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
7. Support for NuGet API v3 Registries.
8. Support for NuGet SemVer 2.0 Package Support.

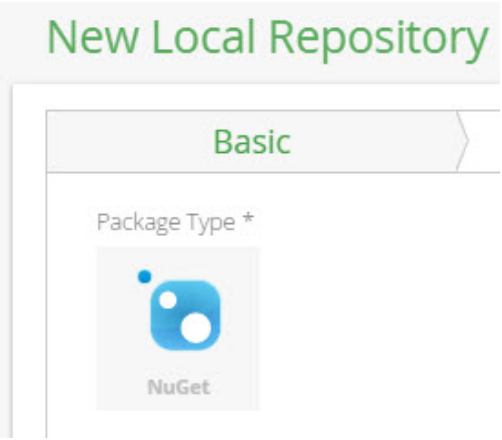
Metadata updates

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".

Configuration

Local Repositories

To create a local repository for which Artifactory will calculate NuGet package metadata, set **NuGet** to be the **Package Type.**



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.

Page Contents

Read more

- Microsoft Symbol Server

Integration Benefits

JFrog Artifactory and NuGet Repositories

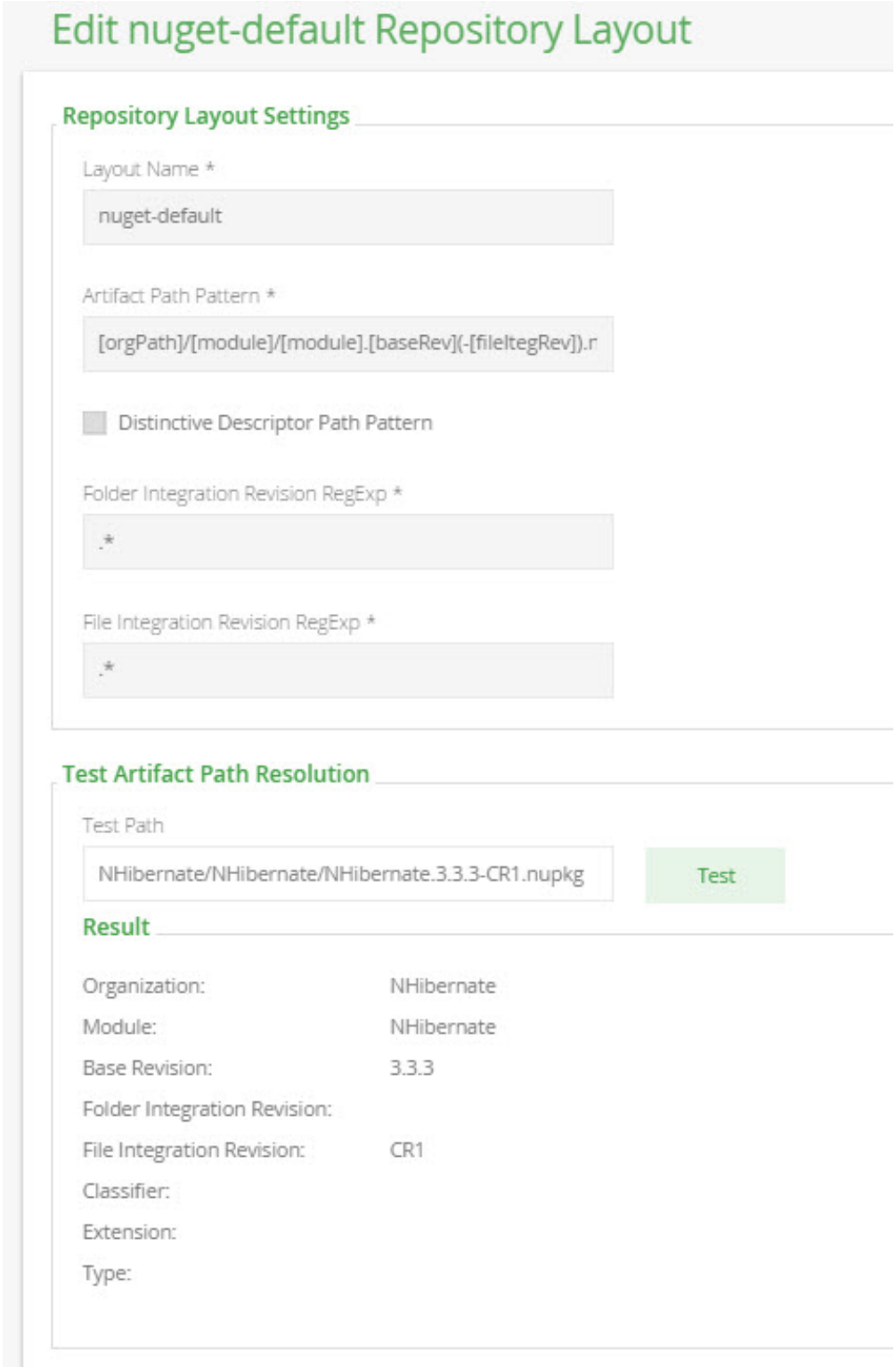
Placing packages to match your repository layout

