
 4. Mount the `$CLUSTER_HOME` directory as defined in the `cluster.home` property of `$ARTIFACTORY_HOME/etc/ha-node.properties`
 5. [Test your HA configuration](#) after each cluster node that you add to your system.
 6. After you have installed the first cluster node and verified that your system is working correctly as an HA installation, you should configure the **Custom URL Base**.
In the **Admin** tab under **Configuration | General**, set the **Custom URL Base** field to the URL of the Load Balancer.

Once you have successfully installed, configured and tested a cluster node you may go on to the next one that you want to add to your system.

Testing Your HA Configuration

The following are a series of tests you can do to verify that your system is configured correctly as an HA installation:

1. Directly Access the Artifactory UI for the server you have just configured
 2. In the **Admin** module go to **Advanced | System Logs** to view the log and verify that you see an entry for **Artifactory Cluster Home**.

3. The bottom of the module navigation bar should also indicate that you are running with Artifactory HA. In case of an error you will see an error message in the page header.



4. Access Artifactory through your load balancer and log in as **Admin**.
5. In the **Home** module, you should see that Artifactory HA is Available.

6. In the **Admin** module go to **Configuration**. There should be a section called **High Availability**. When selected you should a table with details on all the Artifactory nodes in your cluster as displayed below.

ID	Start Time	URL	Membership Port	State	Role	Last Heartbeat	Version	Revision	Release Date
ha_artifactory_1_1	22-07-15 10:12:39 UTC	http://172.17.0.50:8081/artifactory	10042	Running	Primary	22-07-15 10:13:19 UTC	4.0.0	1677	22-07-15 09:45:32 UTC
ha_artifactory_1_2	22-07-15 10:13:17 UTC	http://172.17.0.20:8081/artifactory	10042	Running	Member	22-07-15 10:13:21 UTC	4.0.0	1677	22-07-15 09:45:32 UTC
ha_artifactory_1_3	22-07-15 10:13:17 UTC	http://172.17.0.21:8081/artifactory	10042	Running	Member	22-07-15 10:13:20 UTC	4.0.0	1677	22-07-15 09:45:32 UTC

7. In the **Admin** module under **Configuration | General**, verify that the **Custom URL Base** field is correctly configured to the URL of the Load Balancer.

Upgrading Artifactory HA

From version 3.5, the procedure for upgrading Artifactory HA changed

The sections below provide upgrade instructions according to your current version, and assume you are upgrading to the **latest version**.

Upgrading from Any Version below 3.5

Upgrading Artifactory HA below Version 3.5 requires shutting down all of your Artifactory HA nodes, upgrading and restarting the Master node, and then upgrading and restarting the slaves one node at a time as follows:

Configure an explicit IP in your context.url

Before continuing with your upgrade, make sure you have configured the `context.url` of your cluster nodes with an explicit IP address.

1. Shutdown all of your Artifactory HA nodes one at a time
2. Upgrade the Master node using the regular procedure described in [Upgrading Artifactory](#), and then restart it.
3. Upgrade and restart all the other Slave nodes one at a time using the regular procedure described in [Upgrading Artifactory](#).

Upgrading from Version 3.5+

If your current version is 3.5 or higher, upgrading an HA cluster is done by first upgrading the master node, restarting it, and then upgrading the rest of the slave nodes one at a time.

At any time, at least one Artifactory node continues to run which means that there is no disruption of service.

This zero-downtime upgrade process should be executed as follows:

1. Perform a graceful shutdown of the Artifactory master node. While the master node is down, the load balancer should redirect all queries to the slave nodes.
2. Upgrade the master node using the regular procedure described in [Upgrading Artifactory](#).
3. Restart the master node. When the master starts up, it recognizes that the HA cluster nodes are not all running the same version of Artifactory, and consequently the system is limited to allowing uploads and downloads. Any attempt to perform other actions such as changing the DB schema, modifying permissions, changing repository configuration and more, are strictly blocked. This limitation will continue until all the cluster nodes are once again running the same version.

Version inconsistency generates exceptions

Running the HA cluster nodes with different versions generates exceptions which can be seen in the log files and reflect the temporary inconsistent state during the upgrade process. This is normal and should be ignored until all the cluster nodes are once again running the same version.

4. For each slave node
 - a. Perform a graceful shutdown of Artifactory,
 - b. Upgrade it using the regular procedure described in [Upgrading Artifactory](#).
 - c. Restart the node

Once all nodes have been upgraded to the same version, your Artifactory HA installation will be fully functional again.

Managing the HA Cluster

Overview

You can view the status of, and manage your HA cluster nodes in the **Admin** module under **Configuration | High Availability**.

This screen displays a table with details on all the Artifactory nodes in your cluster as displayed below:

All Artifactory Nodes								
ID	Start Time	URL	Membership Port	State	Role	Last Heartbeat	Version	Revision
ha_artifactory_1_1	22-07-15 10:13:39 UTC	http://172.17.0.19:8081/artifactory	10042	Running	Primary	22-07-15 10:13:19 UTC	4.0.0	1677
ha_artifactory_1_2	22-07-15 10:13:17 UTC	http://172.17.0.20:8081/artifactory	10042	Running	Member	22-07-15 10:13:17 UTC	4.0.0	1677
ha_artifactory_1_3	22-07-15 10:13:17 UTC	http://172.17.0.21:8081/artifactory	10042	Running	Member	22-07-15 10:13:20 UTC	4.0.0	1677

Page Contents

- Overview
- Monitoring for Unresponsive Nodes
- Removing an Unused Node

The table columns are as follows:

ID	Unique descriptive name of the server.
Start Time	The time that the server was started .
URL	The context URL that should be used to communicate with this server within the cluster.
Membership Port	The port that should be used to communicate with this server within the cluster.
State	<p>The current state of the server as follows:</p> <ul style="list-style-type: none"> • Offline - The node has started in an invalid state (For example, same HA license, or the same node id has been set on two different nodes). In this case you should the specific server logs for details. • Starting - The node is starting up. • Running - The node is up and running. This is the normal state for a node. • Stopping - The node is in the process of shutting down. • Stopped - The node is shut down. • Converting - The node is converting database tables and configuration files.
Role	<p>The role of the server as follows:</p> <ul style="list-style-type: none"> • Primary - The primary node. • Member - A regular member node. • Standalone - The node is not configured into your HA cluster. It is running as a separate installation of Artifactory (Pro or Open Source).
Last Heartbeat	The last time this server signaled that it is up and running. By default, each node signals every 5 seconds.
Version	The Artifactory version running on this cluster node.
Revision	The Artifactory revision number running on this cluster node.
Release	The Artifactory release running on this cluster node.

You can use the **Last Heartbeat** field to identify unresponsive nodes. If a node abruptly stops working (e.g. system crash on the server), then it may not be able to correctly update its **State** value, and will continue to appear as **Running**.

However, since the server Heartbeat does not get updated for a long interval of time, it is displayed in red with a warning sign as displayed below.

In this case you should check that the corresponding server is up and running and fully connected to your HA cluster and the database.

Configure High Availability										
All Artifactory Nodes										
ID	Start Time	URL	Membership Port	State	Role	Last Heartbeat	Version	Revision	Release Date	
ha_artifactory_1_1	22-07-15 10:12:39 UTC	http://172.17.0.19:8081/artifactory	10042	Running	Primary	22-07-15 10:15:24 UTC	4.0.0	1677	22-07-15 09:45:32 UTC	
ha_artifactory_1_2	22-07-15 10:14:23 UTC	http://172.17.0.20:8081/artifactory	10042	Running	Member	⌚22-07-15 10:14:23 UTC	4.0.0	1677	22-07-15 09:45:32 UTC	
ha_artifactory_1_3	22-07-15 10:13:17 UTC	http://172.17.0.21:8081/artifactory	10042	Running	Member	22-07-15 10:15:25 UTC	4.0.0	1677	22-07-15 09:45:32 UTC	

Removing an Unused Node

If you remove a node from your cluster, it will still appear in your database and will therefore be displayed in the list of cluster nodes.

To avoid confusion you should remove it from your list of cluster nodes (and detach it from the database) so that it doesn't interfere with normal cluster behavior.

To do so, hover over the corresponding server from the list and click "Delete".

Configure High Availability										
All Artifactory Nodes										
ID	Start Time	URL	Membership Port	State	Role	Last Heartbeat	Version	Revision	Release Date	
ha_artifactory_1_1	22-07-15 10:12:39 UTC	http://172.17.0.19:8081/artifactory	10042	Running	Primary	22-07-15 10:15:24 UTC	4.0.0	1677	22-07-15 09:45:32 UTC	
ha_artifactory_1_2	22-07-15 10:14:23 UTC	http://172.17.0.20:8081/artifactory	10042	Running	Member	⌚22-07-15 10:14:23 UTC	4.0.0	1677	22-07-15 09:45:32 UTC	
ha_artifactory_1_3	22-07-15 10:13:17 UTC	http://172.17.0.21:8081/artifactory	10042	Running	Member	22-07-15 10:15:25 UTC	4.0.0	1677	22-07-15 09:45:32 UTC	

When to remove a node

The "Remove" button is only available once a node has not signaled a Heartbeat for a "long" time.

We recommend that you only remove a node from your list if it has indeed been removed from your system.

In case of an error in one of the nodes, or if a node is shut down for maintenance, the **Heartbeat** will not be updated and Artifactory will alert you to this as described in the previous section.

In this case there is no need to remove the node from your list. Once you fix the error and restart the server the **Heartbeat** will be updated and the warning will be dismissed.

Troubleshooting HA

Artifactory Does Not Start Up

▼ **There are no log file entries in \$ARTIFACTORY_HOME/logs/artifactory.log**

Cause	Something within your <code>\$ARTIFACTORY_HOME</code> or <code>\$CLUSTER_HOME</code> directory is either not defined, or misconfigured
Resolution	In some cases in which the <code>\$ARTIFACTORY_HOME</code> directory tree is not validly constructed, log file entries are written to <code>\$ARTIFACTORY_HOME/logs/catalina/localhost/<date>/logs</code> . Check the contents of this file to see which specific errors were logged.

▼ **The log says "Stopping Artifactory start up ,another server running converting process".**

Cause	You are upgrading more than one server at a time.
Resolution	When upgrading your system, make sure you complete the upgrade process on one server before starting to upgrade the next one.

▼ **The log says "Stopping Artifactory start up ,another server with different version has been found".**

Cause	You are trying to install different versions of Artifactory into the same system.
Resolution	Make sure that all the instances of Artifactory installed in your system are the same version.

Page Contents

- Artifactory Does Not Start Up
- Artifactory Starts Up But Remains Offline
- Artifactory Starts Up But Not as an HA Installation

▼ **The log says "Stopping Artifactory since duplicate node ids have been found in registry. If you restarted this server, make sure to wait at least 30 seconds**

before re-activating it"

Cause	This may happen in one of two cases: <ol style="list-style-type: none">1. Two servers are configured into your system with the same <i>node.id</i> specified in the <code>\$ARTIFACTORY_HOME/etc/ha-node.properties</code> file.2. You have shut down an Artifactory server and tried to restart it within 30 seconds.
Resolution	Make sure that all servers within your Artifactory HA installation have a unique <i>node.id</i> value. Shut down your server and wait at least 30 seconds before you restart it.

Artifactory Starts Up But Remains Offline

▼ **The log says "Changing Artifactory mode to offline since the server is configured as HA but the license does not exist or is not an HA License"**

Cause	Your server does not have a valid HA license installed
Resolution	Install a valid HA license in your server and restart it

▼ **The log says "Changing Artifactory mode to offline since the local server is running as HA but found no HA server in registry."**

Cause	You are starting an Artifactory server as an HA installation, however you already have an Artifactory Pro (or OSS version) running within the same system.
Resolution	Make sure your system is consistent - either you have a set of Artifactory Pro instances running separately, or you all of your servers are configured as Artifactory HA

▼ **The log says "Could not find cluster properties"**

Cause	Your <code>\$CLUSTER_HOME/ha-etc/cluster.properties</code> file is not defined.
--------------	---

Resolution

When installing Artifactory HA, you need to manually create a `$CLUSTER_HOME` directory and the `$CLUSTER_HOME/ha-etc/cluster.properties` file. For details please refer to [Configuring the Cluster](#).

Artifactory Starts Up But Not as an HA Installation**Artifactory starts up as an instance of Artifactory Pro****Cause**

You have not created a valid `$ARTIFACTORY_HOME/etc/ha-node.properties` file

Resolution

Create a valid `$ARTIFACTORY_HOME/etc/ha-node.properties` file as described in [Configuring a Cluster Node](#).

Bintray Integration

Bintray is JFrog's platform for storage and distribution of software libraries on the cloud. It is the new way for developers to publish, download and share software across one unified community around the world. The free, cloud-based platform empowers developers to control and streamline the entire process of making software libraries publicly available, with all the services needed to collaborate, advertise and deploy a new software solution.

Naturally, Artifactory integrates with Bintray in more than one way:

- Remote Search in Bintray's JCenter repository - the most comprehensive collection of Maven artifacts.
- Information insight from Bintray on artifacts - package description and latest released version.
- Pushing artifacts to Bintray (one by one or a whole build outcomes).
- Complete continuous delivery stack for selected OSS projects based on oss.jfrog.org and Bintray.

Read More

- [Bintray info panel](#)
- [Push to Bintray](#)
- [Deploying Snapshots to oss.jfrog.org](#)

Bintray info panel

Overview

As part of Artifactory's integration with Bintray information about components stored in Bintray is fetched and displayed in the Tree Browser under **Package Information**.

To view Bintray Package Information:

- You need to be logged in
- You need to have the Bintray user and API Key configured in your [Artifactory profile](#).
- The selected file type is supported (e.g., pom, jar, war, ear)

- Overview

 [artifactory-papi-2.6.4-javadoc.jar](#) ↳ Download 

General
[Effective Permissions](#)
[Properties](#)
[Watchers](#)
[Builds](#)
[Governance](#)

Info

Name:	artifactory-papi-2.6.4-javadoc.jar
Repository Path:	libs-release-local/org/artifactory/artifactory-papi/2.6.4/artifactory-papi-2.6.4-javadoc.jar 
Module ID:	org.artifactory:artifactory-papi:2.6.4:javadoc
Deployed by:	admin
Size:	363.49 KB
Created:	30-04-15 11:30:59 UTC 
Last Modified:	27-09-12 21:14:33 UTC
Licenses:	Not Found Add Scan Search Archive License File
Downloaded:	2
Last Downloaded:	15-07-15 14:28:43 UTC
Last Downloaded By:	admin
Watching Since:	04-05-15 15:28:42 UTC
<input type="checkbox"/> Filtered 	

Package Information

Name:	org.artifactory:artifactory-papi	
Latest Version:	3.9.2	

Push to Bintray

Overview

Bintray is a JFrog's Distribution as a Service platform through which you can freely share your release binaries with the world. Artifactory allows you to upload artifacts directly to Bintray [using the UI](#), or with the [REST API](#). This page describes the process of pushing a single artifact, a complete release build, or an arbitrary set of files as a version from Artifactory to [Bintray](#). Once you have pushed your binaries, you need log in to your Bintray account and publish them in order to make them visible and available for download.

Under the hood Artifactory stores the information needed to push binaries to Bintray as the following [Properties](#):

Property	Description
<code>bintray.repo</code>	A target repository in Bintray, in the format of <code>{username}/{repository}</code>

<i>bintray.package</i>	A target package name under the repository. You must first create the package in Bintray if it does not exist.
<i>bintray.version</i>	A target version under the package. If the version does not yet exist in Bintray, it is created automatically.
<i>bintray.path</i>	<p>A target path in the repository under which to save the file.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>The path is considered optional as Artifactory will use the same path the file is stored in the repository.</p> </div>

Add: [Property](#) [Property Set](#)

Name *	Value	Add										
<input type="checkbox"/> Recursive												
<input type="text" value="Filter by Property"/> ✖ Delete Delete Recursively < page 1 of 1 >												
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 90%;">Value(s)</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="radio"/> Property</td> <td></td> </tr> <tr> <td><input type="radio"/> bintray.package</td> <td>multi1</td> </tr> <tr> <td><input type="radio"/> bintray.repo</td> <td>user/myrepo</td> </tr> <tr> <td><input type="radio"/> bintray.version</td> <td>1.5.0.6</td> </tr> </tbody> </table>			Value	Value(s)	<input checked="" type="radio"/> Property		<input type="radio"/> bintray.package	multi1	<input type="radio"/> bintray.repo	user/myrepo	<input type="radio"/> bintray.version	1.5.0.6
Value	Value(s)											
<input checked="" type="radio"/> Property												
<input type="radio"/> bintray.package	multi1											
<input type="radio"/> bintray.repo	user/myrepo											
<input type="radio"/> bintray.version	1.5.0.6											

Usually, these properties will not exist on the artifact if it was not pushed before, using the UI above will attach them according to the user input.
 All of the properties can be pre-populated (for example by your build tool), in this case Artifactory will use any existing property and **will ignore** the user input from the UI unless the property doesn't exist.

Page Contents

- [Overview](#)
- [Configuring Push to Bintray](#)
 - [Entering your Bintray credentials](#)
 - [Pushing a Single Artifact](#)
 - [Push a Complete Build](#)
- [Using the REST API](#)
 - [Creating Repositories, Packages and Versions](#)
 - [Packages and Versions](#)
 - [Repositories](#)
 - [Pushing a Build](#)
 - [Pushing a Set of Files](#)

Configuring Push to Bintray

Entering your Bintray credentials

Before you start pushing artifacts to Bintray, you need to enter your **Bintray username** and **API key** in the profile page, see [Updating Your Profile](#).

Bintray Settings

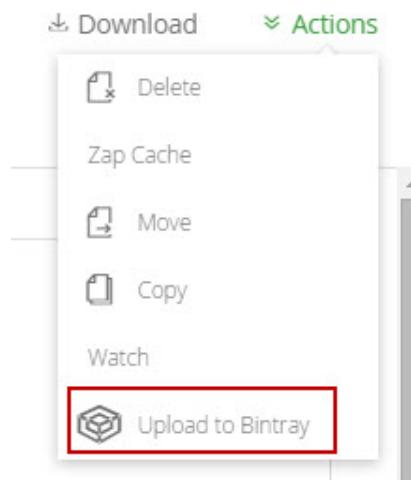
Bintray Username
admin

Bintray API Key
..... 

Test

Pushing a Single Artifact

To push a single artifact to Bintray, select the artifact in the **Artifact Repository Browser**, and select Upload to Bintray from the **Actions** droplist



Fill in the dialog details and click "Push"

Push To Bintray

X

Distribute this artifact to users by uploading it to [Bintray](#)

Bintray is a public online service through which you can share your release binaries with the world.
Note that once artifacts are pushed, you need to publish them in Bintray in order to make them world-visible.

Bintray Repository*

jfrog/jfrog-jars

▼

Bintray Package Name*

build-info-extractor-maven3

▼

Bintray Package Version*

2.1.0

▼

File Path*

org/jfrog/test/build-info-extractor-maven3/2.1.0.jar

Push

Push a Complete Build

Pushing a complete released build is done from the **Build Browser** and essentially pushes all the build artifacts one by one. From the **General Build Info** tab of the build you want to upload, click "Upload to Bintray"

The screenshot shows the Artifactory interface. On the left is a dark sidebar with icons for Home, Artifacts, Builds (selected), and Admin. The main area is titled 'Build Browser' and shows 'Build #19'. Below the title is a navigation bar with tabs: General Build Info (selected), Published Modules, Environment, Issues, and Licenses. Under the 'General Build Info' tab, there's a section titled 'General Info' containing build details like Name, Number, Agent, Start Date, Duration, and URL. To the right of this section is a green button labeled 'Upload to Bintray'.

Fill in the dialog details and click "Push":

Push To Bintray

Distribute this build's artifacts to users by uploading them to [Bintray](#)

Bintray is a public online service through which you can share your release binaries with the world.
Note that once artifacts are pushed, you need to publish them in Bintray in order to make them world-visible.

Bintray Repository*

jfrog/jfrog-jars

Bintray Package Name*

build-info-extractor-maven3

Bintray Package Version*

2.1.0

Use Bintray-specific artifact properties

Send Email Notification

Push

Background Push

- Use Bintray-specific artifact properties - Marking this option tells Artifactory to look for the properties attached to each build artifact and ignore the input from the UI (in case a property exists).
- Send Email Notification - Mark it to receive an email once the operation is finished (regardless of its status).
- Push - pushes all the build artifacts synchronously.
- Background Push - pushes all the build artifacts asynchronously (usually best when used with an email notification).

Disabling the 'Push To Bintray' option can be done by setting this property in the artifactory.system.properties (located under \$ARTIFACTORY_HOME/etc):

```
artifactory.bintray.ui.hideUploads=true
```

Using the REST API

In addition to pushing a build or an artifact through the UI, you can also use the Artifactory REST API to [push a build or an arbitrary set of files as a version](#). When pushing to Bintray using the REST API, properties annotating files are not used (although they are updated when the operation completes); instead, you need to provide a JSON descriptor, either as one of your build artifacts, or by specifying its path in the API call , and this is used to specify the various parameters Artifactory can automatically update for you. Here is an example:

Click to view the bintray-info.json descriptor

```
{  
  "repo": {  
    "name": "test",  
    "type": "generic",  
    "private": false,  
    "premium": false,  
    "desc": "My test repo",  
    "labels": ["label1", "label2"],  
    "url": "http://  
  }  
}  
http://  
  "label1": "label1",  
  "label2": "label2"  
}  
}
```

```
- "updateExisting": false
},
+ "package": {
+   "name": "auto-upload",
+   "repo": "test",
+   "subject": "myBintrayUser",
-   "desc": "I was pushed completely automatically",
-   "website_url": "www.jfrog.com",
-   "issue_tracker_url":
"https://github.com/bintray/bintray-client-java/issues",
+   "vcs_url":
"https://github.com/bintray/bintray-client-java.git",
+   "licenses": ["MIT"],
-   "labels": ["cool", "awesome", "gorilla"],
-   "public_download_numbers": false,
-   "public_stats": false,
-   "attributes": [{"name": "att1", "values" : ["val1"],
"type": "string",
          {"name": "att2", "values" : [1, 2.2, 4], "type": "number"},

          {"name": "att5", "values" :
["2014-12-28T19:43:37+0100"], "type": "date"}]
        },
+   "version": {
+     "name": "0.5",
-     "desc": "This is a version",
-     "released": "2015-01-04",
-     "vcs_tag": "0.5",
-     "attributes": [{"name": "VerAtt1", "values" :
["VerVal1"], "type": "string",
          {"name": "VerAtt2", "values" : [1, 3.3, 5], "type": "number"},

          {"name": "VerAtt3", "values" :
["2015-01-01T19:43:37+0100"], "type": "date"}],
-     "gpgSign": false
      },
      "applyToFiles": ["repo1/org/jfrog/*.*",
"repo2/org/jfrog/test/module*/*.jar",
"repo3/org/jfrog/test/**/*.*",
"repo2/org/jfrog/test/**/art.?ar"],
      "applyToRepoFiles": ["/org/jfrog/*.*", jfrog/test/**/*.*"],
      "applyToProps": [{"upload.prop1": ["val1", "val2"]}],
      {"upload.prop2": ["*"]}, {"*:
```

```
[ "valueRegardlessOfProperty"] } ] ,  
  "publish": true  
}
```

The file's name itself must contain the string `bintray-info` (anywhere in the name) and have a `.json` extension

Most of the fields are self-explanatory, however below are descriptions for those fields whose purpose may be less obvious:

Field	Purpose
<code>updateExisting</code>	Signifies Artifactory should update an existing repository with the same values as specified in the jsondescriptor with the values in the jsondescriptor (applies only to the ' <code>labels</code> ' and ' <code>cds</code>)
<code>subject</code>	Can either be your Bintray user name or the organization you are pushing to. The credentials that are used in the operation are those you defined in your user profile (or in the <code>default</code> section).

applyToFiles

If you are pushing a complete build, this field should remain empty. When pushing files, this field should contain a comma-separated list (in JSON format) that should be pushed. A file matching **any** of the file specifications will be pushed (i.e. an "OR" relationship).

You may use wildcards as follows:

- *: match any 0 or more characters
- **: recursively match any sub-folders (in the path section only)
- ?: match any 0 or 1 character

Here are some examples of valid search paths:

Path	Meaning
<i>repo1/org/jfrog/myTest.jar</i>	The file <i>myTest.jar</i> under <i>repo1/org/jfrog</i>
<i>repo1/org/jfrog/*.*</i>	All files under <i>repo1/org/jfrog</i>
<i>repo2/org/jfrog/test/module*/*.jar</i>	All <i>.jar</i> files under subfolder of <i>repo2/jfrog/test</i> whos starts with "module"
<i>repo2/org/jfrog/test/**/*.jar</i>	All <i>.jar</i> files under subfolder of <i>repo2/jfrog/test</i>
<i>repo2/org/jfrog/test/**/art.?ar</i>	All files named "art" a file extension that or 3 characters and with "ar" under any subfolder of <i>repo2/jfrog/test</i>

<code>applyToRepoFiles</code>	<p>If you are pushing a complete build, this field should remain empty.</p> <p>When pushing files, this field should contain a comma-separated list (in JSON format) that should be pushed. A file matching any of the file specifications will be pushed (i.e. an "OR" relationship).</p> <p>This field behaves similarly to <code>applyToFiles</code>, including wildcards as described above, only it refers to relative paths inside the repo that contains the descriptor file:</p> <ul style="list-style-type: none"> • If the path starts with a leading '/' then the parent for this path is the repository's root, so the path <code>/org/jfrog/*.*</code> will actually point to <code>containingRepo/org/jfrog/*.*</code> • If the path doesn't start with a '/' then the parent for this path is the one containing the descriptor. So if the descriptor resides in <code>/org/jfrog/bintray-info.json</code>, the path <code>/test/myPackage/*.*</code> will actually point to <code>containingRepo/org/jfrog/test/myPackage/*.*</code>
<code>applyToProps</code>	<p>0 or more key:value pairs with which to filter the selected files by property. The '*' and '?' wildcards are supported in this filter as well.</p> <p>A file matching all of the property specifications will be pushed (i.e. an "AND" relationship)</p>
<code>publish</code>	<p>If set to true, the version will be automatically published once the push operation is complete.</p>
<code>gpgSign</code>	<p>If set to true and no passphrase was passed as a parameter to the REST API call, Artifactory will attempt to sign the version without any passphrase.</p> <p>If you provide the <code>gpgPassphrase</code> parameter in the REST API call, this will cause the call to ignore this flag and the version will be signed with the passphrase that was passed.</p>

Creating Repositories, Packages and Versions

Packages and Versions

Artifactory uses `get / create` logic for packages and versions, meaning that these will be created for you, based on the information given in the JSON descriptor, if they do not exist already.

If they do exist the updatable fields will be **overwritten** with the new values in the descriptor.

Repositories

For repositories, Artifactory requires that you include the `repo` clause in the descriptor for it to be created according to the supplied information.

If you wish values to be updated (only applicable for the `labels` and `desc` fields) this must be explicitly specified in the `updateExisting` field.

Refer to the [Bintray REST API](#) for more information about the various fields for each item.

Repository name fields

As the `repo` clause is entirely optional, the `package` clause must also contain a `repoName` field, which can be omitted if the `repo` clause is present with its mandatory `name` field (and vice versa)

If there is a mismatch between these 2 fields the push operation will fail.

The 'repo' clause

The `'repo'` clause is supported in Artifactory versions 3.9.0 and above.

Pushing a Build

Using this mode requires that you leave the `applyToFiles` field of the JSON descriptor empty because the command pushes an entire set of build artifacts (as is shown in the **Builds** pane).

The build artifacts can be additionally filtered by properties that are defined in the `applyToProps` field.

As a convenience, in this mode you can actually omit the descriptor file in the build artifacts altogether and simply specify the 4 mandatory parameters in the REST query.

For more details, please refer to the REST API documentation for [Push Build to Bintray](#).

Pushing a Set of Files

In this mode you can specify any combination of the filters (file paths with `applyToFiles` and properties with `applyToProps`) or leave them empty. When the `applyToFiles` field is left empty, Artifactory will find all files in any subfolders of the folder containing the descriptor and push them to Bintray, you can optionally filter them by properties as well.

The descriptor file's upload location should depend on its required function: if the "`applyToFiles`" field is empty, it should be in the root of the set of files you want to upload. Otherwise it can be placed anywhere(as it has exact locations to pick up files from).

For more details, please refer to the REST API documentation for [Push a set of Artifacts to Bintray](#).

You can specify the maximum number of files that a user is permitted to push to Bintray in a single operation in the **Admin** panel under [Configuration | Bintray](#).

Deploying Snapshots to oss.jfrog.org

Overview

What is OJO

[oss.jfrog.org](#), or **OJO** for short, is an [Artifactory Cloud](#) instance for hosting your maven-compatible build snapshots, provided free of charge for selected opensource software projects.

All projects in OJO are public (i.e., all the artifacts and builds can be viewed by anyone).

Existing Bintray users are granted deploy permissions to relevant folders in Artifactory, according to the Maven Group ID of the packages they build.

Target Audience

This page is designed to help OSS contributors who want a free repository to host build snapshots, and eventually publish release versions to distribution via [Bintray](#).

At a Glance

The process of on-boarding to OJO, working with it to deploy continuous snapshots, and finally, promoting these snapshots from OJO to Bintray for distribution involves three simple steps:

1. Creating an account on OJO
2. Building and deploying to the OJO Artifactory
3. Promoting a snapshot build to Bintray

Page Contents

- Overview
 - What is OJO
 - Target Audience
 - At a Glance
- Getting Started with OJO
- Working with OJO
 - Resolving from and Publishing to OJO
 - Releasing to Bintray
 - Promoting a Release Build

Getting Started with OJO

To get account on OJO you must first have an account on Bintray.

1. Create a Maven repo on Bintray if it does not exist yet, and create your package inside this repo.
You can give your package any logical name, for example: `maven2gradle`
2. Ask for inclusion of the package in JCenter, by clicking the "**Add to JCenter**" button in the package main page.
In the request form, check "**Host my snapshot build artifacts on the OSS Artifactory at <https://oss.jfrog.org>**" and enter the desired Maven group ID for your package.
For example: `org.github.jbaruch.maven2gradle`

Request to include the package 'maven2gradle' in 'jcenter'

Host my snapshot build artifacts on the OSS Artifactory at
<https://oss.jfrog.org>

This allows you to have your project's builds snapshots deployed to
<https://oss.jfrog.org> and to release and publish them to Bintray in one click.

Enter a Maven group ID under which your artifacts can be uploaded. Your groupID is expected to be uniquely used by you. For example: '`org.acme.space-utils`'.

`org.github.jb.maven2gradle`

Comments

Send

 Cancel

After your request has been approved by the Bintray Team (usually within a few hours), you'll receive a confirmation email on the inclusion of your package in JCenter and the creation of your new OJO account.

OJO is Artifactory!

OJO is just a regular Artifactory Pro server, so [getting familiar with Artifactory](#) is recommended.

Bintray Organizations Support

When requesting an OJO account for a repository belonging to an Bintray organization, the permissions in OJO will be granted to all the organization members, not only the member who asked for the OJO account.

Working with OJO

Once your OJO account has been created, you (and all the team members in the case of an organization) should be able to login to OJO using

Feedback

your **Bintray username** and **API key** as the password.

You will see that a folder corresponding to the Maven Group ID has been created in OJO in the `oss-release-local` and the `oss-snapshot-local` repositories:

The screenshot shows the 'Artifact Repository Browser' interface. At the top, there are tabs for 'Tree' and 'Simple' with a dropdown arrow, and a checked checkbox for 'Compress Empty Folders'. The main area displays a tree view of repositories. Under 'oss-release-local', there is a 'org' folder containing a 'github/jb/maven2gradle' folder, which is highlighted with a red box. There is also an 'openjdk/tools' folder. Under 'oss-snapshot-local', there is another 'org' folder containing the same 'github/jb/maven2gradle' folder, also highlighted with a red box, and an 'openjdk/tools' folder.

You have deploy permissions to these folders:

User Permissions							
Filter by Permission Target							
Permission Target ▲	Applied To	Repositories	Manage	Delete/Overwrite	Deploy/Cache	Annotate	Read
Anything	jb	1 ANY					<input checked="" type="checkbox"/>
jb	jb	1 ANY		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Resolving from and Publishing to OJO

There is nothing unusual about working with repositories in OJO. You can configure your build tool to resolve release and snapshot dependencies from the `libs-release` and the `libs-snapshot` OJO virtual repositories, respectively; and to deploy build snapshots to the `os-snapshot-local` repository. As long as the `<groupId>` in your pom (for Maven) or the `project.group` (for Gradle) matches the group ID you requested during onboarding, the deployment should succeed.

Please consult the Artifactory documentation on how to set up [Maven](#) or [Gradle](#) for resolution and deployment.

Artifactory Build Info is a must!

In order to release and promote snapshots to Bintray you need to deploy a Build Info BOM to Artifactory. The easiest way to achieve this automatically it is to use the [Build Integration feature of Artifactory](#) or the [Gradle Artifactory Plugin](#); These and other options are described in the next section.

Releasing to Bintray

Currently, you have two ways to deploy artifacts to Bintray:

1. Promoting a Release Build

This will promote snapshot artifacts to release and then deploy them to Bintray:

- a. [Use promotion from the Jenkins Artifactory plugin](#) - This allows you to use the Jenkins UI to promote the snapshot artifacts from a selected job run.
- b. [Invoking promotion with REST](#) - This allows promotion of a build created with any build server/tool and full programmatic automation of the promotion process.

2. Uploading Release Artifacts

Directly upload deployed release artifact to Bintray. If you have a released version of an artifact or a build, you can deploy them to Bintray using the [regular Bintray integration](#).

Promoting a Release Build

Promoting a Build from Jenkins

Promotion from Jenkins is performed by invoking a custom "snapshotsToBintray" promotion plugin. Here's what you need to do:

1. Install the Jenkins Artifactory Plugin and configure Artifactory servers and repositories as described in the Jenkins documentation. You should configure the oss-release-local and oss-snapshot-local as release and snapshot targets, respectively.
2. In your build configuration, add the "Deploy artifacts to Artifactory" post-build action and check "Deploy Maven artifacts", "Capture and publish build info" and "Allow promotion of non-staged builds":

The screenshot shows the Jenkins 'Post-build Actions' configuration page for the 'Deploy artifacts to Artifactory' action. The configuration includes:

- Artifactory server:** https://oss.jfrog.info/artifactory
- Target releases repository:** oss-release-local
- Target snapshots repository:** oss-snapshot-local
- Custom staging configuration:** None
- Checkboxes (unchecked):** Override default deployer credentials, Deploy even if the build is unstable.
- Checkboxes (checked and highlighted with a red box):** Deploy maven artifacts, Capture and publish build info, Include environment variables.
- Text input fields:** Include Patterns (empty), Exclude Patterns (*password*, *secret*).
- Checkboxes (checked and highlighted with a red box):** Allow promotion of non-staged builds.

3. Run your build. Upon successful completion, the build result page will have a link to the "Artifactory Release Promotion" action:

Jenkins

Jenkins > [Project Name]

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[Modules](#)

[Artifactory Build Info](#)

[Artifactory Release Staging](#)

Build History (trend)	
#4	May 12, 2013 11:49:24 AM

Release promotion

4. In the promotion configuration screen, select "snapshotsToBintray" promotion plugin:

Artifactory Release Promotion

Artifactory Pro Release Promotion

Promotion plugin

releaseVersion

timestamped

There are two parameters to configure here:

- Override the release version. By default, the version is calculated by removing the -SNAPSHOT suffix from the snapshot version, e.g. 1.0-SNAPSHOT will be released to Bintray as version 1.0. Specifying a value in this field overrides the default version scheme.
- Append a timestamp to the version. This will add a timestamp string (in Maven's timestamp format: yyyyMMdd . HHmmss) to the release version. Values of true, y or 1 will cause the timestamp suffix to be appended.

5. Click the "Update" button. Your release artifacts are now uploaded to Bintray.

Promoting a Build Using REST API

If you don't use Jenkins or if you need fully automated promotion, you can issue an HTTP PUT request that will trigger promotion and release to Bintray. Promotion still operates on a Build Info BOM, previously saved in Artifactory. Here's what you need to do:

1. Deploy a build to Artifactory in one of the following ways:
 - a. Using a build server with an Artifactory plugin. Plugins currently exist for Jenkins, Hudson, Bamboo and TeamCity. Please see the [Artifactory Build Integration documentation](#) for further instructions on getting the build into BOM into Artifactory.
 - b. Using the [Gradle Artifactory Plugin](#).
 - c. Configure Maven to use Artifactory Listener as described [here](#).
2. Execute the [build promotion plugin call](#). The call accepts the same parameters as the invocation of Jenkins promotion plugin. Here's an example using CURL:

```
curl -X POST -u bintrayUser:apiKey
http://oss.jfrog.org/api/plugins/build/promote/snapshotsToBintray/buildName/3
```

Artifactory Pro

Overview

Artifactory Pro exposes an extensive set of features on top of the core repository management features that are available to you from Artifactory Open Source:

Package Management	
Docker Repositories	Host your own secure private Docker registries and proxy external Docker registries such as Docker Hub.
NuGet Repositories	Host and proxy NuGet packages in Artifactory, and pull libraries from Artifactory into your various Visual Studio .NET applications.
npm Repositories	Host your own node.js packages, and proxy remote npm repositories like npmjs.org through Artifactory.
Bower Repositories	Boost your front end development by hosting your own Bower components and proxying the Bower registry in Artifactory.
Git LFS Repositories	Optimize your workflow when working with large media files and other binary resources.
Vagrant Repositories	Securely host your Vagrant boxes in local repositories.

VCS Repositories	Consume source files packaged as binaries.
YUM Repositories	Distribute RPMs directly from your Artifactory server, acting as fully-featured YUM repository .
Debian Repositories	Host and provision Debian packages complete with GPG signatures.
RubyGems Repositories	Use Artifactory to host your own gems and proxy remote gem repositories like rubygems.org .
PyPI Repositories	Host and proxy PyPI distributions with full support for pip.
P2 Repositories	Proxy and host all your Eclipse plugins via an Artifactory P2 repository, allowing users to have a single-access-point for all Eclipse updates.
Features	
Repository Replication	Actively synchronize your repository content and metadata with remote Artifactory repositories using pull or push replication.
Artifactory Query Language	A simple way to formulate complex queries that can find artifacts based on any number of search criteria.
User Plugins	Extend Artifactory by plugging in your own custom Groovy scripts.
REST API	Automate your repository management and release life-cycle with a powerful REST API.
LDAP Groups	Synchronize your LDAP groups with Artifactory and leverage your existing organizational structure to manage group-based permissions.

Properties	Annotate your artifacts and folders with fully-searchable properties.
Smart Search	Assemble a set of artifacts from multiple searches, and then perform bulk operations on the whole result set.
Repository Layouts	Define the layout by which software modules are identified in your repository for automatic cleanup of old versions and cross-repository layout conversion.
License Control	Manage and control your organization's licensing policies for third-party dependencies used by your software.
Filtered Resources	Provision common settings and configuration to clients by turning any textual artifact into a dynamic template based on request parameters, current user identity and artifact properties.
WebStart and Jar Signing	Manage your JKS key-pairs from Artifactory and have requested artifacts and their dependencies automatically signed.
SSO	Integrate with SSO infrastructures such as Apache HTTPD , Atlassian Crowd and SAML .
Black Duck Code Center integration	Automate security and license governance of open source components and software.
Watches	Watch selected artifacts, folders, or repositories for any event , and receive email notifications on changes that are interesting to you.
Ecosystem	

Build Tool Integration	Resolve artifacts through Artifactory and deploy build artifacts to repositories in Artifactory transparently with all common build tools like Maven , Gradle , Ivy and SBT .
Continuous integration Systems	Deploy your build artifacts into Artifactory directly from industry standard CI servers such as Jenkins/Hudson , TeamCity , Bamboo and TFS/MSBuild .
Bintray Integration	Integrate with Bintray for a fully automated software delivery pipeline, end-to-end.

Page Contents

- [Overview](#)
- [Download](#)
- [Installation and Upgrade](#)
- [Activating Artifactory Pro](#)

Read More

- [Artifactory Comparison Matrix](#)
- [Build Integration](#)
- [Docker Repositories](#)
- [Repository Replication](#)
- [Artifactory Query Language](#)
- [S3 Object Storage](#)
- [User Plugins](#)
- [NuGet Repositories](#)
- [Npm Repositories](#)
- [Bower Repositories](#)
- [Git LFS Repositories](#)
- [Vagrant Repositories](#)
- [SBT Repositories](#)
- [LDAP Groups](#)
- [Atlassian Crowd Integration](#)
- [Single Sign-on](#)
- [SAML SSO Integration](#)
- [OAuth Integration](#)
- [VCS Repositories](#)
- [YUM Repositories](#)
- [Debian Repositories](#)
- [RubyGems Repositories](#)
- [PyPI Repositories](#)
- [Properties](#)
- [Smart Searches](#)
- [Repository Layouts](#)
- [License Control](#)
- [Black Duck Code Center Integration](#)
- [Filtered Resources](#)
- [P2 Repositories](#)
- [Watches](#)
- [WebStart and Jar Signing](#)

Comparing ArtifactoryPro , Artifactory OSS and Artifactory Online

To compare the features and services offered by each version of Artifactory please refer to the [Artifactory Version Comparison Matrix](#) to see which version of Artifactory best fits your needs.

For more information please contact support@jfrog.com.

Download

If you need a license, please visit the JFrog website and either [purchase a license or request an evaluation license](#).

Once you submit the corresponding form, a download link will be provided to you by email.

You may also access the latest version through the [Artifactory Pro Download Site](#).

Installation and Upgrade

Performing a clean installation of Artifactory Pro is identical to installing Artifactory OSS. Please refer to [Installing Artifactory](#).

To upgrade from a previous version of Artifactory Pro or Artifactory OSS, please refer to [Upgrading Artifactory](#).

Data is preserved when upgrading from Artifactory OSS to Artifactory Pro

To upgrade an instance of Artifactory OSS to Artifactory Pro **of the same version** you only need to replace the `artifactory.war` file and enter a valid license key. All data stored in Artifactory is preserved in the process.

Once you have entered a valid Artifactory Pro license key, all Artifactory Pro features will be available with the same settings and content you had on the Artifactory OSS version from which you upgraded.

Activating Artifactory Pro

Whether you have requested an evaluation of Artifactory Pro, or have purchased a license, your license key is provided in the same email that contains the download link sent to you.

Your Artifactory administrator should enter the license key into the corresponding field in the **Admin** module under [Configuration | Register License](#).

Administrator

You must be an Artifactory administrator in order to access the License Key field.

Artifactory License

License Details

Licensed to: JFrog
Valid Through: May 15, 2018
License Type: Commercial

License Key

Reset

Save

Using encrypted passwords

If you are using encrypted passwords with an IBM JDK/JRE, you may encounter encryption restrictions. For details please refer to [Using Your Secure Password](#).

Artifactory Comparison Matrix

Choose the Artifactory Edition that Fits You Best

	Artifactory OSS	Artifactory Pro	Artifactory Online	Artifactory Enterprise
Proxy and Cache Remote Repository Artifacts	✓	✓	✓	✓
Deploy Artifacts via the UI or via REST/HTTP	✓	✓	✓	✓
Bulk Artifact Deployment (from Archive)	✓	✓	✓	✓
Intuitive Slick Web UI	✓	✓	✓	✓
LDAP Authentication	✓	✓	✓	✓
Role Based Authorization	✓	✓	✓	✓
Include/exclude patterns for Stored Artifacts	✓	✓	✓	✓

Search by Name, Class, Module Info, XPath Values	✓	✓	✓	✓
UI-based Move/Copy/Delete	✓	✓	✓	✓
Smart "No-Duplicates" Storage	✓	✓	✓	✓
Share Remote Repository Configuration	✓	✓	✓	✓
Incremental and Historical Backup Services	✓	✓	✓	✓
Integration with All Leading CI-servers	✓	✓	✓	✓
Artifactory Query Language	✓	✓	✓	✓
Repository Replication	✗	✓	✓	✓
Multi-push Replication	✗	✗	✓ (Dedicated server)	✓
S3 Object Storage	✗	✗	✓ (Dedicated server)	✓
Managing Build Artifacts for Reproducible Builds	✗	✓	✓	✓
Promotion, Demotion and Cleanup of Build Artifacts	✗	✓	✓	✓
Automatic 3rd Party License Violation Detection per Build	✗	✓	✓	✓
3rd party license and security governance with Black Duck Code Center	✗	✓	✓	✓
Powerful RESTful API for Release Automation	✗	✓	✓	✓
Annotate Artifacts with Searchable Properties	✗	✓	✓	✓
Custom Repository Layout for Non-Maven Module Mgmt.	✗	✓	✓	✓

Docker Repositories	✗	✓	✓	✓
Bower Repositories	✗	✓	✓	✓
Git LFS Repositories	✗	✓	✓	✓
Vagrant Repositories	✗	✓	✓	✓
PyPI Repositories	✗	✓	✓	✓
Debian Repositories	✗	✓	✓	✓
NPM Repositories	✗	✓	✓	✓
NuGet Repositories	✗	✓	✓	✓
RubyGems Repositories	✗	✓	✓	✓
YUM Repositories	✗	✓	✓	✓
P2 Repositories	✗	✓	✓	✓
Aggregate and Run Bulk Operations on Search Results	✗	✓	✓	✓
Focused Email Notifications for Artifact Changes	✗	✓	✓	✓
On Demand Jar Signing and Web Start Application Hosting	✗	✓	✓	✓
Support for Multiple LDAP Servers	✗	✓	✓	✓
LDAP Groups Synchronization	✗	✓	✓	✓
Atlassian Crowd Authentication	✗	✓	✓	✓
Atlassian Crowd Groups Synchronization	✗	✓	✓	✓
Powerful SSO integration for NTLM, Kerberos, Etc.	✗	✓	✗	✓
Extend Artifactory with Groovy-based User Plugins	✗	✓	✗	✓
JFrog Email Support	✗	✓	✓	✓
Free Add-ons for the Time of your Contract	✗	✓	✓	✓

SaaS-based Maintenance-free Hosted Repository	✗	✗	✓	✗
Always up-to-date Artifactory Version	✗	✗	✓	✗
Setup Free Automated Backups	✗	✗	✓	✗
SLA-based Support	✗	✗	✓	✗
Five-nines Availability	✗	✗	✗	✓
Redundant Cluster of Servers	✗	✗	✗	✓
Unlimited Scalability	✗	✗	✗	✓
Near-zero Maintenance Downtime	✗	✗	✗	✓
JFrog Mission Control	✓ (view only)	✓ (view only)	✗	✓

Build Integration

Overview

Artifactory supports build integration whether you are running builds on one of the common CI servers in use today, on cloud-based CI servers or standalone without a CI server.

Integration of Artifactory into your build ecosystem provides important information that supports fully reproducible builds through visibility of artifacts deployed, dependencies and information on the build environment.

The Artifactory Build Integration Add-on provides a set of plugins you can use with industry standard CI systems and build tools that enable you to:

- See all the builds that are published and their build results in Artifactory.
- Explore the modules of each build, including published artifacts and corresponding dependencies.
- Obtain information about the build environment.
- Check if a specific artifact is required for or is a result of a build, and providing alerts if such an artifact should be targeted for removal.
- Treat all the artifacts and/or dependencies from a specific build as a single unit and perform bulk operations such as move, copy, export etc.
- Receive bidirectional links between build and artifact information inside the build server and Artifactory pages.

Running Builds on a CI Server

Artifactory can easily be added to a continuous integration build ecosystem by treating the CI server as a regular build client, so that it resolves dependencies from Artifactory, and deploys artifacts into a dedicated repository within Artifactory.

CI servers that are currently supported, each through a specific plugin are: [Jenkins/Hudson](#), [TeamCity](#), [Bamboo](#) and [TFS](#).

The build tools supported on all of these CI servers are: [Maven 3 and 2](#), [Gradle](#), [Ivy/Ant](#), [.Net](#), [MSBuild](#) as well as [Generic](#) build tools. For details please refer to the documentation for each CI server plugin.

Running Standalone Builds or on a Cloud-based CI Server

In the last few years, the popularity of cloud-based CI servers has grown. Some examples are, [Travis CI](#), [drone.io](#) and [Codeship](#). The problem is that none of these are "pluggable" in the traditional way. Therefore, to

support builds running on cloud-based build servers, as well as standalone builds, Artifactory provides plugins for industry standard build tools such as Maven, Gradle, Ivy/Ant and MSBuild. These plugins provide all the benefits of Artifactory that facilitate fully reproducible builds without the need for a CI server. For more details please refer to [Working with Maven](#), [Working with Gradle](#), [Working with Ivy](#) and [MSBuild Artifactory Plugin](#).

Build Integration for Artifactory open source version vs. Artifactory Pro

When using the OSS version of Artifactory, Build Integration includes the Generic BuildInfo View and the ability to traverse and view build information using Artifactory's REST APIs.

Artifactory Power Pack extends these capabilities and provides Module Artifacts and Dependencies View, Repository View of Builds and the ability to export and manipulate build items.

Page Contents

- Overview
 - Running Builds on a CI Server
 - Running Standalone Builds or on a Cloud-based CI Server
- Inspecting Builds
 - Builds and Build History
 - Build-level Information
 - General Build Information
 - Published Modules
 - Module Artifacts and Dependencies
 - Environment
 - Issues
 - Licenses
 - Governance
 - Build Diff
 - Release History
 - Build Info JSON
 - Generic BuildInfo View
- Exporting and Manipulating Build Items
- Repository View of Builds
- Behind the Scenes
- Release Management

Read More

- [Jenkins \(Hudson\) Artifactory Plug-in](#)
- [TeamCity Artifactory Plug-in](#)
- [Bamboo Artifactory Plug-in](#)
- [MSBuild Artifactory Plugin](#)

Inspecting Builds

Builds and Build History

All CI server projects that deploy their output to Artifactory can be viewed in the **Build Browser** which is accessed in the **Artifacts** module under **Builds**.

Build Browser

Filter by Project Name			
	Project Name	Last Build	Last Build Time
<input checked="" type="checkbox"/>	genericDeploy	30	30-06-15 15:16:59 UTC
<input checked="" type="checkbox"/>	mavenProject-NOTfreestyle	19	25-06-15 11:13:06 UTC
<input checked="" type="checkbox"/>	upload-nuget-packes	23	18-05-15 08:26:12 UTC
<input checked="" type="checkbox"/>	gradle-example	12	02-10-12 07:23:52 UTC
<input checked="" type="checkbox"/>	Demo :: Maven	3	01-10-12 18:46:02 UTC
<input checked="" type="checkbox"/>	maven-example	60	01-10-12 18:19:52 UTC

Selecting a project displays all runs of that build reflecting the build history in the CI server.

All Builds > maven-example

History for Build 'maven-example'					
Filter by Build Number					
	Build Number	CI Server	Number Of Modules/Artif...	Status	Last Build Time
<input checked="" type="checkbox"/>	60	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Staged	01-10-12 18:19:52 UTC
<input checked="" type="checkbox"/>	59	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...		01-10-12 16:17:51 UTC
<input checked="" type="checkbox"/>	58	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...		27-09-12 22:57:26 UTC
<input checked="" type="checkbox"/>	57	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Released	27-09-12 22:55:22 UTC
<input checked="" type="checkbox"/>	56	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Staged	27-09-12 22:52:49 UTC
<input checked="" type="checkbox"/>	55	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...		27-09-12 22:49:23 UTC
<input checked="" type="checkbox"/>	53	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Staged	27-09-12 22:35:56 UTC
<input checked="" type="checkbox"/>	52	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...		27-09-12 22:33:12 UTC
<input checked="" type="checkbox"/>	51	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...		27-09-12 22:26:57 UTC

Selecting a build item from the list displays complete **build-level information**. You can also view the build in the CI server by selecting the corresponding link under the **CI Server** column.

Permissions

To view build information you must have the 'deploy' permission on some repository path.

Build-level Information

You can select the **Build Number** to drill down into a specific build. This displays detailed information about the build, and enables you to compare it with another build as described in the following sections.

There are three categories of information:

1. General build information about the build and its environment.

2. Build modules along with their artifacts and dependencies.
3. Generic view of the build information in JSON format.

General Build Information

This tab displays general information about the build:

The screenshot shows the Jenkins Build Browser interface for Build #60. The top navigation bar includes 'All Builds' > 'maven-example' > '60'. The main title is 'Build #60'. Below it is a navigation menu with tabs: General Build Info (which is active), Published Modules, Environment, Issues, Licenses, Governance, Diff, and a double arrow icon. Under the 'General Build Info' tab, there is a section titled 'General Info' containing the following build details:

Name:	maven-example	Upload to Bintray
Number:	60	
Agent:	Jenkins/1.483	
Build Agent:	Maven/3.0.3	
Started:	2012-10-01T18:19:52.442+0000	
Duration:	13.3 seconds	
Principal:	eli	
Artifactory Principal:	jenkins	
URL:	http://repo-demo:8080/jenkins/job/maven-example/60/	

Name	The name assigned to the component being built
Number	The specific run of the build
Type	The build tool used
Agent	The CI server managing the build
Build Agent	The specific version of build tool used
Started	The time stamp when the build was started
Duration	The duration of the build
Principal	The factor that triggered this build. This may be a CI server user, or another build

Artifactory Principal	The Artifactory user that triggered this build
URL	Link to the build information directly on the build server

Published Modules

This tab displays the modules published into Artifactory as a result of the build, along with the number of artifacts and dependencies that they contain.

The screenshot shows the Artifactory Build Browser interface. At the top, there is a breadcrumb navigation: All Builds > maven-example > 60. Below this, the title "Build Browser" is displayed. Underneath the title, the specific build number "Build #60" is shown. A horizontal navigation bar includes links for General Build Info, Published Modules (which is underlined in green), Environment, Issues, Licenses, Governance, Diff, and a double arrow icon. A message "4 matches found" is displayed above a search/filter input field labeled "Filter by Module ID". To the right of the search field, there is a page navigation indicator showing "page 1 of 1". The main content area is a table with four rows, each representing a published module. The columns are "Module ID", "Number Of Artifacts", and "Number Of Dependencies". The data is as follows:

Module ID	Number Of Artifacts	Number Of Dependencies
org.jfrog.test:multi1:2.1.8	4	13
org.jfrog.test:multi2:2.1.8	2	1
org.jfrog.test:multi3:2.1.8	2	15
org.jfrog.test:multi:2.1.8	1	0

Module Artifacts and Dependencies

Selecting a published module that was built will display its artifacts and dependencies. You can group these by type or scope by clicking the corresponding column header.

You can click any item to download it directly, or click its **Repo Path** to view it in the **Tree Browser**.

All Builds > maven-example > 60

Build Browser

Build #60

General Build Info Published Modules Environment Issues Licenses Governance Diff Release History Build Info JSON

4 matches found Filter by Module ID < page 1 of 1 >

Module ID	Number Of Artifacts	Number Of Dependencies
org.jfrog.test:multi1:2.1.8	4	13
org.jfrog.test:multi2:2.1.8	2	1
org.jfrog.test:multi3:2.1.8	2	15
org.jfrog.test:multi:2.1.8	1	0

Module Details: org.jfrog.test:multi2:2.1.8

Compare with previous build

Artifacts

2 matches found Filter by Artifact Name < page 1 of 1 >

Artifact Name	Type	Repo Path
multi2-2.1.8.jar	jar	
multi2-2.1.8.pom	pom	

Dependencies

1 matches found Filter by Dependency ID < page 1 of 1 >

Dependency ID	Scope	Type	Repo Path
junit:junit:3.8.1	test	jar	gradle-libs-cache:junit:junit/3.8.1/junit-3.8.1.jar

Environment

The Environment tab displays an extensive list of properties and environment settings defined for the selected build. You can use these to reproduce the environment precisely if you need to rerun the build.

All Builds > maven-example > 60

Build Browser

Build #60

General Build Info Published Modules Environment Issues Licenses Governance Diff >>

Environment Variables

Key	Value
buildInfo.env._	/System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/bin/java
buildInfo.env._CF_USER_TEXT_ENCODING	0x1F5:0:0
buildInfo.env.Apple_PubSub_Socket_Render	/tmp/launch-c42X4o/Render
buildInfo.env.Apple_Ubiquity_Message	/tmp/launch-Wgt166/Apple_Ubiquity_Message
buildInfo.env.BUILD_CAUSE	USERIDCAUSE
buildInfo.env.BUILD_CAUSE_USERIDCAUSE	true
buildInfo.env.BUILD_ID	2012-10-01_20-18-46
buildInfo.env.BUILD_NUMBER	60

Issues

The **Issues** provides integration between Artifactory, Jenkins CI server and JIRA issue tracker. When using Jenkins CI, if you set the [Enable JIRA Integration](#) option in the Jenkins Artifactory Plugin, the **Issues** tab will display any JIRA issues that have been addressed by this build.

All Builds > maven-example > 60

Build Browser

Build #60

General Build Info Published Modules Environment **Issues** Licenses Governance Diff >

Key	Summary	Previous Build
RTFACT-4931	Saving large artifactory.config.xml from the UI throw internal e...	<input type="checkbox"/>
RTFACT-4932	Download latest version/integration for non-Maven artifacts	<input type="checkbox"/>

Licenses

The **Licenses** tab displays the results of a detailed license analysis of all artifacts and their dependencies.

All Builds > maven-example > 60

Build Browser

Build #60

General Build Info Published Modules Environment Issues **Licenses** Governance Diff >

⚠ Summary: Unapproved: 3 Not Found: 13 Unknown: 0 Neutral: 0 Approved: 2

Includes

- Include Published Artifacts
- Include dependencies of the following scopes:
 - compile
 - test
 - provided
 - runtime

[Export to CSV](#) [Auto-find Licenses](#)

Filter by Artifact ID ◀ page of 1 ▶

Artifact ID	Scopes	Repo Path	License
aopalliance:aopalliance:1.0	compile	gradle-libs-cache:aopalliance/aopalliance/1.0...	Public Domain
org.springframework:spring-beans:2.5.6	compile	Not in repository (externally resolved or dele...	Not Found
org.springframework:spring-aop:2.5.6	compile	Not in repository (externally resolved or dele...	Not Found
org.springframework:spring-core:2.5.6	compile	Not in repository (externally resolved or dele...	Not Found
org.testng:testng:5.9	test	gradle-libs-cache:org/testng/testng/5.9/testn...	Not Found
javax.servlet:servlet-api:2.5	provided	java.net.m1-cache:javax/servlet/servlet-api/2...	Not Found
javax.mail:mail:1.4	compile	gradle-libs-cache:javax/mail/mail/1.4/mail-1...	Not Found

The **Summary** line displays the number of artifacts found with the following statuses:

<i>Unapproved</i>	<p>The license found has not been approved for use</p> <div style="border: 1px solid #ccc; padding: 10px;"> <p>Approving licenses</p> <p>You can approve a license for use in the Admin tab under Configuration Licenses. For details please refer to License Control.</p> </div>
<i>Not Found</i>	No license requirements were found for the artifact.
<i>Unknown</i>	The artifact requires a license that is unknown to Artifactory
<i>Neutral</i>	A license requirement that is not approved has been found for the artifact, however there is another license that is approved.
<i>Approved</i>	All license requirements for the artifact are approved in Artifactory.

Governance

The **Governance** tab displays the results of running the selected build through Black Duck Code Center license analysis.

In addition to viewing the licensing status of components in the build, you can see a list of security vulnerabilities that were detected.

The report displays one of the following statuses for components:

<i>Approved</i>	The component has been approved by the Black Duck Code Center analysis
<i>Pending</i>	The component is pending examination by Black Duck
<i>Rejected</i>	The component was rejected by the Black Duck Code Center analysis

See the [Black Duck Code Center Integration](#) to learn more about this feature.

All Builds > maven-for-black-duck > 4

Build Browser

Build #4

General Build Info Published Modules Environment Issues Licenses **Governance** Diff Release History Build Info JSON

Includes

Include Published Artifacts
 Include dependencies of the following scopes:
 compile test runtime

Components

Summary: Total: 40 Pending: 18 NotInUse: 11 Stale: 11

Status	Artifact ID	License	Scopes	Repo Path
NotInUse				
Pending				
Stale	(Not in build)			

< page 1 of 1 >

Vulnerabilities

Summary: Total: 12 Medium: 9 High: 3

Severity	Artifact ID	Vulnerability	Description
High			
Medium			
Medium	94940	CVE-2013-6429	The SourceHttpMessageConverter in Spring MVC in Spring Framework 2.5.x before 2.5.6 SEC02, 2.5.7, SpringSource Spring Framework 2.5.x before 2.5.6 SEC02, 2.5.7.
Medium	94940	CVE-2010-1622	Algorithmic complexity vulnerability in the java.util.regex.Pattern class.
Medium	94940	CVE-2009-1190	The SourceHttpMessageConverter in Spring MVC in Spring Framework 2.5.x before 2.5.6 SEC02, 2.5.7.
Medium	95234	CVE-2013-6429	SpringSource Spring Framework 2.5.x before 2.5.6 SEC02, 2.5.7.
Medium	95234	CVE-2010-1622	Algorithmic complexity vulnerability in the java.util.regex.Pattern class.

< page 1 of 1 >

Build Diff

The **Diff** tab allows you to compare the selected build with any other build. Once you select a build number in the **Select A Build To Compare Against** field, Artifactory displays all the differences between the builds that were detected including new artifacts added, dependencies deleted, properties changed and more.

All Builds > maven-example > 60

Build Browser

Build #60

General Build Info Published Modules Environment Issues Licenses Governance **Diff** >>

Select A Build To Compare Against:

59

Exclude Internal Dependencies

Artifacts (9 Results)

Name (Current Build)	Name (Build #59)	Status	Module	Repo Path
multi2-2.1.8.jar	multi2-2.1.8.jar	Updated	org.jfrog.test:multi2:2.1.8	
multi3-2.1.8.war	multi3-2.1.8.war	Updated	org.jfrog.test:multi3:2.1.8	
multi-2.1.8.pom	multi-2.1.8.pom	Updated	org.jfrog.test:multi:2.1.8	http://10.100.1.110:8081/artifactory...
multi2-2.1.8.pom	multi2-2.1.8.pom	Updated	org.jfrog.test:multi2:2.1.8	http://10.100.1.110:8081/artifactory...
multi1-2.1.8-tests.jar	multi1-2.1.8-tests.jar	Updated	org.jfrog.test:multi1:2.1.8	
multi1-2.1.8-sources.jar	multi1-2.1.8-sources.jar	Updated	org.jfrog.test:multi1:2.1.8	
multi1-2.1.8.jar	multi1-2.1.8.jar	Updated	org.jfrog.test:multi1:2.1.8	
multi1-2.1.8.pom	multi1-2.1.8.pom	Updated	org.jfrog.test:multi1:2.1.8	http://10.100.1.110:8081/artifactory...
multi3-2.1.8.pom	multi3-2.1.8.pom	Updated	org.jfrog.test:multi3:2.1.8	http://10.100.1.110:8081/artifactory...

< page 1 of 1 >

Release History

The Release History tab displays a list of the selected build's release landmarks.

This screenshot shows the Artifactory Build Browser interface. At the top, there is a breadcrumb navigation: All Builds > maven-example > 60. Below the breadcrumb, the title "Build Browser" is displayed. A sub-header "Build #60" is shown above a horizontal navigation bar with tabs: General Build Info, Published Modules, Environment, Issues, Licenses, Governance, and Release History. The "Release History" tab is currently active, indicated by a green underline. The main content area contains a table with four rows of build metadata:

Repository:	ext-release-local
Comment:	Staging version 2.1.8
CI User:	eli
Artifactory User:	jenkins

Build Info JSON

Generic BuildInfo View

This tab displays the raw `BuildInfo` JSON representation of the build information in Artifactory. This data can be accessed via the REST API or used for debugging and is also available in the Artifactory OSS version.

This screenshot shows the Artifactory Build Browser interface with the "Build Info JSON" tab selected. The top navigation bar and breadcrumb are identical to the previous screenshot. The main content area displays the raw JSON representation of the build information. The JSON is a single object with many properties, some of which are redacted (indicated by ellipses). The visible properties include vendor information, Java runtime details, and various system paths.

```
1 {
2   "properties" : {
3     "java.vendor" : "Apple Inc.",
4     "buildInfo.env.USER" : "eli",
5     "sun.java.launcher" : "SUN_STANDARD",
6     "buildInfo.env.BUILD_TAG" : "jenkins-maven-example-60",
7     "sun.management.compiler" : "HotSpot 64-Bit Tiered Compilers",
8     "os.name" : "Mac OS X",
9     "buildInfo.env.PATH" :
10    "/usr/share/maven/bin:/System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home//bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/X11/bin",
11    "buildInfo.env.LOGNAME" : "eli",
12    "sun.boot.class.path" :
13      "/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Classes/jsfd.jar:/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Classes/classes.jar:/System/Library/Frameworks/JavaVM.framework/JavaVM.framework/JavaRuntimeSupport.framework/Resources/Java/JavaRuntimeSupport.jar:/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Classes/ui.jar:/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Classes/laf.jar:/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Classes/sunrsasign.jar:/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Classes/jsse.jar:/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Classes/charsets.jar",
14    "java.vm.specification.vendor" : "Sun Microsystems Inc.",
15    "java.runtime.version" : "1.6.0_33-b03-424-11M3720",
16    "user.name" : "eli",
17    "buildInfo.env.buildInfoConfig.propertiesFile" : "/usr/local/Cellar/tomcat/7.0.29/libexec/temp/buildInfo3361433366778964639.properties",
18    "guice.disable.misplaced.annotation.check" : "true",
19    "awt.nativeDoubleBuffering" : "true",
20    "m3plugin.lib" : "/Users/eli/.jenkins/plugins/artifactory-2.1.3-SNAPSHOT/WEB-INF/lib",
21    "buildInfo.env.BUILD_ID_URI" : "http://reno-demo:8080/jenkins/job/maven-example/60/".
```

Exporting and Manipulating Build Items

You can view a build in the repository browser and perform actions on it as a whole with all its artifacts and dependencies. For example, you could promote it to another repository, copy it, or export it to a disk.

The screenshot shows the Artifact Repository Browser interface. On the left, a tree view displays various local repositories like RubyGems-local, ext-release-local, and ext-snapshot-local. The 'ext-release-local' node is expanded, showing sub-folders such as org, repodata, and test. The 'test' folder contains sub-folders like multi, multi1, and multi2, with 'maven-metadata.xml' highlighted. The main right panel shows the details for 'maven-metadata.xml'. The 'General' tab is selected, displaying information such as Name: maven-metadata.xml, Repository Path: ext-release-local/org/jfrog/test/multi1/maven-metadata.xml, Created: 30-04-15 11:30:58 UTC, Deployed by: admin, Size: 383 bytes, Last Modified: 08-07-15 09:38:43 UTC, Module ID: N/A, Licenses: Not Found, Downloaded: 1, Last Downloaded: 22-06-15 11:39:54 UTC, and Last Downloaded By: admin. Below this is the 'Dependency Declaration' section, which shows the XML code for the dependency declaration:

```

1 <dependency>
2   <groupId>org.jfrog</groupId>
3   <artifactId>test</artifactId>
4   <version>multi1</version>
5   <type>ta.xml</type>
6 </dependency>

```

Repository View of Builds

When viewing an artifact within the **Tree Browser**, you can see all of the builds with which that artifact is associated, whether directly or as a dependency in the **Builds** tab

Moreover, if you try to remove the artifact you will receive a warning that the build will no longer be reproducible.

The screenshot shows the Artifact Repository Browser interface. The tree view on the left shows the 'libs-snapshot-local' repository expanded, with the 'org' and 'jfrog' sub-folders visible. The 'test' folder under 'jfrog' is expanded, showing sub-folders like 'multi', 'multi1', and '2.18-SNAPSHOT'. The '2.18-SNAPSHOT' folder contains 'maven-metadata.xml' and several JAR files. The main right panel shows the details for 'mult1-2.18-20150625.110536-1.pom'. The 'Builds' tab is selected, displaying a table of builds associated with this artifact. The table includes columns for Build Name, Build Number, Module ID, Started at, and CI Server. There are five entries in the table:

Build Name	Build Number	Module ID	Started at	CI Server
mavenProject-NOTfreest...	10	org.jfrog.test:mult1:2.18...	1435230341754	http://10.0.0.134:8080/jo...
mavenProject-NOTfreest...	14	org.jfrog.test:mult1:2.18...	1435230542843	http://10.0.0.134:8080/jo...
mavenProject-NOTfreest...	11	org.jfrog.test:mult1:2.18...	1435230392479	http://10.0.0.134:8080/jo...
mavenProject-NOTfreest...	13	org.jfrog.test:mult1:2.18...	1435230493924	http://10.0.0.134:8080/jo...
mavenProject-NOTfreest...	12	org.jfrog.test:mult1:2.18...	1435230443839	http://10.0.0.134:8080/jo...

The association of an artifact with a build is retained even if you move or copy it within Artifactory, because the association linked to the artifact's checksum which remains constant, regardless of its location.

Behind the Scenes

Behind the scenes, the Artifactory plug-in for your CI server performs two major tasks:

1. It resolves all dependencies from a resolution repository in Artifactory.
2. It deploys all the artifacts to Artifactory as an atomic operation at the end of the build, guaranteeing a more coherent deployment when building multi-module projects
(Maven and Ivy deploy each module at the end of its build cycle. If one of the modules fails, this can result in partial deployments).
3. It sends a `BuildInfo` data object to Artifactory via the REST API at the end of deployment. This is a structured JSON object containing all the data about the build environment, artifacts and dependencies, in a standard and open format.

You can find the latest `BuildInfo` Java-binding artifacts [here](#) and the source [here](#).

Release Management

Artifactory supports release management through its plugins for [Jenkins](#), [TeamCity](#) and [Bamboo](#).

When you run your builds using [Maven](#) or [Gradle](#) with jobs that use [Subversion](#), [Git](#) or [Perforce](#) as your version control system, you can manually stage a release build allowing you to:

- Change values for the release and next development version
- Choose a target staging repository to which to deploy the release
- Create a VCS tag for the release

Staged release builds can later be **promoted** or **rolled-back**, changing their release status in Artifactory, with the option to move the build artifacts to a different target repository.

Inside Artifactory, the history of all build status change activities (staged, promoted, rolled-back, etc.) is recorded and displayed for full traceability.

To learn more about release management specific to your CI server, please refer to:

 [Release Management in the Jenkins Documentation](#)

 [TeamCity Artifactory Plugin - Release Management](#)

 [Bamboo Artifactory Plug-in - Release Management](#)

Jenkins (Hudson) Artifactory Plug-in

Overview

Artifactory provides tight integration with both Jenkins and Hudson using a plugin which you need to install using the corresponding Plugin Manager. The information on this page refers to Jenkins however it applies equally to Hudson. For more information, please refer to the [Jenkins documentation](#) and [Hudson documentation](#) for the corresponding plugin.

The plug-in provides:

- Easy setup to resolve dependencies and deploy build artifacts through Artifactory.
- Capture exhaustive build information such as artifacts deployed, dependencies resolved, system and environment information and more to enable fully traceable builds.
- Enhanced deployment that transfers additional important build information to Artifactory.
- UI integration providing links from a Jenkins build directly to Artifactory.
- Release management with staging and promotion.

Before you begin

Please refer to the general information about [Artifactory's Build Integration](#) before using the Jenkins Artifactory Plugin.

Source Code Available!

The Jenkins Artifactory Plugin is an open source project on GitHub which you can freely browse and fork.

Supported Build Technologies

The Jenkins Artifactory Plugin currently supports [Maven 3](#), [Maven2](#), [Gradle](#), and [Ivy/Ant](#), as well as [generic \(free style\)](#) builds that use other build tools.

Page Contents

- [Overview](#)
 - [Supported Build Technologies](#)
- [Installing and Configuring the Plug-in](#)
- [Navigating Between Jenkins and Artifactory](#)
- [Watch the Screencast](#)

Installing and Configuring the Plug-in

Configuring the Jenkins Artifactory Plug-in involves two steps:

1. Setting up Jenkins to recognize your Artifactory server
2. Pointing the Jenkins build at a target local repository in Artifactory to which it should deploy the build artifacts. Jenkins queries Artifactory's available target repositories using Artifactory's REST API and allows you to select one.

We recommend that you also configure Jenkins to use Artifactory for dependency resolution.

For more information, please refer to the instructions at the [Jenkins Plug-in Center](#).

Navigating Between Jenkins and Artifactory

Each build viewed in Artifactory corresponds to its Jenkins job counterpart, and contains a build history of all runs corresponding to the build history in Jenkins.

You can [drill down](#) and view details about each item in the build history, or view the build in the CI server by selecting the corresponding link in the **CI Server** column.

You can also navigate back to Artifactory from Jenkins, as shown below:

Build Number	CI Server	Number Of Modules/Artif...	Status	Last Build Time
60	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Staged	01-10-12 18:15:52 UTC
59	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Staged	01-10-12 16:17:51 UTC
58	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Released	27-09-12 22:57:26 UTC
57	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Released	27-09-12 22:55:22 UTC
56	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Staged	27-09-12 22:52:49 UTC
55	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Staged	27-09-12 22:49:23 UTC
53	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Staged	27-09-12 22:35:58 UTC
52	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Staged	27-09-12 22:33:12 UTC
51	http://repo-demo:8080/jenkin...	Modules-4, Artifacts-9, Depen...	Staged	27-09-12 22:26:57 UTC

Jenkins

Jenkins > maven-example > #146 [ENABLE AUTO REFRESH](#)

[Back to Project](#) [Status](#) [Changes](#) [Console Output](#) [Edit Build Information](#) [Delete Build](#) [Artifactory Build Info](#) [Environment Variables](#) [Git Build Data](#) [No Tags](#) [Test Result](#) [See Fingerprints](#) [Rebuild](#) [Previous Build](#)

Build #146 (Aug 12, 2014 2:49:29 PM)

Started 20 days ago
Took [14 sec](#) on master

[add description](#)

Changes

1. Update pom.xml ([detail](#))

Artifactory Build Info

Started by anonymous user

Revision: 12cd5cbaa38b0bb87ca3e17b74d3dd7f061eaa41

git

• origin/master

[Test Result](#) (no failures)

Module Builds

Simple Multi Modules Build	1.7 sec
Multi 1	3.3 sec
Multi 2	0.82 sec
Multi 3	1.2 sec

[Help us localize this page](#) Page generated: Sep 1, 2014 4:36:05 PM [REST API](#) [Jenkins ver. 1.571](#)

Watch the Screencast

TeamCity Artifactory Plug-in

Overview

Artifactory provides tight integration with TeamCity CI Server through the TeamCity Artifactory Plug-in. Beyond managing efficient deployment of your artifacts to Artifactory, the plug-in lets you capture information about artifacts deployed, dependencies resolved, environment data associated with the TeamCity build runs and more, that effectively provides full traceability for your builds.

From version 2.1.0 the TeamCity Artifactory Plug-in provides powerful features for release management and promotion. For details please refer to [TeamCity Artifactory Plugin - Release Management](#).

Before you begin

Please refer to the general information about [Artifactory's Build Integration](#) before using the TeamCity Artifactory Plugin.

Source Code Available!

The TeamCity Artifactory Plugin is an open source project on [GitHub](#) which you can freely browse and fork.

Build Runner Support

The TeamCity Artifactory plugin supports most build runner types, including: **Maven2**, **Maven 3**, **Ivy/Ant** (with Ivy modules support), **Gradle**, **NAnt**, **MSBuild**, **FxCop** and **Ipr**.

Page Contents

- [Overview](#)
 - [Build Runner Support](#)
- [Installing the Plugin](#)
- [Configuration](#)
 - [Configuring System-wide Artifactory Servers](#)
 - [Configuring Project-specific Runners](#)
 - [Editing Project-specific Configuration](#)
 - [Running License Checks](#)
 - [Black Duck Code Center Integration](#)
 - [Advanced Options](#)
 - [Attaching Searchable Parameters to Build-Info and to Published Artifacts](#)
 - [Viewing Project-specific Configuration](#)
- [Running a Build with the Artifactory Plugin](#)
- [Triggering Builds in Reaction to Changes in Artifactory](#)
- [Proxy Configuration](#)
- [Licence](#)
- [Watch the Screencast](#)
- [Change Log](#)

Installing the Plugin

Some features of the plugin are not supported on older versions of Artifactory and TeamCity.

To make full use of the TeamCity Artifactory plugin, the recommended requirements are:

- Artifactory 2.2.5 or later.
- TeamCity 5.1.3 or later.

The compatibility matrix below specifies which version of the TeamCity Artifactory Plug-in you should install based on your versions of TeamCity and Artifactory

Remove older versions

If you have an older version of the plug-in, be sure to remove it before upgrading to a newer one

Artifactory plugin version	TeamCity version	Artifactory version	
2.1.5	7.1+	2.6.6+	Download
2.1.6	7.1+	2.6.6+	
2.1.7	7.1+	2.6.6+	Download
2.1.8	7.1+	2.6.6+	Download
2.1.9	7.1+	2.6.6+	Download

2.1.10	7.1+	2.6.6+	Download
2.1.11	7.1+	2.6.6+	Download
2.1.12	7.1+	2.6.6+	Download
2.1.13	7.1+	2.6.6+	Download

Plugins are deployed to TeamCity by placing the packaged plugin into the `$TEAMCITY_USER_HOME/plugins` folder and restarting TeamCity. You can also accomplish this via the TeamCity UI via **Administration | Plugins List | Upload Plugin Zip** and choosing the zip-file from your file-system. You will need to restart TeamCity (tomcat) for the plugin to take effect.

Configuration

To use the TeamCity Artifactory plugin you first need to configure your Artifactory servers in TeamCity's server configuration. You can then set up a project build runner to deploy artifacts and Build Info to a repository on one of the Artifactory servers configured.

Configuring System-wide Artifactory Servers

To make Artifactory servers globally available to project runner configurations, they must be defined in **Administration | Integrations | Artifactory**.

Select **Create new Artifactory server configuration** and fill in the URL of the Artifactory server.

Deployer credentials can be set at the global level for all builds, but they can also be overridden and set at a project build level.

Specifying a username and password for the resolver repository is optional. It is only used when querying Artifactory's REST API for a list of configured repositories and then only if the target instance does not allow anonymous access.

The screenshot shows the TeamCity Administration interface with the 'Artifactory' tab selected. A modal dialog titled 'Edit Artifactory Server Configuration' is open, prompting for 'Artifactory Server URL' (set to http://localhost:8081/artifactory), 'Default Deployer Username' (set to admin), and 'Default Deployer Password' (set to a masked value). There is also a checked checkbox for 'Use Different Resolver Credentials'. At the bottom of the dialog are 'Cancel', 'Save', and 'Test connection' buttons. A green banner at the top of the page indicates that 'Artifactory server configuration was updated.'

Configuring Project-specific Runners

Editing Project-specific Configuration

To set up a project runner to deploy build info and artifacts to Artifactory go to **Administration | Projects** and select the project you want to configure.

Then, under the **Build Configurations** section, click the **Edit** link for the build you want to configure.

Under **Build Configuration Settings**, select the relevant **Build Step** and click the **Edit** link for the build step you want to configure.

When you select a value in the **Artifactory server URL** field, the selected server is queried for a list of configured repositories (using the credentials configured in the corresponding **Artifactory Server Configuration**). This populates the **Target Repository** field with a list of repositories to which you can select to deploy.

Clicking on the **Free-text mode** checkbox enables you to type in repository name as free text. You may also include variables as part of the text. For example: libs-%variableName%

Configuration errors

If the **Target Repository** list remains empty, check that the specified Artifactory server URL, credentials and proxy information (if provided) are valid.

Any information about communication errors that might occur can be found in the TeamCity server logs.

Deploy Artifacts To Artifactory

Artifactory server URL:	<input type="text" value="http://localhost:8081/artifactory"/> <small>Select an Artifactory server.</small>
Target repository:	<input type="text" value="libs-release-local"/> <small>Free-text mode Specify a target deployment repository.</small>
Target snapshot repository:	<input type="text" value="libs-snapshot-local"/> <small>Free-text mode Specify a target deployment.</small>
Override default deployer credentials: <input checked="" type="checkbox"/> <small>Use different deployer user name and password than the default ones defined in the Artifactory settings under the administrative system configuration page. If no global defaults are defined and no user name and password are given here anonymous deployment will be attempted.</small>	
Deployer username:	<input type="text" value="admin"/> <small>Name of a user with deployment permissions on the target repository.</small>
Deployer password:	<input type="password" value="*****"/> <small>The password of the user entered above.</small>
Deploy Maven artifacts:	<input checked="" type="checkbox"/> <small>Uncheck if you do not wish to deploy Maven artifacts from the plugin (a more efficient alternative to Mavens own deploy goal).</small>
Deployment include patterns:	<input type="text"/> <small>Comma or space-separated list of Ant-style patterns of files that will be included in publishing. Include patterns are applied on the published file path before any exclude patterns.</small>
Deployment exclude patterns:	<input type="text"/> <small>Comma or space-separated list of Ant-style patterns of files that will be excluded from publishing. Exclude patterns are applied on the published file path after any include patterns.</small>
Publish build info:	<input checked="" type="checkbox"/> <small>Uncheck if you do not wish to deploy build information from the plugin.</small>
Include Environment Variables:	<input type="checkbox"/> <small>Check if you wish to include all environment variables accessible by the builds process.</small>
Run license checks:	<input type="checkbox"/> <small>Check if you wish automatic license scanning to run after the build is complete. (Requires Artifactory Pro).</small>

Running License Checks

If you are using Artifactory Pro, you can benefit from the **License Control** feature to discover and handle third party dependency licensing issues as part of the build.

If you check the **Run License Checks** checkbox, Artifactory will scan and check the licenses of all dependencies used by this build. You can also specify a list of recipients who should receive any license violation notifications by email.

Black Duck Code Center Integration

If you are using Artifactory Pro and have an account with Black Duck Code Center, you can run the build through an automated, non-invasive, open source component approval process, and monitor for security vulnerabilities.

Run Black Duck Code Center compliance checks: Check if you wish that automatic Black Duck Code Center compliance checks will occur after the build is completed. (Requires Artifactory Pro).

Code Center application name: The existing Black Duck Code Center application name.

Code Center application version: The existing Black Duck Code Center application version.

Send compliance report email to: Input recipients that will receive a report email after the automatic Black Duck Code Center compliance checks finished.

Limit checks to the following scopes: A list of dependency scopes/configurations to run Black Duck Code Center compliance checks on. If left empty all dependencies from all scopes will be checked.

Include published artifacts: Include the build's published module artifacts in the Black Duck Code Center compliance checks if they are also used as dependencies for other modules in this build.

Auto-create missing component requests: Auto create missing components in Black Duck Code Center application after the build is completed and deployed in Artifactory.

Auto-discard stale component requests: Auto discard stale components in Black Duck Code Center application after the build is completed and deployed in Artifactory.

Advanced Options

The project-specific runner configuration offers some advanced options for the following runner types:

- Command Line
- FxCop
- MSBuild
- Rake
- Powershell
- XCode Project
- NuGet Publish
- NAnt
- Visual Studio (sln)
- Visual Studio 2003
- SBT, Scala build tool

Custom published artifacts	Allows you to specify which artifact files produced by the build should be published to Artifactory. At the end of the build the plugin locates artifacts in the build's checkout directory according to the specified artifact patterns, and publishes them to Artifactory to one or more locations, optionally applying a mapping for the target path of each deployed artifact. The pattern and mapping syntax for Published Artifacts is similar to the one used by TeamCity for Build Artifacts .
Custom build dependencies	Allows you specify dependency patterns for published artifacts that should be downloaded from Artifactory before the build is run. You can have detailed control over which artifacts are resolved and downloaded by using query-based resolution, adding to your artifact paths a query with the properties that the artifact should have before it can be downloaded. For further information read here about Resolution by Properties .

Custom published artifacts:

```
Edit published artifacts:  
D:\Work\AntExample2\**\*.class=>class2.zip  
D:\Work\AntExample2\**\*.class=>class2
```

New line or comma separated paths to build artifacts that will be published to Artifactory. Supports ant-style wildcards like dir/**/* .zip and target directories like *.zip=>winFiles, unix/distro.tgz=>linuxFiles, where winFiles and linuxFiles are target directories. Artifacts archiving is also supported, for example: test-results=>test-results.zip, where test-results is a source directory.

Custom build dependencies:

```
Edit build dependencies:  
*:*/*.jar@multi :: multi#*> one-jar-build  
*:*/*.class@GENERIC :: ant#22=>generic-class  
*:*/*.jar@gradle-perforce :: GRADLE#LATEST
```

New line or comma separated references to other build artifacts that this build should use as dependencies. Each reference is specified in the format of: repo_key:path_pattern[:prop=val1,val2[:prop2+=val3]]#build_name#build_number[=>target_dir], where:
repo_key - A key of the Artifactory repository that contains the dependencies (may contain the * and the ? wildcards).
path_pattern - An Ant-like pattern of the dependencies path within the Artifactory (may contain the * and the ? wildcards, including **).
For example: repo-key:dir/*/*/*.zip (** wildcards are not supported)
Artifacts can be downloaded conditionally based on their property values in Artifactory. For example, to download all zip files marked as production ready:
repo-key:dir/*/*/*.zip;status=prod. For more details see the plug-in's documentation.
build_name - The name of the build that was published to Artifactory, of which dependencies should be resolved.
build_number - A specific build run number. Can be LATEST to depend on the latest build run, or LAST_RELEASE to depend on the latest build with a "release" status. For example: repo-key:dir/*/*/*.zip@myBuild#LATEST
target_dir - An optional target directory to where resolved dependencies will be downloaded. By default dependencies will be downloaded to a path under the build workspace. For example: repo-key:*.zip=>winFiles, repo-key:unix/distro.tgz=>linuxFiles, where winFiles and linuxFiles are target directories. Target directories can either be absolute or relative to the working directory.

As of version 2.1.4, the above configuration is not backward compatible and you may need to re-save the builds configuration for them to run properly.

If no matching artifacts are found, remember that these parameters may be case sensitive depending on the operating system, the agent and the server they are running on.

Attaching Searchable Parameters to Build-Info and to Published Artifacts

In the **Build Configuration Settings** you can select **Parameters** to define system properties or environment variables that should be attached to artifacts and their corresponding build info.

To define a parameter click on the **Add new parameter** button.

