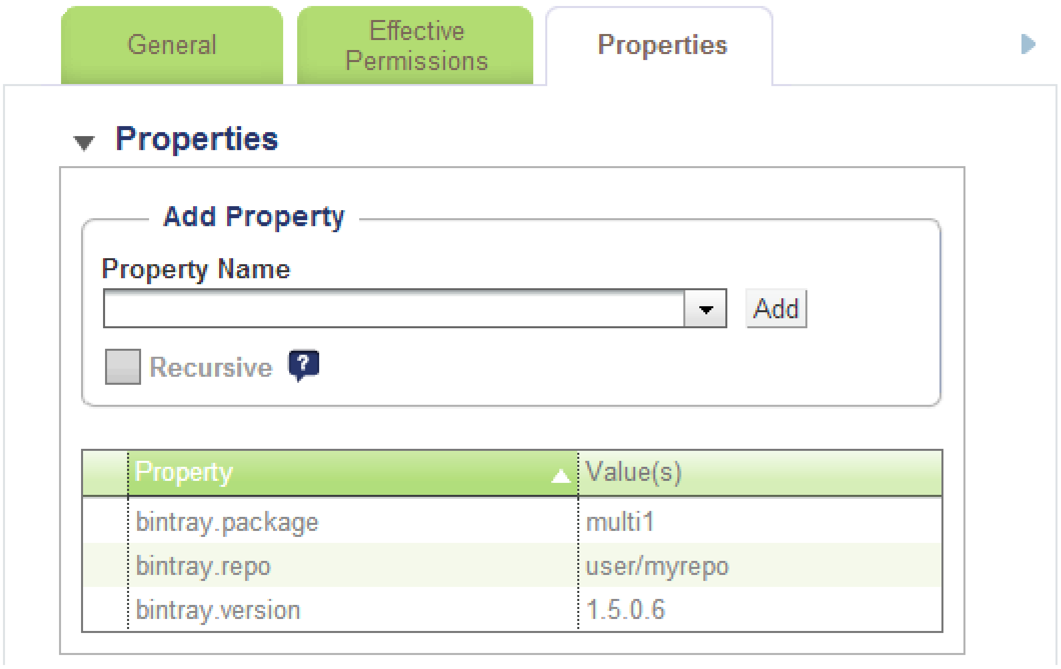
Push to Bintray

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
bintray.repo	A target repository in Bintray, in the format of *`{username}/{repository}`*
bintray. package	A target package name under the repository.You must first create the package in Bintray if it does not exist.
bintray. version	A target version under the package.If the version does not yet exist in Bintray, it is created automatically.
bintray.path	A target path in the repository under which to save the file. The path is considered optional as Artifactory will use the same path the file is stored in the repository.



Property	Value(s)
bintray.package	multi1
bintray.repo	user/myrepo
bintray.version	1.5.0.6

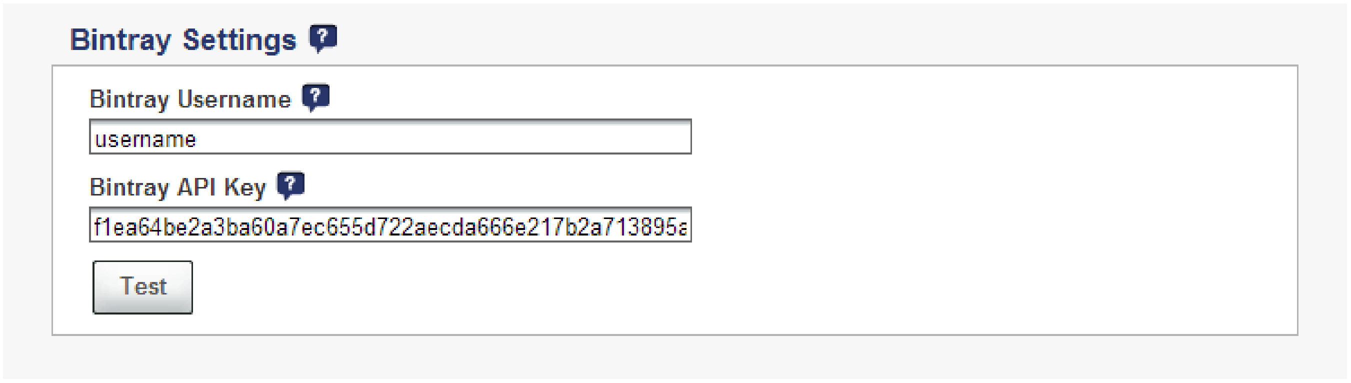
Usually, these properties will not exist on the artifact if it was not pushed before, using the UI above will attach them according to the user input.
All of the properties can be pre-populated (for example by your build tool), in this case Artifactory will use any existing property and **will ignore** the user input from the UI unless the property doesn't exist.