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`.



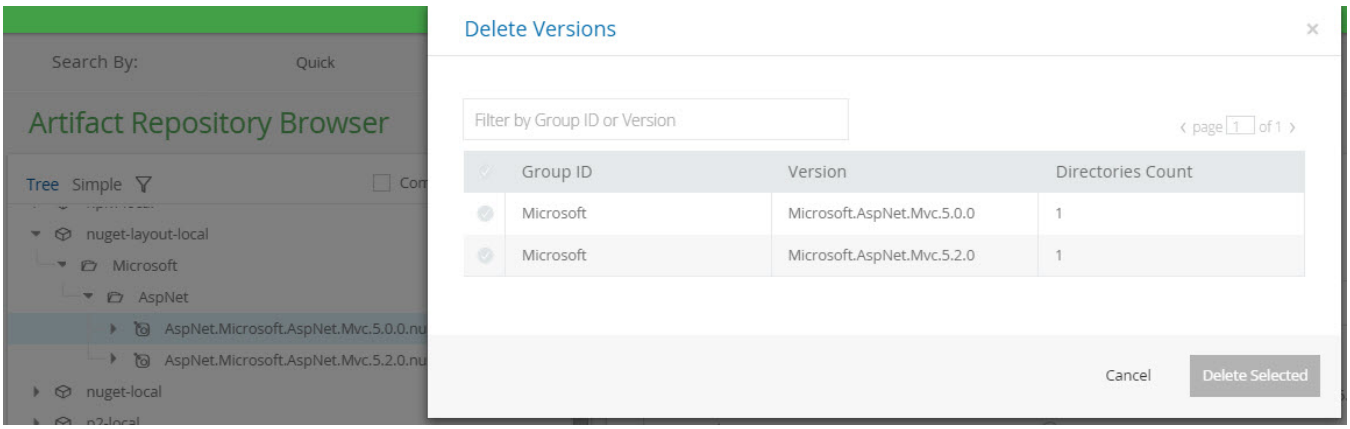
In this example, the three fields that are mandatory for module identification are:

- 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:

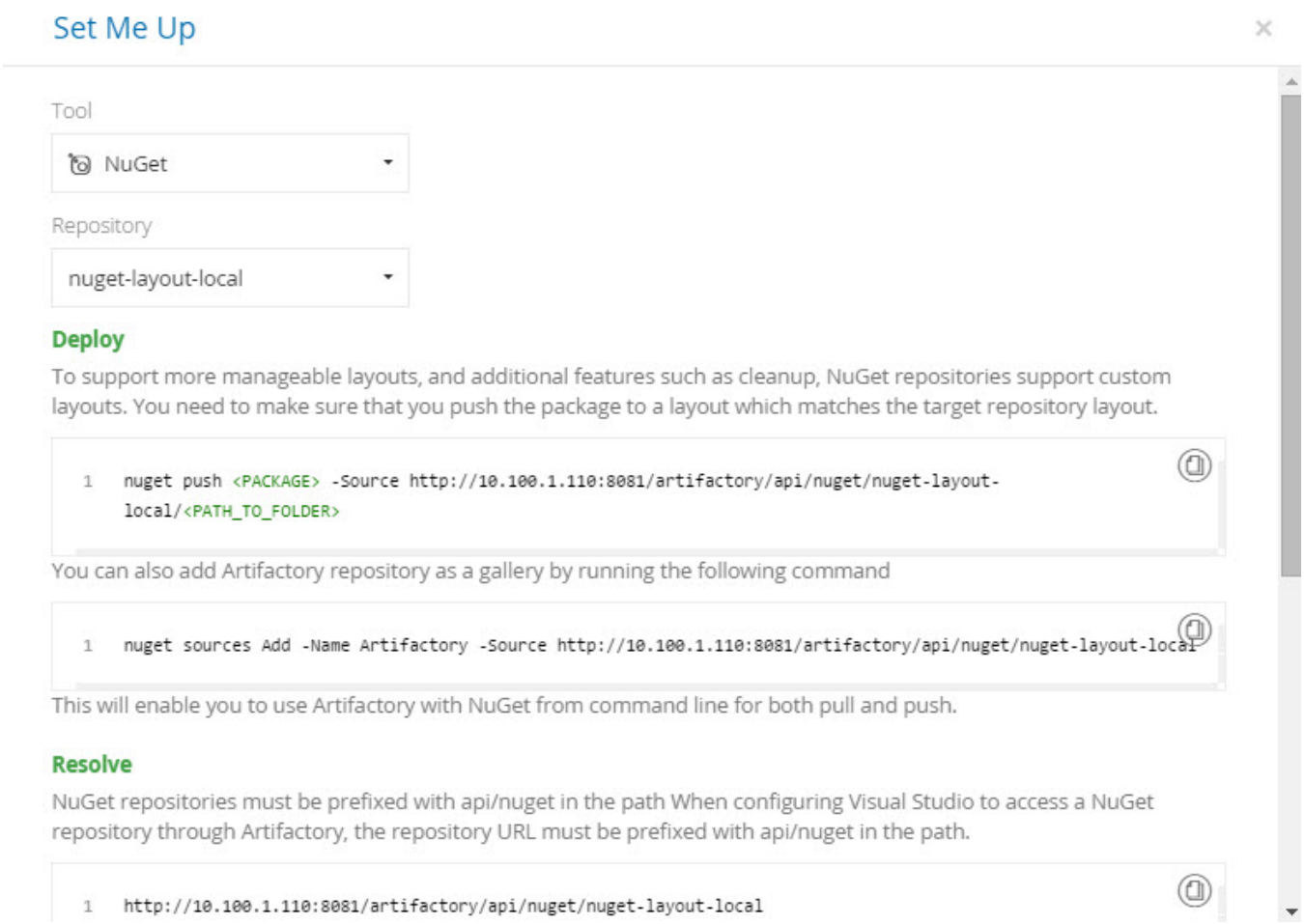
```
<repoLayout>
        <name>nuget-default</name>
    <artifactPathPattern>[orgPath]/[module]/[module].[baseRev](-[fileItegRev]).[ext]</artifactPathPattern>
    <distinctiveDescriptorPathPattern>false</distinctiveDescriptorPathPattern>
    <folderIntegrationRevisionRegExp>.*</folderIntegrationRevisionRegExp>
    <fileIntegrationRevisionRegExp>.*</fileIntegrationRevisionRegExp>
</repoLayout>
```

Since the package layout is in a corresponding folder hierarchy, the Artifactory Version Cleanup tool correctly detects previously installed versions.



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.