Add New Parameter

Name: *	<input type="text"/>
Kind:	<input type="text" value="Configuration parameter"/> ▼
Value:	<input type="text"/> <small>Type '%' for reference completion</small>
Spec:	<input type="button" value="Edit..."/> Show raw value <small>Defines parameter control presentation and validation.</small>
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Fill in the corresponding fields.

Parameters relevant for builds run through Artifactory are:

- `buildInfo.property.*` - All properties starting with this prefix are added to the root properties of the build-info
- `artifactory.deploy.*` - All properties starting with this prefix are attached to any deployed produced artifacts

You can specify all the properties in a single file, and then define another property pointing to it.

To point the plugin to a properties file, define a property called `buildInfoConfig.propertiesFile` and set its value to the absolute path of the properties file.

It is also possible to point the plugin to a properties file containing the aforementioned properties.

The properties file should be located on the machine running the build agent, **not on the server!**

[+ Add new parameter](#)

Configuration Parameters

Configuration parameters are not passed into build, can be used in references only. [?](#)

None defined

System Properties (system.)

System properties will be passed into the build (without system. prefix), they are only supported by the build runners that understand the property notion. [?](#)

Name	Value	Edit	Delete
system.buildinfo.property.bob.number	%system.build.vcs.number%	Edit	Delete
system.buildInfoConfig.propertiesFile	this/is/a/path.properties	Edit	Delete

Environment Variables (env.)

Environment variables will be added to the environment of the processes launched by the build runner (without env. prefix). [?](#)

Name	Value	Edit	Delete
env.buildInfoConfig.deploy.foo	%maven.project.name%	Edit	Delete

Viewing Project-specific Configuration

Existing project configuration can be viewed in **Settings** under **Projects | \$PROJECT_NAME | \$BUILD_NAME**:

Build Step: Maven [edit »](#)

Step 1:

Runner type: Maven (Runs Maven builds)
Execute: [②](#) If all previous steps finished successfully (zero exit code)
POM file path: maven-example/pom.xml
Goals: clean install
Maven used: default
Additional Maven command line parameters: none specified
User settings provided by default
Maven metadata disabled: false
Use own local artifact repository: false
Build only modules affected by changes (incremental building): false
JDK home path: not specified
Build working directory: not specified
JVM command line parameters: not specified

Deploy artifacts to Artifactory: enabled
Artifactory server: <http://10.0.0.235:8080/artifactory>
Target repository: libs-release-local
Target snapshot repository: libs-snapshot-local
Deployer username: admin
Deploy artifacts: true
Deployment include patterns: not specified
Deployment exclude patterns: not specified
Publish Build Info: true
Include Environment Variables: true
Environment Variables Include Patterns: not specified
Environment Variables Exclude Patterns: *password*, *secret*
Run license checks (Requires pro): true
License violation notifications recipients: none specified
Limit checks to the following scopes: none specified
Include published artifacts: false
Disable automatic license discovery: false
Enable Artifactory release management: true
VCS tags base URL/name: none
Git release branch name prefix: REL-BRANCH-
Alternative Maven goals: none
Alternative Maven command line parameters: none
Default module version configuration: Global

Java code coverage: disabled

Running a Build with the Artifactory Plugin

Once you have completed setting up a project runner you can run a project build. The Artifactory plugin takes effect at the end of the build and does the following:

1. For all build runner types - Publishes the specified Published Artifacts to the selected target repository and applies corresponding path mappings.
2. For Maven or Ivy/Ant build runner - Deploys all artifacts to the selected target repository together at the end of the build (as opposed to deploying separately at the end of each module build as done by Maven and Ivy).
3. Deploys the Artifactory BuildInfo to Artifactory, providing [full traceability of the build in Artifactory](#), with links back to the build in TeamCity.

```

[15:38:09]: [INFO] [source:jar {execution: attach-sources}]
[15:38:09]: [INFO] Building jar: /home/noam/Work/JetBrains/tc-5.1/buildAgent/work/db3222d3a8bdbdddf/build-info-extractor-gradle/target/build-
[15:38:09]: [INFO] [install:install {execution: default-install}]
[15:38:09]: [INFO] Installing /home/noam/Work/JetBrains/tc-5.1/buildAgent/work/db3222d3a8bdbdddf/build-info-extractor-gradle/target/build-inf
[15:38:10]: [INFO] Installing /home/noam/Work/JetBrains/tc-5.1/buildAgent/work/db3222d3a8bdbdddf/build-info-extractor-gradle/target/build-inf
[15:38:10]: [INFO]
[15:38:10]: [INFO] -----
[15:38:10]: [INFO] Reactor Summary:
[15:38:10]: [INFO] -----
[15:38:10]: [INFO] JFrog Build-Info ..... SUCCESS [3.653s]
[15:38:10]: [INFO] JFrog Build-Info API ..... SUCCESS [6.799s]
[15:38:10]: [INFO] JFrog Build-Info Client ..... SUCCESS [3.941s]
[15:38:10]: [INFO] JFrog Build-Info Extractor ..... SUCCESS [2.566s]
[15:38:10]: [INFO] JFrog Build-Info Maven 3 Extractor ..... SUCCESS [3.415s]
[15:38:10]: [INFO] Build Info Gradle Extractor ..... SUCCESS [2.006s]
[15:38:10]: [INFO] -----
[15:38:10]: [INFO] BUILD SUCCESSFUL
[15:38:10]: [INFO] -----
[15:38:10]: [INFO] Total time: 22 seconds
[15:38:10]: [INFO] Finished at: Sun Apr 25 15:38:10 IDT 2010
[15:38:10]: [INFO] Final Memory: 78M/479M
[15:38:10]: [INFO] -----
[15:38:10]: [Maven Watcher] starting handling projects
[15:38:10]: [Maven Watcher] building report document...
[15:38:10]: [Maven Watcher] building report document done
[15:38:10]: [Maven Watcher] writing report to /home/noam/Work/JetBrains/tc-5.1/buildAgent/temp/buildTmp/maven-build-info.xml
[15:38:10]: [Maven Watcher] done writing report
[15:38:10]: Number of processed tests: 35
[15:38:11]: Deploying artifacts to http://amphibian.jfrog.org:8081/artifactory
[15:38:11]: Deploying artifact: org/jfrog/build-info-parent/1.2.x-SNAPSHOT/build-info-parent-1.2.x-SNAPSHOT.pom
[15:38:13]: Deploying artifact: org/jfrog/build-info-api/1.2.x-SNAPSHOT/build-info-api-1.2.x-SNAPSHOT.jar
[15:38:13]: Deploying artifact: org/jfrog/build-info-api/1.2.x-SNAPSHOT/build-info-api-1.2.x-SNAPSHOT-sources.jar
[15:38:13]: Deploying artifact: org/jfrog/build-info-client/1.2.x-SNAPSHOT/build-info-client-1.2.x-SNAPSHOT.jar
[15:38:13]: Deploying artifact: org/jfrog/build-info-client/1.2.x-SNAPSHOT/build-info-client-1.2.x-SNAPSHOT-sources.jar
[15:38:13]: Deploying artifact: org/jfrog/build-info-extractor/1.2.x-SNAPSHOT/build-info-extractor-1.2.x-SNAPSHOT.jar
[15:38:13]: Deploying artifact: org/jfrog/build-info-extractor/1.2.x-SNAPSHOT/build-info-extractor-1.2.x-SNAPSHOT-sources.jar
[15:38:13]: Deploying artifact: org/jfrog/build-info-extractor-maven3/1.2.x-SNAPSHOT/build-info-extractor-maven3-1.2.x-SNAPSHOT.jar
[15:38:13]: Deploying artifact: org/jfrog/build-info-extractor-maven3/1.2.x-SNAPSHOT/build-info-extractor-maven3-1.2.x-SNAPSHOT-sources.jar
[15:38:13]: Deploying artifact: org/jfrog/build-info-extractor-gradle/1.0-SNAPSHOT/build-info-extractor-gradle-1.0-SNAPSHOT.jar
[15:38:13]: Deploying artifact: org/jfrog/build-info-extractor-gradle/1.0-SNAPSHOT/build-info-extractor-gradle-1.0-SNAPSHOT-sources.jar
[15:38:13]: Deploying build info ...
[15:38:14]: Build finished

```

[Help](#) [Feedback](#)

TeamCity Enterprise 5.1 (build 13360)

You can also link directly to the build information in Artifactory from a build run view:

Test > Maven > #9 (16 Apr 14 12:56)

Run ... Actions Edit Configuration Settings

Overview	Changes	Tests	Build Log	Parameters	Artifacts	Maven Build Info	#8 All history Last recorded build
Result: ✓ Tests passed: 2	Agent: Default Agent						
Time: 16 Apr 14 12:56:47 - 12:56:56 (9s)	Triggered by: you on 16 Apr 14 12:56						
Branch: master							

O Artifactory Build Info

✓ 2 tests passed

Triggering Builds in Reaction to Changes in Artifactory

The plugin allows you to set a new type of trigger that periodically polls a path in Artifactory, a folder or an individual file. Whenever a change is detected in the polled element, the TeamCity build is triggered. For example, the build could be triggered when new artifacts have been deployed to the specified path in Artifactory.

Artifactory Pro required

Triggering builds is only available with Artifactory Pro

To configure a new build trigger, under **Administration**, select **\$PROJECT_NAME | \$BUILD_NAME**. Then, under **Build Configuration Settings** select **Triggers**.

Click the **Add new trigger** button to select an **Artifactory Build Trigger**

The screenshot shows the 'Build Configuration Settings' page for a Maven build step. The 'Triggers' section is selected. A modal dialog titled 'Add New Trigger' is open, displaying a dropdown menu with various trigger options. The 'Artifactory Build Trigger' option is highlighted.

- General Settings
- Version Control Settings (1)
- Build Step: Maven
- Triggers
- Failure Conditions
- Build Features
- Dependencies
- Parameters (3)
- Agent Requirements

Created 18 minutes ago by admin (view history)

Triggers

Triggers are used to add builds to the queue either when an event occurs (like a VCS check-in) or periodically with some configurable interval. [?](#)

+ Add new trigger

Add New Trigger

-- Please choose a trigger --

VCS Trigger

Schedule Trigger

Finish Build Trigger

Artifactory Build Trigger

Branch Remote Run Trigger

Maven Artifact Dependency Trigger

Maven Snapshot Dependencies Trigger

NuGet Dependency Trigger

Retry Build Trigger

Select the **Artifactory Server URL** and the **Target repository**.

Complete the username and a password fields of a valid deployer for the selected repository.

Deploy permission

The specified user must have deploy permissions on the repository

Then, in **Items to watch**, specify the paths in the selected repository in which a change should automatically trigger a build.

Add New Trigger

Artifactory Build Trigger

Artifactory Server Settings

Artifactory server URL: * <http://10.0.0.235:8080/artifactory>

Target repository: * [libs-release-local](#)

Username: [admin](#)

Password: [.....](#)

Items to watch: * [Edit items to watch:](#)

com/acme/releases/milestone5
com/acme/snapshots

New line or comma separated paths in Artifactory that will be polled for changes. Paths denote files or folders relative to the target repository. For example:
com/acme/releases/milestone-5.

Note: Artifactory folders are scanned for changes recursively. To avoid slow performance it is recommended to narrow down the scope of scanning.

Polling interval seconds: [0](#)

Save **Cancel**

Be as specific as possible in Items to watch

In order to establish if there has been a change, Artifactory must traverse all the folders and their sub-folders specified in **Items to watch**. If the specified folders have a lot of content and sub-folders, this is a resource intensive operation that can take a long time.

Therefore, we recommend being as specific as possible when specifying folders in **Items to watch**.

Proxy Configuration

If the Artifactory server is accessed via a proxy, you need to configure the proxy by setting the following properties in the \$TEAMCITY_USER_HOME/.BuildServer/config/internal.properties file. If the file does not exist, you'll need to create it.

```
org.jfrog.artifactory.proxy.host
org.jfrog.artifactory.proxy.port
org.jfrog.artifactory.proxy.username
org.jfrog.artifactory.proxy.password
```

Licence

The TeamCity Artifactory plugin is available under the Apache v2 License.

Watch the Screencast

To see the Teamcity plugin in action you can watch the short demo screencast below.

Change Log

▼ Click to see change log details...

2.1.13 (4 May 2015)

1. Support multi Artifactory Build Triggers ([TCAP-222](#))
2. Support SBT build tool ([TCAP-223](#))
3. Bug fix ([TCAP-214](#))

2.1.12 (12 Mar 2015)

1. Adding support for free text repository configuration ([TCAP-217](#))

2.1.11 (7 Dec 2014)

1. Compatibility with Gradle 2.x ([TCAP-211](#))
2. Bug Fixed ([TCAP-205](#))

2.1.10 (8 May 2014)

1. Bug Fixed ([TCAP-206](#), [TCAP-72](#))

2.1.9 (17 Apr 2014)

1. Adding Version Control Url property to the Artifactory Build Info JSON. ([TCAP-203](#))
2. Support for TeamCity 8.1 Release management feature issues
3. Support working with maven 3.1.1
4. Bug Fixed ([TCAP-197](#), [TCAP-161](#))

2.1.8 (15 Jan 2014)

1. Allow remote repository caches to be used for build triggering - [TCAP-196](#)
2. Bug Fixes

2.1.7 (18 Dec 2013)

1. Add support for blackduck integration - [TCAP-185](#)

2.1.6 (03 Sep 2013)

1. TeamCity 8.0.x full compatibility issue - [TCAP-172](#)
2. Global and build credentials issue - [TCAP-153](#)
3. Repositories refreshed by credential issue - [TCAP-166](#)
4. Generic deployment resolution on Xcode builds - [TCAP-180](#)
5. Working directory in Gradle build issue - [TCAP-125](#)

2.1.5 (07 Jul 2013)

1. Fix security issue - [TCAP-172](#)
2. Improve generic resolution - [BI-152](#)

2.1.4 (21 Aug 2012)

1. Compatible with TeamCity7.1.
2. Bug Fixes

2.1.3 (30 May 2012)

1. Compatible with TeamCity7.
2. Support 'Perforce' in release management.
3. Support multiple deploy targets for the same source pattern in generic deploy.
4. Support for custom build dependencies resolution per build.

2.1.2 (12 Dec 2011)

1. Compatible with Gradle 1.0-milestone-6.

2.1.1 (09 Aug 2011)

1. Support for Gradle milestone-4
2. Better support for releasing nested Maven projects
3. Fixed minor Maven deployments discrepancies

2.1.0 (14 Jul 2011)

1. Release management capabilities
2. Bug fixes

2.0.1 (9 Jan 2011)

1. Auto Snapshot/Release target repository selection
2. Add ivy/artifact deploy patterns
3. Improved Gradle support
4. Bug fixes

2.0.0 (5 Dec 2010)

1. Support for Gradle builds
2. Support for maven3 builds
3. Default deployer add resolver credentials
4. Support for muti steps builds

1.1.3 (21 Nov 2010)

1. Include/exclude pattern for artifacts deployment

1.1.2 (7 Nov 2010)

1. Control for including published artifacts when running license checks
2. Limiting license checks to scopes
3. Control for turning off license discovery when running license checks

TeamCity Artifactory Plugin - Release Management

Overview

The TeamCity Artifactory Plugin includes release management capabilities for Maven and Gradle runners that use Subversion, Git or Perforce for version control.

When you run your builds using [Maven](#) or [Gradle](#) with jobs that use [Subversion](#), [Git](#) or [Perforce](#) as your version control system, you can manually stage a release build allowing you to:

- Change values for the release and next development version
- Choose a target staging repository to which to deploy the release
- Create a VCS tag for the release

Staged release builds can later be promoted or rolled-back, changing their release status in Artifactory, and build artifacts may optionally be moved to a different target repository.

Inside Artifactory, the history of all build status changes (staged, promoted, rolled-back, etc.) is recorded and displayed for full traceability.

Page Contents

- [Overview](#)
- [Maven Release Management](#)
 - [Configuring Maven Runners](#)
 - [Staging a Maven Release Build](#)
 - [Promoting a Release Build](#)
- [Gradle Release Management](#)

- Configuring Gradle Runners
- Staging a Gradle Release Build
- Promoting a Release Build
- Working with Subversion
- Working with Git

Maven Release Management

The TeamCity Artifactory Plugin manages a release with Maven running the build only once using the following basic steps:

1. Change the POM versions to the release version (before the build starts).
2. Trigger the Maven build (with optionally different goals).
3. Commit/push changes to the tag (Subversion) or the release branch (Git).
4. Change the POM versions to the next development version.
5. Commit/push changes to the trunk.

If the build fails, the plugin attempts to rollback the changes (both local and committed).

For more information including configuration of Maven Runners, and Jobs and staging a release build, please refer to [TeamCity Artifactory Plugin](#)

Configuring Maven Runners

To enable release management in Maven runners, edit the runner's step configuration and check the **Enable Artifactory release management** checkbox.

Enable Artifactory release management:	<input checked="" type="checkbox"/>
VCS tags base URL/name:	<input type="text" value="http://scm.acme.com/killer_project/tags"/> <small>For subversion this is the URL of the tags location, for Git this is the name of the tag.</small>
Git release branch name prefix:	<input type="text" value="REL-BRANCH-"/> <small>The prefix of the release branch name (applicable only to Git).</small>
Alternative Maven goals:	<input type="text" value="clean install"/> <small>Alternative goals to execute for a Maven build running as part of the release. If left empty, the build will use original goals instead of replacing them.</small>
Alternative Maven command line parameters:	<input type="text" value="-Prelease"/> <small>Alternative options to execute for a Maven build running as part of the release. If left empty, the build will use original options instead of replacing them.</small>
Default module version configuration:	<input type="button" value="Per Module"/> <small>Default versioning strategy to use:</small> <ul style="list-style-type: none"> • Global - One version for all modules • Per Module - Allows different version per module • None - Don't change module version

Staging a Maven Release Build

Once release management is enabled, the Artifactory Release Management tab appears at the top of the build page.



Clicking on the tab reveals configuration options for the release build:

Artifactory Release Staging

Last build version: 2.14-SNAPSHOT

Module Version Configuration

One version for all modules

Use same version for all modules.

Version per module

Select version per module.

Use existing module versions

Don't change module versions.

VCS Configuration

Create VCS tag

Create tag in the version control system.

Tag URL/name:

`http://scm.acme.com/killer_project/tags/2.14`

The tag URL. Build will fail if the tag already exists.

Tag comment:

`[artifactory-release] Release version 2.14`

The comment to use when creating the tag.

Next development version comment:

`[artifactory-release] Next development version`

The comment to use when committing changes for the next development version.

Artifactory Configuration

Repository to stage the release to (Artifactory Pro):

`libs-release-local`

Target repository to stage the release published artifacts to.

Staging comment:

`w00t!`

Comment that will be added to the promotion action.

Build and Release to Artifactory

The release staging page displays the last version built (the version tag is that of the root POM, and is taken from the last build that is not a release). Most of the fields in the form are populated with default values.

Version configuration controls how the plugin changes the version in the POM files (global version for all modules, version per module or no version changes).

If the **Create VCS tag** checkbox is checked (default), the plugin commits/pushes the POMs with the release version to the version control system with the commit comment. When using Git, there's also an option to create a release branch.

Click on the **Build and Release to Artifactory** button to trigger the release build.

Target server is Artifactory Pro?

If the target Artifactory server is a Pro edition, you can change the target repository, (the default is the release repository configured in Artifactory publisher) and add a staging comment which is included in the build info deployed to Artifactory.

Promoting a Release Build

You can promote a release build after it completes successfully.

This is not a mandatory step but is very useful because it allows you to mark the build as released in Artifactory, and move or copy the built artifacts to another repository so they are available to other users.

To promote a build, browse to the build's result page and click the **Artifactory Release Promotion** link.

Artifactory Pro Required

Promotion features are only available with Artifactory Pro

Result: ● Tests passed: 2

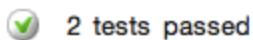
Time: 06 Jul 11 14:47:28 - 14:47:57 (29s)



[Artifactory Build Info](#)

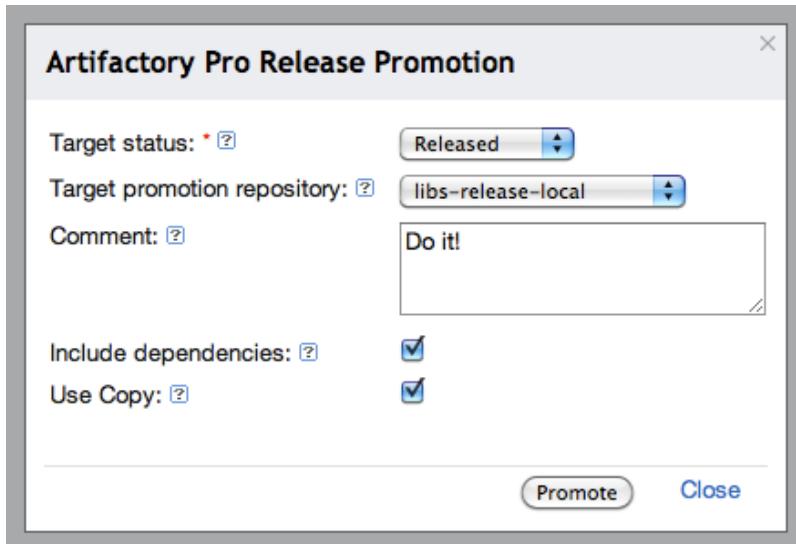


[Artifactory Release Promotion](#)



2 tests passed

Clicking on the link will open the **Release Promotion** dialog:



Select the target status of the build ("Released" or "Rolled-Back"). You may also enter a comment to display in the build in Artifactory.

To move or copy the build artifacts, select the **Target promotion repository**.

Gradle Release Management

The [TeamCity Artifactory Plugin](#) supports release management when running builds with Gradle. This relies on the version property (and others) managed by the `gradle.properties` file. The plugin reads the properties from the Artifactory release management configuration, and modifies those properties in the `gradle.properties` file.

The plugin manages a release using the following basic steps:

1. Modify properties in the `gradle.properties` to release values (before the build starts).
2. Trigger the Gradle build (with optionally different tasks and options).
3. Commit/push changes to the tag (Subversion) or the release branch (Git)
4. Modify the `gradle.properties` to the next integration values.
5. Commit/push changes to the trunk.

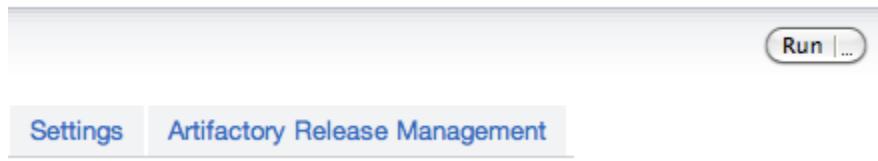
Configuring Gradle Runners

To enable release management for Gradle runners, edit the runner's step configuration and check the **Enable Artifactory release management** checkbox.

Enable Artifactory release management:	<input checked="" type="checkbox"/>
VCS tags base URL/name:	thetag
For subversion this is the URL of the tags location, for Git this is the name of the tag.	
Git release branch name prefix:	REL-BRANCH-
The prefix of the release branch name (applicable only to Git).	
Release properties:	latest-release-version,version
Properties in your project's 'gradle.properties' file whose value should change upon release.	
Next integration properties:	other-version
Properties in your project's 'gradle.properties' file whose value should change upon release, but also for work on the next Integration/development version after the release has been created.	
Alternative Gradle tasks:	clean build
Alternative tasks to execute for a Gradle build running as part of the release. If left empty, the build will use original tasks instead of replacing them.	
Alternative Gradle options:	-I init.gradle
Alternative options to execute for a Gradle build running as part of the release. If left empty, the build will use original options instead of replacing them.	

Staging a Gradle Release Build

Once release management is enabled, the **Artifactory Release Management** tab appears at the top of the build page.



Clicking on the tab reveals configuration options for the release build:

build-info-extractor-maven3-version

Release value:	2.0.3
	The release version to use in the gradle.properties and for tagging.
Next integration value:	2.0.4-SNAPSHOT
	The next development value to use in the gradle.properties.

VCS Configuration

Use release branch

Flag whether to create a Git release branch.

Release branch:	REL-BRANCH-2.0.2
	The name of the Git release branch.

Create VCS tag

Create tag in the version control system.

Tag URL/name:	thetag2.0.2
	The tag URL. Build will fail if the tag already exists.

Tag comment:	[artifactory-release] Release version 2.0.2
	The comment to use when creating the tag.

Next development version comment:	[artifactory-release] Next development version
	The comment to use when committing changes for the next development version.

Artifactory Configuration

Repository to stage the release to (Artifactory Pro):	libs-release-local
	Target repository to stage the release published artifacts to.

Staging comment:	
	Comment that will be added to the promotion action.

The **Release staging** tab displays the **Release** and **Next development** properties configured for the runner. The plugin reads these values from the `gradle.properties` file and attempts to calculate and display **Release** and **Next integration version** in the text fields.

If **Create VCS tag** is checked (default), the plugin commits/pushes the POMs with the release version to the version control system with the commit comment. When using Git, if **Use release branch** is checked, the **Next release version** changes are carried out on the release branch instead of the current checkout branch. The final section allows you to change the target repository (the default is the release repository configured in Artifactory publisher) and an optional staging comment which includes the build info deployed to Artifactory.

Click on the **Build and Release to Artifactory** button to trigger the release build.

Promoting a Release Build

Promotion is the same as in [Promoting a Release Build for Maven](#).

Working with Subversion

Release management with [TeamCity Artifactory Plug-in](#) supports Subversion when using one checkout directory.

During the release the plugin does the following:

1. Commits the release version directly to the tag (if **Create tag** is checked). The release version is not committed to the working branch.
2. Commits the next development version to the working branch

Working with Git

Release management with [TeamCity Artifactory Plug-in](#) supports Git, however the project must perform source checkout on the agent side. If the build is configured to checkout on the server, the release plugin modifies the checkout configuration for the duration of the build and reverts the change when the build is performed. The remote URL should allow Read+Write access.

During the release, the plugin performs the following steps:

1. If **Create Branch** is checked, create and switch to the release branch.
2. Commit the release version to the current branch.
3. Create a release tag.
4. Push the changes.
5. Switch to the checkout branch and commit the next development version.
6. Push the next development version to the working branch

Git support

When using Release Management - Git should be configured via SSH.

Changes

Changes are only committed if the files are modified (POM files or `gradle.properties`)

Bamboo Artifactory Plug-in

Overview

Artifactory provides tight integration with Bamboo through the Bamboo Artifactory Plug-in. Beyond managing efficient deployment of your artifacts to Artifactory, the plug-in lets you capture information about artifacts deployed, dependencies resolved, environment data associated with the Bamboo build runs and more, that effectively facilitates fully traceable builds.

Build Runner Support

The Bamboo Artifactory Plug-in currently provides full support for **Maven 3**, **Gradle** and **Ivy** builds. **Generic Deployment Tasks** are available for all builder types.

Before you begin

Please refer to the general information about [Artifactory's Build Integration](#) before using the Bamboo Artifactory Plug-in.

Source Code Available!

The Bamboo Artifactory Plugin is an open-source project on GitHub which you can freely browse and fork.

Installing the Plug-in

Requirements

- Artifactory 2.2.5 or later. For best results and optimized communication, we recommend using the latest version of Artifactory.
- [Artifactory Pro](#) is required for advanced features, such as [License Control](#) and enhanced [Build Integration](#).
- Atlassian Bamboo 2.6.0 or later running on JDK 1.6 or later.
- Maven 3.
- Gradle 0.9 or later.
- Ant and Ivy 2.1.0 or later.

Installation

Download the latest version of the plugin according to the compatibility matrix below:

Remove older versions

If you have an older version of the plug-in, be sure to remove it before upgrading to a newer one

Bamboo Version	Artifactory Plugin Version	
5.4	1.7.0+	Download
5.4	1.7.1+	Download
5.4	1.7.2+	Download
5.4	1.7.3+	Download
5.6	1.7.4+	Download
5.6	1.7.5+	Download
5.7.x	1.7.6+	Download
5.8.x	1.7.7+	Download
5.9.x	1.8.0+	Download
5.9.x	1.8.1+	Download
5.9.x	1.8.2	Download

Page Contents

- Overview
 - Build Runner Support
- Installing the Plug-in
 - Requirements
 - Installation
 - Plugins are deployed to Bamboo by placing the packaged plugin into the `$BAMBOO_INSTALLATION_HOME/atlassian-bamboo/WEB-INF/lib` folder and restarting Bamboo.
- Configuration
 - Configuring Maven 3, Gradle and Ivy Builders
 - Configuring System-wide Artifactory Server(s)
 - Configuring a Project Builder
 - Selecting Artifactory Servers and Repositories
 - Running Licence Checks
 - Black Duck Code Center Integration
 - The Artifactory Generic Resolve Task
 - The Artifactory Generic Deploy Task
 - The Artifactory Deployment Task
 - Attaching Searchable Parameters
 - Controlling the Build JDK using Bamboo Variables
 - Release Management
 - Push to Bintray
- Running a Build
- License
- Changelog

Plugins are deployed to Bamboo by placing the packaged plugin into the `$BAMBOO_INSTALLATION_HOME/atlassian-bamboo/WEB-INF/lib` folder and restarting Bamboo.

For more details please refer to the Bamboo documentation for [Installing a New Plugin](#).

Configuration

To use the Bamboo Artifactory plug-in you need to set up your Artifactory server(s) in Bamboo's server configuration. You can then set up a project builder to deploy artifacts and build-info to a repository on one of the configured Artifactory servers.

Configuring Maven 3, Gradle and Ivy Builders

Before you begin with any Artifactory-specific setup, ensure that Maven 3, Gradle and/or Ivy builders are available for project configurations.

These builders are defined as **Server Capabilities** in Bamboo

To define Server Capabilities for builders:

1. Under the **Administration** menu, select **Overview** to view the **Bamboo administration** page.
2. Then, under **Build Resources** select **Server Capabilities**
3. Select **Executable** as the **Capability Type**
4. Select **Artifactory Maven 3, Artifactory Gradle** or **Artifactory Ivy** as the type from the **Types** list.
5. Make sure that **Path** points to an installation directory of the selected builder type.

Add capability

Capability type	Executable
Type	Ant
Executable label	Ant Artifactory Generic Deploy Artifactory Gradle Artifactory Ivy Artifactory Maven 3
Path	Command Grails Maven 1.x Maven 2.x Maven 3.x MSBuild NAnt NUnit Runner PHPUnit PHPUnit 3.3.x Visual Studio

Configuring System-wide Artifactory Server(s)

To make Artifactory servers available to project configurations, they must be defined under **Administration | Plugins | Artifactory Plugin**.

Enter the Artifactory server URL in the **Add New Server Configuration** fields.

Username and Password

Username and password are optional and are only used when querying Artifactory's REST API for a list of configured repositories (credentials are only required if the target instance does not allow anonymous access).

Manage Artifactory Servers

You can use this page to add, edit and delete Artifactory server configurations.

Artifactory Server URL	Username	Timeout	Operations
http://[REDACTED]:8081/artifactory	admin	300	Edit Delete

Add New Server Configuration

Artifactory Server Details

Configure an Artifactory server that will be available to project configurations for deployment of artifacts and build info. If anonymous user is enabled in Artifactory server, you can leave the username/password empty

Artifactory Server *****

URL
Specify the root URL of your Artifactory installation, like
`http://repo.jfrog.org/artifactory`

Username

User with permissions to query the list of Artifactory repositories. Leave empty if anonymous is enabled.

Password

Password of the user with permissions to query the list of Artifactory repositories. Leave empty if anonymous is enabled.

Request Timeout *****

Network timeout in seconds to use both for connection establishment and for unanswered requests.

Save **Test** **Cancel**

Configuring a Project Builder

To set up a project task to deploy build-info and artifacts to Artifactory:

1. Go to the **Tasks** step of your jobs configuration.
2. When adding a task type, select the Artifactory Maven 3, Gradle or Ivy builder.

Task types

All
Builder
Tests
Deployment
Source Control

	Artifactory Generic Deploy Artifactory Generic Deploy Task		Artifactory Generic Resolve Artifactory Generic Deploy Task
	Artifactory Gradle Artifactory Gradle task		Artifactory Ivy Artifactory Ivy task
	Artifactory Maven 3 Artifactory Maven 3 Task		Command Execute a globally defined command
	Deploy Tomcat Application Deploys a war file to a Tomcat server		Grails Execute Grails commands as part of your build

Get more tasks on the [Atlassian Marketplace](#) or write your own

Cancel

3. The builder configuration fields appear and include Artifactory and build-info configuration options.

Selecting Artifactory Servers and Repositories

Select an Artifactory server URL for resolving artifacts and to deploy build artifacts in the corresponding fields.

If you have configured the [System Wide Artifactory Servers](#) correctly with the required credentials, then once you select an Artifactory server, the corresponding fields are populated to let you choose a **Resolution Repository** and **Target Repository**.

Repository list empty?

If the **Resolution Repository** or **Target Repository** fields remain empty, check that you have entered valid credentials when defining the Artifactory servers.

The **Target Repository** field is populated with a list of available target repositories as returned by the server (queried with the credentials in the server configuration, if provided).

If the repository list remains empty, ensure the specified Artifactory server URL and credentials (if provided) are valid.

Select the target repository you want Bamboo to deploy artifacts and build-info to.

Resolve artifacts from Artifactory

Check if you wish all dependency resolution to go through Artifactory.

Notice: this will override any external repository definition in Maven settings or POM files.

Resolution Artifactory Server URL

▾

Select an Artifactory server.

Resolution repository

▾

Resolver Username

Name of a user with read permissions on the target repository.

Resolver Password

Password of a user with read permissions on the target repository.

Artifactory Server URL

▾

Select an Artifactory server.

Target Repository

▾

Select a target deployment repository.

Deployer Username

Name of a user with deployment permissions on the target repository.

Deployer Password

The password of the user entered above.

Running Licence Checks

If you have an Artifactory Pro license, you can set the **Run License Checks** checkbox so that Artifactory will scan all dependencies used by the build to check for any license violations.

This feature offers the following options:

Send License Violation Notifications to	A list of email addresses of users who should receive license violation notifications.
Limit Checks to the Following Scopes	The Maven dependency scopes on which the license check should be run. If left empty, all scopes will be checked.
Include Published Artifacts	Indicates that any build artifacts that are dependencies for other build artifacts, should also be scanned for license violations
Disable Automatic License Discovery	Tells Artifactory not to try and automatically analyze and tag the build's dependencies with license information upon deployment. You can still attach license information manually by running Auto-Find from the build's Licenses tab in Artifactory.

Run License Checks (Requires Pro)

Check if you wish that automatic license scanning will occur after build is complete.

Send License Violation Notifications to

admin@company.com

Whitespace-separated list of recipient addresses.

Limit Checks To The Following Scopes

Compile

Space-separated list of scopes.

Include Published Artifacts

Include the builds published module artifacts in the license violation checks if they are

Disable Automatic License Discovery

Tells Artifactory to not try and automatically analyze and tag the builds dependencies w
Auto-Find from the builds Licenses tab in Artifactory.

Black Duck Code Center Integration

If you have an Artifactory Pro license, and a Black Duck Code Center account you can use the [Artifactory Black Duck Code Center integration](#) to automatically initiate an open source component approval process, and proactively monitor for security vulnerabilities.

This feature offers the following options:

Code Center application name	The name of the Black Duck Code Center application that should be invoked.
------------------------------	--

Code Center application version	The Black Duck Code Center application version.
Send compliance report email to	A list of email addresses of users who should receive license violation notifications.
Limit checks to the following scopes	The Maven dependency scopes on which the compliance check should be run. If left empty, all scopes will be checked.
Include Published Artifacts	Indicates that any build artifacts that are dependencies for other build artifacts, should also be scanned for license violations
Auto create missing component requests	Automatically create missing components in Black Duck Code Center application after the build is completed and deployed in Artifactory.
Auto discard stale component requests	Automatically discard stale components in Black Duck Code Center application after the build is completed and deployed in Artifactory.

- Run Black Duck Code Center compliance checks (requires Artifactory Pro)

Check if you wish that automatic Black Duck Code Center compliance checks will occur after the build is completed.

Code Center application name

Salamtak API

The existing Black Duck Code Center application name.

Code Center application version

1.0

The existing Black Duck Code Center application version.

Send compliance report email to:

admin@company.com

Input recipients that will receive a report email after the automatic Black Duck Code Center compliance checks finished.

Limit checks to the following scopes:

Compile

A list of dependency scopes/configurations to run Black Duck Code Center compliance checks on. If left empty all dependencies from all scopes will be checked.

Include published artifacts

Include the builds published module artifacts in the Black Duck Code Center compliance checks if they are also used as dependencies for other modules in this build.

Auto-create missing component requests

Auto create missing components in Black Duck Code Center application after the build is completed and deployed in Artifactory.

Auto-discard stale component requests

Auto discard stale components in Black Duck Code Center application after the build is completed and deployed in Artifactory.

The Artifactory Generic Resolve Task

The Generic Resolve task can be used in any job with any combination of tasks.

It lets you specify dependency patterns for published artifacts that should be downloaded from Artifactory. You can have detailed control over which artifacts are resolved and downloaded by using query-based resolution, adding a query with the properties that the artifact should have before it can be downloaded, to your artifact paths. For more details please refer to [Resolution by Properties](#).

The resolved files will be clustered as dependencies in the Build information.

Artifactory Generic Resolve

Artifactory Server URL

▾

Select an Artifactory server.

Resolution repository

▾

Resolver Username

Name of a user with read permissions on the target repository.

Resolver Password

Password of a user with read permissions on the target repository.

Edit Resolved Artifacts

```
libs-release-local:org/jfrog/test/multi/maven-metadata.xml  
libs-release-local:**/*.*sh@generic free style#LATEST
```

Save

[Cancel](#)

The Artifactory Generic Deploy Task

The Generic Deploy task can be used in any job with any combination of tasks, and is provided to offer minimal Build Info support for all types.

This task collects all available information regarding the build from Bamboo, and provides a deployment mechanism for the artifacts produced.

Adding the Generic Deploy task automatically deploys Build Info collected from the Published Artifacts declaration in addition to the artifacts themselves.

In addition to specifying the **Artifactory Server URL**, the **Target Repository** and the **Deployer username** and **Password**, you can specify the following parameters:

The 'Published Artifacts' declaration lets you specify which artifact files produced by the build are published to Artifactory. At build conclusion, the plugin locates artifacts in the build's checkout directory according to the specified artifact patterns and publishes them to Artifactory, optionally applying mapping for the target path of each deployed artifact.

Make sure to add the Generic Deploy task as a final step!

Artifactory Generic Deploy configuration

Task description

Disable this task

Artifactory Generic Deploy

Artifactory Server URL

Select an Artifactory server.

Save

Cancel

The Artifactory Deployment Task

The Bamboo Artifactory Plugin also supports Bamboo Deployment projects (read more about Deployment projects [here](#)). The Artifactory Deployment task collects the build artifacts which are shared by the build plan associated with the deployment task, and uploads them to Artifactory.

The screenshot shows the 'Artifactory Deployment configuration' dialog box. On the left, there's a sidebar with a tree view of tasks: 'Clean working directory task', 'Artifact download', 'Download release contents', 'Artifactory Deployment' (which is selected and highlighted in blue), and 'Final tasks'. Below the sidebar is a note: 'Drag tasks here to make them final' and a 'Add task' button. The main area contains the following fields:

- Task description:** 'artifactory deployment'
- Disable this task:**
- Artifactory Deployment:**
- Artifactory Server URL:** 'http://localhost:8081/artifactory'
- Select an Artifactory server:**
- Target Repository:** 'ext-release-local'
- Select a target deployment repository:**
- Deployer Username:** 'admin'
- Name of a user with deployment permissions on the target repository:**
- Deployer Password:** '*****'
- The password of the user entered above:**

At the bottom are 'Save' and 'Cancel' buttons.

The "Artifacts Download" Task

The **Artifacts Download** task must be prior to the **Artifactory Deployment** task in the Deployment job flow.

The Artifacts Directory

We recommend configuring a subdirectory for the **Artifacts Download** task.

Attaching Searchable Parameters

You can define parameters that should be attached to build info and artifacts that are deployed by the plugin.

To define a parameter, under **Administration** go to **Build Resources | Global Variables**, fill in a **Key/Value** pair and click **Save**.

The available parameter types are:

- `buildInfo.property.*` - All properties starting with this prefix will be added to the root properties of the build-info.

- `artifactory.deploy.*` - All properties starting with this prefix will be attached to any produced artifacts that are deployed.

Using a Properties File

Instead of defining properties manually, you can point the plug-in to a properties file.

To do so, define a property named `buildInfoConfig.propertiesFile` and set its value to the absolute path of the properties file.

Global Variables

You can use this page to view, add, edit and delete global variables. Global variables are available on every build run in Bamboo and can be accessed using `#{bamboo.globalVarName}`

Key	Value	
<code>artifactory.deploy.moo</code>	mcmoo	<input type="button" value="Add"/>
<code>buildInfo.property.bob</code>	mcbob	<input type="button" value="Delete"/>

The given path and file should be present on the machine that is running the build agent, not the server.

Controlling the Build JDK using Bamboo Variables

For an Artifactory Maven, Gradle or Ivy task, you can define variables from the task configuration page. In the case of the Build JDK, Bamboo Artifactory Plugin lets you define this using Bamboo variables.

You can define them as global or plan-specific variables as follows:

<code>artifactory.task.override.jdk</code>	If set to true, check the value of artifactory.task.override.jdk.env.var . If that variable is populated with an environment variable, use the value of that environment variable as the Build JDK path. If artifactory.task.override.jdk.env.var is not defined, use the value of <code>JAVA_HOME</code> for the Build JDK.
<code>artifactory.task.override.jdk.env.var</code>	Stores the name of another environment variable whose value should be used for the build JDK.

Release Management

The Artifactory Plugin provides a powerful feature for release management and promotion. For details please refer to Bamboo Artifactory Plugin - Release Management.

Push to Bintray

Bintray users can publish build artifacts to Bintray by using the Artifactory Bintray integration.

This can be done on the **Push to Bintray** tab of the Bamboo Artifactory Plugin in one of two ways:

1. You can configure your Bintray details in a [descriptor file](#) which should be added to your list of build artifacts
2. You can check the **Override descriptor file** checkbox and specify the details in the [Push to Bintray tab UI](#).

Using a Descriptor File

1. Create a descriptor file named `bintray-info.json`. You can read about file's structure and see an example under [Using the REST API](#) on the [Push to Bintray](#) page.
2. Commit the descriptor file to your source control along with your project sources.
3. Modify your build script to attach the file to your build artifacts.

Using the "Push to Bintray" Tab UI

1. Check the **Override descriptor file** checkbox in the **Push to Bintray** tab.
2. Fill in the fields that are displayed.

Bintray Required Fields
 For Bintray OSS users, all fields are mandatory.
 For Bintray Pro accounts, the **Licenses** and **VCS URL** fields are optional .

Job Summary Artifactory Build Info Tests Commits Artifacts Logs Metadata Push to Bintray

 Push to Bintray

Override descriptor file

Sign method: Sign

GPG Passphrase:

Push to Bintray

Job Summary Artifactory Build Info Tests Commits Artifacts Logs Metadata Push to Bintray

 Push to Bintray

Override descriptor file

Override descriptor

Subject	my_company
Repository	generic
Package name	new_package
Version	release_1
Licenses	MIT, Apache-1.0
VCS URL	https://github.com/my_code
Sign method	Don't Sign
GPG Passphrase	

Push to Bintray

Running a Build

Once you have completed setting up a project builder you can run it. The Artifactory plug-in commences at the end of the build and:

1. Deploys all artifacts to the selected target repository in one go (as opposed to the deploy at the end of each module build, used by Maven/Ivy).
2. Deploys the Artifactory build-info to the selected server, which provides [full traceability of the build in Artifactory](#), with links back to the build in Bamboo.

```

08-Aug-2010 12:30:07 [INFO] Attaching shaded artifact.
08-Aug-2010 12:30:07 [INFO]
08-Aug-2010 12:30:07 [INFO] --- naven-install-plugin:2.3:install (default-install) @ build-info-extractor-ivy ---
08-Aug-2010 12:30:07 [INFO] Installing /home/noam/Work/atlassian/bamboo/artifactory/plugin/trunk/target/bamboo/home/xml-data/build-dir/MCBOB-DEF/build-info-extractor-ivy/target/build-info-extractor-ivy-1.0-SNAPSHOT.jar
08-Aug-2010 12:30:07 [INFO] /home/noam/.m2/repository/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT/build-info-extractor-ivy-1.0-SNAPSHOT.jar
08-Aug-2010 12:30:07 [INFO] Installing /home/noam/Work/atlassian/bamboo/artifactory/plugin/trunk/target/bamboo/home/xml-data/build-dir/MCBOB-DEF/build-info-extractor-ivy/target/build-info-extractor-ivy-1.0-SNAPSHOT-sources.jar
08-Aug-2010 12:30:07 [INFO] /home/noam/.m2/repository/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT/sources.jar
08-Aug-2010 12:30:07 [INFO] Installing /home/noam/Work/atlassian/bamboo/artifactory/plugin/trunk/target/bamboo/home/xml-data/build-dir/MCBOB-DEF/build-info-extractor-ivy/target/build-info-extractor-ivy-1.0-SNAPSHOT-uber.jar
08-Aug-2010 12:30:07 [INFO] /home/noam/.m2/repository/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT-uber.jar
[INFO] Artifactory Build Info Recorder: Deploying artifacts to http://192.168.1.9:8080/artifactory/
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-client/1.4.x-SNAPSHOT/build-info-client-1.4.x-SNAPSHOT.pom
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-api/1.4.x-SNAPSHOT/build-info-api-1.4.x-SNAPSHOT.pom
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT/build-info-extractor-ivy-1.0-SNAPSHOT-uber.jar
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-maven/3/1.0.x-SNAPSHOT/build-info-extractor-maven-3/1.0.x-SNAPSHOT.pom
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT/build-info-extractor-ivy-1.0-SNAPSHOT.pom
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT-uber.jar
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT-sources.jar
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-maven/3/1.0.x-SNAPSHOT/build-info-extractor-maven-3/1.0.x-SNAPSHOT.pom
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT/build-info-extractor-ivy-1.0-SNAPSHOT-sources.jar
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-maven/3/1.0.x-SNAPSHOT-uber.jar
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT/build-info-extractor-ivy-1.0-SNAPSHOT-sources.jar
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT-uber.jar
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT/build-info-extractor-ivy-1.0-SNAPSHOT.pom
08-Aug-2010 12:30:07 [INFO] Deploying artifact: http://192.168.1.9:8080/artifactory/libs-snapshots-local/org/jfrog/buildinfo/build-info-extractor-ivy/1.0-SNAPSHOT-uber.jar
08-Aug-2010 12:30:07 [INFO] Artifactory Build Info Recorder: Deploying build info ...
08-Aug-2010 12:30:07 [INFO] Deploying build info to: http://192.168.1.9:8080/artifactory/api/build
08-Aug-2010 12:30:07 [INFO]
08-Aug-2010 12:30:07 [INFO] Reactor Summary:
08-Aug-2010 12:30:07 [INFO]
08-Aug-2010 12:30:07 [INFO] JFrog Build-Info ..... SUCCESS [1.503s]
08-Aug-2010 12:30:07 [INFO] JFrog Build-Info API ..... SUCCESS [8.584s]
08-Aug-2010 12:30:07 [INFO] JFrog Build-Info Client ..... SUCCESS [2.485s]
08-Aug-2010 12:30:07 [INFO] JFrog Build-Info Extractor ..... SUCCESS [2.199s]
08-Aug-2010 12:30:07 [INFO] JFrog Build-Info Maven 3 Extractor ..... SUCCESS [4.969s]
08-Aug-2010 12:30:07 [INFO] JFrog Build-Info Ivy Extractor ..... SUCCESS [3.432s]
08-Aug-2010 12:30:07 [INFO] BUILD SUCCESS

```

You can also link directly to the information in Artifactory from a build run view in Bamboo:

The screenshot shows a Bamboo build run details page. At the top, it says "Job: Default Job was successful". Below that, there's a navigation bar with tabs: Job Summary, Tests, Artifactory Build Info, Commits, Artifacts, Logs, Metadata, and Artifactory Release & Promotion. The "Artifactory Build Info" tab is currently active. On the left, there's a sidebar with sections for Stages & jobs, Default Stage (set to Default Job), and a list of stages: JFrog Build-Info, JFrog Build-Info API, JFrog Build-Info Client, JFrog Build-Info Extractor, JFrog Build-Info Maven 3 Extractor, and JFrog Build-Info Ivy Extractor. Each stage has a status indicator (green for success). At the bottom of the sidebar, it says "BUILD SUCCESS".

Since Bamboo Artifactory Plug-in **version 1.7.0**, the Build Info link is under the renamed tab "Artifactory Build Info".

License

The Bamboo Artifactory plug-in is available under the Apache v2 License.

Changelog

▼ Click to see change log details

1.8.2 (27 Oct 2015)

- Bug fixes (BAP-289, BAP-292, BAP-302)

1.8.1 (4 Aug 2015)

- Bug fix (BAP-282)

1.8.0 (15 Jun 2015)

- Add push to Bintray support (BAP-257)
- Make Artifactory Upload Task available for Deployment projects (BAP-264)
- Ability not to promote the version on Gradle Release Staging (BAP-258)
- Bug fixes (BAP-270, BAP-269, BAP-267, BAP-266, BAP-261, BAP-260, BAP-254, BAP-246)

1.7.7 (30 Mar 2015)

1. Support for Bamboo 5.8.x ([BAP-249](#))

1.7.6 (14 Jan 2015)

1. Support for Bamboo 5.7.x ([BAP-230](#))
2. Compatibility with Maven 3.2.5 ([BAP-244](#))
3. Enable overriding the Build JDK value using Bamboo variables ([BAP-240](#))
4. Bug fix ([BAP-241](#))

1.7.5 (10 Nov 2014)

1. Support Atlassian Stash source control management ([BAP-206](#))
2. Artifactory generic Resolve task ([BAP-207](#))
3. Maven 3 tasks - Record Implicit Project Dependencies and Build-Time Dependencies ([BAP-225](#))

1.7.4 (12 Aug 2014)

1. Support for Bamboo 5.6 ([BAP-218](#))
2. Bug fix ([BAP-219](#))

1.7.3 (29 Jul 2014)

1. Add support for Gradle 2.0 ([GAP-153](#))
2. Bug fix ([BAP-212](#))

1.7.2 (25 Jun 2014)

1. Bug fixes ([BAP-196](#), [BAP-208](#), [BAP-166](#))

1.7.1 (26 MAY 2014)

1. A new check box that gives the ability to ignore artifacts that are not deployed according to include/exclude patterns. ([BAP-180](#))

1.7.0 (06 Apr 2014)

1. Fix Support for Bamboo 5.4+
2. Supporting Git Shared Credentials in Release Management functionality ([BAP-189](#))
3. Adding Version Control Url property to the Artifactory Build Info JSON. ([BAP-200](#))
4. Bug fixes ([BAP-197](#))

1.6.2 (24 Nov 2013)

1. Fix Support for Bamboo 5.2
2. Add Artifactory BlackDuck integration
3. Bug fixes ([BAP-182](#) [BAP-184](#) [BAP-186](#) [BAP-184](#))

1.6.1 (03 Oct 2013)

1. Support form Bamboo 5.1.1
2. Bug fixes 1.6.1

1.6.0 (16 Jul 2013)

1. Support form Bamboo 5.0

1.5.6 (03 Sep 2013)

1. Support form Bamboo 4.2

1.5.5 (03 Sep 2012)

1. Support for include/exclude captured environment variables ([BAP-143](#))
2. Bug fixes ([MAP-41](#) [MAP-40](#) [GAP-129](#) [BAP-148](#) [IAP-32](#))

1.5.4 (25 Jun 2012)

1. Support Bamboo 4.1.
2. Bug fixes. ([JIRA](#))

1.5.3 (02 Apr 2012)

1. Support Bamboo 4.0.

1.5.2 (02 Apr 2012)

1. Support Perforce for release management. ([BAP-133](#))
2. Bug fixes. ([JIRA](#))

1.5.1 (05 Jan 2012)

1. Compatible release plugin for version 3.4.2. ([BAP-116](#))
2. Support for Gradle properties deployment. ([BAP-117](#))
3. Unique icon for each Artifactory task type.
4. Setting Bamboo job requirements correctly for all builder types. ([BAP-125](#))

1.5.0 (11 Dec 2011)

1. Compatible with bamboo version 3.3.x.
2. Compatible with Gradle 1.0-milestone-6.

1.4.2 (19 Sep 2011)

1. Bug fix ([BAP-91](#))

1.4.1 (01 Aug 2011)

1. Support for Bamboo 3.2.x
2. Bug fix ([BAP-90](#))

1.4.0 (14 Jul 2011)

1. Introducing Release Management capabilities.
2. Generic Build Info support for all job types.
3. Bug fixes.

1.3.2 (14 Jun 2011)

1. Bug fix ([BAP-65](#))

1.3.1 (13 Jun 2011)

1. Bug fix ([BAP-64](#))

1.3.0 (30 May 2011)

1. Support for Bamboo 3.1.x

1.2.0 (2 Mar 2011)

1. Support for Bamboo 3.x

1.1.0 (2 Jan 2011)

1. Gradle Support - Gradle builds are now fully supported with the new Gradle builder
2. Ivy builds now support custom Ivy patterns for artifacts and descriptors
3. Support for Bamboo 2.7.x

1.0.3 (21 Nov 2010)

1. Add Include/exclude pattern for artifacts deployment
2. Bug fix ([BAP-26](#))

1.0.2 (7 Nov 2010)

1. Control for including published artifacts when running license checks
2. Limiting license checks to scopes
3. Control for turning off license discovery when running license checks

Bamboo Artifactory Plugin - Release Management

Overview

Artifactory supports release management through the [Bamboo Artifactory Plugin](#).

When you run your builds using [Maven](#) or [Gradle](#) with jobs that use [Subversion](#), [Git](#) or [Perforce](#) as your version control system, you can manually stage a release build allowing you to:

- Change values for the release and next development version
- Choose a target staging repository for deployment of the release, and
- Create a VCS tag for the release.

Staged release builds can later be **promoted** or **rolled-back**, changing their release status in Artifactory and, optionally, moving the build artifacts to a different target repository.

Inside Artifactory, the history of all build status change activities (staged, promoted, rolled-back, etc.) is [recorded and displayed](#) for full traceability.

When release management is enabled, the Artifactory release staging link appears on the top header bar in the job page.

Displaying the Release and Promotion Tab

To display the **Artifactory Release & Promotion** tab you need to click the small arrow indicated below.



The screenshot shows a job summary page for job #10. The top navigation bar includes tabs for Job Summary, Artifactory Build Info, Tests, Commits, Artifacts, Logs, Metadata, and Artifactory Release & Promotion. The 'Artifactory Release & Promotion' tab is highlighted with a red box. Below the tabs, there's a 'Job result summary' section with details like completion time, duration, and agent information. A small red box highlights the small arrow icon next to the tab name.

Release management tab moved from plugin version 1.7.0

From Bamboo Artifactory Plugin version 1.7.0, the Release Management tab was moved from the **Plan** page level to the **Job** page level because the process applies to artifacts in the context of a single job rather than a whole plan (which may hold several jobs).

The tab name was also changed from **Artifactory Release management** to **Artifactory Release & Promotion**.

Page Contents

- Overview
- Maven Release Management
 - Configuring Maven Jobs
 - Staging a Maven Release Build
- Gradle Release Management
 - Configuring Gradle Jobs
 - Staging a Gradle Release Build
- Promoting a Release Build
- Working with Subversion
- Working with Git
- Working with Perforce

Maven Release Management

The Bamboo Artifactory Plugin manages a release with Maven running the build only once using the following basic steps:

1. Change the POM versions to the release version (before the build starts).
2. Trigger the Maven build (with optionally different goals).
3. Commit/push changes to the tag (Subversion) or the release branch (Git).
4. Change the POM versions to the next development version.
5. Commit/push changes to the trunk.

If the build fails, the plugin attempts to rollback the changes (both local and committed).

For more information including configuration of Maven Runners, and Jobs and staging a release build, please refer to [Bamboo Artifactory Plugin](#).

Configuring Maven Jobs

To enable release management in Maven jobs, edit the job configuration and check the **Enable Artifactory release management** checkbox.

Enable Release Management

Enable Release Management to Artifactory

VCS tags base URL/name

RELEASE-TAG-

For subversion this is the URL of the tags location, for Git this is the name of the tag.

Git release branch name prefix

REL-BRANCH-

The prefix of the release branch name (applicable only to Git).

Alternative Maven tasks and options

clean install -Prelease

Alternative Maven and options to execute for a Maven build running as part of the release. If left empty, the build will use original tasks and options instead of replacing them.

Staging a Maven Release Build

Clicking on the release staging link opens a new page with configuration options for the release build:



Artifactory Pro Release Staging

- Module Version Configuration
- One version for all modules.
 - Version per module
 - Use existing module versions

Release Value

Next Integration Value

VCS Configuration

Create VCS Tag

Tag URL/name:

Tag comment

Next development version comment

Publishing Repository
Select a publishing repository.

Staging Comment:

The release staging page displays the last version built (the version tag is that of the root POM, and is taken from the last build that is not a release). Most of the fields in the form are populated with default values.

Version configuration controls how the plugin changes the version in the POM files (global version for all modules, version per module or no version changes).

If the **Create VCS tag** checkbox is checked (default), the plugin commits/pushes the POMs with the release version to the version control system with the commit comment. When using Git, there's also an option to create a release branch.

Click on the **Build and Release to Artifactory** button to trigger the release build.

Target server is Artifactory Pro?

If the target Artifactory server is a Pro edition, you can change the target repository, (the default is the release repository configured in Artifactory publisher) and add a staging comment which is included in the build info deployed to Artifactory.

The [Bamboo Artifactory Plugin](#) supports release management when running builds with Gradle. This relies on the version property (and others) managed by the `gradle.properties` file. The plugin reads the properties from the Artifactory release management configuration, and modifies those properties in the `gradle.properties` file.

The plugin manages a release using the following basic steps:

1. Modify properties in the `gradle.properties` to release values (before the build starts).
2. Trigger the Gradle build (with optionally different tasks and options).
3. Commit/push changes to the tag (Subversion) or the release branch (Git)
4. Modify the `gradle.properties` to the next integration values.
5. Commit/push changes to the trunk.

Configuring Gradle Jobs

To enable Gradle release management, edit the Artifactory Gradle Task configuration and check the **Enable Release Management** checkbox.

Enable Release Management

Enable Release Management to Artifactory

VCS tags base URL/name

For subversion this is the URL of the tags location, for Git this is the name of the tag.

Git release branch name prefix

The prefix of the release branch name (applicable only to Git).

Release properties

Properties in your projects `gradle.properties` file whose value should change upon release.

Next integration properties

Properties in your projects `gradle.properties` file whose value should change upon release, but also for work on the next integration/development version after the release has been created.

Alternative Gradle tasks and options

Alternative tasks and options to execute for a Gradle build running as part of the release. If left empty, the build will use original tasks and options instead of replacing them.

Staging a Gradle Release Build

Once release management is enabled, the Artifactory **Release staging** tab appears in the top header bar on the job page.

Clicking on the **Release staging** tab opens a new page with configuration options for the release build:



Artifactory Pro Release Staging

Property Key	wharf-core-version
Current Value	1.49-SNAPSHOT
Release Value	1.49
Next Integration Value:	1.50-SNAPSHOT

VCS Configuration

<input checked="" type="checkbox"/> Use Release Branch:	
Release branch:	REL-BRANCH-1.49
<input checked="" type="checkbox"/> Create VCS Tag	
Tag URL/name:	1.49
Tag comment	[artifactory-release] Release version 1.49
Next development version comment	[artifactory-release] Next development version
Publishing Repository	libs-release-local <input type="button" value="▼"/>
	Select a publishing repository.
Staging Comment:	

The **Release staging** tab displays the **Release** and **Next development** properties configured for the job. The plugin reads these values from the `gradle.properties` file and attempts to calculate and display **Release** and **Next integration version** in the text fields.

If **Create VCS tag** is checked (default), the plugin commits/pushes the POMs with the release version to the version control system with the commit comment. When using Git, if **Use release branch** is checked, the **Next release version** changes are carried out on the release branch instead of the current checkout branch. The final section allows you to change the target repository (the default is the release repository configured in Artifactory publisher) and an optional staging comment which includes the build info deployed to Artifactory.

Click on the **Build and Release to Artifactory** button to trigger the release build.

Promoting a Release Build

You can promote a release build after it completes successfully.

This is not a mandatory step but is very useful because it allows you to mark the build as released in Artifactory, and move or copy the built artifacts to another repository so they are available to other users.

To promote a build, browse to the build's result page and click the **Artifactory Release & Promotion** tab.

Artifactory Pro required

Promotion features are only available with Artifactory Pro

Job Summary

Tests

Artifactory Build Info

Commits

Artifacts

Logs

Metadata

Artifactory Release & Promotion



Artifactory Release Promotion

Promotion Mode Normal

Target Status

Released

Comment

(empty text area)

Target promotion repository

Select a promotion repository.

Include dependencies

Use Copy

Update

Select the target status of the build ("Released" or "Rolled-Back"). You may also enter a comment to display in the build in Artifactory.

To move or copy the build artifacts, select the **Target promotion repository**.

Release management

From Bamboo Artifactory Plug-in version 1.7.0, the Artifactory Release Promotion was moved from the **Artifactory** tab to the new **Artifactory Release & Promotion** tab.

Working with Subversion

Release management with [Bamboo Artifactory Plugin](#) supports Subversion when using one checkout directory.

During the release the plugin does the following:

1. Commits the release version directly to the tag (if **Create tag** is checked). The release version is not committed to the working branch.
2. Commits the next development version to the working branch

Working with Git

To work with Git, the Git plugin must be configured to build one branch **AND** to checkout to the same local branch.

The remote URL should allow Read+Write access.

The [Bamboo Artifactory Plugin](#) uses the Git client installed on the machine and uses its credentials to push back to the remote Git repository.

Source Repository

Source Control ?

Repository URL* ?
The URL of Git repository.

Branch ?
The name of the branch (or tag) containing source code.

Authentication Type ?

SSH Key
SSH private key you want to use to access the repository.

SSH Passphrase

Use shallow clones
Fetches the shallowest commit history possible. Do not use if your build depends on full repository history.

Source Repository

Source Control ?

Username*
The GitHub user required to access the repositories.

Change password?

Repository
Select the repository you want to use for your Plan.

Branch ?
Choose a branch you want to check out your code from.

Use shallow clones
Fetches the shallowest commit history possible. Do not use if your build depends on full repository history.

During the release, the plugin performs the following steps:

1. If **Create Branch** is checked, create and switch to the release branch.
2. Commit the release version to the current branch.
3. Create a release tag.
4. Push the changes.
5. Switch to the checkout branch and commit the next development version.
6. Push the next development version to the working branch

Shallow Clones

Bamboo's Git plugin allows the use of shallow clones, however this causes the "push" not to work.

Therefore, when using the Artifactory Bamboo Plugin, you must have shallow clones **unchecked**.

For more information about shallow clones, please refer to [git-clone Manual Page](#).

Release management with [Bamboo Artifactory Plugin](#) supports Perforce when using one checkout directory.

During the release the plugin does the following:

1. Commits the release version directly to the tag (if **Create VCStag** is checked). The release version is not committed to the working branch.
2. Commits the next development version to the working branch

Changes

Changes are only committed if the files are modified (POM files or `gradle.properties`).

MSBuild Artifactory Plugin

Overview

Artifactory brings Continuous Integration to MSBuild, TFS and Visual Studio through the MSBuild Artifactory Plugin. This allows you to capture information about deployed artifacts, resolve Nuget dependencies and environment data associated with MSBuild build runs, and deploy artifacts to Artifactory. In addition, the exhaustive build information captured by Artifactory enables fully traceable builds.

MSBuild Artifactory Plugin

The MSBuild Artifactory plugin supports programs written for .NET Framework 4.5 and above, and is an [open source project on GitHub](#) which you can freely browse and fork.

Sample code

To get yourself started, [here is an example](#) of a solution with multiple projects that use the MSBuild Artifactory Plugin

The MSBuild Artifactory Plugin can be used whether you are running standalone builds or using a CI server. In either case, you should note the following points:

1. Standalone Integration

The MSBuild Artifactory Plugin fully integrates with the MSBuild process, so it can run as part of a standard build.

The plugin uses a conventional MSBuild XML configuration file to influence different stages of the build process.

2. CI Server Integration

When running MSBuild builds in your continuous integration server, using the plugin is transparent since it is effectively an integral part of the MSBuild process. The only difference is that the plugin collects information from the CI server.

The MSBuild Artifactory Plugin fully supports TFS and collects exhaustive build information to enable fully traceable builds. Support for additional CI servers such as Jenkins, TeamCity and Bamboo is partial, and the build information collected when running with these tools is correspondingly partial.

Page Contents

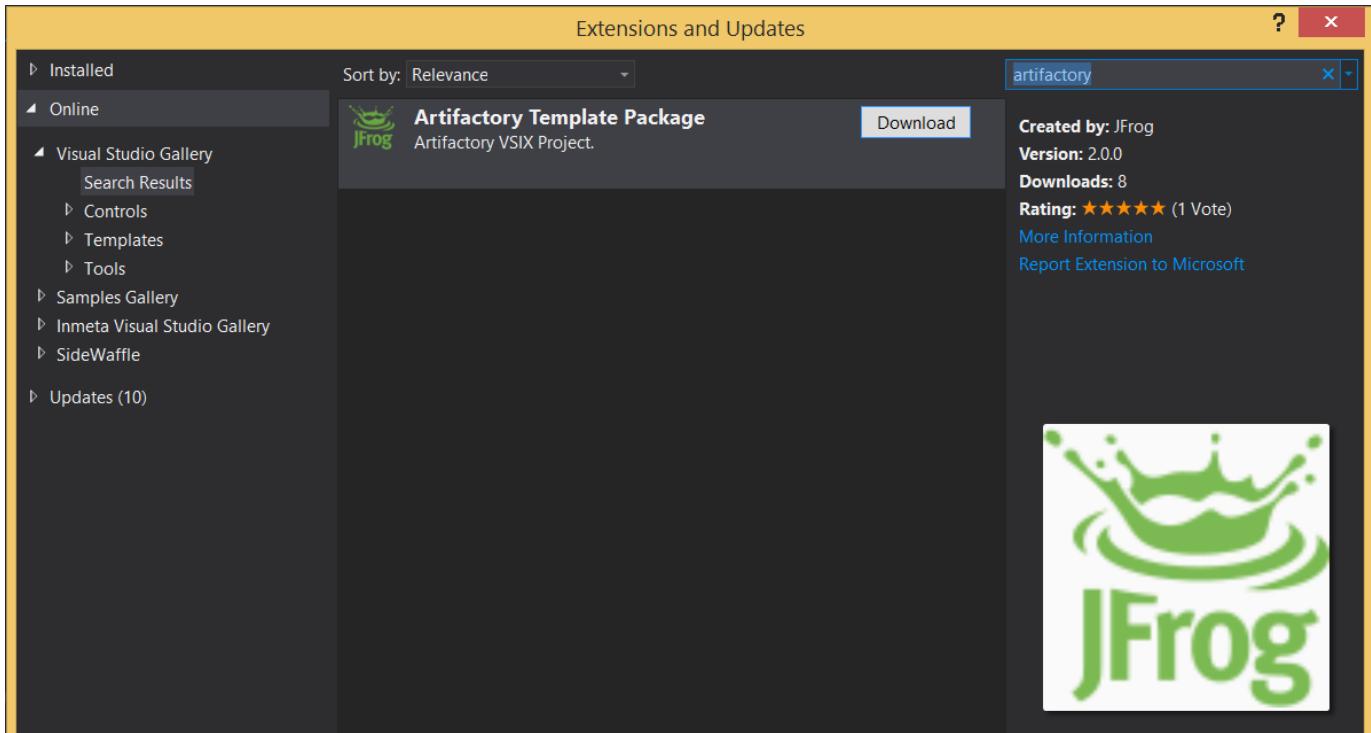
- Overview
 - MSBuild Artifactory Plugin
- Installation
- Update
- Uninstalling
- Migration from the old plugin implementation
- Configuration
 - General Information
 - Resolution
 - Deployment
 - Checksum
 - Deployment

- Project-specific Deployment
- Environment Variables
- License Control
- Black Duck Code Center Integration
- Network Configuration
 - Deploying via Proxy
- Running a build with MSBuild Artifactory Plugin
- Team Foundation Server (TFS) Integration
 - MSBuild Arguments in TFS
 - Package Restore with Team Foundation Build
- License
- Changelog

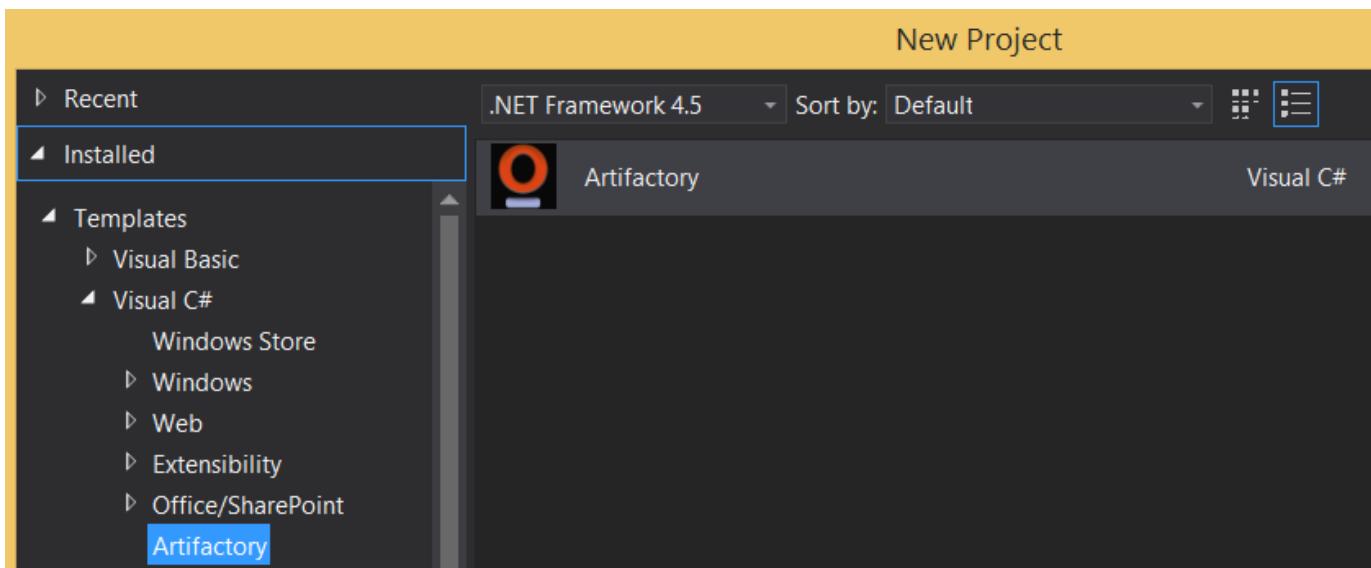
Installation

The MSBuild Artifactory Plugin is installed as a "Project Template" using Visual Studio as follows:

- Under **Tools**, choose **Extensions and Updates...**, select the **Visual Studio Gallery** source under the **Online** section, and run a search for "Artifactory".
- Select **Artifactory Template Package** extension found, and click **Install**.



- Once the installation is complete, select the **Solution** into which you want to install the plugin. right-click the solution node and select **Add | New Project....** Select **Artifactory** and click OK.

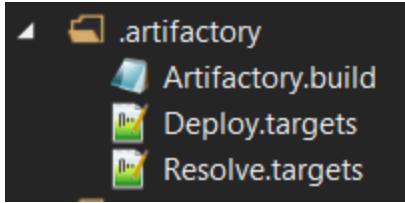


Creating Artifactory project

To resolve Nuget dependencies, the plugin requires a `.nuget` directory within your Solution. Before you start installation, make sure that this directory exists.

You should see the following changes to your Solution:

- A new custom project linked to the Artifactory plugin.
- A `.artifactory` folder is added to your Solution. Make sure this directory is committed to source control.
- Under the `.artifactory` folder, an `Artifactory.build` file is created. This is the main plugin configuration file.
- Two `.targets` files are created. These are used internally by the plugin and should not be modified manually.



- New [Nuget](#) packages related to the plugin should have been installed into the Solution.
- The plugin also imports its own MSBuild configuration file to the `artifactory.csproj` and `.nuget\NuGet.targets` files in order to be added to the MSBuild process.

The last step will be to link your relevant projects to the Artifactory project via the added **Project Reference**. Only the linked projects will be monitored by the plugin.

Installing plugin dependencies

The following two dependencies: [Nuget.core v2.8.2](#) and [Microsoft.Web.Xdt v2.1.1](#) must also be installed together with the plugin.

If you are installing the plugin from an instance of Artifactory, you need to ensure that Artifactory has access to the plugin and its dependencies. For example, you might have a virtual repository that references a local repository containing the plugin, and a remote repository that references nuget.org. For more details on configuring Artifactory, please refer to [NuGet Repositories](#).

Update

To update Artifactory extensions, execute the following steps:

- Under **Tools**, select **Extensions and Updates...**, and then, the "Visual Studio Gallery" source under the **Updates** section.
- Select the **Artifactory Template Package** extension found, and click **Update**.

Existing Project (Optional Step)

If you already have an existing Artifactory project template in your solution, and you want to update it to the latest one, execute the following steps:

- Right-click on the Artifactory project and select **Manage NuGet Packages...**
- In the **Manage NuGet Packages** window, select a source (e.g. nuget.org) under the **Updates** section.
- Select the **Artifactory** package found, and click **Update**.

Uninstalling

- Remove the `.artifactory` folder and its contents from the solution.
- Remove the custom Artifactory project from the solution.

Make sure to delete the items from the file system also.

Migration from the old plugin implementation

To migrate from the old plugin implementation, you need to uninstall it and then install the new implementation

- Follow the steps described in [Uninstalling](#).
- Follow the steps described in [Installation](#).

Configuration

General Information

The MSBuild Artifactory Plugin is configured in the `Artifactory.build` configuration file. The file is structured using **MSBuild** language conventions, so all the properties can be externally overridden using **Reserved Properties** or **Environment Properties**.

MSBuild can collect properties that were configured in the build scope, or in the Environment Variables. This ability can be helpful in different cases:

- You can dynamically override the plugin configuration according to the build context that it runs in.
- You can prevent sensitive information from being checked into source control.
For example, if the build runs under a build server such as TFS, all the Artifactory credentials can be defined by the server administrator, and will therefore, not be Checked in/Committed to source control.

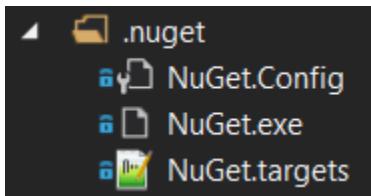
For more information about MSBuild properties, please refer to the [MSBuild Reference Documentation](#).

Configuration Instructions

For more details on configuration, please refer to the [Artifactory.build configuration file](#).

Resolution

To resolve packages, the MSBuild Artifactory Plugin uses the **NuGet Package Restore** feature with the **MSBuild-Integrated Package Restore** approach. To support this, the project that installed the plugin must be a part of a solution with the `.nuget` folder. If the `.nuget` folder is absent, the plugin will not override the Package Restore in the solution.



Please note the following points:

- Even though the plugin is installed on a project level, it overrides the NuGet resolution on **all** the projects under the same solution.
- Manual configuration in the `.nuget/NuGet.Config` file is ignored by the plugin.
- Modifying the `.nuget/NuGet.targets` file can cause unexpected behavior in the resolution process. We strongly recommend that you do not modify these files manually.
- The `.nuget` folder must be committed to source control.

Ensuring package resolution through Artifactory

In order to mitigate situations in which a network connection is not available, the NuGet client locally caches any artifacts downloaded from a remote repository in the **NuGet Local Cache** (under `%AppData%\Local\NuGet\Cache`). Subsequently, the NuGet client first checks the cache when trying to resolve packages. Therefore, artifacts downloaded from a remote repository in Artifactory or from the NuGet Gallery, typically get stored in this local cache and will be provided from the cache next time you try to reference them.

To ensure that the NuGet client resolves packages through Artifactory, you need to delete the NuGet Local Cache.

Deployment

In order to support a wide variety of project templates, solution structures and artifact types, the MSBuild Artifactory Plugin is designed to be very flexible and allows the user great freedom in configuring how to deploy packages.

- Using an **Input Pattern**, the user can specify the path to files that the plugin will collect for deployment. The path is relative to the project in which the plugin is installed, and to other projects referenced by it in the solution.
- Using an **Output Pattern**, the user can specify a deployment path in Artifactory that corresponds to the specified **Input Pattern**.
- The user can also specify **Custom Properties** that should annotate all the artifacts resulting from the specified **Input Pattern**.

Target repository layout

You may define a custom layout for your target repository, but it is up to you to specify the right Output Pattern to ensure that your artifacts are deployed to the right location within the repository. For more details, please refer to [Local Repositories](#).

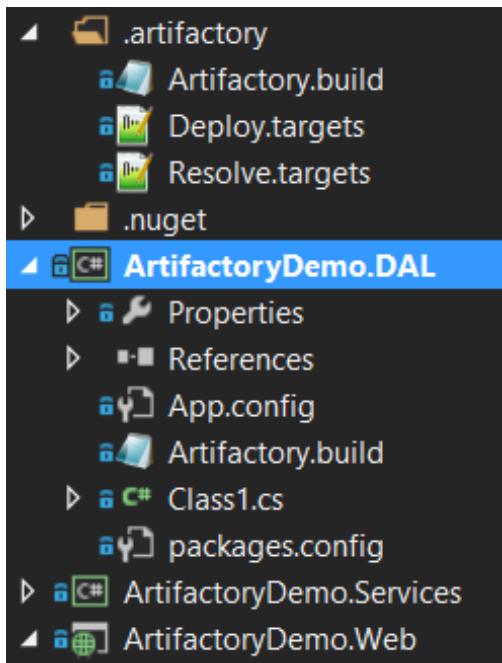
Checksum Deployment

To support Artifactory's "Once-And-Only-Once" Content Storage, the plugin efficiently deploys packages to Artifactory using Checksum Deployment. Before an artifact is actually deployed, the plugin passes its checksum to Artifactory.

If the package already exists then Artifactory does not accept a new copy, it just creates a new metadata entry in the database to indicate that another "copy" of the artifact exists in specified deployment path.

Project-specific Deployment

The `Artifactory.build` file under the `.artifactory` directory applies to all projects within the solution. However, you can override the deployment configuration for a specific project providing an `Artifactory.build` file within the project scope. For example, you could use this to specify the full path of an artifact that needs to be deployed. The plugin detects the project-specific `Artifactory.build` file and applies the deployment configuration to that project, overriding the general deployment configuration. The example below shows the "ArtifactoryDemo.DAL" project with its own `Artifactory.build` file to override the general solution deployment configuration.



Environment Variables

You can enable the **EnvironmentVariables** tag so that MSBuild Artifactory Plugin uses all environment variables accessible by the build process and registers them in the build info. If running under a build server, the server's properties also used and registered. You may define **IncludePatterns** and **ExcludePatterns** to control which variables are included in the published build info.

Pattern wildcards

A pattern may contain the * and the ? wildcards. Include patterns are applied before exclude patterns.

Extensive build information may slow down deployment

Including all environment variables as part of the captured build information may result in very large build objects which in turn, may slow down deployment.

License Control

MSBuild Artifactory Plugin supports Artifactory Pro License Control. This feature is controlled by several tags in the `Artifactory.build` configuration file.

LicenseControl	Enables or disables the license control feature
LicenseViolationRecipients	Specifies addresses of recipients that should receive license alerts by email

AutomaticLicenseDiscovery	<p>When set, Artifactory will analyze and tag the build's dependencies with license information upon deployment</p> <div style="border: 1px solid #f0e68c; padding: 10px;"> <p>Resource intensive</p> <p>Automatic license discovery is a resource intensive operation which may slow down deployment. If you do not run automatic license discovery, you can still attach license information manually by running 'Auto-Find' from the build's Licenses tab in the Artifactory UI. For more details, please refer to Examining Build Licenses.</p> </div>
IncludePublishedArtifacts	<p>License checks are usually only run on the dependencies of the published package. If this flag is set, a license check is run on the deployed artifact itself (only valid for NuGet packages)</p>
ScopesForLicenseAnalysis	<p>Lets you specify the scopes on which license analysis should be run.</p>

Black Duck Code Center Integration

If you are using Artifactory Pro and have an account with [Black Duck](#) Code Center, you can run the build through an automated, non-invasive, open source component approval process, and monitor for security vulnerabilities.

BlackDuckComplianceCheck	<p>Enables or disables the license control feature</p>
CodeCenterApplicationName	<p>The existing Black Duck Code Center application name</p>
CodeCenterApplicationVersion	<p>The existing Black Duck Code Center application version</p>
LicenseViolationRecipients	<p>Specifies addresses of recipients that should receive license alerts by email</p>
ScopesForLicenseAnalysis	<p>Lets you specify the scopes on which license analysis should be run.</p>
IncludePublishedArtifacts	<p>License checks are usually only run on the dependencies of the published package. If this flag is set, a license check is run on the deployed artifact itself (only valid for NuGet packages)</p>

AutoCreateMissingComponent	Auto create missing components in Black Duck Code Center application after the build is completed and deployed in Artifactory.
AutoDiscardStaleComponent	Auto discard stale components in Black Duck Code Center application after the build is completed and deployed in Artifactory.

Network Configuration

Deploying via Proxy

MSBuild Artifactory Plugin supports deployments via your network proxy. If the values in the **ProxySettings** tag of the *Artifactory.build* configuration file are not recognized by the plugin, it will fall back to using the **http_proxy** Environment Variables for proxy configuration using the format `http://<username>:<password>@proxy.com`.

You can bypass the proxy by setting the **Bypass** tag in the plugin configuration.

Resolution Proxy

Due to a [technical issue in the Nuget Client](#), you cannot configure the NuGet client for resolution via a proxy through the plugin. For the Nuget client to resolve artifacts via a proxy, you need to configure the proxy settings in `%APPDATA%\NuGet\NuGet.Config`.

For more information on how to configure a Nuget proxy, please refer to [NuGet Config Settings](#).

Running a build with MSBuild Artifactory Plugin

Once you have completed setting up the MSBuild Artifactory Plugin, you can run a project build. The plugin takes effect at the end of the build and does the following:

1. Publishes the specified published artifacts to the selected target repository and applies the proper path mappings.
2. Deploys the BuildInfo to Artifactory, providing [full traceability of the build](#), with links back to the build in TFS.

The example below shows Visual Studio output of a build log (minimum **verbosity** log level: Normal) with some deployed artifacts. At the bottom there is a link to the Build Info report on Artifactory.

```
Output
Show output from: Build
3> Found packages.config. Using packages listed as dependencies
3> Successfully created package 'bin\ArtifactoryDemo.Web.1.0.0.0.symbols.nupkg'.
3>[Artifactory]:
3> [Artifactory] Artifactory Post-Build task started
3> [Artifactory] Processing build info...
3> [Artifactory] Processing build modules...
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/nuget/ArtifactoryDemo.Web.1.0.0.0.nupkg
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/nuget/ArtifactoryDemo.Web.1.0.0.0.symbols.nupkg
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/ArtifactoryDemo.DAL.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/ArtifactoryDemo.Web.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/ArtifactoryDemo.Web.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/AWSSDK.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/Castle.Windsor.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/EntityFramework.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/EntityFramework.SqlServer.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/Lewtonsoft.Json.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/System.Net.Http.Formatting.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/System.Web.Http.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/System.Web.Mvc.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/System.Web.Optimization.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/System.Web.Razor.dll
3> [Artifactory] Deploying artifact: http://192.168.56.1:8080/artifactory/nuget-local/general/System.Web.WebPages.Razor.dll
3> [Artifactory] Uploading build info to Artifactory...
3> [Artifactory] Build successfully deployed. Browse it in Artifactory under http://192.168.56.1:8080/artifactory/webapp/builds/ArtifactoryDemo.Web/1411980146/2014-09-29T11:42:20.20+03:00/
3>
3>Build succeeded.
3>
3>Time Elapsed 00:00:07.86
===== Rebuild All: 3 succeeded, 0 failed, 0 skipped ======
```

Team Foundation Server (TFS) Integration

MSBuild Artifactory Plugin brings CI Build Integration to TFS users allowing you to efficiently deploy your artifacts to Artifactory. In addition to the BuildInfo that the plugin already registers, all parameters associated with TFS are also recorded to facilitate fully traceable builds.

For more information about how build information is used in Artifactory, please refer to [BuildInfo](#).

MSBuild Arguments in TFS

The MSBuild Artifactory Plugin configuration file supports MSBuild **Reserved Properties** or **Environment Properties**, and the best practice is to define these properties in the TFS build configuration. This lets you protect sensitive information and run the same build with different properties. Below is an example of properties configured in the TFS build definition.

General
Trigger
Source Settings
Build Defaults
Process
Retention Policy

Team Foundation Build uses a build process template defined by a Windows Workflow (XAML) file. The behavior of this template can be customized by setting the build process parameters provided by the selected template.

Build process template:
Default Template Show details

Build process parameters:

2. Configurations	Any CPU Debug
3. Clean build	True
4. Output location	SingleFolder
5. Advanced	
MSBuild arguments	/p:ARTIFACTORY_USER="admin",ARTIFACTORY_PASSWORD="password"
MSBuild platform	Auto
Perform code analysis	AsConfigured
Post-build script arguments	
Post-build script path	
Pre-build script arguments	
Pre-build script path	
3. Test	1 set(s) of tests specified.
1. Automated tests	
2. Advanced	
4. Publish Symbols	
5. Advanced	

Package Restore with Team Foundation Build

For Team Foundation Build 2013 on-premises, the default Build Process Templates already implement the **NuGet Package Restore Workflow** without any special configuration.

To avoid NuGet Package Restore from outside the plugin, you need to remove it from the workflow.

License

The MSBuild Artifactory plugin is available under the [Apache v2 License](#).

Changelog

▼ Click here to expand...

