

Npm Registry

Overview

Artifactory provides full support for managing npm packages and ensures optimal and reliable access to *npmjs.org*. Aggregating multiple npm registries under a virtual repository Artifactory provides access to all your npm packages through a single URL for both upload and download.

As a fully-fledged npm registry on top of its capabilities for [advanced artifact management](#), Artifactory's 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.

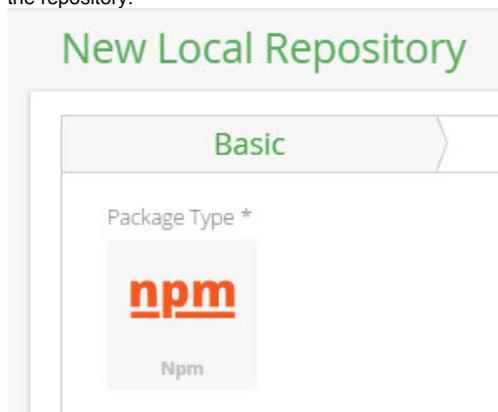
Page Contents

- Overview
- Configuration
 - Local Npm Registry
 - Repository Layout
 - Remote Npm Registry
 - Virtual Npm Registry
 - Advanced Configuration
- Using the Npm Command Line
- Npm Publish (Deploying Packages)
 - Setting Your Credentials
 - Deploying Your Packages
- Working with Artifactory without Anonymous Access
- Using OAuth Credentials
- Npm Search
- Cleaning Up the Local Npm Cache
- Npm Scope Packages
 - Configuring the npm Client for a Scope Registry
 - Using Login Credentials
- Automatically Rewriting External Dependencies
 - Rewriting Workflow
- Viewing Individual Npm Package Information

Configuration

Local Npm Registry

To enable calculation of npm package metadata in local repositories so they are, in effect, npm registries, set the **Package Type** to **npm** when you create the repository:



Repository Layout

Artifactory allows you to define any layout for your npm registries. 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 Npm Registry

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 registry.

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 with the following fields:

- Package Type ***: npm (with npm logo)
- Repository Key ***: npm-remote
- URL ***: https://registry.npmjs.org
- Test** button

3. Click "Save & Finish"

Virtual Npm Registry

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 registry, create a [virtual repository](#), set the **Package Type** to be **npm**, and select the underlying local and remote npm registries to include in the **Basic** settings tab.

The screenshot shows the 'Repositories' configuration page with the following sections:

- Filter...** (input field)
- Available Repositories**: mytestnpm-local, npm
- Selected Repositories**: npm-local, 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 SaaS the URL would be:

```
https://<server name>.jfrog.io/<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 ✕

Tool

npm npm

Remove User Data

Repository

npm-local

General

In order for your npm command line client to work with Artifactory you will firstly need to set the relevant authentication. For getting authentication details run the following command:

```
1 curl -uadmin:AP77uEE4HsJppEUJjjUBwRjweey http://10.100.1.110:8081/artifactory/api/npm/auth
```

The response should be pasted in the `~/.npmrc` (in Windows `%USERPROFILE%\\.npmrc`) file. Here is an example of the content of the file:

```
1 _auth = YwRtaiW46QVA3N3VFRTRic0pwcEVVSmpqVUJXUmp3ZiW5
2 email = youremail@email.com
3 always-auth = true
```

Artifactory also support scoped packages. For getting authentication details run the following command:

```
1 curl -uadmin:AP77uEE4HsJppEUJjjUBwRjweey "http://10.100.1.110:8081/artifactory/api/npm/npm-local/auth/<SCOPE>"
```

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 = YWRtaW46e0RFU2VkZX1uOFRaaXh1Y0t3bHN4c2RCTVIwNjF3PT0=
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.

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.

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`.

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 <https://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. In order work with npm scoped 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
```

Note that since Artifactory **version 4.4.3**, the bundled Tomcat is configured by default to enable encoded slashes. If you are using a previous version you will need to adjust the Tomcat property above.



URL decoding and reverse proxy

If Artifactory is running behind a reverse proxy, make sure to disable URL decoding on the proxy itself in order to work with npm scope packages.

For Apache, add the "`AllowEncodedSlashes NoDecode`" directive inside the `<VirtualHost *:xxx>` block.

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=admin
//localhost:8080/artifactory/api/npm/npm-local/:email=admin@jfrog.com
//localhost:8080/artifactory/api/npm/npm-local/:always-auth=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.