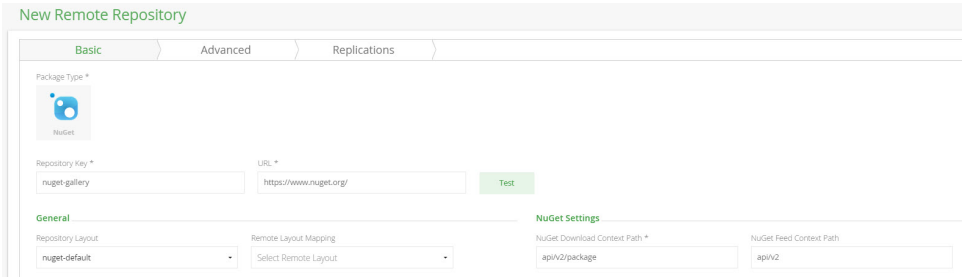
Remote Repositories

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:

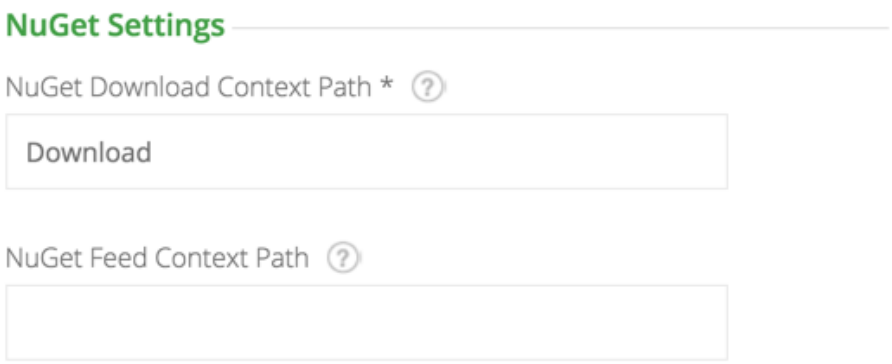
- The NuGet gallery exposes its feed resource at https://www.nuget.org/api/v2/Packages and its download resource at https://www.nuget.org/api/v2/package
  Therefore, to define this as a new repository you should set the repository **URL** to *`https://www.nuget.org`*, its **Feed Context Path** to *`api/v2`* and the **Download Context Path** to *`api/v2/package`*.



- 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`.

- For NuGet V3 API and metadata configuration it is necessary to configure the **NuGet V3 Feed http://host:port/api/v3/nuget/repoKey** while the URL can stay with http://host:port/artifactory/api/nuget/repoKey/

- Another Artifactory repository (i.e. a Smart Remote NuGet 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*, like this:



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:

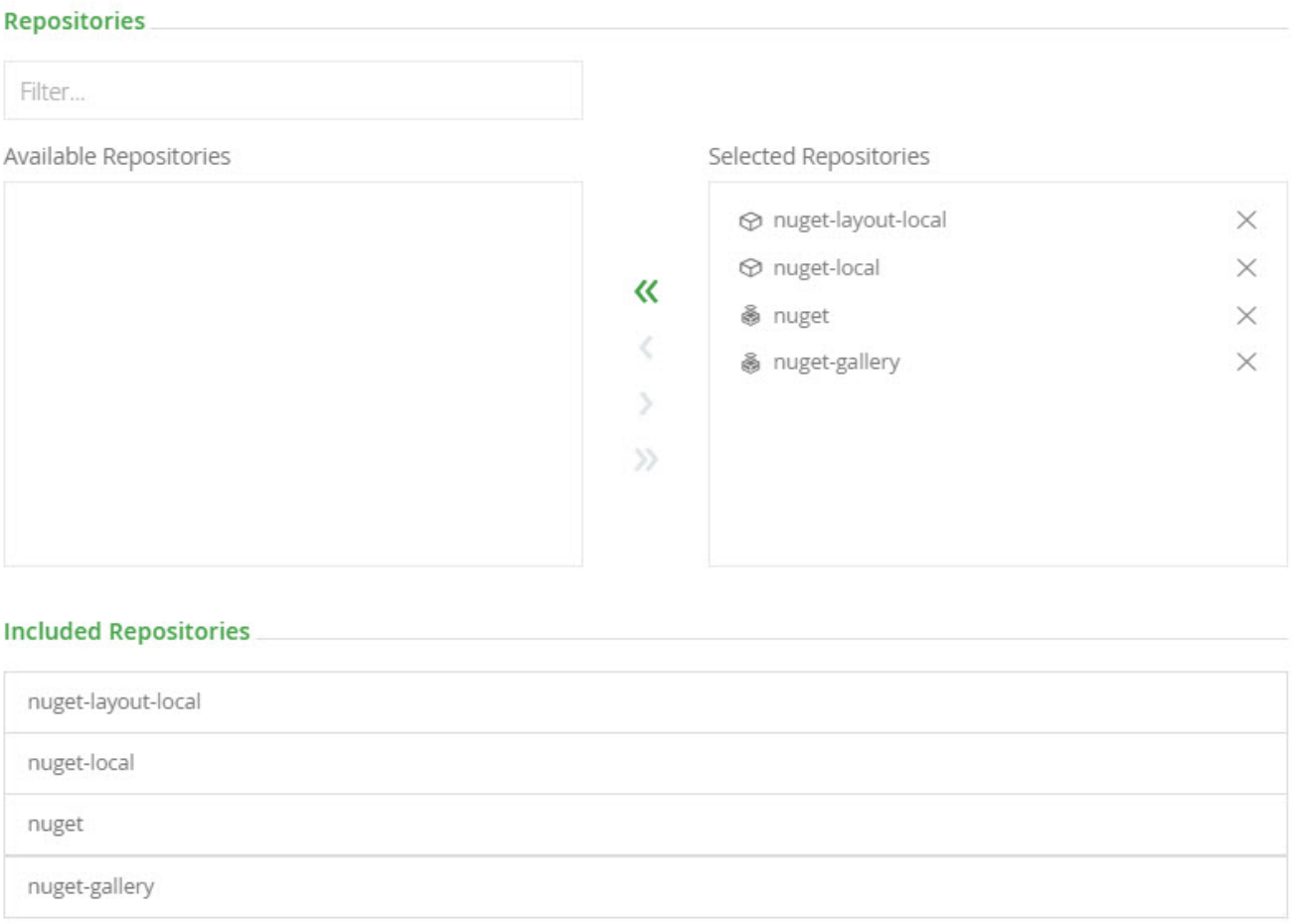
1. *.nuget.org
2. az320820.vo.msecnd.net (NuGet Gallery's current CDN domain)

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 NuGet repository set **NuGet** to be its **Package Type,** and select the underlying local and remote NuGet repositories to include under the **Repositories** section.



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 Artfactory, 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>*