2.1.0 (18 Aug 2015)

1. Artifactory plugin now supports Visual Studio 2015
2. Bug fix [MSBAI-7](#)

2.0.0 (30 Apr 2015)

1. Artifactory plugin as a **Visual Studio Extension**, and used as a "Project template" in the solution.
2. Supporting [Black Duck](#) Code Center as part of the build process.

1.0.0 (30 Sep 2014)

1. First release version.

Docker Repositories

Artifactory as a Docker Registry

Artifactory lets you use repositories as fully-fledged [Docker](#) registries in every way.

On top of Artifactory's existing support for advanced artifact management, Artifactory support for Docker provides:

1. Distribution and sharing of Docker images within your organization
2. Secure Docker push and pull using [local repositories](#) as secure, private Docker registries
3. Extensive security features that give you fine-grained access control over registries and images
4. Reliable and consistent access to Docker images using [remote repositories](#)
5. Smart search for images
6. Support for the relevant calls of the [Docker Registry API](#) and [Docker Hub API](#) so that you can transparently use the Docker client to access images through Artifactory
7. Integration with Bintray for automated distribution of Docker images for both Docker V1 and Docker V2 repositories

Running Artifactory as a Docker container

Artifactory is also available on Bintray as a Docker image and can be run as a container. For more details, please refer to [Running with Docker](#).

Artifactory for Docker Users

New to Artifactory?

If you are new to Artifactory, we strongly recommend reading this section that provides some basic information about using Artifactory with Docker.

Artifactory serves all your needs when using Docker and manages all images used in your organization, whether created internally or downloaded from remote Docker registries.

Docker V1 and V2

Supporting both V1 and V2 of the Docker Registry API, Artifactory works transparently with the Docker client and lets you define as many Docker repositories as you wish, each functioning as an individual Docker registry. This enables you to manage each project in a distinct registry and exercise better access control to your Docker images.

Secure private docker Registry with local repositories

[Local Docker repositories](#) are where you store internal Docker images for distribution across your organization. Thanks to the internal [security features](#) built-in to Artifactory, a local repository is effectively a secure private Docker registry.

Consistent and reliable access to remote images

Artifactory also lets you define [remote Docker repositories](#) that reduce networking and provide consistent and reliable access to remote images. This is done by proxying external resources such as Docker Hub, or a remote Docker repository in another instance of Artifactory, and caching images that are downloaded from them.

Registries and Repositories

Both Artifactory and Docker use the term "repository", but each uses it in a different way.

In Docker, a **registry** contains any number of **repositories**. Each repository contains the different **tags** of a given image. For example, Docker Hub is a registry that contains many repositories. One of those is the Ubuntu repository, which contains several images, one for each tag released.

In Artifactory, you can define any number of **Docker repositories**. Each repository represents a whole

Docker registry, and can contain any number of images. For example, you can define a [local Docker repository](#) to manage all the images you create internally in your organization. You can also define a [remote Docker repository](#) to be a caching proxy for Docker Hub. This means that through a single remote repository in Artifactory, you gain access to all the images stored within Docker Hub.

Using Docker V1?

This page shows how to use Artifactory with the Docker V2 Registry API. If you are using the Docker V1 Registry API, please refer to [Using Docker V1](#).

Page Contents

- Artifactory as a Docker Registry
- Artifactory for Docker Users
 - Registries and Repositories
- Getting Started with Artifactory and Docker
 - 1. Setting up NGINX as a Reverse Proxy
 - 2. Creating a Local Docker Repository
 - Working with Artifactory Online
 - 3. Setting Up Authentication
 - Anonymously Access Disabled
 - Anonymously Access Enabled
 - 4. Pushing and Pulling Images
 - Docker Push
 - Docker Pull
 - Set Me Up
 - Watch the Screencast
- Proxying a Remote Docker Registry
- Virtual Docker Repositories
- Browsing Docker Repositories
 - Docker Info
- Searching for Docker Images
- Promoting Docker

- Images
- Advanced Topics
 - Using a Self-signed SSL Certificate
 - Alternative Proxy Servers
 - Apache Configuration
 - Port Bindings
 - Multiple Docker Repositories with Artifactory
 - Docker Repository Path and Domain
 - Setting Your Credentials Manually
- Migrating from Docker V1 to Docker V2
- Support Matrix

Read More

- [Using Docker V1](#)

Getting Started with Artifactory and Docker

Artifactory supports Docker transparently, meaning you can point the Docker client at Artifactory and issue push, pull and other commands in exactly the same way that you are used to when working directly with a private registry or Docker Hub.

To get started using Docker with Artifactory you need to execute the following steps:

1. Set up a web server as a reverse proxy (you can skip this step if you are working with Artifactory Online)
2. Create a local repository
3. Set up authentication
4. Push and pull images

The [screencast](#) at the end of this section provides a demonstration.

1. Setting up NGINX as a Reverse Proxy

Artifactory can only be used with Docker through a reverse proxy due to the following limitations of the Docker client:

1. You cannot provide a context path when providing the registry path (e.g `localhost:8080/artifactory` is not valid)
2. Docker will only send basic HTTP authentication when working against an HTTPS host

Working with Artifactory Online

If you are working with Artifactory Online there is no need to set up a reverse proxy so you can skip this step

For Artifactory to work with Docker, the preferred web server is **NGINX v1.3.9** and above configured as a reverse proxy.

For other supported web servers, please refer to [Alternative Proxy Servers](#).

Below is a sample configuration for NGINX which configures SSL on port 443 to a specific local repository in Artifactory (named `docker-local`) on a server called `artprod.company.com`.

▼ NGINX configuration

This code requires NGINX to support chunked transfer encoding which is available from NGINX v1.3.9.

```

[...]
http {

## 
# Basic Settings
##
[...]

server {
    listen 443;
    server_name artprod.company.com;

    ssl on;
    ssl_certificate /etc/ssl/certs/artprod.company.com.crt;
    ssl_certificate_key /etc/ssl/private/artprod.company.com.key;

    access_log /var/log/nginx/artprod.company.com.access.log;
    error_log /var/log/nginx/artprod.company.com.error.log;

    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Original-URI $request_uri;
    proxy_pass_header Server; # To help debugging, list the server that actually
did the reply rather than nginx
    proxy_read_timeout 900;

    client_max_body_size 0; # disable any limits to avoid HTTP 413 for large image
uploads

    # required to avoid HTTP 411: see Issue #1486
    (#https://github.com/docker/docker/issues/1486)
    chunked_transfer_encoding on;

    location /v2 {
        # Do not allow connections from docker 1.5 and earlier
        # docker pre-1.6.0 did not properly set the user agent on ping
        if ($http_user_agent ~ "^(docker\\v1\\.(3|4|5(?:\\.[0-9]-dev))).*$" ) {
            return 404;
        }

        proxy_pass
        http://artprod.company.com:8080/artifactory/api/docker/docker-local/v2;
    }
}
}

```

Multiple Docker repositories and port bindings

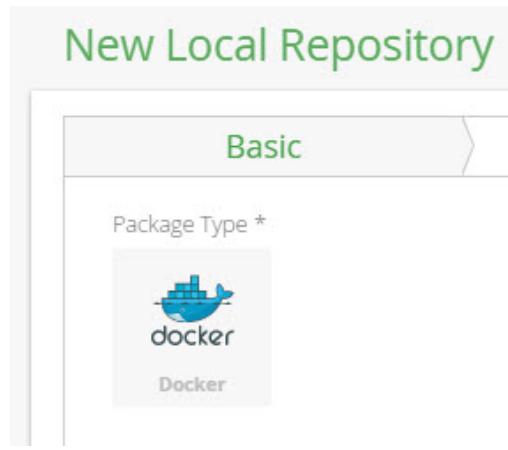
If you want to use multiple Docker repositories, you need to copy this configuration and bind different ports or `server_names` to each local repository in Artifactory. For details, please refer to [Port Bindings](#).

Repository URL prefix

When accessing a Docker repository through Artifactory, the repository URL must be prefixed with `api/docker` in the path. For details, please refer to [Docker repository path and domain](#).

2. Creating a Local Docker Repository

To create a local Docker repository to host your Docker images, create a new Local Repository and set **Docker** as the **Package Type**.



In the Docker Settings section, select the Docker API version this repository should work with.

Docker Settings

API Version

V1

V2

Force Authentication (?)

Working with Artifactory Online

▼ Click to expand...

When working with Artifactory Online, there is a special pattern to follow when creating Docker v2 repositories.

First, you need to create a new Docker V2 repository by the name `dockerv2-{local/remote/virtual}`.

For example, in case of Docker V2 remote repository the name should be `dockerv2-remote`. For Docker V2 virtual repository, the name should be `dockerv2-virtual`.

Then, use the following address when working with the Docker client: `${account_name}-docker-dockerv2-{local/remote/virtual}.artifactoryonline.com`

For example, if your Artifactory Online account is called "acme" and you configured a Docker V2 local repository, then you would use the following address: `$acme-docker-dockerv2-local.artifactoryonline.com`

3. Setting Up Authentication

Artifactory may be configured to [allow anonymous access](#). This lets you provide access to users who have not logged in (according to permissions you define for anonymous access). While this is a global setting in Artifactory, you can override it specifically for Docker repositories.

Anonymous Access Disabled

With anonymous access disabled, Artifactory will require authentication and then you can log in to Docker in the usual way using `docker login`.

Anonymous Access Enabled

With anonymous access enabled, Artifactory will allow access to your repositories in general, however, you may still require the Docker client to

be authenticated (especially for the `docker login` command). To do so, ensure that **Force Authentication** is enabled in your Docker Repository configuration, and then log in to Docker in the usual way using `docker login`.

Docker Settings

API Version

V1

V2

Force Authentication [\(?\)](#)

Unable to login, or using Artifactory v4.0.0

Force Authentication is supported from Artifactory v4.0.1.

If you are unable to login to Docker, or you are using Artifactory 4.0.0, please refer to [Setting Your Credentials Manually](#).

4. Pushing and Pulling Images

Docker `push` and `pull` commands are in the form of:

```
docker push/pull [registry_hostname[:port]/][user_name/](repository_name:version_tag)
```

Docker Push

In order to use `docker push` you need to tag your image with the proper reverse proxy URL described above under [Configuring a Reverse Proxy](#).

For example, to push the [official Ubuntu base image](#) into our local Artifactory repository:

```
# First we need to pull the official Ubuntu base image
$ docker pull ubuntu

# Next we tag it with our own endpoint URL
$ docker tag ubuntu artprod.company.com/ubuntu

# Finally we can push the tagged image
$ docker push artprod.company.com/ubuntu
```

Docker Pull

You can simply execute `docker pull` from the endpoint URL as defined above:

```
$ docker pull artprod.company.com/ubuntu
```

Set Me Up

To get the corresponding `docker push` and `docker pull` commands for any Docker repository, select it in the Tree Browser and click **Set Me Up**.

Set Me Up

X

Repository

docker-local

General

Using Docker with Artifactory requires a reverse proxy such as Nginx or Apache. For more details please visit our [documentation](#).

Deploy

Execute "docker push" from the endpoint URL.

For example, to push to a server named **artprod.company.com** that is behind Nginx or Apache:

```
1 docker tag ubuntu artprod.company.com/<DOCKER_REPOSITORY>:<DOCKER_TAG>
```

```
1 docker push artprod.company.com/<DOCKER_REPOSITORY>:<DOCKER_TAG>
```

Resolve

Execute docker pull from the endpoint URL.

For example, to pull from server named **artprod.company.com** that is behind Nginx or Apache:

```
1 docker pull artprod.company.com/<DOCKER_REPOSITORY>:<DOCKER_TAG>
```

Watch the Screencast

Once you have completed the above setup you should be able use the Docker client to transparently push images to and pull them from Docker repositories in Artifactory. You can see this in action in the screencast below.

Proxying a Remote Docker Registry

You can proxy a remote Docker registry through remote repositories. A [Remote Repository](#) defined in Artifactory serves as a caching proxy for a registry managed at a remote URL such as <https://registry-1.docker.io/> (which is the Docker Hub), or even a Docker repository managed at a remote site by another instance of Artifactory .

Docker images requested from a remote repository are cached on demand. You can remove downloaded images from the remote repository cache, however, you can not manually push Docker images to a remote Docker repository.

To define a remote repository to proxy a remote Docker registry follow the steps below:

1. Create a new Remote Repository and set **Docker** as the **Package Type**.
2. Set the **Repository Key** value, and specify the URL to the remote registry in the **URL** field

New Remote Repository

Basic Advanced Replications

Package Type *

Docker

Repository Key *

docker-remote

URL *

https://registry-1.docker.io/

Test

General

Repository Layout

Remote Layout Mapping

Docker Settings

Enable Token Authentication

Proxying the official Docker Hub
If you are proxying the official Docker Hub, use <https://registry-1.docker.io/> as the URL, and make sure to check the **Enable Token Authentication** checkbox as well since the Docker Hub only supports token-based authentication.

- Once you have configured your remote Docker repository, you can test your configuration by using the Docker client to pull an image from Docker Hub (or whichever remote resource your repository proxies) through Artifactory.

Pull replication is not supported for remote Docker repositories

Due to a limitation in the Docker client, you cannot configure pull replication from an Artifactory remote Docker repository.

Virtual Docker Repositories

From version 4.1, Artifactory supports virtual Docker Repositories. A **Virtual Repository** defined in Artifactory aggregates packages from both local and remote repositories.

This allows you to access images that are hosted locally on local Docker repositories, as well as remote images that are proxied by remote Docker repositories, and access all of them from a single URL defined for the virtual repository.

To create a virtual Docker repository set **Docker** to be its **Package Type**, and select the underlying local and remote Docker repositories to include under the **Repositories** section.

Repositories

Available Repositories

Selected Repositories

<<
<
>
>>

Included Repositories

docker-local

docker-remote

Browsing Docker Repositories

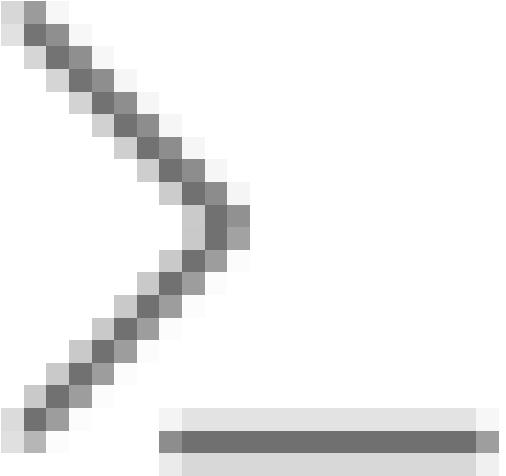
For general information on how to browse repositories, please refer to [Browsing Artifactory](#).

Docker Info

The **Docker Info** tab maps the entire set of commands used to generate the selected Docker image. Essentially, you would see the same series of commands using `docker history`.

The **Tag Info** section displays details for the selected tag. In addition, you can select any layer of the image in the **Layers** section to view the following properties:

Symbol	Property
</>	The layer ID
	The layer digest
	The layer size

	The timestamp when the layer was created
	The command that created the layer

	d7057cb02084
latest	
cfa753dfea5e	

Docker Info

Tag Info
Title: busybox:latest
Digest: 000409ca75cd0b754155d790402405fdc35f051af1917ae35a9f4d96ec06ae50
Total Size: 1.1 MB

Layers

	d7057cb02084	cfa753dfea5e	

	d7057cb02084
</>	d7057cb02084f245031d27b76cb18af05db1cc3a96a29fa7777af755ac91a3
	sha256:a3ed95cae02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
	0 B
	21-09-15 23:15:47 IDT
	CMD ["sh"]

Searching for Docker Images

You can search for Docker images by using Artifactory's [Property Search](#) to search for the **docker.manifest** property through the UI or through the [REST API](#). To narrow down your search, you can also search on the **docker.repoName** property.

Search by:	Quick	Archive	GAVC	Property	Checksum	Remote
Add: Property Property Set						Back To Browse
Name *	<input type="text"/>	Value *	<input type="text"/>	<input type="button" value="Clear"/>	<input type="button" value="Add"/>	
① Search is case sensitive & supports *, ?						
Property	Value(s)					
<input type="text" value="docker.manifest"/>	<input type="button" value="Search"/>					
Limit to Specific Repositories						
Search Results (1 Results) put results in stash <Stash is empty>						
Filter by Item						
<input checked="" type="radio"/> Item	Type	Path		Repository	Modified	
<input type="radio"/> manifest.json		ubuntu/latest		docker-local	26-07-15 13:02:00 UTC	

Promoting Docker Images

Artifactory supports promoting Docker images from one Docker repositories in Artifactory to another.

Promotion can be triggered using the following endpoint with cURL:

```

POST api/docker/<repoKey>/v2/promote
{
    "targetRepo" : "<targetRepo>",
    "dockerRepository" : "<dockerRepository>",
    "tag" : "<tag>",
    "copy": <copy>
}

```

where:

<repoKey>	Source repository key (For example, <i>docker-local</i> as used in this page)
<targetRepo>	The target repository to move or copy
<dockerRepository>	The docker repository name to promote
<tag>	An optional tag name to promote, if null - the entire docker repository will be promoted. Default: ""
<copy>	Whether to copy instead of move. Default: false

This is useful when you need to promote Docker images, for example, from a development repository to production.

An example for promoting the docker image "*jfrog/ubuntu*" with all of its tags from *docker-local* to *docker-prod* using cURL would be:

Promoting Docker Images

```

curl -i -uadmin:password -X POST "https://artprod.company.com/v2/promote" -H
"Content-Type: application/json" -d
'{"targetRepo":"docker-prod","dockerRepository":"jfrog/ubuntu"}'

```

Notice that the above example is executed through your reverse proxy. To go directly through Artifactory, you would execute this command as follows:

```

curl -i -uadmin:password -X POST
"http://localhost:8080/artifactory/api/docker/docker-local/v2/promote" -H
"Content-Type: application/json" -d
'{"targetRepo":"docker-prod","dockerRepository":"jfrog/ubuntu"}'

```

Advanced Topics

Using a Self-signed SSL Certificate

You can use self-signed SSL certificates with `docker push/pull` commands, however for this to work, you need to specify the `--insecure-`

registry daemon flag for each insecure registry.

For full details please refer to the Docker documentation.

For example, if you are running Docker as a service, edit the `/etc/default/docker` file, and append the `--insecure-registry` flag with your registry URL to the `DOCKER_OPTS` variable as in the following example:

Edit the DOCKER_OPTS variable

```
DOCKER_OPTS="--H unix:///var/run/docker.sock --insecure-registry artprod.company.com"
```

For this to take effect, you need to restart the Docker service.

If you are using **Boot2Docker**, please refer to the **Boot2Docker** documentation for **Insecure Registry**.

If you do not make the required modifications to the `--insecure-registry` daemon flag, you should get the following error:

Error message

```
v2 ping attempt failed with error: Get https://artprod.company.com/v2/: x509: cannot validate certificate for artprod.company.com because it doesn't contain any IP SANs
```

Alternative Proxy Servers

In addition to NGINX, you can setup Artifactory to work with Docker using Apache.

Apache Configuration

The sample configuration below configures SSL on port 443 and a server name of `artprod.company.com`.

Apache Configuration

```
<VirtualHost *:443>
    ServerName artprod2.company.com

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    SSLEngine on
    SSLCertificateFile/etc/ssl/certs/artprod.company.com.pem
    SSLCertificateKeyFile /etc/ssl/private/artprod.company.com.key

    ProxyRequests off
    ProxyPreserveHost on

    ProxyPass        /
    http://artprod.company.com:8080/artifactory/api/docker/docker-local/
    ProxyPassReverse /
    http://artprod.company.com:8080/artifactory/api/docker/docker-local/
</VirtualHost>
```

Port Bindings

If you want to use multiple repositories, you need to copy the NGINX configuration and bind different ports to each local repository in Artifactory.

When binding a port other than 443, note that the configuration for the proxy header must be appended with the port number on the `proxy_set_header` line. For example, for a server running on port 444 you should write `proxy_set_header Host $host:444`.

Multiple Docker Repositories with Artifactory

A limitation of Docker is that it can only use the fully qualified domain name (FQDN) and port to distinguish repositories. Therefore, if you want to have several Docker repositories in an artifactory installation you need to map artifactory URLs according to Docker requirements.

There are a couple of ways to do this:

- **Distinguishing repositories by port number**

This the easiest way to distinguish repositories since you don't have to make any changes to DNS. Examples on this page usually use this method so they can be self contained. However, the problem with this method is users need to remember which port is assigned to which repository. For example, if you specify a repository as something like `http://example.com:42/megadodo/hhgttg`, your users need to remember which repository is assigned to port 42.

- **Using the FQDN**

If you use the FQDN, you can refer to a repository using its name. For example, you could use something like `http://publishinghouses.example.com/megadodo/hhgttg` to refer to the publishinghouses repository in Artifactory. This makes it obvious which repository your docker pull command is coming from. To implement this solution, you first need to appropriate a wildcard DNS entry for your Docker registries. For example, you could acquire the domain "`*.docker.mycompany.com`", and have that point to your Artifactory HA cluster's load balancer which will redirect to the collection of webservers (NGINX or Apache) used for your Artifactory HA instances. Then, you need to configure the webserver at each HA node with one of the configurations below:

▼ NGINX configuration for Artifactory HA node...

```
upstream artifactory {
    ip_hash;

    # if nginx+ use instead
    # sticky cookie artifactory domain=.jfrog.info path=/artifactory;

    server localhost:8081;
    #repeat previous line for each server
}

server {
    listen 443;
    server_name _;

    client_max_body_size 0; # disable any limits to avoid HTTP 413
    chunked_transfer_encoding on; # to avoid HTTP 411

    rewrite ^/$ http://$host/artifactory/;

    proxy_read_timeout 900;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Port $server_port;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_pass_header Server; # return the server header of the server that
    actually serviced the request

    location /artifactory {
        proxy_pass http://artifactory/artifactory;
    }

}

# Standard https
server {
    listen 443 ssl;
    server_name _;
```

```
ssl_certificate      /etc/nginx/ssl/demo.pem;
ssl_certificate_key /etc/nginx/ssl/demo.key;

ssl_session_cache shared:SSL:1m;
ssl_session_timeout 5m;

ssl_ciphers  HIGH:!aNULL:!MD5;
ssl_prefer_server_ciphers  on;

client_max_body_size 0; # disable any limits to avoid HTTP 413
chunked_transfer_encoding on; # to avoid HTTP 411

rewrite ^/$ $scheme://$host/artifactory/;

proxy_read_timeout 900;
proxy_set_header Host $http_host;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Port $server_port;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_pass_header Server;

proxy_set_header X-Forwarded-Ssl on;

location /artifactory {
    proxy_pass          http://artifactory/artifactory;
}

location /v2 {
    proxy_pass          http://artifactory/artifactory/api/docker/docker-virtual/v2
}
}

server {
    listen 443 ssl;
    server_name dockerremote;

    ssl_certificate      /etc/nginx/ssl/demo.pem;
    ssl_certificate_key /etc/nginx/ssl/demo.key;

    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout 5m;

    ssl_ciphers  HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers  on;

    client_max_body_size 0; # disable any limits to avoid HTTP 413
    chunked_transfer_encoding on; # to avoid HTTP 411

    proxy_read_timeout 900;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Port $server_port;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_pass_header Server;

    proxy_set_header X-Forwarded-Ssl on;
```

```
location /v2/ {
    proxy_pass
http://artifactory/artifactory/api/docker/docker-remote/v2/;
}
}

server {
    listen 443 ssl;
    server_name dockerremote2;

    ssl_certificate      /etc/nginx/ssl/demo.pem;
    ssl_certificate_key  /etc/nginx/ssl/demo.key;

    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  5m;

    ssl_ciphers  HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers  on;

    client_max_body_size 0; # disable any limits to avoid HTTP 413
    chunked_transfer_encoding on; # to avoid HTTP 411

    proxy_read_timeout 900;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Port $server_port;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_pass_header Server;

    proxy_set_header X-Forwarded-Ssl on;

location /v1 {
    proxy_pass
http://artifactory/artifactory/api/docker/docker-prod-local/v1;
}

location /v2/ {
    proxy_pass
http://artifactory/artifactory/api/docker/docker-prod-local2/v2/;
}
}

server {
    listen 443 ssl;
    server_name dockerremote3;

    ssl_certificate      /etc/nginx/ssl/demo.pem;
    ssl_certificate_key  /etc/nginx/ssl/demo.key;

    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  5m;

    ssl_ciphers  HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers  on;

    client_max_body_size 0; # disable any limits to avoid HTTP 413
    chunked_transfer_encoding on; # to avoid HTTP 411
```

```
proxy_read_timeout 900;
proxy_set_header Host $http_host;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Port $server_port;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_pass_header Server;

proxy_set_header X-Forwarded-Ssl on;

location /v1 {
    proxy_pass
http://artifactory/artifactory/api/docker/docker-dev-local/v1;
}

location /v2/ {
    proxy_pass
http://artifactory/artifactory/api/docker/docker-dev-local2/v2/;
}
}
```

Apache configuration for Artifactory HA node...

```
<Proxy balancer://tomcats>
# Artifactory server #1
BalancerMember http://nfs_artifactory_1:8081 route=art1

# Artifactory server #2
BalancerMember http://nfs_artifactory_1:8081 route=art2

ProxySet lbmethod=byrequests
ProxySet stickysession=ROUTEID
</Proxy>

RewriteEngine On
RewriteRule ^/$ /artifactory [R,L]

LogLevel warn
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

ProxyRequests off
ProxyPreserveHost on
Header add Set-Cookie "ROUTEID=.${BALANCER_WORKER_ROUTE}; path=/artifactory/"
env=BALANCER_ROUTE_CHANGED

<VirtualHost *:80>
    ProxyPass / balancer://tomcats/
</VirtualHost>

<VirtualHost *:443>
ServerName dockerone.com
SSLEngine on
SSLCertificateFile      /etc/ssl/certs/demo.pem
SSLCertificateKeyFile   /etc/ssl/private/demo.key
SSLProxyEngine on

ProxyPreserveHost on

ProxyPass          /v1
balancer://tomcats/artifactory/api/docker/docker-prod-local/v1
ProxyPassReverse  /v1
balancer://tomcats/artifactory/api/docker/docker-prod-local/v1

ProxyPass          /v2
balancer://tomcats/artifactory/api/docker/docker-prod-local2/v2
ProxyPassReverse  /v2
balancer://tomcats/artifactory/api/docker/docker-prod-local2/v2

ProxyPass / balancer://tomcats/

</VirtualHost>
```

```
<VirtualHost *:443>
ServerName dockertwo.com
SSLEngine on
SSLCertificateFile      /etc/ssl/certs/demo.pem
SSLCertificateKeyFile   /etc/ssl/private/demo.key
SSLProxyEngine on

ProxyPreserveHost on

ProxyPass          /v1
balancer://tomcats/artifactory/api/docker/docker-dev-local/v1
ProxyPassReverse  /v1
balancer://tomcats/artifactory/api/docker/docker-dev-local/v1

ProxyPass          /v2
balancer://tomcats/artifactory/api/docker/docker-dev-local2/v2
ProxyPassReverse  /v2
balancer://tomcats/artifactory/api/docker/docker-dev-local2/v2
```

```
ProxyPass / balancer://tomcats/  
</VirtualHost>
```

Docker Repository Path and Domain

When accessing a Docker repository through Artifactory, the repository URL must be prefixed with **api/docker** in the path.

You can copy the full URL from the UI using **Set Me Up** when the repository is selected in the Tree Browser.

For example, if you are using Artifactory standalone or as a local service, you would access your Docker repositories using the following URL:

`http://localhost:8081/artifactory/api/docker/<repository key>`

Also, the domain of your Docker repository must be expressed as an explicit IP address. The only exception is when working locally, you can use the *localhost* domain name as the proxy pass.

Setting Your Credentials Manually

If you are unable to log in to Docker, you may need to set your credentials manually.

Manually setting your Docker credentials

The Docker command line tool supports authenticating sensitive operations, such as push, with the server using basic HTTP authentication.

To enforce authenticated access to docker repositories you need to provide the following parameters to the Docker configuration file.

- The Docker endpoint URL (must use HTTPS for basic authentication to work)
- Your Artifactory username and password (formatted `username:password`) as Base64 encoded strings
- Your email address

You can use the following command to get these strings directly from Artifactory and copy/paste them into your `~/.dockercfg` file:

sudo

If you are using Docker commands with "sudo" or as a root user (for example after installing the Docker client), note that the Docker configuration file should be placed under `/root/.dockercfg`

Getting .dockercfg entries directly from Artifactory

```
$ curl -uadmin:password  
"https://artprod.company.com/<v1|v2>/auth"  
{  
"https://artprod.company.com" : {  
"auth" : "YWRtaW46QVA1N05OaHZTMnM5Qk02RkR5RjNBVmF4TVFl",  
"email" : "admin@email.com"  
}  
}
```

The Docker configuration file may contain a separate authentication block for each registry that you wish to access.

Below is an example with two URL endpoints:

```
{
  "https://artprod.company.com": {
    "auth": "YWRtaW46cGFzc3dvcmQ=",
    "email": "myemail@email.com"
  },
  "https://artprod2.company.com": {
    "auth": "YWRtaW46cGFzc3dvcmQ=",
    "email": "myemail@email.com"
  }
}
```

Migrating from Docker V1 to Docker V2

If you are still using Docker V1, we strongly recommend upgrading to Docker V2. This requires that you migrate any Docker repositories that were created for Docker V1, and is done with a simple cURL endpoint.

For details, please refer to [Migrating a V1 repository to V2](#) under the [Using Docker V1](#) documentation.

Support Matrix

This matrix provides information on features supported as the versions of Artifactory progress.

<i>Artifactory Version</i>	<i>Docker Client Version</i>	<i>Docker V1 API</i>	<i>Docker V2 API</i>	Remote Repositories*	Virtual Repositories*
4.1+	1.8	✓	✓	✓	✓
4.0.2+	1.8	✓	✓	✓	
4.0.0+	1.6+	✓	✓	✓	
4.0.0+	<1.6	✓			

* Supported for Docker V2 API only

Using Docker V1

Overview

This page describes how to use Artifactory with the Docker V1 Registry API. If you are using the Docker V2 Registry API, please refer to [Docker Repositories](#).

For general information on using Artifactory with Docker, please refer to [Artifactory as a Docker Registry](#).

Getting Started with Artifactory and Docker

Artifactory supports Docker transparently, meaning you can point the Docker client at Artifactory and issue push, pull and other commands in exactly the same way that you are used to when working directly with a private registry or Docker Hub.

To get started using Docker with Artifactory you need to execute the following steps:

1. Set up a web server as a reverse proxy
2. Create a local repository
3. Set up authentication
4. Push and pull images

The [screencast](#) at the end of this section provides a demonstration.

1. Setting up NGINX as a Reverse Proxy

Artifactory can only be used with Docker through a reverse proxy due to the following limitations of the Docker client:

1. You cannot provide a context path when providing the registry path (e.g `localhost:8080/artifactory` is not valid)
2. Docker will only send basic HTTP authentication when working against an HTTPS host

For Artifactory to work with Docker, the preferred web server is **NGINX v1.3.9** and above configured as a reverse proxy.

For other supported web servers, please refer to [Alternative Proxy Servers](#).

Below is a sample configuration for NGINX which configures SSL on port 443 to a specific local repository in Artifactory (named `docker-local`) on a server called `artprod.company.com`.

▼ **NGINX Configuration for Docker V1**

This code requires NGINX to support chunked transfer encoding which is available from NGINX v1.3.9.

```

[...]

http {

    ## 
    # Basic Settings
    ##
    [...]

    server {
        listen 443;
        server_name artprod.company.com;

        ssl on;
        ssl_certificate
        /etc/ssl/certs/artprod.company.com.crt;
        ssl_certificate_key
        /etc/ssl/private/artprod.company.com.key;

        access_log
        /var/log/nginx/artprod.company.com.access.log;
        error_log
        /var/log/nginx/artprod.company.com.error.log;

        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For
        $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Original-URI $request_uri;
        proxy_read_timeout 900;

        client_max_body_size 0; # disable any limits to
        avoid HTTP 413 for large image uploads

        # required to avoid HTTP 411: see Issue #1486
        (#https://github.com/docker/docker/issues/1486)
        chunked_transfer_encoding on;

        location /v1 {
            proxy_pass
            http://artprod.company.com:8080/artifactory/api/docker/docker-local/v1;
        }
    }
}

```

Multiple Docker repositories and port bindings

If you want to use multiple Docker repositories, you need to copy this configuration and bind different ports to each local repository in Artifactory. For details, please refer to [Port Bindings](#).

Repository URL prefix

When accessing a Docker repository through Artifactory, the repository URL must be prefixed with **api/docker** in the path. For details, please refer to [Docker Repository Path and Domain](#).

2. Creating a Local Docker Repository

This is done in the same way as when [configuring a local repository](#) to work with Docker V2, however, in the Docker Settings section, you should make sure to select V1 as the Docker API version.

Docker Settings

API Version

V1

V2

Force Authentication [?](#)

Page Contents

- Overview
- Getting Started with Artifactory and Docker
 - 1. Setting up NGINX as a Reverse Proxy
 - 2. Creating a Local Docker Repository
 - Working with Artifactory Online
 - 3. Setting Up Authentication
 - 4. Pushing and Pulling Images
 - Watch the Screencast
 - Browsing Docker Repositories
 - Viewing the Docker Images Tree
 - Viewing Individual Docker image Information
 - Searching for Docker Images
 - Promoting Docker Images with V1
 - Migrating a V1 repository to V2
 - Deletion and Cleanup
 - Advanced Topics
 - Using a Self-signed SSL Certificate
 - Alternative Proxy Servers
 - Apache Configuration
 - Port Bindings
 - Docker Repository Path and Domain
 - Support Matrix

Working with Artifactory Online

▼ Click here to expand...

Due to limitations of the Docker client, in Artifactory Online there is a special configuration for each server with a sub-domain.

You need to create a new Docker enabled local repository named `docker-local`.

Then, use the following address when working with the Docker client: `"${account_name}.artifactoryonline.com"`

3. Setting Up Authentication

When using Artifactory with Docker V1, you need to set your credentials manually as described in the Docker V2 documentation under [Setting Your Credentials Manually](#).

4. Pushing and Pulling Images

Pushing and pulling images when using Docker V1 is done in the same way as when using Docker V2. Please refer to [Pushing and Pulling Images](#) under the Docker Repositories page.

Watch the Screencast

Once you have completed the above setup you should be able to use the Docker client to transparently push images to and pull them from Docker repositories in Artifactory. You can see this in action in the screencast below.

Browsing Docker Repositories

Artifactory stores docker images in a layout that is made up of 2 main directories:

- **.images**: Stores all the flat docker images.
- **repositories**: Stores all the repository information with tags (similar to how repositories are stored in the Docker Hub).

In addition, Artifactory annotates each deployed docker image with two properties:

- **docker.imageId**: The image id
- **docker.size**: The size of the image in bits

Deployed tags are also annotated with two properties:

- **docker.tag.name**: The tag name
- **docker.tag.content**: The id of the image that this tag points to

The screenshot shows the Artifactory Artifact Repository Browser interface. On the left, there is a tree view of local repositories: bower-local, debian-local, docker-local, and dockerv1. Under dockerv1, there is a .images directory containing images 42, 43, ed, and gf. The gf folder is expanded, showing its contents: _checksum.json, ancestry.json, json.json, and layer.tar. A subfolder 9fd3c8c9af32 is selected. The main panel displays the properties for this image. The 'Properties' tab is active, showing a table with two rows: docker.size (1895) and docker.imageId (9fd3c8c9af32ddd1793ccb5f6535e12d735eacae16f8f8c4214f42f33fe3d29). There are tabs for General, Docker Info, Docker Ancestry, Effective Permissions, and Watchers.

The screenshot shows the Artifactory Artifact Repository Browser interface. On the left, there is a tree view of local repositories: bower-local, debian-local, docker-local, and dockerv1. Under dockerv1, there is a .images directory containing repositories. One of these repositories has a library folder, which contains an ubuntu folder. Inside the ubuntu folder, there is a latest folder containing a tag.json file. This tag.json file is selected. The main panel displays the properties for this tag. The 'Properties' tab is active, showing a table with two rows: docker.tag.name (latest) and docker.tag.content (6d4946999d4fb403f40e151ecbd13cb866da125431eb1df0cdfd4dc72674e3c6). There are tabs for General, Effective Permissions, Watchers, Builds, and Governance.

Viewing the Docker Images Tree

Artifactory lets you view the complete images tree for a specific image directly from the UI in a similar way to what you would get from the `docker images --tree` command.

In the **Artifacts** module **Tree Browser**, drill down to select the image you want to inspect. The metadata is displayed in the **Docker Ancestry** tab.

6d4946999d4f

General	Docker Info	Docker Ancestry	Effective Permissions	Properties	Watchers
<hr/>					
		__ 428b411c28f0 Virtual Size: 188.1 Mbit			
		__ 435050075b3f Virtual Size: 188.3 Mbit			
		__ 9fd3c8c9af32 Virtual Size: 188.3 Mbit			
		__ 6d4946999d4f Virtual Size: 188.3 Mbit			

Viewing Individual Docker image Information

In the **Artifacts** module **Tree Browser**, drill down to select image you want to inspect. The metadata is displayed in the **Docker Info** tab.

General	Docker Info	Docker Ancestry	Effective Permissions	Properties	Watchers	Actions
<hr/>						
Package Info						
Image Id:	6d4946999d4fb403f40e151ecbd13cb866da125431eb1df0cdfd4dc72674e3c6					
Parent Id:	9fd3c8c9af32dddb1793ccb5f6535e12d735eacae16f8c4214f42f33fe3d29					
Created:	2015-06-12T15:32:30.680894574Z					
Container:	9201d6220b01338011f2f21c28429d2155625625392e6654fe378c2772cc46bd					
Docker Version:	1.6.0					
Architecture:	amd64					
OS:	linux					
Size:	0 bits (0 bit)					
Config						
Hostname:	d3659c5e113e					
DomainName:						
User:						
Memory:	0					
MemorySwap:	0					
CpuShares:	0					
AttachStdin:	false					
AttachStdout:	false					
AttachStderr:	false					
Tty:	false					
OpenStdin:	false					
StdinOnce:	false					

Searching for Docker Images

In addition to other properties related to Docker repositories, you can also search for repositories using a property called `docker.repoName`, which represents the repository name (e.g., "library/ubuntu").

The screenshot shows a JFrog Xray interface for managing Docker images. The current view is for the file '_index_images.json'. The 'Properties' tab is active. A table displays a single property entry:

Property	Value(s)
docker.repoName	library/ubuntu

Promoting Docker Images with V1

Promoting Docker images with Docker V1 is done in exactly the same way as when [Promoting Images with Docker V2](#).

Migrating a V1 repository to V2

We recommend using Docker V2 repositories when possible (provided your Docker client is version 1.6 and above).

If you have an existing Docker V1 repository, you can migrate its content into a V2 repository using the following endpoint with cURL:

```
POST api/docker/<repoKey>/v1/migrate
{
    "targetRepo" : "<targetRepo>",
    "dockerRepository" : "<dockerRepository>",
    "tag" : "<tag>"
}
```

where:

<repoKey>	Source repository key (For example, <i>docker-local</i> as used in this page)
<targetRepo>	The target Docker V2 repository to migrate to (For example, <i>docker-local2</i> as used in this page). The repository should be created before running the <code>migrate</code> endpoint.
<dockerRepository>	An optional docker repository name to migrate, if null - the entire source repository will be migrated. Default: ""
<tag>	An optional tag name to promote, if null - the entire docker repository will be promoted. Default: ""

An example for migrating the docker image "*jfrog/ubuntu*" with all of it's tags from *docker-local* to *docker-local2* using cURL would be:

```
curl -i -uadmin:password -X POST  
"http://localhost:8080/artifactory/api/docker/docker-local/v1/migrate" -H  
"Content-Type: application/json" -d  
'>{"tagetRepo":"docker-local2","dockerRepository":"jfrog/ubuntu"}'
```

Deletion and Cleanup

Artifactory natively supports removing tags and repositories and complies with the [Docker Hub Spec](#).

Deletion of Docker tags and repositories automatically cleans up any orphan layers that are left (layers not used by any other tag/repository).

Currently, the Docker client does not support DELETE commands, but deletion can be triggered manually using cURL. Here are some examples:

Removing repositories and tags

```
//Removing the "jfrog/ubuntu" repository  
curl -uadmin:password -X DELETE  
"https://artprod.company.com/v1/repositories/jfrog/ubuntu"  
  
//Removing the "12.04" tag from the "jfrog/ubuntu" repository  
curl -uadmin:password -X DELETE  
"https://artprod.company.com/v1/repositories/jfrog/ubuntu/tags/12.04"
```

Empty Directories

Any empty directories that are left following removal of a repository or tag will automatically be removed during the next folder pruning job (which occurs every 5 minutes by default).

Advanced Topics

Using a Self-signed SSL Certificate

From Docker version 1.3.1, you can use self-signed SSL certificates with `docker push/pull` commands, however for this to work, you need to specify the `--insecure-registry` daemon flag for each insecure registry.

For full details please refer to the [Docker documentation](#).

For example, if you are running Docker as a service, edit the `/etc/default/docker` file, and append the `--insecure-registry` flag with your registry URL to the `DOCKER_OPTS` variable as in the following example:

Edit the DOCKER_OPTS variable

```
DOCKER_OPTS="-H unix:///var/run/docker.sock --insecure-registry artprod.company.com"
```

For this to take effect, you need to restart the Docker service.

If you are using **Boot2Docker**, please refer to the [Boot2Docker](#) documentation for [Insecure Registry](#).

If you do not make the required modifications to the `--insecure-registry` daemon flag, you should get the following error:

Error message

```
Error: Invalid registry endpoint https://artprod.company.com/v1/: Get  
https://artprod.company.com/v1/_ping: x509: certificate signed by unknown authority.
```

Using previous versions of Docker

In order to use self-signed SSL certificates with previous versions of Docker, you need to manually install the certificate into the OS of each machine running the Docker client (see [Issue 2687](#)).

Alternative Proxy Servers

In addition to NGINX, you can setup Artifactory to work with Docker using Apache.

Apache Configuration

The sample configuration below configures SSL on port 443 and a server name of `artprod.company.com`.

Apache config for docker V1

```
<VirtualHost *:443>  
    ServerName artprod.company.com  
  
    ErrorLog ${APACHE_LOG_DIR}/error.log  
    CustomLog ${APACHE_LOG_DIR}/access.log combined  
  
    SSLEngine on  
    SSLCertificateFile /etc/ssl/certs/artprod.company.com.pem  
    SSLCertificateKeyFile /etc/ssl/private/artprod.company.com.key  
  
    ProxyRequests off  
    ProxyPreserveHost on  
  
    ProxyPass /  
    http://artprod.company.com:8080/artifactory/api/docker/docker-local/  
    ProxyPassReverse /  
    http://artprod.company.com:8080/artifactory/api/docker/docker-local/  
</VirtualHost>
```

Port Bindings

If you want to use multiple repositories, you need to copy the [NGINX configuration](#) and bind different ports to each local repository in Artifactory.

When binding a port other than 443, note that the configuration for the proxy header must be appended with the port number on the `proxy_set_header` line.

For example, for a server running on port 444 you should write `proxy_set_header Host $host:444`.

Docker Repository Path and Domain

When accessing a Docker repository through Artifactory, the repository URL must be prefixed with `api/docker` in the path.

You can copy the full URL from the UI using **Set Me Up** when the repository is selected in the Tree Browser.

For example, if you are using Artifactory standalone or as a local service, you would access your Docker repositories using the following URL:

`http://localhost:8081/artifactory/api/docker/<repository key>`

Also, the domain of your Docker repository must be expressed as an explicit IP address. The only exception is when working locally, you can use

the `localhost` domain name as the proxy pass.

Support Matrix

Please refer to the support matrix under Docker Repositories.

Repository Replication

Overview

Through the Replication Add-on in Artifactory Pro, Artifactory allows replication of repositories to support development by different teams distributed over distant geographical sites. The benefits of replication are:

- Ensuring developers all work with the same version of remote artifacts
- Ensuring build artifacts are shared efficiently between the different development teams
- Overcome connectivity issues such as network latency and stability when accessing remote artifacts
- Accessing specific versions of remote artifacts

Artifactory versions for replication

We strongly recommend that replication is only performed between servers running the same version of Artifactory Pro.

Two main methods of replication are supported:

- Push replication
- Pull replication

Push Replication

Push replication is used to synchronize [Local Repositories](#), and is implemented by the Artifactory server on the near end invoking a synchronization of artifacts to the far end.

There are two ways to invoke push replication:

- **Scheduled push:** Pushes are scheduled asynchronously at regular intervals
- **Event-based push:** Pushes occur in nearly in real-time since each create, copy, move or delete of an artifact is immediately propagated to the far end.

Advantages

- It is fast because it is asynchronous.
- It minimizes the time that repositories are not synchronized.
- It reduces traffic on the master node in case of a replication chain ("Server A" replicates to "Server B", "Server B" then replicates to "Server C" etc.).

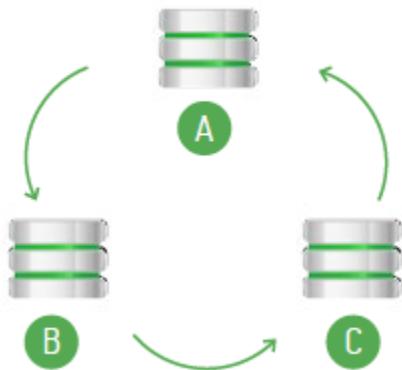
Avoid Replication Loops ("Cyclic Replication")

A replication loop occurs ("Cyclic" or "Bi-directional" replication) occurs when two instances of Artifactory running on different servers are replicating content from one to the other concurrently.

For example, "Server A" is configured to replicate its repositories to "Server B", while at the same time, "Server B" is configured to replicate its repositories to "Server A".

Or "Server A" replicates to "Server B" which replicates to "Server C" which replicates back to "Server A".

We strongly recommend avoiding cyclic replication since this can have disastrous effects on your system causing loss of data, or conversely, exponential growth of disk-space usage.



Replication loop to be strictly avoided

When to Use Push Replication

Event-based push replication is recommended when it is important for the repository at the far end to be updated in near-real-time for any change (create, copy, move or delete of an artifact) in the repository at the near end.

In addition, we recommend running regular scheduled replication on top of event-based replication to guarantee full copy consistency even in cases of server downtime and network partitions.

Multi-push Replication

With an Enterprise license, Artifactory supports multi-push replication allowing you to replicate a local repository from a single source to multiple enterprise target sites simultaneously.

Pull Replication

Pull replication is invoked by a remote repository, and runs according to a defined schedule to synchronize repositories (local, remote or virtual) at regular intervals.

This provides a convenient way to proactively populate a remote cache, and is very useful when waiting for new artifacts to arrive on demand (when first requested) is not desirable due to network latency.

Pull replication runs as a scheduled task.

Advantages

- Many target servers can pull from the same source server efficiently implementing a one-to-many replication.
- It is safer since each package only has one "hop".
- It reduces traffic on target servers since they do not have to pass on artifacts in a replication chain.

When to Use Pull Replication

Pull replication is recommended in the following cases:

- When you need to replicate a repository to many targets.
- When your source repository is located behind a proxy that prevents push replication (e.g. replicating a repository hosted on Artifactory online to a local repository at your site)

Page Contents

- Overview
 - Push Replication
 - Advantages
 - Multi-push Replication
 - Pull Replication
 - Advantages
- Scheduling and Configuring Replication
 - Using the UI
 - Configuring Push Replication
 - Configuring Pull Replication
 - Replicating with REST API
 - Replication Properties
- Watch the Screencast

Scheduling and Configuring Replication

Using the UI

Replication is configured via the user interface as a scheduled task. Local repositories can be configured for push replication, and remote repositories can be configured for pull replication.

All replication messages are logged in the main [Artifactory log file](#) (`artifactory.log`).

Configuring Push Replication

A push replication task for a Local Repository is configured in the **Replication** tab of the [Edit Local Repository](#) dialog.

First, in the **Cron Expression** field define the replication task schedule using a valid `cron` expression.

The **Next Replication Time** will indicate update accordingly.

The repository is replicated to all targets using the same schedule

Replication of this repository to all of its targets occurs simultaneously according to the **Cron Expression** you define.

To add a target site for this replication, click **Add** to display the **Replication Properties** dialog, and fill in the details as follows.

Field Name	Description
<i>Enable Active Replication of this Repository</i>	When set, this replication will be enabled when saved

<i>URL</i>	The URL of the target local repository on a remote Artifactory server. For repositories of most package types, you should omit the api/<pkg> that normally prefixes the repository key in the URL (* See note below).
<i>Username</i>	The HTTP authentication username.
<i>Password</i>	The HTTP authentication password.
<i>Proxy</i>	A proxy configuration to use when communicating with the remote instance.
<i>Socket Timeout</i>	The network timeout in milliseconds to use for remote operations.
<i>Sync Deletes</i>	When set, items that were deleted remotely should also be deleted locally (also applies to properties metadata).

Sync Properties	When set, the task also synchronizes the properties of replicated artifacts.
Path Prefix (optional)	A subpath to replicate within the repository.

Replication Properties

X

Enable Active Replication of this Repository

Url*

http://[REDACTED]:8080/artifactory/lib-release-local

Username*

admin

Password

[REDACTED]

Network Proxy Reference

Socket Timeout*

1500

Sync Deleted

Sync Properties

Path prefix

Test

Save

*** Don't prefix the repository path with api/<pkg>, but watch out for exceptions**

For most packaging formats, when using the corresponding client to access a repository through Artifactory, the repository key in the URL needs to be prefixed with **api/<pkg>** in the path. For example, in the case of **Npm** repositories, the repository key should be prefixed with **api/npm**. In the case of replication, the corresponding package client is not involved in the action, so the **api/<pkg>** prefix should be omitted from the URL of the target repository.

There are exceptions to this rule. For example, when replicating PyPI repositories, you do need to prefix the remote repository path with **api/pypi**.

Once you have configured the replication properties for each of your replication targets, the **Replication** tab for your repository displays them.

Edit libs-release-local

Basic Advanced Replications

Cron Expression * Next Replication Time

 Wed Jul 15 14:00:00 UTC 2015

Enable Event Replication

Replication

Url	Sync Deletes	Sync Properties	Enabled
http://...:8080/artifactory/lib-release-local	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
http://...:8080/artifactory/lib-release-local	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

New
< page 1 of 1 >

Cancel < Back Next > **Save & Finish**

Field Name	Description
Push to	The replication targets you have defined
Enabled	When set, enables replication of this repository to the target specified in Push to
Enable Event Replication	When set, event-based push replication is enabled

Number of replication targets

If you do not have an Enterprise license, you may only define **one** replication target. With an Enterprise license, Artifactory supports multi-push replication and you may define as many targets as you need.

Configuring Pull Replication

Configuring Pull Replication for a Remote Repository is very similar to configuring Push Replication for a Local Repository. Please refer to Configuring Push Replication for details.

Edit gradle-libs

Basic > Advanced > Replications >

Enable Active Replication of this Repository

Cron Expression*

Next Replication Time

Sync Deletes Sync Properties

Path Prefix

Test

Cancel < Back Next > **Save & Finish**

Regarding credentials of the remote repository configuration

The remote repository's file listing for replication is retrieved using the repository's credentials defined under the repository's Advanced configuration section.

The remote files retrieved depend on the effective permissions of the configured user on the remote repository (on the other Artifactory instance).

Replicating with REST API

Both Push and Pull Replication are supported by Artifactory's REST API. For details please refer to the following:

- Get Repository Replication Configuration
- Set Repository Replication Configuration
- Update Repository Replication Configuration
- Delete Repository Replication Configuration
- Scheduled Replication Status
- Pull/Push Replication

Replication Properties

Once replication has been invoked, Artifactory annotates the source repository being replicated and annotates it with properties that indicate the status of the replication. These can be viewed, along with other properties that may annotate the repository, in the **Properties** tab of the **Tree Browser**.

For single push replication operations, the following properties are created/updated:

Key	Value
artifactory.replication.<source_repo_key>.started	Indicates when the replication started

artifactory.replication.<source_repo_key>.status	Indicates the status of the replication operation once complete. It can take the following values: ok: The replication succeeded failure: The replication failed. You should check the log files for errors
artifactory.replication.<source_repo_key>.finished	Indicates when the replication finished

libs-snapshot-local

Actions

General Effective Permissions Properties Watchers

Filter by Property

Value(s)

Property	Value(s)
artifactory.replication.libs-snapshot-local.started	1437483921588
artifactory.replication.libs-snapshot-local.finished	1437484260530
artifactory.replication.libs-snapshot-local.result	ok

For multi-push replication operations (available to Enterprise customers only), the following properties are created/updated:

Key	Value
artifactory.replication.<source_repo_key>_<target_repo_URL>.started	Indicates when the replication started

artifactory.replication.<source_repo_key>_<target_repo_URL>.status	Indicates the status of the replication operation once complete. It can take the following values: ok: The replication succeeded failure: The replication failed. You should check the log files for errors
artifactory.replication.<source_repo_key>_<target_repo_URL>.finished	Indicates when the replication finished

Watch the Screencast

To see the Replication in action you can watch the short demo screencast below.

Artifactory Query Language

Overview

Artifactory Query Language (AQL) is specially designed to let you uncover any data related to the artifacts and builds stored within Artifactory. Its syntax offers a simple way to formulate complex queries that specify any number of search criteria, filters, sorting options, and output parameters. AQL is exposed as a RESTful API which uses data streaming to provide output data resulting in extremely fast response times and low memory consumption. Currently, AQL can only extract data that resides in your instance of Artifactory, so it runs on [local repositories](#) and [remote repository caches](#).

Here are a few simple examples:

```

// Return all artifacts of the "artifactory" build.
items.find({"@build.name": {"$eq": "artifactory"}})

// Return all builds that have a dependency with a license
// that is not Apache.
builds.find({"module.dependency.item@license": {"$nmatch": "Apache-*"}})

// Return all archives containing a file called
// "org/artifactory/Main.class".
items.find({"archive.entry.name": {"$eq": "Main.class"} },
           {"archive.entry.path": {"$eq": "org/artifactory"}})

```

Here is a slightly more complex example.

```

// Return all entries of any archive named "Artifactory.jar"
// from any build named "Artifactory" with build number 521.
archive.entries.find(
    {"archive.item.name": {"$eq": "Artifactory.jar"},

     "archive.item.artifact.module.build.name": {"$eq": "Artifactory"
    },
     "archive.item.artifact.module.build.number": {"$eq": "521"}
    )

```

Page Contents

- Overview
- Architecture
 - Supported Domains
- Usage
 - Syntax
 - Using Fields
 - Execution
- Entities and Fields
- Constructing Search Criteria
 - Field Criteria
 - Properties Criteria
 - Compounding Criteria
 - Matching Criteria on a Single Property (\$msp)
 - Comparison Operators
 - Using Wildcards
 - Using Wildcards with \$match and \$nmatch
 - "Catch all" Notation on Properties
 - Examples
 - Date and Time Format
- Specifying Output Fields
 - Displaying All Fields
 - Displaying Specific Fields
 - Filtering Properties by Key
- Sorting
- Display Limits and Pagination

Here is another example that shows the full power of AQL to mine information from your repositories in a way that no other tool can match.

```

// Compare the contents of artifacts in 2 "maven+example" builds
items.find(
{
  "name": {"$match": "multi2*.jar"},
  "$or": [
    {
      "$and": [
        {"artifact.module.build.name": {"$eq": "maven+example"}},
        {"artifact.module.build.number": {"$eq": "317"}}
      ]
    },
    {
      "$and": [
        {"artifact.module.build.name": {"$eq": "maven+example"}},
        {"artifact.module.build.number": {"$eq": "318"}}
      ]
    }
  ]
}).include("archive.entry")

```

▼ Click to view the output of this query...

```
{
  "results": [
    {
      "repo": "ext-snapshot-local",
      "path": "org/jfrog/test/multi2/3.0.0-SNAPSHOT",
      "name": "multi2-3.0.0-20151012.205507-1.jar",
      "type": "file",
      "size": 1015,
      "created": "2015-10-12T22:55:23.022+02:00",
      "created_by": "admin",
      "modified": "2015-10-12T22:55:23.013+02:00",
      "modified_by": "admin",
      "updated": "2015-10-12T22:55:23.013+02:00",
      "archives": [
        {
          "entries": [
            {
              "entry.name": "App.class",
              "entry.path": "artifactory/test"
            },
            {
              "entry.name": "MANIFEST.MF",
              "entry.path": "META-INF"
            }
          ]
        },
        {
          "repo": "ext-snapshot-local",
          "path": "org/jfrog/test/multi2/3.0.0-SNAPSHOT",
          "name": "multi2-3.0.0-20151013.074226-2.jar",
          "type": "file",
          "size": 1015,
          "created": "2015-10-13T09:42:39.389+02:00",
          "created_by": "admin",
          "modified": "2015-10-13T09:42:39.383+02:00",
          "modified_by": "admin",
          "updated": "2015-10-13T09:42:39.383+02:00",
          "archives": [
            {
              "entries": [
                {
                  "entry.name": "App.class",
                  "entry.path": "artifactory/test"
                },
                {
                  "entry.name": "MANIFEST.MF",
                  "entry.path": "META-INF"
                }
              ]
            }
          ],
          "range": {
            "start_pos": 0,

```

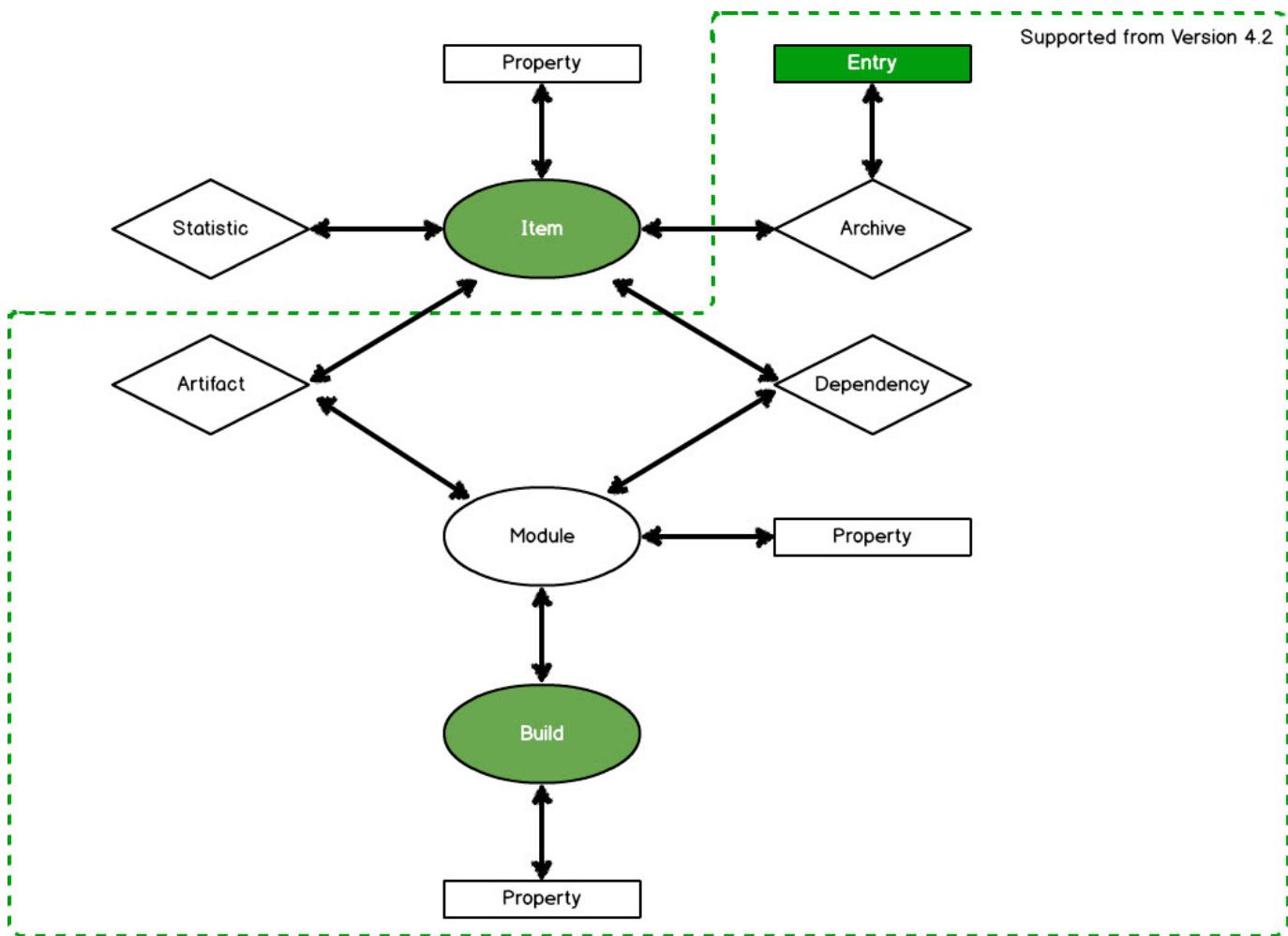
```
"end_pos" : 2,  
"total" : 2  
}  
}
```

Architecture

AQL is constructed as a set of interconnected domains as displayed in the diagram below. You may run queries only one of the domains at a time, and this is referred to as the **Primary** domain of the query.

Currently, the following are supported as primary domains: **Item**, **Build** or **Entry**. i.e., your queries may be of the form: `items.find(...)`, `builds.find(...)` or `archive.entries.find(...)`.

You may use fields from other domains as part of your search criteria or to specify fields to display in the output, but in that case, you need to follow the conventions described in [Using Fields](#).



Supported Domains

AQL was introduced in Artifactory V3.5.0 with support for **Item** as a primary domain with its attached **Property**, as well as **Statistic** as a secondary domain.

From version 4.2 the additional domains demarcated in the diagram above are also supported with **Item**, **Build** and **Archive**.**Entry** as primary domains

Usage

Syntax

```
<domain_query>.find(<criteria>).include(<fields>).sort(<order_and_fields>).limit(<num_records>).offset(<offset_records>)
```

where:

domain_query	The query corresponding to the primary domain. Must be one of items , builds or entries .
criteria	The search criteria in valid JSON format
fields	(Optional) There is a default set of fields for query output. This parameter lets you specify a different set of fields that should be included in the output
order_and_fields	(Optional) The fields on which the output should be sorted, and the sort order. A default set of fields and sort order is defined for each domain.
num_records	(Optional) The maximum number of records that should be extracted. If omitted, all records answering the query criteria will be extracted.
offset	(Optional) The offset from the first record from which to display results (i.e. how many results should be skipped for display)

Limitation

Sort, limit and offset elements only works in the following cases:

- Your query does not have an **include** element
- If you do have an **include** element, you only specify fields from the primary domain in it.

For example, in the following query, **sort**, **limit** and **offset** will not work because the primary domain is **item**, but the **include** element specifies that fields from the **artifact**, **module** and **build** domains should be displayed:

```
items.find().include("artifact", "artifact.module", "artifact.module.build")
```

Using Fields

Any fields from your primary domain can be used directly anywhere in your query. If you use fields from other domains, they must be specified using a complete relation path from the primary domain.

For example, to find all items in a repository called "myrepo" you would use:

```
items.find({"repo": "myrepo"})
```

But to find all items created by modules named "mymodule" you would use:

```
items.find({"artifact.module.name" : "mymodule"})
```

And since you may also issue a query from the **build** domain, to find all builds that generated an item called "artifactory.war", you could also use:

```
builds.find({"module.artifact.item.name": "artifactory.war"})
```

Execution

To execute an AQL query, use the [Artifactory Query Language REST API](#).

Entities and Fields

You may issue a **find** request according to the [syntax](#) above, and configure your request to display fields from any of the domains.

Domain	Field Name	Type	Description
item	repo	String	The name of the repository in which this item is stored
	path	String	The full path associated with this item
	name	String	The name of the item
	created	Date	When the item was created
	modified	Date	File system timestamp indicating when the item was last modified
	updated	Date	When the item was last uploaded to a repository.
	created_by	String	The name of the item owner
	modified_by	String	The name of the last user that modified the item
	type	Enum	The item type (file/folder/any). If type is not specified in the query, the default type searched for is file
	depth	int	The depth of the item in the path from the root folder
	original_md5	String	The item's md5 hash code when it was originally uploaded

	actual_md5	String	The item's current md5 hash code
	original_sha1	String	The item's sha1 hash code when it was originally uploaded
	actual_sha1	String	The item's current sha1 hash code
	size	long	The item's size on disk
archive			The archive domain currently contains no fields
entry	name	String	The entry's name
	path	String	The path of the entry within the repository
build	url	String	The URL of the build
	name	String	The build name
	number	String	The build number
	created	Date	File system timestamp indicating when the item was last modified
	created_by	String	The name of the user who created the build
	modified	Date	File system timestamp indicating when the build was last modified
	modified_by	String	The name of the last user that modified the build
property	key	String	The property key
	value	String	The property value
stat	downloaded	date	The last time an item was downloaded

	downloads	int	The total number of downloads for an item
	downloaded_by	String	The name of the last user to download this item
artifact	name	String	The name of the artifact
	type	String	The type of the artifact
	sha1	String	The SHA1 hash code of the artifact
	md5	String	The MD5 hash code of the artifact
module	name	String	The name of the module
dependency	name	String	The name of the dependency
	scope	String	The scope of the dependency
	type	String	The type of the dependency
	sha1	String	The SHA1 hash code of the dependency
	md5	String	The MD5 hash code of the dependency

Constructing Search Criteria

The **criteria** element must be a valid JSON format statement composed of the criteria that specify the items that should be returned. It is essentially a compound boolean statement, and only elements for which the statement evaluates to **true** are returned by the query.

Each criterion is essentially a comparison statement that is applied either to a field or a property. Please see the full list of [Comparison Operators](#). While each criterion may be expressed in complete general format, AQL defines shortened forms for readability as described below.

Field Criteria

The general way to specify a criterion on a field is as follows:

```
{ "<field>" : { "<comparison operator>" : "<value>" } }
```

If the query applied is to a different domain, then field names must be pre-pended by a relation path to the primary domain.

For example:

```

//Find items whose "name" field matches the expression "*test.*"
items.find({ "name": { "$match" : "*test.*" }})

//Find items that have been downloaded over 5 times.
//We need to include the "stat" specifier in "stat.downloads" since downloads is a
//field of the stat domain and not of the item domain.
items.find({ "stat.downloads": { "$gt": "5" }})

//Find builds that use a dependency that is a snapshot
builds.find({ "module.dependency.item.name": { "$match": "*SNAPSHOT*" }})

```

Short notation for Field criteria

AQL supports a short notation for search criteria on fields.

An "equals" ("\$eq") criterion on a field may be specified as follows:

```
{ "<field>" : "<value>" }
```

Example	Find items whose "name" field equals "ant-1.9.4.jar"
Regular notation	items.find({ "name" : { "\$eq" : "ant-1.9.4.jar" } })
Short notation	items.find({ "name" : "ant-1.9.4.jar" })

Properties Criteria

Artifactory lets you attach, and search on properties in three domains: **items**, **modules** and **builds**.

The general way to specify a criterion on a property is as follows:

```
{ "@<property_key>" : { "operator": "<property_value>" } }
```

Accessing the right properties

If you are specifying properties from the primary domain of your query, you may simply enter the property key and value as described above. If you are specifying properties from one of the other domains, you need to specify the full relational path to the property.

In the example below, the primary domain is the **build** domain, but we want to find builds based a property in the **item** domain, so we must specify the full path to the property:

```
builds.find( { "module.artifact.item@qa_approved" : { "$ne" : "true" } })
```

Here are some examples:

```

//Find items that have been approved by QA
items.find({ "@qa_approved" : { "$eq" : "true" }})

//Find builds that were run on a linux machine
builds.find({ "@os" : { "$match" : "linux*" }})

//Find items that were created in a build that was run on a linux machine.
items.find({ "artifact.module.build@os" : { "$match" : "linux*" }})

```

Short notation for properties criteria

AQL supports a short notation for search criteria on properties.

An "equals" ("\$eq") criterion on a property may be specified as follows:

```
{"@<property_key>" : "<property_value>"}
```

Example	Find items with associated properties named "license" with a value that equals "GPL"
Regular notation	items.find({ "@artifactory.licenses" : { "\$eq" : "GPL" } })
Short notation	items.find({ "@artifactory.licenses" : "GPL" })

Compounding Criteria

Search criteria on both fields and properties may be nested and compounded into logical expressions using "\$and" or "\$or" operators. If no operator is specified, the default is \$and

```
<criterion>={<"$and" | "$or">:[{<criterion>},{<criterion>}]}
```

Criteria may be nested to any degree

Note that since search criteria can be nested to any degree, you may construct logical search criteria with any degree of complexity required.

Here are some examples:

```

//This example shows both an implicit "$and" operator (since this is the default, you
don't have to expressly specify it, but rather separate the criteria by a comma), and
an explicit "$or" operator.
//Find all items that are files and are in either the jcenter or my-local
repositories.
items.find({"type" : "file","$or": [{"repo" : "jcenter", "repo" : "my-local"}]})

//Find all the items that are either in a repository called "debian" and whose name
ends with ".deb" or are in a repository called "yum" and whose name ends with ".rpm".
items.find(
{
  "$or":
  [
    {
      "$and":
      [
        {"artifact.module.build.name" : "my_debian_build"} ,
        {"name" : {"$match" : "*.*deb"}}
      ]
    },
    {
      "$and":
      [
        {"artifact.module.build.name" : "my_yum_build"} ,
        {"name" : {"$match" : "*.*rpm"}}
      ]
    }
  ]
}
)

//Find all items in a repository called "my_local" that have a property with a key
called "license" and value that is any variant of "LGPL".
items.find({"repo" : "my_local"},{@artifactory.licenses" : {"$match" : "**LGPL**"}})

```

Matching Criteria on a Single Property (\$msp)

A search that specifies several criteria on properties may sometimes yield unexpected results.

This is because items are frequently annotated with several properties, and as long as any criterion is true for any property, the item will be returned in a regular **find**.

But sometimes, we need to find items in which a single specific property answers several criteria. For this purpose we use the **\$msp (match on single property)** operator.

The fundamental difference between a regular **find** and using the **\$msp** operator is:

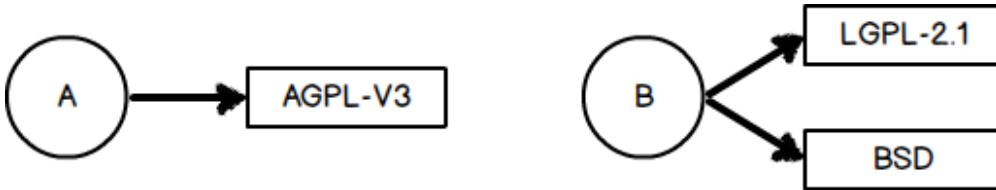
- **find** will return an item if **ANY** of its properties answer **ALL** of the criteria in the search term.
- **\$msp** will only return an item if **ONE** (and only one) of its properties answers **ALL** of the criteria in the **\$msp** term.

Here is an example.

Consider two items A and B.

A has a license property with value **AGPL-V3**

B has two license properties . One is **LGPL-2.1**, and the other **BSD**



Now let's assume we want to find items that use any variety of GPL license as long as it's NOT LGPL.

In our example we would expect to only get **Item A** returned since item B does use LGPL-2.1.

As a first thought, we might write our query as follows:

```
items.find({"@artifactory.licenses": {"$match": "*GPL*"}, {"@artifactory.licenses": {"$nmatch": "*LGPL*"}})
```

But this query returns **both items A and B** even though B does have an LGPL license.

Item A is returned because it clearly answers both criteria.

However, B is also returned. The LGPL-2.1 license answers the first criterion. The BSD license answers the second criterion (it's not an LGPL license).

If we use the **\$msp** operator as follows:

```
items.find({"$msp": [{"@artifactory.licenses": {"$match": "*GPL*"}, {"@artifactory.licenses": {"$nmatch": "*LGPL*"}]}])
```

Then only Item A is returned. As soon as the find detects the LGPL license in Item B, it is rejected.

Note that the **\$msp** operator works equally well on all domains that have properties: **item**, **module** and **build**.

Comparison Operators

The following table lists the full set of comparison operators allowed:

Operator	Types	Meaning
\$ne	string, date, int, long	Not equal to
\$eq	string, date, int, long	Equals
\$gt	string, date, int, long	Greater than
\$gte	string, date, int, long	Greater than or equal to
\$lt	string, date, int, long	Less than
\$lte	string, date, int, long	Less than or equal to
\$match	string	Matches
\$nmatch	string	Does not match

Using Wildcards

To enable search using non-specific criteria, AQL supports wildcards in common search functions.

Using Wildcards with \$match and \$nmatch

When using the "\$match" and "\$nmatch" operators, the "*" wildcard replaces any string and the "?" wildcard replaces a single character.

"Catch all" Notation on Properties

In addition to supporting "\$match" and "\$nmatch", AQL supports a notation that uses wildcards to match **any** key or **any** value on properties.

If you specify "@*" as the property key, then it means a match on any key.

If you specify "*" as the property value, then it means a match on any value

Example	Find items that have any property with a value of "GPL"
Regular notation	<code>items.find({ "\$and" : [{ "property.key" : { "\$eq" : "*" } } , { "property.value" : { "\$eq" : "GPL" } }] })</code>
Short notation	<code>items.find({ "@*" : "GPL" })</code>

Example	Find any items annotated with any property whose key is "license" (i.e. find any items with a "license" property)
Regular notation	<code>items.find({ "\$and" : [{ "property.key" : { "\$eq" : "license" } } , { "property.value" : { "\$eq" : "*" } }] })</code>
Short notation	<code>items.find({ "@artifactory.licenses" : "*" })</code>

Be careful not to misuse wildcards

Wildcard characters ("*" and "?") used in queries that do not conform to the above rules are interpreted as literals.

Examples

To avoid confusion, here are some examples that use the "*" and "?" characters explaining why they are interpreted as wildcards or literals.

Query	Wildcard or Literal	Explanation	What query returns
<code>items.find({ "name" : "?" })</code>	?	Interpreted as a literal question mark.	Items with a name containing a question mark.

<code>items.find({ "name" : { "\$match" : "ant-1.9.4.*" } })</code>	Wildcard	Wildcards on fields are allowed with the \$match operator.	All items whose name matches the expression "ant-
<code>items.find({ "name" : { "\$eq" : "ant-1.9.4.*" } })</code>	Literal	Wildcards on fields are only allowed with the \$match and \$n match operators.	Only items whose name is literally "ant-
<code>items.find({ "@artifactory.licenses" : "*" })</code>	Wildcard	For properties, this short notation is allowed and denotes any value	All items with a property whose value is "lic
<code>items.find({ "@artifactory.licenses" : "*GPL" })</code>	Literal	This is the short notation replacing the \$eq operator for properties, but it does not use the "catch all" notation for properties.	All items with a license whose value is literally "*GP
<code>items.find({ "@artifactory.licenses" : { "\$match" : "*GPL*" } })</code>	Wildcard	Wildcards on properties are allowed with the \$match operator.	All items with a license matching the expression "*GP

Date and Time Format

AQL supports Date and Time formats according to a [W3C profile](#) of the ISO 8601 Standard for Date and Time Formats.

The complete date and time notation is specified as:

YYYY-MM-DDThh:mm:ss.sTZD (eg 2012-07-16T19:20:30.45+01:00)

Date/Time specified in partial precision is also supported: (i.e. specify just the year, or year and month, year, month and day etc.)

For example, the following query will return all items that were modified after July 16, 2012 at 30.45 seconds after 7:20pm at GMT+1 time zone:

```
//Find all the items that have been modified after 2012-07-16T19:20:30.45+01:00
items.find({"modified" : {"$gt" : "2012-07-16T19:20:30.45+01:00"}})

//Find all the builds that have were created after 2012-07-01
builds.find({"created" : {"$gt" : "2012-07-01"}})
```

For full details, please refer to the W3C documentation.

Specifying Output Fields

Each query displays a default set of fields in the result set, however you have complete control this and may specify which fields to display using an optional **include** element in your query.

You can even specify to display fields from other entities related to your result set.

Displaying All Fields

Use: `.include("*")`

For example:

```
//Find all items, and display all the item fields
items.find().include("*")
```

Displaying Specific Fields

Each query displays a default set of fields in the output. Using the `.include` element you can override this default setting and specify any particular set of fields you want to receive in the output.

Use: `.include("<field1>", "<field2>"...)`

For example:

```
//Find all items, only display the "name" and "repo" fields
items.find().include("name", "repo")
```

You can also display specific fields from other entities associated with those returned by the query.

If you specify any field from the **item** domain, then this will override the default output setting, and only the **item** fields you expressly specified will be displayed.

If you only specify fields from the **property** or **stat** domains, then the output will display the default fields from the **item** domain, and in addition, the other fields you expressly specified from the **property** or **stat** domains.

For example:

```

//Find all items, and display the "name" and "repo" fields as well as the number of
//downloads from the corresponding "stat" entity
items.find().include("name", "repo", "stat.downloads")

//Find all items, and display the default item fields fields as well as the stat
//fields
items.find().include("stat")

//Find all items, and display the default item fields as well as the stat and the
//property fields
items.find().include("stat", "property")

//Find all items, and display the "name" and "repo" fields as well as the stat fields
items.find().include("name", "repo", "stat")

//Find all builds that generated items with an Apache license, and display the build
//fields as well as the item "name" fields. Click below to view the output of this query
builds.find({
    "module.artifact.item.@license": {"$match": "Apache*"}
})
.include("module.artifact.item.name")

```

Click to view the output of the last query

Note that the output displays the default fields of the "build" domain, and the "name" field from the item domain. Fields from the module and artifact domains are not displayed since they were not specified in the include element.

```
{
    "results": [
        {
            "build.created": "2015-09-06T15:49:01.156+03:00",
            "build.created_by": "admin",
            "build.name": "maven+example",
            "build.number": "313",
            "build.url": "http://localhost:9595/jenkins/job/maven+example/313/",
            "modules": [
                {
                    "artifacts": [
                        {
                            "items": [
                                {
                                    "name": "multi-3.0.0-20150906.124843-1.pom"
                                }
                            ]
                        }
                    ]
                }
            ],
            "build.created": "2015-09-06T15:54:40.726+03:00",
            "build.created_by": "admin",
            "build.name": "maven+example",
            "build.number": "314",
            "build.url": "http://localhost:9595/jenkins/job/maven+example/314/",
            "modules": [
                {
                    "artifacts": [
                        {
                            "items": [
                                {
                                    "name": "multi-3.0.0-20150906.124843-1.pom"
                                }
                            ]
                        }
                    ]
                }
            ]
        },
        {
            "range": {
                "start_pos": 0,
                "end_pos": 2,
                "total": 2
            }
        }
    ]
}
```

```
}
```

Filtering Properties by Key

As described above, the primary use of the `.include` element is to specify output fields to display in the result set.

This notion is applied in a similar way in regard to properties. Each item may be annotated with several (even many) properties. In many cases you may only be interested in a specific subset of the properties, and only want to display those.

So the `.include` element can be used to filter out unwanted properties from the result, and only display (i.e. "include") those you are interested in.

For example, to display all the properties annotating an item found :

```
//Find all items, and display the "name" and "repo" fields, as well as all properties
//associated with each item
items.find().include("name", "repo", "property.*")
```

However, if you are only interested in a specific property (e.g. you just want to know the version of each item returned), you can filter out all other properties and only include the property with the key you are interested in:

```
//Find all items, and display the "name" and "repo" fields, as well as the key and
//value of the "version" property of each item
items.find().include("name", "repo", "@version")
```

Sorting

AQL implements a default sort order, however, you can override the default and specify any other sort order using fields in your output by adding the `.sort` element to the end of your query as follows:

```
.sort({"$asc | $desc" : ["<field1>", "<field2>,..."]})
```

You can only specify sorting on fields that are displayed in the output (whether they are those displayed by default or due to a `.include` element).

Here are some examples:

```
// Find all the jars in artifactory and sort them by repo and name
items.find({"name" : {"$match":".jar"}).sort({"$asc" : ["repo","name"]})

// Find all the jars in artifactory and their properties, then sort them by repo and
// name
items.find({"name" : {"$match":".jar"}).include("@").sort({"$asc" : ["repo","name"]})
```

Display Limits and Pagination

Using the `.limit` elements, you can limit the number of records that will be displayed by your query.

```
// Find all the jars in artifactory and sort them by repo and name, but only display
// the first 100 results
items.find({"name" : {"$match":".jar"}).sort({"$asc" : ["repo","name"]}).limit(100)
```

You can also implement pagination when you want to focus on a subset of your results using the `.offset` element.

```
//Run the same example, but this time, display up to 50 items but skipping the first  
100  
items.find({"name" : {"$match": "*.jar"}).sort({"$asc" :  
["repo", "name"]}).offset(100).limit(50)
```

limitation

Note the [limitation](#) on `sort`, `limit` and `offset` described above.

S3 Object Storage

Overview

Artifactory fully supports S3 object storage for distributed file systems so your Artifactory filestore can reside on the cloud. This present several benefits:

1. Unlimited scalability

Since your files are now stored on the cloud, this means that your Artifactory filestore is scalable and effectively unlimited (to the extent offered by your storage provider). You may freely continue to upload files without having to install or maintain any file storage devices. You can even upload files larger than 5 GB using multi-part upload.

2. Security

Enjoy the same security and authentication mechanisms provided by your S3 provider.

3. Disaster recovery

Since your files are replicated and stored with redundancy, this offers the capability for disaster recovery.

4. Support any S3 compatible distributed file system

Arifactory's support is based on the S3 protocol. Any provider that uses S3, such as Ceph, Swift and others, will also be supported by Artifactory. With support for AWS S3 version 4, you can sign AWS requests using [Signature Version 4](#).

Support for S3 object storage is included with Artifactory Pro Enterprise Value Pack.

In order to use S3 object storage with Artifactory, make sure you first [install](#) or [upgrade](#) to Artifactory V3.6 or later.

Backup your system. Your current filestore will be deleted.

Setting up Artifactory to use S3 will delete all files in your current filestore.

If you already have a running installation of Artifactory, then before you setup Artifactory to use S3 and migrate your filestore to the cloud, we strongly recommend that you do a [complete system backup](#).

Page Contents

- [Overview](#)
- [Setting up Artifactory to Use S3](#)
 - [Setting Your License](#)
 - [Configuring Artifactory to Use S3](#)
 - [Migrating Your Filestore](#)
 - [Automatic Filestore Migration \(Recommended\)](#)
 - [Manual Filestore Migration](#)

Setting up Artifactory to Use S3

First time installation or upgrade

If you are moving your filestore to S3 in the context of upgrading Artifactory, or a first time installation, we recommend that you first do a standard installation of Artifactory using the default settings, or a standard upgrade using your current settings.

In order to move your Artifactory filestore to the cloud, you need to execute the following steps:

- Shut down Artifactory.
- [Set your enterprise license](#)
- [Configure Artifactory to use your S3 object storage provider](#)
- [Migrate your files to the cloud manually or automatically](#)
- Start up Artifactory

Setting Your License

To use Artifactory's support for S3, you need to have an enterprise license with your Artifactory installation.

To do so, make sure your `$ARTIFACTORY_HOME/etc/artifactory.lic` file contains your enterprise license.

Configuring Artifactory to Use S3

To configure Artifactory to use your S3 object storage provider, you need to add or modify the following parameters to your `$ARTIFACTORY_HOME/etc/storage.properties` file:

Parameter	Description
-----------	-------------

<i>binary.provider.type</i>	This must be set to S3 .
	<p>Were you using fulldb? If you were storing your binaries as blobs in the database (i.e. this was set to fulldb), make sure you remove all parameters connected to using fulldb.</p>
<i>binary.provider.s3.identity</i>	Your username or IAM user key for your S3 provider.
<i>binary.provider.s3.credential</i>	Your password or IAM secret for your S3 provider.
<i>binary.provider.s3.endpoint</i>	<p>The URL of your provider's endpoint for access.</p> <p>Connect to the nearest endpoint If your S3 provider supports connections to different endpoints, we recommend connecting to the geographically nearest endpoint for best performance. For example, if you are using AWS-S3, you should configure your endpoint according to the regions specified in the AWS documentation.</p>
<i>binary.provider.s3.bucket.name</i>	<p>(Optional) The name of the bucket at your S3 provider Default: artifactory</p>
<i>binary.provider.s3.bucket.path</i>	<p>(Optional) Path within your bucket Default: filestore</p>

Keep your database settings

Make sure you don't change your database settings in your *storage.properties* file.

For example, a *storage.properties* file that uses an S3 object storage provider should look like this:

```

## Database settings - these should not be modified when migrating your filestore to
S3
type=derby
url=jdbc:derby:{db.home};create=true
driver=org.apache.derby.jdbc.EmbeddedDriver

## Specify object storage provider
binary.provider.type=S3

## S3 identity
binary.provider.s3.identity=<Your S3 provider username>

## S3 credential
binary.provider.s3.credential=<Your S3 provider password>

## S3 endpoint - in this example, this is the endpoint for AWS
binary.provider.s3.endpoint= http://s3.amazonaws.com

```

Migrating Your Filestore

If you have an existing installation of Artifactory, you need to migrate your filestore over to your S3 provider.

There are two ways do this:

- Automatically (recommended)
- Manually

Automatic Filestore Migration (Recommended)

To make sure your filestore migration completes successfully without corrupting files, we recommend configuring Artifactory to do this migration for you automatically. To do so, you need to create a link with the name `_add` in `$ARTIFACTORY_HOME/data/eventual` that points to the `$ARTIFACTORY_HOME/data/filestore` directory.

With this setting, as soon as Artifactory starts up, it will automatically move your complete filestore over to your S3 provider.

Your current filestore will be deleted

The process of moving your filestore to your S3 provider will delete your current filestore. We strongly recommend you do a [complete system backup](#) before doing this migration.

Manual Filestore Migration

To migrate your filestore manually, you need to execute the following steps:

- Stop Artifactory
- Copy the `$ARTIFACTORY_HOME/data/filestore` directory to your S3 object storage to the bucket name and path specified when you [configured Artifactory to use S3](#) (the default values are `artifactory` and `filestore`).
- Start Artifactory

User Plugins

About Plugins

Artifactory Pro allows you to easily extend Artifactory's behavior with your own plugins written in [Groovy](#).

