Stac Documentation

Release 1.1.0

SmarterTravel

April 04, 2016

Stac is a tiny Artifactory client designed for getting the most recent version (or versions) of a project from an Artifactory server. The target use case is downloading project artifacts as part of a deploy process.

Given a few pieces of information, it can generate URLs to the most recent version of an artifact to be downloaded as part of your deploy process. Currently, only Maven repository layouts (in Artifactory parlance) are supported.

CHAPTER 1

Installation

To install Stac, simply run:

```
$ pip install stac
```

CHAPTER 2

Dependencies

- requests by Kenneth Reitz

CHAPTER 3

Usage

Using Stac is easy!

```
>>> from stac.api import new_maven_client
>>> client = new_maven_client('https://www.example.com/artifactory', 'libs-release')
>>> version = client.get_latest_version('com.example.services.authentication')
>>> version
'1.2.3'
>>> url = client.get_version_url('com.example.services.authentication', 'jar', version)
>>> url
'https://www.example.com/artifactory/libs-release/com/example/services/authentication/1.2.3/authentic
```

CHAPTER 4

Contents

4.1 Quickstart

The following examples will walk through some simple uses of the *stac.client.ArtifactoryClient* implementation for Maven repository layouts (*stac.client.GenericArtifactoryClient* paired with *stac.client.MavenArtifactUrlGenerator*). Since this client is focused on deploy related use cases, the examples below will as well.

4.1.1 Get a Specific Version

If you already know what version you want to download / deploy, Stac can turn that version number into a URL for you based on the name of your project and repository.

```
import stac.api

client = stac.api.new_maven_client('https://www.example.com/artifactory', 'libs-release')
war = client.get_version('com.example.services.users', 'war', '1.2.4')
print(war) # 'https://www.example.com/artifactory/libs-release/com/example/services/users/1.2.4/user
```

In the example above, the `war` variable will be the full URL to download version 1.2.4 of some hypothetical users service.

4.1.2 Get the Latest Release Version

If you want to get the most recent release version of a project, Stac can determine that based on the name of your project and repository.

```
import stac.api

client = stac.api.new_maven_client('https://www.example.com/artifactory', 'libs-release')

version = client.get_latest_version('com.example.services.auth')
print(version) # '1.3.0'

jar = client.get_version_url('com.example.services.auth', 'jar', version)
print(jar) # 'https://www.example.com/artifactory/libs-release/com/example/services/auth/1.3.0/auth-
```

In the example above, `version` will be the string `1.3.0` and the `jar` variable will be the full URL to download version 1.3.0 (the most recent release) of some hypothetical authentication service. If there haven't been any releases of this service an exception will be raised.

4.1.3 Get the Latest Snapshot Version

Maybe you want to determine the most recent snapshot version of your project (potentially to deploy it to your continuous integration server). Stac can also help you out with this. However, you need to explicitly tell the client that you are looking for snapshot, or integration, versions.

```
import stac.api

client = stac.api.new_maven_client('https://www.example.com/artifactory', 'libs-snapshot', is_snapsho

version = client.get_latest_version('com.example.services.locations')
print(version) # '4.1.0-SNAPSHOT'

jar = client.get_version_url('com.example.services.locations', 'jar', version)
print(jar) # 'https://www.example.com/artifactory/libs-snapshot/com/example/services/locations/4.1.0
```

In the example above, `version` will be the string `4.1.0-SNAPSHOT` and the `jar` variable will be the full URL to download version 4.1.0 (the most recent snapshot / integration version) of some hypothetical location service. If there haven't been any snapshot versions of this service created, an exception will be raised.

4.1.4 Get the Latest N Release Versions

If you need to get more than a single most recent release version, the process is outlined below (spoiler: it's pretty much the same as getting the single most recent release version).

```
import stac.api

client = stac.api.new_maven_client('https://www.example.com/artifactory', 'libs-release')

versions = client.get_latest_versions('com.example.services.auth', limit=3)
print(versions)
# [
#   '1.3.0',
#   '1.2.8',
#   '1.2.3'
# ]

jars = [client.get_version_url('com.example.services.auth', 'jar', version) for version in versions]
print(jars)
# [
#   'https://www.example.com/artifactory/libs-release/com/example/services/auth/1.3.0/auth-1.3.0.jar
#   'https://www.example.com/artifactory/libs-release/com/example/services/auth/1.2.8/auth-1.2.8.jar
#   'https://www.example.com/artifactory/libs-release/com/example/services/auth/1.2.3/auth-1.2.3.jar
# ]
```

As you can see, the `versions` variable will contain the version numbers of the three most recent releases and the `jars` variable is the the full URL to each of the respective releases, ordered with the most recent version first.