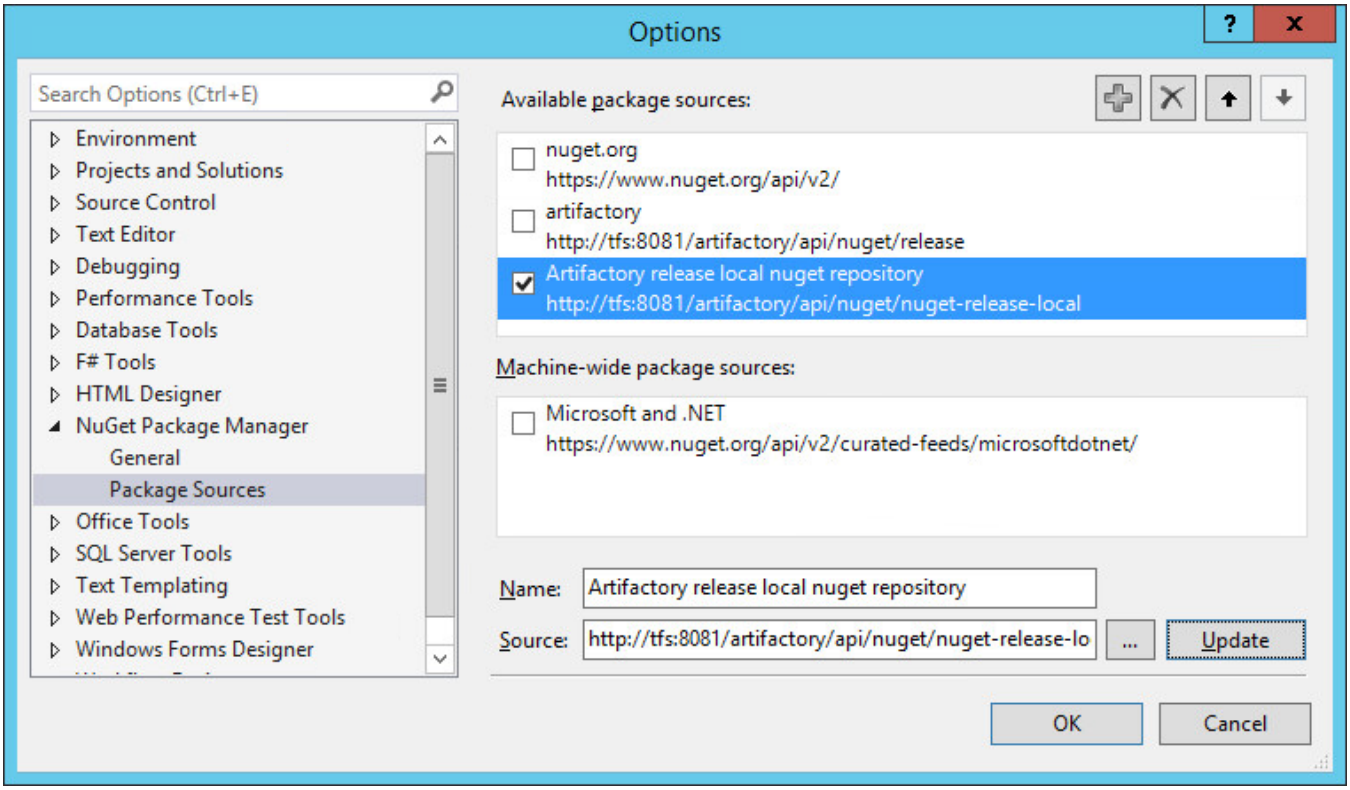
Or, if you are using Artifactory SaaS the URL would be:

*https://<server name>.jfrog.io/<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 use V3 requests, you need to Configure Visual Studio with NuGet v3 API.

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 Artfactory, 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 SaaS the URL would be:
>
> *https://<server name>.jfrog.io/<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>
```

To use V3 requests, you need to Configure NuGet CLI with NuGet v3 API.

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>
```

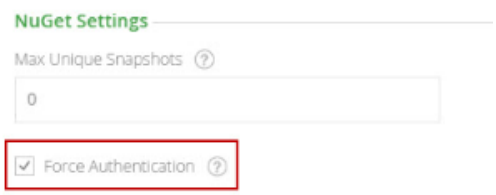
⚠ *NuGet.Config* file can also be placed in your project directory, for further information please refer to NuGet Configuration File

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.