User plugins are used for running user's code in Artifactory. Plugins allow you to perform the following tasks:

- Add scheduled tasks
- Extend Artifactory with your own security realms
- Change resolution rules
- Manipulate downloaded content
- Respond to any storage events on items and properties
- Deploy and query artifacts and metadata
- Perform searches
- Query security information
- Invoke custom commands via REST
- Execute custom promotion logic
- Provide information and strategies for Artifactory's Build Servers Plugins.

During the development phase, you can change plugin source files and have your plugins redeployed on-the-fly. You can even debug the plugin code using your favorite IDE.

Groovy Version

Groovy 2.4 is supported

Page Contents

- [About Plugins](#)
- [Deploying Plugins](#)
 - [Auto Reload](#)
 - [Plugins Lib Directory](#)
- [Writing Plugins](#)
 - [The Artifactory Public API \(PAPI\)](#)
 - [Globally Bound Variables](#)
 - [Plugin Execution Points](#)
 - [Execution Context](#)
- [Plugin Template Source](#)
 - [General Info](#)
 - [Download](#)
 - [Storage](#)
 - [Jobs](#)
 - [Executions](#)
 - [Realms](#)
 - [Build](#)
 - [Promotions](#)
 - [Staging](#)
 - [Replication](#)
- [Controlling Plugin Log Level](#)
- [Sample Plugin](#)

Deploying Plugins

Place your plugin files under `${ARTIFACTORY_HOME} /etc/plugins`.

Artifactory HA Plugins Directory

If you are working with a [High Availability](#) cluster, your plugins should go under:

`${CLUSTER_HOME} /ha-etc/plugins`

Any file name ending with `.groovy` is loaded on startup. You can have multiple plugin files which are loaded in alphabetical order. Callbacks

defined in plugins are called by the order they were loaded.

Auto Reload

By default, plugins are not reloaded after Artifactory has started-up. You can configure Artifactory to automatically detect plugin changes on disk or new plugin files and automatically reload them in runtime (plugin removals are not detected).

To do this, set the number of seconds to check for plugin updates to a number greater than 0, by changing the following property in `ARTIFACTORY_HOME}/etc/artifactory.system.properties`, or by specifying the property with -D to the JVM running Artifactory:

```
artifactory.plugin.scripts.refreshIntervalSecs=0
```

NOTE! that deleting or renaming plugin files while auto-reloading is active is not fully supported and requires an Artifactory restart.

Disabling Plugin Reloading for Production

Ensure plugin auto-reloading is disabled in a production environment.

Plugins Lib Directory

If your plugin requires any external dependencies, you can place them under the `ARTIFACTORY_HOME}/etc/plugins/lib` directory.

Writing Plugins

Artifactory plugins are written as Groovy scripts in regular files and have a simple DSL to wrap users code in closures inside well-known extension points.

Scripts have a couple of helper objects that are globally bound (see the plugin script template).

Naming conventions

Note that Groovy scripts must follow the same [naming conventions](#) as those specified for Java.

The Artifactory Public API (PAPI)

Scripts have access to the full classpath of Artifactory, however, the only API supported for plugins is the [!\[\]\(135f55897915cc5eabb6a0912b4954c8_img.jpg\) Artifactory Public API](#), defined in the `artifactory-papi.jar`.

The `artifactory-papi.jar` can be found under `WEB-INF/lib` folder inside the `artifactory.war`.

Please see the [Plugin Code Template](#) and [Sample Plugin](#) below for more details.

IDE code completion

All major IDEs have good Groovy editing and debugging capabilities.

In order to make your developing experience complete, we provide support for our own DSL for IntelliJ IDEA. IntelliJ IDEA's Groovy DSL script for Artifactory User Plugins can be found in our [GitHub repo](#). Eclipse DSLD file is also available courtesy of James Carnegie.

Globally Bound Variables

Variable Name	Variable Type	Comments
---------------	---------------	----------

<i>log</i>	org.slf4j.Logger	Writes to Artifactory log logger name is the name of the script file
<i>repositories</i>	org.artifactory.repo.Repositories	Allows queries and operations on repositories and artifacts
<i>security</i>	org.artifactory.security.Security	Provides information about current security context, (e.g. current user and her permissions)
<i>searches</i>	org.artifactory.search.Searches	API for searching for artifacts and builds Since 2.3.4
<i>builds</i>	org.artifactory.build.Builds	Allows CRUD operations on builds Since 2.6

Plugins Repository

Enhancing Artifactory with user plugins is community-driven effort.

If you are looking to go beyond Artifactory's out-of-the-box functionality take a look at already contributed plugins on [GitHub](#), you might find what you are thinking about. If not, your contribution is very welcome!

Plugin Execution Points

The following table summarizes the available execution points. For more details about specific plugin look follow the section links.

Plugin Type	Code block name	When executed	Description
Download			
Event Callback (with return values)	altResponse	On any download	Provide an alternative response, by setting a success/error status code value and an optional error message or by setting new values for the inputStream and size context variables (For succeeded resolutions).
	altRemotePath	When reaching out to remote repositories	Provides an alternative download path under the same remote repository, by setting a new value to the path variable.

	altRemoteContent	After fetching content from remote repositories	Provide an alternative download content, by setting new values for the inputStream and size context variables.
	afterDownloadError	After failing during content fetching from remote repositories	Provide an alternative response, by setting a success/error status code value and an optional error message or by setting new values for the inputStream and size context variables (For failed resolutions).
Event Callback (without return value)	beforeRemoteDownload	Before fetching content from remote repositories	Handle before remote download events.
	afterRemoteDownload	After fetching content from remote repositories	Handle after remote download events.
	beforeDownload	On any download	Handle before download events.
	beforeDownloadRequest	On any download	Handle before download request events, executed before Artifactory starts to handle the original client request, useful for intercepting expirable resources (other than the default ones like maven-metadata.xml).
Storage			
Event Callback (without return value)	before/after Create, Delete, Move, Copy, PropertyCreate, PropertyDelete	Before / After selected storage operation	Handle events before and after Create, Delete, Move and Copy operations
Jobs			

Scheduled execution	any valid Groovy (Java) literal as execution name	According to provided interval/delay or cron expression	Job runs are controlled by the provided interval or cron expression, which are mutually exclusive. The actual code to run as part of the job should be part of the job's closure.
Executions			
User-driven execution	any valid Groovy (Java) literal as execution name	By REST call	External executions are invoked via REST requests.
Realms			
Event Callback (without return value)	any valid Groovy (Java) literal as realm name with nested blocks: <pre>authenticate userExists</pre>	During user authentication	Newly added realms are added before any built-in realms (Artifactory internal realm, LDAP, Crowd etc.). User authentication will be attempted against these realms first, by the order they are defined.
Build			
Event Callback (without return value)	beforeSave	Before the build info is saved in Artifactory	Handle before build info save events
	afterSave	After the build info is saved in Artifactory	Handle after build info save events
Promotions			
User or build server driven execution	any valid Groovy (Java) literal as promotion name	By REST call	Promotes integration (a.k.a. snapshot) build to be a release invoking any code associated with it.
Staging Strategy			

build server driven execution	any valid Groovy (Java) literal as staging strategy name	During build server driven staging build configuration	The strategy provides the build server with the following information: <ul style="list-style-type: none"> • How the artifacts in the staged build should be versioned; • How the artifacts in the next integration build should be versioned; • Should the build server create a release branch/tag/stream in VCS and how it should be called; • To which repository in Artifactory the built artifacts should be submitted.
Replication			
Event callback (with return value)	beforeFileReplication	Before file is replicated	Handle before file replication events. File replication can be skipped.
	beforeDirectoryReplication	Before directory is replicated	Handle before directory replication events. Directory replication can be skipped.
	beforeDeleteReplication	Before file/directory is deleted	Handle before file or directory are deleted.
	beforePropertyReplication	Before properties are replicated	Handle properties replication.

Execution Context

The **Download**, **Storage**, **Execution** and **Build** plugin types are executed under the identity of the user request that triggered them.

It is possible to force a block of plugin code to execute under the "system" role, which is not bound to any authorization rules and can therefore perform actions that are otherwise forbidden for the original user.

To run under the "system" role wrap your code with the `asSystem` closure:

```
... someCode ...

asSystem {
    //This code runs as the system role
}

... someOtherCode ...
```

The **Realm** and **Job** plugin types already execute under the "system" role. This cannot be changed.

Plugin Template Source

General Info

▼ General info....

```
/*
 * Copyright (C) 2011 JFrog Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/**
 *
 * Globally bound variables:
 *
 * log (org.slf4j.Logger)
 * repositories (org.artifactory.repo.Repositories)
 * security (org.artifactory.security.Security)
 * searches (org.artifactory.search.Searches) [since: 2.3.4]
 * builds (org.artifactory.build.Builds) [since 2.5.2]
 *
 * ctx (org.artifactory.spring.InternalArtifactoryContext) - NOT A PUBLIC API - FOR
INTERNAL USE ONLY!
 */
```

Download

▼ Handling and manipulating "download" events...

```
download {

    /**
     * Provide an alternative response, by one of the following methods:
```

```

    * (1) Setting a success/error status code value and an optional error message.
    * (2) Provide an alternative download content, by setting new values for the
inputStream and size context variables.
    *
    * Note that, unless specifically handled, checksum requests for altered
responses will return the checksum of the
    * original resource, which may not match the checksum of the alternate response.
    *
    * Will not be called if the response is already committed (e.g. a previous error
occurred).
    * Currently called only for GET requests where the resource was found.
    *
    * Context variables:
    * status (int) - a response status code. Defaults to -1 (unset).
    * message (java.lang.String) - a text message to return in the response body,
replacing the response content.
    *
    *           Defaults to null.
    * inputStream (java.io.InputStream) - a new stream that provides the response
content. Defaults to null.
    * size (long) - the size of the new content (helpful for clients processing the
response). Defaults to -1.
    *
    *
    * Closure parameters:
    * request (org.artifactory.request.Request) - a read-only parameter of the
request.
    * responseRepoPath (org.artifactory.repo.RepoPath) - a read-only parameter of
the response RepoPath (containing the
    *
    *           physical repository the
resource was found in).
    */
    altResponse { request, responseRepoPath ->
}

/**
 * Provides an alternative download path under the same remote repository, by
setting a new value to the path
 * variable.
 *
 * Context variables:
 * path (java.lang.String) - the new path value. Defaults to the
originalRepoPath's path.
 *
 * Closure parameters:
 * repoPath (org.artifactory.repo.RepoPath) - a read-only parameter of the
original request RepoPath.
 */
    altRemotePath { repoPath ->
}

/**
 * Provide an alternative download content, by setting new values for the
inputStream and size context variables.
 *
 * Context variables:
 * inputStream (java.io.InputStream) - a new stream that provides the response
content. Defaults to null.
    * size (long) - the size of the new content (helpful for clients processing the
response). Defaults to -1.

```

```

*
* Closure parameters:
* repoPath (org.artifactory.repo.RepoPath) - a read-only parameter of the
original request RepoPath.
*/
altRemoteContent { repoPath ->
}

/**
 * In case of resolution error provide an alternative response, by setting a
success/error status code value and an optional error message.
 * Will not be called if the response is already committed (e.g. a previous
error occurred).
 * As opposite to altResponse, called only for GET requests during which error
occurred (e.g. 404 - not found, or 409 - conflict).
*
* Context variables:
* status (int) - a response error status code (may be overridden in the
plugin).
* message (java.lang.String) - a response error message (may be overridden in
the plugin).
* inputStream (java.io.InputStream) - a new stream that provides the response
content. Defaults to null.
* size (long) - the size of the new content (helpful for clients processing the
response). Defaults to -1.
*
* Closure parameters:
* request (org.artifactory.request.Request) - a read-only parameter of the
request.
*/
afterDownloadError { request ->
}

/**
 * Handle before remote download events.
*
* Context variables:
* headers (java.util.Map<String, String>) - Map containing the extra headers to
insert into the remote server request
*
* Usage example:
* headers = ["ExtraHeader": "SpecialHeader"]
*
* Note: The following cannot be used as extra headers and Artifactory will
always override them:
* "X-Artifactory-Originated". "Origin-Artifactory", "Accept-Encoding"
*
* Closure parameters:
* request (org.artifactory.request.Request) - a read-only parameter of the
request. [since: 2.3.4]
* repoPath (org.artifactory.repo.RepoPath) - a read-only parameter of the
original request RepoPath.
*/
beforeRemoteDownload { request, repoPath ->
}

/**
 * Handle after remote download events.
*

```


resource was found in).

*/

```
        beforeDownloadRequest { request, repoPath ->
    }
}
```

Storage

Handling and manipulating "storage" events...

If you want to abort an action, you can do that in 'before' methods by throwing a runtime org.artifactory.exception.CancelException with an error message and a proper http error code.

```
storage {

    /**
     * Handle before create events.
     *
     * Closure parameters:
     * item (org.artifactory.fs.ItemInfo) - the original item being created.
     */
    beforeCreate { item ->
    }

    /**
     * Handle after create events.
     *
     * Closure parameters:
     * item (org.artifactory.fs.ItemInfo) - the original item being created.
     */
    afterCreate { item ->
    }

    /**
     * Handle before delete events.
     *
     * Closure parameters:
     * item (org.artifactory.fs.ItemInfo) - the original item being deleted.
     */
    beforeDelete { item ->
    }

    /**
     * Handle after delete events.
     *
     * Closure parameters:
     * item (org.artifactory.fs.ItemInfo) - the original item deleted.
     */
    afterDelete { item ->
    }

    /**
     * Handle before move events.
     *
     * Closure parameters:
     *
     * item (org.artifactory.fs.ItemInfo) - the source item being moved.
     * targetRepoPath (org.artifactory.repo.RepoPath) - the target repoPath for the
     * move.
    
```

```
* properties (org.artifactory.md.Properties) - user specified properties to add to
the item being moved.
*/
beforeMove { item, targetRepoPath, properties ->
}

/**
* Handle after move events.
*
* Closure parameters:
* item (org.artifactory.fs.ItemInfo) - the source item moved.
* targetRepoPath (org.artifactory.repo.RepoPath) - the target repoPath for the
move.
* properties (org.artifactory.md.Properties) - user specified properties to add to
the item being moved.
*/
afterMove { item, targetRepoPath, properties ->
}

/**
* Handle before copy events.
*
* Closure parameters:
* item (org.artifactory.fs.ItemInfo) - the source item being copied.
* targetRepoPath (org.artifactory.repo.RepoPath) - the target repoPath for the
copy.
* properties (org.artifactory.md.Properties) - user specified properties to add to
the item being moved.
*/
beforeCopy { item, targetRepoPath, properties ->
}

/**
* Handle after copy events.
*
* Closure parameters:
* item (org.artifactory.fs.ItemInfo) - the source item copied.
* targetRepoPath (org.artifactory.repo.RepoPath) - the target repoPath for the
copy.
* properties (org.artifactory.md.Properties) - user specified properties to add to
the item being moved.
*/
afterCopy { item, targetRepoPath, properties ->
}

/**
* Handle before property create events.
*
* Closure parameters:
* item (org.artifactory.fs.ItemInfo) - the item on which the property is being
set.
* name (java.lang.String) - the name of the property being set.
* values (java.lang.String[]) - A string array of values being assigned to the
property.
*/
beforePropertyCreate { item, name, values ->
}
/**
* Handle after property create events.

```

```
*
* Closure parameters:
* item (org.artifactory.fs.ItemInfo) - the item on which the property has been
set.
* name (java.lang.String) - the name of the property that has been set.
* values (java.lang.String\[\]) - A string array of values assigned to the property.

*/
afterPropertyCreate { item, name, values ->
}

$$/**$$

* Handle before property delete events.
*
* Closure parameters:
* item (org.artifactory.fs.ItemInfo) - the item from which the property is being
deleted.
* name (java.lang.String) - the name of the property being deleted.
*/
beforePropertyDelete { item, name ->
}

$$/**$$

* Handle after property delete events.
*
* Closure parameters:
* item (org.artifactory.fs.ItemInfo) - the item from which the property has been
deleted.
* name (java.lang.String) - the name of the property that has been deleted.
*/
```

```
    afterPropertyDelete { item, name ->
}
}
```

Jobs

▼ Defining scheduled jobs...

```
jobs {

    /**
     * A job definition.
     * The first value is a unique name for the job.
     * Job runs are controlled by the provided interval or cron expression, which are
     * mutually exclusive.
     * The actual code to run as part of the job should be part of the job's closure.
     *
     * Parameters:
     * delay (long) - An initial delay in milliseconds before the job starts running
     * (not applicable for a cron job).
     * interval (long) - An interval in milliseconds between job runs.
     * cron (java.lang.String) - A valid cron expression used to schedule job runs
     * (see: http://www.quartz-scheduler.org/docs/tutorial/TutorialLesson06.html)
     */

    myJob(interval: 1000, delay: 100) {
    }

    mySecondJob(cron: "0/1 * * * * ?") {
    }
}
```

Executions

▼ Defining external executions...

```
curl -X GET -v -u admin:password
"http://localhost:8080/artifactory/api/plugins/execute/myExecution?params=msg=And+the+result+is:|no1=10|no2=15&async=0"
```

```

executions {

    /**
     * An execution definition.
     * The first value is a unique name for the execution.
     *
     * Context variables:
     * status (int) - a response status code. Defaults to -1 (unset). Not applicable
     for an async execution.
     * message (java.lang.String) - a text message to return in the response body,
     replacing the response content.
     *                                         Defaults to null. Not applicable for an async
     execution.
     *
     * Plugin info annotation parameters:
     * version (java.lang.String) - Closure version. Optional.
     * description (java.lang.String) - Closure description. Optional.
     * httpMethod (java.lang.String, values are GET|PUT|DELETE|POST) - HTTP method
     this closure is going
     *      to be invoked with. Optional (defaults to POST).
     * params (java.util.Map<java.lang.String, java.lang.String>) - Closure default
     parameters. Optional.
     * users (java.util.Set<java.lang.String>) - Users permitted to query this
     plugin for information or invoke it.
     * groups (java.util.Set<java.lang.String>) - Groups permitted to query this
     plugin for information or invoke it.
     *
     * Closure parameters:
     * params (java.util.Map) - An execution takes a read-only key-value map that
     corresponds to the REST request
     *      parameter 'params'. Each entry in the map contains an array of values. This
     is the default closure parameter,
     *      and so if not named it will be "it" in groovy.
     * ResourceStreamHandle body - Enables you to access the full input stream of
     the request body.
     *      This will be considered only if the type ResourceStreamHandle is declared
     in the closure.
    */

    myExecution(version:version, description:description, httpMethod: 'GET',
    users:[], groups:[], params:[:]) { params ->
    }

    execWithBody(version:version, description:description, httpMethod: 'GET',
    users:[], groups:[], params:[:]) { params, ResourceStreamHandle body ->
    }

}

```

Realms

Management of security realms...

Realms defined here are added before any built-in realms (Artifactory internal realm, LDAP, Crowd etc.). User authentication will be attempted against these realms first, by the order they are defined.

```

realms {

    /**
     * A security realm definition.
     * The first value is a unique name for the realm.
     *
     * Closure parameters:
     * autoCreateUsers (boolean) - Whether to automatically create users in
     Artifactory upon successful login. Defaults to
     * true. When false, the user will be transient and his privileges will be
     managed according to permissions defined for auto-join groups.
     * realmPolicy (org.artifactory.security.RealmPolicy): (Optional) - If included
     with value RealmPolicy.ADDITIVE, plugin will be executed only if the user has
     previously been authenticated, and allows enrichment of the authenticated
     * user with additional data.
     * See our public GitHub for an example here:
     https://github.com/JFrogDev/artifactory-user-plugins/blob/master/security/synchronizeLdapGroups/synchronizeLdapGroups.groovy
    */

    myRealm(autoCreateUsers: true, realmPolicy: RealmPolicy.ADDITIVE) {
        /**
         * Implementation should return true/false as the result of the authentication.
         *
         * Context variables:
         * groups (java.lang.String[]) - An array of groups that the authenticated user
         should be associated with (since 3.0.2).
         * user (org.artifactory.security.User) - The authenticated user.
         *
         * Closure parameters:
         * username (java.lang.String) - The username
         * credentials (java.lang.String) - The password
         */
        authenticate { username, credentials ->
            }

        /**
         * Implementation should return true if the user is found in the realm.
         * Closure parameters:
         * username (java.lang.String) - The username
         */
        userExists { username ->
            }
    }
}

```

Build

Handling "Build Info" events...

```
build {

    /**
     * Handle before build info save events
     *
     * Closure parameters:
     * buildRun (org.artifactory.build.DetailedBuildRun) - Build Info model to be
     saved. Partially mutable.
     */
    beforeSave { buildRun ->
    }

    /**
     * Handle after build info save events
     *
     * Closure parameters:
     * buildRun (org.artifactory.build.DetailedBuildRun) - Build Info that was saved.
     Partially mutable.
     */
    afterSave { buildRun ->
    }
}
```

Promotions

- ▼ [Defining REST executable build promotion operations...](#)

```
promotions {

    /**
     * A REST executable build promotion definition.
     *
     * Context variables:
     * status (int) - a response status code. Defaults to -1 (unset).
     * message (java.lang.String) - a text message to return in the response body,
     replacing the response content. Defaults to null.
     *
     * Plugin info annotation parameters:
     * version (java.lang.String) - Closure version. Optional.
     * description (java.lang.String) - Closure description. Optional.
     * params (java.util.Map<java.lang.String, java.lang.String>) - Closure
     parameters. Optional.
     * users (java.util.Set<java.lang.String>) - Users permitted to query this plugin
     for information or invoke it.
     * groups (java.util.Set<java.lang.String>) - Groups permitted to query this
     plugin for information or invoke it.
     *
     * Closure parameters:
     * buildName (java.lang.String) - The build name specified in the REST request.
     * buildNumber (java.lang.String) - The build number specified in the REST
     request.
     * params (java.util.Map<java.lang.String, java.util.List<java.lang.String>>) -
     The parameters specified in the REST request.
    */
    promotionName(version, description, users, groups, params) { buildName,
    buildNumber, params ->
    }
}
```

Staging

▼ Defining REST retrievable build staging strategy construction...

```

/**
 * Set of staging strategy definitions to be used by the build server during
staging process.
 * The strategy provides the build server with the following information:
 * 1. How the artifacts in the staged build should be versioned;
 * 2. How the artifacts in the next integration build should be versioned;
 * 3. Should the build server create a release branch/tag/stream in VCS and how it
should be called;
 * 4. To which repository in Artifactory the built artifacts should be submitted.
 *
 * This user plugin is called by the build server using REST call.
 */
staging {

 /**
 * A build staging strategy definition.
 *
 * Closure delegate:
 * org.artifactory.build.staging.BuildStagingStrategy - The strategy that's to be
returned.
 *
 * Plugin info annotation parameters:
 * version (java.lang.String) - Closure version. Optional.
 * description (java.lang.String) - Closure description. Optional.
 * params (java.util.Map<java.lang.String, java.lang.String>) - Closure parameters.
Optional.
 * users (java.util.Set<java.lang.String>) - Users permitted to query this plugin
for information or invoke it.
 * groups (java.util.Set<java.lang.String>) - Groups permitted to query this plugin
for information or invoke it.
 *
 * Closure parameters:
 * buildName (java.lang.String) - The build name specified in the REST request.
 * params (java.util.Map<java.lang.String, java.util.List<java.lang.String>>) - The
parameters specified in the REST request.
 */
strategyName(version, description, users, groups, params) { buildName, params ->
}
}

```

Replication

Handling and filtering replication events (since version 3.0.4)...

```
replication {
    /**
     * Handle before file replication events.
     *
     * Context variables:
     * skip (boolean) - whether to skip replication for the current item. Defaults to false. Set to false to skip replication.
     *
     * Closure parameters:
     * localRepoPath (org.artifactory.repo.RepoPath) - the repoPath of the item on the local Artifactory server.
     */
    beforeFileReplication { localRepoPath ->
    }
    /**
     * Handle before directory replication events.
     *
     * Context variables:
     * replicate (int) - whether to replicate the current item. Defaults to true. Set to false to skip replication.
     *
     * Closure parameters:
     * localRepoPath (org.artifactory.repo.RepoPath) - the repoPath of the item on the local Artifactory server.
     */
    beforeDirectoryReplication { localRepoPath ->
    }
    /**
     * Handle before delete replication events.
     *
     * Context variables:
     * replicate (int) - whether to replicate the current item. Defaults to true. Set to false to skip replication.
     *
     * Closure parameters:
     * localRepoPath (org.artifactory.repo.RepoPath) - the repoPath of the item on the local Artifactory server.
     */
    beforeDeleteReplication { localRepoPath ->
    }
    /**
     * Handle before property replication events.
     *
     * Context variables:
     * replicate (int) - whether to replicate the current item. Defaults to true. Set to false to skip replication.
     *
     * Closure parameters:
     * localRepoPath (org.artifactory.repo.RepoPath) - the repoPath of the item on the local Artifactory server.
     */
    beforePropertyReplication { localRepoPath ->
    }
}
```

Controlling Plugin Log Level

The default log level for user plugins is "warn". To change a plugin log level, add the following to \${ARTIFACTORY_HOME}/etc/logback.xml:

```
<logger name="my-plugin">
  <level value="info"/>
</logger>
```

The logger name is the name of the plugin file without the ".groovy" extension (in the example above the plugin file name is `my-plugin.groovy`). The logging levels can be either error, warn, info, debug or trace.

Sample Plugin

Sample plugin is [available to download](#).

NuGet Repositories

Overview

From version 2.5, Artifactory provides complete support for NuGet repositories on top of Artifactory's [existing support](#) for advanced artifact management.

Artifactory support for NuGet provides:

1. The ability to provision NuGet packages from Artifactory to NuGet clients from all repository types
2. Metadata calculation for NuGet packages hosted in Artifactory's local repositories
3. The ability to define proxies and caches to access Remote NuGet repositories
4. Multiple NuGet repository aggregation through virtual repositories
5. APIs to deploy or remove packages that are compatible with [NuGet Package Manager Visual Studio extension](#) and the [NuGet Command Line Tool](#)
6. Debugging NuGet packages directly using Artifactory as a [Microsoft Symbol Server](#)

Metadata updates

NuGet metadata is automatically calculated and updated when adding, removing, copying or moving NuGet packages. The calculation is only invoked after a package-related action is completed.

It is asynchronous and its performance depends on the overall system load, therefore it may sometimes take up to 30 seconds to complete.

You can also invoke metadata calculation on the entire repository by selecting "Reindex Packages".

Page Contents

- Overview
- Configuration
 - Local Repositories
 - Local Repository Layout
 - Publishing to a Local Repository
 - Remote Repositories
 - Virtual Repositories
- Accessing NuGet Repositories from Visual Studio
- Using the NuGet

- Command Line
- NuGet API Key
- Authentication
- Anonymous Access to NuGet Repositories
 - Working Without Anonymous Access
 - Allowing Anonymous Access
- Viewing Individual NuGet Package Information
- Watch the Screencast

Read More

- Microsoft Symbol Server

Configuration

Local Repositories

To create a local repository for which Artifactory will calculate NuGet package metadata, set **NuGet** to be the **Package Type**.

New Local Repository

Basic

Package Type *



NuGet

Local Repository Layout

To support a more manageable repository layout, you may store NuGet packages inside folders that correspond to the package structure.

Artifactory will find your packages by performing a property search so the folder hierarchy does not impact performance.

To use a hierarchical layout for your repository you should define a [Custom Layout](#). This way, different maintenance features like [Version Cleanup](#) will work correctly with NuGet packages.

Placing packages to match your repository layout

Defining a [Custom Layout](#) for your repository does not force you to place your packages in the corresponding structure, however it is recommended to do so since it allows Artifactory to perform different maintenance tasks such as [Version Cleanup](#) automatically.

It is up to the developer to correctly deploy packages into the corresponding folder. From NuGet 2.5 you can push packages into a folder source as follows:

```
nuget push mypackage.1.0.0.nupkg -Source  
http://10.0.0.14:8081/artifactory/api/nuget/nuget-local/path/to/folder
```

Below is an example of a [Custom Layout](#) named nuget-default.

Edit nuget-default 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

Result

Organization: NHibernate

Module: NHibernate

Base Revision: 3.3.3

Folder Integration Revision:

File Integration Revision: CR1

Classifier:

Extension:

Type:

In this example, the three fields that are mandatory for module identification are:

- Organization = "orgPath"
- Module = "module"
- Base Revision ("baseRev") is not a part of the layout hierarchy in this example, but it is included here as one of the required fields.

You can configure this Custom Layout as displayed in the image above, or simply copy the below code snippet into the relevant section in your Global Configuration Descriptor:

```

<repoLayout>
  <name>nuget-default</name>

  <artifactPathPattern>[orgPath]/[module]/[module].[baseRev](-[fileItegRev]).[ext]</artifactPathPattern>
    <distinctiveDescriptorPathPattern>false</distinctiveDescriptorPathPattern>
    <folderIntegrationRevisionRegExp>.*</folderIntegrationRevisionRegExp>
    <fileIntegrationRevisionRegExp>.*</fileIntegrationRevisionRegExp>
</repoLayout>

```

Since the package layout is in a corresponding folder hierarchy, the Artifactory Version Cleanup tool correctly detects previously installed versions.

Group ID	Version	Directories Count
Microsoft	Microsoft.AspNet.Mvc.5.0.0	1
Microsoft	Microsoft.AspNet.Mvc.5.2.0	1

Publishing to a Local Repository

When a NuGet repository is selected in the **Artifacts** module Tree Browser, click **Set Me Up** to display the code snippets you can use to configure Visual Studio or your NuGet client to use the selected repository to publish or resolve artifacts.

Set Me Up



Tool

NuGet

Repository

nuget-layout-local

Deploy

To support more manageable layouts, and additional features such as cleanup, NuGet repositories support custom layouts. You need to make sure that you push the package to a layout which matches the target repository layout.

```
1 nuget push <PACKAGE> -Source http://10.100.1.110:8081/artifactory/api/nuget/nuget-layout-local/<PATH_TO_FOLDER>
```

You can also add Artifactory repository as a gallery by running the following command

```
1 nuget sources Add -Name Artifactory -Source http://10.100.1.110:8081/artifactory/api/nuget/nuget-layout-local
```

This will enable you to use Artifactory with NuGet from command line for both pull and push.

Resolve

NuGet repositories must be prefixed with api/nuget in the path When configuring Visual Studio to access a NuGet repository through Artifactory, the repository URL must be prefixed with api/nuget in the path.

```
1 http://10.100.1.110:8081/artifactory/api/nuget/nuget-layout-local
```



Remote Repositories

When working with remote NuGet repositories, your Artifactory configuration depends on how the remote repositories are set up.

Different NuGet server implementations may provide package resources on different paths, therefore the feed and download resource locations in Artifactory are customizable when proxying a remote NuGet repository.

Here are some examples:

- The **NuGet gallery** exposes its feed resource at <https://www.nuget.org/api/v2/Packages> and its download resource at <https://www.nuget.org/api/v2/package>.
Therefore, to define this as a new repository you should set the repository **URL** to <https://www.nuget.org>, its **Feed Context Path** to `api/v2` and the **Download Context Path** to `api/v2/package`.

New Remote Repository

Repository Key *	URL *
nuget-gallery	https://www.nuget.org
General	
Repository Layout	Remote Layout Mapping
nuget-default	Select Remote Layout
NuGet Settings	
NuGet Download Context Path *	NuGet Feed Context Path
<code>api/v2/package</code>	<code>api/v2</code>

- The **NuGet .Server** module exposes its feed resource at <http://host:port/nuget/Packages> and its download resource at <http://host:port/api/v2/package>.
To define this as a new repository you should set the repository **URL** to `http://host:port`, its **Feed Context Path** to `nuget` and its **Download Context Path** to `api/v2/package`.

ad Context Path to `api/v2/package`.

- Another Artifactory repository exposes its feed resource at <http://host:port/artifactory/api/nuget/repoKey/Packages> and its download resource at <http://host:port/artifactory/api/nuget/repoKey/Download>.

To define this as a new repository you should set the repository URL to <http://host:port/artifactory/api/nuget/repoKey>, its **Feed Context Path** should be left empty and its **Download Context Path** to Download.

Using a proxy server

If you are accessing NuGet Gallery through a proxy server you need to define the following two URLs in the proxy's white list:

1. *.nuget.org
2. az320820.vo.msecnd.net (our current CDN domain)

For more details please refer to [Allowing Access to NuGet.org Through Firewalls](#).

Virtual Repositories

A Virtual Repository defined in Artifactory aggregates packages from both local and remote repositories.

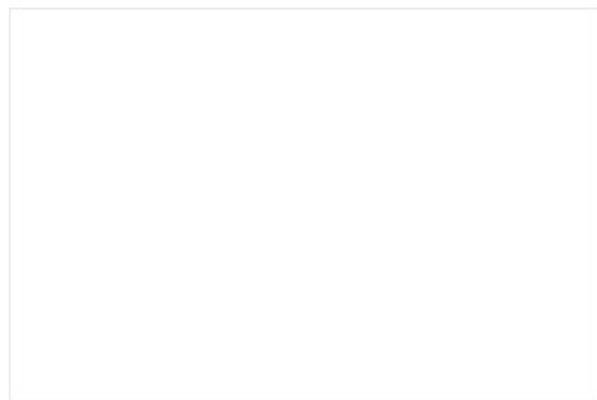
This allows you to access both locally hosted NuGet packages and remote proxied NuGet libraries from a single URL defined for the virtual repository.

To create a virtual Bower repository set **NuGet** to be its **Package Type**, and select the underlying local and remote NuGet repositories to include under the **Repositories** section.

Repositories

Filter...

Available Repositories



Selected Repositories

	nuget-layout-local	
	nuget-local	
	nuget	
	nuget-gallery	

Included Repositories

nuget-layout-local
nuget-local
nuget
nuget-gallery

Accessing NuGet Repositories from Visual Studio

NuGet repositories must be prefixed with `api/nuget` in the path

When configuring Visual Studio to access a NuGet repository through Artifactory, the repository URL must be prefixed with **api/nuget** in the path.

For example, if you are using Artifactory standalone or as a local service, you would configure Visual Studio using the following URL:

`http://localhost:8081/artifactory/api/nuget/<repository key>`

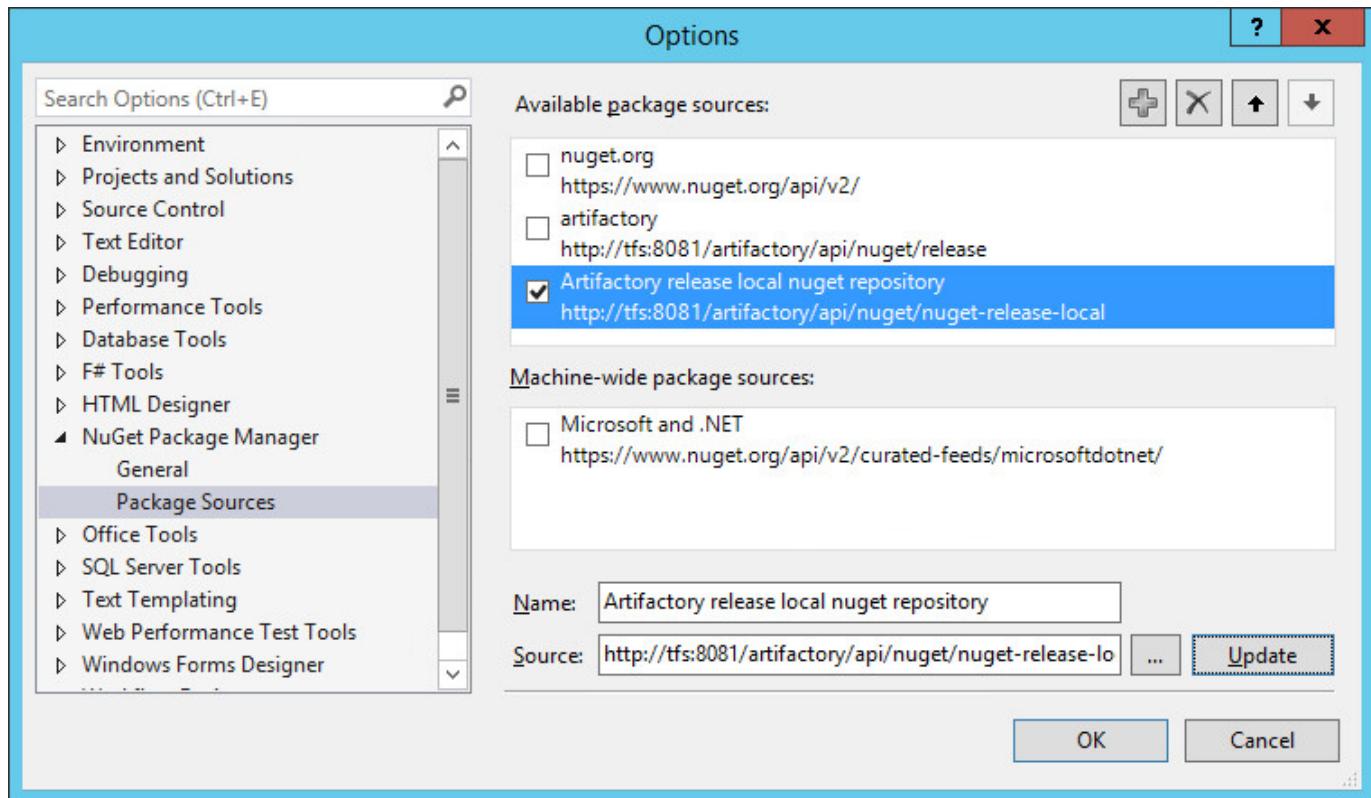
Or, if you are using Artifactory Online the URL would be:

`https://<server name>.artifactoryonline.com/<server name>/api/nuget/<repository key>`

Artifactory exposes its NuGet resources via the REST API at the following URL: `http://localhost:8081/artifactory/api/nuget/<repository key>`.

This URL handles all NuGet related requests (search, download, upload, delete) and supports both V1 and V2 requests.

To configure the NuGet Visual Studio Extension to use Artifactory, check the corresponding repositories in the "Options" window: (You can access Options from the Tools menu).



Using the NuGet Command Line

NuGet repositories must be prefixed with api/nuget in the path

When using the NuGet command line to access a repository through Artifactory, the repository URL must be prefixed with **api/nuget** in the path. This applies to all NuGet commands including `nuget install` and `nuget push`.

For example, if you are using Artifactory standalone or as a local service, you would access your NuGet repositories using the following URL:

`http://localhost:8081/artifactory/api/nuget/<repository key>`

Or, if you are using Artifactory Online the URL would be:

`https://<server name>.artifactoryonline.com/<server name>/api/nuget/<repository key>`

To use the NuGet Command Line tool:

1. Download NuGet.exe
2. Place it in a well known location in your file system such as c:\utils
3. Make sure that NuGet.exe is in your path

For complete information on how to use the NuGet Command Line tool please refer to the [NuGet Docs Command Line Reference](#).

First configure a new source URL pointing to Artifactory:

```
nuget sources Add -Name Artifactory -Source  
http://localhost:8081/artifactory/api/nuget/<repository key>
```

NuGet API Key Authentication

NuGet tools require that sensitive operations such as push and delete are authenticated with the server using an `apikey`. The API key you should use is in the form of `username:password`, where the password can be either clear-text or [encrypted](#).

Set your API key using the NuGet Command Line Interface:

```
nuget setapikey admin:password -Source Artifactory
```

Now you can perform operations against the newly added server. For example:

```
nuget list -Source Artifactory  
  
nuget install log4net -Source Artifactory
```

Anonymous Access to NuGet Repositories

By default, Artifactory allows anonymous access to NuGet repositories. This is defined under [Security | General Configuration](#). For details please refer to [Allow Anonymous Access](#).

Working Without Anonymous Access

In order to be able to trace how users interact with your repositories we recommend that you uncheck the **Allow Anonymous Access** setting described above. This means that users will be required to enter their user name and password when using their NuGet clients.

You can configure your NuGet client to require a username and password using the following command:

```
nuget sources update -Name Artifactory -UserName admin -Password password
```

You can verify that your setting has taken effect by checking that the following segment appears in your `%APPDATA%\NuGet\NuGet.Config` file:

```
<packageSourceCredentials>  
  <Artifactory>  
    <add key="Username" value="admin" />  
    <add key="Password" value="...encrypted password..." />  
  </Artifactory>  
</packageSourceCredentials>
```

NuGet.Config file can also be placed in your project directory, for further information please refer to [NuGet Configuration File](#)

Allowing Anonymous Access

Artifactory supports NuGet repositories with **Allow Anonymous Access** enabled.

When **Allow Anonymous Access** is enabled, Artifactory will not query the NuGet client for authentication parameters by default, so you need to indicate to Artifactory to request authentication parameters in a different way.

You can override the default behavior by setting the **Force Authentication** checkbox in the New or Edit Repository dialog.

The screenshot shows the 'NuGet Settings' dialog. At the top, there is a field for 'Max Unique Snapshots' with a value of '0'. Below it is a checkbox labeled 'Force Authentication' which is checked and highlighted with a red border. A question mark icon is located next to the checkbox.

When set, Artifactory will first request authentication parameters from the NuGet client before trying to access this repository.

Viewing Individual NuGet Package Information

You can view all the metadata annotating a NuGet package by choosing the NuPkg file in Artifactory's tree browser and selecting the **NuPkg Info** tab:

The screenshot shows the details page for the NuGet package 'AspNet.Microsoft.AspNet.Mvc.5.0.0.nupkg'. The top navigation bar includes tabs for General, NuPkg Info (which is selected and highlighted in green), Effective Permissions, Properties, Watchers, Builds, and Governance. On the far right, there are 'Download' and 'Actions' buttons. The main content area is divided into sections: 'Package Info', 'Description', and 'Dependencies'. The 'Package Info' section contains detailed metadata such as ID, Title, Version, Authors, Owners, License URL, Languages, and Release Notes. The 'Description' section provides a brief overview of the package. The 'Dependencies' section lists the packages this one depends on, including Microsoft.AspNet.WebPages and Microsoft.AspNet.Razor, both at version 3.0.0. A pagination control at the bottom right indicates 'page 1 of 1'.

Watch the Screencast

Microsoft Symbol Server

Overview

Microsoft Symbol Server is a Windows technology used to obtain debugging information (symbols) needed in order to debug an application with various Microsoft tools. Symbol files (which are .pdb files) provide a footprint of the functions that are contained in executable files and dynamic-link libraries (DLLs), and can present a roadmap of the function calls that lead to the point of failure.

A Symbol Server stores the .PDB files and binaries for all your public builds. These are used to enable you to debug any crash or problem that is reported for one of your stored builds. Both Visual Studio and WinDBG know how to access Symbol Servers, and if the binary you are debugging is from a public build, the debugger will get the matching PDB file automatically.

The TFS 2010 build server includes built-in build tasks to index source files and copy symbol files to your Symbol Server automatically as part of your build.

Page Contents

- Overview
- Using Artifactory as Your Symbol Server
 - Configuring Your Debugger
 - Configuring Your Build
 - Configure Repositories in Artifactory
 - Install the Artifactory Symbol Server Plugin
 - Configure IIS
 - Configuring Visual Studio

Using Artifactory as Your Symbol Server

Configuring your system to use Artifactory as your Symbol Server requires the following main steps:

1. Configure your debugger
2. Configure your build
3. Configure repositories in Artifactory
4. Install the Artifactory Symbol Server Plugin
5. Configure IIS
6. Configure Visual Studio

Configuring Your Debugger

To enable you to step into and debug your source files, your debugger needs the corresponding .pdb files and searches for them in the following order until they are found:

1. The directory from which your binary was loaded
2. The hard-coded build directory specified in the **Debug Directories** entry of your portable executable (PE) file
3. Your Symbol Server cache directory (assuming you have a Symbol Server set up)
4. The Symbol Server itself

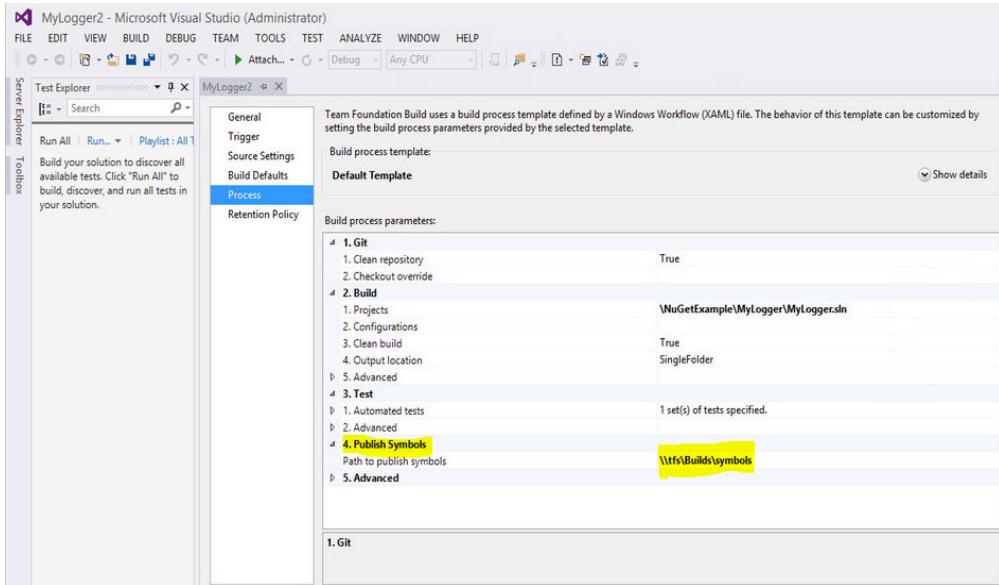
So to fully support this search order, you need to specify the Symbol Server URL in your debugger.

Under **Tools | Options | Debugging | Symbols** enter <http://msdl.microsoft.com/download/symbols>

Configuring Your Build

You need to configure your build machine to publish your .pdb files into a known directory which is later used in your IIS configuration.

Assuming you are using TFS, and want to publish your .pdf files into a directory called *Builds/symbols*, your build definition would look something like the below:



Configure Repositories in Artifactory

Create the following repositories:

Repository name	Description/Instructions
microsoft-symbols-IIS	<p>A Remote repository.</p> <p>Set the repository URL to point to the virtual directory configured in your IIS below. (For the example on this page we will use http://localhost/symbols)</p>
microsoft-symbols	<p>A Remote repository.</p> <p>Set the repository URL to point to the Microsoft Symbol Server URL: http://msdl.microsoft.com/download/symbols.</p>
symbols	<p>A virtual repository.</p> <p>Configure this repository to aggregates the other two repositories, resolving from microsoft-symbols-IIS first. Once configured, this repository will aggregate all the NuGet packages with symbol (.pdb) files.</p> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p>Make this a NuGet repository Be sure to specify NuGet as the Package Type for this repository</p> </div>

Install the Artifactory Symbol Server Plugin

The Artifactory Symbol Server Plugin listens for requests for symbol files and then redirects them to the Microsoft Symbol Server.

Download the Artifactory Symbol Server Plugin from [GitHub](#) and install it in your `$ARTIFACTORY_HOME/etc/plugin` directory.

Check your system properties

To ensure that the plugin is loaded, check that your `artifactory.plugin.scripts.refreshIntervalSecs` system property is not 0.

If you do modify this system property, you need to restart Artifactory for this modification to take effect.

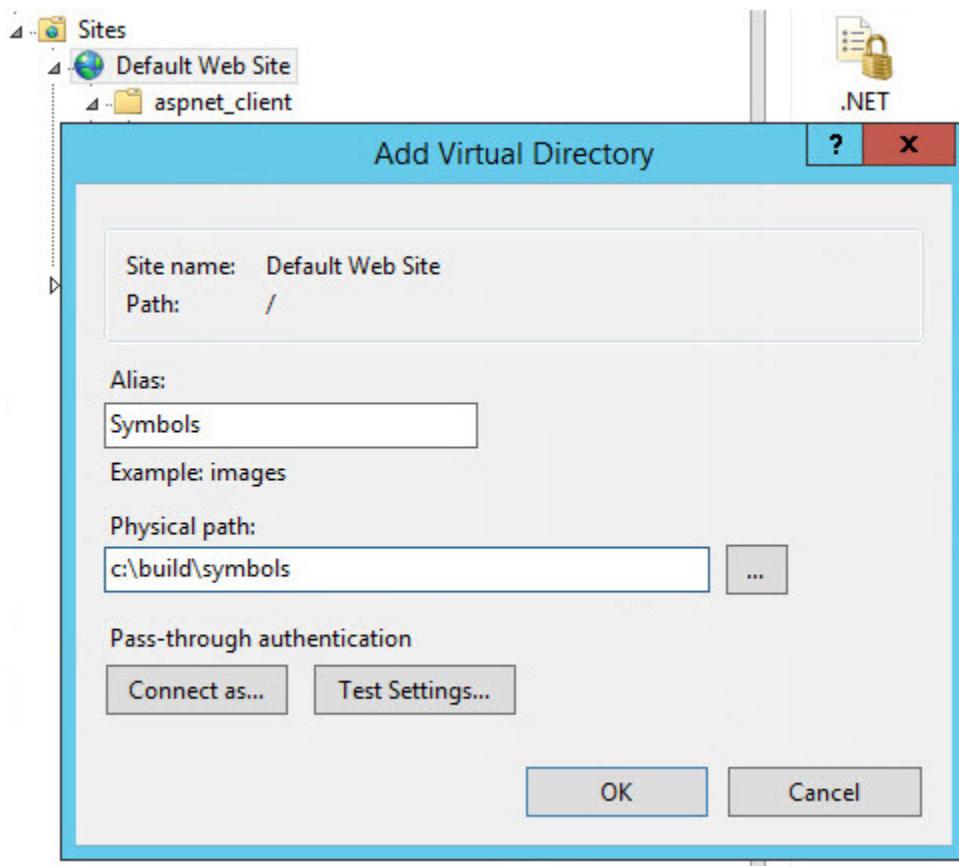
Configure IIS

To configure your Internet Information Services (IIS) machine, you need to

- Add a virtual directory on which your `.pdb` files reside
- Define a MIME type to be associated with `.pdb` files.
- Enable directory browsing.

To add a virtual directory on your IIS, execute the following steps:

- Under **Control Panel | System and Security | Administrative Tools | Internet Information Service(IIS) Manager**, right click **Default Website** and select **Add Virtual Directory**.
- Set **Physical path** to `C:\build\symbols`

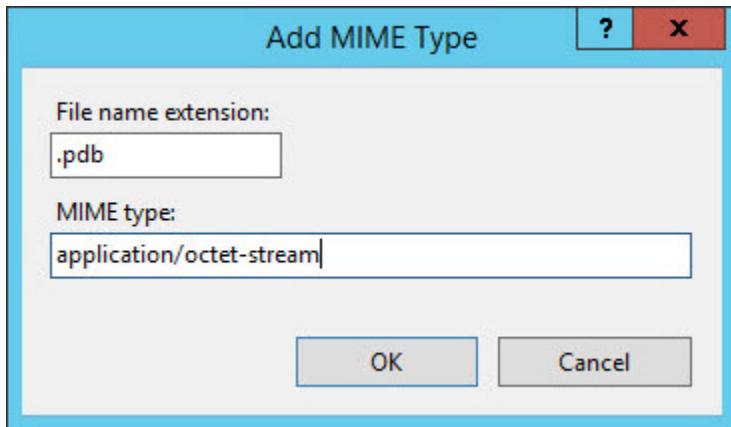


To define a MIME type so that your IIS associates the `.pdb` file extension with Symbol files, execute the following steps:

- Right click the MIME Type symbol and select **Open Feature**.
- Click **Add...** on the right side of the window fill in the fields as follows:

Field	Value
File name extension	.pdb

MIME type	application/octet-stream
-----------	--------------------------



Configuring Visual Studio

Before you configure Visual Studio, you need to remove the symbol cache located under `C:\Users\Administrator\AppData\Local\Temp\1\SymbolCache`

Once you have removed the symbol cache, you need to change the location of the symbol (.pdb) file. Under **Tools | Options | Debugging | Symbols** add a new symbol server pointing to the **symbols** virtual directory you defined in Artifactory above.

This should be `http://tfs:::8081/artifactory/symbols`

Note that there is no way to set the path directly to Artifactory since the symbol server cannot take a URL as a path.

Npm Repositories

Overview

Artifactory fully supports npm repositories on top of Artifactory's existing support for advanced artifact management.

Artifactory support for npm provides:

1. The ability to provision npm packages from Artifactory to the npm command line tool from all repository types
2. Calculation of Metadata for npm packages hosted in Artifactory's local repositories
3. Access to remote npm registries (such as <https://registry.npmjs.org>) through Remote

- Repositories which provide the usual proxy and caching functionality
- 4. The ability to access multiple npm registries from a single URL by aggregating them under a [Virtual Repository](#). This overcomes the limitation of the npm client which can only access a single registry at a time.
 - 5. Compatibility with the [npm command line tool](#) to deploy and remove packages and more.
 - 6. Support for [flexible npm repository layouts](#) that allow you to organize your npm packages and assign access privileges according to projects or development teams.

Npm version support

Artifactory supports NPM version 1.4.3 and above.

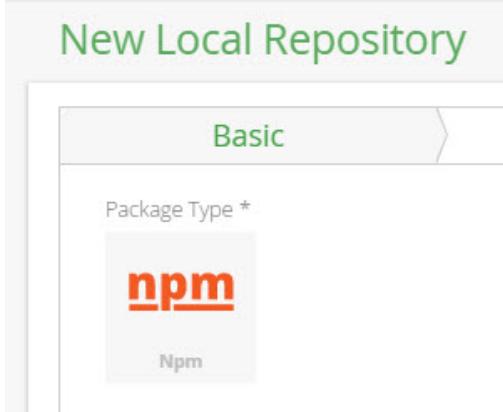
Page Contents

- Overview
- Configuration
 - Local Repositories
 - Remote Repositories
 - Virtual Repositories
 - Advanced Configuration
- Using the Npm Command Line
- Npm Publish (Deploying Packages)
 - Setting Your Credentials
 - Deploying Your Packages
- Working with Artifactory without Anonymous Access
- Npm Search
- Cleaning Up the Local Npm Cache
- Npm Scope Packages
 - Configuring the npm Client for a Scope Registry
 - Using Login Credentials
 - Using OAuth Credentials
- Automatically Rewriting External Dependencies
- Viewing Individual Npm Package Information

Configuration

Local Repositories

To enable calculation of npm package metadata in local repositories, set the **Package Type** to **npm** when you create the repository:



The screenshot shows the 'New Local Repository' dialog. At the top, it says 'New Local Repository'. Below that is a tabs section with 'Basic' selected. In the 'Package Type *' field, the 'npm' option is chosen, indicated by a red 'npm' logo and the word 'Npm' below it.

Repository Layout

Artifactory allows you to define any layout for your npm repositories. In order to upload packages according to your custom layout, you need to package your npm files using `npm pack`.

This creates the .tgz file for your package which you can then upload to any path within your local npm repository.

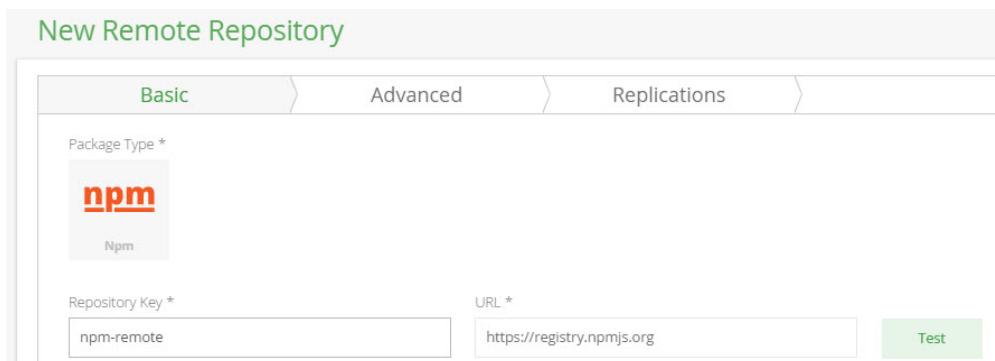
Remote Repositories

A **Remote Repository** defined in Artifactory serves as a caching proxy for a registry managed at a remote URL such as <https://registry.npmjs.org>.

Artifacts (such as tgz files) requested from a remote repository are cached on demand. You can remove downloaded artifacts from the remote repository cache, however, you can not manually deploy artifacts to a remote npm repository.

To define a remote repository to proxy a remote npm registry follow the steps below:

1. In the **Admin** module, under **Repositories | Remote**, click "New".
2. In the New Repository dialog, set the **Package Type** to **npm**, set the **Repository Key** value, and specify the URL to the remote registry in the **URL** field as displayed below



The screenshot shows the 'New Remote Repository' dialog. At the top, it says 'New Remote Repository'. Below that is a tabs section with 'Basic' selected. In the 'Package Type *' field, the 'npm' option is chosen. In the 'Repository Key *' field, 'npm-remote' is entered. In the 'URL *' field, 'https://registry.npmjs.org' is entered. To the right of the URL field is a green 'Test' button.

3. Click "Save & Finish"

Virtual Repositories

A **Virtual Repository** defined in Artifactory aggregates packages from both local and remote repositories.

This allows you to access both locally hosted npm packages and remote proxied npm registries from a single URL defined for the virtual repository.

To define a virtual npm repository, create a [virtual repository](#), set the **Package Type** to be **npm**, and select the underlying local and remote npm repositories to include in the **Basic** settings tab.

Repositories

Filter...

Available Repositories		Selected Repositories	
<input type="checkbox"/> mytestnpm-local	<input type="checkbox"/> npm	<input type="checkbox"/> npm-local	<input type="checkbox"/> npm-remote
<<		>>	
<		>	
>		>>	

Included Repositories

npm-local
npm-remote

Click "Save & Finish" to create the repository.

Advanced Configuration

Edit npm-virtual Repository

Basic Advanced

Artifactory Requests Can Retrieve Remote Artifacts [\(?\)](#)

External Dependency Rewrite

Enable Dependency Rewrite

Remote Repository For Cache

npm-remote

Patterns Whitelist [\(?\)](#)

New Pattern

Add

/github.com/

The fields under **External Dependency Rewrite** are connected to automatically rewriting external dependencies for npm packages that need them.

Enable Dependency Rewrite	When checked, automatically rewriting external dependencies is enabled.
Remote Repository For Cache	The remote repository aggregated by this virtual repository in which the external dependency will be cached.
Patterns Whitelist	<p>A white list of Ant-style path expressions that specify where external dependencies may be downloaded from. By default, this is set to <code>**</code> which means that dependencies may be downloaded from any external source.</p> <p>For example, if you wish to limit external dependencies to only be downloaded from <code>github.com</code>, you should add <code>**/github.com/**</code> (and remove the default <code>**</code> expression).</p>

Using the Npm Command Line

Npm repositories must be prefixed with api/npm in the path

When accessing an npm repository through Artifactory, the repository URL must be prefixed with **api/npm** in the path. This applies to all npm commands including `npm install` and `npm publish`.

For example, if you are using Artifactory standalone or as a local service, you would access your npm repositories using the following URL:

```
http://localhost:8081/artifactory/api/npm/<repository key>
```

Or, if you are using Artifactory Online the URL would be:

```
https://<server name>.artifactoryonline.com/<server name>/api/npm/<repository key>
```

To use the npm command line you need to make sure npm is installed. Npm is included as an integral part of recent versions of [Node.js](#).

Please refer to [Installing Node.js via package manager](#) on GitHub or the [npm README page](#).

Once you have created your npm repository, you can select it in the Tree Browser and click **Set Me Up** to get code snippets you can use to change your npm registry URL, deploy and resolve packages using the npm command line tool.

Set Me Up

X

Tool

npm

Repository

npm-local

Deploy

To deploy your package to an Artifactory repository you can either add the following to the package.json file

```
1 "publishConfig": {"registry": "http://10.100.1.110:8081/artifactory/api/npm/npm-local"}
```

Or provide the local repository to the npm publish command

```
1 npm publish --registry http://10.100.1.110:8081/artifactory/api/npm/npm-local
```

Make sure you configured authentication in your .npmrc file (see Resolve Settings)

Resolve

Run the following command to replace the default npm registry with an Artifactory repository

```
1 npm config set registry http://10.100.1.110:8081/artifactory/api/npm/npm-local
```

The npm command line tool requires that sensitive operations, such as publish, are authenticated with the server using basic HTTP authentication. To support authentication you need to edit your .npmrc file and enter the following:

To replace the default registry with a URL pointing to an npm repository in Artifactory (the example below uses a repository with the key `npm-repo`):

Replacing the default registry

```
npm config set registry http://localhost:8081/artifactory/api/npm/npm-repo
```

We recommend referencing a [Virtual Repository URL](#) as a registry. This gives you the flexibility to reconfigure and aggregate other external sources and local repositories of npm packages you deployed.

Note that If you do this, you need to use the `--registry` parameter to specify the local repository into which you are publishing your package when using the `npm publish` command.

Once the npm command line tool is configured, every `npm install` command will fetch packages from the npm repository specified above. For example:

```
$ npm install request
npm http GET http://localhost:8081/artifactory/api/npm/npm-repo/request
npm http 200 http://localhost:8081/artifactory/api/npm/npm-repo/request
npm http GET
http://localhost:8081/artifactory/api/npm/npm-repo/request/-/request-2.33.0.tgz
npm http 200
http://localhost:8081/artifactory/api/npm/npm-repo/request/-/request-2.33.0.tgz
```

Npm Publish (Deploying Packages)

Setting Your Credentials

The npm command line tool requires that sensitive operations, such as `publish`, are authenticated with the server using basic HTTP authentication.

To support authentication you need to edit your `.npmrc` file and enter the following:

- Your Artifactory username and password (formatted `username:password`) as `Base64` encoded strings
- Your email address (`npm publish` will not work if your email is not specified in `.npmrc`)
- You need to set `always-auth = true`

Getting `.npmrc` entries directly from Artifactory

You can use the following command to get these strings directly from Artifactory:

```
$ curl -uadmin:password
"http://localhost:8081/artifactory/api/npm/auth"
_auth =
YWRtaW46e0RFU2VkJX1uOFRaaxh1Y0t3bHN4c2RCTVIwNjF3PT0=
email = myemail@email.com
always-auth = true
```

`.npmrc` file location

Windows: `%userprofile%\ .npmrc`

Linux: `~/.npmrc`

Currently not supported

Artifactory does not support the `npm adduser` command, therefore to publish packages you need to ensure that you have previously created a user on Artifactory.

Note also that the `-tag` option for `npm publish` is not currently supported.

Deploying Your Packages

There are two ways to deploy packages to a local repository:

- Edit your `package.json` file and add a `publishConfig` section to a local repository:
`"publishConfig": {"registry": "http://localhost:8081/artifactory/api/npm/npm-local"}`
- Provide a local repository to the `npm publish` command:
`npm publish --registry http://localhost:8081/artifactory/api/npm/npm-local`

Working with Artifactory without Anonymous Access

By default, Artifactory allows anonymous access to npm repositories. This is defined in the **Admin** module under **Security | General**. For details please refer to [Allow Anonymous Access](#).

If you want to be able to trace how users interact with your repositories you need to uncheck the [Allow Anonymous Access](#) setting. This means that users will be required to enter their username and password as described in [Setting Your Credentials](#) above.

Npm Search

Artifactory supports a variety of ways to search of artifacts. For details please refer to [Searching Artifacts](#).

Artifactory also supports `npm search [search terms ...]`, however, packages may not be available immediately after being published for the following reasons:

When publishing a package to a local repository, Artifactory calculates the search index asynchronously and will wait for a "quiet period" to lapse before indexing the newly published package.

Since a virtual repository may contain local repositories, a newly published package may not be available immediately for the same reason.

You can specify the indexing "quiet period" (time since the package was published) by setting the following system properties (in `$ARTIFACTORY_HOME/etc/artifactory.system.properties`) .

```
artifactory.npm.index.quietPeriodSecs=60  
artifactory.npm.index.cycleSecs=60
```

In the case of remote repositories, a new package will only be found once Artifactory checks for it according to the [Retrieval Cache Period](#) setting.

Artifactory annotates each deployed or cached npm package with two properties: `npm.name` and `npm.version`

You can use [Property Search](#) to search for npm packages according to their name or version.

Cleaning Up the Local Npm Cache

The npm client saves caches of packages that were downloaded, as well as the JSON metadata responses (named `.cache.json`).

The JSON metadata cache files contain URLs which the npm client uses to communicate with the server, as well as other ETag elements sent by previous requests.

We recommend removing the npm caches (both packages and metadata responses) before using Artifactory for the first time. This is to ensure that your caches only contain elements that are due to requests from Artifactory and not directly from <http://registry.npmjs.org>.

The default cache directory on Windows is `%APPDATA%\npm-cache` while on Linux it is `~/.npm`.

Npm Scope Packages

Artifactory fully supports [npm scope packages](#). The support is transparent to the user and does not require any different usage of the npm client.

Npm 'slash' character encoding

By default, the npm client encodes slash characters ('/') to their ASCII representation ("%2f") before communicating with the npm registry. If you are running Tomcat as your HTTP container (the default for Artifactory), this generates an "HTTP 400" error since Tomcat does not allow encoded slashes by default. To avoid this error when using npm scope packages, you can override this default behavior by defining the following property in the `catalina.properties` file of your Tomcat:

```
org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=true
```

Configuring the npm Client for a Scope Registry

Using Login Credentials

Scopes can be associated with a separate registry. This allows you to seamlessly use a mix of packages from the public npm registry and one or more private registries.

For example, you can associate the scope `@jfrog` with the registry `http://localhost:8081/artifactory/api/npm/npm-local/` by manually altering your `~/.npmrc` file and adding the following configuration:

```
@jfrog:registry=http://localhost:8081/artifactory/api/npm/npm-local/  
//localhost:8081/artifactory/api/npm/npm-local/:_password=cGFzc3dvcmQ=  
//localhost:8081/artifactory/api/npm/npm-local/:username=admin  
//localhost:8081/artifactory/api/npm/npm-local/:email=myemail@email.com  
//localhost:8081/artifactory/api/npm/npm-local/:always-auth=true
```

Getting .npmrc entries directly from Artifactory

From Artifactory 3.5.3, you can use the following command to get these strings directly from Artifactory:

```
$ curl -uadmin:password  
"http://localhost:8081/artifactory/api/npm/npm-local/auth/  
jfrog"  
@jfrog:registry=http://localhost:8080/artifactory/api/npm/  
npm-local/  
//localhost:8080/artifactory/api/npm/npm-local/:_password=  
QVA1N05OaHZTMnM5Qk02RkR5RjNBVmF4TVF1  
//localhost:8080/artifactory/api/npm/npm-local/:username=a  
dmin  
//localhost:8080/artifactory/api/npm/npm-local/:email=admi  
n@jfrog.com  
//localhost:8080/artifactory/api/npm/npm-local/:always-aut  
h=true
```

User email is required

When using scope authentication, npm expects a valid email address. Please make sure you have included your email address in your Artifactory user profile.

The password is just a base64 encoding of your Artifactory password, the same way used by the old authentication configuration.

Recommend npm command line tool version 2.1.9 and later.

While npm scope packages have been available since version 2.0 of the npm command line tool, we highly recommend using npm scope packages with Artifactory only from version 2.1.9 of the npm command line tool.

Using OAuth Credentials

Artifactory uses GitHub Enterprise as its default OAuth provider. If you have an account, you may use your GitHub Enterprise login details to be authenticated when using `npm login`.

Automatically Rewriting External Dependencies

Packages requested by the npm client frequently use external dependencies as defined in the packages' `package.json` file. However, since it is a best practice to consume external packages through Artifactory, you may use this option to configure Artifactory to rewrite the dependencies so that the npm client accesses dependencies through a virtual repository as follows:

- If you check **Enable Dependency Rewrite** in the npm virtual repository advanced configuration, when downloading an npm package, Artifactory analyzes the list of dependencies needed by the package.
- If any of the dependencies are hosted on external sources (e.g. on github.com),
 - Artifactory will download the dependency from the external source.
 - Artifactory will cache the dependency in one of the remote repositories aggregated by the virtual repository (as specified in the virtual repository advanced configuration).
 - Artifactory will then modify the dependency's entry in the package's `package.json` file indicating its new location in the Artifactory remote repository cache before returning it to the npm client.
 - Consequently, every time the npm client needs to access the dependency, it will be provisioned from its new location in the Artifactory remote repository cache.

Viewing Individual Npm Package Information

Artifactory lets you view selected metadata of an npm package directly from the UI.

In the **Tree Browser**, drill down to select the tgz file you want to inspect. The metadata is displayed in the **Npm Info** tab.

The screenshot shows the Artifactory interface for viewing an npm package. At the top, there is a file icon followed by the package name "request-2.31.0.tgz". To the right are "Download" and "Actions" buttons. Below this is a navigation bar with tabs: General, Npm Info (which is highlighted with a red box), Effective Permissions, Properties, Watchers, Builds, and Governance. The main content area has two sections: "Package Info" and "Dependencies".

Package Info

Name:	request
Version:	2.31.0
Description:	Simplified HTTP request client.
Repository:	

Dependencies

Name	Version
qs	~0.6.0
json-stringify-safe	~5.0.0
forever-agent	~0.5.0
node-uuid	~1.4.0
mime	~1.2.9

At the bottom right of the dependencies table, there is a pagination control: < page of 1 >

Bower Repositories

Overview

Artifactory supports [bower](#) repositories on top its [existing support](#) for advanced artifact management.

Artifactory support for Bower provides:

1. The ability to provision Bower packages from Artifactory to the Bower command line tool from all repository types.
2. Calculation of Metadata for Bower packages hosted in Artifactory's local repositories.
3. Access to remote Bower registries (such as <http://bower.herokuapp.com>) through [Remote Repositories](#) which provide the usual proxy and caching functionality.
4. The ability to access multiple Bower registries from a single URL by aggregating them under a [Virtual Repository](#).
5. Assign access privileges according to projects or development teams.

Page Contents

- Overview
- Configuration
 - Local Repositories
 - Deploying Bower Packages
 - Remote Repositories
 - Virtual Repositories
 - Advanced Configuration
- Using the Bower Command Line
 - Using Bower Version 1.5 and above
 - Using Older Versions of Bower
- Working with Artifactory without Anonymous Access
- Cleaning Up the Local Bower Cache
- Automatically Rewriting External Dependencies
- Viewing Individual Bower Package Information

Configuration

Local Repositories

To enable calculation of Bower package metadata set **Bower** to be the **Package Type** when you create your local Bower repository.

New Local Repository

Basic

Package Type *



Bower

Deploying Bower Packages

The Bower client does not provide a way to deploy packages and relies on a Git repository to host the Bower package code.

To deploy a Bower package into Artifactory, you need to use Artifactory's [REST API](#) or the [Web UI](#).

A Bower package is a simple `tar.gz` file which contains your project code as well as a `bower.json` file describing the package name and version.

Usually, you will use a custom [Grunt/ Gulp](#) task to pack your project into an archive file and deploy it to Artifactory.

Remote Repositories

The public [bower registry](#) does not contain any actual binary packages; it is a simple key-value store pointing from a package name to its equivalent Git repository.

Since 99% of the packages are hosted in GitHub, you will want to create a [Remote Repository](#) which serves as a caching proxy for [github.com](#). If necessary, you can do the same for [bitbucket.org](#) or any other remote repository you want to access.

Working with Stash?

If your packages are hosted on Bitbucket (formerly Stash), you need to ensure that the Stash Archive Plugin is installed on your Bitbucket server.

Artifacts (such as tar.gz files) requested from a remote repository are cached on demand. You can remove downloaded artifacts from the remote repository cache, however you can not manually deploy artifacts to a remote repository.

To define a remote repository to proxy [github.com](#) as well as the public Bower registry follow the steps below:

1. Create a new remote repository and set **Bower** to be its **Package Type**
2. Set the **Repository Key** value, and enter `https://github.com` in the **URL** field as displayed below

New Remote Repository

Basic

Advanced

Replications

Package Type *



Bower

Repository Key *

bower-remote-github

URL *

`https://github.com`

Test

3. In the **Bower Settings** section, select **GitHub** as the **Git Provider**, and leave the leave the default **Registry URL** (<https://bower.herokuapp.com>).

Finally, click "Save & Finish"

Bower Settings

Git Provider

GitHub

Registry URL

https://bower.herokuapp.com

Cancel

< Back Next >

Save & Finish

Bower Registry URL

Usually, you will point the **Bower Registry URL** field at the public registry as displayed above.

However, if you are using a private bower registry or a remote Artifactory instance, simply set the same URL as configured in **URL** field

Virtual Repositories

A Virtual Repository defined in Artifactory aggregates packages from both local and remote repositories.

This allows you to access both locally hosted Bower packages and remote proxied Bower registries from a single URL defined for the virtual repository.

To create a virtual Bower repository set **Bower** to be its **Package Type**, and select the underlying local and remote bower repositories to include under the **Repositories** section.

Repositories

Filter...

Available Repositories

Selected Repositories

 bower-local	X
 bower-remote-github	X

« < > »

Included Repositories

bower-local

bower-remote-github

Advanced Configuration

The fields under **External Dependency Rewrite** are connected to automatically rewriting external dependencies for Bower packages that need them.

Edit bower-virtual Repository

Basic > Advanced

Artifactory Requests Can Retrieve Remote Artifacts [?](#)

External Dependency Rewrite

Enable Dependency Rewrite

Remote Repository For Cache

bower-remote

Patterns Whitelist [?](#)

New Pattern [Add](#)

/github.com/

Enable Dependency Rewrite	When checked, automatically rewriting external dependencies is enabled.
Remote Repository For Cache	The remote repository aggregated by this virtual repository in which the external dependency will be cached.
Patterns Whitelist	A white list of Ant-style path expressions that specify where external dependencies may be downloaded from. By default, this is set to ** which means that dependencies may be downloaded from any external source. For example, if you wish to limit external dependencies to only be downloaded from <code>github.com</code> , you should add <code>**/github.com/**</code> (and remove the default <code>**</code> expression).

Using the Bower Command Line

Bower repositories must be prefixed with `api/bower` in the path

When accessing a Bower repository through Artifactory, the repository URL must be prefixed with `api/bower` in the path. This applies to all Bower commands including `bower install` and `bower info`.

For example, if you are using Artifactory standalone or as a local service, you would access your Bower repositories using the following URL:

```
http://localhost:8081/artifactory/api/bower/<repository key>
```

Or, if you are using Artifactory Online the URL would be:

```
https://<server name>.artifactoryonline.com/<server name>/api/bower/<repository key>
```

Artifactory has been updated to work seamlessly with the latest version of the Bower client from version 1.5, and also supports older versions of Bower.

Older versions of Bower

If your version of Bower is below 1.5, please refer to [Using Older Versions of Bower](#).

Using Bower Version 1.5 and above

In order to use Bower with Artifactory you need 2 components (npm packages):

1. `bower-art-resolver` - A custom, pluggable Bower resolver which is dedicated to integrate with Artifactory.
2. `bower` - Bower version **1.5.0** and above.

Once Bower is installed, add the Artifactory Bower resolver by editing your `~/.bowerrc` configuration file

Adding a Pluggable Resolver

```
{  
  "resolvers": [  
    "bower-art-resolver"  
  ]  
}
```

Bower Documentation

For more information, please refer to the Bower documentation on [Pluggable Resolvers](#).

Replace the default registry with a URL pointing to a Bower repository in Artifactory by editing your `~/.bowerrc` configuration file (the example below uses a repository with the key `bower-repo`):

Replacing the default registry

```
{  
  "registry": "http://localhost:8081/artifactory/api/bower/bower-repo"  
}
```

.bowerrc file location

Windows: `%userprofile%\bowerrc`

Linux: `~/.bowerrc`

We recommend referencing a [Virtual Repository URL](#) as a registry. This gives you the flexibility to reconfigure and aggregate other external sources and local repositories of Bower packages you deployed.

Once the Bower command line tool is configured, every `bower install` command will fetch packages from the bower repository specified above. For example:

```
$ bower install bootstrap
bower bootstrap#*
bower bootstrap#*
bower bootstrap#*
bower bootstrap#*
not-cached art://twbs/bootstrap#*
resolve art://twbs/bootstrap#*
extract archive.tar.gz
resolved art://twbs/bootstrap#e-tag:0b9cb774e1
```

Using Older Versions of Bower

Using Bower below version 1.5...

Version support

Older versions of Bower are only supported by Artifactory up to version 4.2.0.

In order to use Bower below version 1.5 with Artifactory you need 2 components (npm packages):

1. `bower-art-resolver` - A custom Bower resolver dedicated to integrate with Artifactory.
2. `bower-art` - A temporary custom Bower CLI with the pluggable resolvers mechanism currently in [pending pull request](#).

The `bower-art` package is a peer dependency of `bower-art-resolver`. Therefore, both can be easily installed with:

```
npm install -g bower-art-resolver
```

Use `bower-art` instead of `bower`

While Artifactory support for Bower is in Beta, after installing the required components, you need to execute `bower-art` instead of each `bower` command.

For example, use `bower-art install <pkg>` instead of `bower install <pkg>`

Updating Resolver

In order to update Artifactory resolver, please **uninstall** the "bower-art" npm package first, and then install the resolver. This step is necessary because npm doesn't update peer dependencies.

Once `bower-art` is installed, replace the default registry with a URL pointing to a Bower repository in Artifactory by editing your `~/.bowerrc` configuration file (the example below uses a repository with the key `bower-repo`):

Replacing the default registry

```
{
  "registry": "http://localhost:8081/artifactory/api/bower/bower-repo"
}
```

.bowerrc file location

Windows: `%userprofile%\bowerrc`

Linux: `~/bowerrc`

We recommend referencing a [Virtual Repository](#) URL as a registry. This gives you the flexibility to reconfigure and aggregate other external sources and local repositories you deployed.

Once the Bower command line tool is configured, every `bower-art install` command will fetch packages from the bower repository specified above. For example:

```
$ bower-art install bootstrap
bower bootstrap##          not-cached art://twbs/bootstrap#*
bower bootstrap##          resolve art://twbs/bootstrap#*
bower bootstrap##          extract archive.tar.gz
bower bootstrap##          resolved art://twbs/bootstrap#e-tag:0b9cb774e1
```

Working with Artifactory without Anonymous Access

By default, Artifactory allows anonymous access to Bower repositories. This is defined under [Security | General Configuration](#). For details please refer to [Allow Anonymous Access](#).

If you want to be able to trace how users interact with your repositories you need to uncheck the [Allow Anonymous Access](#) setting. This means that users will be required to enter their username and password.

Unfortunately, the Bower command line tool does not support authentication and you will need to add your credentials to the URL of the bower registry configured in `~/.bowerrc`:

Replacing the default registry with credentials

```
{
  "registry": "http://admin:password@localhost:8081/artifactory/api/bower/bower-repo"
}
```

Use an encrypted password

Use an encrypted password instead of clear-text; see [Centrally Secure Passwords](#).

Cleaning Up the Local Bower Cache

The Bower client saves caches of packages that were downloaded, as well as metadata responses.

We recommend removing the Bower caches (both packages and metadata responses) before using Artifactory for the first time. This is to ensure that your caches only contain elements that are due to requests from Artifactory and not directly from <http://bower.herokuapp.com>.

To clear the bower cache use:

Clean Bower Cache

```
bower-art cache clean
```

Automatically Rewriting External Dependencies

Packages requested by the Bower client frequently use external dependencies as defined in the packages' `bower.json` file. However, since it is a best practice to consume external packages through Artifactory, you may use this option to configure Artifactory to rewrite the dependencies so that the Bower client accesses dependencies through a virtual repository as follows:

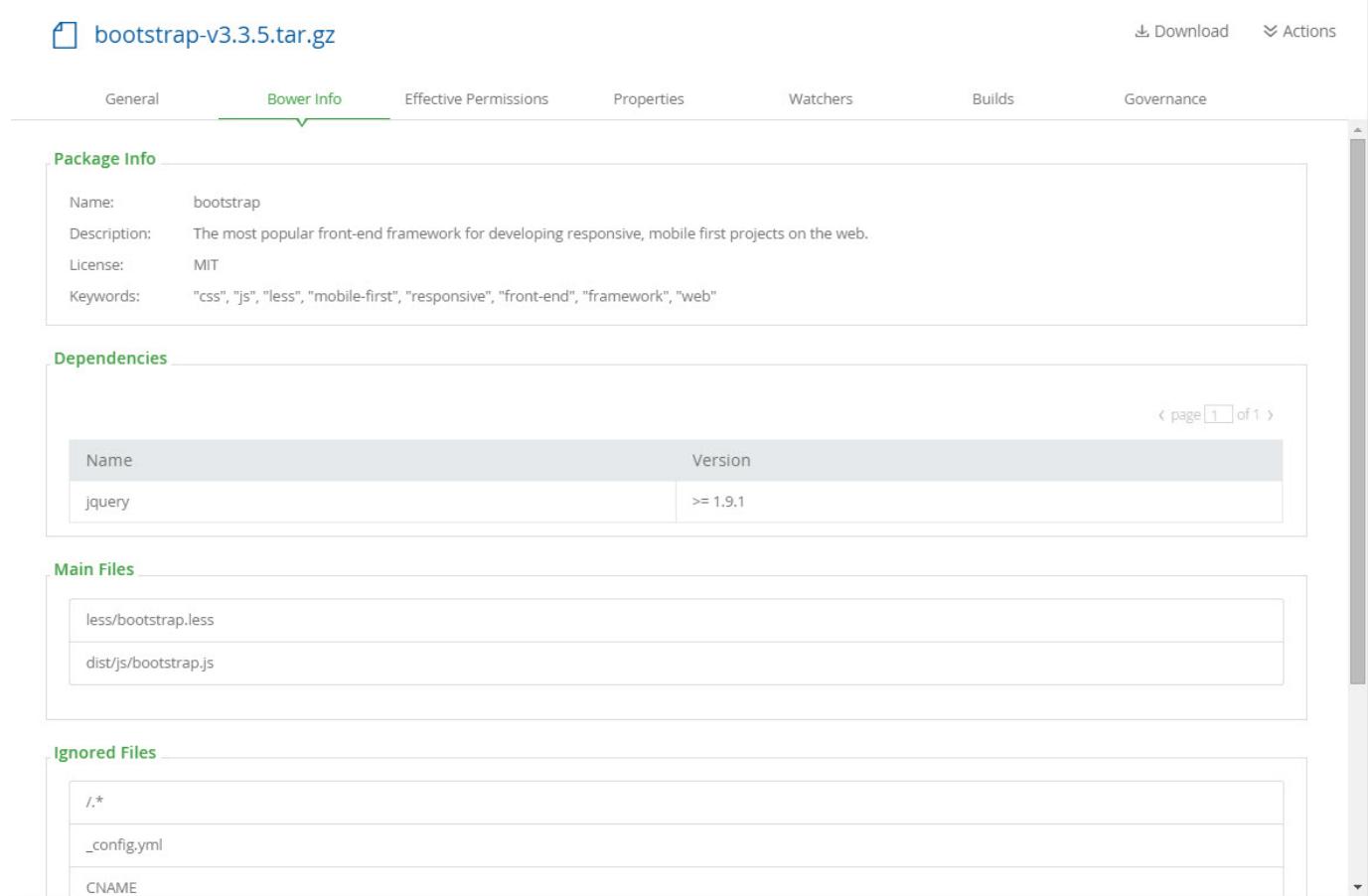
- If you check **Enable Dependency Rewrite** in the Bower virtual repository advanced configuration, when downloading an Bower package, Artifactory analyzes the list of dependencies needed by the package.

- If any of the dependencies are hosted on external sources (e.g. on github.com),
 - Artifactory will download the dependency from the external source.
 - Artifactory will cache the dependency in one of the remote repositories aggregated by the virtual repository (as specified in the virtual repository advanced configuration).
 - Artifactory will then modify the dependency's entry in the package's `bower.json` file indicating its new location in the Artifactory remote repository cache before returning it to the Bower client.
 - Consequently, every time the Bower client needs to access the dependency, it will be provisioned from its new location in the Artifactory remote repository cache.
-

Viewing Individual Bower Package Information

Artifactory lets you view selected metadata of a Bower package directly from the UI.

In the **Artifacts** tab, select **Tree Browser** and drill down to select the `zip/tar.gz` file you want to inspect. The metadata is displayed in the **Bower Info** tab.



The screenshot shows the Artifactory interface for viewing a Bower package. At the top, there is a navigation bar with tabs: General, Bower Info (which is active and highlighted in green), Effective Permissions, Properties, Watchers, Builds, and Governance. Below the navigation bar, the package name is displayed as "bootstrap-v3.3.5.tar.gz". On the right side, there are "Download" and "Actions" buttons. The main content area is divided into several sections:

- Package Info**: This section contains the following details:

Name:	bootstrap
Description:	The most popular front-end framework for developing responsive, mobile first projects on the web.
License:	MIT
Keywords:	"css", "js", "less", "mobile-first", "responsive", "front-end", "framework", "web"
- Dependencies**: This section lists the dependencies for the package. A table shows the dependency name and version requirement:

Name	Version
jquery	>= 1.9.1

Page navigation: < page 1 of 1 >
- Main Files**: This section lists the main files included in the package:

less/bootstrap.less
dist/js/bootstrap.js
- Ignored Files**: This section lists the files that are ignored:

/*
_config.yml
CNAME

Git LFS Repositories

Overview

From version 3.9, Artifactory supports [Git Large File Storage \(LFS\)](#) repositories on top of Artifactory's [existing support](#) for advanced artifact management.

Artifactory support for Git LFS provides you with a fully functional LFS server that works with the Git LFS client.

LFS blobs from your Git repository can be pushed and maintained in Artifactory offering the following benefits:

- **Performance:**

With Artifactory's file storage on your local or corporate network, file download times may be significantly reduced. When considering the number of files that may be needed for a build, this can drastically reduce your build time and streamline your workflow.

- **Reliable and consistent access to binaries:**

With Artifactory as your LFS repository, all the resources you need for development and build are stored on your own local or corporate network and storage. This keeps you independent of the external network or any 3rd party services.

- **Security and access control:**

Artifactory lets you define which users or groups of users can access your LFS repositories with a full set of permissions you can configure. You can control where developers can deploy binary assets to, whether they can delete assets and more. And if it's access to your servers that you're concerned about, Artifactory provides full integration with the most common access protocols such as LDAP, SAML, Crowd and others.