4.1.5 Get the Latest N Snapshot Versions

If you need to get more than a single most recent snapshot version, the process is outlined below (you might have guessed: it's pretty much the same as getting the single most recent snapshot version). This differs from getting the most recent N release versions because you must tell the client you are explicitly looking for snapshot versions.

```
import stac.api
```

```
client = stac.api.new_maven_client('https://www.example.com/artifactory', 'libs-snapshot', is_snapsho

versions = client.get_latest_versions('com.example.services.locations', limit=3)
print(versions)
# [
#   '4.1.0-SNAPSHOT',
#   '4.0.0-SNAPSHOT',
#   '3.12.0-SNAPSHOT'
# ]

jars = [client.get_version_url('com.example.services.locations', 'jar', version) for version in versi
print(jars)
# [
#   'https://www.example.com/artifactory/libs-snapshot/com/example/services/locations/4.1.0-SNAPSHOT/
#   'https://www.example.com/artifactory/libs-snapshot/com/example/services/locations/4.0.0-SNAPSHOT/
#   'https://www.example.com/artifactory/libs-snapshot/com/example/services/locations/3.12.0-SNAPSHOT
# ]
```

As you can see, the `versions` variable will contain the versions numbers of the three most recent snapshots and the `jars` variable is the to each of the respective releases, ordered with the most recent version.

4.2 Advanced Usage

Some more advanced or non-typical usages of Stac will be outlined below.

4.2.1 Search Remote Repositories

If the repository you're using is a virtual repository and you want to find the latest version of an artifact in one of the repositories being mirrored by it, you'll need to tell the Stac client that it should be searching them. Luckily, this is pretty easy.

```
import stac.api

client = stac.api.new_maven_client('https://internal.example.com/artifactory', 'libs-release')
version = client.get_latest_version('com.example.services.locations', remote=True)
print(version) # '4.0.5'
```

4.2.2 Use HTTP Authentication

You might have noticed we aren't using authentication to access the Artifactory API anywhere in the Quickstart section. If you've set up your Artifactory API (and the artifacts contained within it) to require authentication, this is fairly easy to work with in Stac.

```
import stac.api

client = stac.api.new_maven_client(
    'https://internal.example.com/artifactory', 'libs-release', username="deploy", password="authIs4
version = client.get_latest_version('com.example.services.ads')
print(version) # '5.4.1'
```

4.2.3 Use a Custom HTTP Session

Stac uses the Requests library for making HTTP requests (if you aren't familiar with Requests, check it out, it's awesome). In most cases, Stac will create a new `requests.Session` object when a client is created and you really shouldn't need to worry about this detail. However, if you've got special requirements (maybe you need to disable certificate validation or something) you can supply your own `requests.Session` object to the client.

Doing this is a little more involved than just creating a standard client but it's still not *that* bad.

```
import requests
import stac.api

# Create a custom session object...
session = requests.Session()
# And configure it
session.verify = False

# Construct the configuration for the client
client_config = stac.api.GenericArtifactoryClientConfig()
client_config.http_dao = stac.api.VersionApiDao(session, 'https://repo.example.com/artifactory', 'lib
client_config.url_generator = stac.api.MavenArtifactUrlGenerator('https://repo.example.com/artifactor

# Create the client instance
client = stac.api.GenericArtifactoryClient(client_config)

# Use it as normal
version = client.get_latest_version('com.example.services.locations')
print(version) # '4.0.5'
```

4.2.4 Get Custom Assemblies

At Smarter Travel, when we build and release an application jar to Artifactory, we also release a few associated jars at the same time. Source code, documentation, and runtime configuration are typically built and released at the same time. In Maven terms, these are known as "assemblies". Stac has support for finding these assemblies by passing the `descriptor='blah'` argument to the desired method. An example is given below.

```
import stac.api

client = stac.api.new_maven_client('https://www.example.com/artifactory', 'libs-release')

version = client.get_latest_version('com.example.services.mail')
print(version) # '9.2.1'

source_jar = client.get_version_url('com.example.services.mail', 'jar', version, descriptor='sources
print(source_jar) # 'https://www.example.com/artifactory/libs-release/com/example/services/mail/9.2..

config_jar = client.get_version_url('com.example.services.mail', 'jar', version, descriptor='config')
print(config_jar) # 'https://www.example.com/artifactory/libs-release/com/example/services/mail/9.2..
```

As you can see, we were able to find the most recent version of the source code and configuration associated with a hypothetical mail service.