Automatically Rewriting External Dependencies

Packages requested by the Npm client frequently use external dependencies as defined in the packages' `package.json` file. These dependencies may, in turn, need additional dependencies. Therefore, when downloading an Npm package, you may not have full visibility into the full set of dependencies that your original package needs (whether directly or transitively). As a result, you are at risk of downloading malicious dependencies from unknown external resources. To manage this risk, and maintain the best practice of consuming external packages through Artifactory, you may specify a "safe" whitelist from which dependencies may be downloaded, cached in Artifactory and configure to rewrite the dependencies so that the Npm client accesses dependencies through a virtual repository as follows:

- Check **Enable Dependency Rewrite** in the Npm virtual repository [advanced configuration](#).
- Specify a whitelist patterns of external resources from which dependencies may be downloaded.
- Specify the remote repository in which those dependencies should be cached.
It is preferable to configure a dedicated remote repository for that purpose so it is easier to maintain.

In the example below the external dependencies will be cached in "npm" remote repository and only package from `https://github.com/jfrogdev` are allowed to be cached.

New Virtual Repository

Basic > **Advanced**

Artifactory Requests Can Retrieve Remote Artifacts ?

External Dependency Rewrite

Enable Dependency Rewrite

Remote Repository For Cache

npm

Patterns Whitelist ?

New Pattern +

/github.com/jfrogdev/

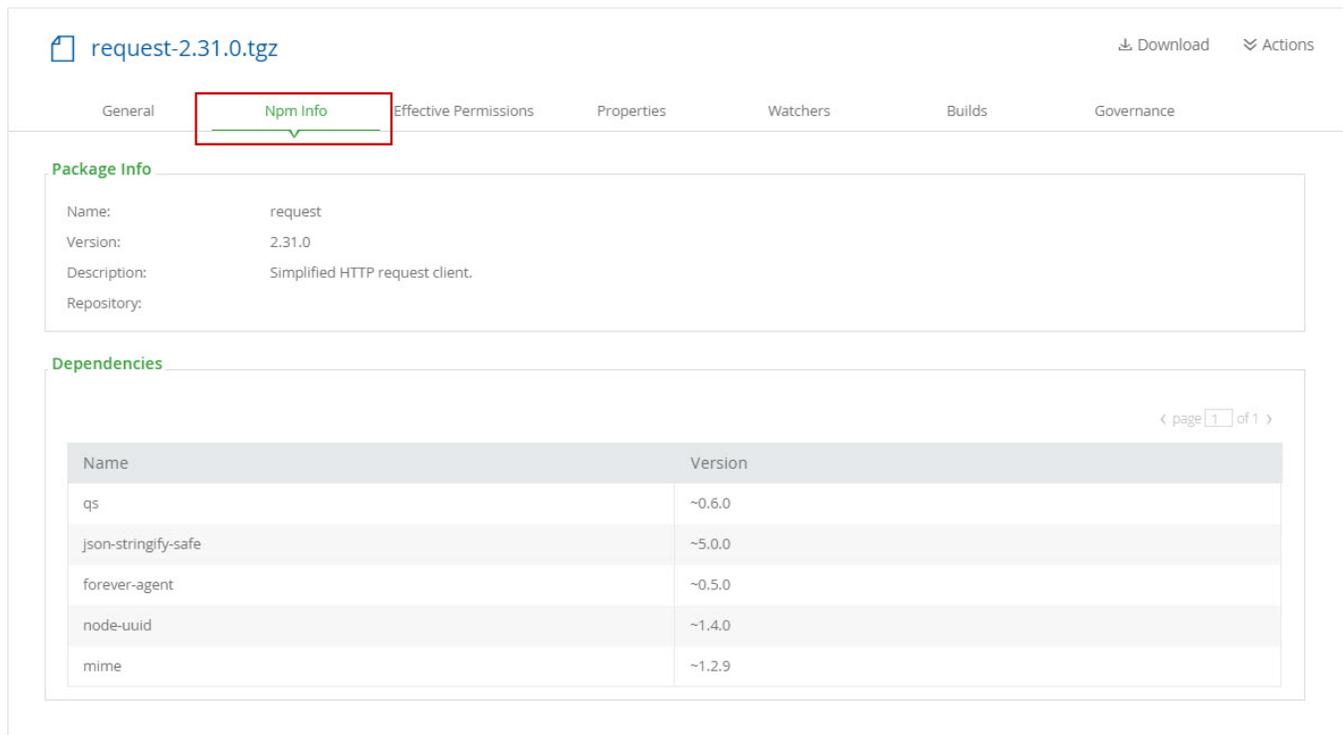
Rewriting Workflow

1. When downloading an Npm package, Artifactory analyzes the list of dependencies needed by the package.
2. If any of the dependencies are hosted on external resources (e.g. on *github.com*), and those resources are specified in the white list,
 - a. Artifactory will download the dependency from the external resource.
 - b. Artifactory will cache the dependency in the remote repository configured to cache the external dependency.
 - c. 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.
3. 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.



request-2.31.0.tgz

Download Actions

General **Npm Info** Effective Permissions Properties Watchers Builds Governance

Package Info

Name: request
Version: 2.31.0
Description: Simplified HTTP request client.
Repository:

Dependencies

< page 1 of 1 >

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