- **One solution for all binaries:**

Once you are using Artifactory to store media assets there is no need to use a 3rd party LFS provider. Artifactory can now handle those along with all the other binaries it already manages for you.

Page Contents

- Overview
- Configuration
 - Local Repositories
 - Setting Up the Git LFS Client to Point to Artifactory
- Working with Artifactory without Anonymous Access
- Metadata
- Storage

Configuration

Local Repositories

To create a Git LFS repository and enable calculation of LFS package metadata set **GitLfs** as the **Package Type**.

New Local Repository

Basic

Advanced

Package Type *



GitLfs

Repository Key *

lfs-local

Setting Up the Git LFS Client to Point to Artifactory

In order for your client to upload and download LFS blobs from artifactory the [lfs] clause should be added to the `.gitconfig` file of your Git repository in the following format.

.gitconfig

```
[lfs]
  url = "https://<artifactory server URL>/artifactory/api/lfs/<LFS local repo key>"
```

For example:

```
[lfs]
  url = "https://localhost:8080/artifactory/api/lfs/lfs-local"
```

You can also set different LFS endpoints for different remotes on your repo (as supported by the Git LFS client), for example:

.gitconfig different lfs url for remotes

```
[remote "origin"]
  url = https://...
  fetch = +refs/heads/*:refs/remotes/origin/*
  lfsurl = "http://localhost:8081/artifactory/api/lfs/lfs-local"
```

Copy these clauses using Set Me Up

If you select your GitLFS repository in the Tree Browser and click **Set Me Up**, Artifactory will display these clauses in a dialog from which you can simply copy and paste them.

Set Me Up

Tool
GitLfs

Repository
lfs-local

Resolve

In order for your client to upload and download LFS blobs from artifactory the [lfs] clause should be added to the .gitconfig file of your Git repository in the following format.

```
1 [lfs]
2 url = "http://10.100.1.110:8081/artifactory/api/lfs/lfs-local"
```

You can also set different LFS endpoints for different remotes on your repo (as supported by the Git LFS client), for example:

```
1 [remote "origin"]
2 url = <URL>
3 fetch = +refs/heads/*:refs/remotes/origin/*
4 lfsurl = "http://10.100.1.110:8081/artifactory/api/lfs/lfs-local"
```

Working with Proxies and HTTPS

When using HTTPS (i.e. behind a proxy) with a self signed certificate your configuration might also require you to add the following:

.gitconfig http section

```
[http]
sslverify = false
```

Always consult your System Administrator before bypassing secure protocols in this way.

When running Artifactory behind a proxy, defining a [base url](#) is usually required (depending on configuration) due to the operation of the Git LFS client which expects to receive redirect urls to the exact upload \ download location of blobs.

LFS repositories must be prefixed with api/lfs in the path

When accessing a Git LFS repository through Artifactory, the repository URL must be prefixed with **api/lfs** in the path, **except when configuring replication**.

For example, if you are using Artifactory standalone or as a local service, you would access your LFS repositories using the following URL:

<http://localhost:8081/artifactory/api/lfs/<repository key>>

Or, if you are using Artifactory Online the URL would be:

<https://<server name>.artifactoryonline.com/<server name>/api/lfs/<repository key>>

When configuring replication, reference the repository's browsable url i.e.;

`http://localhost:8081/artifactory/<repository key>`

Working with Artifactory without Anonymous Access

By default, Artifactory allows anonymous access to Git LFS repositories. This is defined in the **Admin** module under **Security | General**. For details please refer to [Allow Anonymous Access](#).

If you want to be able to trace how users interact with your repositories you need to uncheck the [Allow Anonymous Access](#) setting. This means that users will be required to enter their username and password.

The Git LFS client will ask for credentials for the Artifactory LFS repo when accessing it - if anonymous access is allowed you can just enter blank credentials, otherwise you should enter your Artifactory user name and password (not your Git one).

To make the authentication process automatic you can use [Git Credential Helpers](#) to store these for you, and have the Git LFS client authenticate automatically.

Git stores credentials in plain text by default

You should take extra measures to secure your username and password when using Git credential helpers

Metadata

As the Git LFS client supplies only limited data about the blob being uploaded (only its sha256 checksum, or 'OID', and its size) Artifactory does not store or process any metadata for LFS blobs.

You can set properties on the blobs for your own convenience but this requires extra logic to infer a sha256-named file stored in artifactory from the actual pointer stored in your Git repository.

Storage

Artifactory stores LFS blobs in a manner similar to the Git LFS client, using the provided sha256 checksum.

Git LFS blobs will be stored under a path such as `<lfs_repo>/objects/ad/1b/ad1b8d6e1cafdf33e941a5de462ca7edfa8818a70c79feaf68e5ed53dec414c4`

Where **ad** and **1b** are the 1st and 2nd, and the 3rd and 4th characters in the blob's name respectively.

Git LFS behavior when download from the LFS endpoint fails

The Git LFS client will download the pointer file it created in your remote Git repository if downloading the blob from the LFS endpoint failed (i.e. wrong credentials, network error etc.).

This will cause the actual file in your local repo to be substituted with the pointer created by the LFS client **with the same name** and lead to any number of problems this behavior can cause.

This is a limitation of the LFS client tracked by issue [89](#).

Vagrant Repositories

Overview

On top of [general support](#) for advanced artifact management, Artifactory support for [Vagrant](#) provides:

1. Distribution and sharing of Vagrant boxes within your organization.
2. Calculation of Metadata for Vagrant boxes hosted in Artifactory's local repositories
3. Extensive security features that give you fine-grained access control over boxes.

4. Support for flexible repository layouts that allow you to organize your boxes and assign access privileges according to projects or development teams.
5. Smart searches for boxes.

Page Contents

- Overview
- Configuration
 - Local Repositories
- Deploying Vagrant Boxes
 - Deploying a package using the UI
 - Set Me Up
 - Deploying a package using Matrix Parameters
 - Setting the Target Path
- Provisioning Vagrant Boxes
- Authenticated Access to Servers
- Watch the Screencast

Configuration

Local Repositories

To create a local Vagrant repository to host your Vagrant boxes, create a new Local Repository and set **Vagrant** as the **Package Type**.

New Local Repository

Basic

Advanced

Package Type *



Vagrant

Repository Key *

vagrant-local

Deploying Vagrant Boxes

Deploying a package using the UI

To deploy a Vagrant box to Artifactory, select the repository to which you want to deploy your Vagrant box and click **Deploy..**

The **Deploy** dialog is displayed with your selected repository as the **Target Repository** and a default **Target path**.

The screenshot shows the 'Deploy' dialog box. At the top, it says 'Target Repository' with 'vagrant-local' selected. Below that, 'Package Type: Vagrant' is chosen. Under 'Type', 'Single' is selected. A progress bar indicates 'Upload Completed'. In the 'Target Path' field, the path '/precise64-virtualbox-1.0.0.box; precise64-virtualbox-' is entered. A large button at the bottom right is labeled 'Deploy'.

You can add properties you wish to attach to your box as parameters to the target path.

For example, to upload the box **precise64-virtualbox-1.0.0.box**, and specify that its name is **precise64**, with a provider of **virtualbox** and the version is **1.0.0**, you would enter:

Specifying the Target Path

```
/precise64-virtualbox-1.0.0.box;box_name=precise64;box_provider=virtualbox;box_version=1.0.0
```

Set Me Up

You can also select your repository and click **Set Me Up** to view the cURL command you can use to upload your box.

Set Me Up

X

Tool

Vagrant

Repository

vagrant-local

Deploy

To deploy Vagrant boxes to this Artifactory repository using an explicit URL with Matrix Parameters use:

```
1 curl -i -u<USERNAME>:<API_KEY> -T <PATH_TO_FILE> "http://10.100.1.110:8081/artifactory/vagrant-local/{vagrantBoxName.box};box_name={name};box_provider={provider};box_version={version}"
```



Resolve

To provision a Vagrant box, all you need is to construct it's name in the following manner.

```
1 vagrant box add "http://10.100.1.110:8081/artifactory/api/vagrant/vagrant-local/{boxName}"
```



Be careful with spaces

Make sure you don't enter any superfluous spaces in the Target Path specification.

Once you have deployed your Vagrant box, and Artifactory has recalculated the repository index, your repository should be organized as displayed below:

The screenshot shows the Artifactory Repository Browser interface. At the top, there are search and filter options: 'Search by:' (dropdown), 'Quick', and 'Archive'. Below this is the title 'Artifact Repository Browser'. Underneath the title, there are tabs: 'Tree' (selected), 'Simple', and a magnifying glass icon. To the right of the tabs is a checkbox labeled 'Compress Empty Folders'. The main area displays a tree view of the 'vagrant-local' repository. The root node 'vagrant-local' has two children: 'precise64-virtualbox-1.0.0.box' and 'precise64-virtualbox-2.0.0.box'. Each file node has a small icon to its left.

Deploying a package using Matrix Parameters

You can also deploy Vagrant boxes to Artifactory with an explicit URL using [Matrix Parameters](#).

The URL is built similarly to the [Target Path](#) format as follows:

Deploying a package using Matrix Parameters

```
PUT  
"http://$ARTIFACTORY_HOME/{vagrantRepoKey}/{vagrantBoxName.box};box_name={name};box_provider={provider};box_version={version}"
```

For example, to upload the box **precise64-virtualbox-1.0.0.box**, and specify that it's name is **precise64**, with a provider of **virtualbox** and the version is **1.0.0**, you would enter:

Example

```
PUT  
"http://localhost:8080/artifactory/vagrant-local/precise64-virtualbox-1.0.0.box;box_name=precise64;box_provider=virtualbox;box_version=1.0.0"
```

Setting the Target Path

The **Target Path** can be anywhere in the repository, but it has to contain the 3 mandatory matrix parameters: **box_name**, **box_provider** and **box_version** and the file name must end with **.box**. The format is as follows:

Target Path Format

```
[path/to/box.box];box_name=[name];box_provider=[provider];box_version=[version]
```

path	The repository path where the Vagrant box should be stored. Artifactory supports storing Vagrant boxes anywhere within the repository. The examples on this page show Vagrant boxes stored under the root of the local repository.
name	The value to assign to the <code>box_name</code> property used to specify the Vagrant box name.
provider	The value to assign to the <code>box_provider</code> property used to specify the Vagrant box provider (<code>virtualbox/lxc</code> or others).
version	The value to assign to the <code>box_version</code> property used to specify the Vagrant box version (must comply with Vagrant's versioning schema)

Provisioning Vagrant Boxes

Vagrant boxes are available through the following URL:

Vagrant box URL

```
http://$ARTIFACTORY_HOME/api/vagrant/{vagrantRepoKey}/{boxName}
```

For example, to provision a Vagrant box called **precise64** from a repository called **vagrant-local**, you would construct it's name in the following manner:

Provisioning a Vagrant box

```
vagrant box add http://localhost:8080/artifactory/api/vagrant/vagrant-local/precise64
```

You can select the repository from which you want to provision your box, and click Set Me Up to get the specific URL for the selected repository.

You can also, optionally, pass parameters to specify a specific box version or provider. For example:

Provisioning a Vagrant box by version

```
vagrant box add http://localhost:8080/artifactory/api/vagrant/vagrant-local/precise64  
--provider virtualbox --box-version 1.0.0
```

In addition, boxes can be provisioned using properties; this is useful when you want to download the latest box tagged by a specific property. The properties query parameter value should comply with [Using Properties in Deployment and Resolution](#).

For example:

Provisioning a Vagrant box by version

```
vagrant box add  
http://localhost:8080/artifactory/api/vagrant/vagrant-local/precise64?properties=qa%2B  
=verified
```

Authenticated Access to Servers

If you need to access a secured Artifactory server that requires a username and password, you need to specify 2 environment variables:

1. ATLAS_TOKEN - A [Base64](#) encoded string of the user credentials (formatted `username:password`).
2. VAGRANT_SERVER_URL - The base URL for the Artifactory server.

Setting ATLAS_TOKEN and VAGRANT_SERVER_URL

```
export ATLAS_TOKEN=YWRtaW46QVAzWGHzWmlDU29NVmtaQ2dCZEY3SFNmdkVu  
export VAGRANT_SERVER_URL=http://localhost:8080
```

Getting the ATLAS_TOKEN directly from Artifactory

You can use the following command to get the ATLAS_TOKEN string directly from Artifactory:

```
$ curl -uadmin:password  
"http://localhost:8080/artifactory/api/vagrant/auth"  
YWRtaW46QVAzWGhzWmlDU29NVmtaQ2dCZEY3SFNmdkVu
```

Watch the Screencast

SBT Repositories

Overview

Artifactory provides integration with sbt, allowing you to configure it to resolve dependencies from, and deploy build output to sbt repositories. All you need to do is make minor modifications to your `build.sbt` configuration file.

Configuration

Local Repositories

A local sbt repository is used as a target to which you can deploy the output of your `build.sbt` script. To create an sbt repository, set the **Package Type** to **SBT**.

New Local Repository

Basic

Package Type *



SBT

Page Contents

- Overview
- Configuration
 - Local Repositories
 - Remote Repositories
 - Virtual Repositories
- Configuring sbt
 - Configuring Proxy Repositories
 - Configuring Artifact Resolution
 - Deploying Artifacts
- Sample Project

Remote Repositories

A **Remote Repository** defined in Artifactory serves as a caching proxy for a registry managed at a remote URL.

Artifacts (such as JAR files) requested from a remote repository are cached on demand. You can remove downloaded artifacts from the remote repository cache, however, you can not manually deploy artifacts to a remote SBT repository.

To define a remote sbt repository to proxy a remote sbt registry follow the steps below:

1. In the **Admin** module, under **Repositories | Remote**, click "New".
2. In the New Repository dialog, set the **Package Type** to **SBT**, set the **Repository Key** value, and specify the URL to the remote registry in the **URL** field as displayed below:

New Remote Repository

Basic Advanced Replications

Package Type *



Repository Key *

URL *

Test

3. Click "Save & Finish"

The parameters needed to configure remote sbt repositories are identical to those used for Maven repositories. For more details, please refer to [Type-Specific Basic Settings](#) under [Remote Repositories](#).

Virtual Repositories

A Virtual Repository defined in Artifactory aggregates packages from both local and remote repositories.

This allows you to access both locally hosted JARS and remote proxied sbt registries from a single URL defined for the virtual repository. To define a virtual sbt repository, create a [virtual repository](#), set the **Package Type** to be sbt, and select the underlying local and remote sbt repositories to include in the **Basic** settings tab.

New Virtual Repository

Basic Advanced

Package Type *



Repository Key *

General

Repository Layout

Public Description

Internal Description

Include Patterns (?)

Exclude Patterns (?)

Repositories

Filter...

Available Repositories

- ⊕ plugins-release-local
- ⊕ plugins-snapshot-local
- ⊕ jcenter
- ⊕ libs-release
- ⊕ libs-snapshot
- ⊕ plugins-release
- ⊕ plugins-snapshot
- ⊕ remote-repos

Selected Repositories

- ⊕ sbt-local
- ⊕ sbt-remote

Included Repositories

- sbt-local
- sbt-remote

Save & Finish

Click "Save & Finish" to create the repository.

The parameters needed to configure virtual sbt repositories are identical to those used for Maven repositories. For more details, please refer to [Virtual Repositories](#).

Configuring sbt

To configure sbt to resolve and deploy artifacts through sbt repositories defined in Artifactory, simply select one of the sbt repositories in the Tree Browser and click **Set Me Up**. Artifactory will display code snippets you can use in the relevant sbt files.

Set Me Up

Tool
sbt SBT

Repository
sbt-local

General

You can define proxy repositories in the `~/.sbt/repositories` file in the following way:

```
1 [repositories]
2 local
3 my-ivy-proxy-releases: http://localhost:8081/artifactory/sbt-local/,
[organization]/[module]/(scala_[scalaVersion])/)(sbt_[sbtVersion])/)[revision]/[types]/[artifact](-
[classifier]).[ext]
4 my-maven-proxy-releases: http://localhost:8081/artifactory/sbt-local/
```

In order to specify that all resolvers added in the sbt project added should be ignored in favor of those configured in the repositories configuration, add the following configuration option to the sbt launcher script:

```
1 -Dsbt.override.build.repos=true
```

You can add this setting to the `/usr/local/etc/sbt.opts` file

Configuring Proxy Repositories

To configure a repository defined in Artifactory as a proxy repository for sbt, add the code snippet below to your `~/.sbt/repositories` file.

```
[repositories]
local
my-ivy-proxy-releases: http://<host>:<port>/artifactory/<repo-key>,
[organization]/[module]/(scala_[scalaVersion])/)(sbt_[sbtVersion])/)[revision]/[types]/[artifact](-[classifier]).[ext]
my-maven-proxy-releases: http://<host>:<port>/artifactory/<repo-key>/
```

Where `<host>:<port>` are the host URL and port on which Artifactory is running.

For example, if you are running Artifactory on your local machine, on port 8081, and want to proxy Ivy repositories through a repository called `sbt-ivy-proxy`, and proxy Maven repositories through a repository called `sbt-maven-proxy` you would use:

```
[repositories]
local
my-ivy-proxy-releases: http://localhost:8081/artifactory/sbt-ivy-proxy/,
[organization]/[module]/(scala_[scalaVersion])/)(sbt_[sbtVersion])/)[revision]/[types]/[artifact](-[classifier]).[ext]
my-maven-proxy-releases: http://localhost:8081/artifactory/sbt-maven-proxy/
```

Proxying Maven and Ivy repositories separately

We recommend using different repositories to proxy Maven and Ivy repositories as a best practice described in [Proxying Ivy Repositories](#) in the [SBT Reference Manual](#).

To specify that all resolvers added in the sbt project should be ignored in favor of those configured in the repositories configuration, add the following configuration option to the sbt launcher script:

```
-Dsbt.override.build.repos=true
```

You can also add this setting to your `/usr/local/etc/sbtopts`

For more details on sbt proxy repositories, please refer to [Proxy Repositories](#) in the [SBT Reference Manual](#).

Configuring Artifact Resolution

To resolve artifacts through Artifactory, simply add the following code snippet to your `build.sbt` file:

```
resolvers += "Artifactory" at "http://<host>:<port>/artifactory/<repo-key>/"
```

Where `<host>:<port>` are the host URL and port on which Artifactory is running, and `repo-key` is the Artifactory repository through which you are resolving artifacts

Deploying Artifacts

To deploy sbt build artifacts to repositories in Artifactory, add the following code snippets to your `build.sbt` file.

For **releases**, add:

```
publishTo := Some("Artifactory Realm" at  
"http://<host>:<port>/artifactory/<repo-key>")  
credentials += Credentials("Artifactory Realm", "<host>", "<USERNAME>", "<PASS>")
```

For **snapshots**, add:

```
publishTo := Some("Artifactory Realm" at  
"http://<host>:<port>/artifactory/<repo-key>;build.timestamp=" + new  
java.util.Date().getTime)  
credentials += Credentials("Artifactory Realm", "<host>", "<USERNAME>", "<PASS>")
```

Where `<host>:<port>` are the host URL and port on which Artifactory is running, and `repo-key` is the Artifactory repository to which you are deploying artifacts.

Sample Project

A sample SBT project that uses Artifactory is available on [GitHub](#) and can be freely forked.

LDAP Groups

Overview

The LDAP Groups Add-on allows you to synchronize your LDAP groups with Artifactory and leverage your existing organizational structure for managing group-based permissions.

Unlike many LDAP integrations, LDAP groups in Artifactory use super-fast caching, and has support for both

Static, Dynamic and Hierarchical mapping strategies. Powerful management is accomplished with multiple switchable LDAP settings and visual feedback about the up-to-date status of groups and users coming from LDAP.

LDAP groups synchronization works by instructing Artifactory about the external groups authenticated users belong to. Once logged-in, you are automatically associated with your LDAP groups and inherit group-based permission managed in Artifactory.

Page Contents

- Overview
- Usage
 - Group Synchronization Strategies
- Synchronizing LDAP Groups with Artifactory
- Watch the Screencast

Usage

LDAP Groups settings are available in the **Admin** module under **Security | LDAP**.

To use LDAP groups you must first [set up an LDAP server for authentication](#) from the LDAP Settings screen. You must also alert Artifactory about the correct LDAP group settings to use with your existing LDAP schema.

Active Directory Users

For specific help with setting up LDAP groups for an Active Directory installation please see [Managing Security with Active Directory](#).

New LDAP Group Setting

LDAP Group Settings

Settings Name *	LDAP Setting
<input type="text"/>	<input type="button" value="▼"/>
Mapping Strategy Static Dynamic Hierarchy	
Group Member Attribute *	Group Name Attribute *
<input type="text" value="uniqueMember"/>	<input type="text" value="cn"/>
Description Attribute *	Filter *
<input type="text" value="description"/>	<input type="text" value="(objectClass=groupOfNames)"/>
Search Base	<input type="checkbox"/> Sub-tree Search

Synchronize LDAP Groups

<input type="text" value="Filter by Username"/>	Refresh Import	
Group Name	Description	Sync State

[Cancel](#) [Create](#)

Group Synchronization Strategies

Artifactory supports three ways of mapping groups to LDAP schemas:

- **Static:** Group objects are aware of their members, however, the users are not aware of the groups they belong to. Each group object such as `groupOfNames` or `groupOfUniqueNames` holds its respective member attributes, typically `member` or `uniqueMember`, which is a user DN.
- **Dynamic:** User objects are aware of what groups they belong to, but the group objects are not aware of their members.

Each user object contains a custom attribute, such as `group`, that holds the group DNs or group names of which the user is a member.

- **Hierarchy:** The user's DN is indicative of the groups the user belongs to by using group names as part of user DN hierarchy.
Each user DN contains a list of `ou`'s or custom attributes that make up the group association.
For example,
`uid=user1,ou=developers,ou=uk,dc=jfrog,dc=org` indicates that `user1` belongs to two groups: `uk` and `developers`.

Synchronizing LDAP Groups with Artifactory

Once you have configured how groups should be retrieved from your LDAP server, you can verify your set up by clicking the Refresh button on the [Synchronize LDAP Groups](#) sub-panel. A list of available LDAP groups is displayed according to your settings.

You are now ready to synchronize/import groups into Artifactory. The groups table allows you to select which groups to import and displays the sync-state for each group:

A group can either be completely new or already existing in Artifactory. If a group already exists in Artifactory it can become outdated (for example, if the group DN has changed) - this is indicated in the table so you can select to re-import it.

Once a group is imported (synced) a new external LDAP group is created in Artifactory with the name of the group.

Once you have imported LDAP groups, you can [Manage Permissions](#) on them as with regular Artifactory groups. Users association to these groups is external and controlled strictly by LDAP.

Make sure that LDAP group settings is enabled (in the [LDAP Groups Settings](#) panel) in order for your settings to become effective.

Watch the Screencast

Atlassian Crowd Integration

Overview

The integration between Artifactory and Crowd allows you to delegate authentication requests to Atlassian Crowd, use authenticated Crowd users and have Artifactory participate in a transparent SSO environment managed by Crowd.

In addition, this integration allows the use of JIRA User Server as an authentication server, but without support of SSO.

Page Contents

- [Overview](#)
- [Usage](#)
- [Crowd Groups](#)

Usage

Crowd integration can then be configured in the **Admin** module under **Security | Crowd**.

Crowd Integration

Crowd Server Integration

Enable Atlassian Crowd Integration

Crowd Server URL *

Crowd Application Name *

Crowd Application Password *

Session Validation Interval (Minutes) *

Use JIRA User Server

Use Default Proxy Configuration

Auto Create Artifactory Users

Reset Test Save

Synchronize Crowd Groups

Filter by:

Filter by User Name

Import < page of 1 >

Group Name	Description	Synced

Field Name	Description
Enable Atlassian Crowd Integration	Set this checkbox to enable security integration with Atlassian Crowd.
Crowd Server URL	The full URL of the Crowd server to use.
Crowd Application Name	The application name configured for Artifactory in Crowd.
Crowd Application Password	The application password configured for Artifactory in Crowd.
Session Validation Interval	The time window, in minutes, in which the session does not need to be revalidated.

Use Default Proxy Configuration	If this checkbox is set and a default proxy definition exists, it is used to pass through to the Crowd Server.
Use JIRA User Server	<p>Setting this checkbox will allow integration with JIRA User Server, and Artifactory will be able to authenticate users against the supplied JIRA server.</p> <p>This has a side-effect of disabling Single Sign On which is not supported by JIRA User Server.</p>
Auto Create Artifactory Users	<p>When automatic user creation is off, authenticated users will not be automatically created inside Artifactory. Instead, for every request from a Crowd user, the user is temporarily associated with default groups (if such groups are defined), and the permissions for these groups applies.</p> <p>Without automatic user creation, you will need to manually create the user inside Artifactory in order to manage user permissions that are not attached to his default groups.</p>
Filter by Username	Filter the search to see only groups of the specified username. If unchecked, all Crowd groups are shown.

To enable Crowd integration:

1. First define Artifactory as a [Custom Application Client](#) inside Crowd.
2. Complete the Crowd server URL, and the application credentials defined in Step 1.
3. The session validation interval defines the principal token validity time in minutes. If left at the default of 0, the token expires only when the session expires.
4. If you are using JIRA User Server provide its URL in the "Crowd Server URL" and check the "Use JIRA User Server". This will disable SSO, which is not supported by JIRA.
5. If you have a proxy server between the Artifactory server and the Crowd server, you may set the [Use Default Proxy Configuration](#) check-box.
6. You may instruct Artifactory to treat externally authenticated users as temporary users, so that Artifactory does not automatically create them in its security store. In this case, permissions for such users are based on the permissions given to auto-join groups.
7. Test the configured connection and save it.

System properties

Crowd configuration properties may be added to the run time system properties or to the `$ARTIFACTORY_HOME/etc/artifactory.system.properties` file.

NOTE that setting a configuration through properties overrides configurations set through the user interface.

Crowd Groups

To use Crowd groups:

1. Set up a Crowd server for authentication as detailed above.
2. Verify your setup by clicking the [Refresh](#) button on the **Synchronize Crowd Groups** sub-panel. A list of available Crowd groups, according to your settings is displayed.
3. The groups table allows you to select which groups to import into Artifactory and displays the sync-state for each group. A group can either be completely new or may already exist in Artifactory.
4. Select and import the groups that you wish to import to Artifactory. Once a group is imported (synced) a new external Crowd group is

created in Artifactory with the name of the group.

Synchronize Crowd Groups

Filter by: [Username](#) [Group Name](#)

Filter by User Name

[import](#) < page of 1 >

Group Name	Description	Synced

You can [Manage Permissions](#) on the synced Crowd groups in the same way you manage them for regular Artifactory groups.

Users association to these groups is external and controlled strictly by Crowd.

Ensure the Crowd group settings is enabled in order for your settings to become effective.

Single Sign-on

Overview

The Single Sign-on (SSO) add-on allows you to reuse existing HTTP-based SSO infrastructures with Artifactory, such as the SSO modules offered by Apache HTTPD.

You can have Artifactory's authentication work with commonly available SSO solutions, such as native NTLM, Kerberos etc.

SSO works by letting Artifactory know what trusted information it should look for in the HTTP request, assuming this request has already been authenticated by the SSO infrastructure that sits in front of Artifactory.

Page Contents

- [Overview](#)
- [Usage](#)
- [Integrating Apache and Tomcat](#)

Usage

To access the Single Sign-On (SSO) add-on, in the **Admin** module, select **Security | HTTP SSO**.

To enable SSO you must alert Artifactory that it is running behind a secure HTTP server that forwards trusted requests to it.

Then you must tell Artifactory in which variable to look for trusted authentication information.

The default is to look for a REMOTE_USER header or the request variable, which is set by Apache's AJP and JK connectors.

You can choose to use any request attribute (as defined by the Servlet specification) by providing a different variable name.

Adding Your Own SSO Integration

You can write a simple servlet filter to integrate with custom security systems and set a request attribute on the request to be trusted by the SSO add-on.

Finally, you can instruct Artifactory to treat externally authenticated users as temporary users, so that Artifactory does not create them in its

security database.

In this case, permissions for such users are based on the permissions given to auto-join groups.

HTTP SSO

Artifactory is Proxied by a Secure HTTP Server

Remote User Request Variable
REMOTE_USER

Auto Create Artifactory Users

Reset Save

Field Name	Description
Artifactory is Proxied by a Secure HTTP Server	<p>When checked, Artifactory trusts incoming requests and reuses the remote user originally set on the request by the SSO of the HTTP server.</p> <p>This is extremely useful if you want to use existing enterprise SSO integrations, such as the powerful authentication schemes provided by Apache (mod_auth_ldap, mod_auth_ntlm, mod_auth_kerb, etc.).</p> <p>When Artifactory is deployed as a webapp on Tomcat behind Apache:</p> <ul style="list-style-type: none">• If using mod_proxy_ajp, make sure to set tomcatAuthentication="false" on the AJP connector.• If using mod_jk, make sure to use the "JkEnvVar REMOTE_USER" directive in Apache's configuration.
Remote User Request Variable	The name of the HTTP request variable to use for extracting the user identity. Default is: REMOTE_USER.

Auto Create Artifactory Users	<p>When not checked, authenticated users are not automatically created inside Artifactory. Instead, for every request from a SSO user, the user is temporarily associated with default groups (if such groups are defined) and the permissions for these groups apply.</p> <p>Without auto user creation, you must manually create the user inside Artifactory to manage user permissions not attached to its default groups.</p>
--------------------------------------	---

Integrating Apache and Tomcat

When Artifactory is deployed as a webapp on Tomcat behind Apache:

- If using **mod_proxy_ajp** - Make sure to set `tomcatAuthentication="false"` on the AJP connector.
- If using **mod_jk** - Make sure to use the `JkEnvVar REMOTE_USER` directive in Apache's configuration.
- If using **mod_proxy** (requires **mod_proxy_http**, **mod_headers** and **mod_rewrite**) - There are two known working methods that forward the header:

```
RequestHeader set REMOTE_USER %{REMOTE_USER}e
```

or

```
RewriteEngine On
RewriteCond %{REMOTE_USER} (.+)
RewriteRule . - [E=RU:%1]
RequestHeader set REMOTE_USER %{RU}e
```

SAML SSO Integration

SAML (Security Assertion Markup Language)

SAML is an XML standard that allows you to exchange user authentication and authorization information between web domains.

Artifactory offers a SAML-based Single Sign-On service allowing federated Artifactory partners (identity providers) full control over the authorization process.

Using SAML, Artifactory acts as service provider which receives users' authentication information from external identity providers.

In this case, Artifactory is no longer responsible for authentication of the user although it still has to redirect the login request to the identity provider and verify the integrity of the identity provider's response.

Page Contents

- SAML (Security Assertion Markup Language)
- Artifactory's SAML configuration
- Understanding Artifactory's SAML-based SSO Login Process
- Understanding the Artifactory's SAML-based SSO Logout Process
- Artifactory Profiles and Bindings
 - After SAML Setup
 - Login Failure

Artifactory's SAML configuration

To use SAML-based SSO in Artifactory:

1. Login to Artifactory with administrator privileges.
2. In the **Admin** module, go to **Security | SAML SSO**.
3. Enable the SAML integration by checking the **Enable SAML Integration** checkbox.
4. Enable or disable the “Auto Create Artifactory users” (Using SAML login) which allows new users to persist in the database.
5. Provide the **SAML Login URL** and **SAML Logout URL**

SAML Logout URL

In order to simultaneously logout from your SAML provider and Artifactory, you need to correctly set your provider's logout URL **SAML Logout URL** field. Setting this incorrectly will keep your users logged in with the SAML provider even after logging out from Artifactory.

6. Provide the service provider name (Artifactory name in SAML federation)
7. Provide the X.509 certificate that contains the public key. The public key can use either the DSA or RSA algorithms. Artifactory uses this key to verify SAML response origin and integrity. Make sure to match the embedded public key in the X.509 certificate with the private key used to sign the SAML response.

SAML SSO

Enable SAML Integration

SAML Login URL *

SAML Logout URL *

SAML Service Provider Name *

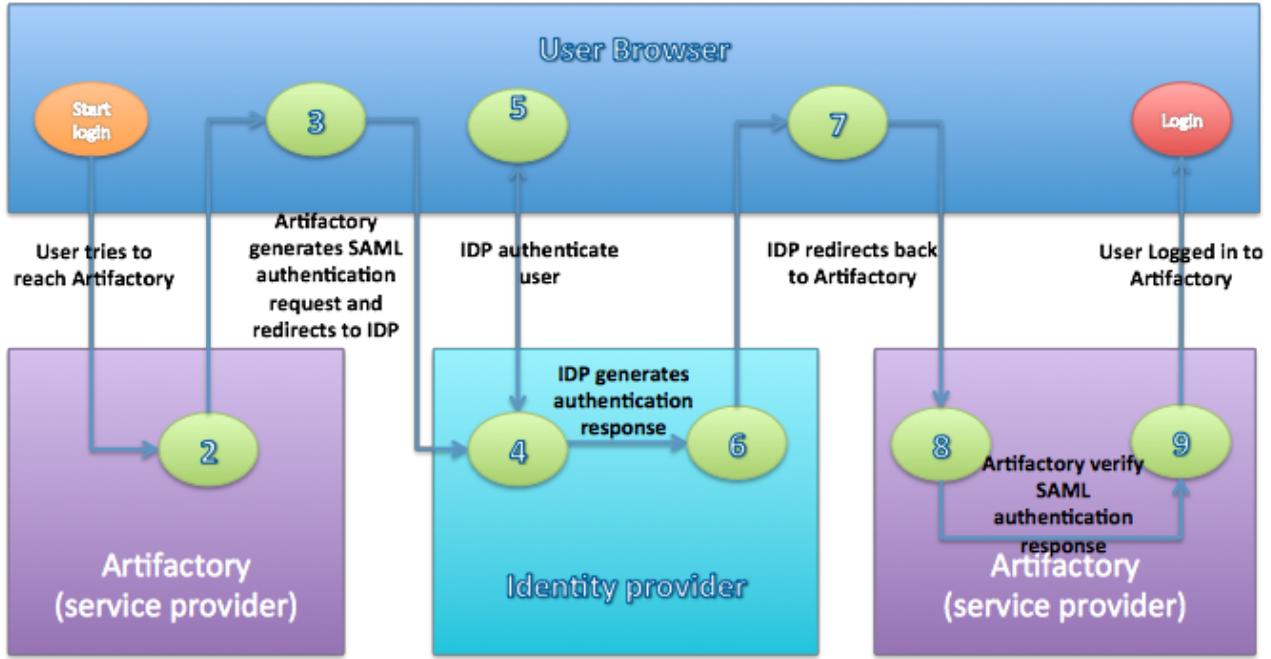
SAML Certificate:

Auto Create Artifactory Users

Understanding Artifactory's SAML-based SSO Login Process

1. The user attempts to reach a hosted Artifactory, Home Page.
2. Artifactory generates a SAML authentication request.
3. The SAML request is encoded and embedded into the identity provider URL.
4. Artifactory sends a redirect to the user's browser. The redirect URL includes the encoded SAML authentication request that should be submitted to the identity provider.
5. The identity provider decodes the SAML message and authenticates the user. The authentication process can proceed by asking for valid login credentials or by checking for valid session cookies.
6. The identity provider generates a SAML response that contains the authenticated user's username. In accordance with the SAML 2.0 specification, this response is digitally signed with the identity provider's private DSA/RSA keys.
7. The identity provider encodes the SAML response and returns that information to the user's browser. The identity provider redirects back to Artifactory with the signed response.
8. Artifactory's ACS verifies the SAML response using the partner's public key. If the response is successfully verified, the ACS redirects the user to the destination URL.
9. The user has been redirected to the destination URL and is logged in to Artifactory.

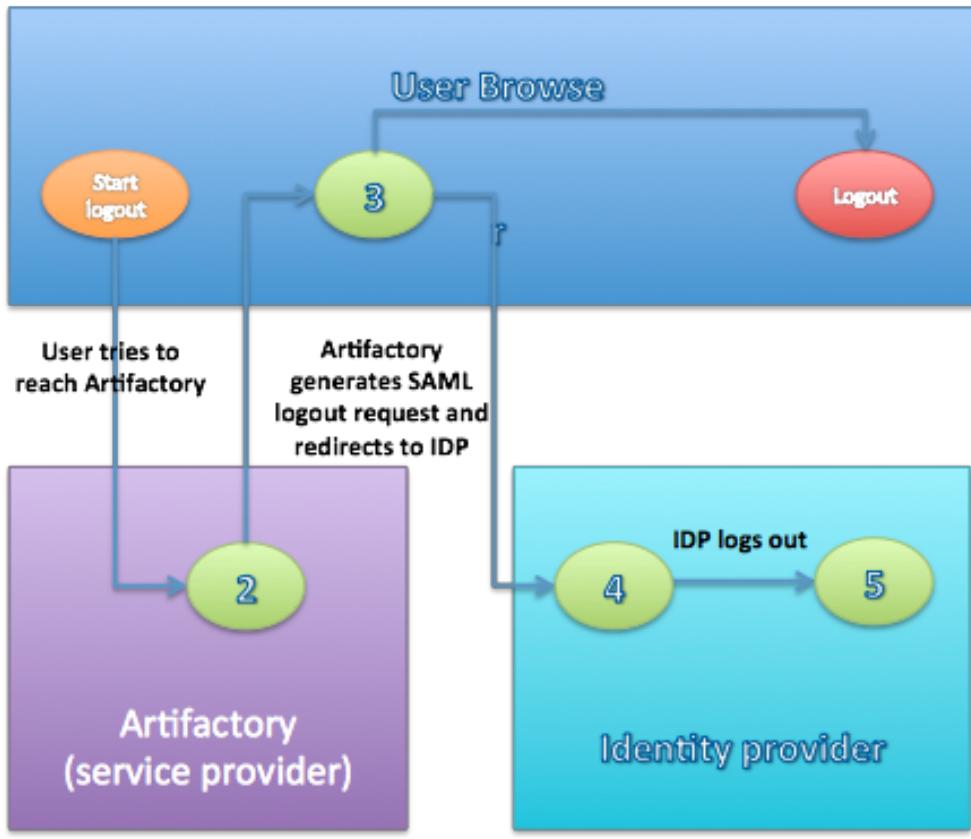
Figure (2) Artifactory's SAML-based SSO login process.



Understanding the Artifactory's SAML-based SSO Logout Process

1. The user attempts to reach a hosted Artifactory logout link.
2. Artifactory logs the client out and generates a SAML logout request.
3. Artifactory redirects to the identity provider with the encoded SAML logout request.
4. The identity provider decodes the SAML message and logs the user out.
5. The user is redirected to the configured URL in the identity provider.

Figure (3) Artifactory's SAML-based SSO logout process.



Artifactory Profiles and Bindings

Artifactory currently supports the Web Browser SSO and Single Logout Profiles.

The Web Browser SSO Profile uses HTTP redirect binding to send the AuthnRequest from the service provider to the identity provider, and HTTP POST to send the authentication response from the identity provider to the service provider.

Similar to the previous profile, the Single Logout Profile uses HTTP redirect binding to send the LogoutRequest from the service provider to the identity provider and HTTP POST to send the logout response from the identity provider to the service provider.

If your IDP supports uploading service provider metadata, you can use the following metadata XML:

Figure (4) Artifactory's service provider metadata XML.

Artifactory SP metadata XML

```

<ns2:EntityDescriptor xmlns="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns2="urn:oasis:names:tc:SAML:2.0:metadata" entityID="<SP_NAME_IN_FEDERATION>">

<ns2:SPSSODescriptor WantAssertionsSigned="true" AuthnRequestsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">

<ns2:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</ns2:NameIDFormat>
<ns2:AssertionConsumerService index="1"
Location="<ARTIFACTORY_URL>/webapp/saml/loginResponse"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"/>
</ns2:SPSSODescriptor>
</ns2:EntityDescriptor>

```

NOTE! that to use the service provider metadata:

Do not forget to update the following fields in the service provider metadata XML:

- entityID - Artifactory's ID in the federation
- Location - Artifactory's home URL

After SAML Setup

Using SAML, Artifactory automatically redirects the request to IDP which Authenticates the user and after a successful login redirects back to Artifactory.

If "Anonymous User" is enabled, Artifactory doesn't have to authenticate the user therefore it doesn't redirect to the IDP. If the user still wants to sign in through SAML, they can do so by clicking the "SSO login" link in the login page.

Login Failure

In case of IDP failover or bad configuration, Artifactory allows you to bypass SAML login by using Artifactory login page:

http://<ARTIFACTORY_URL>/webapp/login.html

OAuth Integration

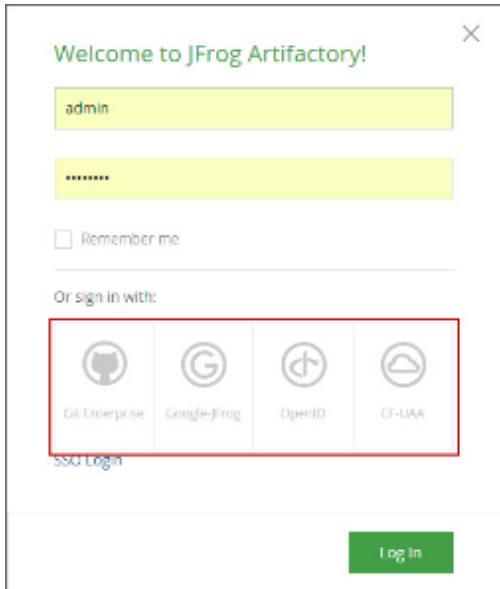
Overview

From version 4.2, Artifactory is integrated with OAuth allowing you to delegate authentication requests to external providers and let users login to Artifactory using their accounts with those providers.

Currently, the provider types supported are **Google**, **OpenID Connect**, **GitHub Enterprise**, and **Cloud Foundry UAA**. You may define as many providers of each type as you need.

Usage

When OAuth is enabled in Artifactory, users may choose to sign in through any of the supported OAuth providers. To log in through a provider, simply click on the provider's button in the login screen.



You will be redirected to the login screen of the corresponding provider.

If you are already logged in to any of that provider's applications you will not need to log in again, but you may have to authorize Artifactory to access your account information, depending on the provider type.

Page Contents

- Overview
- Usage

- Configuring OAuth
 - Adding a New Provider
- Binding Existing User Accounts
- Creating OAuth Provider Accounts
 - GitHub OAuth Setup
 - Google OAuth Provider Setup
 - Cloud Foundry UAA Setup

Configuring OAuth

To access OAuth integration settings, in the **Admin** module, select **Security | OAuth SSO**.

OAuth SSO Configuration

General OAuth Settings

Enable OAuth

Auto Create Artifactory Users

Default Provider

Git Enterprise

[Reset](#) Save

Enable OAuth	If checked, authentication with an OAuth provider is enabled and Artifactory will display all OAuth providers configured. If not checked, authentication is by Artifactory user/password.
Auto Create Artifactory Users	If checked, Artifactory will create an Artifactory user account for any new user logging in to Artifactory for the first time.
Default Provider	<p>Specifies the provider through which different clients (such as NPM, for example) should authenticate their login to gain access to Artifactory.</p> <div style="border: 1px solid orange; padding: 10px; margin-top: 10px;"> <p>Default provider</p> <p>Currently, only a GitHub Enterprise OAuth provider may be defined as the Default Provider.</p> </div>

Adding a New Provider

The list of providers defined in Artifactory is displayed in the **Providers** section.

Providers					 New
4 Providers					« Page 1 of 1 »
Name	Type	ID	Auth Url	Enabled	
CF-UAA	Cloud Foundry	login	https://cloud Foundry UAA URL	<input checked="" type="checkbox"/>	
Git Enterprise	GitHub	https://git enterprise URL	https://git enterprise URL	<input checked="" type="checkbox"/>	
Google-JFrog	Google	JFrog	https://google JFrog URL	<input checked="" type="checkbox"/>	
OpenID	OpenID	admin	https://openid URL	<input checked="" type="checkbox"/>	

To add a new provider, click "New". Artifactory displays a dialog letting you enter the provider details. These may vary slightly depending on the provider you are configuring.

Create New Provider

Provider Settings

Enabled

Provider Name *

Provider Type *

GitHub

Provider ID *

admin

Secret *

.....

Basic URL *

Auth URL *

API URL *

Token URL *

The following table describes the settings required by each supported provider, and the corresponding values you should use (where available):

	GitHub Enterprise	
--	--------------------------	--

Enabled		
Provider Name		
Provider Type		
Provider ID		

<i>Secret</i>		
<i>Domain</i>		
<i>Basic URL</i>	<i>https://github.com</i>	
<i>Auth URL</i>	<i>https://github.com/login/oauth/authorize</i>	<i>https://accounts.g...</i>

API URL	<code>https://api.github.com/user</code>	<code>https://www.google.com</code>
Token URL	<code>https://github.com/login/oauth/access_token</code>	<code>https://www.google.com</code>

Binding Existing User Accounts

If you already have an account in Artifactory, in order to be able to login using any of your OAuth provider accounts, you need to bind your Artifactory account to the corresponding account.

To bind your account, go to your [Profile](#) page and enter your Artifactory password to unlock it.

Under **OAuth User Binding**, select **Click to bind** next to the OAuth provider you wish to bind to.

OAuth User Binding



Git Enterprise

[Click to bind](#)



Google-JFrog

[Click to bind](#)



OpenID

[Click to bind](#)



CF-UAA

[Click to bind](#)

Creating OAuth Provider Accounts

In order to use OAuth authentication, you need to set up an account with each OAuth provider you wish to use in order to get the various parameters (such as Provider ID and Secret) you will need to set up OAuth integration in Artifactory.

GitHub OAuth Setup

To set up your OAuth account on GitHub, execute the following steps:

1. Login to your GitHub account. Under your personal profile settings, select **Applications** and click the [Developer Applications](#) tab.
2. Click **Register new application**.
3. Set the **Application name**. For example, **ArtifactoryAOL OAuth**.
4. Set the **Homepage Url**. This is your Artifactory server host URL (`https://<artifactory-server>/`).
For example, `https://mycompany.artifactoryonline.com/mycompany/`

5. Set the **Authorization Callback Url** as follows:
 - a. For Artifactory on-prem installation: `http://<server_host>/artifactory/api/oauth2/loginResponse`
For example, `http://mycompany.artifactory.com/artifactory/api/oauth2/loginResponse`
 - b. For Artifactory Online: `https://<server_name>.artifactoryonline.com/<server_name>/api/oauth2/loginResponse`.
For example, `https://mycompany.artifactoryonline.com/mycompany/api/oauth2/loginResponse`.
6. Click **Register application** to generate your **Client ID** and **Client Secret**.
Make a note of these; you will need them to configure OAuth authentication through GitHub on Artifactory.

Authorized applications Developer applications

Register a new OAuth application

Application name
Artifactory AOL OAuth

Something users will recognize and trust

Homepage URL
`http://internalsandbox.artifactoryonline.com/internalsandbox`

The full URL to your application homepage

Application description
AOL OAuth

This is displayed to all potential users of your application

Authorization callback URL
`https://internalsandbox.artifactoryonline.com/internalsandbox/api/oauth2/login`

Your application's callback URL. Read our [OAuth documentation](#) for more information

Register application

Google OAuth Provider Setup

To set up your OAuth account on Google, execute the following steps:

1. Login to [Google Developer Console](#).
2. Create a new Web Application project. For example, "Artifactory OAuth".
3. Once the project is created, in the left navigation bar, select **APIs & auth | Credentials**.
4. Select the **OAuth consent screen** tab and configure the consent screen end users will see when logging in with the Google credentials.

Google Developers Console Artifactory OAuth ▾

Sign up for a free trial. Try the beta console

Home
Permissions
APIs & auth
APIs
Credentials
Monitoring
Source Code
Cloud Launcher
Deployments
Compute
Networking
Storage
Big Data

Credentials OAuth consent screen Domain verification

The consent screen will be shown to users whenever you request access to their private data using your client ID

Note: This screen will be shown for all applications using this project's OAuth 2.0 client IDs

Email address

Product name

Homepage URL (Optional)

Product logo URL (Optional) http://www.example.com/logo.png

This is how your logo will look to end users
Max size: 120x120 px 

Privacy policy URL (Optional)

Terms of service URL (Optional)

 Logo

Project Name would like to:

Know your basic profile info and list of people in your circles.

Make your listen, app and comment activity available via Google, visible to: Your circles

By clicking Accept, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

5. Back in the **Credentials** tab, Click **Add Credential** and select **OAuth 2.0 client ID**

Credentials OAuth consent screen Domain verification

APIs Credentials API

You need credentials to access APIs. [Enable the APIs you plan to use](#) and then create the credentials they require. Depending on the API, you need an API key, a service account, or an OAuth 2.0 client ID. [Refer to the API documentation](#) for details.

API key
Identifies your project using a simple API key to check quota and access.
For APIs like Google Translate.

OAuth 2.0 client ID
Requests user consent so your app can access the user's data.
For APIs like Google Calendar.

Service account
Enables server-to-server, app-level authentication using robot accounts.
For use with Google Cloud APIs.

6. Under **Create client ID**, select **Web application**.

7. Enter a **Name** and set the **Authorized redirect URIs**

For Artifactory on-prem: https://<server_host>/artifactory/api/oauth2/loginResponse.

For Artifactory Online: https://<server_name>.artifactoryonline.com/<server_name>/api/oauth2/loginResponse.

Create client ID

Application type

Web application
 Android [Learn more](#)
 Chrome App [Learn more](#)
 iOS [Learn more](#)
 PlayStation 4
 Other

Name

Artifactory OAuth

Authorized JavaScript origins

Enter JavaScript origins here or redirect URIs below (or both) [?](#)
Cannot contain a wildcard (`http://*.example.com`) or a path (`http://example.com/subdir`).

`http://www.example.com`

Authorized redirect URIs

Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.

`https://mycompany.artifactoryonline.com/mycompany/api/oauth2/loginResponse` [×](#)
`http://www.example.com/oauth2callback`

[Create](#) [Cancel](#)

8. Click **Create** to generate your **Client ID** and **Client Secret**.

Credentials OAuth consent screen Domain verification

OAuth client

Here is your client ID

[REDACTED]

for details.

Here is your client secret

[REDACTED]

8okoleh3

OK

Make a note of these; you will need them to configure OAuth authentication through Google on Artifactory.

Cloud Foundry UAA Setup

OAuth authentication with Cloud Foundry UAA is supported from Artifactory version 4.2.1.

To setup your OAuth authentication with Cloud Foundry UAA, fill in the fields as needed.

UAA	Vm Credentials
	Admin Credentials
	Push Console Credentials
	Run Smoke Tests Credentials
	System Services Credentials
	System Verification Credentials
	System Passwords Client Credentials
	Login Client Credentials

VCS Repositories

Overview

Artifactory fully supports VCS repositories on top of Artifactory's [existing support](#) for advanced artifact management.

Today, many technologies that are consumed as pure source files are deployed as binaries (for example, PHP, Rails, Node.js, Ruby etc.). As a Binary Repository Manager, Artifactory completes the picture by providing you an easy, safe and secure way to consume those binaries.

Artifactory support for VCS provides:

1. The ability to list all the tags and branches of any VCS repository.
2. Access to remote VCS repositories (such as <https://github.com>) through [Remote Repositories](#) which provide the usual proxy and caching functionality.
3. On-demand local caching of tags and branches for later use in case of network instability or hosted VCS service downtime.
4. The ability to assign access privileges according to projects or development teams.

Page Contents

- Overview
- Configuration
 - Repository Layout
 - Remote Repositories
 - Git Providers
- Using the API
 - Get VCS Tags
 - Get VCS Branches
 - Download Tag
 - Download File within a Tag
 - Download Branch
 - Download File within a Branch
 - Download Release
 - Examples
- Accessing Private VCS Repositories

Configuration

Repository Layout

Before you start creating remote repositories, you need to create a custom layout to support a more hierarchical layout of the cached items.

To use a hierarchical layout for your repository, you should define a [Custom Layout](#). This will ensure that different maintenance features like [Version Cleanup](#) will work correctly.

Repository layout is optional

Defining a [Custom Layout](#) for your repository is an optional step, however it is recommended to do so since it allows Artifactory to perform different maintenance tasks such as [Version Cleanup](#) automatically.

Once a remote repository is created you cannot change its layout so we recommend that you define it beforehand.

Below is an example of a [Custom Layout](#) named `vcs-default`:

Edit vcs-default Repository Layout

Repository Layout Settings

Layout Name *

vcs-default

Artifact Path Pattern * [?](#)

`[orgPath]/[module]/[refs<tags|branches>]/[baseRev]/[module]-[baseRev](-[file|tagRev])-[-classifier].[ext]`

Distinctive Descriptor Path Pattern [?](#)

Folder Integration Revision RegExp * [?](#)

`.*`

File Integration Revision RegExp * [?](#)

`[a-zA-Z0-9]{40}`

Test Artifact Path Resolution

Test Path

Test

Regular Expression View

Resolve

You can configure this Custom Layout as displayed in the image above, or simply copy the below code snippet into the relevant section in your Artifactory Central Configuration (in the **Admin** module, under **Advanced | Config Descriptor**):

```

<repoLayout>
  <name>vcs-default</name>

  <artifactPathPattern>[orgPath]/[module]/[refs<tags|branches>]/[baseRev]/[module]-[base
  Rev](-[fileItegRev])(-[classifier]).[ext]</artifactPathPattern>
    <distinctiveDescriptorPathPattern>false</distinctiveDescriptorPathPattern>
    <folderIntegrationRevisionRegExp>.*</folderIntegrationRevisionRegExp>
    <fileIntegrationRevisionRegExp>[a-zA-Z0-9]{40}</fileIntegrationRevisionRegExp>
</repoLayout>

```

If a repository package layout is in a corresponding folder hierarchy, the Artifactory Version Cleanup tool correctly detects previously installed versions.

Delete Versions

Group ID	Version	Directories Count
junit	3.8.1	1
junit	4.7	1

< Page 1 of 1 >

Cancel
Delete Selected

Searching for artifact versions using the REST API also works properly:

```

$ curl
"http://localhost:8081/artifactory/api/search/versions?g=jquery&a=jquery&repos=github-
cache"
{
  "results" : [ {
    "version" : "2.0.3",
    "integration" : false
  }, {
    "version" : "master-062b5267d0a3538f1f6dee3df16da536b73061ea",
    "integration" : true
  } ]
}

```

Remote Repositories

You need to create a [Remote Repository](#) which serves as a caching proxy for [github.com](#). If necessary, you can do the same for [bitbucket.org](#) or any other remote git repository that you need.

Artifacts (such as tar.gz files) requested from a remote repository are cached on demand. You can remove downloaded artifacts from the remote repository cache, however, you can not manually deploy artifacts to a remote repository.

To create a remote repository to proxy `github.com` follow the steps below:

1. In the **Admin** module, under **Repositories | Remote**, click "New" and set **VCS** to be the **Package Type**.
2. Set the **Repository Key**, and specify the **URL** to be `https://github.com` as displayed below:

New Remote Repository

Basic Advanced Replications

Package Type *

 VCS

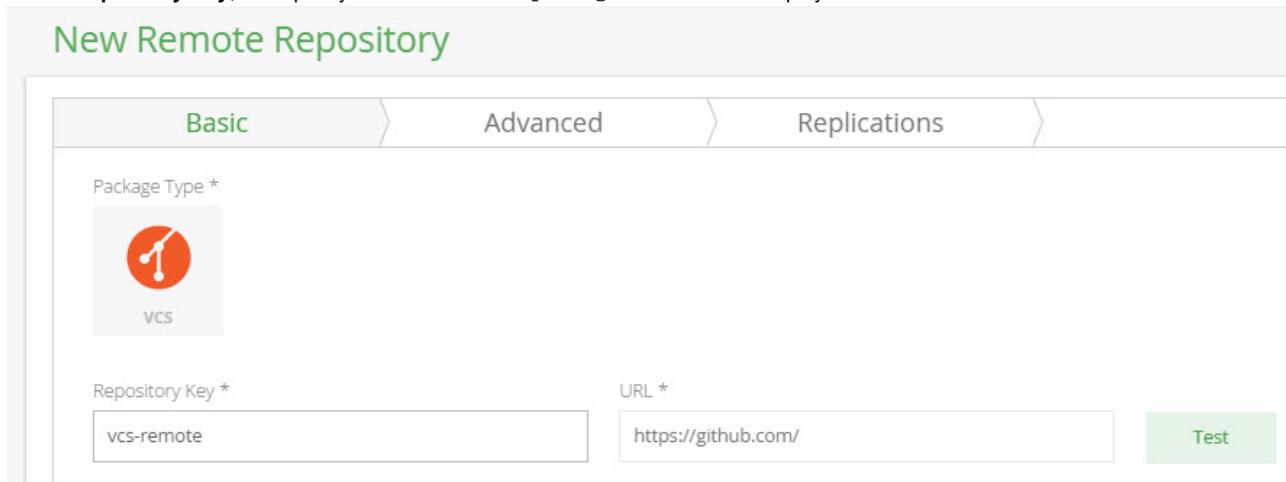
Repository Key *

vcs-remote

URL *

`https://github.com/`

Test



3. Under VCS Settings, select the GitHub provider in the **Git Provider** field and click "Save & Finish".

VCS Settings

Git Provider

Max Unique Snapshots [\(?\)](#)

GitHub

0

List Remote Folder Items [\(?\)](#)

Git Providers

Artifactory supports proxying the following Git providers out-of-the-box: GitHub, Bitbucket, Stash, a remote Artifactory instance or a custom Git repository as displayed below:

VCS Settings

Git Provider

Select Git provider...

GitHub

BitBucket

Stash

Artifactory

Custom

Use the custom provider if you have a Git repository which does not exist in the pre-defined list. In this case, you need to provide Artifactory with the download paths for your Git tarballs.

You do so by providing 4 placeholders:

Placeholder	Description
{0}	Identifies the username or organization name.
{1}	Identifies the repository name.
{2}	Identifies the branch or tag name.
{3}	Identifies the file extension to download.

For example, GitHub exposes tarball downloads at: `https://github.com/<user>/<repo>/archive/<tag/branch>. <extension>`

Therefore, the custom download path configured for Artifactory should be {0}/{1}/archive/{2}.{3}

Using the API

VCS repositories must be prefixed with api/vcs in the path

When accessing a VCS repository through Artifactory, the repository URL must be prefixed with **api/vcs** in the path.

For example, if you are using Artifactory standalone or as a local service, you would access your VCS repositories using the following URL:

`http://localhost:8081/artifactory/api/vcs/<repository key>`

Or, if you are using Artifactory Online, the URL would be:

`https://<server name>.artifactoryonline.com/<server name>/api/vcs/<repository key>`

Artifactory exposes REST APIs that let you do the following with VCS repositories:

- [List all tags](#)
- [List all branches](#)
- Download a specific tag
- Download a file within a tag
- Download a specific branch
- Download a file within a branch

To help you build the API call correctly, you can select the VCS repository you want to interact with and click **Set Me Up**.

Set Me Up



Tool

VCS

Repository

aql-elastic

General

Artifactory supports downloading tags or branches using a simple GET request. You can also specify to download a specific tag or branch as a tar.gz or zip, and a specific file within a tag or branch as a zip file.

Resolve

Use the following command to list all tags

```
1 curl -i -u<USERNAME>:<API_KEY> -XGET http://localhost:8080/artifactory/api/vcs/tags/aql-elastic
  /<USER_ORG>/<REPO>
```



Use the following command to list all branches

```
1 curl -i -u<USERNAME>:<API_KEY> -XGET http://localhost:8080/artifactory/api/vcs/branches/aql-elastic
  /<USER_ORG>/<REPO>
```



Get VCS Tags

Description: Lists all VCS tags.

Since: 3.6.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/vcs/tags/{repoKey}/{userOrg}/{repo}

Produces: application/json

Sample Output:

```
GET /api/vcs/tags/github/jquery/jquery
[ {
  "name" : "1.0",
  "commitId" : "bcc8a837055fe720579628d758b7034d6b520f2e",
  "isBranch" : false
}, {
  "name" : "1.0.1",
  "commitId" : "bcc8a837055fe720579628d758b7034d6b520f2e",
  "isBranch" : false
}
... ]
```

Get VCS Branches

Description: Lists all VCS branches.

Since: 3.6.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/vcs/branches/{repoKey}/{userOrg}/{repo}

Produces: application/json

Sample Output:

```
GET /api/vcs/branches/github/jquery/jquery
[ {
  "name" : "1.11-stable",
  "commitId" : "852529c9f148de6df205be01659a79731ce8ebef",
  "isBranch" : true
}, {
  "name" : "1.x-master",
  "commitId" : "73c1ceaf4280bd0318679c1ad832181f3f449814",
  "isBranch" : true
}
...]
```

Download Tag

Description: Download a complete tarball (tar.gz/zip, default tar.gz) of a tag.

Downloading can be executed conditionally according to properties by specifying the properties query param. In this case only cached artifacts are searched.

Since: 3.6.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/vcs/downloadTag/{repoKey}/{userOrg}/{repo}/{tag-name}?ext=tar.gz/zip (default tar.gz)

Produces: application/octet-stream

Sample Output:

```
GET /api/vcs/downloadTag/github/jquery/jquery/2.0.1
<Tag binary content>
```

Download File within a Tag

Description: Download a specific file from within a tag.

Since: 3.6.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/vcs/downloadTagFile/{repoKey}/{userOrg}/{repo}/{tag-name}!{file-path}

Produces: application/octet-stream

Sample Output:

```
GET /api/vcs/downloadTagFile/github/jquery/jquery/2.0.1!AUTHORS.txt
<AUTHORS.txt content>
```

Download Branch

Description: Downloads a tarball (tar.gz/zip, default tar.gz) of a complete branch.

Downloading can be executed conditionally according to properties by specifying the properties query param. In this case only cached artifacts are searched.

Since: 3.6.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/vcs/downloadBranch/{repoKey}/{userOrg}/{repo}/{branch-name}?ext=tar.gz/zip[&properties=qa=approved]

Produces: application/octet-stream

Sample Output:

```
GET /api/vcs/downloadBranch/github/jquery/jquery/master  
<Branch binary content>
```

Download File within a Branch

Description: Downloads a specific file from within a branch.

Since: 3.6.0

Security: Requires a privileged user (can be anonymous)

Usage: GET /api/vcs/downloadBranchFile/{repoKey}/{userOrg}/{repo}/{branch-name}!{file-path}

Produces: application/octet-stream

Sample Output:

```
GET /api/vcs/downloadBranchFile/github/jquery/jquery/master!README.md  
<AUTHORS.txt content>
```

Download Release

Description: Downloads a complete release tarball (tar.gz/zip, default tar.gz) of a tag from GitHub.

Since: 4.3.0

Security: Requires a privileged user (can be anonymous)

VCS Usage: GitHub only

Usage: GET /api/vcs/downloadRelease/{repoKey}/{userOrg}/{repo}/{release-name}?ext=tar.gz/zip (default tar.gz)

Produces: application/octet-stream

Sample Output:

```
GET /api/vcs/downloadRelease/git-remote/google/protobuf/v3.0.0-beta-1?ext=tar.gz/zip  
<Tag binary content>
```

Examples

Below are some examples of working with the API using cURL:

Download jquery master branch from GitHub

```
curl -i  
"http://localhost:8080/artifactory/api/vcs/downloadBranch/github/jquery/jquery/master"
```

Download a specific tag from Bitbucket

```
curl -i  
"http://localhost:8080/artifactory/api/vcs/downloadTag/bitbucket/1systems/angular-exte  
nded-notifications/1.0.0"
```

Download a file within the tag 2.0.1 of jquery, '!' is escaped as '%21'

```
curl -i  
"http://localhost:8080/artifactory/api/vcs/downloadTag/github/jquery/jquery/2.0.1%21AU  
THORS.txt"
```

When files are already cached, you can conditionally request them using a properties query param:

Download a file within the tag 2.0.1 of jquery, '!' is escaped as '%21'

```
curl -i  
"http://localhost:8080/artifactory/api/vcs/downloadBranch/github/jquery/jquery/2.0.1?p  
roperties=qa=approved"
```

Accessing Private VCS Repositories

Artifactory also supports accessing private VCS repositories such as a private GitHub or any self-hosted authenticated one.

To do so, simply add your credentials under **Advanced Settings** of the remote repository configuration panel.

Credentials when redirected

Some git providers (GitHub included) redirects download requests to a CDN provider.

You will need your credentials to pass along with the redirected request, simply check the **Lenient Host Authentication** and the credentials will pass transparently on each redirected request.

YUM Repositories

Overview

Artifactory is a fully-fledged YUM local repository. As such, it enables:

1. RPM metadata calculation for RPMs hosted in Artifactory local repositories.
2. Provisioning RPMs directly from Artifactory to YUM clients.
3. Detailed RPM metadata views from Artifactory's web UI.

RPM Metadata for Hosted RPMs

The RPM metadata generated by Artifactory is identical to the basic-mode output of the Red Hat-based Linux command [createrepo](#).

A folder named `repodata` is created in the configured location within a local repository with the following files in it:

File	Description
<code>primary.xml.gz</code>	Contains an XML file describing the primary metadata of each RPM archive.

<i>filelists.xml.gz</i>	Contains an XML file describing all the files contained within each RPM archive.
<i>other.xml.gz</i>	Contains an XML file describing miscellaneous information regarding each RPM archive.
<i>repomd.xml</i>	Contains information regarding all the other metadata files.

YUM Support is Platform Independent!

Artifactory's RPM metadata calculation is based on **pure Java**.

It does not rely on the existence of the `createrepo` binary or on running external processes on the host on which Artifactory is running.

Page Contents

- Overview
 - RPM Metadata for Hosted RPMs
 - Triggering RPM Metadata Updates
- Configuration
 - Local Repositories
 - Remote Repositories
- Using Yum to install RPM packages from Artifactory
- YUM Groups
 - Attaching a YUM Group
 - YUM Group Commands
 - Setting Group Properties
- Yum Authentication
 - Proxy Server Settings
 - SSL Setting
- Using Yum Variables
- Creating A GPG Key and Signing RPMs
 - Creating a Public/Private GPG Key Pair
 - Exporting Private/Public Keys for Safe Keeping
- Viewing Individual RPM Information
 - Metadata Fields as Properties
- Watch the Screencast

Triggering RPM Metadata Updates

When enabled, the metadata calculation is triggered automatically by some actions, and can also be invoked manually by others. Either way, the metadata produced is served to YUM clients.

Automatic

RPM metadata is automatically calculated:

1. When deploying/removing/copying/moving an RPM file.
2. When performing content import (both system and repository imports).

Manual

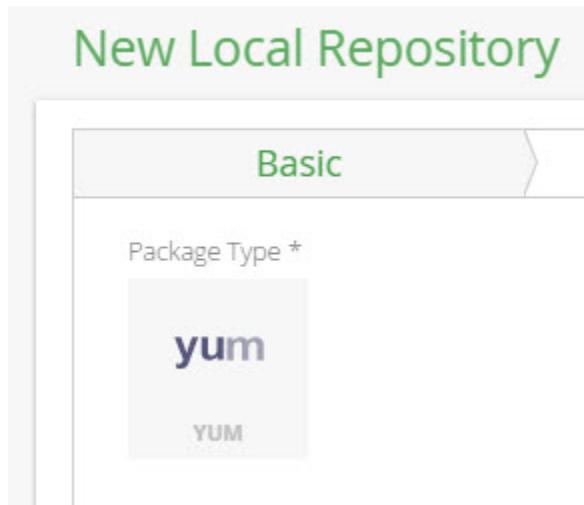
You can manually invoke RPM metadata calculation:

1. By selecting the local repository in the Tree Browser and clicking **Recalculate Index** in the **Actions** menu.
2. Via Artifactory's [REST-API](#).

Metadata calculation cleans up YUM metadata that already existed as a result of manual deployment or import. This includes RPM metadata stored as SQLite database files.

Configuration

To create a YUM local repository, select **YUM** as the **Package Type** when you create the repository.



Local Repositories

To enable automatic RPM metadata calculation on a local YUM repository, in the **YUM Settings** section of the **Basic** settings screen, set **Auto-calculate YUM Metadata**.

YUM Settings

YUM Metadata Folder Depth	YUM Group File Names
<input type="text" value="0"/>	<input type="text" value="groups.xml"/>
<input checked="" type="checkbox"/> Auto Calculate YUM Metadata	
Field	Description

YUM Metadata Folder Depth	<p>Informs Artifactory under which level of directory to search for RPMs and save the <code>repodata</code> directory.</p> <p>By default this value is 0 and refers to the repository's root folder. In this case, Artifactory searches the entire repository for RPMs and saves the <code>repodata</code> directory at <code>\$REPO-KEY/repodata</code>.</p> <p>Using a different depth is useful in cases where generating metadata for a repository separates its artifacts by name, version and architecture.</p> <p>For example:</p> <p>If the repository layout is similar to that shown below and you want to generate RPM metadata for every artifact divided by name, set the Depth to 1 and the <code>repodata</code> directory is saved at <code>REPO_ROOT/ARTIFACT_NAME/repodata</code>:</p> <pre style="border: 1px solid black; padding: 10px;"> REPO_ROOT/\$ARTIFACT_NAME/\$ARTIFACT_VERSION/\$ARCHITECTURE/FILE_NAME - or - libs-release-local/foo/1.0/x64/foo-1.0-x64.rpm </pre>
YUM Group File Names	<p>A comma-separated list of YUM group files associated with your YUM packages.</p> <p>Note that at each level (depth), the <code>repodata</code> directory in your repository may contain a different group file name, however each <code>repodata</code> directory may contain only 1 group metadata file (multiple groups should be listed as different tags inside the XML file. For more details, please refer to the YUM Documentation).</p>
Auto-calculate YUM Metadata	<p>When set, YUM metadata calculation is automatically triggered by the actions described above.</p>

Metadata calculation is asynchronous and does not happen immediately when triggered, whether automatically or manually.

Artifactory invokes the actual calculation only after a certain "quiet period", so the creation of metadata normally occurs only 1-2 minutes after the calculation was triggered.

Remote Repositories

Artifactory remote repositories support YUM out-of-the-box, and there is no need for any special configuration needed in order to work with RPMs in a remote repository.

All you need to do is point your YUM client at the remote repository, and you are ready to use YUM with Artifactory.

To define a remote repository to proxy a YUM remote repository, follow the steps below:

1. In the **Admin** module under **Repositories | Remote**, click "New" to create a new remote repository.
2. Set the **Repository Key** value, and specify the URL to the remote repository in the **URL** field as displayed below.

New Remote Repository

Basic Advanced Replications

Package Type *

yum

YUM

Repository Key * URL *

targetCentos http://mirror.centos.org/centos/6.6/os/x86_64

Test

3. Click "Save & Finish"
4. Back in the **Artifacts** module, in the **Tree Browser**, select the repository. Note that in the Tree Browser, the repository name is appended with ".cache".
5. Click **Set Me Up** and copy the value of the **baseurl** tag.

Set Me Up

Tool
yum YUM

Repository
targetCentos

Resolve
To resolve .rpms using yum client edit or create the following file with root privileges

```
1 [Artifactory]
2 name=Artifactory
3 baseurl=http://10.100.1.110:8081/artifactory/targetCentos/
4 enabled=1
5 gpgcheck=0
```

Then paste the configuration

```
1 [Artifactory]
2 name=Artifactory
3 baseurl=http://10.100.1.110:8081/artifactory/targetCentos/
4 enabled=1
5 gpgcheck=0
```

And execute

6. Next, create the `/etc/yum.repos.d/targetCentos.repo` file and paste the following configuration into it:

```
[targetCentos]
name=targetCentos
baseurl=http://localhost:8081/artifactory/targetCentos/
enabled=1
gpgcheck=0
```

Using Yum to install RPM packages from Artifactory

After configuring the `yum-local` repository in Artifactory, you need to configure your local machine to install software packages from it by executing the following steps:

1. Edit the `artifactory.repo` file with root privileges

```
sudo vi /etc/yum.repos.d/artifactory.repo
```

2. Paste the following configuration into the `artifactory.repo` file:

```
[Artifactory]
name=Artifactory
baseurl=http://localhost:8081/artifactory/yum-local/
enabled=1
gpgcheck=0
```

Now, every RPM file deployed to the root of the `yum-local` repository can be installed using:

```
yum install <package_name>
```

YUM Groups

A YUM group is a set of RPM packages collected together for a specific purpose. For example, you might collect a set of "Development Tools" together as a YUM group.

A group is specified by adding a group XML file to same directory as the RPM packages included in it. The group file contains the metadata of the group including pointers to all the RPM files that make up the group.

Artifactory supports attaching a [YUM Group file](#) to the YUM calculation essentially mimicking the `createrepo -g` command.

A group file can also be created by running the following command:

```
sudo yum-groups-manager -n "My Group" --id=mygroup --save=mygroups.xml --mandatory yum
glibc rpm
```

Attaching a YUM Group

The process of attaching YUM group metadata to a local repository is simple:

1. Create an XML file in the groups format used by YUM. You can either just type it out manually using any text editor, or run the `yum-groups-manager` command from `yum-utils`.
2. Deploy the created group file to the `repodata` folder.
Artifactory will automatically perform the following steps:
 - Create the corresponding `.gz` file and deploy it next to the deployed group XML file.
 - Invoke a YUM calculation on the local repository.
 - Attach the group information (both the XML and the `.gz` file) to the `repomd.xml` file.
3. Make sure the group file names are listed in the **YUM Group File Names** field of the Packages tab. This tells Artifactory which files should be attached as repository group information.

YUM Group Commands

The following table lists some useful YUM group commands:

Command	Description
<code>yum groupinstall <Group ID></code>	Install the YUM group. The group must be deployed to the root of the YUM local repository.

<code>yum groupremove <Group ID></code>	Remove the RPM group
<code>yum groupupdate <Group ID></code>	Update the RPM group. The group must be deployed to the root of the YUM local repository.
<code>yum groupinfo <Group ID></code>	List the RPM packages within the group.
<code>yum grouplist more</code>	List the YUM groups

Setting Group Properties

YUM group properties can be set in the `/etc/yum.config` file as follows:

Setting	Allowed values	Description
overwrite_groups	0 or 1	<p>Determines YUM's behavior if two or more repositories offer package groups with the same name.</p> <p>If set to 1 then the group packages of the last matching repository will be used.</p> <p>If set to 0 then the groups from all matching repositories will be merged together as one large group.</p>
groupremove_leaf_only	0 or 1	<p>Determines YUM's behavior when the groupremove command is run.</p> <p>If set to 0 (default) then all packages in the group will be removed.</p> <p>If set to 1 then only those packages in the group that aren't required by another package will be removed.</p>

enable_group_conditionals	0 or 1	Determines whether YUM will allow the use of conditionals packages. If set to 0 then conditionals are not allowed If set to 1 (default) package conditionals are allowed.
group_package_types	optional, default, mandatory	Tells YUM which type of packages in groups will be installed when <code>groupinstall</code> is called. Default is: default, mandatory

Yum Authentication

Proxy Server Settings

If your organization uses a proxy server as an intermediary for Internet access, specify the `proxy` settings in `/etc/yum.conf`. If the proxy server also requires authentication, you also need to specify the `proxy_username`, and `proxy_password` settings.

```
proxy=<proxy server url>
proxy_username=<user>
proxy_password=pass
```

If you use the yum plugin (`yum-rhn-plugin`) to access the ULN, specify the `enableProxy` and `httpProxy` settings in `/etc/sysconfig/rhn/up2date`. In addition, If the proxy server requires authentication, you also need to specify the `enableProxyAuth`, `proxyUser`, and `proxyPassword` settings as shown below.

```
enableProxy=1
httpProxy=<proxy server url>
enableProxyAuth=1
proxyUser=<user>
proxyPassword=<password>
```

SSL Setting

YUM supports SSL from version 3.2.27.

To secure a repository with SSL, execute the following steps:

- Generate a private key and certificate using [OpenSSL](#).
- Define your protected repository in a `.repo` file as follows:

```
[protected]
name = SSL protected repository
baseurl=<secure repo url>
enabled=1
gpgcheck=1
gpgKey=<URL to public key>
sslverify=1
sslclientcert=<path to .cert file>
sslclientkey=<path to .key file>
```

where:

gpgkey is a URL pointing to the ASCII-armored GPG key file for the repository . This option is used if YUM needs a public key to verify a package and the required key has not been imported into the RPM database.

If this option is set, YUM will automatically import the key from the specific URL. You will be prompted before the key is installed unless the **assumeyes** option is set.

Using Yum Variables

You can use and reference the following built-in variables in `yum` commands and in all YUM configuration files (i.e. `/etc/yum.conf` and all `.repo` files in the `/etc/yum.repos.d/` directory):

Variable	Description
<code>\$releasever</code>	This is replaced with the package's version, as listed in <code>distroverpkg</code> . This defaults to the version of the <code>redhat-release</code> package.
<code>\$arch</code>	This is replaced with your system's architecture, as listed by <code>os.uname()</code> in Python.
<code>\$basearch</code>	This is replaced with your base architecture. For example, if <code>\$arch=i686</code> then <code>\$basearch=i386</code>

The following code block is an example of how your `/etc/yum.conf` file might look:

```
[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=3
[comments abridged]
```

Creating A GPG Key and Signing RPMs

In order to create a GPG Key for signing RPMs ,use the **gnupg** package and gpg utilities (notice that gnupg2 will not work with these steps).

Creating a Public/Private GPG Key Pair

The following commands create a Private/Public GPG Key Pair (Note you should actually login to console/ssh with this user, not sudo to it):

```
$ gpg --gen-key
```

You will be asked a series of questions, answer as follows (answers in italic):

- Please select what kind of key you want: (*1) DSA and ElGamal (default*)
- What keysize do you want? *2048*
- Key is valid for? *0 (key does not expire)*
- Real name: *John Doe*
- Email Address: *jdoe@example.com*
- Comment: *RPM Development*

Generating GPG keys

It is not recommended to generate private/public key pair on your production systems because they are exposed to the public internet and are therefore more vulnerable to being exposed.

Generating the GPG key may take some time, and once complete will display the following output:

```
pub 1024D/454CBFAB 2009-12-30
Key fingerprint = 2D38 3000 DD7D 68C0 45D3 76C0 F08E B2EE 454C BFB
uid John Doe (RPM Developer) <jdoe@example.com>
sub 2048g/767143C3 2009-12-30
```

The ID of your public key is displayed after the first line after the slash ('/') character. In the above example, the ID of the public key is 454CBFAB.

Exporting Private/Public Keys for Safe Keeping

You must export the keys so that you can save them in a secure location (and also, so that they can be imported in the future). To list the keys you have installed execute the following command:

```
$ gpg --list-secret-keys
/home/you/.gnupg/secring.gpg
-----
sec 1024D/454CBFAB 2009-12-30
uid John Doe (RPM Developer) <jdoe@example.com>
sub 2048g/767143C3 2009-12-30
```

Exporting the Public GPG Key

In order to use the GPG Key with RedHat Network (RHN), you will need to make the Public portion available in a Public GPG Key by executing the following commands:

```
$ gpg --export-secret-key -a 454CBFAB > MYORG-GPG-KEY.private
$ gpg --export -a 454CBFAB > MYORG-GPG-KEY.public
```

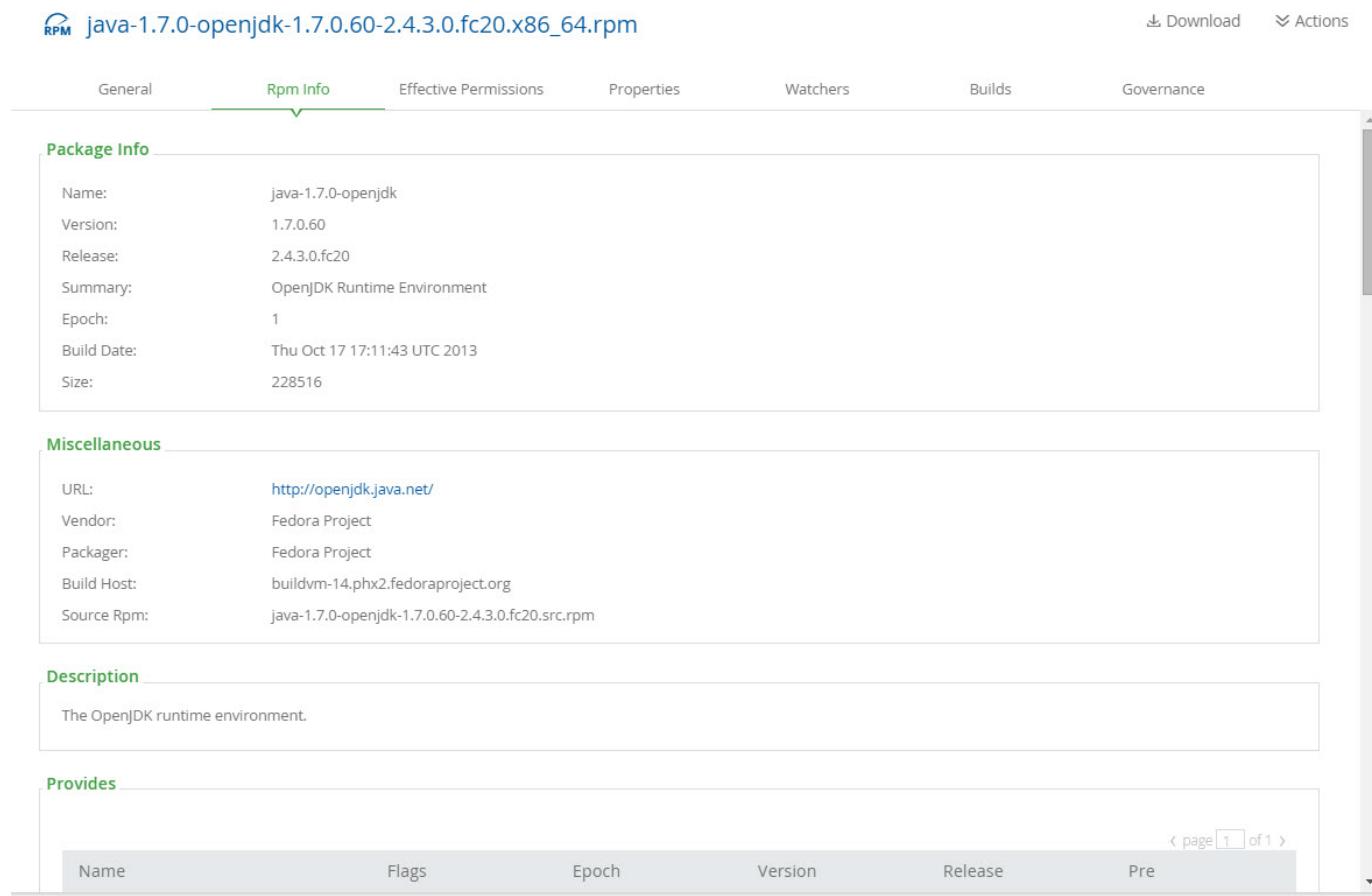
The **Public Key** is the only portion that should ever be made public. The Private Key must be kept secure, preferably on a system that does not allow outside access from the public internet. It is also recommended to make a hard copy (burned on a CD/USB key) of the Public/Private key pair and keep that somewhere secure like a Safe Deposit Box. Anyone who obtains your Private/Public key pair can sign RPMs as if they were signed by you and potentially perform malicious acts in your name.

You will need to make the public key available, and a common way to do this is to upload it to your web server (usually in the `/pub` folder or root of your repository).

For this document we will assume that the MYORG-GPG-KEY is available at the following URL: <http://<Domain>/<root folder>/MYORG-GPG-KEY>

Viewing Individual RPM Information

You can view all the metadata that annotates an RPM by choosing it in Artifactory's tree browser and selecting the **RPM Info** tab:



The screenshot shows the Artifactory interface for viewing an RPM artifact. At the top, there is a navigation bar with a 'RPM' icon, the artifact name 'java-1.7.0-openjdk-1.7.0.60-2.4.3.0.fc20.x86_64.rpm', and download/actions buttons. Below the navigation bar is a tabs menu with 'General', 'Rpm Info' (which is selected and highlighted in green), 'Effective Permissions', 'Properties', 'Watchers', 'Builds', and 'Governance'. The main content area is divided into several sections: 'Package Info' (containing details like Name: java-1.7.0-openjdk, Version: 1.7.0.60, Release: 2.4.3.0.fc20, Summary: OpenJDK Runtime Environment, Epoch: 1, Build Date: Thu Oct 17 17:11:43 UTC 2013, Size: 228516); 'Miscellaneous' (containing URL: http://openjdk.java.net/, Vendor: Fedora Project, Packager: Fedora Project, Build Host: buildvm-14.phx2.fedoraproject.org, Source Rpm: java-1.7.0-openjdk-1.7.0.60-2.4.3.0.fc20.src.rpm); 'Description' (containing 'The OpenJDK runtime environment.'); and 'Provides' (a table showing columns: Name, Flags, Epoch, Version, Release, Pre, with one row visible). A page navigation bar at the bottom right of the 'Provides' section indicates 'page 1 of 1'.

Metadata Fields as Properties

The corresponding RPM metadata fields are automatically added as properties of an RPM artifact in YUM repositories accessed through Artifactory:

- rpm.metadata.name
- rpm.metadata.arch
- rpm.metadata.version
- rpm.metadata.release
- rpm.metadata.epoch
- rpm.metadata.group
- rpm.metadata.vendor
- rpm.metadata.summary

Properties can be used for searching and other functions. For more details please refer to [Properties](#).

Watch the Screencast

Watch this short screencast to learn how easy it is to host RPMs in Artifactory.

Debian Repositories

Overview

From version 3.3, on top of Artifactory's [existing support](#) for advanced artifact management, Artifactory supports Debian repositories whether they use the [current Automatic](#) Debian architecture or the deprecated [Trivial](#) architecture. As a fully-fledged Debian repository, Artifactory generates index files that are fully compliant with Debian clients.

Artifactory support for Debian provides:

1. The ability to provision Debian packages from Artifactory to a Debian client from local and remote repositories.
2. Calculation of Metadata for Debian packages hosted in Artifactory's local repositories.
3. Access to remote Debian resources (such as `us.archive.ubuntu.com`) through [Remote Repositories](#) which provide the usual proxy and caching functionality.
4. Providing GPG signatures that can be used by Debian clients to verify packages.
5. Complete management of GPG signatures using the Artifactory UI and the REST API.