4.3 API

This section of the documentation covers the public interfaces of Stac.

Note: When using the library, always make sure to access classes and functions through the `stac.api` module, not each individual module.

4.3.1 Clients

The classes and functions in the `stac.client` module make up the main interface to the Stac library. Unless you're doing something non-typical, this is probably all you need to worry about.

class `stac.client.`**`ArtifactoryClient`**

Interface for getting URLs and versions of artifacts.

How artifact names, packaging, and descriptors are interpreted is implementation specific and typically based on a particular repository layout. For example a Maven layout based client would use `full_name` for the full group and artifact (e.g. 'com.example.project.service'). While a Python layout based client would use `full_name` as unique name in a flat namespace (e.g 'my-project').

class `stac.client.`**`GenericArtifactoryClient`**(*config*)

Artifactory client for use with multiple different repository layouts.

Different `ArtifactUrlGenerator` implementations can be used with this client to support different repository layouts. The logic within this client should be relatively layout agnostic.

This class is thread safe.

`__init__`(*config*)

Create a new generic client instance based on the supplied configuration.

Parameters **`config`** (`GenericArtifactoryClientConfig`) – Required configuration for this client

`get_latest_version`(*full_name*, *remote=False*)

Get the most recent version of the given project.

The name of the artifact should be composed of the group ID and artifact ID (if available). E.g. "com.example.project.service". Depending on the repository layout, the `full_name` might only be the artifact name.

Example usage:

```
>>> client = new_maven_client('https://www.example.com/artifactory', 'libs-release')
>>> client.get_latest_version('com.example.users.service')
'1.5.0'
```

The example above returns the latest version of a hypothetical user service, 1.5.0.

This method makes a single network request.

Parameters

- **`full_name`** (*`str`*) – Fully qualified name of the artifact to get the version of.
- **`remote`** (*`bool`*) – Should remote repositories be searched to find the latest version (for example if the repository being checked is a virtual repository)? Note that this can make the search much slower. The default is not to check remote repositories.

Returns Version number of the latest version of the artifact

Return type str

Raises *stac.exceptions.NoMatchingVersionsError* – If no matching artifact could be found

get_latest_versions (*full_name*, *remote=False*, *limit=5*)

Get the most recent versions of the given project, ordered most recent to least recent.

The name of the artifact should be composed of the group ID and artifact ID (if available). E.g. "com.example.project.service". Depending on the repository layout, the `full_name` might only be the artifact name.

Example usage:

```
>>> client = new_maven_client('https://www.example.com/artifactory', 'libs-release')
>>> client.get_latest_versions('com.example.auth.service', limit=3)
['1.6.0', '1.5.4', '1.5.3']
```

The example above would return a list of the three most recent versions of some hypothetical authentication service.

This method makes a single network request.

Parameters

- **full_name** (*str*) – Full qualified name of the artifacts to get the versions of.
- **remote** (*bool*) – Should remote repositories be searched to find the latest versions (for example if the repository being checked is a virtual repository)? Note that this can make the search much slower. The default is not to check remote repositories.
- **limit** (*int*) – Only get the `limit` most recent versions.

Returns Most recent versions of the artifact with the given name, ordered with most recent first.

Return type list

Raises

- **ValueError** – If limit is negative or zero
- *stac.exceptions.NoMatchingVersionsError* – If no matching artifact could be found

get_version_url (*full_name*, *packaging*, *version*, *descriptor=None*)

Get the URL to a specific version of the given project, optionally using a descriptor to get a particular variant of the version (sources, javadocs, etc.).

The name of the artifact should be composed of the group ID and artifact ID (if available). E.g. "com.example.project.service". Depending on the repository layout, the `full_name` might only be the artifact name.

Packaging should be the type of file used for the artifact, e.g. 'war', 'jar', 'pom', etc.

The descriptor may be used to select javadoc jars, sources jars, or any other assemblies created as part of the version of the artifact.

Example usage:

```
>>> client = new_maven_client('https://www.example.com/artifactory', 'libs-release')
>>> client.get_version_url('com.example.users.service', '1.4.5', 'jar', descriptor='sources'
'https://www.example.com/artifactory/libs-release/com/example/users/service/1.4.5/service-1.
```

The example above would return a path object for the sources jar of version 1.4.5 of some hypothetical user service.

This method does not make any network requests.

Parameters

- **`full_name`** (*`str`*) – Fully qualified name of the artifact to get the path of.
- **`packaging`** (*`str`*) – Type of packaging / file format used for the artifact
- **`version`** (*`str`*) – Version of the artifact to get the path of.
- **`descriptor`** (*`str`*) – Tag to get a particular variant of a release.