Page Contents

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

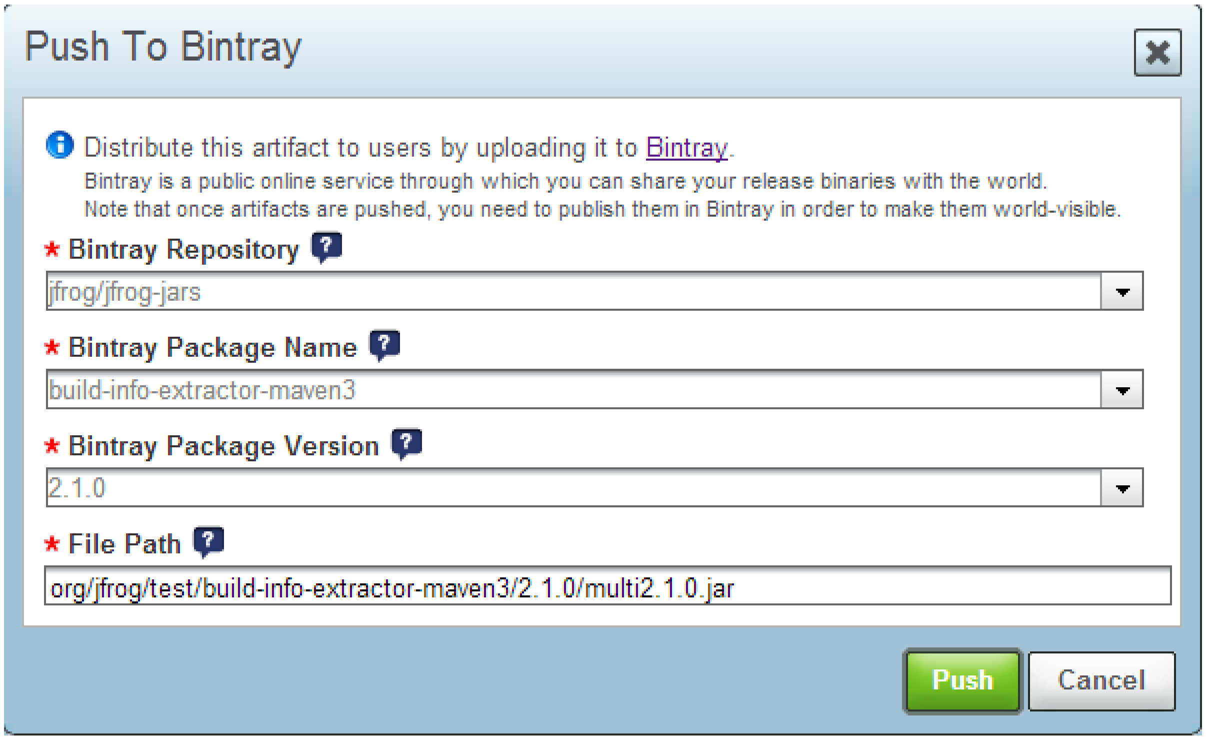
username

Bintray API Key

f1ea64be2a3ba60a7ec655d722aecda666e217b2a713895a

Pushing a Single Artifact

Pushing a single artifact is done from the Tree Browser, simply click on the file and choose the General Info tab:


Push To Bintray

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

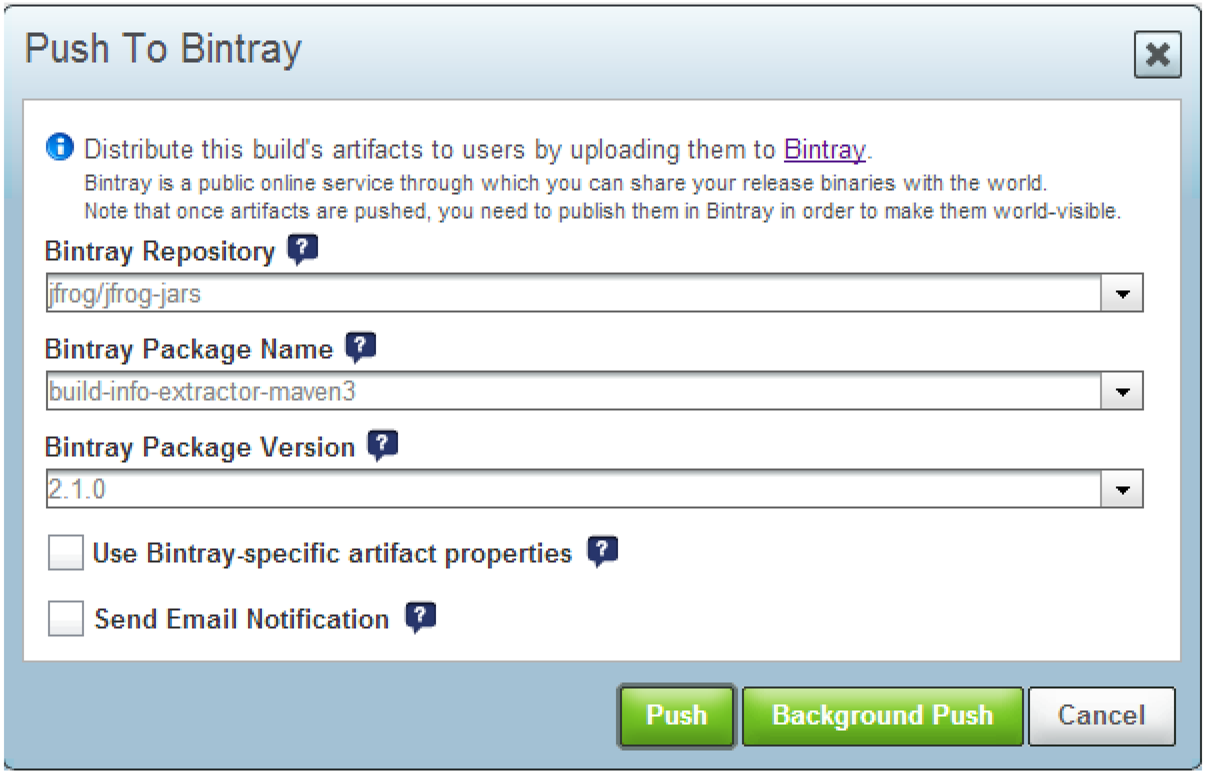
* Bintray Package Name: build-info-extractor-maven3

* Bintray Package Version: 2.1.0

* File Path: org/jfrog/test/build-info-extractor-maven3/2.1.0/multi2.1.0.jar

Push a Complete Build

Pushing a complete released build is done from the build General Info panel and it essentially pushes all the build artifacts one by one:


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

- 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 it's 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:

```
{
-        "repo": {
+                "name": "test",
-                "type": "generic",
-                "private": false,
-                "premium": false,
-                "desc": "My test repo",
-                "labels": ["label1", "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
updateExisting	Signifies Artifactory should update an existing repository with the same name with the values in the json descriptor (applies only to the *'labels'* and *'desc'* fields)
subject	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 *default* 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 of files (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
repo1/org/jfrog/myTest.jar	The file *myTest.jar* under *repo1/org/jfrog*
repo1/org/jfrog/.**	All files under *repo1/org/jfrog*
repo2/org/jfrog/test/module/*.jar*	All *.jar* files under any subfolder of *repo2/org/jfrog/test* whose name starts with "*module*"
*repo2/org/jfrog/test/**/*.jar*	All *.jar* files under any subfolder of *repo2/org/jfrog/test*
*repo2/org/jfrog/test/**/art.?ar*	All files named "*art*" with a file extension that has 2 or 3 characters and ends with "*ar*" under any subfolder of *repo2/org/jfrog/test*

Field	Purpose
applyToRepoFiles	If you are pushing a complete build, this field should remain empty. When pushing files, this field should contain a comma-separated list of files(in JSON format) that should be pushed. A file matching **any** of the file specifications will be pushed (i.e. an "OR" relationship). This field behaves similarly to applyToFiles, including wildcards as described above, only it refers to **relative paths** inside the repo that contains the json descriptor file: • If the path starts with a leading ' / ' then the parent for this path is the repository's root, so the path */org/jfrog/*.** will actually point to *containingRepo/org/jfrog/*.** • If the path **doesn't** start with a ' / ' then the parent for this path is the folder containing the descriptor. So if the descriptor resides in /org/jfrog/bintray-info.json, the path */test/myPackage/*.** will actually point to *containingRepo/org/jfrog/test/myPackage/*.**
applyToProps	0 or more key:value pairs with which to filter the selected files by properties. The '*' and '?' wildcards are supported in this filter as well. A file matching **all** of the property specifications will be pushed (i.e. an "AND relationship)
publish	If set to true, the version will be automatically published once the push operation is complete.
gpgSign	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. If you provide the *gpgPassphrase* 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.

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 vise 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 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**.