Page Contents

- [Overview](#)
- [Configuration](#)
 - [Local Repositories](#)
 - [Deploying a package using the UI](#)
 - [Deploying a package using Matrix Parameters](#)
 - [Setting the Target Path](#)
 - [Specifying multiple layouts](#)
 - [Remote Repositories](#)
- [Signing Debian Packages](#)
 - [Generating Keys](#)
 - [Uploading Keys](#)
 - [Downloading the Public Key](#)
- [Authenticated Access to Servers](#)
- [REST API Support](#)

Configuration

You can only deploy Debian packages to a local repository that has been created with the Debian [Package Type](#).

You can download packages from a local or a remote Debian repository.

Local Repositories

To create a new local repository that supports Debian, under the **Basic** settings, set the **Package Type** to be **Debian**.

If you are using Debian with a *Trivial* layout, in the **Debian Settings** section, set the **Trivial Layout** checkbox.

New Local Repository

Basic Advanced Replications

Package Type *

 debian
Debian

Repository Key *

General

Repository Layout

simple-default

Debian Settings

Trivial Layout

Deploying a package using the UI

To deploy a Debian package to Artifactory, in the Artifactory Repository Browser, click **Deploy**.

Select your Debian repository as the **Target Repository**, upload the file you want to deploy.

Deploy

Target Repository

debian-local

Package Type: Debian

Type: **Single** Multi

Uploading planckdb-08-2015.deb

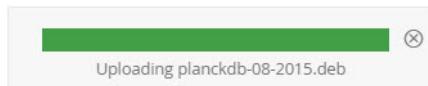
Target Path

pool/planckdb-08-2015.deb

Check the **Deploy as Debian Artifact** checkbox and fill in the **Distribution**, **Component** and **Architecture** fields in the **Debian Artifact** section. Notice that the **Target Path** is automatically updated to reflect your input.

Deploy

Type: **Single** Multi



Target Path

```
pool/planckdb-08-2015.deb;deb.distribution=trusty;d
```

Deploy as Debian Artifact

Debian Artifact

Distribution

```
trusty
```

Component

```
main
```

Architecture

```
amd64,i386
```

Deploy

Setting the target path manually? Be careful with spaces

We recommend using the fields in the **Debian Artifact** section to set your **Target Path**. Nevertheless, if you choose to specify the **Target Path** manually, make sure you don't enter any superfluous spaces.

For example to upload package **planckdb-08-2015.deb**, and specify that its layout is from the **trusty** distribution, in the **main** component and the **i386** architecture, you would enter:

```
pool/planckdb-08-2015.deb;deb.distribution=trusty;deb.component=main;deb.architecture=i386
```

You can also deploy Debian packages to Artifactory with an explicit URL using [Matrix Parameters](#).

After you deploy the artifact, you need to wait about one minute for Artifactory to recalculate the repository index and display your upload in the Repository Browser.

Once you have deployed your Debian package, and Artifactory has recalculated the repository index, your repository should be organized as displayed below:

The screenshot shows the Artifactory interface for managing artifacts. On the left, there's a tree view of the repository structure under 'debian-local'. A specific file, 'planckdb-08-2015.deb', is selected in the 'pool' directory. On the right, detailed information about this file is displayed in a card-like interface.

General	Effective Permissions	Properties	Watchers	Builds	Governance
Name: planckdb-08-2015.deb Repository Path: debian-local/pool/planckdb-08-2015.deb Module ID: N/A Deployed by: admin Size: 75.32 KB Created: 01-10-14 09:56:41 UTC Last Modified: 01-10-14 09:55:37 UTC Licenses: Not Found Query Code Center Downloaded: 1 Last Downloaded: 23-07-15 13:17:41 UTC Last Downloaded By: _system_					
Checksums SHA-1: c8d852a7f3e15c4f87f7a18f7f56c807ab2b295e (Uploaded: Identical) MD5: 692315a711ffda1e672693e076182555 (Uploaded: Identical)					

Deploying a package using Matrix Parameters

The URL is built similarly to the **Target Path** format as follows:

Deploying a package using Matrix Parameters

```
PUT
"http://$ARTIFACTORY_HOME/{debianRepoKey}/pool/{debianPackageName};deb.distribution={distribution};deb.component={component};deb.architecture={architecture}"
```

For example, to upload package **libatk1.0_i386.deb**, and specify that its layout is from the **wheezy** distribution, in the **main** component and the **i386** architecture, you would enter:

Example

```
PUT
"http://localhost:8080/artifactory/debian-local/pool/libatk1.0_i386.deb;deb.distribution=wheezy;deb.component=main;deb.architecture=i386"
```

Setting the Target Path

The **Target Path** needs to be entered in a strict and specific format that uses system properties to define where the artifact will be stored and its specific layout as follows:

Target Path Format

```
[path];deb.distribution=[distribution];deb.component=[component];deb.architecture=[architecture]
```

<i>path</i>	The repository path where the package should be stored. Artifactory supports storing Debian packages anywhere within the repository. The examples on this page show Debian packages stored under the pool folder in accordance with the Debian convention.
<i>distribution</i>	The value to assign to the <code>deb.distribution</code> property used to specify the Debian package distribution
<i>component</i>	The value to assign to the <code>deb.component</code> property used to specify the Debian package component name
<i>architecture</i>	The value to assign to the <code>deb.architecture</code> property used to specify the Debian package architecture

Adding Architecture Independent Packages

Uploading a Debian package with `deb.architecture=all` will cause it to appear in the Packages index of all the other architectures under the same Distribution and Component, as well as under a new index branch called `binary-all` which holds all Debian packages that are marked as "all".

Removing an "all" Debian package will also remove it from all other indexes under the same Distribution and Component.

When the last Debian package in an architecture is removed but the Packages index still contains an "all" Debian package, it is preserved in the index.

If you want such an architecture index removed you may do so via the UI or using [Calculate Debian Repository Metadata](#) in the REST API, which cleans up orphaned package files from the index.

Specifying multiple layouts

Whether uploading a package using the UI or Matrix Parameters, you can specify multiple layouts for any Debian package you upload, by including additional values for the distribution, component or architecture separated by a comma,

For example, to upload package **libatk1.0_i386.deb** to both **wheezy** and **trusty** distributions, in both **main** and **contrib** components and both **i386** and **64bit-arm** architectures you would specify the following Target Path to upload using the UI:

Target path for multiple layouts

```
pool/libatk1.0_i386.deb;deb.distribution=wheezy;deb.distribution=trusty;deb.component=main;deb.component=contrib;deb.architecture=i386;deb.architecture=64bit-arm
```

Correspondingly, to upload the file using Matrix Parameters, you would use the following:

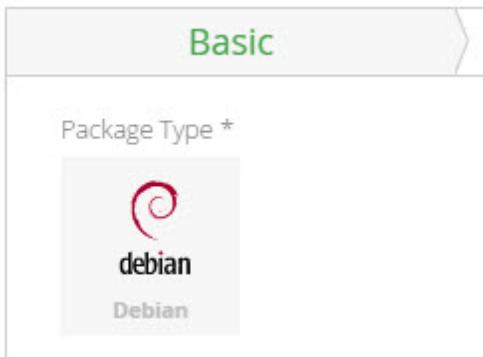
Multiple layouts using Matrix Parameters

```
PUT  
"http://localhost:8080/artifactory/debian-local/pool/libatk1.0_i386.deb;deb.distribution=wheezy;deb.distribution=trusty;deb.component=main;deb.component=contrib;deb.architecture=i386;deb.architecture=64bit-arm"
```

Remote Repositories

You can download Debian packages from Local Debian Repositories as described above, or from Remote Repositories specified as supporting Debian packages.

To specify that a Remote Repository supports Debian packages, you need to set its **Package Type** to **Debian** when it is created.



Note that the index files for remote Debian repositories are stored and renewed according to the [Retrieval Cache Period](#) setting.

Signing Debian Packages

Artifactory manages public and private keys that are used to sign and verify Debian packages.

However, you first need to generate the keys and upload them to Artifactory.

Generating Keys

The way to generate keys is platform dependent.

The example below shows how to generate the public and private keys on Linux:

Generating PGP keys

```
# generate the keys
gpg --gen-key

# list all keys in your system and select the pair you want to use in Artifactory
gpg --list-keys

# resolve the key-id from the lists-keys by selecting the relevant license
pub  2048R/8D463A47 2015-01-19
uid          JonSmith (Jon) <jon.smith@jfrog.com>
key-id = 8D463A47

#export the private key with the specified id to a file
gpg --output {private key file name and path} --armor --export-secret-keys {key-id}

#export the public key with the specified id to a file
gpg --output {public key file name and path} --armor --export {key-id}
```

You also need to specify a pass phrase that must be used together with the Debian Signing Keys. The pass phrase can be saved, or passed in with a REST API call.

Uploading Keys

To upload your Debian signing keys, in the **Admin** tab, go to **Security | Signing Keys**.

Debian Signing Key

Public key : No public key installed

No file chosen

Private key : No private key installed

No file chosen

Pass-phrase

Once you have specified the key file, select the "Upload" button for the corresponding field.

Artifactory will indicate when keys are installed, and you can click on the **Public key is installed** link to download the public key.

If your signing keys were created with a pass-phrase, enter it in the designated field. You can click "Verify" to make sure the pass-phrase matches the uploaded keys.

Click "Save" to save your changes.

Don't forget to click "Save"

To ensure that your Debian signing keys are properly stored in Artifactory's database, you need to click "Save" even if your signing keys do not have a pass-phrase.

Upload your pass-phrase with REST

If you prefer not to upload your pass phrase using the UI, you can set it using the [REST API](#).

Downloading the Public Key

Once you have uploaded your signing keys, you can download your public key whenever needed using the **Public key is installed** link under **Signing Keys Management** in the **Admin** module under **Security | Signing Keys**.

Debian Signing Key

Public key **Public key is installed** (x)

No file chosen (x) Upload

Authenticated Access to Servers

If you need to access a secured Artifactory server that requires a username and password, you can specify these in your Debian `source.list` file by prefixing the artifactory host name with the required credentials as follows:

Accessing Artifactory with credentials

```
http://user:password@$ARTIFACTORY_HOME/{repoKey} {distribution} {components}  
For example:  
http://admin:password@localhost:8081/artifactory/debian-local wheezy main restricted
```

Encrypting your password

You can use your encrypted password as described in [Using Your Secure Password](#).

REST API Support

The Artifactory REST API provides extensive support for Debian signing keys and recalculating the repository index as follows:

- Set the public key
- Get the public key
- Set the private key
- Set the pass phrase
- Recalculate the index

RubyGems Repositories

Overview

Artifactory provides full support for RubyGems repositories including:

- Local repositories with RubyGems API support
- Caching and proxying remote RubyGems repositories
- Virtual repositories that aggregate multiple local and remote repositories including indices of both gems and specifications
- Support for common Ruby tools such as gem and bundler

For general information on configuring Artifactory to work with RubyGems, please refer to [Configuring Repositories](#).

Page Contents

- Overview
- General Configuration
- Local Repositories
 - Usage
- Remote Repositories
 - Usage
- Virtual Repositories
 - Usage
- Using the REST API
- Viewing RubyGems Artifact Information
- Watch the Screencast

General Configuration

All RubyGems repositories must be prefixed with `api/gems` in the path

When using the RubyGems command line to access a repository through Artifactory, the repository URL must be prefixed with `api/gems` in the path.

All RubyGems commands, including `gem source` and `gem push`, must prepend the repository path with `api/gems`.

For example, if you are using Artifactory standalone or as a local service, you would access your RubyGems repositories using the following URL:

`http://localhost:8081/artifactory/api/gems/<repository key>`

Or, if you are using Artifactory Online, the URL would be:

`https://<server name>.artifactoryonline.com/<server name>/api/gems/<repository key>`

Using RubyGems repositories with Artifactory version 3.4.1 and below, and Java 7 update 40 or higher

If you are using RubyGems repositories with Java 7 update 40 or higher, you may receive the following exception:

```
org.jruby.exceptions.RaiseException: (SystemStackError)
stack level too deep
at ...
```

This is due to an [issue](#) with Artifactory's use of JRuby.

If you are using Artifactory version 3.4.1 and below, you need to define the following [System Property](#) and restart Artifactory:

```
jruby.compile.invokedynamic=false
```

In Artifactory version 3.4.2, an enhancement has been implemented to overcome this issue and from this version on, no action is required.

Local Repositories

Local RubyGems repositories are physical, locally-managed repositories into which you can deploy and manage your in-house Gems.

To create a local RubyGems repository, in the **Admin** module, under **Repositories | Local**, click "New" and set **RubyGems** to be the **Package Type**.

New Local Repository

Basic

Package Type *



Gems

You can set up a local RubyGems repository as follows:

You need to add the repository source URL by modifying your `~/.gemrc` file or using the `gem source` command as displayed below. You can extract the source URL by selecting the repository in the **Tree Browser** and clicking **Set Me Up**.

Set Me Up

Tool

Gems

Repository

gem1

General

In order for your client to upload and download Gems from Artifactory repository you will need the it to your `~/.gemrc` file using the following command

```
1 gem source -a http://<USERNAME>:<API_KEY>@http://10.100.1.110:8081/artifactory/api/gems/gem1/
```

In case anonymous access is enabled you can also use the following

```
1 gem source -a http://10.100.1.110:8081/artifactory/api/gems/gem1/
```

You can view the order of resolution by running

```
1 gem source --list
```

Make sure that Artifactory repository is at the top of the list. It is recommended to use virtual repository to encapsulate both local and remote repositories under a single URL.

If you want to setup the credentials for your gem tool either include your `API_KEY` in the `~/.gems/credentials`, or run the

Notice that there are two sources. First, the remote proxy, then the local one. This will effectively allow you to retrieve Gems from both of them in the specified order.

All RubyGems repositories must be prefixed with api/gems in the path

When using the RubyGems command line to access a repository through Artifactory, the repository URL must be prepended with **api/gems** in the path.

```
gem source -a http://localhost:8081/artifactory/api/gems/my-gems-local/
```

Usage

First, setup the appropriate **credentials** for the **gem** tool: either include the API key in the `~/.gem/credentials` file or issue this command:

Setting Up Credentials

```
curl http://localhost:8081/artifactory/api/gems/<repository key>/api/v1/api_key.yaml  
-u admin:password > ~/.gem/credentials
```

Running on Linux

You may need to change the permissions of the credentials file to 600 (`chmod 600`)

Running on Windows

The credentials file is located under `%USERPROFILE%/.gem/credentials`

You also need to set the API key encoding to be "ASCII". For example:

```
curl http://localhost:8081/artifactory/api/gems/<repository key>/api/v1/api_key.yaml | Out-File ~/.gem/credentials -Encoding "ASCII"
```

API keys

You can modify the credentials file manually and add different API keys. You can later use `gem push -k key` to choose the relevant API key.

In order to **push** gems to the local repository, you can set the global variable `$RUBYGEMS_HOST` to point to the local repository as follows:

Setting RUBYGEMS_HOST

```
export RUBYGEMS_HOST=http://localhost:8081/artifactory/api/gems/<repository key>
```

To get this value, select your repository in the Tree Browser and click **Set Me Up**.

Deploy

In order to push gems to this repository, you can set the global variable `$RUBYGEMS_HOST` to point to it as follows:

```
1 export RUBYGEMS_HOST=http://localhost:8080/artifactory/api/gems/gems-local
```

You can also specify the target repository when pushing the gem by using the `--host` option:

```
1 gem push <PACKAGE> --host http://localhost:8080/artifactory/api/gems/gems-local
```

Alternatively you could use the `gem push` command with `--host`, and optionally, `--key` to specify the relevant API key.

Make sure you are familiar with your effective sources and their order as specified in the `~/.gemrc` file.

Also, make sure you are familiar with your global `$RUBYGEMS_HOST` variable before you issue a `gem push` command or use the `--host` option.

When a local repository is first created, it is populated with `rubygems-update-* .gem` by default. In order to disable this behavior, you must change the [System Properties](#) to include:
`artifactory.gems.gemsAfterRepoInitHack=false`

Remote Repositories

A remote RubyGems repository serves as a caching proxy for a repository managed at a remote URL such as <http://rubygems.org>.

Once requested, artifacts (Gems) are cached on demand. They can then be removed, but you cannot manually deploy anything into a remote repository.

To create a remote RubyGems repository, execute the following steps:

1. in the **Admin** module, under **Repositories | Remote**, click "New" and set **RubyGems** to be the **Package Type**.
2. Set the **Repository Key**, and specify the **URL** to the remote repository. The example below references rubygems.org.

New Remote Repository

Basic Advanced Replications

Package Type *

 Gems

Repository Key *

URL *

Test

Usage

In order to allow the integration with the gem command line tool, you must add the relevant source URL to your RubyGems configuration.

1. In the [Tree Browser](#), select your newly created repository and click **Set Me Up**.
2. Copy the source URL from the **RubyGems Sources** section.

Set Me Up

X

Tool

Gems

Repository

gem1

General

In order for your client to upload and download Gems from Artifactory repository you will need the it to your `~/.gemrc` file using the following command

```
1 gem source -a http://<USERNAME>:<API_KEY>@http://10.100.1.110:8081/artifactory/api/gems/gem1/
```

In case anonymous access is enabled you can also use the following

```
1 gem source -a http://10.100.1.110:8081/artifactory/api/gems/gem1/
```

You can view the order of resolution by running

```
1 gem source --list
```

Make sure that Artifactory repository is at the top of the list. It is recommended to use virtual repository to encapsulate both local and remote repositories under a single URL.

If you want to setup the credentials for your gem tool either include your `API_KEY` in the `~/.gems/credentials`, or run the

3. Add this URL by modifying your `~/.gemrc` file or using the `gem source` command as follows:

All RubyGems repositories must be prefixed with `api/gems` in the path

When using the RubyGems command line to access a repository through Artifactory, the repository URL must be prefixed with `api/gems` in the path.

```
gem source -a http://localhost:8081/artifactory/api/gems/rubygems/
```

Additional Gem Commands You Can Use

You can remove previous source entries by modifying your `~/.gemrc` file manually or by running `gem sources -r`
You can also run `gem sources --list` to know what your effective sources are and their order.

Overcoming Unauthorized Status Failures

Some gem/bundler commands may fail with an Unauthorized (401) status. In some cases these can be overcome by using one of the following options:

- Enable the "Anonymous User" by checking **Allow Anonymous Access** in **Security General Configuration** as described in [Managing Users](#).
- Create a specialized [Permission Target](#) that allows anonymous access only to the remote repository.
- Use a source URL with embedded credentials, such as: `gem sources -a http://user:password@host/...`

Virtual Repositories

A Virtual RubyGems repository (or "repository group") can aggregate multiple local and remote RubyGems repositories, seamlessly exposing them under a single URL.

The repository is virtual in that you can resolve and retrieve artifacts from it but you cannot deploy anything to it. For more information please refer to [Virtual Repositories](#).

The procedure for setting up a virtual repository is very similar to setting up a local or remote repository, however as a last step, you need to select the repositories that will be included in the virtual repository you are creating.

Repositories

Filter...

Available Repositories

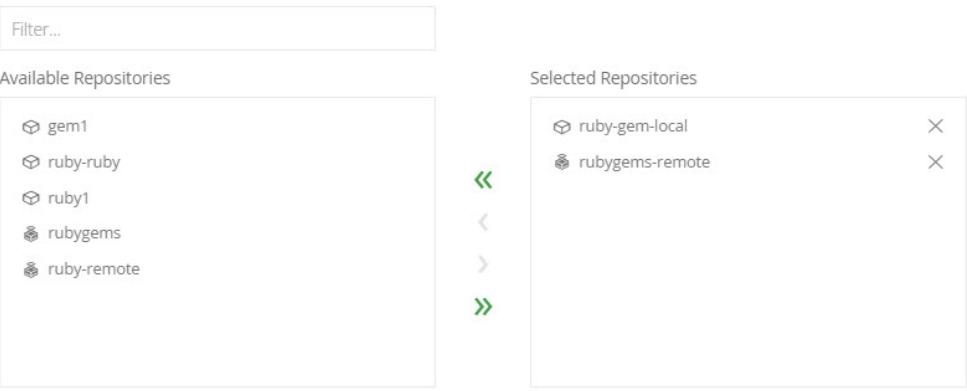
- gem1
- ruby-ruby
- ruby1
- rubygems
- ruby-remote

Selected Repositories

- ruby-gem-local
- rubygems-remote

Included Repositories

- ruby-gem-local
- rubygems-remote



Usage

Using a virtual RubyGems repository you can aggregate both your local and remote repositories.

You need to set the right repository source URL, in the same way as described in [Usage](#) for a local RubyGems repository, just with the appropriate repository key as follows:

source: `http://localhost:8081/artifactory/api/gems/<repository key>/`

target: `http://localhost:8081/artifactory/api/gems/<repository key> (no slash at the end!)`

Using the REST API

The REST API provides access to the Gems Add-on through the repository key using the following URL:

`http://localhost:8081/artifactory/api/gems/<repository key>/`

In addition to the basic binary repository operations, such as download, deploy, delete etc., the following [RubyGems.org](#) API Gem commands are supported:

Gem command	Curl syntax example	Remarks
Info	<code>curl http://localhost:8081/artifactory/api/gems/<repository key>/api/v1/gems/my_gem.(json xml yaml)</code>	Optionally indicate JSON / XML / YAML (default: JSON)

Search	<code>curl http://localhost:8081/artifactory/api/gems/<repository key>/api/v1/search.(json xml yaml)?query=[query]</code>	Will search for gems with name containing query
Dependencies	<code>curl http://localhost:8081/artifactory/api/gems/<repository key>/api/v1/dependencies?gems=[gem1,...]</code>	Use a csv of gem names for the value of gems
Yank	<code>curl -X DELETE http://localhost:8081/artifactory/api/gems/<repository key>/api/v1/yank -d 'gem_name=gem_name' -d 'version=0.0.1' -d 'platform=ruby'</code>	Deletes the specific gem file from the repository

Indexing is done automatically by Artifactory in the background, however if you still need to recreate or update the spec index files, the following REST API commands are also available:

REST command	Curl syntax example	Remarks
ReIndex	<code>curl -X POST http://localhost:8081/artifactory/api/gems/<repository key>/reindex</code>	Re-creates all spec index files.
Update index	<code>curl -X POST http://localhost:8081/artifactory/api/gems/<repository key>/updateIndex</code>	Updates all spec index files if needed.

For full details please refer to [Artifactory REST API](#).

Viewing RubyGems Artifact Information

If you select a RubyGems artifact in the Tree Browser you can select the **RubyGems** tab to view detailed information on the selected artifact.

[General](#)[RubyGems](#)[Effective Permissions](#)[Properties](#)[Watchers](#)[Builds](#)[Governance](#)

Package Info

Name :	rubygems-update
Version:	2.0.6
Platform:	ruby
Summary:	RubyGems is a package management framework for Ruby
Authors:	Jim Weirich, Chad Fowler, Eric Hodel
Homepage:	http://rubygems.org
Repository Path:	ruby-gem-local:gems/rubygems-update-2.0.6.gem ⓘ
Description:	RubyGems is a package management framework for Ruby. This gem is an update for the RubyGems software. You must have an installation of RubyGems before this update can be applied. See Gem for information on RubyGems (or 'ri Gem') To upgrade to the latest RubyGems, run: \$ gem update --system # you might need to be an administrator or root See UPGRADE.rdoc for more details and alternative instructions. ---- If you don't have RubyGems installed, you can still do it manually: * Download from: https://rubygems.org/pages/download * Unpack into a directory and cd there * Install with: ruby setup.rb # you may need admin/root privilege For more details and other options, see: ruby setup.rb --help

Dependencies

Name	Version	Type
minitest	~> 5.0	Development
rdoc	~> 4.0	Development
builder	~> 2.1	Development
hoe-seattlerb	~> 1.2	Development
ZenTest	~> 4.5	Development
rake	~> 0.9.3	Development
hoe	~> 3.7	Development

Watch the Screencast

PyPI Repositories

Overview

Artifactory fully supports PyPI repositories on top of Artifactory's [existing support](#) for advanced artifact management.

Artifactory support for PyPI provides:

1. The ability to provision PyPI packages from Artifactory to the `pip` command line tool from all repository types.
2. Calculation of Metadata for PyPI packages hosted in Artifactory's local repositories.
3. Access to remote PyPI repositories (such as <https://pypi.python.org/>) through Remote

- Repositories which provide proxy and caching functionality.
4. The ability to access multiple PyPI repositories from a single URL by aggregating them under a [Virtual Repository](#).
 5. Compatibility with the `setuptools` and its predecessor `distutils` libraries for uploading PyPI packages.

Configuration

Local Repositories

To create a new PyPI local repository, in the [New Local Repository](#) screen, set the **Package Type** to **PyPI**.

New Local Repository

Basic Advanced

Package Type *



Pypi

Repository Key *

Page Contents

- Overview
- Configuration
 - Local Repositories
 - Remote Repositories
 - Virtual Repositories
- Resolving from Artifactory Using PIP
 - Specifying the Repository on the Command Line
 - Using Cred ential s
 - Using a Configuration File
 - Using a Requirements File
- Publishing to Artifactory
 - Using `distutils` or `setuptools`
 - Creat e the \$HO ME/.p ypirc File
 - Uploa ding
 - Publishing Manually Using the Web UI or REST
- Searching for PyPI Packages

- Using PIP
- Artifactory Search
- Viewing Metadata of PyPI Packages
- Watch the Screencast

Remote Repositories

A Remote Repository defined in Artifactory serves as a caching proxy for a registry managed at a remote URL such as <https://pypi.python.org/>.

Artifacts (such as .whl files) requested from a remote repository are cached on demand. You can remove downloaded artifacts from the remote repository cache, however you can not manually deploy artifacts to a remote PyPI repository.

To create a repository to proxy a remote PyPI repository follow the steps below:

1. In the **Admin** module under **Repositories | Remote**, select "New"
2. Set the **Package Type** to **PyPI**, enter the **Repository Key** value, and specify the URL to the remote repository in the **URL** field as displayed below:

New Remote Repository

Basic Advanced Replications

Package Type *



Pypi

Repository Key *

URL *

Test

PyPI remote repository URL
 You should not include "/pypi" or "/simple" in the the PyPI remote repository URL. These suffixes are added by Artifactory when accessing the remote repository.
 If you use a custom PyPI remote repository, you need to make sure it has a simple index (directory listing style) accessible by <URL>/simple.

3. Click "Save & Finish"

Remote Artifactory

If the remote repository is also managed by an Artifactory server, then you need to point to its PyPI API URL, for example `http://my.remote.artifactory/artifactory/api/pypi/python-project`

Virtual Repositories

A Virtual Repository defined in Artifactory aggregates packages from both local and remote repositories.

This allows you to access both locally hosted PyPI packages and remote proxied PyPI repositories from a single URL defined for the virtual repository.

To define a virtual PyPI repository, create [virtual repository](#), set its **Package Type** to be PyPI, select the underlying local and remote PyPI repositories to include in the **Basic** settings tab, click "Save & Finish".

New Virtual Repository

Basic Advanced

Package Type *  PyPI

Repository Key * pypi-virtual

General

Repository Layout simple-default

Public Description Internal Description

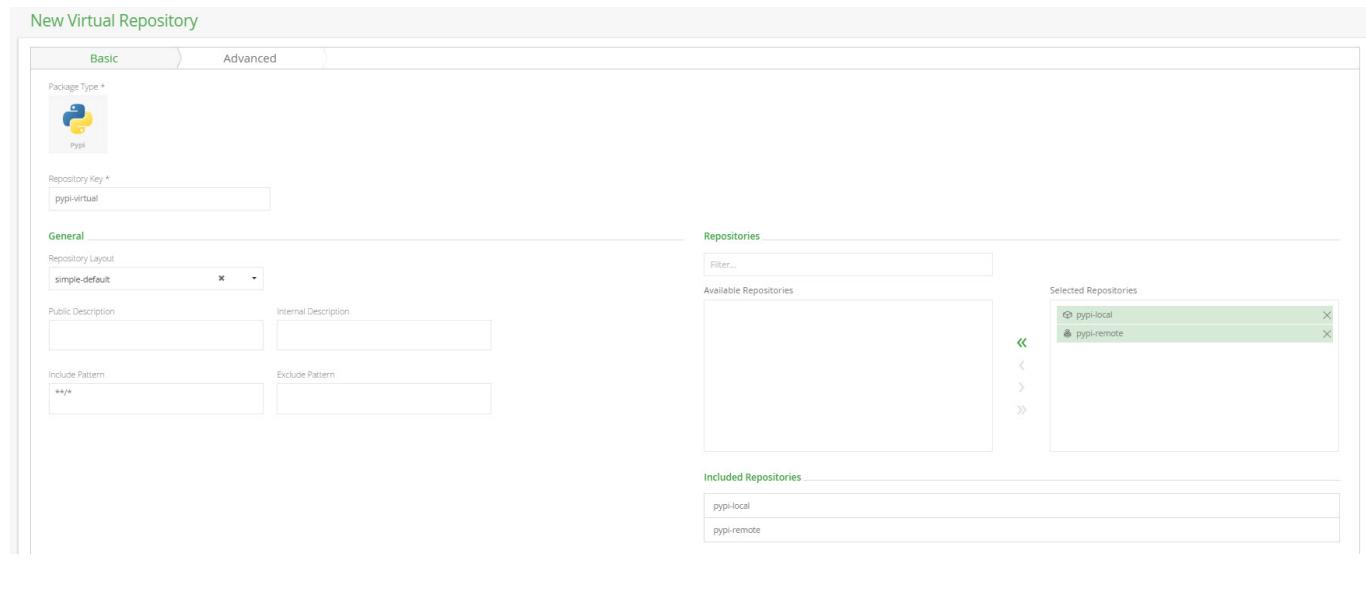
Include Pattern **/* Exclude Pattern

Repositories

Filter... Available Repositories Selected Repositories

Selected Repositories: pypi-local, pypi-remote

Included Repositories: pypi-local, pypi-remote



Resolving from Artifactory Using PIP

To install the `pip` command line tool refer to [pip documentation pages](#). We recommend using `virtualenv` to separate your environment when installing PIP.

To display code snippets you can use to configure `pip` and `setup.py` to use your PyPI repository, select the repository and then click **Set Me Up**.

Set Me Up

Tool



Repository



Deploy

To deploy package using `setuptools` you will need to add an Artifactory repository to the `.pypirc` file (usually located in your home directory)

```
1 [distutils]
2 index-servers = local
3 [local]
4 repository: http://10.100.1.110:8081/artifactory/api/pypi/pypi-local
5 username: <USERNAME>
6 password: <PASSWORD>
```

To deploy a python egg to Artifactory after changing the `.pypirc` file run the following command

```
1 python setup.py sdist upload -r <LOCAL>
```

where is the index server you defined in `.pypirc`

Resolve

To resolve packages using nin add the following to `~/nin/nin.conf`

Specifying the Repository on the Command Line

Index URL

When accessing a PyPI repository through Artifactory, the repository URL must be prefixed with `api/pypi` in the path. This applies to all `pip` commands and `distutils` URLs including `pip install`.

When using `pip` to resolve PyPI packages it must point to `<Artifactory URL>/api/pypi/<repository key>/simple`.

For example, if you are using Artifactory standalone or as a local service, you would access your PyPI repositories using the following URL:

```
http://localhost:8081/artifactory/api/pypi/<repository key>/simple
```

Or, if you are using Artifactory Online the URL would be:

```
https://<server name>.artifactoryonline.com/<server name>/api/pypi/<repository key>/simple
```

Once pip is installed, it can be used to specify the URL of the repository from which to resolve:

Installing with full repository URL

```
$ pip install frog-bar -i http://localhost:8080/artifactory/api/pypi/pypi-local/simple
```

Using Credentials

Due to its design, pip does not support reading credentials from a file. Credentials can be supplied as part of the URL, for example `http://<username>:<password>@localhost:8080/artifactory/api/pypi/pypi-local/simple`. The password can be omitted (with the preceding colon), and in this case, the user will be prompted to enter credentials interactively.

Using a Configuration File

Aliases for different repositories can be specified through a pip configuration file, `~/.pip/pip.conf`. The file contains configuration parameters per repository, for example:

~/.pip/pip.conf

```
[global]
index-url =
http://user:password@localhost:8080/artifactory/api/pypi/pypi-virtual/simple
```

For more information, please refer to [PIP User Guide](#).

Using a Requirements File

A requirements file contains a list of packages to install. Usually these are dependencies for the current package. It can be created manually or using the `pip freeze` command. The index URL can be specified in the first line of the file, For example:

requirements.txt

```
--index-url http://localhost:8080/artifactory/api/pypi/pypi-local/simple
PyYAML==3.11
argparse==1.2.1
frog-bar==0.2
frog-fu==0.2a
nltk==2.0.4
wsgiref==0.1.2
```

Publishing to Artifactory

Using distutils or setuptools

setuptools vs. distutils and python versions

Artifactory is agnostic to whether you use `setuptools` or `distutils`, and also to the version or implementation of Python your project uses.

The following instruction were written for Python 2.7 and `setuptools` in mind. Using different version of Python, or different tools such `zest`, `distutils` and others may require minor modification to the instructions below.

Uploading to Artifactory using a `setup.py` script is supported in a similar way to uploading to PyPI. First, you need to add Artifactory as an index server for your user.

For instructions on using `setuptools` to package Python projects and create a `setup.py` script, please refer to the [setuptools documentation](#) and [this tutorial project](#).

Create the `$HOME/.pypirc` File

To upload to Artifactory, an entry for each repository needs to be made in `$HOME/.pypirc` as follows:

```
[distutils]
index-servers =
    local
    pypi

[pypi]
repository: https://pypi.python.org/pypi
username: mrBagthrope
password: notToBeSeen

[local]
repository: http://localhost:8080/artifactory/api/pypi/pypi-local
username: admin
password: password
```

Notice that the URL does not end with `/simple`.

The `HOME` environment variable

`setuptools` requires that the `.pypirc` file be found under `$HOME/.pypirc`, using the `HOME` environment variable.

On unix-like systems this is usually set by your system to `/home/yourusername/` but in certain environments such as build servers you will have to set it manually.

On Windows it must be set manually.

Uploading

After creating a `.pypirc` file and a `setup.py` script at the root of your project, you can upload packages as follows:

```
~/python_project $ python setup.py sdist upload -r local
```

Where **local** is the name of the section in your `.pypirc` file that points to your Artifactory PyPI repository.

Default upload

By default, both `setuptools` and `distutils` will upload to `https://pypi.python.org/pypi` if no repository is specified.

The 'register' command should be omitted

When uploading directly to pypi.python.org, the documentation states that your package must first be registered by calling `python setup.py register`.

When uploading to Artifactory this is neither required nor supported and **should be omitted**.

Publishing Manually Using the Web UI or REST

PyPI packages can also be uploaded manually using the [Web UI](#) or the [Artifactory REST API](#). For Artifactory to handle those packages correctly as PyPI packages they must be uploaded with **pypi.name** and **pypi.version** [Properties](#).

Automatic extraction of properties

While indexing the newly uploaded packages Artifactory will automatically try to extract required properties from the package metadata saved in the file. Note that not all supported files can be extracted.

Currently, only `zip`, `tar`, `tgz`, `tar.gz`, `tar.bz2`, `egg` and `whl` files can be extracted for metadata.

In addition, indexing starts after a 60 second quiet period, counting from the last upload to the current repository.

Searching for PyPI Packages

Using PIP

Artifactory supports search using `pip`'s `search` command in local, remote and virtual repositories. For example:

```
pip search
$ pip search frog-fu --index http://localhost:8080/artifactory/api/pypi/pypi-virtual/
frog-fu
INSTALLED: 0.2a (latest)

$ pip search irbench --index http://localhost:8080/artifactory/api/pypi/pypi-virtual/
irbench
- Image Retrieval Benchmark.
```

In this example **frog-fu** is a locally installed package, while **irbench** is found at pypi.python.org, both repositories aggregated by the `pypi-virtual` repository.

Specifying the index

When using the search command, the index should be specified explicitly (without the `/simple` at the end), as `pip` will ignore the `index-url` variable in its `pip.conf` file.

Artifactory Search

PyPI packages can also be searched for using Artifactory's [Property Search](#). All PyPI packages have the properties **pypi.name**, **pypi.version** and **pypi.summary** set by the uploading client, or later during indexing for supported file types.

Viewing Metadata of PyPI Packages

Artifactory lets you view selected metadata of a PyPI package directly from the UI.

In the **Artifacts** module **Tree Browser**, drill down to select the file you want to inspect. The metadata is displayed in the **PyPI Info** tab.

 geokey-0.6.tar.gz

[Download](#) [Actions](#)

General PyPI Info Effective Permissions Properties Watchers Builds Governance

Package Info

Name:	geokey
Version:	0.6
Platform:	UNKNOWN
Summary:	Open-source participatory mapping framework
Keywords:	
Homepage:	http://geokey.org.uk
Download URL:	https://github.com/excites/geokey/releases
Author:	Oliver Roick
Author Email:	o.roick@ucl.ac.uk
License:	Apache 2.0

Watch the Screencast

Properties

Overview

Artifactory allows you to place properties on both artifacts and folders.

You can assign properties from the **UI**, via **REST** (see below), or **on deployment**, using **Matrix Parameters**. Properties can also be used to [Control Artifacts Resolution](#).

Properties are **searchable** and can be combined with [Smart Searches](#) to search for items based on their properties and then manipulate all the items in the search result in one go.

Property Sets

You can define the collections of properties called 'Property Sets' in the user interface. In each property-set you can define properties and for each property specify whether the property is open, single-value or multi-value.

This impacts the user interface you see when setting a property value and when searching for property values. Using searchable properties in artifact management is a very powerful feature.

Page Contents

- Overview
 - Property Sets
- Attaching and Reading Properties via REST API
- Watch the Screencast

Read More

- [Using Properties in Deployment and Resolution](#)

Properties are for Guiding, not for Restricting

When you define a property-set with 'strongly-typed' property values, those values are used to provide an intuitive, guiding UI for tagging and locating items.

The actual value does not force a strong relationship to the original property-set's predefined values. This is by design, to not slow-down common repository operations and for keeping artifacts management simple by allowing properties to change and evolve freely over time, without worrying about breaking older property rules.

Properties are therefore a helpful and non-restrictive feature.

Attaching and Reading Properties via REST API

Properties are a special form of metadata and are stored on items just like any metadata - in XML form.

In fact, you can view properties not only from the *Artifacts:Properties* tab, but also from the *Artifacts:Metadata* tab, in which you can examine properties as they are stored in XML form. The properties XML is using the `properties` root tag and has a very simple format.

You can set, retrieve and remove properties from repository items via REST API, as you would do with any other XML-based metadata.

Watch the Screencast

Watch this short screencast to learn more about the power of properties.

Using Properties in Deployment and Resolution

Introducing Matrix Parameters

Matrix parameters key-value pairs separated by a semicolon (;) that you can place anywhere on a URI.

This is a [standard](#) method for specifying parameters in HTTP (in addition to querying parameters and path parameters).

For example:

```
http://repo.jfrog.org/artifactory/libs-releases-local/org/libs-releases-local/or  
g/jfrog/build-info-api/1.3.1/build-info-api-1.3.1.jar;status=DEV;rating=5
```

Artifactory makes use of matrix parameters for:

1. Adding properties to artifacts as part of deployment
2. Controlling artifact resolution using matrix parameters

Page Contents

- Introducing Matrix Parameters
- Dynamically Adding Properties to Artifacts on Deployment
- Controlling Artifact Resolution with Matrix Parameters Queries
 - Non-mandatory Properties
 - Mandatory Properties
- Multi-valued Properties Support

Dynamically Adding Properties to Artifacts on Deployment

You can add key-value matrix parameters to deploy (PUT) requests and those are automatically transformed to properties on the deployed artifact.

Since matrix parameters can be added on any part of the URL, not just at the end, you can add them to the target deployment base URL. At the time of deployment, the artifact path is added after the matrix parameters and the final deployed artifact will be assigned the defined properties.

You can even use dynamic properties, depending on our deployment framework.

When using Maven, for instance, you can add two parameters to the deployment URL: `buildNumber` and `revision`, which Maven replaces at deployment time with dynamic values from the project properties (e.g. by using the Maven build-number plugin).

So, if you define the distribution URL as:

```
http://myserver:8081/artifactory/qa-releases;buildNumber=${buildNumber};revision=${rev  
ision}
```

And deploy to the `qa-releases` repository a jar with the following path:

```
/org/jfrog/build-info-api/1.3.1/build-info-api-1.3.1.jar
```

Upon deployment the URL is transformed to:

```
http://myserver:8081/artifactory/qa-releases;buildNumber=249;revision=1052/org/jfrog/b  
uild-info-api/1.3.1/build-info-api-1.3.1.jar
```

And the deployed build-info-api-1.3.1.jar has two new properties:

```
buildNumber=249  
revision=1052
```

Permissions to attach properties

You must have the 'Annotate' permission in order to add properties to deployed artifacts.

Controlling Artifact Resolution with Matrix Parameters Queries

Matrix parameters can also be used in artifact resolution, to control how artifacts are found and served.

There is currently support for two types of queries:

- Non-conflicting values
- Mandatory values.

Non-mandatory Properties

Resolved artifacts may either have no property with the key specified, or have the property with the key specified and the exact value specified (i.e. the artifact is resolved if it has a property with a non-conflicting value).

Non-mandatory properties are identified by a simple `key=value` parameter.

For example:

Current Artifact Property	Matrix Parameter	Resolution Result
color=black	color=black	OK (200)
None or height=50	color=black	OK (200)
color=red	color=black	NOT_FOUND (404)

Mandatory Properties

Resolved artifacts must have a property with the key specified and the exact value specified.

Mandatory properties are identified with a plus sign (+) after the property key: `key+=value`.

For example:

Current Artifact Property	Matrix Parameter	Resolution Result
---------------------------	------------------	-------------------

color=black	color+=black	OK (200)
None or height=50	color+=black	NOT_FOUND (404)
color=red	color+=black	NOT_FOUND (404)

Multiple properties in queries

Multiple key-value matrix parameters are additive, forming an AND query between each key-value subsection.

Multi-valued Properties Support

All matrix parameters can support multiple values by separating values with a comma (,). For example:

```
colors=red,gold,green
```

Smart Searches

Overview

Smart search is a feature that allows you to assemble a custom set of artifacts returned by a series of separate searches actions. This is done by saving search results in a Stash.

The Stash provides easy access to artifacts found without having to run the series of searches again, and also provides a convenient way to perform bulk operations on the result set using the [Stash Browser](#).

Using the Stash you can save a search result, then use additional searches to add, remove and intersect new results with the original result. Effectively, you are assembling a 'shopping cart' of artifacts, which you can then manipulate as one unit.

For example, you can search for all artifacts deployed by a certain build (by build number), remove all the sources from the search results (by running another search) and promote the final result set to a public repository. Or, you can search all POMs containing a specific license and move them to a repository of approved artifacts, or attach an "approved" property to them.

Page Contents

- [Overview](#)
- [Saving Search Results in the Stash](#)
- [Stash Browser](#)
- [From Staging to Promotion](#)

Saving Search Results in the Stash

To save search results after running a search, click **Stash Results**. To save only a subset of the search results, first select the items you want to save and then click **Stash Results**. If you don't select any items, the whole result set will be saved.

Artifact	Group ID	Artifact ID	Version	Classifier	Repository	Modified
bintray-info-3.6-20150720.105830-1.json	org.jfrog.test	bintray-info	3.6-20150720.105830-1		libs-snapshot-local	20-07-15 10:59:05 UTC
bintray-info-3.6-20150720.105830-1.pom	org.jfrog.test	bintray-info	3.6-20150720.105830-1		libs-snapshot-local	20-07-15 10:59:05 UTC
bintray-multi-3.6-20150720.105830-1.pom	org.jfrog.test	bintray-multi	3.6-20150720.105830-1		libs-snapshot-local	20-07-15 10:58:47 UTC

Once you have items stored in the stash, Artifactory displays the number of items stored and offers several actions you can perform.

Artifact	Group ID	Artifact ID	Version	Classifier	Repository	Modified
doxia-1.0-alpha-7.pom	org.apache.maven.doxia	doxia	1.0-alpha-7		gradle-libs-cache	20-06-11 12:27:30 UTC
doxia-sink-api-1.0-alpha-7.jar	org.apache.maven.doxia	doxia-sink-api	1.0-alpha-7		gradle-libs-cache	31-08-08 11:46:46 UTC
doxia-sink-api-1.0-alpha-7.pom	org.apache.maven.doxia	doxia-sink-api	1.0-alpha-7		gradle-libs-cache	20-06-11 12:27:19 UTC

View	This displays the Stash Browser showing all items currently stored in the stash
Clear	Remove all items from the stash
Actions 	Add: Adds to the stash items found in the current result set that are not already stored in the stash Subtract: Items found in the current search result set, that are also in the stash, are subtracted (i.e. removed) from the stash Intersect: Items that are in the intersection of the current search results and the current stash contents are kept in the stash. All other items are removed.

Stash Browser

The stash browser displays all items that are in the stash. You can browse through the items and view relevant information corresponding to the item type just like you would in the [Tree Browser](#).

The screenshot shows the JFrog Artifactory interface. On the left is a sidebar with links for Home, Artifacts, Builds, and Admin. The main area is titled 'Stash Browser' and shows a search results tree. A specific item, 'plexus-components-1.1.14.pom', is selected. To the right of the item details, there is a 'Actions' menu with several options: Download, View, Delete, Discard from Stash, and Show In Tree. The 'Delete' option is highlighted with a red box.

If you select one of the items in the stashed search results tree, the specific information panel relevant to the selected item is displayed. The **Actions** available are:

Delete	Delete the item.
Discard from Stash	Remove the item from the stash without deleting it.
Show in Tree	Display the item in the Artifact Tree Browser.
View	View the contents of the file
Download	Download the artifact or folder

If you are on the root **Stashed Search Results** item, you can perform bulk actions on all the contents of the stash at once.

The screenshot shows the JFrog Artifactory interface. On the left is a sidebar with links for Home, Artifacts, Builds, and Admin. The main area is titled 'Stash Browser' and shows a search results tree. The root item is 'Stashed Search Results'. To the right of the item details, there is a 'Actions' menu with three options: Copy Stash to Repository, Move Stash to Repository, and Discard Search Results. The 'Copy Stash to Repository' option is highlighted with a red box.

Copy Stash to Repository

Copies the entire stash contents to a repository

Move Stash to Repository	Move the entire stash contents from their current location to a repository
Discard search results	Removes all items from the stash without deleting them.

On the root **Stashed Search Results** item you can also perform an export of the entire stash in the same way you would [export a repository](#).

To go back to the Artifacts Tree Browser, click **Back to Repository Browser**.

From Staging to Promotion

For more detailed information about using Smart Searches for powerful, yet simple, promotion support please see [this blog entry](#).

For a short demo of this, please watch the following screencast:

A screencast using Version 4 is coming soon...

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!

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)

Page Contents

- Overview
- The Freedom of Custom Layouts
- Bundled Layouts
- Modules and Path Patterns used by Repository Layouts
 - Module Fields
 - Using Module Fields to Define Path Patterns
 - Path Pattern Tokens
 - Artifact Path Patterns
 - Descriptor Path Patterns
- Configuration
 - Layout Configuration
 - Testing Layouts
 - Path Patterns
 - Regular Expressions for File and Folder Integration Revision
 - Repository Configuration
 - Local Repository Configuration
 - Remote Repository Configuration
 - Virtual Repository Configuration

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 "1.2-20110202.14453-3" the file integration revision is "20110202.14453-3"	
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:

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

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

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**.

Repositories Layouts

[+ New](#)

Filter by Name

< page 1 of 1 >

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

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

Basic > Advanced > Replications

Package Type *

npm

Npm

Repository Key *

npm-local

General

Repository Layout

Select Repository Layout...

npm-default

bower-default

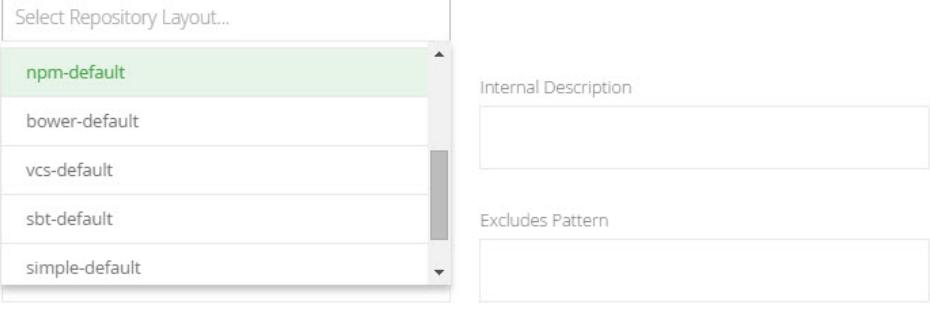
vcs-default

sbt-default

simple-default

Internal Description

Excludes Pattern



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 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 **Remote 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

Basic

Advanced

Replications

Package Type *



Maven

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

License Control

Controlling Third Party Licenses

The License Control Add-on completes the [Artifactory Build Integration](#) Add-on allowing you full control over the licenses of the dependencies used by your builds (and eventually in your software).

This Add-on is part of the Artifactory Pro Power Pack.

As part of the Build Server deployment to Artifactory, it analyzes the used dependencies and tries to match them against a set of license management rules.

Notifications can be sent to a selected list of recipients about dependencies with unknown or unapproved license information.

To support this feature Artifactory includes a new license management facility where rules about license matching and approval status are defined. These rules are consulted as part of the license analysis.

How does license analysis work?

Automatic analysis is performed upon deployment by examining information found in artifact module files. Currently **Maven POM**, **Ivy Descriptor**, **NuGet**, and **RPM** files are supported.

You can always override the automatic results and assign license information manually to dependencies. You can also compare the current license status to the auto calculated one and decide what results of the automatic analysis to accept.

License information is stored with the artifact and reused by the automatic license analysis on subsequent builds.

Page Contents

- Controlling Third Party Licenses
- Central License Management
 - Editing License Information
- Using Build Licenses
 - Build Server Configuration
 - Examining Build Licenses
 - Running Manual License Discovery
- Setting License Information Manually
- Licenses REST API
- Watch the Screencast

Central License Management

Licenses are managed under the Admin tab and then *Configuration -> Licenses*.

Licenses Configuration				
Filter by License Key				
	License Key	Long Name	Url	Status
	AFL-3.0	The Academic Free License 3.0	http://www.opensource.org/licenses/afl-3.0.php	Unapproved
	AGPL-V3	GNU AFFERO GENERAL PUBLIC LICENSE v3	http://www.opensource.org/licenses/agpl-v3.html	Approved
	APL-1.0	Adaptive Public License 1.0	http://www.opensource.org/licenses/apl1.0.php	Unapproved
	Apache-2.0	The Apache Software License, Version 2.0	http://www.opensource.org/licenses/apache2.0.php	Approved
	Apache-1.0	The Apache Software License, Version 1.0	http://apache.org/licenses/LICENSE-1.0	Unapproved
	Apache-1.1	The Apache Software License, Version 1.1	http://apache.org/licenses/LICENSE-1.1	Approved
	APS-2.0	The Apple Public Source License 2.0	http://www.opensource.org/licenses/apsl-2.0.php	Unapproved
	Artistic-License-2.0	Artistic License 2.0	http://www.opensource.org/licenses/artistic-license-2.0.php	Unapproved
	Attribution	The Attribution Assurance License	http://www.opensource.org/licenses/attribution.php	Approved
	BSL-1.0	Boost Software License 1.0 (BSL1.0)	http://www.opensource.org/licenses/bsl1.0.html	Unapproved
	CA-TOSL-1.1	Computer Associates Trusted Open Source License 1.1	http://www.opensource.org/licenses/ca-tosl-1.1.php	Unapproved
	CDDL-1.0	Common Development and Distribution License (CDDL) 1.0	http://www.opensource.org/licenses/cddl1.php	Unapproved
	CDDL-1.0.1	Common Development and Distribution License (CDDL) 1.0.1	http://www.opensource.org/licenses/cddl1.0.1.php	Approved
	CDDL-1.1	Common Development and Distribution License (CDDL) 1.1	http://glassfish.java.net/public/CDDL+GPL_1_1.html	Unapproved
	Codehaus	Copyright 2002 (C) The Codehaus	http://classworlds.codehaus.org/license.html	Unapproved
	CCAG-2.5	Creative Commons Attribution 2.5 Generic	http://creativecommons.org/licenses/by/2.5/	Unapproved
	CPAL-1.0	Common Public Attribution License Version 1.0 (CPAL)	http://www.opensource.org/licenses/cpal_1.0	Approved
	CUAOFFICE-1.0	CUA Office Public License	http://www.opensource.org/licenses/cuoffice.php	Unapproved
	Day	Day Spec License	http://www.day.com/dam/day/downloads/sr283/day-spec-license.htm	Unapproved
	Day-Addendum	Day Specification License Addendum	http://www.day.com/content/dam/day/downloads/sr283/LICENSE.txt	Unapproved
	Bouncy-Castle	Bouncy Castle License	http://www.bouncycastle.org/licence.html	Unapproved
	EUDATAGRID	EU DataGrid Software License	http://www.opensource.org/licenses/eudatagrid.php	Unapproved
	Enovi	Copyright (c) 2005 Enovi Solutions LLC		Unapproved
	CPL-1.0	Common Public License	http://www.opensource.org/licenses/cpl1.0.txt	Unapproved

Editing License Information

For each license, you can configure general license information, the regular expression by which to match the license (by comparing it to license

information in module files) and whether the license is an approved one or not.

If you leave the regexp field blank, Artifactory attempts an exact match against the license key.

Artifactory comes preconfigured with all the common OSI licenses and JFrog has already tuned these licenses against common project builds.

Edit License AGPL-V3

License Key *

Long Name

URLS

Notes

Regexp

Approved

Finally, you can export the license list and import it later on to new Artifactory instances.

Using Build Licenses

Build Server Configuration

When you run a build from your CI server (Hudson, TeamCity or Bamboo), configure the Artifactory Plugin to run license checks as part of the build.

Below is a sample section from the Hudson configuration of the Artifactory Plugin:

Run License Checks (requires Pro) ?

Send license violation notifications to: ?

Comma-separated list of recipient addresses.

You can configure whether or not you wish license checks to take place as part of deploying Build Info to Artifactory (the Build Info Bill of

Materials must be deployed to Artifactory for license checks to run).

You can also set a list of recipients to be notified about license violations as soon as they occur. This way whenever a dependency with an unknown or unapproved license is added to the build recipients receive an immediate email notification and can tend to any potential license violation.

Sending license violation notifications is performed through Artifactory and requires a valid mail server to be configured.

Not failing the build

Currently, Artifactory does not fail the build as a result of license violations.

This is an informed decision in the spirit of allowing technical development to continue, while alerting others about the advent of unauthorized dependencies in near or real-time, so they can be addressed early on by the appropriate parties.

Examining Build Licenses

Once the build has finished on the build server and Build Info has deployed to Artifactory, license checks are run.

You can view detailed license information in the **Licenses** tab of the **Build Browser**. This tab displays information about all the dependencies used in the build and the license they are associated with. To group the information by **Scope** or **License** click the corresponding column header.

The screenshot shows the Artifactory Build Browser interface. At the top, there's a breadcrumb navigation: All Builds > maven-example > 60. Below it, the title "Build Browser" is displayed above "Build #60". A horizontal menu bar includes General Build Info, Published Modules, Environment, Issues, **Licenses** (which is highlighted in green), Governance, Diff, and a double arrow icon. Under the "Licenses" tab, a summary panel shows: Unapproved: 3, Not Found: 13, Unknown: 0, Neutral: 0, Approved: 2. Below this is a "Includes" section with checkboxes for "Include Published Artifacts" and "Include dependencies of the following scopes:" with options: compile (selected), test, provided, and runtime. At the bottom of this section are "Export to CSV" and "Auto-find Licenses" buttons. A "Filter by Artifact ID" input field is present. The main content area is a table with columns: Artifact ID, Scopes, Repo Path, and License. The table rows show the following data:

Artifact ID	Scopes	Repo Path	License
aopalliance:aopalliance:1.0	compile	gradle-libs-cache:aopalliance/aopalliance/1.0...	Public Domain
org.springframework:spring-beans:2.5.6	compile	Not in repository (externally resolved or dele...	Not Found
org.springframework:spring-aop:2.5.6	compile	Not in repository (externally resolved or dele...	Not Found
org.springframework:spring-core:2.5.6	compile	Not in repository (externally resolved or dele...	Not Found
org.testng:testng:5.9	test	gradle-libs-cache:org/testng/testng/5.9/testn...	Not Found
javax.servlet:servlet-api:2.5	provided	java.net.m1-cache:javax/servlet/servlet-api/2...	Not Found
javax.mail:mail:1.4	compile	gradle-libs-cache:javax/mail/mail/1.4/mail-1...	Not Found

The summary panel displays the overall count of licenses by status and inside the table itself, licenses are displayed in different colors according to their status:

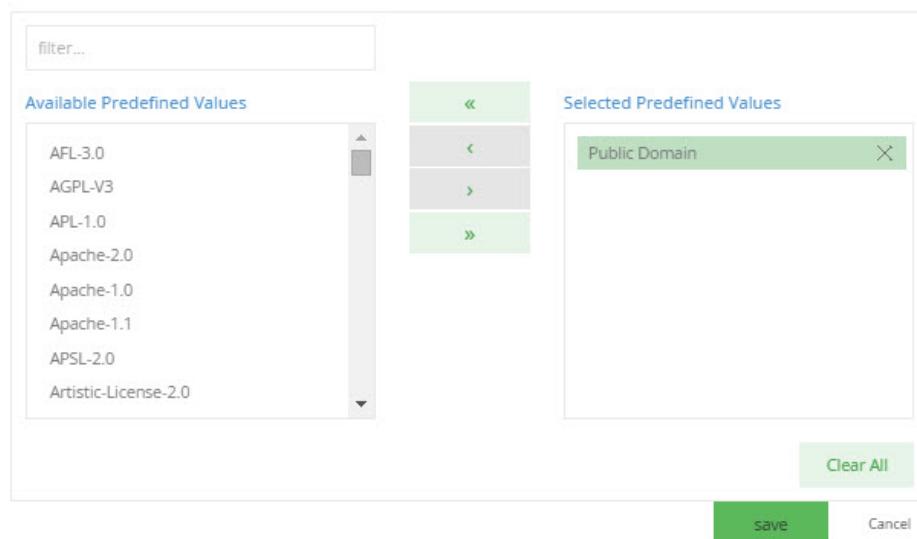
License Status	Description
----------------	-------------

Unapproved	The license found is not an approved license
Unknown	License information was found but cannot be related to any license managed in Artifactory
Not Found	No license information could be found for the artifact.
Neutral	The license found is unapproved, however another approved license was found for the artifact (Only applicable for artifacts that are associated with multiple licenses)
Approved	The license found is an approved license

Inline License Editing

From the Build Browser, an Artifactory administrator can manually change the license information for any artifact displayed. Clicking the entry under the **License** column for any artifact will display the **Edit 'artifactory.licenses' Property** dialog where the administrator can specify the licenses for that artifact. For example, clicking the *Public Domain* license entry from the screenshot above will display the following dialog:

Edit 'artifactory.licenses' Property



Running Manual License Discovery

You can manually run the license discovery rules after a build has already run. There are several reasons why you may want to do this:

1. License rules (configured licenses and regular expressions) have changed and you want to compare the existing build licenses with the results of the new rules, or use them to complete missing license information.
2. To test the current rules against the dependencies and tweak the rules, if necessary.
3. To check which license information can come from rules and which license information must be set manually.

To trigger license discovery select the "Auto-find Licenses" button.

Any license conflicts are displayed to the right of the table. You can override the existing license information with the discovered license by checking the corresponding checkbox (you must have annotate permissions for the

artifacts for which you want to override licenses).

The screenshot shows the Artifactory Build Browser interface for Build #60. The 'Licenses' tab is selected. A sidebar on the left titled 'Includes' contains checkboxes for 'Include Published Artifacts' and 'Include dependencies of the following scopes:' with options 'compile', 'test', 'provided', and 'runtime'. Below this are buttons for 'Export to CSV' and 'Auto-find Licenses'. A search bar labeled 'Filter by Artifact ID' is present. The main area is a table with columns: Artifact ID, Scopes, Repo Path, License, Found Licenses, and Override. The table lists various artifacts like org.springframework:spring-beans, org.springframework:spring-aop, org.springframework:spring-core, org.testing:testng, javax.servlet:servlet-api, javax.mail:mail, javax.servlet.jsp:jsp-api, aopalliance:aopalliance, org.codehaus.plexus:plexus-utils, commons-logging:commons-logging, commons-io:commons-io, and hsqldb:hsqldb. Most artifacts have 'Not Found' in the License column, while some like testng and servlet-api show 'Apache-2.0'. The 'Override' column contains checkboxes. At the bottom right of the table is a 'Override Selected Licenses' button.

Artifact ID	Scopes	Repo Path	License	Found Licenses	Override
org.springframework:spring-beans:2.5.6	compile	Not in repository (externally resolved or de... Not Found		Not Found	
org.springframework:spring-aop:2.5.6	compile	Not in repository (externally resolved or de... Not Found		Not Found	
org.springframework:spring-core:2.5.6	compile	Not in repository (externally resolved or de... Not Found		Not Found	
org.testing:testng:5.9	test	gradle-libs-cache.org/testng/testng/5.9/tes... Not Found		Apache-2.0	
javax.servlet:servlet-api:2.5	provided	java.net.m1-cache(javax/servlet/servlet-api... Not Found		Not Found	
javax.mail:mail:1.4	compile	gradle-libs-cache(javax/mail/mail/1.4/mail-... Not Found		CDDL-1.0	
javax.servlet.jsp:jsp-api:2.1	compile	gradle-libs-cache(javax/servlet/jsp/jsp-api/... Not Found		Not Found	
aopalliance:aopalliance:1.0	compile	gradle-libs-cache/aopalliance/aopalliance/1... Public Domain		Public Domain	
org.codehaus.plexus:plexus-utils:1.5.1	compile	gradle-libs-cache/org/codehaus/plexus/ple... Day		Apache-2.0	
org.codehaus.plexus:plexus-utils:1.5.1	compile	gradle-libs-cache.org/codehaus/plexus/ple... Day-Addendum		Apache-2.0	
org.codehaus.plexus:plexus-utils:1.5.1	compile	gradle-libs-cache.org/codehaus/plexus/ple... Apache-2.0		Apache-2.0	
org.codehaus.plexus:plexus-utils:1.5.1	compile	gradle-libs-cache.org/codehaus/plexus/ple... Apache-1.1		Apache-2.0	
commons-logging:commons-logging:1.1.1	compile	gradle-libs-cache/commons-logging/comm... Not Found		Apache-2.0	
commons-io:commons-io:1.4	compile	gradle-libs-cache/commons-io/commons-i... Not Found		Apache-2.0	
hsqldb:hsqldb:1.8.0.10	runtime	repo1-cache.hsqldb/hsqldb/1.8.0.10/hsqld... Not Found		HSQLDB	
hsqldb:hsqldb:1.8.0.10	compile	repo1-cache.hsqldb/hsqldb/1.8.0.10/hsqld... Not Found		CCW-1.0	

Setting License Information Manually

To set license information for artifacts manually, when viewing an artifact's details in the **Artifact Repository Browser**, in the **General** tab **Licenses** entry, click **Add**.

This will display the **Add Artifactory Licenses** Property dialog where you can specify the licenses for the selected artifact.

Multiple licenses

Note that an artifact may be associated with multiple licenses

Scanning artifact Maven/Ivy model for license

Another option for editing the license information is by scanning the Maven/Ivy model for licenses, that is, looking for an existing pom matching the artifact.

Once you have the artifact selected in the tree browser go to the General tab and under the License label choose Scan and confirm licenses found in the scan results, if any.

Yet another option would be to use the 'Search For Archive License File' link, which will scan the artifact archive for a 'License' or 'License.txt' entry and ask for confirmation, if found.

License Information as Properties

Internally, license information is stored as regular properties, using the built-in `artifactory.licenses` property name.

Therefore, all operations with properties are available to license information (searches, recursive assignment, property-based deployment and resolution etc.)

Licenses REST API

License-oriented searches and management operations are available through the REST API.

Refer to the [REST API Documentation](#) for usage information.

Watch the Screencast

To see the License Control Add-on in action you can watch the short demo screencast below.

Black Duck Code Center Integration

Introduction

The integration between Black Duck Code Center and Artifactory offers you an automated, non-invasive approach to the open source component approval process, in addition to proactively monitoring for security vulnerabilities that may be associated with specific binary components. License, security vulnerability and approval status are pulled from the Black Duck Knowledge Base.

This chapter describes:

- How to configure Artifactory with Code Center
- Viewing additional artifact Information
- Artifactory Code Center build integration

The add-on adds a Governance tab in Builds, allowing automation of the approval process of an existing Black Duck application in accordance with the build info.

Black Duck integration is supported for Maven and NuGet

Component governance through Artifactory's integration with Black Duck is only supported for Maven and NuGet artifacts. Build component governance is provided through the [Jenkins Artifactory Plugin](#), and the [MSBuild Artifactory Plugin](#) respectively.

Page Contents

- Introduction
- Configuring Artifactory with Code Center
- Additional Artifact Information
- Artifactory Code Center Build Integration
 - CI Configuration
 - Governance Status Summary View
- Grouping and Sorting

Configuring Artifactory with Code Center

To configure Artifactory with Code Center click on the Admin tab and then go to *Configuration -> Black Duck*.

Black Duck Configuration

Black Duck Settings

Enable Black Duck Code Center Integration

Server URI:

https://jfrogcc.blackducksoftware.com

username:

admin

Password:

.....

Connection Timeout:

20000

Test

Save

Cancel

Field Name	Description
Server URI	URI of the Black Duck Code Center instance
Username	Black Duck Code Center authentication username
Password	Black Duck Code Center authentication password
Connection Timeout	Network timeout in milliseconds. Default is set to 20 seconds

Test connection

You can click on the Test button to verify that the credentials are correct.

Proxy

If Artifactory is using a proxy to access remote resources (as described in Managing Proxies), be aware that communication to Code Center will go through the same proxy.

Additional Artifact Information

The window is divided into three sections with the information coming from the Code Center Knowledge Base:

- General information including the **Component Name**, **Version** and **ID** together with a link to the **Homepage** and **Description** of the artifact
- Details of the license
- List of known security vulnerabilities, if any.

To view the additional metadata received from Code Center, select the component in the **Artifact Repository Browser** of the **Artifacts** module, and then select the **Governance** tab.

junit-3.8.1.pom

View Download Actions

General Pom View Effective Permission Properties Watchers Builds Governance

Info

Component Name:	JUnit	
Component Version:	3.8.1	
Component ID:	10107	
External Component ID:	<input checked="" type="checkbox"/>	
Homepage:	http://www.junit.org/	
Description:	JUnit is a simple framework for writing and running automated tests. As a political gesture, it celebrates programmers testing their own software.	
Catalog Component:	true	

License

Common Public License 1.0

Manually editing the Component ID

Click the icon next to External Component ID to manually override the Component ID with a different one in Code Center

The information appearing in the Governance tab is also cached in the Properties tab and can be both searched for and edited.

junit-3.8.1.pom

View Download Actions

General Pom View Effective Permission Properties Watchers Builds Governance

Add: **Property** Property Set

Name* Value Add

Recursive

Filter by Property Delete Delete Recursively < page 1 of 1 >

Property	Value(s)
blackduck.cc.id	10107
blackduck.cc.license	Common Public License 1.0

Artifactory Code Center Build Integration

Builds performed in the CI Server and deployed in Artifactory can be integrated into the Code Center approval process in an automated and non-invasive approach. When a build completes successfully, Artifactory can run compliance checks and allow you to receive a report to

see the current state of the build in terms of governance via the user interface.

CI Configuration

To run the Code Center compliance checks, you must first configure the CI Server Job.

Run Black Duck Code Center compliance checks (requires Artifactory Pro)

Code Center application name:

Black Duck Code Center application name.

Code Center application version:

Black Duck Code Center application version.

Send compliance report email to:

Space-separated list of recipient addresses.

Limit checks to the following scopes:

Space-separated list of scopes.

Include published artifacts

Auto-create missing component requests

Auto-discard stale component requests

The Application Name and Application Version are mandatory fields and represent the **existing Code Center application**. You can optionally add the email address of where the compliance report is to be sent.

For additional information on the remaining fields, click on the ? icon on each field.

Governance Status Summary View

Once the CI Job is completed, compliance checks are run automatically.

To view the build integration with Code Center, select the corresponding build from the **Build Browser** in the **Build** module, and then select the **Governance** tab.

The Code Center Application section displays application information as it appears in the Code Center and includes the overall approval status.

Build Browser

Build #4

General Build Info Published Modules Environment Issues Licenses **Governance** Diff Release History Build Info JSON

Code Center Application

Name:	shay-licenses
Version:	3.6
Description:	This is some test description
ApprovalStatus:	Rejected
Owner User:	dan

Includes

Include Published Artifacts
 Include dependencies of the following scopes:
 compile test provided runtime

In addition, the Components and Vulnerabilities are displayed.

The Components section shows how many components were found in the BOM and created in the Code Center application. Details of their status (pending, rejected etc..) are given together with licensing details taken from the knowledge base of Black Duck.

The Vulnerabilities section displays the aggregated vulnerabilities found in the application. These details are also taken from the knowledge base of Black Duck.

Build Browser

Build #4

General Build Info Published Modules Environment Issues Licenses Governance Diff Release History Build Info JSON

Components

⚠ Summary: Total: 30 Pending: 9 NotInUse: 10 Stale: 11

< page 1 of 1 >

Status ⓘ	Artifact ID	License ⓘ	Scopes ⓘ	Repo Path
NotInUse				
Pending				
Stale	(Not in build)			

Vulnerabilities

⚠ Summary: Total: 12 Medium: 9 High: 3

< page 1 of 1 >

Severity ⓘ	Artifact ID ⓘ	Vulnerability	Description
High	94940	CVE-2011-2730	VMware SpringSource Spring Framework before 2.5.6...
High	95234	CVE-2011-2730	VMware SpringSource Spring Framework before 2.5.6...
High	95087	CVE-2011-2730	VMware SpringSource Spring Framework before 2.5.6...
Medium			

Once you have updated the status in the Code Center - whether approved or rejected, click on the Governance tab again to refresh the updated information in Artifactory.

You can click on the **Artifact ID** to link out to the Code Center UI where you can perform other tasks such as approving or rejecting the artifact.

Grouping and Sorting

Components can be sorted according to any field. You can also group components according to License, Status or Scope by clicking on the group icon in the corresponding column header providing you with a variety of ways to view the current status of the build.

For example, the screenshot below shows the build components grouped by **License**.

Build Browser

Build #4

General Build Info Published Modules Environment Issues Licenses Governance Diff Release History Build Info JSON

Components

⚠ Summary: Total: 30 Pending: 9 NotInUse: 10 Stale: 11

◀ page 1 of 1 ▶

License ⓘ	Artifact ID	Status ⓘ	Scopes ⓘ	Repo Path
Apache License 2.0	XML Commons External Components XML ...	Pending	compile	jcenter-cache:xml-apis/xml-apis/1.0.b2/xml...
Apache License 2.0	jsp-api	Pending	compile	empty:empty/libs-release-local2/javax/serv...
Apache License 2.0	Commons Logging - commons-logging.com...	Stale		Not in repository (externally resolved or de...
Apache License 2.0	Spring Framework: AOP (Not in build)	Stale		Not in repository (externally resolved or de...
Apache License 2.0	Spring Framework: Beans (Not in build)	Stale		Not in repository (externally resolved or de...
Apache License 2.0	Commons Email (Not in build)	Stale		Not in repository (externally resolved or de...
Apache License 2.0	Plexus Common Utilities (Not in build)	Stale		Not in repository (externally resolved or de...
Apache License 2.0	TestNG (Not in build)	Stale		Not in repository (externally resolved or de...
Common Development and Distribution...				
MIT License				
Mozilla Public License 1.1				
Unspecified				

Filtered Resources

Overview

The Filtered Resources Add-on (introduced in Artifactory version 2.3.3) allows treating any textual file as a filtered resource by processing it as a [FreeMarker](#) template.

Each file artifact can be marked as 'filtered' and upon receiving a download request, the content of the artifact is passed through a FreeMarker processor before being returned to the user.

This is an extremely powerful and flexible feature because Artifactory applies some of its own APIs to the filtering context (see below), allowing you to create and provision dynamic content based on information stored in Artifactory.

For example, you can provision different content based on the user's originating IP address or based on changing property values attached to the artifact.

Page Contents

- [Overview](#)
- [Marking an Artifact as a Filtered Resource](#)
- [Filtering Context](#)
- [Provisioning Build Tool Settings](#)
- [Example](#)

Marking an Artifact as a Filtered Resource

Any artifact can be specified as filtered by selecting it in the **Artifact Repository Browser** and setting the **Filtered** checkbox in the **General** tab.

Permissions

You must have **Annotate** permissions on the selected artifact in order to specify it as "Filtered".

Filtering Context

Artifactory provides the following environment variables for the FreeMarker template:

- **"properties"** (*org.artifactory.md.Properties*) - Contains the properties of the requested artifact and any matrix params included in the request; when a clash of properties with identical keys occurs, the former takes precedence
- **"request"** (*org.artifactory.request.Request*) - The current request that was sent for the artifact
- **"security"** (*org.artifactory.security.Security*) - Artifactory's current security object

Provisioning Build Tool Settings

When logged-in as an admin user, you can provision your user-generated settings for the various build tools (Maven, Gradle and Ivy) using the Filtered Resources features.

To provision user-generated settings:

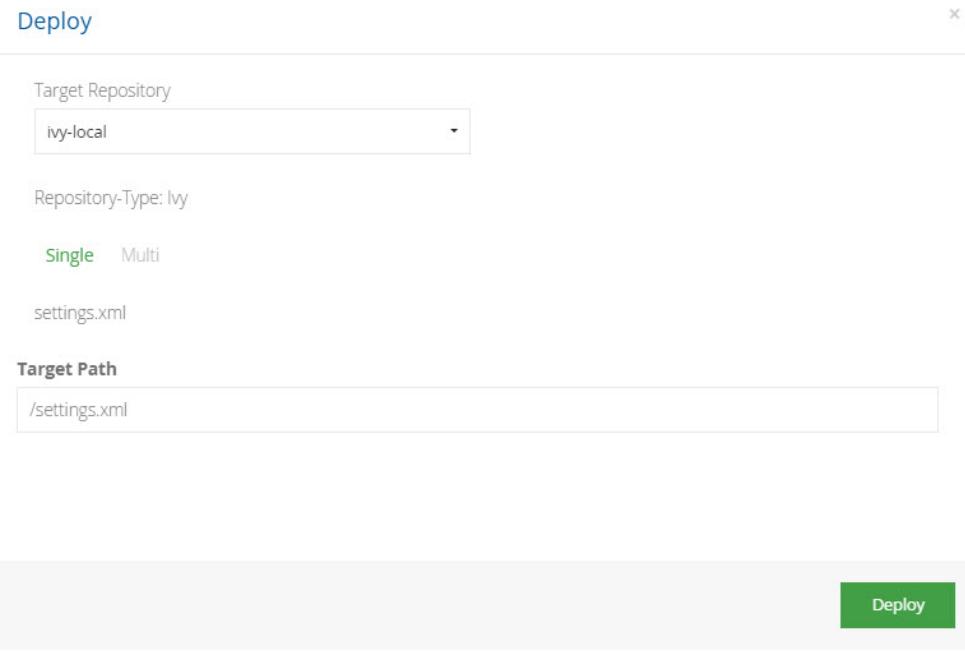
1. In the **Artifact Repository Browser**, click "Set Me Up" to display the settings generator.
2. Select your build tool, set the appropriate repositories and click "Generate Settings".

```

<?xml version="1.0" encoding="UTF-8"?>
<settings xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.1.0
http://maven.apache.org/xsd/settings-1.1.0.xsd" xmlns="http://maven.apache.org/SETTINGS/1.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<servers>

```

3. Download the generated settings and edit them as required.
4. Back in the **Artifact Repository Browser**, click "Deploy".
5. In the Deploy dialog, set your **Target Repository**, upload your settings file and set your **Target Path**.
6. Click "Deploy" to deploy your settings.



Example

The following example demonstrates provisioning a different resource based on the current user group and a property on the requested artifact.

In this example, the artifact 'vcsProj.conf.xml' has a property 'vcs.rootUrl' which holds the root URL for the version control system. Depending on the user group a different project version control URL is returned.

For the template of 'vcsProj.conf.xml':

```
<servers>
<#list properties.get("vcs.rootUrl") as vcsUrl>
    <#list security.getCurrentUserGroupNames() as groupName>
        <vcs>${vcsUrl}<#if groupName == "dev-product1">product1<#elseif groupName ==
"dev-product2">product2<#else>global</#if></vcs>
    </#list>
</#list>
</servers>
```

If, for example, the value of the the 'vcs.rootUrl' property on the 'vcsProj.conf.xml' artifact is 'http://vcs.company.com' and the file is downloaded by a developer belonging to the 'dev-product2' group, then the returned content is:

```
<servers>
    <vcs> http://vcs.company.com/product2 </vcs>
</servers>
```

P2 Repositories

Overview

From version 2.4, Artifactory provides advanced support for proxying and caching of P2 repositories and aggregating P2 metadata using an Artifactory virtual repository which serves as a single point of distribution (single URL) for Eclipse, [Tycho](#) and any other P2 clients.

This virtual repository aggregates P2 metadata and P2 artifacts from underlying repositories in Artifactory (both local and remote) providing you with full visibility of the P2 artifact sources and allowing powerful management of caching and security for P2 content.

For more information on defining virtual repositories please refer to [Virtual Repositories](#).

For P2 support we recommend using Eclipse Helios (version 3.6) and above.

Older versions of Eclipse may not work correctly with Artifactory P2 repositories.

Page Contents

- Overview
- Configuration
 - Defining a Virtual Repository
 - Selecting Local Repositories
 - Selecting Remote Repositories
 - Creating the Repositories
 - Eclipse
- Integration with Tycho Plugins
- Multiple Remote Repositories with the Same Base URL
- Configuring Google Plugins Repository

Configuration

To use P2 repositories, follow the steps below:

- Define a virtual repository in Artifactory
- Select local repositories to add to your virtual repository
- Select remote repositories to add to your virtual repository
- Create the selected local and remote repositories in your virtual repository
- Configure Eclipse to work with your virtual repository

Defining a Virtual Repository

- Create a new virtual repository and set **P2** as the **Package Type**

New Virtual Repository

Basic

Advanced

Package Type *



P2

Repository Key *

p2-virtual

If developers in your organization use different versions of Eclipse (e.g. Helios and Juno), we recommend that you define a different P2 virtual repository for each Eclipse version in use.

Selecting Local Repositories

Adding a local repository to your virtual P2 repository does not require any special configuration:

- Simply select the desired local repository from the **Local Repository** field
- In the **Path Prefix** field, specify the path to the P2 metadata files (`content.jar`, `artifacts.jar`, `compositeContent.xml` etc.)
 - . If left empty, the default is to assume that the P2 metadata files are directly in the repository root directory.
- Click the "Add" button.

Local P2 Repositories

Local Repository

p2-local

Path Suffix

path/to/metadata

Add

If you have a Tycho repository deployed to a local repository as a single archive, specify the archive's root path. For example: `eclipse-repository.zip!/_`

Local P2 Repositories

Local Repository

p2-tycho-local

Path Suffix

eclipse-repository.zip!/

Add

Selecting Remote Repositories

To add a remote P2 repository to Artifactory, enter the URL to the corresponding P2 metadata files (`content.jar`, `artifacts.jar`, `compositeContent.xml`, etc.) and click the "Add" button

Two common examples are:

1. The main Juno repository: <http://download.eclipse.org/releases/juno>
2. The Google plugins repository for Indigo (GWT, GAE, etc.): <http://dl.google.com/eclipse/plugin/3.7>

Remote P2 Repositories

P2 Repository URL

<http://download.eclipse.org/releases/juno>

Add

Artifactory analyzes the added URL and identifies which remote repositories should be created in Artifactory based on the remote P2 metadata (since remote P2 repositories may aggregate information from different hosts).

When P2 metadata files reside inside an archived file, simply add "!" to the end of the URL.

For example: [http://eclipse.org/equinox-sdk.zip!/?](http://eclipse.org/equinox-sdk.zip!/)

Creating the Repositories

Once you have selected the local and remote repositories to include in your virtual repository, Artifactory will indicate what action will be taken once you select the "Save & Finish" button.

The possible actions are as follows:

Create*	Creates a new, P2 enabled, remote repository with the given key (you may still edit the remote repository key).
Modify*	Enables P2 support in an existing remote repository.
Include	Adds the repository to the list of repositories aggregated by this virtual repository.
Included	No action will be taken. This repository is already included in the virtual repository.

*For remote repositories only

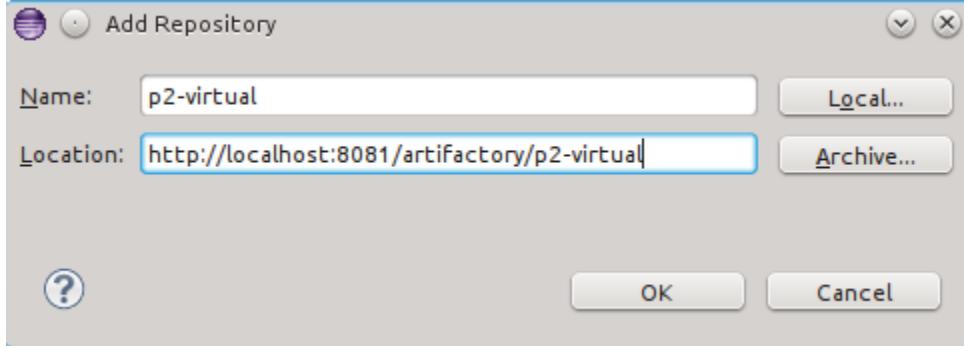
Filter by Repository		
Action	Repository	URL
include	download.eclipse.org	http://download.eclipse.org
include	p2-local	local://p2-local/path/to/metadata
include	p2-tycho-local	local://p2-tycho-local/eclipse-repositor...

Eclipse

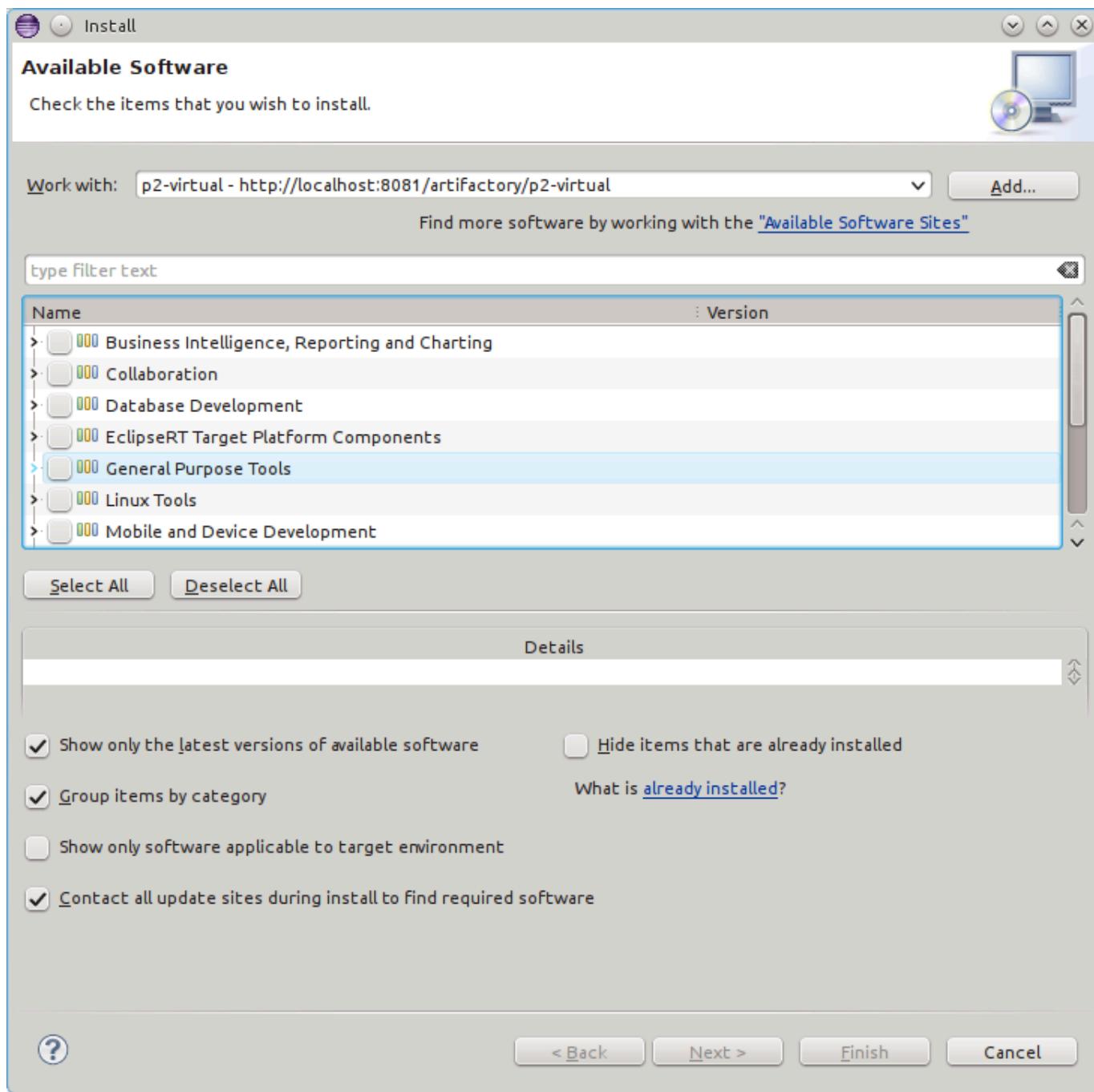
You are now ready to configure eclipse to work with the virtual repository you have created above.

In the Eclipse menu, select **Help | Install new Software** and then click **Add**.

In the **Add Repository** popup, enter the name of your virtual repository and click "OK":



Eclipse will then query Artifactory for available packages and update the screen to display them as below:



Integration with Tycho Plugins

Artifactory fully supports hosting of Tycho plugins as well as resolving Tycho build dependencies.

