Bintray Distribution Repositories

Overview

Distribution repositories provide an easy way to move artifacts from Artifactory to Bintray, for distribution to end users. As opposed to other repositories, distribution repositories are not typed to a particular package format, but rather, are governed by a set of rules that specify how an artifact that gets to the distribution repository should be routed to its corresponding repository in Bintray.

Configuring a Bintray Distribution Repository

The following workflow describes the main steps for creating a Bintray Distribution repository:

1. Under the Distribution tab, click New and select Bintray Distribution Repository.
2. Connect to Bintray: Obtain authorization from your Bintray account for Artifactory to connect and deploy packages.
3. Configure distribution: Set the basic distribution parameters.
4. Configure Advanced Settings: Set the advanced settings.
5. Define rules: Set the rules that govern how this distribution repository will deploy packages to Bintray.

Creating the Repository

Access your Distribution repositories in the Administration module under Repositories | Repositories tab.

Connecting to Bintray

Artifacts are synchronized from your the Distribution repository to Bintray through a Bintray organization to which you have administrative privileges. To set up the connection, you first need Artifactory to access your Bintray organization. Artifactory will display a popup dialog where you can enter your Bintray credentials.
After authorizing access to your account, you need to select the organization in that account that you authorize Artifactory to manage.

Bintray will issue an authorization code which you need to copy, and then paste in your distribution repository configuration.

When you close the popup with the authorization code, Artifactory will display a popup for you to enter it.
Once the process is complete, you can verify that Artifactory has been authorized to access your Bintray organization, in that organization's profile page under OAuth Applications.

**Setting Basic Distribution Parameters**

Once you have set up Artifactory as an authorized application in your Bintray organization, you can set up the distribution parameters.

**Basic Distribution Parameters**

Once you have set up Artifactory as an authorized application in your Bintray organization, you can set up the distribution parameters.
The repository key.

A description of the repository.

When set, indicates that artifacts distributed through this repository should be linked to a product. Artifactory will create the product (if necessary) and link deployed packages to the product.

The product name.

The client ID assigned by Bintray to Artifactory as an authorized application.

The Bintray organization which Artifactory is authorized to manage for distribution.

If Artifactory creates a new repository on Bintray for distribution, this setting specifies if the repository should be private or public. If the repository exists, Artifactory will not override its access regardless of this setting.
## Default Package Settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Licenses</td>
<td>Specifies the OSS licenses that should be attached to any packages distributed through this repository.</td>
</tr>
<tr>
<td>VCS URL</td>
<td>Specifies the VCS URL for packages distributed through this repository.</td>
</tr>
</tbody>
</table>

## Advanced Settings

### New Distribution Repository

#### Basic

- **Proxy**
- **GPG Signing**
- **GPG Passphrase**

#### Advanced

- **Map Properties to Bintray Attributes**

#### Rules

- **Proxy**
- **GPG Signing**
- **GPG Passphrase**
- **Map Properties to Bintray Attributes**

### Managing Rules

As opposed to local, remote and virtual repositories in Artifactory, distribution repositories are not limited to a specific package type. Instead, you specify a set of rules, based on package type, and different filters, that give you fine-grained control over how different packages are pushed to Bintray for distribution. To view the rules defined for your repository, click the **Rules** tab. New distribution repositories come with a pre-defined set of rules which you can modify, delete or add to as needed.
Rule Order

The order in which rules are displayed specifies the order in which they are applied. To change the rule order, you can select a rule and drag it to a new location in the list.

Filtering and deleting

Start typing the name of a rule in the filter box to find the rule you are looking for.

Hover over a rule and click the delete icon on the right to delete it, or select a number of rules in the left column and click Delete to remove several rules at a time.

Conflicting rules

While it is possible to specify rules that conflict, since rules are applied in the order in which they appear, the first rule that is applied will take precedence.

Specifying Rule Parameters

To specify rule parameters, click New for a new rule, or the Name of a rule you want to edit.
<table>
<thead>
<tr>
<th>Name</th>
<th>A logical name for this rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artifactory Input</td>
<td>Parameters that determine which packages in Artifactory this rule applies to.</td>
</tr>
<tr>
<td>Package Type</td>
<td>Specifies the artifact package type for which this rule should be applied. Artifacts of other package types are ignored</td>
</tr>
<tr>
<td>Available Tokens</td>
<td>According to the package type selected, this specifies which tokens can be used to specify the deployment path in Bintray</td>
</tr>
<tr>
<td>Repository Filter</td>
<td>Wildcard expression that specifies for which original source repositories this rule should be applied. Packages in other repositories are ignored.</td>
</tr>
<tr>
<td>Path Filter</td>
<td>Wildcard expression that specifies the artifact path for which this rule should be applied. Packages that have a different path are ignored.</td>
</tr>
<tr>
<td>Bintray Output</td>
<td>Parameters that determine how artifacts should be deployed to Bintray.</td>
</tr>
<tr>
<td>Repository</td>
<td>The Bintray repository to which artifacts should be deployed.</td>
</tr>
<tr>
<td>Package</td>
<td>The Bintray package in the specified Bintray repository to which the artifact should be deployed. If the package does not exist, it will be created</td>
</tr>
<tr>
<td>Version</td>
<td>The Bintray version in the specified Bintray package and repository to which the artifact should be deployed. If the version does not exist, it will be created.</td>
</tr>
<tr>
<td>Path</td>
<td>The path in Bintray into which the artifact should be deployed in the specified repository.</td>
</tr>
</tbody>
</table>
Repository and Path Filter Parameters

