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 processer 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.

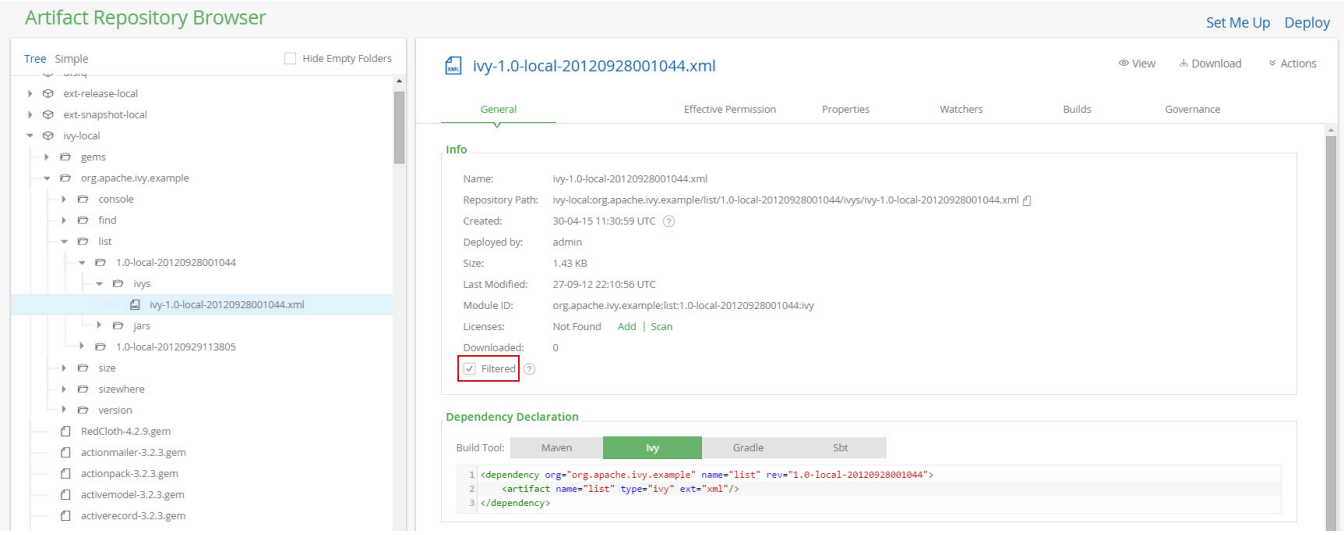
For example, you can provision different content based on the user's originating IP address or based on changing property values attached to the artifact.

Page Contents

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

---

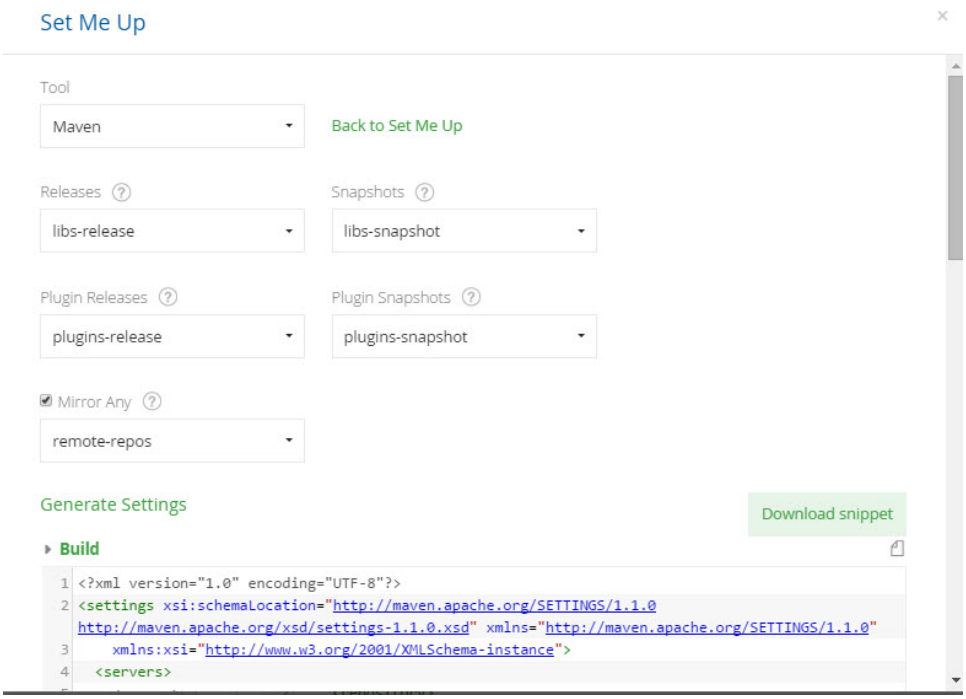
Provisioning Build Tool Settings

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.

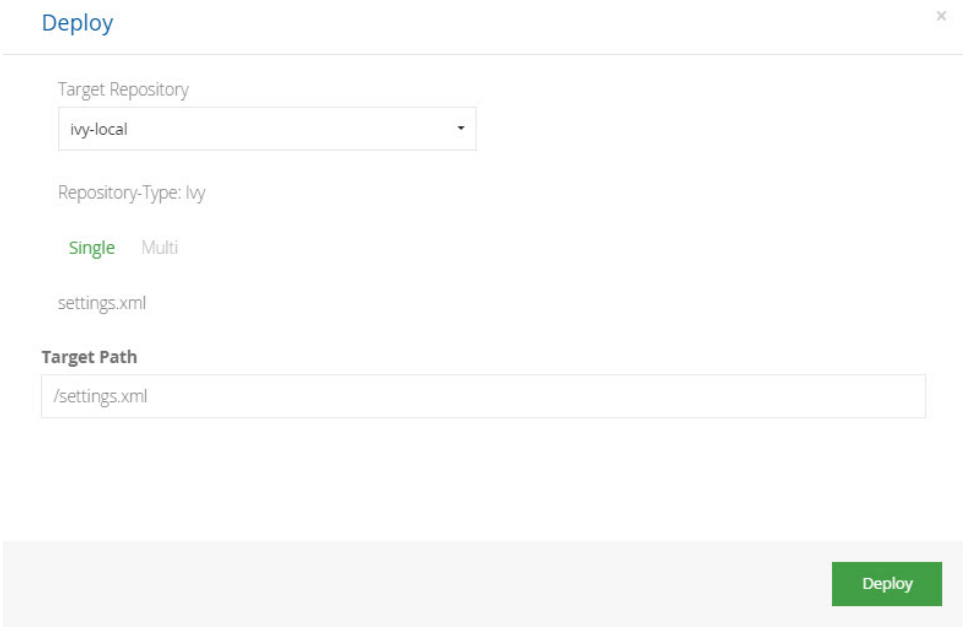
To provision user-generated settings:

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".



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>
```