To resolve all build dependencies through Artifactory, simply change the repository URL tag of your build pom.xml file and point it to a dedicated virtual repository inside Artifactory

For example:

```

<repository>
    <id>eclipse-indigo</id>
    <layout>p2</layout>
    <url>http://localhost:8081/artifactory/p2-virtual</url>
</repository>

```

The P2 virtual repository should contain URLs to all local repositories with an optional sub-path in them where Tycho build artifacts reside.

Multiple Remote Repositories with the Same Base URL

When using P2-enabled repositories with multiple remote repositories that have the same base URL (e.g <http://download.eclipse.org>) , you need to ensure that only 1 remote repository is created within your virtual repository (for each base URL). When creating your virtual repository, Artifactory takes care of this for you, but if you are creating the remote repositories manually, you must ensure to create only a single remote repository, and point the sub-paths accordingly in the P2 virtual repository definition.

In the example below, <http://download.eclipse.org/releases/helios> and <http://download.eclipse.org/releases/juno> were both added to the same virtual repository...repository.

Filter by Repository		< page <input type="text" value="1"/> of 1 >	
Action	Repository	URL	
included	download.eclipse.org	http://download.eclipse.org/releases/helios	
included	download.eclipse.org	http://download.eclipse.org/releases/juno	

...but in fact, the virtual repository only really includes one remote repository

Virtual Repositories

Virtual Repositories			
Filter by Repository key or Type			
Repository key	Type	Included Repositories	Selected Repositories
p2-virtual-multiremote	P2	1	download.eclipse.org
plugins-release	Maven	3	plugins-release-local ;ext-release-local;remote-repos
plugins-snapshot	Maven	3	plugins-snapshot-local ;ext-snapshot-local;remote-repos
remote-repos	Maven	3	java.net.m1 ;repo1;gradle-libs

Configuring Google Plugins Repository

The Google Plugins repository (<http://dl.google.com/eclipse/plugin/3.7>) is aggregated across 3 different URLs. Therefore you need to configure Artifactory to create all of them in order to resolve P2 artifacts correctly:

Filter by Repository

< page 1 of 1 >

Action	Repository	URL
create	dl-ssl.google.com	http://dl-ssl.google.com
create	dl.google.com	http://dl.google.com
create	dl.google.com-1	https://dl.google.com-1

Watches

Overview

The Watches feature allows you to monitor selected artifacts, folders or repositories for storage events (create/delete/modify) and receive detailed email notifications on repository changes that are of interest to you.

You can add and remove Watches from the 'General' tab in the tree browser. Watches or folders intercept changes on all children. An admin can view and manage watches via the 'Watches' tab in the tree browser.

Watch notifications are aggregated at around 1 minute intervals and sent in a single email message.

All notifications respect the read permissions of the watcher on the watched item(s).

Watch the Screencast

WebStart and Jar Signing

Overview

Java Web Start is a technology developed by Sun Microsystems (now Oracle) to allow you to download and run Java applications directly from your browser with one-click activation.

Java Web Start requires that any JAR downloaded is signed by the software vendor. To support this requirement, Artifactory lets you manage a set of signing keys that are used to automatically sign JAR files downloaded from a virtual repository.

For more information, please refer to the [Oracle documentation for Java Web Start](#).

Managing Signing Keys

Signing keys are managed in the **Admin** module under **Security | Signing Keys**.

Debian Signing Key

Debian signing keys are also managed on this page, however these are not related to JAR signing. For details, please refer to [Debian Signing Keys](#).

Generating JAR Signing Keys

In order to sign JAR files, you first need to create a keystore, and generate and add key pairs to it. These can be created with Oracle's `keytool` utility, that comes built into your Java Runtime Environment (JRE), by executing the following command:

```
keytool -keystore <keystore filename> -keypass <key_password>
-storepass <store_password> -alias <store_alias> \
-genkeypair -dname "cn=<cName>, ou=<orgUnit>, o=<orgName>,
S=<stateName>, c=<country>" -validity <days>
```

For details, please refer to the Oracle [keytool - Key and Certificate Management Tool](#) documentation.

Page Contents

- [Overview](#)
- [Managing Signing Keys](#)
 - [Generating JAR Signing Keys](#)
 - [Setting Your Keystore and Keys](#)
 - [Removing a Key Pair](#)
 - [Configuring Virtual Repositories to Sign JARs](#)
- [Screencast - Using Artifactory with JFrog JavaFX Maven Plug-in](#)
 - [Resources used in the screencast](#)
- [How to build FishSim](#)

Setting Your Keystore and Keys

Before you can add a keystore, you must set the password that will be needed to make any later changes to the keystore. You will need this password to remove or update the keystore.

Set the password and click "Create". This will unlock the rest of the keystore management fields.

Change Key Store Password

Password *

.....

Retype Password *

.....

Create

Once your keystore password is set and you have created a keystore and a set of signing keys, you can add them to Artifactory.

First upload your keystore file under **Add Key-Store** and enter the keystore password. Click "Unlock"

Add Key-Store

Key-Store *

acme-demo.store

Key-Store Password *

.....

Unlock

Once your keystore is set in Artifactory you may add key pairs under **Add Key-Pair**.

Add Key-Pair

Key-Pair Name *

Key-Pair Alias Name *

Private Key Password *

[Cancel](#)

[Save Keypair](#)

Removing a Key Pair

To remove a key pair, simply select the key pair and click "Remove".

Remove Key-Pair

Choose a Key-Pair to Remove

[Remove](#)

Configuring Virtual Repositories to Sign JARs

Once Artifactory has a keystore and key pairs, you can configure a virtual repository with the key pair you wish to use for JAR signing. This is done in the **Advanced** settings of the virtual repository configuration screen.

Edit libs-release Repository

Basic

Advanced

Artifactory Requests Can Retrieve Remote Artifacts [?](#)

Cleanup Repository References in POMs [?](#)

Discard active references

Key-Pair

acme

Screencast - Using Artifactory with JFrog JavaFX Maven Plug-in

Screencast for V4 is coming soon...

Resources used in the screencast

- The JavaFX Maven Plugin provided by JFrog has its own [documentation page](#).
- The personal test PKS (acme-demo.store file) was done using the java keytool:

```
keytool -keystore acme-demo.store -keypass password -storepass password -alias acme-demo \
-genkeypair -dname "cn=Acme Dev, ou=r&d, o=ACME, S=California, c=US" -validity 365
```

- The FishSim demo subversion is [here](#).
- You can test this Add-on for free (no questions asked) for 30 days, with the two main repositories required (jfrog plugins, jfrog libs), using [Artifactory Online](#).

How to build FishSim

If you are not using Artifactory as illustrated in the screencast, you can activate the profile "jfrog" to access the [repo.jfrog.org](#) required resources. This enables build running "mvn -Pjfrogs install" to work.

Working with Maven

Overview

Artifactory fully supports working with Maven both as a source for artifacts needed for a build, and as 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

- Overview
- Viewing Maven Artifacts
- Resolving Artifacts through Artifactory
 - Automatically Generating Settings
 - Provisioning Dynamic Settings for Users
 - Manually Overriding the Built-in Repositories
 - Additional Mirror Any Setup
 - Configuring Authentication
- Deploying Artifacts Through Artifactory
 - Setting Up Distribution Management
 - Setting Up Security in Maven Settings
- Watch the Screencast

Read More

- [Maven Artifactory Plugin](#)

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

The screenshot shows the 'Artifact Repository Browser' interface. On the left, there's a tree view of local repositories. In the center, a specific file, 'maven-metadata.xml', is selected. The right side has tabs for General, XML View, Effective Permission, Properties, Watchers, Builds, and Governance. The XML View tab is active, showing the XML code for the maven-metadata.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata>
  <groupId>org.jfrog.test</groupId>
  <artifactId>multi</artifactId>
  <version>2.20-SNAPSHOT</version>
  <clusterv>2.20-SNAPSHOT</clusterv>
  <overlays>1.0-20150625.111301-1</overlays>
  <timestamp>20150625111355</timestamp>
  <updated>20150625111355</updated>
</metadata>
```

POM View

The screenshot shows the 'Artifact Repository Browser' interface. On the left, there's a tree view of local repositories. In the center, a specific file, 'multi-2.20-20150625.111301-1.pom', is selected. The right side has tabs for General, POM View, Effective Permission, Properties, Watchers, Builds, and Governance. The POM View tab is active, showing the XML code for the pom.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.jfrog.test</groupId>
  <artifactId>multi</artifactId>
  <version>2.20-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>Simple Multi Modules Build</name>
  <modules>
    <module>multi1</module>
    <module>multi2</module>
    <module>multi3</module>
  </modules>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
    
```

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

Set Me Up

Tool
Maven

Releases ② Snapshots ②
libs-release libs-snapshot

Plugin Releases ② Plugin Snapshots ②
plugins-release plugins-snapshot

Mirror Any ②
remote-repos

Generate Settings

This screenshot shows the 'Set Me Up' dialog for the Maven tool. It includes dropdown menus for 'Releases' (set to 'libs-release') and 'Snapshots' (set to 'libs-snapshot'), and another pair for 'Plugin Releases' (set to 'plugins-release') and 'Plugin Snapshots' (set to 'plugins-snapshot'). A checkbox for 'Mirror Any' is checked, and a dropdown menu for 'remote-repos' is open. At the bottom is a green 'Generate Settings' button.

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.

Sample snippet from `settings.xml`

```

...
<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>\{DE5ede\}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*.

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**.

Set Me Up



Tool

Maven

Generate Maven Settings

Repository

libs-release-local

General

Click on "Generate Maven Settings" in order to resolve artifacts through Virtual or Remote repositories.

Deploy

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

```
1 <distributionManagement>
2   <repository>
3     <id>Arti4-Demo</id>
4     <name>Arti4-Demo-releases</name>
5     <url>http://10.100.1.110:8081/artifactory/libs-release-local</url>
6   </repository>
7 </distributionManagement>
```

Remember that you can not deploy build artifacts to remote or virtual repositories, 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

Maven Artifactory Plugin

Overview

Using a set of plugins, Artifactory supports Maven builds on commonly used build servers such as [Jenkins](#), [TeamCity](#) and [Bamboo](#), however, in the last few years, the popularity of cloud-based build servers has grown.

Some examples are, [Travis CI](#), [drone.io](#) and [Codeship](#). The problem is that none of these are "pluggable" in the traditional way. Therefore, to support Maven builds running on cloud-based build servers, you can use the **Maven Artifactory Plugin**.

Source Code Available!

The Maven Artifactory Plugin is an [open-source project on GitHub](#) which you can freely browse and fork.

Page Contents

- [Overview](#)
- [Usage](#)
- [Configuration](#)
- [Reading Environment Variables and System Properties](#)

Through the Maven Artifactory Plugin, Artifactory is fully integrated with Maven builds and allows you to do the following:

1. Attach properties to published artifacts in Artifactory metadata.
2. Capture a [BuildInfo](#) object which can be passed to the [Artifactory REST API](#) to provide a fully traceable build context.
3. Automatically publish all build artifacts at the end of the build.

Usage

The Maven Artifactory Plugin coordinates are `org.jfrog.buildinfo:artifactory-maven-plugin:2.4.0`. It can be viewed on [Bintray](#), and can be download via the [JCenter Repository](#).

A typical build plugin configuration would be as follows:

```
<build>
  <plugins>
    ...
    <plugin>
      <groupId>org.jfrog.buildinfo</groupId>
      <artifactId>artifactory-maven-plugin</artifactId>
      <version>2.4.0</version>
      <inherited>false</inherited>
      <executions>
        <execution>
          <id>build-info</id>
          <goals>
            <goal>publish</goal>
          </goals>
          <configuration>
            <deployProperties>
              <gradle>awesome</gradle>
              <review.team>qa</review.team>
            </deployProperties>
            <publisher>
              <contextUrl>https://oss.jfrog.org</contextUrl>
              <username>deployer</username>
              <password>{DESede}...</password>
              <repoKey>libs-release-local</repoKey>
              <snapshotRepoKey>libs-snapshot-local</snapshotRepoKey>
            </publisher>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

The plugin's invocation phase is "`validate`" by default and we recommend you don't change it so the plugin is called as early as possible in the lifecycle of your Maven build.

Configuration

The example above configures the Artifactory *publisher*, to deploy build artifacts either to the `releases` or the `snapshots` repository of the `public` OSS instance of Artifactory when `mvn deploy` is executed.

However, the Maven Artifactory Plugin provides many other configurations which you can see by running `mvn -X validate` and are displayed below:

```
<deployProperties> .. </deployProperties>
<artifactory>
    <envVarsExcludePatterns> .. </envVarsExcludePatterns>
    <envVarsIncludePatterns> .. </envVarsIncludePatterns>
    <includeEnvVars>true/false</includeEnvVars>
    <timeoutSec>N</timeoutSec>
</artifactory>
<publisher>
    <contextUrl> .. </contextUrl>
    <username> .. </username>
    <password> .. </password>
    <repoKey> .. </repoKey>
    <snapshotRepoKey> .. </snapshotRepoKey>
    <publishArtifacts>true/false</publishArtifacts>
    <publishBuildInfo>true/false</publishBuildInfo>
    <excludePatterns> .. </excludePatterns>
    <includePatterns> .. </includePatterns>
    <filterExcludedArtifactsFromBuild>true/false</filterExcludedArtifactsFromBuild>
</publisher>
<buildInfo>
    <agentName> .. </agentName>
    <agentVersion> .. </agentVersion>
    <buildName> .. </buildName>
    <buildNumber> .. </buildNumber>
    <buildNumbersNotToDelete> .. </buildNumbersNotToDelete>
    <buildRetentionMaxDays>N</buildRetentionMaxDays>
    <buildRetentionCount>N</buildRetentionCount>
    <buildUrl> .. </buildUrl>
    <principal> .. </principal>
    <vcsRevision> .. </vcsRevision>
</buildInfo>
<licenses>
    <autoDiscover>true/false</autoDiscover>
    <includePublishedArtifacts>true/false</includePublishedArtifacts>
    <runChecks>true/false</runChecks>
    <scopes> .. </scopes>
    <violationRecipients> .. </violationRecipients>
</licenses>
<blackDuck>
    <appName> .. </appName>
    <appVersion> .. </appVersion>
    <autoCreateMissingComponentRequests>true/false</autoCreateMissingComponentRequests>
    <autoDiscardStaleComponentRequests>true/false</autoDiscardStaleComponentRequests>
    <includePublishedArtifacts>true/false</includePublishedArtifacts>
    <reportRecipients> .. </reportRecipients>
    <scopes> .. </scopes>
</blackDuck>
```

<code><deployProperties></code>	Specifies properties you can attach to published artifacts. For example:
	<pre> <deployProperties> <groupId>\${project.groupId}</groupId> <artifactId>\${project.artifactId}</artifactId> <version>\${project.version}</version> </deployProperties> </pre>
<code><artifactory></code>	Specifies whether environment variables are published as part of BuildInfo metadata and which include or exclude patterns are applied when variables are collected
<code><publisher></code>	<p>Defines an Artifactory repository where build artifacts should be published using a combination of a <code><contextUrl></code> and <code><repoKey></code> / <code><snapshotRepoKey></code>.</p> <p>Build artifacts are deployed if the <code>deploy</code> goal is executed and only after all modules are built</p>
<code><buildInfo></code>	Updates BuildInfo metadata published together with build artifacts. You can configure whether or not BuildInfo metadata is published using the <code><publisher></code> configuration.
<code><licenses></code>	Controls auto-discovery and violation monitoring of third-party licenses
<code><blackDuck></code>	Configures Artifactory BlackDuck integration. Note that you need to specify <code><runChecks>true</runChecks></code> to activate it.

Reading Environment Variables and System Properties

Every build server provides its own set of environment variables. You can utilize these variables when configuring the plugin as shown in the following example:

```

<publisher>
    <contextUrl>{{ARTIFACTORY_CONTEXT_URL| "https://oss.jfrog.org" }}</contextUrl>
    ...
</publisher>
<buildInfo>

<buildNumber>{{DRONE_BUILD_NUMBER|TRAVIS_BUILD_NUMBER|CI_BUILD_NUMBER|BUILD_NUMBER| "33
3" }}</buildNumber>
    <buildUrl>{{DRONE_BUILD_URL|CI_BUILD_URL|BUILD_URL}}</buildUrl>
</buildInfo>

```

Any plugin configuration value can contain several `{{ ... }}` expressions. Each expression can contain a single or multiple environment variables or system properties to be used.

The expression syntax allows you to provide enough variables to accommodate any build server requirements according to the following rules:

- Each expression can contain several variables, separated by a '`|`' character to be used with a configuration value
- The last value in a list is the default that will be used if none of the previous variables is available as an environment variable or a system property

For example, for the expression `{{v1|v2|"defaultValue"}}` the plugin will attempt to locate environment variable v1, then system property v1, then environment variable or system property v2, and if none of these is available, "defaultValue" will be used.

If the last value is not a string (as denoted by the quotation marks) and the variable cannot be resolved, `null` will be used (for example, for expression `{{v1|v2}}` where neither v1 nor v2 can be resolved).

You can attach additional artifacts to your module using the [Build Helper Maven Plugin](#).

Keeping your Artifactory publisher credentials secure

If you prefer to keep your Artifactory publisher credentials (username and password) secure (rather than providing them as free text in the plugin configuration), we recommend storing them as environment variables or system properties and have the plugin read them when needed. Since the usual Maven deploy does not support environment variables or system properties in `settings.xml`, this capability is unique to the Maven Artifactory Plugin.

Examples

The projects listed below provide working examples of using the plugin:

- [Maven Artifactory Plugin](#)

Projects on cloud build servers:

- [TeamCity Artifactory Plugin](#)
- [Jenkins Artifactory Plugin](#)
- [Travis CI Netty Project](#)

Working with Gradle

Overview

Artifactory provides tight integration with Gradle. All that is needed is a simple modification of your `build.gradle` script file with a few configuration parameters.

Both the new and older publishing mechanisms of Gradle are supported, however some of the steps to configure the [Gradle Artifactory Plugin](#) depend on the version you are using, and these are detailed in the documentation pages.

The Gradle Artifactory Plugin can be used whether you are running builds using a CI server, or running standalone builds. In either case, you should note the following points:

1. CI Server Integration

When running Gradle builds in your continuous integration server, we recommend using one of the Artifactory Plugins for Jenkins, TeamCity or Bamboo.

You can use your build server UI to configure resolving and publishing artifacts through Artifactory to capture exhaustive build information.

2. Standalone Integration

The Gradle Artifactory plugin offers a simple DSL to perform the following steps in your Gradle build:

- a. Define the default dependency resolution from Artifactory.
- b. Define configurations that publish artifacts to Artifactory after a full (multi-module) successful build.
- c. Define properties that should be attached to published artifacts in Artifactory metadata.
- d. Capture and publish a [build-info](#) object to the Artifactory build-info REST API to provide a fully traceable build context.

Source Code Available!

This Gradle Artifactory Plugin is an [open source project on GitHub](#) which you can freely browse and fork.

The following sections describe the main configuration steps and provide a sample Gradle script that shows the information you need to get started using Gradle with Artifactory.

Page Contents

- Overview
- Configuring Artifact Resolution
 - Using the Gradle Build Script Generator
 - Provisioning Dynamic Settings for Users
- Sample Build Script and Properties
- Running Gradle
- Dependency Declaration Snippets
- Watch the Screencast

Read More

- [Gradle Artifactory Plugin](#)

Configuring Artifact Resolution

Using the Gradle Build Script Generator

With Artifactory's [Gradle Build Script Generator](#), you can easily create a Gradle init script that handles resolution.

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 Gradle Settings". You can now specify the settings you want to configure for Gradle.

Plugin/Libs Resolver	The repository that should be used to resolve plugins/libraries
Use Maven/Use Ivy	When checked, specifies that resolving should be done using the Maven/Ivy pattern

Libs Publisher	The repository that should be used to publish libraries
Use Maven/Use Ivy	When checked, specifies that library should be published using a Maven/Ivy descriptor
Repository Layout	Specifies the layout of the corresponding repository

Once you have configured the settings for Gradle you can click "Generate Settings" to generate and save the `build.gradle` and `gradle.properties` file.

Set Me Up

Tool

Gradle

[Back to Set Me Up](#)

Plugin Resolver <div style="display: flex; justify-content: space-between;"> Repository Key ? Libs Resolver Libs Publisher </div> <div style="display: flex; justify-content: space-between;"> <div style="width: 33%;"> <input type="text" value="plugins-release"/> <div style="font-size: small; margin-top: 5px;"><input type="checkbox"/> Use Maven ? <input type="checkbox"/> Use Ivy ?</div> </div> <div style="width: 33%;"> <input type="text" value="libs-release"/> <div style="font-size: small; margin-top: 5px;"><input checked="" type="checkbox"/> Use Maven ? <input checked="" type="checkbox"/> Use Ivy ?</div> </div> <div style="width: 33%;"> <input type="text" value="libs-release-local"/> <div style="font-size: small; margin-top: 5px;"><input checked="" type="checkbox"/> Use Maven ? <input checked="" type="checkbox"/> Use Ivy ?</div> </div> </div>
Repository Layout <div style="display: flex; justify-content: space-between;"> maven-2-default ivy-default maven-2-default </div>
Generate Settings

Provisioning Dynamic Settings for Users

Artifactory lets you 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 [Provisioning Build Tool Settings](#) section under [Filtered Resources](#).

Sample Build Script and Properties

You can download sample scripts from the JFrog [GitHub](#) public repository.

Running Gradle

For Gradle to build your project and upload generated artifacts to Artifactory, you need to run the following command:

```
gradle artifactoryPublish
```

For more details on building your projects with Gradle, please refer to the [Gradle Documentation](#).

Getting debug information from Gradle

We highly recommend running Gradle with the `-d` option to get useful and readable information if something goes wrong with your build.

Dependency Declaration Snippets

Artifactory can provide you with dependency declaration code snippets that you can simply copy into the **Gradle Dependency Declaration** section of your `build.gradle` file.

In the **Artifact Repository Browser** of the **Artifacts** module, drill down in the repository tree and select a relevant artifact. Under the **Dependency Declaration** section, select **Gradle** to display the corresponding dependency declaration that you can copy into your `build.gradle` file.

The screenshot shows the Artifactory interface. On the left, there's a tree view of repositories: libs-snapshot-local, maven-playground, npm-local, nuget-local, p2-local, plugins-release-local, plugins-snapshot-local, pom-repo, download.eclipse.org-cache, and gradle-libs-cache. Under gradle-libs-cache, there are sub-folders like appliance, classworlds, com (with google and thoughtworks), commons, commons-parent, commons-codec, commons-collections, commons-io, commons-lang, commons-logging, and java. A specific artifact, xstream-1.3.1.jar, is selected in the tree. On the right, the details for this artifact are shown. The 'Info' tab displays basic metadata: Name: xstream-1.3.1.jar, Repository Path: gradle-libs-cache:com/thoughtworks/xstream/xstream/1.3.1/xstream-1.3.1.jar, Created: 30-04-15 11:31:02 UTC, Deployed by: admin, Size: 421.29 KB, Last Modified: 06-12-08 15:50:15 UTC, Module ID: com.thoughtworks:xstream:xstream:1.3.1, Licenses: Not Found, and Downloaded: 0. Below the info tab is the 'Dependency Declaration' tab, which contains a code snippet for Gradle:

```
1 | compile(group: 'com.thoughtworks.xstream', name: 'xstream', version: '1.3.1')
```

The 'Build Tool' dropdown is set to 'Gradle'. Other options include Maven, Ivy, and Sbt. There are also tabs for Virtual Repository Associations (gradle, libs-release, libs-snapshot, plugins-release, plugins-snapshot, remote-repos) and Checksums (MD5 and SHA-1).

Watch the Screencast

Gradle Artifactory Plugin

Overview

The Gradle Artifactory Plugin supports all versions of Gradle including Gradle 2.0 and is available from the [Gradle Plugins Gallery](#).

The plugin supports both Gradle Configurations and Publications for deploying artifacts to Artifactory.

Using the Correct Plugin ID

The Gradle Artifactory Plugin ID depends on two parameters:

- Whether you are using the old or the new Gradle publishing mechanism
- The version of the plugin

To simplify these documentation pages, the plugin ID will be referred to using a <PLUGIN ID> placeholder.

This placeholder should be replaced with the plugin IDs listed in the table below:

	Plugin version 2.x	Plugin version 3.0.1 and higher
When using Gradle publications	artifactory-publish	com.jfrog.artifactory
When using Gradle configurations	artifactory	com.jfrog.artifactory-upload

Page Contents

- Overview
 - Using the Correct Plugin ID
- Download and Installation
 - Automatic Installation
 - Manual Installation
- Configuration
 - Using the Artifactory Plugin DSL
 - The Artifactory Project Publish Task
 - Controlling the Published Modules
 - Controlling the Build Name Number in BuildInfo
- Examples

Download and Installation

Automatic Installation

Build script snippet for use in all Gradle versions

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath "org.jfrog.buildinfo:build-info-extractor-gradle:3.1.0"
    }
}
apply plugin: "com.jfrog.artifactory"
```

Build script snippet for use in Gradle 2.1 and above

```
plugins {
    id "com.jfrog.artifactory" version "3.1.0"
}
```

Currently the "plugins" notation cannot be used for applying the plugin for sub projects, when used from the root build script

Manual Installation

The latest plugin jar file can be downloaded from the [JFrog Artifactory public demo](#). Copy the jar into your gradle home plugins directory (`~/.gradle/plugins`).

Then add the following line to your project build script:

```
buildscript.dependencies.classpath files(new File(gradle.gradleUserHomeDir,  
'plugins/build-info-extractor-gradle-3.0.1-uber.jar'))
```

Configuration

Using the Artifactory Plugin DSL

The Gradle Artifactory plugin can be configured using its own Convention DSL inside the `build.gradle` script of your root project.

The syntax of the Convention DSL is described below:

Mandatory items within the relevant context are prefixed with '+'. All other items are optional.

```
artifactory {  
    +contextUrl = 'http://repo.myorg.com/artifactory'      //The base Artifactory URL if  
not overridden by the publisher/resolver  
    publish {  
        contextUrl = 'http://repo.myorg.com/artifactory'      //The base Artifactory URL for  
the publisher  
        //A closure defining publishing information  
        repository {  
            +repoKey = 'integration-libs'          //The Artifactory repository key to publish to  
            +username = 'deployer'                 //The publisher user name  
            password = 'deployerPaS*'           //The publisher password  
            ivy {  
                //Optional section for configuring Ivy publication. Assumes Maven repo layout  
if not specified  
                ivyLayout = '[organization]/[module]/[revision]/[type]s/ivy-[revision].xml'  
                artifactLayout =  
'[organization]/[module]/[revision]/[module]-[revision](-[classifier]).[ext]'  
                mavenCompatible = true //Convert any dots in an [organization] layout value to  
path separators, similar to Maven's groupId-to-path conversion. True if not specified  
            }  
        }  
        defaults {  
            //This closure defines defaults for all 'artifactoryPublish' tasks of all  
projects the plugin is applied to  
            publications ('ivyJava','mavenJava','foo')                      //Optional list  
of publications (names or objects) to publish.  
            properties = ['qa.level': 'basic', 'q.os': 'win32, deb, osx'] //Optional map  
of properties to attach to all published artifacts  
            /*  
             * The properties closure in the "defaults" task uses the following syntax:  
             properties {  
                 publicationName 'group:module:version:classifier@type', key1:'value1',
```

```

key2:'value2', ...
}
    publicationName: A valid name for a publication of the project. You can use
all to apply the properties to all publications.
    group:module:version:classifier@type: A filter that specifies the artifacts to
which properties should be attached.
    The filter may contain wildcards: * for all characters or ? for a single character.
    key:'value': A list of key/value properties that will be attached to the
published artifacts matching the filter.
/*
    properties {                                         //Optional
closure to attach properties to artifacts based on a list of artifact patterns per
project publication
    foo '*:*:*:@*', platform: 'linux', 'win64'           //The property
platform=linux,win64 will be set on all artifacts in foo publication
        mavenJava 'org.jfrog:*:*:@*', key1: 'val1'          //The property
key1=val1 will be set on all artifacts part of the mavenJava publication and with
group org.jfrog
        all 'org.jfrog:shared:1.??:@*', key2: 'val2', key3: 'val3' //The
properties key2 and key3 will be set on all published artifacts (all publications)
with group:artifact:version
            //equal to org.jfrog:shared:1.?
    }
    publishBuildInfo = true    //Publish build-info to Artifactory (true by
default)
    publishArtifacts = true   //Publish artifacts to Artifactory (true by default)
    publishPom = true         //Publish generated POM files to Artifactory (true by
default).
    publishIvy = true         //Publish generated Ivy descriptor files to Artifactory
(true by default).
}
}
resolve {
    contextUrl = 'http://repo.myorg.com/artifactory'    //The base Artifactory URL for
the resolver
    repository {
        +repoKey = 'libs-releases'  //The Artifactory (preferably virtual) repository
key to resolve from
        username = 'resolver'       //Optional resolver user name (leave out to use
anonymous resolution)
        password = 'resolverPaS*'  //The resolver password
        maven = true                //Resolve Maven-style artifacts and descriptors
(true by default)
        ivy {
            //Optional section for configuring Ivy-style resolution. Assumes Maven repo
layout if If not specified
            ivyLayout = ...
            artifactLayout = ...
            mavenCompatible = ...
        }
    }
}
// Redefine basic properties of the build info object
clientConfig.setIncludeEnvVars(true)
clientConfig.info.addEnvironmentProperty('test.adding.dynVar',new
java.util.Date().toString())
clientConfig.info.setBuildName('new-strange-name')
clientConfig.info.setBuildNumber('' + new
java.util.Random(System.currentTimeMillis()).nextInt(20000))

```


Using the old Gradle publishing mechanism?

› [Expand source](#)

If you are using the old Gradle publishing mechanism, you need to replace the above defaults closure with the following one:

```
defaults {
    //This closure defines defaults for all 'artifactoryPublish' tasks of all
    projects the plugin is applied to
    publishConfigs ('a','b','foo')                                //Optional list
    of configurations (names or objects) to publish.

    //The
    'archives' configuration is used if it exists and no configuration is specified
    mavenDescriptor = '/home/froggy/projects/proj-a/fly-1.0.pom' //Optional
    alternative path for a POM to be published (can be relative to project baseDir)
    ivyDescriptor = 'fly-1.0-ivy.xml'                            //Optional
    alternative path for an ivy file to be published (can be relative to project baseDir)
    properties = ['qa.level': 'basic', 'q.os': 'win32, deb, osx'] //Optional map
    of properties to attach to all published artifacts

    /*
     * The properties closure in the "defaults" task uses the following syntax:
     properties {
        configuration 'group:module:version:classifier@type', key1:'value1',
        key2:'value2', ...
     }
     configuration: A configuration that is a valid name of a configuration of the
     project. You can use all to apply the properties to all configurations.
     group:module:version:classifier@type: An artifact specification filter for
     matching the artifacts to which properties should be attached.
     The filter may contain wildcards: * for all characters or ? for a single character.
     key:'value': A list of key/value(s) properties that are attached to the
     published artifacts matching the filter.
     */
     properties {                                                 //Optional
    closure to attach properties to artifacts based on a list of artifact patterns per
    project configuration
        foo '*:*:*:@*', platform: 'linux', 'win64'           //The property
    platform=linux,win64 will be set on all artifacts in foo configuration
        archives 'org.jfrog:*:*:@*', key1: 'val1'             //The property
    key1=val1 will be set on all artifacts part of the archives configuration and with
    group org.jfrog
        all 'org.jfrog:shared:1.?:@*', key2: 'val2', key3: 'val3' //The
    properties key2 and key3 will be set on all published artifacts (all configurations)
    with group:artifact:version
                                                //equal to
    org.jfrog:shared:1.?
    }
    publishBuildInfo = true //Publish build-info to Artifactory (true by
    default)
    publishArtifacts = true //Publish artifacts to Artifactory (true by default)
    publishPom = true      //Publish generated POM files to Artifactory (true
    by default)
    publishIvy = false     //Publish generated Ivy descriptor files to
    Artifactory (false by default)
}
```

The Artifactory Project Publish Task

The Artifactory Publishing Plugin creates an `artifactoryPublish` Gradle task for each project the plugin is applied to. The task is configured by the `publish` closure of the plugin.

You can configure the project-level task directly with the task's `artifactoryPublish` closure, which uses identical Syntax to that of the plugin's `publish.defaults` closure.

```
artifactoryPublish {  
    skip = false //Skip build info analysis and publishing (false by default)  
    contextUrl = 'http://repo.myorg.com/artifactory'  
    publications ('a','b','c')  
    properties = ['qa.level': 'basic', 'q.os': 'win32, deb, osx']  
    properties {  
        c '**:***:*@*', cProperty: 'only in c'  
    }  
    clientConfig.publisher.repoKey = 'integration-libs'  
    clientConfig.publisher.username = 'deployer'  
    clientConfig.publisher.password = 'deployerPas'  
}
```

Controlling the Published Modules

To control which modules of a multi-module build, are published to Artifactory you can:

1. Omit the 'artifactory' plugin from a project that does not publish anything. Note that this does not apply to the root project that contains the `convention` object, and so, requires the plugin to be applied.
2. Activate the corresponding `artifactoryPublish` Gradle task manually for each project to which you wish to apply the plugin.
For example in our [Gradle project example](#) you can run:

Activating the plugin manually

```
./gradlew clean api:artifactory:publish shared:artifactoryPublish
```

3. Use the `artifactoryPublish.skip` flag to deactivate analysis and publication.

Controlling the Build Name Number in BuildInfo

By default, `BuildInfo` is published with a build name constructed from the name of your root project and a build number that is the start date of the build.

You can control the build name and number values by specifying the following properties respectively:

Specifying the build name and number

```
buildInfo.build.name=my-super-cool-build  
buildInfo.build.number=r9001
```

The above properties are specified as standard [Gradle properties](#).

You can also control the build name and number from within the `Convention` object DSL as shown at the end of the [Convention DSL syntax](#) above.

Examples

Project examples which use the Gradle Artifactory Plugin are available [here](#).

Working with Ivy

Overview

Artifactory fully supports working with Ivy both as a source for artifacts needed for a build, and as a target to deploy artifacts generated in the build process.

For Ivy to work with Artifactory, the following files must be present and configured:

1. **The Ivy settings file:** `ivysettings.xml` is used to configure resolution and deployment of artifacts using repositories in Artifactory.
2. **The Ivy modules file:** `ivy.xml` is where the project's modules and dependencies are declared.
3. **The Ant build file:** `build.xml` is used to execute the ANT tasks that will, in turn, use Ivy for resolution and deployment of artifacts.

Ivy Settings - `ivysettings.xml`

The `ivysettings.xml` file holds a chain of Ivy resolvers for both regular artifacts and Ivy module files. These are used to resolve and publish (i.e. deploy) artifacts.

There are two ways to configure resolvers in `ivysettings.xml` in order to set up Ivy to work with Artifactory:

1. Automatically, using the Artifactory Ivy Settings Generator
2. Manually defining [IBiblio](#) and [URL](#) resolvers.

Page Contents

- Overview
- Ivy Settings - `ivysettings.xml`
 - Automatic Settings with Artifactory's Ivy Settings Generator
 - Provisioning Dynamic Settings for Users
 - Defining a Manual Resolver
 - The IBiblio Resolver
 - The URL Resolver
 - Using a Chain Resolver
- Ivy Modules - `ivy.xml`
- Ant Build - `build.xml`
- Publishing to Artifactory
 - Using a Dedicated Settings File for Deployment

Automatic Settings with Artifactory's Ivy Settings Generator

To begin quickly, you can define credentials and resolver settings using Artifactory's Ivy Settings Generator. This generates a URL resolver suitable for resolution.

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

Since the **Libs Repository** field only includes virtual or remote repositories, none of these will be suitable for deployment, and you need to modify the deployment URL to point to a local repository.

Set Me Up

Tool
Ivy

Back to Set Me Up

Libs Repository (7)
libs-release

Libs Repository Layout (7)
maven-2-default

Libs Resolver Name (7)

Use Ibiblio Resolver (7) Maven 2 Compatible (7)

Generate Settings

Choose an Ivy Repository Layout
Be sure to select layout that is compatible with Ivy such as **ivy-default** or a custom layout that you have defined.

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 details, please refer to [Provisioning Build Tool Settings](#) under [Filtered Resources](#).

Defining a Manual Resolver

The Ibiblio Resolver

This resolver is only used to resolve dependencies. By default, it assumes artifacts in your repository are laid-out in the popular and standard Maven 2 format (which may not always be the case).

The [Ibiblio resolver](#) can resolve artifacts from remote Maven 2 HTTP repositories, and if you use version ranges it relies on `maven-metadata.xml` files in the remote repository to gather information on the available versions.

To use the Ibiblio resolver, add the following to your `ivysettings.xml` file:

```
<resolvers>
  <ibiblio name="artifactory" m2compatible="true"
root="http://localhost:8080/artifactory/libs-releases"/>
</resolvers>
```

The URL specified in the `root` property must point to an Artifactory repository. In the above example, it is the pre-configured `libs-releases` virtual repository.

The `m2compatible` property configures the resolver with an artifact pattern that follows the standard Maven 2 layout.

The URL Resolver

The URL resolver can be used to resolve dependencies and/or for deployment of both regular artifacts and Ivy module files.

To publish or resolve artifacts to or from Artifactory, you need to configure a URL resolver with the pattern that matches your target repository layout for both Ivy and artifact files.

For example:

```

<!-- Authentication required for publishing (deployment). 'Artifactory Realm' is the
realm used by Artifactory so don't change it. -->
<credentials host="localhost" realm="Artifactory Realm" username="admin"
passwd="password"/>
<resolvers>
    <url name="artifactory-publish">
        <!-- You can use m2compatible="true" instead of specifying your own pattern
-->
        <artifact pattern=
            "http://localhost:8080/artifactory/ivy-local/[organization]/[module]/[revision]/[artif
act]-[revision].[ext]" />
        <ivy
pattern="http://localhost:8080/artifactory/ivy-local/[organization]/[module]/[revision]
]/ivy-[revision].xml" />
    </url>
</resolvers>

```

The URL resolver uses HTML href analysis to learn about the available versions of a remote artifact. This is less reliable than using an IBiblio resolver, however it works well with remote Artifactory servers.

Using a Chain Resolver

You can combine resolver definitions under a chain resolver in Ivy which uses a set of sub resolvers to resolve dependencies and for publishing.

For details please refer to the Ivy documentation for [Chain Resolver](#).

Ivy Modules - ivy.xml

ivy.xml files contain a list of dependency declarations that must be resolved for the build.

In the **Artifact Repository Browser** of the **Artifacts** module, you can obtain dependency declaration snippets by selecting either an Ivy module, or a POM artifact, and copying the Ivy **Dependency Declaration** section into your ivy.xml file.

The screenshot shows the Artifactory interface for managing artifacts. On the left, the 'Tree' view displays a hierarchy of local repositories: RubyGems-local, dlsfq, ext-release-local, ext-snapshot-local, and ivy-local. Under ivy-local, there are sub-folders for gems, org.apache.ivy.example, and version, with specific versions like 1.0-local-20120928001043 and 1.0-local-20120929113803. The right side of the screen shows the details for the 'ivy-1.0-local-20120928001043.xml' file. The 'Info' tab provides metadata such as Name (ivy-1.0-local-20120928001043.xml), Repository Path (ivy-local/org.apache.ivy.example/version/1.0-local-20120928001043/ivys/ivy-1.0-local-20120928001043.xml), and Deployment details. Below this, the 'Dependency Declaration' tab shows the XML code for the dependency:

```

<dependency org="org.apache.ivy.example" name="version" rev="1.0-local-20120928001043">
    <artifact name="version" type="ivy" ext="xml"/>
</dependency>

```

The 'Ivy' tab is highlighted with a red border, indicating it is the active tool for this declaration.

Ant Build - build.xml

To work with Ivy to resolve dependencies, you need to use `<ivy:configure/>` in your `build.xml` file. This will load the [Ivy settings](#) from `ivysettings.xml`.

Artifacts are resolved using `<ivy:retrieve/>`.

For details please refer to the Ivy documentation for [Ant Tasks](#).

Publishing to Artifactory

You can use the `<ivy:publish>` command to configure Ivy to deploy your artifacts into Artifactory using the specified resolver.

For example:

```
<ivy:publish resolver="artifactory-publish" overwrite="true">
  <!--
    Use overwrite="true" if you wish to overwrite existing artifacts
    and publishivy="false" if you only want to publish artifacts not module descriptors
  -->
  <artifacts/>
</ivy:publish>
```

Using a Dedicated Settings File for Deployment

If you have specified deployment settings with the required credentials in a dedicated settings file, you can refer to them by assigning a unique ID.

For example, the following code snippet assigns the deployment settings with the id `ivy.publish.settings`:

```
<ivy:settings id="ivy.pub.settings" file="publish_to_artifactory_settings.xml"/>
```

Then, the publishing task points to these settings using the following attribute in the `publish` element:

```
settingsRef="ivy.pub.settings"
```

For details please refer to the Ivy documentation for [Ant Tasks](#).

Troubleshooting

Overview

This page provides tips to solve common problems that users have encountered.

If you do not find an answer to your issue here, please try to get help using the [Artifactory Users Mailing Lists](#)

- Archive using Nabble
- Subscribe using mailman

Artifactory Does Not Start Up

▼ There are no log file entries in
`$ARTIFACTORY_HOME/logs/artifactory.log`

Cause	An exception was thrown (possibly by your servlet container) before Artifactory loaded its logging mechanism.
Resolution	Check your servlet container's <code>localhost.log</code> file. For more information, please refer to Artifact Log Files .

Page Contents

- [Overview](#)
- [Artifactory Does Not Start Up](#)

End of Life

Here is the list of versions and their End of Life (EoL) date:

Version	Release Date	EOL Date
3.1.1	09-Feb-2014	09-Aug-2015
3.1.0	15-Dec-2013	15-Jun-2015
3.0.4	27-Oct-2013	27-Apr-2015
3.0.3	11-Aug-2013	11-Feb-2015
3.0.2	08-Jul-2013	08-Jan-2015
3.0.1	13-May-2013	13-Nov-2014
3.0.0	21-Apr-2013	21-Oct-2014
2.6.7	27-Jun-2013	27-Dec-2014
2.6.6	20-Dec-2012	20-Jun-2014
2.6.5	12-Nov-2012	12-May-2014
2.6.4	20-Oct-2012	20-Apr-2014
2.6.3	02-Aug-2012	02-Feb-2014
2.6.2	23-Jul-2012	23-Jan-2014
2.6.1	23-May-2012	23-Nov-2013

2.6.0	06-May-2012	05-Nov-2013
2.5.2	25-Apr-2012	25-Oct-2013
2.5.1.1	06-Mar-2012	06-Sep-2013
2.5.1	15-Feb-2012	15-Aug-2013
2.5.0	31-Jan-2012	31-Jul-2013
2.4.2	29-Nov-2011	29-May-2013
2.4.1	16-Nov-2011	16-May-2013
2.4.0	08-Nov-2011	08-May-2013
2.3.2	03-Jan-2012	03-Jul-2013
2.3.1	14-Feb-2011	14-Aug-2012
2.3.0	20-Oct-2010	20-Apr-2012
2.2.3	25-Apr-2010	25-Oct-2011
2.2.2	22-Mar-2010	22-Sep-2011
2.2.1	17-Feb-2010	17-Aug-2011
2.2.0	08-Feb-2010	08-Aug-2011
2.1.3	21-Dec-2009	21-Jun-2011
2.1.2	03-Nov-2009	03-May-2011
2.1.1	21-Oct-2009	21-Apr-2011
2.1.0	05-Oct-2009	05-Apr-2011
2.0.8	08-Sep-2009	08-Mar-2011
2.0.7	12-Aug-2009	12-Feb-2011
2.0.6	14-May-2009	14-Nov-2010
2.0.5	08-Apr-2009	08-Oct-2010

2.0.4	31-Mar-2009	31-Sep-2010
2.0.3	17-Mar-2009	17-Sep-2010
2.0.2	18-Feb-2009	18-Aug-2010
2.0.1	09-Feb-2009	09-Aug-2010
2.0.0	05-Jan-2009	05-Jul-2010

Release Notes

- [Pivotal Cloud Foundry JFrog Artifactory Tile Release Notes](#)
- [Artifactory 4.3](#)
- [Artifactory 4.2](#)
 - [Artifactory 4.2.2](#)
 - [Artifactory 4.2.1](#)
- [Artifactory 4.1](#)
 - [Artifactory 4.1.3](#)
 - [Artifactory 4.1.2](#)
- [Artifactory 4.0](#)
 - [Artifactory 4.0.2](#)
 - [Artifactory 4.0.1](#)
- [Artifactory 3.9](#)
 - [Artifactory 3.9.2](#)
 - [Artifactory 3.9.1](#)
- [Artifactory 3.8](#)
- [Artifactory 3.7](#)
- [Artifactory 3.6](#)
- [Artifactory 3.5](#)
 - [Artifactory 3.5.3](#)
 - [Artifactory 3.5.2](#)
 - [Artifactory 3.5.1](#)
- [Artifactory 3.4](#)
 - [Artifactory 3.4.2](#)
 - [Artifactory 3.4.1](#)
- [Artifactory 3.3](#)
 - [Artifactory 3.3.1](#)
 - [Artifactory 3.3.0.1](#)
- [Artifactory 3.2](#)
 - [Artifactory 3.2.2](#)
 - [Artifactory 3.2.1](#)
- [Artifactory 3.1](#)
 - [Artifactory 3.1.1](#)
- [Artifactory 3.0](#)
 - [Artifactory 3.0.4](#)
 - [Artifactory 3.0.3](#)
 - [Artifactory 3.0.2](#)
 - [Artifactory 3.0.1](#)
- [Artifactory 2.x](#)
 - [Artifactory 2.6.7](#)
 - [Artifactory 2.6.6](#)
 - [Artifactory 2.6.5](#)
 - [Artifactory 2.6.4](#)
 - [Artifactory 2.6.3](#)
 - [Artifactory 2.6.2](#)
 - [Artifactory 2.6.1](#)
 - [Artifactory 2.6.0](#)
 - [Artifactory 2.5.2](#)
 - [Artifactory 2.5.1.1](#)
 - [Artifactory 2.5.1](#)
 - [Artifactory 2.5.0](#)
 - [Artifactory 2.4.2](#)
 - [Artifactory 2.4.1](#)
 - [Artifactory 2.4.0](#)
 - [Artifactory 2.3.4.1](#)

- Artifactory 2.3.4
- Artifactory 2.3.3
- Artifactory 2.3.2
- Artifactory 2.3.1
- Artifactory 2.3.0
- Artifactory 2.2.5
- Artifactory 2.2.4
- Artifactory 2.2.3
- Artifactory 2.2.2
- Artifactory 2.2.1
- Artifactory 2.2.0
- Artifactory 2.1.3
- Artifactory 2.1.2
- Artifactory 2.1.1
- 1.3.0-RC2
- 1.3.0-RC1
- 1.3.0-beta-6
- 1.3.0-beta-5
- 1.3.0-beta-4
- 1.3.0-beta-3

Artifactory 4.3

November 22, 2015

We are pleased to announce the availability of Artifactory 4.3!

In addition to implementing several bug fixes and minor improvements, this release introduces the following main enhancements:

API Keys

You may now authenticate REST API calls with an [API key](#) that you can create and manage through your profile page or through the [REST API](#).

Package Search

Run a search based on a specific packaging format with dedicated search parameters for the selected format. Performance is improved since search is restricted to repositories with the specified format only.

Support Zone

Generate the information that our support team needs to provide the quickest resolution for your support tickets.

Dependency rewrite for Bower and NPM

Remove the dependence on external artifact resources for Bower and Npm. When downloading a Bower or Npm package, Artifactory will analyze the package metadata to evaluate if it needs external dependencies. If so, Artifactory will download the dependencies, host them in a remote repository cache, and then rewrite the dependency specification in the original package's metadata and point it to the new location within Artifactory.

Improved support for S3 object store

JFrog S3 object store now supports S3 version 4 allowing you to sign AWS with Signature v4. Multi-part upload and very large files over 5 GB in size are now also supported.

Main Updates

This release includes the following main updates:

1. Authentication using [API keys](#)
2. [Package search](#)
3. Convenient Support Zone page for submitting support requests
4. Improved support for [S3 object store](#) with support for S3 version 4.
5. Automatic rewrite of external dependencies for [Npm](#) and [Bower](#) repositories.
6. HTTP request object is now accessible from [Realms](#) closures in user plugins ([RTFACT-8514](#)).
7. REST API to [download a complete release](#) from VCS repositories.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

[Click to download the latest Artifactory Pro Version.](#)

[Click to download Artifactory OSS version.](#)

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 4.2

October 18, 2015

We are pleased to announce the availability of Artifactory 4.2!

In addition to implementing several bug fixes and minor improvements, this release introduces a [Debian Artifactory installation](#) and [Deploy to Virtual repositories](#).

Debian Installation

Artifactory can now be installed as a Debian package.

Deploy to Virtual

Artifactory now supports deploying artifacts to a virtual repository via REST API. All you need to do is specify a local repository aggregated within the virtual repository that will be the deploy target.

OAuth Login

Artifactory now supports login and authentication using OAuth providers. Currently, Google, Open ID and GitHub Enterprise are supported.

Artifactory Query Language (AQL)

AQL has been greatly extended to include several additional domains, including **Build** and **Archive.Entry** as primary domains, giving you much more flexibility in building queries.

Main Updates

This release includes the following main updates:

1. Artifactory installation as a [Debian package](#)
2. Deploy artifacts to a [virtual repository](#)
3. Authentication using OAuth providers
4. [AQL](#) has been extended to include additional domains
5. Improvements to [Smart Remote Repositories](#)
6. REST API to retrieve storage information
7. Overwrite NuGet pre-release packages without delete permissions
8. Pushing Docker images to Bintray is now also supported for Docker V2 repositories
9. Several minor improvements to the UI

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

[Click to download the latest Artifactory Pro Version.](#)

Click to download Artifactory OSS version.

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 4.2.1

November 1, 2015

We are pleased to announce the availability of Artifactory 4.2.1!

In addition to implementing several bug fixes and minor improvements, this release introduces the following main enhancements:

OAuth Provider

Cloud Foundry UAA is now supported as an OAuth provider.

SHA256

In addition to SHA1 and MD5, [SHA2](#) checksums are now supported also.

Main Updates

This release includes the following main updates:

1. Artifactory now supports Cloud Foundry UAA for OAuth authentication.
2. Since Artifactory now fully supports the Bower client, support for [older versions of Bower](#) (below v1.5) that were using `bower-art-resolver` beta version is now deprecated.
3. Internet Explorer compatibility issues have been fixed.
4. Artifactory's HTTP client has been upgraded to version 4.5.
5. Automatic license analysis is now also triggered when [deploying RPMs](#).
6. SHA256 calculation is now available, on demand via the [UI](#) or via [REST API](#).
7. Several minor improvements to the UI.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

Click to download the latest Artifactory Pro Version.

Click to download Artifactory OSS version.

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 4.2.2

November 5, 2015

We are pleased to announce the availability of Artifactory 4.2.2!

This is a minor update that provides several bug fixes.

For a complete list of changes please refer to our [JIRA Release Notes](#).

[Download](#)

Click to download the latest Artifactory Pro Version.
Click to download Artifactory OSS version.

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!

The Artifactory Team

Artifactory 4.1

September 8, 2015

We are pleased to announce the availability of Artifactory 4.1!

In addition to implementing several bug fixes and minor improvements, this release introduces **Smart Remote Repositories** and **Virtual Docker Repositories**.

Smart Remote Repositories

Define a repository in a remote Artifactory instance as your remote repository and enjoy advanced features such as automatic detection, synchronized properties and delete indications.

Virtual Docker Repositories

Aggregate all of your Docker repositories under a single Virtual Docker Repository, and access all of your Docker images through a single URL.

Main Updates

This release includes the following main updates:

1. Support for Smart Remote Repositories
2. Docker enhancements with virtual Docker repositories and detailed Docker image info
3. Context sensitive help
4. Custom message
5. Stash search results
6. Enhanced AQL supporting queries in the **Build** domain
7. Downloading a folder from the UI and REST API

8. Ability to browse the content of tag and tar.gz files
9. Full support for [Bower](#) (out of Beta)
10. Several minor improvements to the UI

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

[Click to download the latest Artifactory Pro Version.](#)

[Click to download Artifactory OSS version.](#)

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 4.1.2

September 20, 2015

We are pleased to announce the availability of Artifactory 4.1.2!

This is a minor update that provides a fix for clients, such as Maven, that do not use preemptive authentication.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

[Click to download the latest Artifactory Pro Version.](#)

[Click to download Artifactory OSS version.](#)

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 4.1.3

September 27, 2015

We are pleased to announce the availability of Artifactory 4.1.3!

This is a minor update that provides a fix for Docker Login with anonymous access.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

[Click to download the latest Artifactory Pro Version.](#)

[Click to download Artifactory OSS version.](#)

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!

The Artifactory Team

Artifactory 4.0

August 2, 2015

We are excited to announce the release of Artifactory 4.0!

JFrog is excited to announce the release of Artifactory 4.0. This release presents major changes in Artifactory providing a fresh look 'n feel with a completely revamped user interface and many other changes described below.

New User Interface

JFrog-Artifactory's user interface has been rebuilt from scratch to provide the following benefits:

- **Intuitive:** Configuration wizards for easy repository management
- **Fresh and modern:** New look and feel providing a rich user experience
- **Set Me Up:** Convenient code snippets to support simple copy/paste integration with software clients and CI tools
- **Context-focused repositories:** Repositories are optimized to calculate metadata for single package types
- **Easy access control:** Easily implement your access policies with intuitive user, group and permission management
- **Smart tables:** Group and filter any data that is presented in tables

Page Contents

- [New User Interface](#)
- [Groovy 2.4 for User Plugins](#)
- [Tomcat 8 as the Container](#)
- [System Requirements](#)
 - Java
 - Browsers
- [Breaking Changes](#)
 - User Plugins
 - Multiple Package Type Repositories
- [Download](#)
- [Installation and Upgrade](#)

Groovy 2.4 for User Plugins

JFrog Artifactory 4 supports Groovy 2.4 letting you enjoy the latest Groovy language features when writing User Plugins.

We strongly recommend you verify that all of your current User Plugins continue to work seamlessly with this version of Groovy.

Tomcat 8 as the Container

JFrog Artifactory 4.0 only supports Tomcat 8 as its container for both RPM and standalone versions. If you are currently using a different container (e.g. Websphere, Weblogic or JBoss), please refer to [Upgrading When Using External Servlet Containers](#) for instructions on how to migrate to Tomcat 8.

System Requirements

Java

JFrog Artifactory 4.0 requires [Java 8](#)

Browsers

JFrog Artifactory 4.0 has been tested with the latest versions of Google Chrome, Firefox, Internet Explorer and Safari.

Breaking Changes

User Plugins

Some features of Groovy 2.4 are not backward compatible with Groovy 1.8. As a result, plugins based on Groovy 1.8 may need to be upgraded to support Groovy 2.4.

Multiple Package Type Repositories

JFrog Artifactory 4.0 requires you to specify a single package type for each repository. For the specified package type, Artifactory will calculate metadata and work seamlessly with the corresponding package format client. For example, a repository specified as **Docker** will calculate metadata for Docker images and work transparently with the **Docker** client.

Artifactory will not prevent you from uploading packages of a different format to any repository, however, metadata for those packages will not be calculated, and the corresponding client for those packages will not recognize the repository. For example, if you upload a Debian package to a NuGet repository, Debian metadata will not be calculated for that package, and the Debian client will not recognize the NuGet repository.

You may specify a repository as **Generic** and upload packages of any type, however, for this type of repository, Artifactory will not calculate any metadata and will effectively behave as a simple file system. These repositories are not recognized by clients of any packaging format.

If your system currently includes repositories that support several package types, please refer to [Single Package Type Repositories](#) to learn how to migrate them to single package type repositories.

Download

Click to download the latest [Artifactory Pro Version](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Artifactory 4.0.1

August 9, 2015

We are pleased to announce the availability of Artifactory 4.0.1!

This is a minor update that provides significant enhancements to our support for Docker, additional UI improvements as well as several bug fixes.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

Click to download the latest [Artifactory Pro Version](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 **To receive automatic notifications whenever there is a new release of Artifactory, please watch us on Bintray.**

Enjoy Artifactory!

JFrog Artifactory Team

Artifactory 4.0.2

August 12, 2015

We are pleased to announce the availability of Artifactory 4.0.2!

This is a minor update that provides support for the latest Docker client 1.8.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

Click to download the latest [Artifactory Pro Version](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 **To receive automatic notifications whenever there is a new release of Artifactory, please watch us on Bintray.**

Enjoy Artifactory!

JFrog Artifactory Team

Artifactory 3.9

June 21, 2015

We are pleased to announce the availability of Artifactory 3.9!

In addition to implementing several bug fixes and minor improvements, this release introduces support for Git Large File Storage (LFS) repositories and remote Docker registries.

Git LFS Repositories

In addition to managing all your binary software artifacts, Artifactory can now be used as your Git LFS server to manage your large binaries, such as media files, audio samples, videos, datasets, etc.

Artifactory's support for Git LFS provides highly performant, reliable and consistent access to media assets and other binary files while exercising security and access control over your LFS repositories.

Remote Docker Registries

Artifactory can now proxy and cache remote docker registries providing highly performant, reliable and consistent access to images stored in remote URLs,

Main Updates

This release includes the following main updates:

1. Support for Git LFS repositories
2. Support for remote Docker registries
3. REST API to reload user plugins

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

Click to download latest [Artifactory Pro Servlet war or a standalone service](#).

Click to download latest [Artifactory Pro RPM Distribution](#).

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.9.1

June 24, 2015

We are pleased to announce the availability of Artifactory 3.9.1!

This is a minor update that provides a fix for list remote folder browsing [issue](#).

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

[Click to download latest Artifactory Pro Servlet war or a standalone service.](#)

[Click to download latest Artifactory Pro RPM Distribution.](#)

[Click to download Artifactory Pro Free Trial.](#)

[Click to download Artifactory OSS version.](#)

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 *To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).*

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.9.2

July 06, 2015

We are pleased to announce the availability of Artifactory 3.9.2!

This is a minor update that provides a fix for Docker tag REST API and improves P2 virtual repository handling.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

[Click to download latest Artifactory Pro Servlet war or a standalone service.](#)

[Click to download latest Artifactory Pro RPM Distribution.](#)

[Click to download Artifactory Pro Free Trial.](#)

[Click to download Artifactory OSS version.](#)

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 *To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).*

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.8

May 31, 2015

We are pleased to announce the availability of Artifactory 3.8!

In addition to implementing several bug fixes and minor improvements, this release introduces support for Vagrant boxes.

Vagrant Boxes

Artifactory's support for Vagrant provides reliable and consistent access to internal boxes, sharing and distribution, smart search, fine-grained access control as a secure private repository and more.

Main Updates

This release includes the following main updates:

1. Support for [Vagrant boxes](#)
2. Fixed support for NuGet package explorer

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

Click to download latest version [Artifactory Pro Servlet war or a standalone service](#).

Click to download latest version [Artifactory Pro RPM Distribution](#).

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.7

May 17, 2015

We are pleased to announce the availability of Artifactory 3.7!

In addition to implementing many bug fixes and minor improvements, this release introduces several significant feature updates to Artifactory:

Docker V2

Artifactory leaps forward supporting the latest changes in Docker.

Range Requests

Artifactory now supports [HTTP Range Requests](#) which enable smart and efficient multi-threaded download of large files by letting you specify a range of bytes that should be downloaded when files are streamed to you.

Architecture-independent Debian Packages

You can now specify "all" as the architecture of your Debian packages. Artifactory will add the package to the index of all architectures available under the same Distribution and Component.

Main Updates

This release includes the following main updates:

1. Support for [Docker Registry V2 API](#).
2. Support for [HTTP Range Requests](#).
3. Support for [architecture-independent Debian packages](#).

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.6

April 12, 2015

We are pleased to announce the availability of Artifactory 3.6!

In addition to implementing many bug fixes and minor improvements, this release introduces several significant feature updates to Artifactory:

VCS Repositories

Artifactory can now proxy version control systems enabling uniform access to your source files. This can simplify your build scripts while providing stable, reliable and secure access to your source code.

S3 Object Storage

For enterprise users, Artifactory fully supports the S3 protocol for distributed file systems. This means your binary filestore can reside on the cloud giving you unlimited scalability, security and disaster recovery capabilities.

Bower Repositories (Beta)

Provision Bower packages directly from Artifactory to the Bower command line tool. Enjoy reliable and consistent access to remote Bower packages, and automatic calculation of metadata for Bower packages stored in our local repositories. Access multiple Bower registries through a single URL.

Main Updates

This release includes the following main updates:

1. [VCS Repositories](#)

2. S3 Object Storage (requires an enterprise license)
3. Bower Repositories (beta)
4. NuGet license and vulnerabilities information added to the Black Duck Code Center Integration
5. Optimizations to AQL
6. Push Docker Tag to Bintray

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.5

February 1, 2015

We are pleased to announce the availability of Artifactory 3.5!

In addition to implementing many bug fixes and minor improvements, this release introduces several significant feature updates to Artifactory:

[Artifactory Query Language](#)

Artifactory Query Language is an easy-to-use query language that gives unprecedented abilities to search through all repositories managed by Artifactory.

Multi-push Replication

Enterprise users can now simultaneously push-replicate from one source repository to multiple target repositories in Enterprise installations.

Main Updates

This release includes the following main updates:

1. [Artifactory Query Language](#)
2. [Multi-push replication](#)
3. REST API to fully automate [pushing releases to Bintray](#)
4. [Improved HA cluster upgrade, stability and node recovery](#)
5. Support for [database password encryption](#)

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards compatible, so

upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 *To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).*

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.5.3

March 22, 2015

We are pleased to announce the availability of Artifactory 3.5.3!

This minor release includes improvements to database transaction management, as well as many bug fixes and minor improvements.

Main Updates

This release includes the following main updates:

1. [Promoting Docker images](#)
2. [AQL](#) supported in Artifactory Open Source version
3. Full support for proxying remote YUM repositories
4. Improved database transaction management when uploading/downloading big binaries
5. Bug fixes and improvements to the License Control

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 *To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).*

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.5.2

February 23, 2015

We are pleased to announce the availability of Artifactory 3.5.2!

This minor release includes improvements to NuGet local repository performance and Debian repository support, as well as several other minor fixes.

Main Updates

This release includes the following main updates:

1. Improved performance of NuGet local repositories
2. Debian calculation is done in one atomic operation per distribution
3. Binary GC deletes the largest binaries first

Version 3.5.2.1 also resolves a regression with Multi-push replication to servers with a trial license.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.5.1

February 4, 2015

We are pleased to announce the availability of Artifactory 3.5.1!

This is a minor update that provides improvements in Artifactory adding [support for the Oracle Maven Repository](#) and opening Artifactory Query Language for regular authenticated users (who are not administrators).

Main Updates

This release includes the following main updates:

1. Support for the Oracle Maven Repository
2. Use of [Artifactory Query Language REST API](#) is available for regular authenticated users

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards-compatible, so upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 *To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).*

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.4

September 30, 2014

We are pleased to announce the availability of Artifactory 3.4!

The main feature update for this release is the addition of support for Docker and PyPI packages!

Artifactory fully supports PyPI packages in local, remote and virtual repositories, and is compatible with the pip command line tool.

Support for Docker extends to local repositories and allows fine-grained security and access control to Docker registries and images.

Main Updates

This release includes the following main updates:

1. Support for [Docker](#) packages
2. Support for [PyPI](#) packages
3. NuGet license analysis in conjunction with our new [MSBuild Artifactory Plugin](#)
4. Multiple [NPM](#) bug fixes and improvements
5. User plugins [lib directory](#)

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards compatible, and upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 *To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).*

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.4.2

November 30, 2014

We are pleased to announce the availability of Artifactory 3.4.2!

This release features support for [flexible NPM layouts](#) and [NPM scoped packages](#), as well as enhancements to our Debian, NuGet and PyPI support.

In addition, we have implemented many bug fixes as well as several improvements.

Main Updates

Tomcat 7.0.56 upgrade

For standalone zip installations, please pay close attention to the [Standalone Installation Instructions](#).

This release includes the following main updates:

1. Support for [flexible Npm layouts](#) and [Npm 2.0 scoped packages](#).
2. The default Tomcat container has been upgraded to version 7.0.56
3. Support for adding custom HTTP headers for remote downloads with user plugins.
4. Better handling of concurrent remote file downloads.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards compatible, so upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 *To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).*

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.4.1

October 21, 2014

We are pleased to announce the availability of Artifactory 3.4.1!

This is minor update that provides improvements in Artifactory's support for Npm and NuGet.

Main Updates

This release includes the following main updates:

1. Support the latest Npm client - v2.1.
2. Significant improvements in performance when searching for NuGet artifacts.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards compatible, and upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

For a full list of features available in your edition of Artifactory please refer to the [Artifactory Comparison Matrix](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.3

July 13, 2014

We are pleased to announce the availability of Artifactory 3.3

The main feature update for this release is the addition of support for Debian packages!

You can now upload and download Debian packages, and manage the public and private keys used to sign them, both through the UI and through the REST API.

Main Updates

This release includes the following main updates:

1. Support for [Debian packages](#)
2. Support for [NuGet Batch queries](#)
3. REST API for encrypted user password

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards compatible and upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.3.1

September 9, 2014

We are pleased to announce the availability of Artifactory 3.3.1

This minor release provides improvements to Artifactory's support for Debian packages a fix for two important security issues; one related to a vulnerability with user authentication and the other related to outgoing SSL connections.

Main Updates

1. Improved support for Debian packages including better client support, improved API and automatic generation of default architectures.
2. Fixed a vulnerability that allows an authenticated user to obtain information from the Artifactory database under a specific set of conditions.
3. Fixed a security issue that affects outgoing SSL connections from Artifactory. For more details, please refer to [CVE-2014-3577](#).

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards compatible and upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.3.0.1

Aug 10, 2014

We are pleased to announce the availability of Artifactory 3.3.0.1

Important update for users of RubyGems.org

This update is only relevant for users who are using Artifactory support for RubyGems repositories.

Recently, there were reports that a large number of requests coming from Artifactory servers were causing excessive load on the [RubyGems.org](#) repository. A joint investigation led by RubyGems.org and JFrog R&D revealed that this behavior is the result of a misconfiguration of Artifactory servers. The investigation showed that many redundant requests were being sent from non-Ruby clients such as Apache Maven. These Maven clients apparently utilize a virtual repository which aggregates RubyGems.org together with other non-Ruby repositories resulting in irrelevant requests being sent to RubyGems.org.

Solution

JFrog is offering two options to resolve this issue:

1. Upgrade Artifactory to V3.3.0.1. This version prevents download and remote listing of non RubyGems resources from remote RubyGems repositories when [RubyGems Support](#) is enabled in Artifactory.
2. If you are not currently able to upgrade to this latest release, we recommend that you manually configure any [virtual repositories](#) in Artifactory with the recommended settings in accordance with [best practice](#) so that only relevant requests are sent to RubyGems.org.

Virtual Repository Best Practice

A virtual repository should not aggregate repositories that are not actually accessed through it.

For example, a virtual repository intended for a Maven client should not aggregate the RubyGems.org repository. This kind of misconfiguration dramatically slows down the resolution process causing an overall impedance on system performance. Similarly, working against the global virtual repository **repo** is also against best practice since it aggregates all other Artifactory repositories. **Repo** propagates any request it receives to all the local and remote repositories that it aggregates which significantly slows down the resolution process.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).



To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.2

March 30, 2014

We are pleased to announce the availability of Artifactory 3.2

The main feature update for this release is the addition of support for NPM packages!

Main Updates

This release includes the following main updates:

1. Support for [NPM](#) packages including publishing and download as well as full support for the [npm](#) command-line tool
2. [Cleanup of virtual repository caches](#) for improved performance and reduction in storage
3. Option to [block](#) downloads from the global virtual repository
4. Option to [force authentication for NuGet](#) enabled virtual repositories
5. Improved support for last modified property in various [WebDav](#) clients

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards compatible and upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).



To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.2.1

June 1, 2014

We are pleased to announce the availability of Artifactory 3.2.1

This minor release provides improvements in networking and proxy authentication.

Upgrading to the latest HTTP client improves Artifactory's outbound networking, making communication with remote repositories, and other external resources more efficient and reliable.

In addition, users of [npm](#), [NuGet](#) and [RPM](#) will be happy to hear that several bugs related to using Artifactory with these repositories, have been fixed.

Main Updates

1. Upgraded to support the latest [HTTP client](#).
2. Support for [NTLMv2 proxy authentication](#).
3. Allow administrator to disable the [Remember Me](#) feature on the user login screen.
4. Remote repository download log entry updated to include [the content length and bandwidth](#).
5. Substantially improved [database table cleanup in PostgreSQL](#).

Users of Websphere and Weblogic

If you are using IBM Websphere, please note the following [additional configuration steps](#) you need to take when upgrading to Artifactory 3.2.1.

If you are using Weblogic, please note the following [system properties](#) you need to define when using Artifactory 3.2.0 or later.

Version 3.2.1.1 also resolves a regression with usage of encrypted passwords in remote repositories.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards compatible and upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.2.2

June 22, 2014

We are pleased to announce the availability of Artifactory 3.2.2

This minor release introduces a new feature that enables you to encrypt passwords stored in the Artifactory global configuration file, as well as additional security related enhancements,

Main Updates

1. [Master Key Encryption](#) for configuration passwords
2. Improved CSRF protection
3. Disabled password fields auto-complete
4. Distribution management UI improved to support additional packaging systems ([NuGet](#), [npm](#), [RubyGems](#))

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards compatible and upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 *To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).*

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.1

December 15, 2013

We are pleased to announce Artifactory 3.1 - with High Availability!

Artifactory 3.1 introduces High Availability, which comes included with Artifactory Enterprise Value Pack.

With Artifactory HA, in addition to all the Pro features, you can prevent service disruptions, ensure smooth maintenance and provide the best response times.

For more details on installing and running Artifactory HA please refer to [Artifactory High Availability](#).

Main Updates

In addition to High Availability, Artifactory 3.1 includes the following main updates:

1. New API for per-file download statistics, available to user plugin and the REST API.
2. Improved Black Duck Code Center integration for open source governance.

For a complete list of changes please refer to our [JIRA Release Notes](#).

Download

Click to download [Artifactory Pro 3.1 Free Trial](#).

Click to download [Artifactory 3.1 OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

Upgrading to Artifactory 3.1

If you are upgrading to Artifactory 3.1, we recommended that you increase the `MaxPermSize` value by adding `-XX:MaxPermSize=256m` to the `$JAVA_OPTIONS` variable

For a standalone installation this is done in `$ARTIFACTORY_HOME/etc/default`.

For a service or RPM installation this is done in `/etc/opt/jfrog/artifactory/default`.

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.1.1

February 9, 2014

We are pleased to announce the availability of Artifactory 3.1.1

This minor release contains many improvements and bug fixes as well as a few new features.

Main Updates

Servlets API

This version requires a servlet container that supports Servlets API v3.0. For an updated list of supported servlet containers please refer to [Deploying on Servlet Containers - Done](#)

This release includes the following main updates:

1. New administrator page to [Monitor Storage](#)
2. NuGet feed and download now works even if the remote repository is offline
3. New REST API allows you to [search for the latest artifact version](#) based on the value assigned to the "version" property
4. New REST API that allows you to [Get, Set, Update or Delete](#) a repository configuration for replication
5. Major performance boost with the new connection pool
6. Selected RPM metadata is added as properties
7. Error responses from REST API calls are now returned in [JSON](#) format
8. [Black Duck Code Center](#) integration improved
9. Property values with up to 4000 characters are now supported
10. Improved the performance of local/cache repository replication
11. Support for the forthcoming JDK 8

Version 3.1.1.1 also fixes the following security vulnerability:

1. Security vulnerability due to [XStream CVE-2013-7285](#)

For a complete list of changes please refer to our [JIRA Release Notes](#).

Artifactory HA users please note: This version contains changes that are not backwards compatible and upgrading requires a full cluster restart.

Download

Click to download [Artifactory Pro Free Trial](#).

Click to download [Artifactory OSS version](#).

Installation and Upgrade

For installation instructions please refer to [Installing Artifactory](#).

To upgrade to this release from your current installation please refer to [Upgrading Artifactory](#).

 To receive automatic notifications whenever there is a new release of Artifactory, please watch us on [Bintray](#).

Enjoy Artifactory!
The Artifactory Team

Artifactory 3.0

We are excited to announce the major release of Artifactory 3!

21 April 2013

JFrog announces the release of the third major version of its flagship product, Artifactory, The Binaries Repository Manager. This release delivers a huge performance boost and a fresh user interface.

Four years following Artifactory 2, the Artifactory 3 major release is available to all users (Open-source, Pro and SaaS versions).

Major Performance Boost

All the common usage scenarios such as checksum-based storage, binaries promotion, mirroring, repository management and remote proxying have been optimized for speed. Depending on a particular usage scenario Artifactory 3 performs x10 to x100 times faster than Artifactory 2.x! On top of that, Artifactory 3 also has a considerably lower memory footprint.

We have reimplemented from scratch the underlying metadata storage (retiring Jackrabbit, which has served us well for the last 4 years). With this change, Artifactory utilizes SQL and relational databases in a unique way, significantly improving CI build performance over competitors and previous versions of Artifactory.

User Interface Changes

The user interface has also undergone a facelift, providing a bright, clean and uncluttered display for users.

Black Duck Code Center Integration

In addition, the Artifactory 3 now supports BlackDuck Code-Center integration for open source license governance and vulnerability control.

Remote Searches with Bintray Integration

Artifactory users can now search for remote artifacts and get information about latest OSS releases from Bintray.com.

Uses Tomcat 7 as the Default Container

Further, default distribution uses Tomcat 7 with both the RPM and the standalone versions of Artifactory.

System Requirements

Artifactory 3.0 requires Java 7

Supported Databases

- Derby (embedded - zero configuration)
- Oracle version 10g and above
- MySQL **version 5.5 and above** with InnoDB

- PostgreSQL is supported from version 3.0.2 and above
- Microsoft SQL Server 2008 and above

Removed / Obsolete

We have discontinued support for XML Metadata, including related user plugin methods and REST API

XML XPath search

[Sync resource REST API](#)

Installation and Upgrade Instructions

For detailed installation instructions please refer to our [user guide](#)

Upgrading from 2.x versions requires only a simple process of exporting and then importing the current repository data.

For detailed upgrade instructions please refer to our [upgrade guide](#)

Changes

For a complete list of bug fixes and improvements in this version please see the [JIRA Release Notes](#)

Enjoy Artifactory!

JFrog Artifactory Team

Artifactory 3.0.1

We are pleased to announce the availability of Artifactory 3.0.1!

This release includes minor improvements to the installation and several bug fixes. Notable issues addressed by this version:

1. NuGet 2.5 support
2. Improved **Bintray info panel** performance
3. **HTML Content browsing** also supported for files outside of an archive
4. Fixed failure in archive indexing with duplicate (case insensitive) entries
5. Preventing duplicate permission target entries

For a complete list of resolved issues please see the [JIRA release notes](#).

Additional info about Artifactory 3 can be found [here](#).

Artifactory Pro 3.0.1 free trial is available for [immediate download](#).

The Artifactory OSS 3.0.1 is available from JFrog's web site or directly from Bintray.

Instructions for upgrading to 3.0.1 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

Artifactory 3.0.2

We are pleased to announce the availability of Artifactory 3.0.2!

07 July 2013

This release brings back support for PostgreSQL database and includes several bug fixes and improvements. Notable issues addressed by this version:

1. **PostgreSQL support**
2. **YUM repository** improvements and bug fixes
3. Users can **override the response Content-Type header** using a custom property

4. Artifacts count and storage size added to the [user plugins](#) public API
5. Simple patterns can be used in [latest version search](#)
6. Force [maven metadata recalculation](#) REST API
7. Improvements and bug fixes around storage GC

For a complete list of resolved issues please see the [JIRA release notes](#).

Additional info about Artifactory 3 can be found [here](#).

Artifactory Pro 3.0.2 free trial is available for [immediate download](#).
The Artifactory OSS 3.0.2 is available from JFrog's web site or directly from Bintray.

Instructions for upgrading to 3.0.2 from previous versions can be found [here](#).

To get automatic notification for new releases of Artifactory you can watch Artifactory on [Bintray](#).

Enjoy Artifactory!

The Artifactory Team

Artifactory 3.0.3

We are pleased to announce the availability of Artifactory 3.0.3!

11 Aug 2013

This release adds support for serving RubyGems, as well as major upgrade for NuGet packages support. It also includes important bug fixes and improvements.

Notable issues addressed by this version:

1. [RubyGems support](#) with gems metadata calculation for local gem hosting, proxying remote gem repositories and merging multiple gem repositories under a single virtual repository
2. Major improvements in [NuGet support](#), including the entire range of queries and filtering
3. Fixed [wrong maven metadata calculation](#) under heavy concurrency
4. Properties replication now properly works for remote and virtual repositories

For a complete list of resolved issues please see the [JIRA release notes](#).

Additional info about Artifactory 3 can be found [here](#).

Artifactory Pro 3.0.3 free trial is available for [immediate download](#).
The Artifactory OSS 3.0.3 is available from JFrog's web site or directly from Bintray.

Instructions for upgrading to 3.0.3 from previous versions can be found [here](#).

To get automatic notification for new releases of Artifactory you can watch Artifactory on [Bintray](#).

Enjoy Artifactory!

The Artifactory Team

Artifactory 3.0.4

We are pleased to announce the availability of Artifactory 3.0.4!

27 Oct 2013

This release adds support for replication content filtering, as well as major upgrades for Crowd integration and JIRA authentication support. It also includes important bug fixes and improvements.

Notable issues addressed in this version:

1. Improved [Crowd integration](#) with support for authentication against JIRA User Server
2. New user plugin interception points for [replication content filtering](#)
3. New user plugin interception point for [custom content expiry](#)
4. Major performance boost to [YUM metadata](#) calculation
5. Fixed [cache deployment denial](#) for users without the "Any Remote" permission

For a complete list of resolved issues please see the [JIRA release notes](#).

Additional info about Artifactory 3 can be found [here](#).

Artifactory Pro 3.0.4 free trial is available for [immediate download](#).

The Artifactory OSS 3.0.4 is available from [JFrog's web site](#) or directly from [Bintray](#).

Instructions for upgrading to 3.0.4 from previous versions can be found [here](#).

To get automatic notifications about new Artifactory releases you can watch Artifactory on [Bintray](#).

Enjoy Artifactory!

The Artifactory Team

Artifactory 2.x

- [Artifactory 2.6.7](#)
- [Artifactory 2.6.6](#)
- [Artifactory 2.6.5](#)
- [Artifactory 2.6.4](#)
- [Artifactory 2.6.3](#)
- [Artifactory 2.6.2](#)
- [Artifactory 2.6.1](#)
- [Artifactory 2.6.0](#)
- [Artifactory 2.5.2](#)
- [Artifactory 2.5.1.1](#)
- [Artifactory 2.5.1](#)
- [Artifactory 2.5.0](#)
- [Artifactory 2.4.2](#)
- [Artifactory 2.4.1](#)
- [Artifactory 2.4.0](#)
- [Artifactory 2.3.4.1](#)
- [Artifactory 2.3.4](#)
- [Artifactory 2.3.3](#)
- [Artifactory 2.3.2](#)
- [Artifactory 2.3.1](#)
- [Artifactory 2.3.0](#)
- [Artifactory 2.2.5](#)
- [Artifactory 2.2.4](#)
- [Artifactory 2.2.3](#)
- [Artifactory 2.2.2](#)
- [Artifactory 2.2.1](#)
- [Artifactory 2.2.0](#)
- [Artifactory 2.1.3](#)
- [Artifactory 2.1.2](#)
- [Artifactory 2.1.1](#)
- [1.3.0-RC2](#)
- [1.3.0-RC1](#)
- [1.3.0-beta-6](#)
- [1.3.0-beta-5](#)
- [1.3.0-beta-4](#)
- [1.3.0-beta-3](#)

1.3.0-RC2

Artifactory 1.3.0-RC2

We are pleased to announce the availability of Artifactory 1.3.0-RC2.

Major Features and Changes in This Release

- Deleting (undeploying) artifacts immediately updates the relevant maven-metadata.xml.
- Ability to upload and deploy a bundle of multiple artifacts (zip archive) arranged in m2 format.
- Bug fixes.

Enjoy 😊

The Artifactory Team

Artifactory is available for immediate download from [here](#).
You can browse the full JIRA release notes [here](#).

Installing

Please refer to the [Installing](#) section of the current user guide, which still applies.

Upgrading from 1.3.0-RC1

Just replace the war file. If you are running on Tomcat make sure to also remove the exploded artifactory webapp directory.

Upgrading from 1.2.2-rc0 through 1.3.0-beta-6.1

Upgrading Artifactory from older version can be done in two ways:

1. From the UI
2. By using the `artadmin` command line tool

Upgrading Using the Web UI

*Upgrading from the UI is currently supported only **from previous 1.3.0-x versions**.*

1. From your old Artifactory installation run Full System Export and save the export to a destination directory.
2. Perform a new clean server installation of the new Artifactory (It should not contain repository data or your customized version of the Artifactory configuration xml file).
3. Install the new Artifactory version, go to *Admin/Export&Import/System/Import System*, select your previous export target directory and let the import run. That's it!

Upgrading Using the `artadmin` Tool

While upgrading from the web UI is easy and straightforward, sometimes the upgrade process can take a long time, especially with very large repositories. In such cases the original web request may time out and the upgrade progress will proceed in the background.

Therefore, if you wish to monitor the progress of the upgrade process in the most reliable way and gain access to more advanced dump options, it is recommended to use the `artadmin` command-line tool.

Here are step-by-step instructions:

1. Stop your old Artifactory.
2. Execute the `artadmin dump` command on the old `$ARTIFACTORY_HOME` or on a copy of it (recommended):

```
artadmin dump $ARTIFACTORY_HOME
```

This will generate a `dumpExport` folder under the current execution directory with the old repository data in a format suitable for importing into a current Artifactory.

NOTE: By default, caches (e.g. `repo1`) are *not exported*. To export caches add the `--caches` parameter.

3. Perform a new clean server installation of the new Artifactory (It should not contain repository data or your customized version of the Artifactory configuration xml file).
4. Start the new Artifactory.
5. Import the `dumpExport` folder into Artifactory, either from the UI, as explained in the previous section or by running:

```
artadmin import dumpExport --username admin --password password
```

The `artadmin` output will display the progress of the import. NOTE: If the `artadmin` process is killed, the import will still run in Artifactory.

Important Information

Please read this carefully before installing or upgrading:

1. If you are running Artifactory under JDK 1.6 you **MUST** use **JDK 1.6u4** and above due to incompatibilities with older JAXB versions embedded in previous JDK 1.6 releases. JDK 1.5 users are not affected by this.
2. If you have used a previous version of Artifactory it is highly recommended to:
 - a. Use a fresh version of Artifactory from the distribution and not try to patch the previous installation.
 - b. Clear your browser's cache before using this version.
3. The default Artifactory user for the standalone installation has been changed to `artifactory` (instead of `jetty`). You might have to update the provided install script or your file system permissions accordingly.
4. Please note that not all documentation has been updated yet to reflect the latest changes in this release.

Important information about the import process

During the import binary artifacts are copied over into a working copy and are imported into Artifactory in by a background thread. This speeds up the import process dramatically and makes Artifactory ready for serving requests as soon as possible. The background import process will take some time to complete, depending on the size of your repository. During this time Artifactory might perform more slowly than usual, but it will still serve any artifact immediately. The log does provide visible information about the progress of this process.

Repeating the upgrade process

Should you wish to repeat the upgrade process from scratch, make sure to remove the `$ARTIFACTORY_HOME/data` folder from your new Artifactory installation before doing so.

Major Features and Changes in release 1.3.0-rc-1

- Flexible validation for remote checksums: Generate if absent (the default), Fail, Ignore and generate and Pass-through (please see the UI for description of each policy).
- Many bug fixes and stability improvements.
- Better [MySQL](#) support.
- Improved support for Safari.
- Artifactory and artadmin are now under a single downloadable distribution.
- Better overall performance.

Major Features and Changes in release 1.3.0-beta-6

- Artifactory is 100% configurable via the newly designed user interface
- Support for running Artifactory with [MySQL](#) for greater scalability, performance and tools support.
- Support for running Artifactory under [Websphere](#).
- Install script for automatic set up under a dedicated [Tomcat](#) instance.
- Ability to remove deployed versions with all their sub-artifacts in one go.
- Dynamic log tailing from the UI.
- Global offline flag for disconnected environments.
- Out of the box support for running standalone Artifactory as a [Windows service](#) (in addition to the existing Unix service installer).
- Unified [artadmin](#) CLI tool for all administrative tasks.
- Greatly improved backup and restore speed.
- Many internal performance and concurrency improvements.
- The usual bug fixes.

Major Features and Changes in release 1.3.0-beta-5

- In place incremental backup.
- Performance improvements around security authentication and authorization.
- Fixed a UI bug with remote repositories cache access control setup.

Major Features and Changes in release 1.3.0-beta-4

- Improved [LDAP](#) Support - support for multiple DNs, for non-anonymous binding against MS Active Directory and for sub-tree searches.
- Fixed a critical issue with deployment using the lightweight HTTP wagon (RTFACT-629).
- UI and performance improvements and the usual bug fixes.

Major Features and Changes in release 1.3.0-beta-3

- Group support with an intuitive powerful UI for groups management.

- Improved permissions control for including/excluding specific repository paths (e.g. disallow certain users/groups from downloading sources). Simple management - no regexp gurus required.
- Support for delete permission and for preventing users from overriding existing artifacts.
- Revised repository tree browsing using tabs and effective permissions per item view.
- Improved import speed by committing binary imports asynchronously.
- Improved [#upgrade](#) procedure.
- Better transaction management using Spring Modules for JCR.
- Even faster in-place incremental backup.
- Low-level admin-only direct WebDAV access to the underlying jackrabbit repository (use: <http://localhost:8081/artifactory/jcr/default/> to connect).
- The usual bug fixes, stability and speed improvements.

The complete release notes from Jira are available [here](#):

[1.3.0-rc-2](#)
[1.3.0-rc-1](#)
[1.3.0-beta-6](#)
[1.3.0-beta-5](#)
[1.3.0-beta-4](#)
[1.3.0-beta-3](#)

1.3.0-RC1

Artifactory 1.3.0-RC1

We are pleased to announce the availability of Artifactory 1.3.0-RC1.

Major Features and Changes in this Release

- Flexible validation for remote checksums: Generate if absent (the default), Fail, Ignore and generate and Pass-through (please see the UI for description of each policy).
- Many bug fixes and stability improvements.
- Better [MySQL](#) support.
- Improved support for Safari.
- Artifactory and artadmin are now under a single downloadable distribution.
- Better overall performance.

Artifactory is available for immediate download from [here](#).
 You can browse the full JIRA release notes [here](#).

Major Features and Changes in release 1.3.0-beta-6

- Artifactory is 100% configurable via the newly designed user interface
- Support for running Artifactory with [MySQL](#) for greater scalability, performance and tools support.
- Support for running Artifactory under [Websphere](#).
- Install script for automatic set up under a dedicated [Tomcat](#) instance.
- Ability to remove deployed versions with all their sub-artifacts in one go.
- Dynamic log tailing from the UI.
- Global offline flag for disconnected environments.
- Out of the box support for running standalone Artifactory as a [Windows service](#) (in addition to the existing Unix service installer).
- Unified [artadmin](#) CLI tool for all administrative tasks.
- Greatly improved backup and restore speed.
- Many internal performance and concurrency improvements.
- The usual bug fixes.

Major Features and Changes in release 1.3.0-beta-5

- In place incremental backup.
- Performance improvements around security authentication and authorization.
- Fixed a UI bug with remote repositories cache access control setup.

Major Features and Changes in release 1.3.0-beta-4

- Improved LDAP Support - support for multiple DNs, for non-anonymous binding against MS Active Directory and for sub-tree searches.
- Fixed a critical issue with deployment using the lightweight HTTP wagon (RTFACT-629).
- UI and performance improvements and the usual bug fixes.

Major Features and Changes in release 1.3.0-beta-3

- Group support with an intuitive powerful UI for groups management.

- Improved permissions control for including/excluding specific repository paths (e.g. disallow certain users/groups from downloading sources). Simple management - no regexp gurus required.
- Support for delete permission and for preventing users from overriding existing artifacts.
- Revised repository tree browsing using tabs and effective permissions per item view.
- Improved import speed by committing binary imports asynchronously.
- Improved [#upgrade](#) procedure.
- Better transaction management using Spring Modules for JCR.
- Even faster in-place incremental backup.
- Low-level admin-only direct WebDAV access to the underlying jackrabbit repository (use: <http://localhost:8081/artifactory/jcr/default/> to connect).
- The usual bug fixes, stability and speed improvements.

The complete release notes from Jira are available [here](#):

[1.3.0-rc-1](#)
[1.3.0-beta-6](#)
[1.3.0-beta-5](#)
[1.3.0-beta-4](#)
[1.3.0-beta-3](#)

Artifactory 1.3.0-beta-6 is available for immediate download [here](#).

Enjoy 

The Artifactory Team

Important Notes Before You Install

Please read this carefully before installing or upgrading:

1. If you are running Artifactory under JDK 1.6 you **MUST** use **JDK 1.6u4** and above due to incompatibilities with older JAXB versions embedded in previous JDK 1.6 releases. JDK 1.5 users are not affected by this.
2. If you have used a previous version of Artifactory it is highly recommended to:
 - a. Use a fresh version of Artifactory from the distribution and not try to patch the previous installation.
 - b. Clear your browser's cache before using this version.
3. The default Artifactory user for the standalone installation has been changed to `artifactory` (instead of `jetty`). You might have to update the provided install script or your file system permissions accordingly.
4. Artifactory now includes a convenience method for specifying system properties. Instead of having to configure properties in the runtime configuration of the hosting container, you can now edit the `$ARTIFACTORY_HOME/etc/artifactory.properties` file and restart the container. As this affects the whole container VM it is recommended to use this feature for specifying Artifactory related properties only (such as repository ids substitution, etc.).
5. Please note that not all documentation has been updated yet to reflect the latest changes in this beta release.

Below is important information that will make it into the final Artifactory 1.3.0 documentation.

Installing

Please refer to the [Installing](#) section of the current user guide, which still applies.

Upgrading from 1.2.2-rc0 through 1.3.0-beta-6.1

To upgrade from older versions you first need to dump the data from your old Artifactory into a 1.3 compatible format and then import it to the new Artifactory.

We expect that with 1.3.0 and above upgrading can be done directly on an existing repository without the need to export it first.

Dumping the old version

Since Artifactory 1.3.0-beta-5, JFrog merged the `artdump` tool with the main CLI administration tool: `artadmin` - creating the CLI "dump" command.

The `artadmin` tool can be found under the `$ARTIFACTORY_HOME/bin` folder.

Important Information for Running the Dump

1. By default, cached repositories (e.g. `repo1`) are **not exported**. To export cached repositories pass the `--caches` parameter.
2. You should shut down the old Artifactory before executing the export.

3. It is recommended to work on a copy of the old \$ARTIFACTORY_HOME folder (even though the export should be a read only process).
4. If you do not specify a destination folder with --dest option, the dump command will create a tmpExport folder under the current execution directory, where it will export all the data from your old Artifactory.

Step by Step Instructions

Here are step by step instructions for running the full upgrade process:

1. Stop the old Artifactory.
2. Optional: Copy the \$ARTIFACTORY_HOME folder to a new location.
3. If you created a copy and wish the old Artifactory to keep serving requests while performing the export, you can start it now.
4. Execute the #artadmin dump command on the old \$ARTIFACTORY_HOME or on a copy of it, by running artadmin dump \$ARTIFACTORY_HOME (This will generate a tmpExport folder if you did not specify a destination folder with --dest option).
5. Perform a new clean server installation of Artifactory 1.3. It should not contain repository data or your customized version of the Artifactory configuration xml file.
6. Start Artifactory 1.3. Note: If in step 3 you chose to restart your old Artifactory and you installed the Artifactory 1.3 on the same machine, you will need to alter the listening port number inside \$ARTIFACTORY_HOME/etc/jetty.xml. It is also highly recommended to use a different port in order to perform the upgrade on a "silent" instance that is not being hit by user requests in parallel.
7. The import can be done in 2 ways:
 - Import with the artadmin command line (highly recommended):
 - a. Execute artadmin import tmpExport --username username --password password.
 - b. The artadmin output will display the progress of the import. NOTE: If the artadmin process is killed, the import will still run in Artifactory.
 - Import via the Web UI:
 - a. Logon with admin/password credentials, click the "Admin" tab and choose "System" from the "Import & Export" sub-menu on the left.
 - b. Enter the export directory into the "System zip file or directory" field in the "Import System" field-set and click the "Import" button.
 - c. The import will run and may take some time to complete (depending on the size of your database).
8. Once the import process completes successfully you can switch to using Artifactory 1.3. You can go back to use the old port number (if you changed it in step 6).

Important information about the import process

During the import binary artifacts are copied over into a working copy and are imported into Arifactory in by a background thread. This speeds up the import process dramatically and makes Artifactory ready for serving requests as soon as possible. The background import process will take some time to completes, depending on the size of your repository. During this time Artifactory might perform more slowly than usual, but it will still serve any artifact immediately. The log does provide visible information about the progress of this process.

Repeating the upgrade process

Should you wish to repeat the upgrade process from scratch, make sure to remove the \$ARTIFACTORY_HOME/data folder from your new Artifactory installation before doing so.

artadmin dump command line usage:

```

usage: dump [Artifactory home folder] ...
Artifactory will try to automatically determine the previous version from
$ARTIFACTORY_HOME/webapps/artifactory.war if present.
If the war file cannot be found at this location, please do one of the
following:
1) link or copy it at this location, or pass the version.
2) pass one of the following version as second parameter:
   1.2.2-rc0  1.2.2-rc1  1.2.2-rc2  1.2.2  1.2.5-rc0
   1.2.5-rc1  1.2.5-rc2  1.2.5-rc3  1.2.5-rc4  1.2.5-rc5
   1.2.5-rc6  1.2.5  1.2.5u1  1.3.0-beta-1  1.3.0-beta-2
   1.3.0-beta-3  1.3.0-beta-4  1.3.0-beta-5
Valid options:
  --dest [destination folder]: the destination folder for the new export
  files. Default value: tmpExport
  --version [version name]: the actual version of the old Artifactory if
  the Update Manager cannot find it
  --caches: include cached repositories in the export (by default caches
  are not exported). If repo option is passed this option will be ignored
  --security: only export the security file from DB, and set the norepo
  flag
  --repo [repo names separated by ':']: export only a specified list of
  repositories. Default value: all-non-cached
  --norepo: does not export the repositories, just convert config and
  security
  --convert: activate the Local and Virtual repository names conversion
  --noconvert: does not activate the Local and Virtual repository names
  conversion during a full export

```

artadmin import command line usage:

```

usage: import [import from path] ...
Valid options:
  --ssl: Activate https instead of http. Default is false
  --server [the server host or ip]: The remote Artifactory server IP or
  host name with optional port number. The default is localhost:8081
  --timeout [timeout in seconds]: Set the timeout of the HTTP connection.
  --url [root url to rest api]: The root URL of the Artifactory REST API.
  The default is http://[servername]/artifactory/api
  --username [username]: Optional username to use when connecting to the
  remote Artifactory
  --password [password]: The user's clear text password
  --noMetadata: Exclude metadata information when importing/exporting
  --symlinks: Use symbolic links to the original import path file (no file
  copying)
  --syncImport: Import directly into Artifactory without using the
  background import process
  --verbose: display maximum details
  --failFast: fail at first error
  --failIfEmpty: fail at empty directories

```

Options for the import process

- When used with the `--symlinks` option, the import process can perform much faster (usually by a factor of 3-5). Note that when using this option your imported path effectively becomes an integral part of Artifactory and you should not move/remove it until the background import process has completed. This is indicated by the log message: Working copy commit done (0 files).
- When using external database (like MySql) it is recommended to deactivate the working copy mechanism. To do this, use the `--syncImport`.

artadmin command line usage:

```
Usage: artadmin <command> [arg] [options]
Type 'artadmin help <command>' for help on a specific command.