Rules give you enormous flexibility in how you deploy artifacts to Bintray through use of regular expressions with capture groups where the capture groups may be named or unnamed.

Unnamed capture groups are back-referenced using placeholder tokens, while named capture groups are back-referenced using their names.

Using Unnamed Capture Groups

For each regular expression used in Repository Filter or Path Filter fields of the Artifactory Input section, you can place tokens in fields of the Bintray Output section to back-reference them.

Tokens are written using the following format:

${source:x}

where:

| source | path: A wildcard from the Path Filter field should be replaced |
| repo: A wildcard from the Repository Filter field should be replaced |
| x      | The wildcard number. |

For example,

${path:1} means replace the first regular expression in the Path Filter field

${repo:2} means replace the second regular expression in the Repository Filter field

Example 1

Under Artifactory Input, set Path Filter = jfrog-(.*)\.rpm
Under Bintray Output, set Repository = rpm-$(path:1)

With these settings, a package called jfrog-artifactory.rpm will be deployed to a repository in Bintray called rpm-artifactory, while a package called jfrog-mission-control.rpm will be deployed to a repository in Bintray called rpm-mission-control

Example 2

Under Artifactory Input, Repository Filter = libs-(.*)
Under Bintray Output, Repository = rpm-$(repo:1)

With these settings:

A package called jfrog-artifactory.rpm from libs-release-local will be deployed to a repository in Bintray called rpm-release-local.

A package called jfrog-artifactory.rpm from libs-snapshot-local will be deployed to a repository in Bintray called rpm-snapshot-local.

Using Named Capture Groups

You can give capture groups in the Repository Filter or Path Filter fields of the Artifactory Input section specific names and then back-reference the capture group using its name in the fields of the Bintray Output section.

Named capture groups are written using the following format:

(?<name>regex)

where:

| name   | A logical name for the capture group |
| regex  | The regular expression that defines repositories or paths that should pass through the filter |

Once capture groups are defined, you can back-reference them in the fields of the Bintray Output section using the following format:

$(source:name)
where:

<table>
<thead>
<tr>
<th>source</th>
<th>path: A capture group from the Path Filter field should be back-referenced</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>repo: A capture group from the Repository Filter field should be back-referenced</td>
</tr>
<tr>
<td>name</td>
<td>The name of the capture group.</td>
</tr>
</tbody>
</table>

**Example 1**

If you set:

Under Artifactory Input, **Repository Filter** = (?<myRepo>-local)

Under Bintray Output, **Repository** = generic-${repo:myRepo}

Then a package in repository **builds-local** will be deployed to a repository in Bintray called **generic-builds**.

**Example 2**

If you set:

Under Artifactory Input, **Path Filter** = jfrog-(?<myPath>.*).rpm

Under Bintray Output, **Repository** = rpm-{$path:myPath}

Then a package called **jfrog-artifactory.rpm** in repository **libs-release-local** will be deployed to a repository in Bintray called **rpm-artifactory**, while a package called **jfrog-mission-control.rpm** will be deployed to a repository in Bintray called **rpm-mission-control**.

**Enumerating Capture Groups**

You can use multiple capture groups when specifying rule parameters, and they are enumerated in the order in which they are received.

For example, if you set:

Under Artifactory Input, **Path Filter** = jfrog-(.*)

Under Bintray Output, **Repository** = ${path:1}-${path:2}

Then:

- **jfrog-artifactory.rpm** will be deployed to a repository in Bintray called **rpm-artifactory**
- **jfrog-artifactory.zip** will be deployed to a repository in Bintray called **zip-artifactory**
- **jfrog-mission-control.rpm** will be deployed to a repository in Bintray called **rpm-mission-control**
- **jfrog-mission-control.zip** will be deployed to a repository in Bintray called **zip-mission-control**

You can even mix both named and unnamed capture groups together when specifying rule parameters, however, they are still enumerated in the order in which they are received.