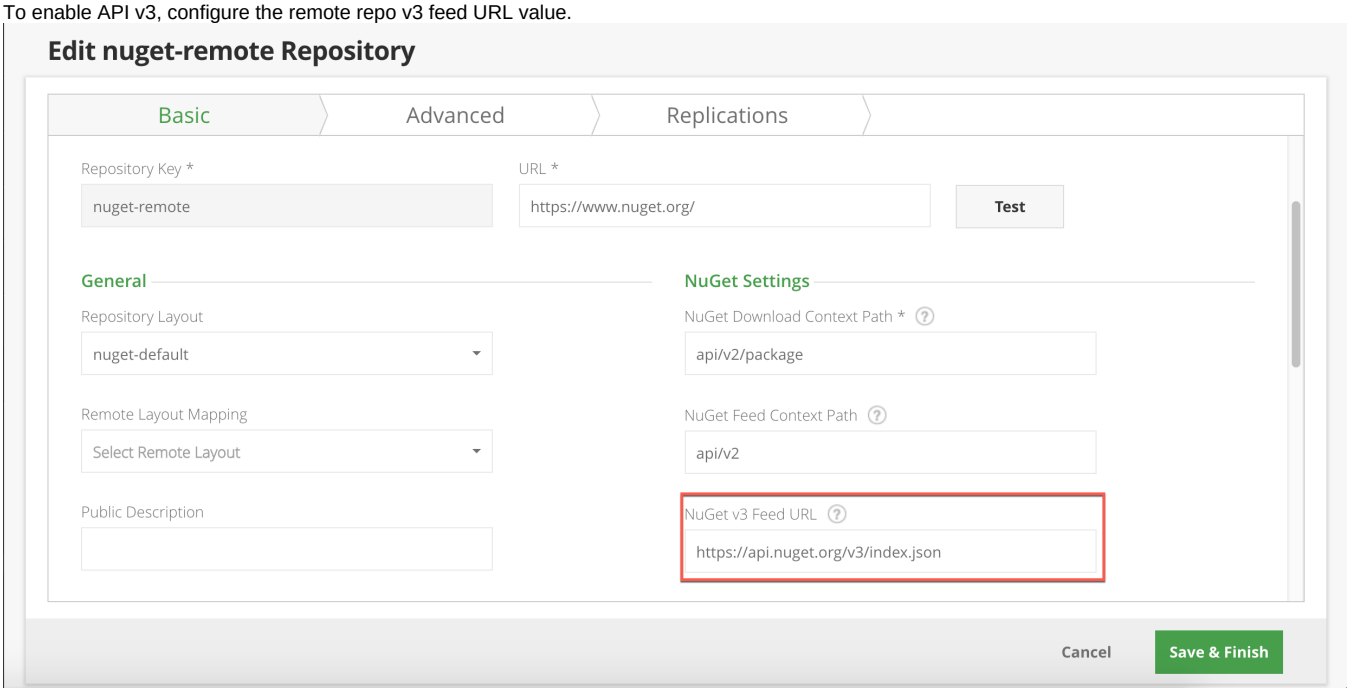


When set, Artifactory will first request authentication parameters from the NuGet client before trying to access this repository.

NuGet API v3 Registry Support

Artifactory now supports NuGet API v3 feeds and allows you to proxy remote NuGet API v3 repositories (e.g. the NuGet gallery) and other remote repositories that are set up with the API v3 feed.

To enable API v3, configure the remote repo v3 feed URL value.



Configure NuGet CLI/ Visual Studio to Work with NuGet v3 API

Manually add the **protocolVersion="3"** attribute to the NuGet.Config file:

- For Linux installation: The file is located under *~/.config/NuGet/NuGet.Config*
- For a Windows installation: Locate the file usually under *%appdata%\NuGet\NuGet.Config* and add "**/v3/**" to the source URL.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<packageSources>
    <add key="ArtifactoryV3Remote" value="http://artifactory:8081/artifactory/api/nuget/v3/nuget-remote"
protocolVersion="3" />
</packageSources>
...
...
</packageSourceCredentials>
</configuration>
```

NuGet SemVer 2.0 Package Support

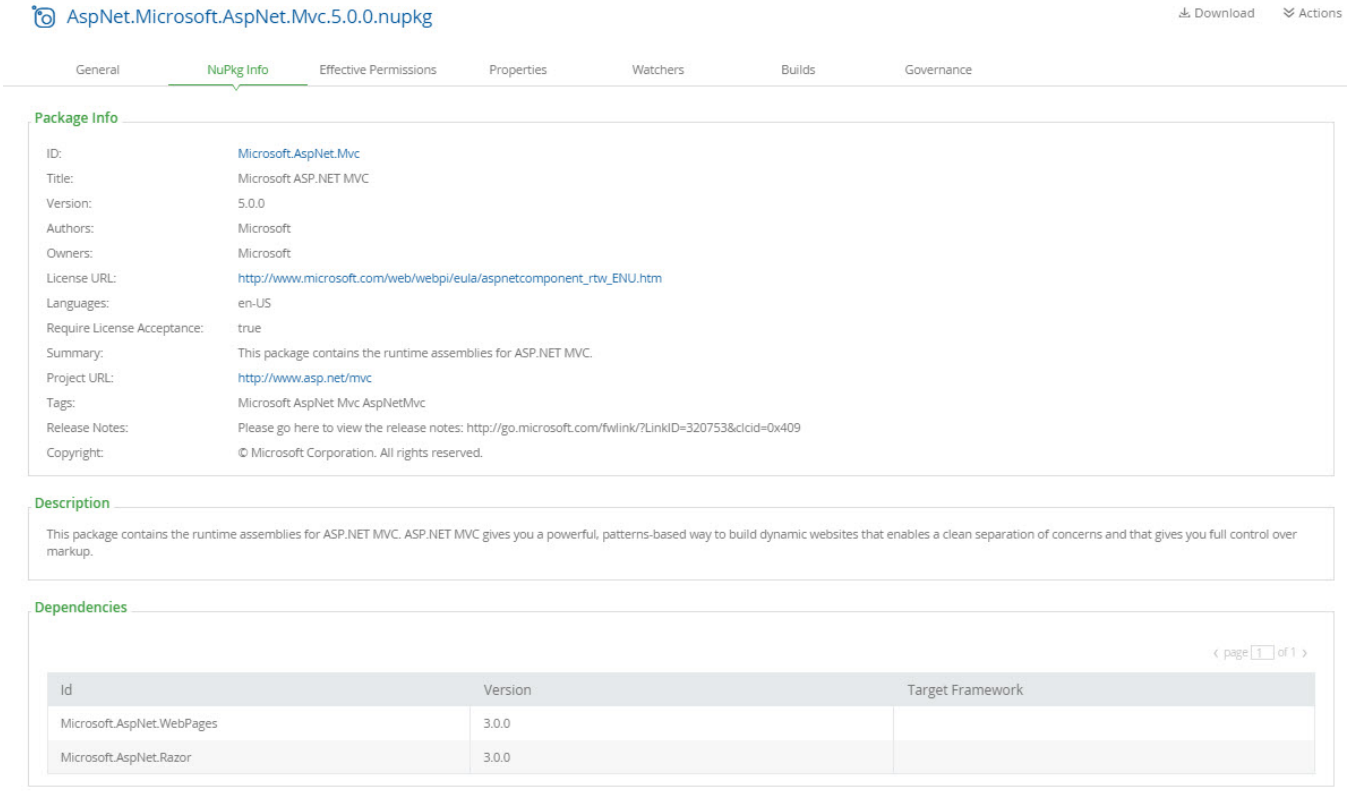
Artifactory now supports SemVer 2.0 rules for NuGet repositories (for both NuGet API v2 and API v3), which means you can now use pre-release numbers with dot notation or add metadata to the version, for example: *MyApp.3.0.0-build.60, MyApp.1.0+git.52406.*

Artifactory serves the requests for downloading packages in SemVer 2.0 rules. For example, if the latest version for a certain package is in SemVer 2.0 convention, Artifactory will return it to the client. NuGet packages with the SemVer 2.0 convention are served from local, remote, and virtual repositories and for both NuGet API v2, and v3 feeds.

NuGet packages with SemVer2 are not available for old NuGet clients (prior to version 4.3.0). This is a breaking change that was made to align with the official global repository behaviour. To retain the old behaviour, use the artifactory.*nuget.disableSemVer2SearchFilterForLocalRepos* flag.

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:



NuGet Build Information

You may store exhaustive build information in Artifactory by running your NuGet builds with JFrog CLI.

JFrog CLI collects build-info from your build agents and then publishes it to Artifactory. Once published, the build info can be viewed in the **Build Browser** under **Builds**.

For more details on NuGet build integration using JFrog CLI, please refer to Building NuGet Packages in the JFrog CLI User Guide

Watch the Screencast