Available commands:
help: The help message
info: Get system information
export: Export artifactory data to destination path
import: Import full system from import path
dump: Dump the database of an older version of Artifactory
compress: Compress the database tables (Derby only) in order to free up
disk space.
```

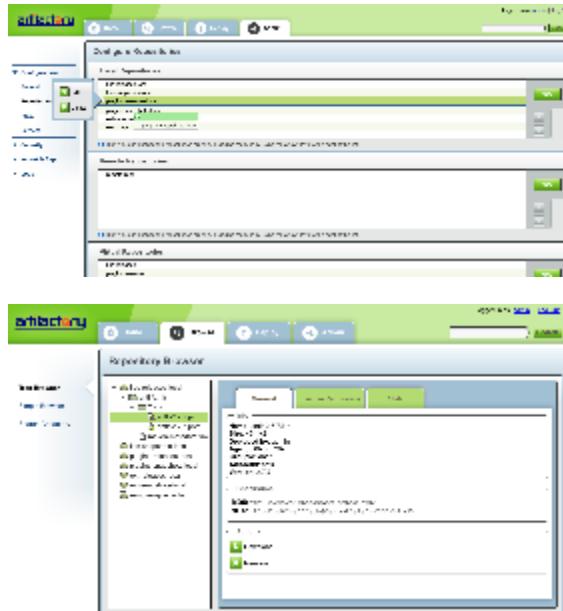
1.3.0-beta-6

Artifactory 1.3.0-beta-6

We are pleased to announce the availability of Artifactory 1.3.0-beta-6.

Major Features and Changes in this Release

- Artifactory is now fully configurable via the newly designed user interface!



- Support for running Artifactory with [MySQL](#) for greater scalability, performance and tools support.
- Support for running Artifactory under [Websphere](#).

- Install script for automatic set up under a dedicated [Tomcat](#) instance.
- Ability to remove deployed versions with all their sub-artifacts in one go.
- Dynamic log tailing from the UI.
- Global offline flag for disconnected environments.
- Out of the box support for running standalone Artifactory as a [Windows service](#) (in addition to the existing Unix service installer).
- Unified [artadmin](#) CLI tool for all administrative tasks.
- Greatly improved backup and restore speed.
- Many internal performance and concurrency improvements.
- The usual bug fixes.

Artifactory is available for immediate download from [here](#).

Major Features and Changes in release 1.3.0-beta-5

- In place incremental backup.
- Performance improvements around security authentication and authorization.
- Fixed a UI bug with remote repositories cache access control setup.

Major Features and Changes in release 1.3.0-beta-4

- Improved [LDAP Support](#) - support for multiple DNs, for non-anonymous binding against MS Active Directory and for sub-tree searches.
- Fixed a critical issue with deployment using the lightweight HTTP wagon (RTFACT-629).
- UI and performance improvements and the usual bug fixes.

Major Features and Changes in release 1.3.0-beta-3

- Group support with an intuitive powerful UI for groups management.
- Improved permissions control for including/excluding specific repository paths (e.g. disallow certain users/groups from downloading sources). Simple management - no regexp gurus required.
- Support for delete permission and for preventing users from overriding existing artifacts.
- Revised repository tree browsing using tabs and effective permissions per item view.
- Improved import speed by committing binary imports asynchronously.
- Improved [#upgrade](#) procedure.
- Better transaction management using Spring Modules for JCR.
- Even faster in-place incremental backup.
- Low-level admin-only direct WebDAV access to the underlying jackrabbit repository (use: <http://localhost:8081/artifactory/jcr/default/> to connect).
- The usual bug fixes, stability and speed improvements.

The complete release notes from Jira are available here:

[1.3.0-beta-6](#)
[1.3.0-beta-5](#)
[1.3.0-beta-4](#)
[1.3.0-beta-3](#)

Artifactory 1.3.0-beta-6 is available for immediate download [here](#).

Enjoy 😊

The Artifactory Team

Important Notes Before You Install

Please read this carefully before installing or upgrading:

1. If you are running Artifactory under JDK 1.6 you MUST use **JDK 1.6u4** and above due to incompatibilities with older JAXB versions embedded in previous JDK 1.6 releases. JDK 1.5 users are not affected by this.
2. If you have used a previous version of Artifactory it is highly recommended to:
 - a. Use a fresh version of Artifactory from the distribution and not try to patch the previous installation.
 - b. Clear your browser's cache before using this version.
3. The default Artifactory user for the standalone installation has been changed to **artifactory** (instead of **jetty**). You might have to update the provided install script or your file system permissions accordingly.
4. Artifactory now includes a convenience method for specifying system properties. Instead of having to configure properties in the runtime configuration of the hosting container, you can now edit the `$ARTIFACTORY_HOME/etc/artifactory.properties` file and restart the container. As this affects the whole container VM it is recommended to use this feature for specifying Artifactory related properties only (such as repository ids substitution, etc.).

5. Please note that not all documentation has been updated yet to reflect the latest changes in this beta release.

Below is important information that will make it into the final Artifactory 1.3.0 documentation.

Installing

Please refer to the [Installing](#) section of the current user guide, which still applies.

Upgrading from 1.2.2-rc0 through 1.3.0-beta-5

To upgrade from older versions you first need to dump the data from your old Artifactory into a 1.3 compatible format and then import it to the new Artifactory.

We expect that with 1.3.0 and above upgrading can be done directly on an existing repository without the need to export it first.

Dumping the old version

Since Artifactory 1.3.0-beta-5, JFrog merged the `artdump` tool with the main CLI administration tool: `artadmin` - creating the CLI "dump" command.

The `artadmin` tool can be found inside the `bin/artifactory-cli-1.3.x.zip` archive, downloadable from [here](#).

Important Information for Running the Dump

1. By default, cached repositories (e.g. `repo1`) are **not exported**. To export cached repositories pass the `--caches` parameter.
2. You should shut down the old Artifactory before executing the export.
3. It is recommended to work on a copy of the old `$ARTIFACTORY_HOME` folder (even though the export should be a read only process).
4. If you do not specify a destination folder with `--dest` option, the dump command will create a `tmpExport` folder under the current execution directory, where it will export all the data from your old Artifactory.

Step by Step Instructions

Here are step by step instructions for running the full upgrade process:

1. Stop the old Artifactory.
2. Optional: Copy the `$ARTIFACTORY_HOME` folder to a new location.
3. If you created a copy and wish the old Artifactory to keep serving requests while performing the export, you can start it now.
4. Execute the `#artadmin dump` command on the old `$ARTIFACTORY_HOME` or on a copy of it, by running `artadmin dump $ARTIFACTORY_HOME` (This will generate a `tmpExport` folder if you did not specify a destination folder with `--dest` option).
5. Perform a new clean server installation of Artifactory 1.3. It should not contain repository data or your customized version of the Artifactory configuration xml file.
6. Start Artifactory 1.3. Note: If in step 3 you chose to restart your old Artifactory and you installed the Artifactory 1.3 on the same machine, you will need to alter the listening port number inside `$ARTIFACTORY_HOME/etc/jetty.xml`. It is also highly recommended to use a different port in order to perform the upgrade on a "silent" instance that is not being hit by user requests in parallel.
7. The import can be done in 2 ways:
 - Import with the `artadmin` command line (highly recommended):
 - a. Execute `artadmin import tmpExport --username username --password password`.
 - b. The `artadmin` output will display the progress of the import. NOTE: If the `artadmin` process is killed, the import will still run in Artifactory.
 - Import via the Web UI:
 - a. Logon with admin/password credentials, click the "Admin" tab and choose "System" from the "Import & Export" sub-menu on the left.
 - b. Enter the export directory into the "System zip file or directory" field in the "Import System" field-set and click the "Import" button.
 - c. The import will run and may take some time to complete (depending on the size of your database).
8. Once the import process completes successfully you can switch to using Artifactory 1.3. You can go back to use the old port number (if you changed it in step 6).

Important information about the import process

During the import binary artifacts are copied over into a working copy and are imported into Artifactory in by a background thread. This speeds up the import process dramatically and makes Artifactory ready for serving requests as soon as possible. The background import process will take some time to complete, depending on the size of your repository. During this time Artifactory might perform more slowly than usual, but it will still serve any artifact immediately. The log does provide visible information about the progress of this process.

Repeating the upgrade process

Should you wish to repeat the upgrade process from scratch, make sure to remove the `$ARTIFACTORY_HOME/data` folder from your new Artifactory installation before doing so.

artadmin dump command line usage:

```
usage: dump [Artifactory home folder] ...
Artifactory will try to automatically determine the previous version from
$ARTIFACTORY_HOME/webapps/artifactory.war if present.
If the war file cannot be found at this location, please do one of the
following:
1) link or copy it at this location, or pass the version.
2) pass one of the following version as second parameter:
   1.2.2-rc0  1.2.2-rc1  1.2.2-rc2  1.2.2  1.2.5-rc0
   1.2.5-rc1  1.2.5-rc2  1.2.5-rc3  1.2.5-rc4  1.2.5-rc5
   1.2.5-rc6  1.2.5  1.2.5u1  1.3.0-beta-1  1.3.0-beta-2
   1.3.0-beta-3  1.3.0-beta-4  1.3.0-beta-5
Valid options:
  --dest [destination folder]: the destination folder for the new export
  files. Default value: tmpExport
  --version [version name]: the actual version of the old Artifactory if
  the Update Manager cannot find it
  --caches: include cached repositories in the export (by default caches
  are not exported). If repo option is passed this option will be ignored
  --security: only export the security file from DB, and set the norepo
  flag
  --repo [repo names separated by ':']: export only a specified list of
  repositories. Default value: all-non-cached
  --norepo: does not export the repositories, just convert config and
  security
  --convert: activate the Local and Virtual repository names conversion
  --noconvert: does not activate the Local and Virtual repository names
  conversion during a full export
```

artadmin import command line usage:

```

usage: import [import from path] ...
Valid options:
  --ssl: Activate https instead of http. Default is false
  --server [the server host or ip]: The remote Artifactory server IP or
host name with optional port number. The default is localhost:8081
  --timeout [timeout in seconds]: Set the timeout of the HTTP connection.
  --url [root url to rest api]: The root URL of the Artifactory REST API.
The default is http://[servername]/artifactory/api
  --username [username]: Optional username to use when connecting to the
remote Artifactory
  --password [password]: The user's clear text password
  --noMetadata: Exclude metadata information when importing/exporting
  --symlinks: Use symbolic links to the original import path file (no file
copying)
  --syncImport: Import directly into Artifactory without using the
background import process
  --verbose: display maximum details
  --failFast: fail at first error
  --failIfEmpty: fail at empty directories

```

Options for the import process

- When used with the --symlinks option, the import process can perform much faster (usually by a factor of 3-5). Note that when using this option your imported path effectively becomes an integral part of Artifactory and you should not move/remove it until the background import process has completed. This is indicated by the log message: Working copy commit done (0 files).
- When using external database (like MySql) it is recommended to deactivate the working copy mechanism. To do this, use the --syncImport.

artadmin command line usage:

```

Usage: artadmin <command> [arg] [options]
Type 'artadmin help <command>' for help on a specific command.

Available commands:
help: The help message
info: Get system information
export: Export artifactory data to destination path
import: Import full system from import path
dump: Dump the database of an older version of Artifactory
compress: Compress the database tables in order to free up disk space.

```

1.3.0-beta-5

Artifactory 1.3.0-beta-5

We are pleased to announce the availability of Artifactory 1.3.0-beta-5.

Major Features and Changes in this Release

- In place incremental backup.
- Performance improvements around security authentication and authorization.

- Fixed a UI bug with remote repositories cache access control setup.

Major Features and Changes in release 1.3.0-beta-4

- Improved #LDAP Support - support for multiple DNs, for non-anonymous binding against MS Active Directory and for sub-tree searches.
- Fixed a critical issue with deployment using the lightweight HTTP wagon (RTFACT-629).
- UI and performance improvements and the usual bug fixes.

Major Features and Changes in release 1.3.0-beta-3

- Group support with an intuitive powerful UI for groups management (see: ).
- Improved permissions control for including/excluding specific repository paths (e.g. disallow certain users/groups from downloading sources). Simple management - no regexp gurus required (see: ).
- Support for delete permission and for preventing users from overriding existing artifacts.
- Revised repository tree browsing using tabs and effective permissions per item view (see: ).
- Improved import speed by committing binary imports asynchronously.
- Improved #upgrade procedure.
- Better transaction management using Spring Modules for JCR.
- Even faster in-place incremental backup.
- Low-level admin-only direct webdav access to the underlying jackrabbit repository (use: <http://localhost:8081/artifactory/jcr/default/> to connect).
- The usual bug fixes, stability and speed improvements.

The complete release notes from Jira are available here:

[1.3.0-beta-5](#)
[1.3.0-beta-4](#)
[1.3.0-beta-3](#)

Artifactory 1.3.0-beta-5 is available for immediate download [here](#).

Enjoy 

The Artifactory Team

Important Notes Before You Install

Please read carefully before installing or upgrading:

1. If you are running Artifactory under JDK 1.6 you MUST use **JDK 1.6u4** and above due to incompatibilities with older JAXB versions embedded in previous JDK 1.6 releases. JDK 1.5 users are not affected by this.
2. If you have used a previous version of Artifactory it is recommended to clear your browser's cache before using this version.
3. To use the new incremental backup feature, specify `<retentionPeriodHours>0</retentionPeriodHours>` under the backup configuration. This will cause backups to be written into a permanent "current" directory in a format that can be used by any incremental file-system based backup utility (such as rsync).
4. The M2Eclipse indexing service runs by default on all repositories every round hour. The run interval can be controlled via the `artifactory.config.xml` file, and repositories can be excluded from being indexed (similar to the configuration of the backup service).
5. Artifactory now includes a convenience method for specifying system properties. Instead of having to configure properties in the runtime configuration of the hosting container, you can now edit the `$ARTIFACTORY_HOME/etc/artifactory.properties` file and restart the container. As this affects the whole container VM it is recommended to use this feature for specifying Artifactory related properties only (such as repository ids substitution, etc.).
6. Please note that not all documentation has been updated yet to reflect the latest changes in this beta release.

Below is important information that will make its way into the final Artifactory 1.3.0 documentation.

Installing

Please refer to the [Installing](#) section of the current user guide, which still applies.

LDAP Support

General

Besides its own users database, Artifactory also supports authenticating users against an LDAP server.

If LDAP is configured in Artifactory configuration file, Artifactory will first attempt to authenticate the user against the LDAP server. If the LDAP authentication fails, Artifactory will try to authenticate via the internal database.

For every LDAP authenticated user Artifactory creates new user in the internal database if that user doesn't exist.

Configuration

LDAP Configuration is placed under the security tag in artifactory.config.xml. For example:

```
<security>
    <anonAccessEnabled>false</anonAccessEnabled>
    <ldapSettings>
        <ldapUrl>ldap://127.0.0.1:389/dc=jfrog,dc=org</ldapUrl>
        <authenticationPatterns>
            <!-- Use the 'direct' method -->
            <authenticationPattern>
                <userDnPattern>uid={0},ou=People</userDnPattern>
            </authenticationPattern>
            <!-- OR, using the 'search' method -->
            <authenticationPattern>
                <searchFilter>sAMAccountName={0}</searchFilter>
                <searchSubTree>true</searchSubTree>
            </authenticationPattern>
        </authenticationPatterns>
        <managerDn>cn=ReadOnlyUser,ou=People,dc=jfrog,dc=org</managerDn>
        <managerPassword>password</managerPassword>
    </ldapSettings>
</security>
```

Where:

- **ldapUrl** - Location of the ldap server in the form of: ldap://myserver:myport/dc=sampledomain,dc=com. It should include the base DN for searching and/or authenticating users.
 - **authenticationPatterns** - A list of user DN patterns or search criteria used to find an authenticated user. Each authentication pattern element will be tried until a user is found and authenticated. If no user is found or authentication failed, a BadCredentials exception will be thrown.
 - **authenticationPattern** - An authentication pattern can include either a userDnPattern for "direct" authentication or searchFilter for user "search" (after binding with a manager DN) and then authentication of the found user. The **managerDn** and **managerPassword** are not used for "direct" DN authentication.
 - **userDnPattern** - A DN pattern that can be used to directly login users to the LDAP database. This pattern is used for creating a DN string for "direct" user authentication, where the pattern is relative to the base DN in the ldapUrl. The pattern argument {0} will be replaced with the username in runtime. This will work only if anonymous binding is allowed and a direct user DN can be used (which is not the default case for Active Directory).
- Example:
- ```
uid={0},ou=People
```
- **searchFilter** - The filter expression used for searching a user. This is an LDAP search filter (as defined in 'RFC 2254' [faqs.org ietf](#)) with optional arguments. See the documentation for the `search` method of `javax.naming.directory.DirContext` for more information. The search method will be called with **searchBase** as name, a single `filterArg` containing the username, a `filterExpr` with this **searchFilter** value and `searchCons` affected by the value of **searchSubTree**. Authentication to LDAP will be done from the DN found.
  - Possible examples are:  
`sAMAccountName={0}`, for use with Active Directory, or  
`uid={0}`, for use with other LDAP servers.
  - **searchBase** - Context name to search in, relative to the base DN in the ldapUrl. This parameter is optional.
  - **searchSubTree** - Flag to enable deep search through the sub tree of the ldapUrl + searchBase. True by default.
  - **managerDn** - Used only with "search" authentication method. It is the DN of the user who will bind to the LDAP server to perform the search.
  - **managerPassword** - Used only with "search" authentication method. It is the password of the user who will bind to the LDAP server to perform the search.

## Upgrading from 1.3.0-beta-3 through 1.3.0-beta-4 to 1.3.0-beta-5

Shutdown Artifactory and replace the war. If you are on Tomcat make sure to remove the webapps/artifactory expanded folder as well.

## Upgrading from 1.2.2-rc0 through 1.3.0-beta-2 to 1.3.0-beta-5

To upgrade from older versions you first need to dump the data from your old Artifactory into a 1.3 compatible format and then import it to the new Artifactory.

We expect that with 1.3.0 and above upgrading can be done directly on an existing repository without the need to export it first.

### Dumping the old version

Since Artifactory 1.3.0-beta-1, JFrog provides a command line tool: `artdump` for dumping (previously: `artfactoryExport`). The `artdump` tool can be found inside the `bin artifactory-update-1.3.x.zip` archive, downloadable from [here](#).

### Important Information for Running the Dump

1. By default cached repositories (e.g. `repo1`) are **not exported**. To export cached repositories pass the `--caches` parameter.
2. You should shut down the old Artifactory before executing the export.
3. It is recommended to work on a copy of the old `$ARTIFACTORY_HOME` folder (that, even though the export should be a read only process).
4. Running the update creates a `tmpExport` folder (if you did not specify a destination folder with `--dest` option) under the current execution directory, where it exports all data from your old Artifactory. This `tmpExport` folder is next used to do a full system import to the new Artifactory.

## Step by Step Instructions

Here are step by step instructions for running the full upgrade process:

1. Stop the old Artifactory.
2. Optional: Copy the `$ARTIFACTORY_HOME` folder to a new location.
3. If you created a copy and wish the old Artifactory to keep serving requests while performing the export, you can start it now.
4. Execute the `#artdump` on the old `$ARTIFACTORY_HOME` or on a copy of it, by running `artdump --home $ARTIFACTORY_HOME`. This will generate a `tmpExport` folder if you did not specify a destination folder with `--dest` option.
5. Perform a new clean server installation of Artifactory 1.3. It should not contain repositories data or your specific Artifactory configuration xml file.
6. Start Artifactory 1.3. Note: If in step 3 you chose to restart your old Artifactory and you installed the Artifactory 1.3 on the same machine, you will need to alter the listening port number inside `$ARTIFACTORY_HOME/etc/jetty.xml`. It is also highly recommended to use a different port in order to perform the upgrade on a "silent" instance that is not being hit by user requests in parallel.
7. The import can be done in 2 ways:
  - Import with the `artadmin` command line (highly recommended):
    - a. Execute `#artadmin --username username --password password --import tmpExport`. Please note that the `artadmin` tool needs to be run with JDK 1.6.
    - b. The `artadmin` output will display the progress of the import. NOTE: If the `artadmin` process is killed the import will still run in Artifactory.
  - Import via the Web UI:
    - a. Logon with admin/password and go to "Export & Import" page, then into the "Full System" panel.
    - b. Enter the export directory into the "System zip file or directory" field and click the "Import" button.
    - c. The import will run and may take some time to complete, depending on the size of your database.
8. Once the import process completes successfully you can switch to using Artifactory 1.3. You can switch back the port number if you did so in step 6.

### Important information about the import process

During the import binary artifacts are copied over into a working copy and are imported into Artifactory in the background thread. This speeds up the import process dramatically and makes Artifactory ready for serving requests as soon as possible. The background import process will take some time to complete, depending on the size of your repository. During this time Artifactory might perform more slowly than usual, but it will still serve any artifact immediately. The log does provide visible information about the progress of this process.

### Repeating the upgrade process

If you wish to repeat the upgrade process from scratch, make sure to remove the `$ARTIFACTORY_HOME/data` folder from your new Artifactory installation before doing so.

**artdump command line usage:**

```

artdump
--help : displays this usage message
--home [old Artifactory home] : mandatory - the home folder of the old
Artifactory
--dest [destination folder] : the destination folder for the new export
files. Default value: tmpExport
--version [version name] : the actual version of the old Artifactory if
the Update Manager cannot find it
--repo [repo names separated by ':'] : export only a specified list of
repositories. Default value: all-non-cached
--norepo : does not export the repositories, just convert config and
security
--convert : activate the Local and Virtual repository names conversion
--noconvert : does not activate the Local and Virtual repository names
conversion during a full export
--security : only export the security file from DB, and set the norepo
flag
--caches : include cached repositories in the export (by default caches
are not exported). If repo option is passed this option will be ignored
The parameter --home is mandatory.

The Artifactory version will be extracted from
${artifactory.home}/webapps/artifactory.war if present
If the war file is not located there, please do one of the following:
1) Link or copy it into this location, or -
2) Pass one of the following version identifiers as second parameter:
 1.2.2-rc0 1.2.2-rc1 1.2.2-rc2 1.2.2 1.2.5-rc0
 1.2.5-rc1 1.2.5-rc2 1.2.5-rc3 1.2.5-rc4 1.2.5-rc5
 1.2.5-rc6 1.2.5 1.2.5u1 1.3.0-beta-1 1.3.0-beta-2

```

- The --home parameter is the old \$ARTIFACTORY\_HOME folder (the old Artifactory should not be running). If that folder also contains a webapps subfolder with the old Artifactory war file, the update manager will automatically determine the old version from it. If the update manager cannot determine a valid version an error message will appear with a list of valid versions and you'd need to manually specify the version of your old Artifactory as a second parameter to artdump.

*artadmin command line usage:*

```

artadmin
--info : Display general system information about Artifactory
--import [import from path] : Activate full system import from a
designated path
--export [export to path] : Activate full system export to path
--server [the server host or ip] : The remote Artifactory server IP or
host name with optional port number. The default is localhost:8081
--ssl : Activate https instead of http. Default is false
--timeout [timeout in seconds] : Set the timeout of the HTTP connection.
--url [root url to rest api] : The root URL of the Artifactory REST API.
The default is http://[servername]/artifactory/api
--username [username] : Optional username to use when connecting to the
remote Artifactory
--password [password] : The user's clear text password
--noMetadata : Exclude metadata information when importing/exporting
--symlinks : Use symbolic links to the original import path file (no file
copying)
--syncImport : Import directly into Artifactory without using the
background import process
--bypassFiltering : Avoid using existing repository filtering rules during
the export process
--createArchive : Zip the resulting folder after the export (slow)

You need to specify one of the following parameters: --info, --import or
--export

```

- When used with the --symlinks option the import process can perform much faster (usually by a factor of 3-5). Note that when using this option your imported path effectively becomes an integral part of Artifactory and you should not move/remove it until the background import process has completed. This is indicated by the log message: Working copy commit done (0 files).

## 1.3.0-beta-4

### Artifactory 1.3.0-beta-4

We are pleased to announce the availability of Artifactory 1.3.0-beta-4.

#### ***Major Features and Changes in this Release***

- Improved #LDAP Support - support for multiple DNs, for non-anonymous binding against MS Active Directory and for sub-tree searches.
- Fixed a critical issue with deployment using the lightweight HTTP wagon (RTFACT-629).
- UI and performance improvements and the usual bug fixes.

#### ***Major Features and Changes in release 1.3.0-beta-3***

- Group support with an intuitive powerful UI for groups management (see: ).
- Improved permissions control for including/excluding specific repository paths (e.g. disallow certain users/groups from downloading sources). Simple management - no regexp gurus required (see: ).
- Support for delete permission and for preventing users from overriding existing artifacts.
- Revised repository tree browsing using tabs and effective permissions per item view (see: ).
- Improved import speed by committing binary imports asynchronously.
- Improved #upgrade procedure.

- Better transaction management using Spring Modules for JCR.
- Even faster in-place incremental backup.
- Low-level admin-only direct webdav access to the underlying jackrabbit repository (use: <http://localhost:8081/artifactory/jcr/default/> to connect).
- The usual bug fixes, stability and speed improvements.

The complete release notes are available here: <http://issues.jfrog.org/jira/browse/RTFACT/fixforversion/10270>

Artifactory 1.3.0-beta-4 is available for immediate download [here](#).

Enjoy 😊

The Artifactory Team

---

### ***Important Notes Before You Install***

**Please read carefully before installing or upgrading:**

1. If you are running Artifactory under JDK 1.6 you **MUST** use **JDK 1.6u4** and above due to incompatibilities with older JAXB versions embedded in previous JDK 1.6 releases. JDK 1.5 users are not affected by this.
2. If you have used a previous version of Artifactory it is recommended to clear your browser's cache before using this version.
3. To use the new incremental backup feature, specify `<retentionPeriodHours>0</retentionPeriodHours>` under the backup configuration. This will cause backups to be written into a permanent "current" directory in a format that can be used by any incremental file-system based backup utility (such as rsync).
4. The M2Eclipse indexing service runs by default on all repositories every round hour. The run interval can be controlled via the `artifactory.config.xml` file, and repositories can be excluded from being indexed (similar to the configuration of the backup service).
5. Artifactory now includes a convenience method for specifying system properties. Instead of having to configure properties in the runtime configuration of the hosting container, you can now edit the `$ARTIFACTORY_HOME/etc/artifactory.properties` file and restart the container. As this affects the whole container VM it is recommended to use this feature for specifying Artifactory related properties only (such as repository ids substitution, etc.).
6. Please note that not all documentation has been updated yet to reflect the latest changes in this beta release.

Below is important information that will make its way into the final Artifactory 1.3.0 documentation.

---

### **Installing**

Please refer to the [Installing](#) section of the current user guide, which still applies.

---

### **LDAP Support**

#### ***General***

Besides its own users database, Artifactory also supports authenticating users against an LDAP server.

If LDAP is configured in Artifactory configuration file, Artifactory will first attempt to authenticate the user against the LDAP server. If the LDAP authentication fails, Artifactory will try to authenticate via the internal database.

For every LDAP authenticated user Artifactory creates new user in the internal database if that user doesn't exist.

#### ***Configuration***

LDAP Configuration is placed under the security tag in `artifactory.config.xml`. For example:

```

<security>
 <anonAccessEnabled>false</anonAccessEnabled>
 <ldapSettings>
 <ldapUrl>ldap://127.0.0.1:389/dc=jfrog,dc=org</ldapUrl>
 <authenticationPatterns>
 <!-- Use the 'direct' method -->
 <authenticationPattern>
 <userDnPattern>uid={0},ou=People</userDnPattern>
 </authenticationPattern>
 <!-- OR, using the 'search' method -->
 <authenticationPattern>
 <searchFilter>sAMAccountName={0}</searchFilter>
 <searchSubTree>true</searchSubTree>
 </authenticationPattern>
 </authenticationPatterns>
 <managerDn>cn=ReadOnlyUser,ou=People,dc=jfrog,dc=org</managerDn>
 <managerPassword>password</managerPassword>
 </ldapSettings>
</security>

```

Where:

- **ldapUrl** - Location of the ldap server in the form of: ldap://myserver:myport/dc=sampleddomain,dc=com. It should include the base DN for searching and/or authenticating users.
- **authenticationPatterns** - A list of user DN patterns or search criteria used to find an authenticated user. Each authentication pattern element will be tried until a user is found and authenticated. If no user is found or authentication failed, a BadCredentials exception will be thrown.
- **authenticationPattern** - An authentication pattern can include either a userDnPattern for "direct" authentication or searchFilter for user "search" (after binding with a manager DN) and then authentication of the found user. The **managerDn** and **managerPassword** are not used for "direct" DN authentication.
- **userDnPattern** - A DN pattern that can be used to directly login users to the LDAP database. This pattern is used for creating a DN string for "direct" user authentication, where the pattern is relative to the base DN in the ldapUrl. The pattern argument {0} will be replaced with the username in runtime. This will work only if anonymous binding is allowed and a direct user DN can be used (which is not the default case for Active Directory).

Example:

uid={0},ou=People

- **searchFilter** - The filter expression used for searching a user. This is an LDAP search filter (as defined in 'RFC 2254' [faqs.org ietf](#)) with optional arguments. See the documentation for the `search` method of `javax.naming.directory.DirContext` for more information. The search method will be called with **searchBase** as name, a single `filterArg` containing the username, a `filterExpr` with this **searchFilter** value and `search_cons` affected by the value of **searchSubTree**. Authentication to LDAP will be done from the DN found.

Possible examples are:

sAMAccountName={0}, for use with Active Directory, or  
uid={0}, for use with other LDAP servers.

- **searchBase** - Context name to search in, relative to the base DN in the ldapUrl. This parameter is optional.
- **searchSubTree** - Flag to enable deep search through the sub tree of the ldapUrl + searchBase. True by default.
- **managerDn** - Used only with "search" authentication method. It is the DN of the user who will bind to the LDAP server to perform the search.
- **managerPassword** - Used only with "search" authentication method. It is the password of the user who will bind to the LDAP server to perform the search.

## Upgrading from 1.3.0-beta-3 to 1.3.0-beta-4

Shutdown Artifactory and replace the war. If you are on Tomcat make sure to remove the webapps/artifactory expanded folder as well.

## Upgrading from 1.2.2-rc0 through 1.3.0-beta-2 to 1.3.0-beta-4

To upgrade from older versions you first need to dump the data from your old Artifactory into a 1.3 compatible format and then import it to the new Artifactory.

We expect that with 1.3.0 and above upgrading can be done directly on an existing repository without the need to export it first.

## Dumping the old version

Since Artifactory 1.3.0-beta-1, JFrog provides a command line tool: `artdump` for dumping (previously: `artfactoryExport`). The `artdump` tool can be found inside the `bin artifactory-update-1.3.x.zip` archive, downloadable from [here](#).

### Important Information for Running the Dump

1. By default cached repositories (e.g. `repo1`) are **not exported**. To export cached repositories pass the `--caches` parameter.
2. You should shut down the old Artifactory before executing the export.
3. It is recommended to work on a copy of the old `$ARTIFACTORY_HOME` folder (that, even though the export should be a read only process).
4. Running the update creates a `tmpExport` folder (if you did not specify a destination folder with `--dest` option) under the current execution directory, where it exports all data from your old Artifactory. This `tmpExport` folder is next used to do a full system import to the new Artifactory.

## Step by Step Instructions

Here are step by step instructions for running the full upgrade process:

1. Stop the old Artifactory.
2. Optional: Copy the `$ARTIFACTORY_HOME` folder to a new location.
3. If you created a copy and wish the old Artifactory to keep serving requests while performing the export, you can start it now.
4. Execute the `#artdump` on the old `$ARTIFACTORY_HOME` or on a copy of it, by running `artdump --home $ARTIFACTORY_HOME`. This will generate a `tmpExport` folder if you did not specify a destination folder with `--dest` option.
5. Perform a new clean server installation of Artifactory 1.3. It should not contain repositories data or your specific Artifactory configuration xml file.
6. Start Artifactory 1.3. Note: If in step 3 you chose to restart your old Artifactory and you installed the Artifactory 1.3 on the same machine, you will need to alter the listening port number inside `$ARTIFACTORY_HOME/etc/jetty.xml`. It is also highly recommended to use a different port in order to perform the upgrade on a "silent" instance that is not being hit by user requests in parallel.
7. The import can be done in 2 ways:
  - Import with the `artadmin` command line (highly recommended):
    - a. Execute `#artadmin --username username --password password --import tmpExport`. Please note that the `artadmin` tool needs to be run with JDK 1.6.
    - b. The `artadmin` output will display the progress of the import. NOTE: If the `artadmin` process is killed the import will still run in Artifactory.
  - Import via the Web UI:
    - a. Logon with `admin/password` and go to "Export & Import" page, then into the "Full System" panel.
    - b. Enter the export directory into the "System zip file or directory" field and click the "Import" button.
    - c. The import will run and may take some time to complete, depending on the size of your database.
8. Once the import process completes successfully you can switch to using Artifactory 1.3. You can switch back the port number if you did so in step 6.

### Important information about the import process

During the import binary artifacts are copied over into a working copy and are imported into Artifactory in by a background thread. This speeds up the import process dramatically and makes Artifactory ready for serving requests as soon as possible. The background import process will take some time to complete, depending on the size of your repository. During this time Artifactory might perform more slowly than usual, but it will still serve any artifact immediately. The log does provide visible information about the progress of this process.

### Repeating the upgrade process

Should you wish to repeat the upgrade process from scratch, make sure to remove the `$ARTIFACTORY_HOME/data` folder from your new Artifactory installation before doing so.

**`artdump` command line usage:**

```

artdump
--help : displays this usage message
--home [old Artifactory home] : mandatory - the home folder of the old
Artifactory
--dest [destination folder] : the destination folder for the new export
files. Default value: tmpExport
--version [version name] : the actual version of the old Artifactory if
the Update Manager cannot find it
--repo [repo names separated by ':'] : export only a specified list of
repositories. Default value: all-non-cached
--norepo : does not export the repositories, just convert config and
security
--convert : activate the Local and Virtual repository names conversion
--noconvert : does not activate the Local and Virtual repository names
conversion during a full export
--security : only export the security file from DB, and set the norepo
flag
--caches : include cached repositories in the export (by default caches
are not exported). If repo option is passed this option will be ignored
The parameter --home is mandatory.

The Artifactory version will be extracted from
${artifactory.home}/webapps/artifactory.war if present
If the war file is not located there, please do one of the following:
1) Link or copy it into this location, or -
2) Pass one of the following version identifiers as second parameter:
 1.2.2-rc0 1.2.2-rc1 1.2.2-rc2 1.2.2 1.2.5-rc0
 1.2.5-rc1 1.2.5-rc2 1.2.5-rc3 1.2.5-rc4 1.2.5-rc5
 1.2.5-rc6 1.2.5 1.2.5u1 1.3.0-beta-1 1.3.0-beta-2

```

- The --home parameter is the old \$ARTIFACTORY\_HOME folder (the old Artifactory should not be running). If that folder also contains a webapps subfolder with the old Artifactory war file, the update manager will automatically determine the old version from it. If the update manager cannot determine a valid version an error message will appear with a list of valid versions and you'd need to manually specify the version of your old Artifactory as a second parameter to artdump.

*artadmin command line usage:*

```

artadmin
--info : Display general system information about Artifactory
--import [import from path] : Activate full system import from a
designated path
--export [export to path] : Activate full system export to path
--server [the server host or ip] : The remote Artifactory server IP or
host name with optional port number. The default is localhost:8081
--ssl : Activate https instead of http. Default is false
--timeout [timeout in seconds] : Set the timeout of the HTTP connection.
--url [root url to rest api] : The root URL of the Artifactory REST API.
The default is http://[servername]/artifactory/api
--username [username] : Optional username to use when connecting to the
remote Artifactory
--password [password] : The user's clear text password
--noMetadata : Exclude metadata information when importing/exporting
--symlinks : Use symbolic links to the original import path file (no file
copying)
--syncImport : Import directly into Artifactory without using the
background import process
--bypassFiltering : Avoid using existing repository filtering rules during
the export process
--createArchive : Zip the resulting folder after the export (slow)

You need to specify one of the following parameters: --info, --import or
--export

```

- When used with the --symlinks option the import process can perform much faster (usually by a factor of 3-5). Note that when using this option your imported path effectively becomes an integral part of Artifactory and you should not move/remove it until the background import process has completed. This is indicated by the log message: Working copy commit done (0 files).

## 1.3.0-beta-3

### Artifactory 1.3.0-beta-3

We are pleased to announce the availability of Artifactory 1.3.0-beta-3.

#### **Major Features and Changes in this Release**

- #LDAP Support!
- Group support with an intuitive powerful UI for groups management (see: ).
- Improved permissions control for including/excluding specific repository paths (e.g. disallow certain users/groups from downloading sources). Simple management - no regexp gurus required (see: ).
- Support for delete permission and for preventing users from overriding existing artifacts.
- Revised repository tree browsing using tabs and effective permissions per item view (see: ).
- Improved import speed by committing binary imports asynchronously.
- Improved #upgrade procedure.
- Better transaction management using Spring Modules for JCR.
- Even faster in-place incremental backup.
- Low-level admin-only direct webdav access to the underlying jackrabbit repository (use: <http://localhost:8081/artifactory/jcr/default/> to connect).
- The usual bug fixes, stability and speed improvements.

The complete release notes are available here: <http://issues.jfrog.org/jira/browse/RTFACT/fixforversion/10250>

Artifactory 1.3.0-beta-3 is available for immediate download [here](#).

Enjoy 😊

The Artifactory Team

### ***Important Notes Before You Install***

#### **Please read carefully before installing or upgrading:**

1. If you are running Artifactory under JDK 1.6 you MUST use **JDK 1.6u4** and above due to incompatibilities with older JAXB versions embedded in previous JDK 1.6 releases. JDK 1.5 users are not affected by this.
2. If you have used a previous version of Artifactory it is recommended to clear your browser's cache before using this version.
3. To use the new incremental backup feature, specify `<retentionPeriodHours>0</retentionPeriodHours>` under the backup configuration. This will cause backups to be written into a permanent "current" directory in a format that can be used by any incremental file-system based backup utility (such as rsync).
4. The M2Eclipse indexing service runs by default on all repositories every round hour. The run interval can be controlled via the `artifactory.config.xml` file, and repositories can be excluded from being indexed (similar to the configuration of the backup service).
5. Artifactory now includes a convenience method for specifying system properties. Instead of having to configure properties in the runtime configuration of the hosting container, you can now edit the `$ARTIFACTORY_HOME/etc/artifactory.properties` file and restart the container. As this affects the whole container VM it is recommended to use this feature for specifying Artifactory related properties only (such as repository ids substitution, etc.).
6. Please note that not all documentation has been updated yet to reflect the latest changes in this beta release.

1.3.0-beta-3 has a bug that prevents deployment using the simple HTTP wagon when anonymous repository access is on. This bug is fixed in 1.3.0-beta-4 (<http://issues.jfrog.org/jira/browse/RTFACT-629>), and with beta-3 the workaround is to use the dav wagon for deployment, which is a simple change of the URL prefix (dav:`http`):

```
<distributionManagement>
 <repository>
 <id>repo-id</id>

 <url>dav:http://localhost:8081/artifactory/libs-releases-local</url>
 </repository>
</distributionManagement>
```

Below is important information that will make its way into the final Artifactory 1.3.0 documentation.

### **Installing**

Please refer to the [Installing](#) section of the current user guide, which still applies.

### **LDAP Support**

#### **General**

Besides its own users database, Artifactory also supports authenticating users against an LDAP server.

If LDAP is configured in Artifactory configuration file, Artifactory will first attempt to authenticate the user against the LDAP server. If the LDAP authentication fails, Artifactory will try to authenticate via the internal database.

For every LDAP authenticated user Artifactory creates new user in the internal database if that user doesn't exist.

#### **Configuration**

LDAP Configuration is placed under the security tag in `artifactory.config.xml`. For example:

```

<security>
 <ldapSettings>
 <ldapUrl>ldap://127.0.0.1:389/dc=jfrog,dc=org</ldapUrl>
 <userDnPattern>uid={0}, ou=People</userDnPattern>
 <managerDn></managerDn>
 <managerPassword></managerPassword>
 </ldapSettings>
</security>

```

Where:

- **ldapUrl** - Url or the ldap server including the base DN to start searching for users.
- **userDnPattern** - The pattern, relative to the base DN in the ldapUrl, which will be used to search for the user. The pattern argument {0} will contain the username.
- **managerDn** - User name to logon the ldap server if it requires authentication to search for users
- **managerPassword** - Password to logon the ldap server if it requires authentication to search for users

## Upgrading (from 1.2.2-rc0 through 1.3.0-beta-2) to 1.3.0-beta-3

To upgrade from older versions you first need to dump the data from your old Artifactory into a 1.3 compatible format and then import it to the new Artifactory.

We expect that with 1.3.0 and above upgrading can be done directly on an existing repository without the need to export it first.

### Dumping the old version

Since Artifactory 1.3.0-beta-1, JFrog provides a command line tool: `artdump` for dumping (previously: `artfactoryExport`). The `artdump` tool can be found inside the `bin artifactory-update-1.3.x.zip` archive, downloadable from [here](#).

### Important Information for Running the Dump

1. By default cached repositories (e.g. `repo1`) are **not exported**. To export cached repositories pass the `--caches` parameter.
2. You should shut down the old Artifactory before executing the export.
3. It is recommended to work on a copy of the old `$ARTIFACTORY_HOME` folder (that, even though the export should be a read only process).
4. Running the update creates a `tmpExport` folder (if you did not specify a destination folder with `--dest` option) under the current execution directory, where it exports all data from your old Artifactory. This `tmpExport` folder is next used to do a full system import to the new Artifactory.

### Step by Step Instructions

Here are step by step instructions for running the full upgrade process:

1. Stop the old Artifactory.
2. Optional: Copy the `$ARTIFACTORY_HOME` folder to a new location.
3. If you created a copy and wish the old Artifactory to keep serving requests while performing the export, you can start it now.
4. Execute the `#artdump` on the old `$ARTIFACTORY_HOME` or on a copy of it, by running `artdump --home $ARTIFACTORY_HOME`. This will generate a `tmpExport` folder if you did not specify a destination folder with `--dest` option.
5. Perform a new clean server installation of Artifactory 1.3. It should not contain repositories data or your specific Artifactory configuration xml file.
6. Start Artifactory 1.3. Note: If in step 3 you chose to restart your old Artifactory and you installed the Artifactory 1.3 on the same machine, you will need to alter the listening port number inside `$ARTIFACTORY_HOME/etc/jetty.xml`. It is also highly recommended to use a different port in order to perform the upgrade on a "silent" instance that is not being hit by user requests in parallel.
7. The import can be done in 2 ways:
  - Import with the `artadmin` command line (highly recommended):
    - a. Execute `#artadmin --username username --password password --import tmpExport`. Please note that the `artadmin` tool needs to be run with JDK 1.6.
    - b. The `artadmin` output will display the progress of the import. NOTE: If the `artadmin` process is killed the import will still run in Artifactory.
  - Import via the Web UI:
    - a. Logon with admin/password and go to "Export & Import" page, then into the "Full System" panel.
    - b. Enter the export directory into the "System zip file or directory" field and click the "Import" button.
    - c. The import will run and may take some time to complete, depending on the size of your database.
8. Once the import process completes successfully you can switch to using Artifactory 1.3. You can switch back the port number if you did so in step 6.

#### **Important information about the import process**

During the import binary artifacts are copied over into a working copy and are imported into Artifactory in by a background thread. This speeds up the import process dramatically and makes Artifactory ready for serving requests as soon as possible. The background import process will take some time to complete, depending on the size of your repository. During this time Artifactory might perform more slowly than usual, but it will still serve any artifact immediately. The log does provide visible information about the progress of this process.

#### **Repeating the upgrade process**

Should you wish to repeat the upgrade process from scratch, make sure to remove the \$ARTIFACTORY\_HOME/data folder from your new Artifactory installation before doing so.

*artdump command line usage:*

```
artdump.sh
--help : displays this usage message
--home [old Artifactory home] : mandatory - the home folder of the old
Artifactory
--dest [destination folder] : the destination folder for the new export
files. Default value: tmpExport
--version [version name] : the actual version of the old Artifactory if
the Update Manager cannot find it
--repo [repo names separated by ':'] : export only a specified list of
repositories. Default value: all-non-cached
--norepo : does not export the repositories, just convert config and
security
--convert : activate the Local and Virtual repository names conversion
--noconvert : does not activate the Local and Virtual repository names
conversion during a full export
--security : only export the security file from DB, and set the norepo
flag
--caches : include cached repositories in the export (by default caches
are not exported). If repo option is passed this option will be ignored
The parameter --home is mandatory.

The Artifactory version will be extracted from
${artifactory.home}/webapps/artifactory.war if present
If the war file is not located there, please do:
1) link or copy it at this location, or pass the version.
2) pass one of the following version as second parameter:
 1.2.2-rc0 1.2.2-rc1 1.2.2-rc2 1.2.2 1.2.5-rc0
 1.2.5-rc1 1.2.5-rc2 1.2.5-rc3 1.2.5-rc4 1.2.5-rc5
 1.2.5-rc6 1.2.5 1.2.5u1 1.3.0-beta-1 1.3.0-beta-2
```

- The --home parameter is the old \$ARTIFACTORY\_HOME folder (the old Artifactory should not be running). If that folder also contains a webapps subfolder with the old Artifactory war file, the update manager will automatically determine the old version from it. If the update manager cannot determine a valid version an error message will appear with a list of valid versions and you'd need to manually specify the version of your old Artifactory as a second parameter to artdump.

*artadmin command line usage:*

```

artadmin
--info : Display general system information about Artifactory
--import [import from path] : Activate full system import from a
designated path
--export [export to path] : Activate full system export to path
--server [the server host or ip] : The remote Artifactory server IP or
host name with optional port number. The default is localhost:8081
--ssl : Activate https instead of http. Default is false
--timeout [timeout in seconds] : Set the timeout of the HTTP connection.
--url [root url to rest api] : The root URL of the Artifactory REST API.
The default is http://[servername]/artifactory/api
--username [username] : Optional username to use when connecting to the
remote Artifactory
--password [password] : The user's clear text password
--noMetadata : Exclude metadata information when importing/exporting
--symlinks : Use symbolic links to the original import path file (no file
copying)
--syncImport : Import directly into Artifactory without using the
background import process
--bypassFiltering : Avoid using existing repository filtering rules during
the export process
--createArchive : Zip the resulting folder after the export (slow)

You need to specify one of the following command parameters: --info,
--import or --export

```

- When used with the --symlinks option the import process can perform much faster (usually by a factor of 3-5). Note that when using this option your imported path effectively becomes an integral part of Artifactory and you should not move/remove it until the background import process has completed. This is indicated by the log message: Working copy commit done (0 files).

## Artifactory 2.1.1

We are pleased to announce the availability of Artifactory 2.1.1.

The major features and changes included in this release are:

### General

- New 'annotate' permission to control who can tag items with metadata
- Support for copying artifacts between repositories
- Servlet 2.4 compatibility restored (Artifactory can run again under Tomcat 5.5)
- Fixed intermittent wrong content-length issues
- Default proxy support (no need to manually assign a proxy for each remote repository)
- Ability to run the datastore garbage collection manually (Admin:Advanced:Maintenance:Storage)
- File system datastore for binaries now properly syncs changes to the search index

### Add-ons

- Watch events are now correctly emitted for move and copy operations
- Properties can now be set on cached items

For a complete list of resolved issues in this version please see the [JIRA](#).

Instructions for simple upgrade from previous versions can be found [here](#).

The latest Artifactory user guide is available [here](#).

Artifactory 2.1.1 is available for immediate download from JFrog's web site or directly from SourceForge.

Enjoy!

The Artifactory Team @ JFrog

## Artifactory 2.2.4

## We are pleased to announce the availability of Artifactory 2.2.4!

This release contains important improvements to Artifactory OSS and the Artifactory Pro Power Pack, as well as important maintenance bug fixes.

Notable improvements and changes in this version are:

1. **Performance Improvements** - Lower memory footprint, better overall memory management and better JVM and storage defaults for the OOTB installation.
2. **Folder & Repository Replication REST API** - Rsync-like replication of a remote repository or remote repository folder between two Artifactory servers. See: 'Folder Synchronization/Replication' under the [REST API documentation](#) (requires Artifactory Pro).
3. **Build Promotion REST API** - Promote (copy/move) a CI server build published artifacts with or without dependencies from selected scopes - all via REST. See: 'Build Promotion' under the [REST API documentation](#) (requires Artifactory Pro).
4. **Support for Multiple LDAP Servers** - Configure authentication and/or group authorization against multiple LDAP instances (requires Artifactory Pro).
5. **Extended Support for CI server Build Integration** - [Build info integration](#) supports non-numeric build numbers and better link-back to the CI server.
6. **Improved Code Highlighting** - Better support for Gradle scripts and JavaFx code.
7. **Better Support for Standalone Artifactory Behind Apache** - Improved request handling when running Artifactory on standalone Jetty behind Apache.
8. **New System Info Page** - For tracking [system resources](#) and serving as a convenient source of information when reporting issues.
9. **Improved Backups** - User-edited files such as artifactory system properties and the mime types are now also part of a system backup.
10. **Bookmarkable Tree Browser** - Artifact tree nodes are fully bookmarkable and can be shared as links (copy the link from the General tab when a node is selected).
11. **'Chroot' Support for UI Browsing** - Ability to run Artifactory with a '[chrooted](#)' UI browsing limiting browsing the server file system from the UI (e.g. for import/export) to a specified root path.

**Artifactory Pro Power Pack 2.2.4 Free Evaluation** is available for immediate download.  
The Artifactory OSS 2.2.4 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.2.4 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.2.5

## We are pleased to announce the availability of Artifactory 2.2.5!

This is a maintenance release for 2.2.4, containing bug fixes and small improvements.

Some changes in this version are:

- Selective scopes (or 'configurations' for Ivy) when saving Build Info artifacts/dependencies as a search result (requires the Pro Power Pack).
- REST API now accepts 'application/json' as content-type from clients when the return type is a json object with a more concrete json sub-type.
- Fixed a security issue with blank LDAP passwords and SunOne LDAP server.
- Support for manually clearing bad checksums from the UI.

Artifactory 2.2.5 is available for immediate download from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.3.3

## We are pleased to announce the availability of Artifactory 2.3.3!

Notable new features and changes in this version are:

1. **Repository Replication** - Rsync-like mirroring of your repository content and metadata from/to remote Artifactory repositories using pull and push (coming soon) replication (requires Artifactory Pro)
2. **Release Staging and Promotion Support** - Works in conjunction with the new release management features in the Jenkins Artifactory plugin. Similar support is coming to TeamCity and Bamboo (some features require Artifactory Pro)
3. **Filtered Resources** - Provision dynamic settings and configuration resources to clients. For example, provision different content base on the user originating IP address, or based on changing property values attached to the requested artifact (requires Artifactory Pro)
4. **S3 Remote Browsing** - Browse remote repositories hosted on Amazon S3
5. **Isolated Resolution Support for Snapshot Build Chains** - When running parallel integration build chains in Jenkins, Artifactory will return for each builds only dependencies that were produced as part of its build chain. Similar support is coming to TeamCity and Bamboo (requires Artifactory Pro)
6. **SSO by any HTTP Header** - The HTTP-SSO add-on now also supports authentication to Artifactory based on any trusted HTTP header
7. **RPM Distribution** - Artifactory can now be installed from an RPM on any RPM-supporting OS
8. **Bad Jars Protection** - Avoid filling up your repository with bad jars. For example, when running behind a captive portal
9. Many bug fixes and improvements

**Artifactory Pro 2.3.3 free trial** is available for [immediate download](#).

The Artifactory OSS 2.3.3 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.3.3 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.3.4

### We are pleased to announce the availability of Artifactory 2.3.4!

Notable new features and changes in this version are:

1. **Push Repository Replication** - Push repository replication completes the existing pull replication feature. You can now push content and metadata to remote Artifactory repositories on servers you are only allowed to make outgoing connections to (requires Artifactory Pro)
2. **Jar Entry Download** - Access and download archive content directly via HTTP GET. This enables, for example, reading javadocs documents directly from their Artifactory hosted jars (requires Artifactory Pro)
3. **Search API for Plugins** - Groovy plugins can now perform searches in Artifactory (requires Artifactory Pro)
4. **Improved REST API** - Item effective permissions, find builds using a dependency, artifacts with bad checksums search, etc.
5. **Test Remote Repository** - Verify you can successfully connect to a remote repository you have created
6. Compatible with the **Release Management** features of the upcoming version of the [TeamCity Artifactory Plug-in](#) and the [Bamboo Artifactory Plug-in](#)
7. Bug fixes and many small improvements

**Artifactory Pro 2.3.4 free trial** is available for [immediate download](#).

The Artifactory OSS 2.3.4 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.3.4 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.3.4.1

### We are pleased to announce the availability of Artifactory 2.3.4.1!

This version of Artifactory adds Java 7 compatibility to Artifactory!

#### Disabling HotSpot Loop Optimizations

Due to critical Hotspot bugs in the first Java 7 release(s) ([7070134](#), [7044738](#) & [7068051](#)), it is **HIGHLY RECOMMENDED** to run Artifactory with the following Hotspot flag in order to avoid JVM crashes and/or Lucene index corruption:

**-XX:-UseLoopPredicate**

Using this flag may not be necessary in future Java 7 releases.

**Artifactory Pro 2.3.4.1 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.3.4.1 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.3.4.1 from previous versions can be found [here](#).

**Please Note:** Due to a minor bug, the artifactory.log file may contain the following warning after upgrading Artifactory to 2.3.4.1. You can safely ignore this warning.

```
[art-init] [WARN] (o.a.v.ArtifactoryVersionReader:102) - Version 2.3.4.1 is not an official release version. The closest revision to 13021 will be used to determine the current version.
```

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.1.2

We are pleased to announce the availability of Artifactory 2.1.2.

Major features and changes in this release:

### General

- Initial Ivy support - Preview Ivy modules and Ivy dependency declaration, XPath search in ivy.xml files, auto-guess Ivy modules properties when uploading ivy.xml files from the UI
- Sharing of remote repository definitions - You can import and reuse remote repository definitions exposed by other Artifactory instances (new Import button in *Admin:Repositories:Remote Repositories*)
- Optimized binaries storage - Including configurable blobs caching and support for storing binaries as regular files (instead of as database blobs)
- Browser compatibility fixes

### Add-ons

- Exporting search results - Saved search results can now be exported to disk

For a complete list of resolved issues in this version please see the [JIRA](#).

Instructions for simple upgrade from previous versions can be found [here](#).

The latest Artifactory user guide is available [here](#).

Artifactory 2.1.2 is available for immediate download from [JFrog's web site](#) or directly from [SourceForge](#).

Enjoy!

The Artifactory Team @ JFrog

## Artifactory 2.1.3

### We are pleased to announce the availability of Artifactory 2.1.3!

The major features and changes in this release are:

- Hudson integration** - Use the [Hudson Artifactory Plugin](#) to deploy builds to Artifactory from Hudson together with build-time information: View builds in Artifactory with information about the deployed artifacts and dependencies (all scopes) and runtime environment per build, and link back to Hudson to obtain fully-reproducible builds.  
Via paid add-on: Visual artifact/dependency views per build; Promote or export all build artifacts and dependencies; See where specific artifacts are used and receive warnings when required build dependencies are removed .
- Automatic cleanup of remote repositories declared in POMs** - A virtual repository can now be configured to auto- clean up rogue

remote repositories declared in POM files, causing dependency resolution issues.

- **Logo branding** - Customize your Artifactory UI with your own logo (uploaded or URL linked) and footer
- **Ivy dependencies for POMs** - Copy Ivy dependency declarations from the POM view.
- **Generic artifact deployment support** - Improved upload screen with ability to deploy artifacts to any path without Maven's layout constraints.
- **Include/exclude patterns on virtual repositories** - This extra level of control supersedes the patterns define on sub-repositories.
- **Limit search to specific repositories**
- **Faster searches**

For a complete list of resolved issues in this version please see the [JIRA](#).

Instructions for simple upgrade from previous versions can be found [here](#).

The latest Artifactory user guide is available [here](#).

Artifactory 2.1.3 is available for immediate download from [JFrog's web site](#) or directly from [SourceForge](#).

Enjoy!

The Artifactory Team @ JFrog

## Artifactory 2.2.0

### We are pleased to announce the availability of Artifactory 2.2!

This forth major release of Artifactory is focused around significant performance improvements, resource usage optimizations, and bug fixes. Please see important [update notes](#) for this version.

This release also brings a couple of new important features to Artifactory:

1. **Open REST API** - Artifactory's RESTful resources are now public and documented. Using the [REST documentation](#) or the auto-generated WADL file, Artifactory can be controlled and automated from external tools or frameworks.
2. **Enterprise-grade LDAP Groups Synchronization** - this new [Power Pack add-on](#) lets you manage permissions in based on your existing LDAP groups. It offers fast caching of LDAP data (which is a must for an organization with hundreds or thousands of users), flexible group synchronization strategies, multiple settings, and tight integration and feedback about LDAP users and groups throughout all security-related management screens in Artifactory.
3. Support for running Artifactory under [GlassFish 3](#).
4. Support for [queueing concurrent downloads](#) of the same remote resource.
5. Ability to [annotate repositories](#) with administrative comments.
6. Improved [UI branding](#).

The LDAP Groups add-on joins the existing set of commercial [Artifactory Power Pack](#) add-ons, that complete the Artifactory professional offering. You might find the following [comparison table](#) helpful for comparing the open source version to the add-ons version.

Artifactory 2.2 is available for immediate download from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.2 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

## Artifactory 2.2.1

### We are pleased to announce the availability of Artifactory 2.2.1!

This is a maintenance bug fixes release for 2.2.0.

Artifactory 2.2.1 is available for immediate download from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.2.1 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.2.2

## We are pleased to announce the availability of Artifactory 2.2.2!

This is a maintenance release for 2.2.1 that also adds a couple of new noticeable features to Artifactory:

1. **Checksum Validation for Uploaded Artifacts** - Local repositories can now be assigned with a custom policy for validating the checksum of artifacts uploaded by client. You can tell Artifactory not to serve an artifact (return 404) until its checksum has been verified against the client's checksum.
2. **Queueing Concurrent Downloads** of Same Remote Resource - Artifactory will avoid bursts of concurrent downloads requests for the same remote resource and will queue all requests except the first one to avoid download overwriting, timeouts and bandwidth exhausting.
3. **Move/Copy via REST API** - Allowing arbitrary move/copy automation between paths and repositories, including metadata move/copy (requires the Artifactory Power Pack).
4. **Trigger Download REST API** - Automate a remote artifact download with or without getting the content back to the client (requires the Artifactory Power Pack).
5. **Property-based Download** - Request an artifact for download based on the existence of certain property value on it, including support for mandatory/non-mandatory values.
6. **Metadata Proxying** - Ability to proxy remote artifacts with their metadata.
7. **System-wide Syntax Coloring Support** - For XML, Java, Groovy files etc.
8. **Jar/Zip Content Browsing** - In tree-view you can drill down into Jars and view files, sources and the Manifest content.
9. **View Java Source from Class** - Ability to view the source of classes returned in search results or when browsing a Jar class that has a matching sources Jar in Artifactory.
10. **Better View of Dependent Build Artifacts** - Faster and uses less memory (requires the Artifactory Power Pack).
11. **Hide Repository in Tree Browse** - For users with no read permissions on any path in the repository.

Artifactory 2.2.2 is available for immediate download from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.2.2 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.2.3

## We are pleased to announce the availability of Artifactory 2.2.3!

This is a maintenance release for 2.2.2, containing bug fixes and small improvements.

Notable changes in this version are:

- Fixed a problem with trusted server checksums, leading to checksum errors.
- Fixed a regression with Maven metadata not being merged correctly for virtual repositories.
- Mime types are now configurable, allowing more file types to be viewed with syntax highlighting (requires index rebuilding).
- Source of class files can be viewed not only when the sources jar is available but also when sources are contained within the same jar.

Artifactory 2.2.3 is available for immediate download from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.2.3 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.3.0

## We are pleased to announce the availability of Artifactory 2.3!

This release contains new features and improvements to Artifactory OSS and Artifactory Pro Power Pack, as well as important maintenance bug fixes and performance improvements.

Notable new features in this version are:

1. **License Control Add-on** - A new Pro add-on that enables you to take full control over the licenses that are used by your third-party dependencies, as part of your builds. You can get immediate notifications about any libraries that violate your organization's license

policy, so you can deal with licensing issues early on during development.

This add-on integrates with Maven, Gradle and Ivy builds using the latest version of the Artifactory plugins for: Hudson, JetBrains' TeamCity and Atlassian's Bamboo.

Watch this screencast for a demo of License Control in action.

2. **User Plugins** - You can now extend Artifactory Pro with your own custom Groovy plugins. Using plugins you can schedule tasks, deploy artifacts, change resolution rules and download content, tend to any storage events etc. Plugin source files are redeployed on the fly during development, and can be edited and debugged in your favorite IDE.
3. **Atlassian Crowd Integration** - Delegate authentication request to a Crowd server and get transparent SSO in a Crowd-enabled SSO environment (requires Pro).
4. **Repository List Browsing** - A new lightweight browsing mode that resembles simple directory listing used by web servers. List browsing uses a unique URL prefix, which allows you to restrict public access only to it in a front-end web server.
5. **Repository Admin via REST** - A new API that enables creating, updating and deleting all repository types via REST (requires Pro).
6. **Automatic Gradle and Ivy Client Settings** - Generate Ivy and Gradle setting for resolution and deployment directly from the UI (to complete the Maven settings generator).
7. **Tomcat 7 Compatibility** - Artifactory is now compatible with Tomcat 7.

**Artifactory Pro 2.3 free trial** is available for [immediate download](#).

The Artifactory OSS 2.3 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.3 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.3.1

### We are pleased to announce the availability of Artifactory 2.3.1!

Notable features and changes in this version are:

1. **License Control Improvements** - Many improvements and small bug fixes mainly around third-party license violation notifications.  
*Note: You will need to upgrade the Artifactory plugin for your CI server to the latest version in order to take advantage of some of the improvements (disabling scanning own artifacts, scope selection etc.).*
2. **Execute Plugin via REST** - New REST API and extension point to execute any user plugin code via REST.
3. **Richer access.log** - More administrative actions and configuration changes are now reported in the access.log file.
4. **Gzip-encoded response** - Responses using gzip compression are now supported (typically returned by proxies not taking the Accept-Encoding header into account).

**Artifactory Pro 2.3.1 free trial** is available for [immediate download](#).

The Artifactory OSS 2.3.1 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.3.1 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.3.2

### We are pleased to announce the availability of Artifactory 2.3.2!

Notable features and changes in this version are:

1. **Remote Repository Browsing** - Browse the content of remote folders, not yet cached locally. This also allows Ivy resolvers to take into account remote versions when using version ranges.
2. **Crowd Groups Integration** - Sync your Atlassian Crowd groups to Artifactory and manage groups permissions on those groups (requires Artifactory Pro).
3. **Pluggable Authentication Realms** - super-easy login integration with any external system using a new Groovy extension point in User Plugins (requires Artifactory Pro).
4. **Non-Maven Repository Layouts** - Define the layout by which modules are identified in your repositories to achieve smart module management in non-Maven repositories, including: automatic cleanup of integration artifacts, cross-repository layout conversion (remote--local/cache--virtual), module metadata searches etc.
5. **Robust WebDAV Mounts** - Ability to mount any local repository or cache as a secure WebDAV share for direct browsing from your native O/S file manager.

6. **Automatic Cleanup of Old Builds** - Use the retention defined by your CI server for cleaning up old builds, when using the Artifactory B  
uild Integration (requires Artifactory Pro).
7. **Improved Memory Consumption** - Substantially improved memory utilization.
8. Over 90 bug fixes and improvements.

**Artifactory Pro 2.3.2 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.3.2 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.3.2 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.4.0

### We are pleased to announce the availability of Artifactory 2.4!

#### What's New in this Release

This major release of Artifactory introduces the following new features and changes:

1. **YUM Repositories and RPM Provisioning** - Artifactory can now act as a fully-featured YUM repository, including auto-updating repo metadata and RPM detailed view directly from the Artifactory UI.
2. **P2 Repositories** - Artifactory can be your single access point for all Eclipse® updates. Eclipse plugins proxying and hosting take advantage of Artifactory's exiting advanced caching and security controls.
3. **Major Performance Improvements** - in storage management, CPU and memory utilization and search speeds
4. **Security is Fully Manageable via REST API**
5. **User Regexp Tokens in Repository Layouts** - You can now add your own custom regexp-based tokens to repository layout definitions for better module identification.
6. New additions to the [Artifactory Public API](#) for [User Plugins](#) (move, copy, search, not downloaded since, etc.)
7. Usability improvements and many bug fixes

#### Before Upgrading - Important, Please Read This

1. **Plan for an Upgrade Downtime** - This major release involves storage-related changes, so while the upgrade process itself automatically runs from start to finish (as always), it will take some time for it to complete.  
The time between starting up Artifactory after upgrading and having it ready to serve requests can vary, depending of the size of your repository, but it is considerably longer compared to previous updates.  
*It is important that you run this upgrade while taking downtime into account.*
2. **Java 6 is Required** - Artifactory no longer runs on Java 5 and now requires Java 6. This enables us to improve the Artifactory codebase and use up-to-date dependencies that require Java 6.
3. **XML Search is Disabled by Default** - XML context indexing (and thus, searching) incurs some performance overhead and has been made optional.  
If you'd like to keep using this feature you will need to opt-in and enable XML searches. It is important that you do this before upgrading Artifactory, since newly created XMLs will not be indexed until XML indexing is enabled. Please see the details on how to do this on the [XML Search page](#).

**Artifactory Pro 2.4 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.4 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.4 from previous versions can be found [here](#).

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.4.1

### We are pleased to announce the availability of Artifactory 2.4.1!

This is a bug-fix minor release. Important issues addressed in this release are:

1. Non-unique snapshots are again updatable with 'deploy' permission (a regression caused 'delete' permission to be required)

2. Fixed potentially wrong checksums after upgrading to Artifactory 2.4.0
3. WebLogic 10.3 compatibility
4. Support for remote browsing of repositories that return a gzip-compressed response

**Artifactory Pro 2.4.1 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.4.1 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.4.1 from previous versions can be found [here](#).

**Important! Before upgrading from 2.3.x or earlier**

Please make sure to read the 'Before Upgrading' section in the [Artifactory 2.4.0 release notes](#) for important information about upgrading from 2.3.x to 2.4.x.

For a complete list of resolved issues in this version please see the [JIRA](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.4.2

### We are pleased to announce the availability of Artifactory 2.4.2!

Important issues addressed in this release are:

1. New REST API to allow tracking of replication status
2. Unlimited results for searches run from Artifactory plugins
3. Fixed a regression with Maven metadata expiry

For a complete list of resolved issues in this version please see the [JIRA](#).

**Artifactory Pro 2.4.2 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.4.2 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.4.2 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.5.0

### We are pleased to announce the availability of Artifactory 2.5. Now with NuGet Support!

Important issues addressed in this release are:

1. **NuGet Support** - Artifactory has across-the-board support for NuGet. This means that .Net developers can now use the full power of Artifactory's advanced artifact management to host, proxy and aggregate NuGet packages (via local, remote and virtual repos, respectively) and pull libraries from Artifactory into Visual Studio projects.
2. A slew of bug fixes and small improvements.

For a complete list of resolved issues in this version please see the [JIRA](#).

**Artifactory Pro 2.5 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.5 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.5 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.5.1

### We are pleased to announce the availability of Artifactory 2.5.1!

Important improvements and fixes included in this minor release are:

1. Maven indexer can now be triggered based on a cron expression
2. All zip-compatible archives can now be searched (previously was limited to jars)
3. New tab displaying NuGet package information (extracted from its embedded nuspec file)
4. Replication no longer aborts on failure to replicate a single artifact
5. Fixed empty response when list-browsing repositories with a short name

For a complete list of resolved issues in this version please see the [JIRA](#).

**Artifactory Pro 2.5.1 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.5 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.5.1 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.5.1.1

### We are pleased to announce the availability of Artifactory 2.5.1.1!

This minor release affects **.Net clients**. It addresses a critical bug that caused proxying of the NuGet Gallery to stop working.

**Artifactory Pro 2.5.1.1 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.5.1.1 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.5.1.1 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.5.2

### We are pleased to announce the availability of Artifactory 2.5.2

This minor release addresses an important security [vulnerability](#) that may allow an attacker to run a cross-site scripting attack by uploading malicious content to Artifactory.

**All users are urged to upgrade to this version.**

Other important issues addressed in this release:

1. YUM metadata is now correctly calculated for repositories hosted in subdirectories.
2. RPM installer now works correctly on SUSE.

For a complete list of resolved issues please see the [JIRA release notes](#).

**Artifactory Pro 2.5.2 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.5.2 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.5.2 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.6.0

### We are pleased to announce the availability of Artifactory 2.6!

Important new features and changes in this version:

1. **Download debugging** - solve artifact resolution issues by getting detailed explain plan of the download request handling in your browser.
2. **Latest Maven snapshot download** - it is now possible to download the latest version of a Maven snapshot via a simple non-unique

- snapshot URL. (Pro)
- 3. **Latest version REST** - Artifactory can return the latest release or integration version of an artifact for any repository layout. (Pro)
  - 4. **Artifact versions REST** - get all the release and/or integration versions of an artifact for any repository layout. (Pro)
  - 5. **Improved replication** - replication was optimized in terms of speed and overall resource consumption. Replication now uses streaming and duplicate target checksum elimination (Pro)
  - 6. **Auto offline for remote repositories** - a remote repository will now be auto put into temporal offline when encountering retrieval errors.
  - 7. **Before/After Build callbacks** - tend to build publishing events in [user plugins](#), including the ability to modify the build info object (e.g. add new artifacts). (Pro)
  - 8. **Cron-based artifact cleanup** - auto artifact cleanup can now be triggered by cron instead of using a time interval.
  - 9. **Custom artifact promotion** - it is now possible to code custom artifact promotion logic, including interacting with builds, directly in [user plugins](#). Promotion logic can be executed via REST or via [CI integration](#). (Pro)
  - 10. **AfterDownloadError callback** - tend to download errors in [user plugins](#). (Pro)
  - 11. **Secure plugin executions** - REST execution points in [user plugins](#) can be secured by specifying allowed users and/or groups. (Pro)
  - 12. **Per repository archive browsing** - ability to turn on archive browsing for selected trusted repositories.
  - 13. Many bug fixes and improvements

For a complete list of resolved issues please see the [JIRA release notes](#).

**Artifactory Pro 2.6.0 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.6.0 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.6.0 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.6.1

### We are pleased to announce the availability of Artifactory 2.6.1!

Important features and changes in this version:

- 1. **Faster GAVC search and quick search** - GAVC and Quick Search performance and resource consumption have been greatly improved
- 2. **Local replication regression** fixed
- 3. **Latest maven snapshot** download now resolves much faster
- 4. **Resolution repositories** are now correctly returned to Artifactory CI plugins
- 5. Minor bug fixes and improvements

For a complete list of resolved issues please see the [JIRA release notes](#).

**Artifactory Pro 2.6.1 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.6.1 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.6.1 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.6.2

### We are pleased to announce the availability of Artifactory 2.6.2!

Important features and changes in this version:

- 1. **Event based replication** - Replication now supports continuous, event-based mirroring that allows you to achieve near real-time push synchronization between repositories. Combined with scheduled replication, repositories are guaranteed to always be consistently in sync even when one of the replication sides becomes unavailable.
- 2. **Improved P2 Integration** - P2 support has been extended with more straight-forward hosting of your own P2 repositories, better composite resolution, and easier aggregation of local and remote P2 repositories under a single virtual repository URL.
- 3. **NuGet 2.0** - Artifactory is now up-to-speed with all the changes introduced by the recently released [NuGet 2.0](#).
- 4. **Consistent path handling** - URL and path logic handling has been rewritten to support across-the-board correct handling of special character encoding.
- 5. Minor bug fixes and improvements

For a complete list of resolved issues please see the [JIRA release notes](#).

**Artifactory Pro 2.6.2 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.6.2 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.6.2 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.6.3

### We are pleased to announce the availability of Artifactory 2.6.3!

Important features and changes in this version:

1. **Deploy artifacts from archive** - Added REST and UI support for batch-deployment of multiple artifacts contained in an uploaded archive, in a single HTTP transaction. Currently supported archive types: zip; tar; tar.gz; and tgz.
2. **NuGet 2.0 packages update** is now working in Visual Studio.
3. Minor bug fixes and improvements

For a complete list of resolved issues please see the [JIRA release notes](#).

**Artifactory Pro 2.6.3 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.6.3 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.6.3 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.6.4

### We are pleased to announce the availability of Artifactory 2.6.4!

Important features and changes in this version:

1. **Storage quota management**
2. Bug fixes and improvements

For a complete list of resolved issues please see the [JIRA release notes](#).

**Artifactory Pro 2.6.4 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.6.4 is available from [JFrog's web site](#) or directly from [SourceForge](#).

Instructions for upgrading to 2.6.4 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.6.5

### We are pleased to announce the availability of Artifactory 2.6.5!

Important features and changes in this version:

1. **Build Diffs**: Visually compare the differences between two build runs for changes in artifacts, dependencies and environment variables
2. **YUM Improvements**: Support for YUM groups and more efficient calculation of YUM metadata
3. **Build Artifacts Archive**: Retrieve an archive file containing all the artifacts related to a specific build
4. **Build Artifacts Search**: Find all the artifacts related to a specific build
5. **Maven Latest Release Download**: Latest release artifact query now also works for Maven-layout repositories
6. **Configurable Request Parameters**: Add custom query params to remote repository requests; This is useful, for example, when needing to add an authentication token param, such as the one required by Maven Central SSL access.
7. Bug fixes and performance improvements

For a complete list of resolved issues please see the [JIRA release notes](#).

**Artifactory Pro 2.6.5 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.6.5 is available from JFrog's web site or directly from SourceForge.

Instructions for upgrading to 2.6.5 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.6.6

### We are pleased to announce the availability of Artifactory 2.6.6!

Important features and changes in this version:

1. **Per-file License Discovery:** You can now manually trigger a license scan per-artifact from the tree browser. This allows you to auto-attach license information to artifacts outside the context of a CI build.
2. **SAML SSO Support:** Login to the Artifactory UI can now be integrated with any SAML IdP (Identity Provider). Artifactory now acts as a SAML Service Provider.
3. **Bintray Integration:** Freely distribute your release artifacts and share them with the world via the [Bintray](#) service (currently in Beta). You can upload individual binaries or build artifacts to Bintray directly from Artifactory.
4. **Build Diff REST:** Compare two builds for their published artifacts, used dependencies and environment using the REST API.
5. **Running Plugin Code as System:** It is now possible to execute selected plugin code blocks under an unrestricted system role, by using the 'asSystem{ }' closure.
6. **Property Events in Plugins:** New storage callbacks are available that allow plugin developers to intercept property change events.
7. Bug fixes and performance improvements

For a complete list of resolved issues please see the [JIRA release notes](#).

**Artifactory Pro 2.6.6 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.6.6 is available from JFrog's web site or directly from SourceForge.

Instructions for upgrading to 2.6.6 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Artifactory 2.6.7

### We are pleased to announce the availability of Artifactory 2.6.7!

This minor release addresses few critical bugs and [security improvement](#).

For a complete list of resolved issues please see the [JIRA release notes](#).

**Artifactory Pro 2.6.7 free trial** is available for [immediate download](#).  
The Artifactory OSS 2.6.7 is available from JFrog's web site or directly from SourceForge.

Instructions for upgrading to 2.6.7 from previous versions can be found [here](#).

Enjoy Artifactory!

The Artifactory Team

## Pivotal Cloud Foundry JFrog Artifactory Tile Release Notes

### Overview

List of modifications and versions for the PCF Tile specifically.

#### Page Contents

- Overview
  - Version 0.6.1
  - Version 0.5.8

## **Version 0.6.1**

Second beta release. Now has the ability to configure an NFS mount. If the tab is left blank, it will generate an NFS mount with bosh. If you are specifying an external NFS release, you should go to 'Resource Config' and set the number of NFS server instances to 0. Note that to use an external NFS, it may require no-root-squash be enabled to function correctly.

Artifactory 4.1.3

## **Version 0.5.8**

First MVP version

Artifactory 4.1.3