Expanding the example above, if you set:

Under Artifactory Input, **Path Filter** = jfrog-(?<type>.*).(*)

Under Bintray Output, **Repository** = ${path:2}-${path:1}

"type" is enumerated as the first capture group, and it is back-referenced using its name.

The unnamed capture group is unnamed and second in the enumeration order, so it is back-referenced using its number, 2.

**Distributing Artifacts**

Once you have your distribution repositories configured, the last step in getting your files to Bintray is to invoke distribution. When you invoke distribution, Artifactory goes through the rules defined for your distribution repository in order until the artifact you are trying to distribute passes one of them, and then uploads the file to Bintray in accordance with that rule. Once distributed, the file will also appear in your distribution repository to indicate that it has been uploaded to Bintray.

There are two ways to do this:

1. Distributing through the UI
2. Distributing via REST API

**Distributing Through the UI**

To distribute a file through the UI, select it in the Artifact Repository Browser and click **Distribute** in the right-click menu.
Artifactory will pop up the Distribution dialog where you can set final parameters for distribution.

**Distribution Repository**
The distribution repository through which the artifact should be uploaded to Bintray. The rules defined for this repository will govern if/how the file is uploaded.

**Distribute Artifacts Asynchronously**
When checked, the file will be uploaded asynchronously. To verify upload to Bintray succeeded, refresh your distribution repository to see the distributed file.

When unchecked, if upload to Bintray fails, Artifactory will display an error message.

If upload to Bintray fails, please refer to the Artifactory System Log for details.

**Publish Distributed Artifacts**
When checked, files uploaded to Bintray are published to make them available for download by end users.

**Override Existing Files**
When checked, the uploaded file will override another file with the same name if it exists in the upload path.

<table>
<thead>
<tr>
<th>Distribution Repository</th>
<th>The distribution repository through which the artifact should be uploaded to Bintray. The rules defined for this repository will govern if/how the file is uploaded.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribute Artifacts Asynchronously</td>
<td>When checked, the file will be uploaded asynchronously. To verify upload to Bintray succeeded, refresh your distribution repository to see the distributed file. When unchecked, if upload to Bintray fails, Artifactory will display an error message. If upload to Bintray fails, please refer to the Artifactory System Log for details.</td>
</tr>
<tr>
<td>Publish Distributed Artifacts</td>
<td>When checked, files uploaded to Bintray are published to make them available for download by end users</td>
</tr>
<tr>
<td>Override Existing Files</td>
<td>When checked, the uploaded file will override another file with the same name if it exists in the upload path.</td>
</tr>
</tbody>
</table>

Unticking **Distribute Artifacts Asynchronously** will produce a UI screen representing the progress and summary of the distribution process:

Once distribution is complete, Artifactory displays the outcome of the process in a **Success** section and an **Error** section.

The **Success** section displays the repository, package and version in which the distributed artifacts will be hosted in Bintray. If the distribution process created a package or version, this will also be indicated in the Success section. In the example below, both a package and a version were created.
If an error occurs in moving distributed files to Bintray, the Distribution dialog will also display an **Errors** section. In the example below, the success section shows that there was an attempt to upload files to version 4.0.2 in package `JSON` in repository `generic`. The **Errors** section provides a detailed error message explaining the failure.

---

**Success**

- **generic Repository**
  - **JSON Package**
    - **4.0.2 Version**

  Created a new package JSON with license MIT
  
  Created new version
  
  coco-local/3vis/JSON/tags/4.0.2/JSON-4.0.2.tar.gz -> generic/JSON/4.0.2/3vis/JSON/tags/4.0.2/JSON-4.0.2.tar.gz

---

**Errors**

File `coco-local/3vis/JSON/tags/4.0.2/JSON-4.0.2.tar.gz` resulted with the following errors

- Error distributing coco-local/3vis/JSON/tags/4.0.2/JSON-4.0.2.tar.gz -> generic/JSON/4.0.2/3vis/JSON/tags/4.0.2/JSON-4.0.2.tar.gz: 409, Conflict Unable to upload files: An artifact with the path '3vis/JSON/tags/4.0.2/JSON-4.0.2.tar.gz' already exists

---

**Dry Run**

A dry run simulates the act of moving your selected files to Bintray without actually distributing them. This is a good way to ensure your configuration is correct and that there is no impediment to moving your files to Bintray.

Click "Dry Run" to start the simulation. Once completed, Artifactory will display the **Success** and **Error** sections as if the files were actually distributed.

- Don't get confused
  In a dry run, no files are moved; it's just a simulation

---

**Distributing via REST API**

JFrog Artifactory exposes a REST API that lets you automate deploying artifacts to Bintray. For details please refer to Distribute Artifact and Distribute Build.