Returns URL to the artifact with given name and version

Return type str

class `stac.client.`**`GenericArtifactoryClientConfig`**

Configuration for construction of a new `GenericArtifactoryClient` instance.

`http_dao` **= None**

DAO for interacting with the Artifactory HTTP API.

`is_integration` **= None**

Does the repository we are searching against contain SNAPSHOT (a.k.a. integration) versions and thus require alternate API calls to determine the latest version? Default is false.

`url_generator` **= None**

URL generator for determining the URL to download an artifact.

class `stac.client.`**`ArtifactUrlGenerator`**

Interface for generating the URL to download a particular version of an artifact.

Implementations will typically be specific to a particular repository layout in Artifactory. I.e. there may be one URL generator for Maven repositories, another one for Python packages, and another for NPM modules.

class `stac.client.`**`MavenArtifactUrlGenerator`**(*base*, *repo*)

URL generator for use with Maven repositories.

`stac.client.`**`new_maven_client`**(*base_url*, *repo*, *is_snapshot=False*, *username=None*, *password=None*)

Get a new implementation of `ArtifactoryClient` for use with Maven repository layouts, optionally using the provided authentication.

Most users will simply call this method to get a new Maven client instance. For example:

```
>>> client = new_maven_client('https://www.example.com/artifactory', 'libs-release')
>>> latest = client.get_latest_version('com.example.users.service', 'war')
'1.6.0'
```

Parameters

- **`base_url`** (*`str`*) – URL to root of the Artifactory installation. Example, "https://artifactory.example.com/artifactory".
- **`repo`** (*`str`*) – Which repository should searches be done against. Example, "libs-release-local" or "libs-snapshot-local".
- **`is_snapshot`** (*`bool`*) – Does the repository to perform searches against contain SNAPSHOT (a.k.a. integration) versions? Default is `False`
- **`username`** (*`str`*) – Optional username for authentication when making API calls and downloading artifacts.
- **`password`** (*`str`*) – Optional password for authentication when making API calls and downloading artifacts.

Returns New Artifactory client for use with Maven repositories

Return type *GenericArtifactoryClient*

4.3.2 HTTP Dao

If you need to customize how the Stac library interacts with Artifactory over HTTP, the `stac.http` module probably has what you're looking for.

class `stac.http.`**`VersionApiDao`**(*session*, *base_url*, *repo*)

HTTP DAO to get one or multiple versions of a particular artifact.

This DAO interacts with the Artifactory API over HTTP or HTTPS.

This class is thread safe.

`__init__`(*session*, *base_url*, *repo*)

Set the factory for requests session and factory for API urls.

Parameters

- **`session`** (*`requests.Session`*) – Session for making HTTP requests to the Artifactory API. This session should be configured with any required credentials for accessing the API.
- **`base_url`** (*`str | unicode`*) – Base URL to the Artifactory installation
- **`repo`** (*`str | unicode`*) – Name of repository to search against.

`get_most_recent_release`(*group*, *artifact*, *remote=False*)

Get the version number of the most recent release (non-integration version) of a particular group and artifact combination.

Parameters

- **`group`** (*`str`*) – Group of the artifact to get the version of
- **`artifact`** (*`str`*) – Name of the artifact to get the version of
- **`remote`** (*`bool`*) – Should remote repositories be searched to find the latest version? Note this can make the request much slower. Default is false.

Returns Version number of the most recent release

Return type str

Raises **`requests.exceptions.HTTPError`** – For any non-success HTTP responses from the Artifactory API.

`get_most_recent_versions`(*group*, *artifact*, *limit*, *remote=False*, *integration=False*)

Get a list of the version numbers of the most recent artifacts (integration or non-integration), ordered by the version number, for a particular group and artifact combination.

Parameters

- **`group`** (*`str`*) – Group of the artifact to get versions of
- **`artifact`** (*`str`*) – Name of the artifact to get versions of
- **`limit`** (*`int`*) – Fetch only this many of the most recent releases
- **`remote`** (*`bool`*) – Should remote repositories be searched to find the latest versions? Note this can make the request much slower. Default is false.
- **`integration`** (*`bool`*) – If true, fetch only "integration versions", otherwise fetch only non-integration versions.

Returns Version numbers of the most recent artifacts

Return type list

Raises

- **`requests.exceptions.HTTPError`** – For any non-success HTTP responses from the Artifactory API.
- **`ValueError`** – If limit is 0 or negative.

4.3.3 Exceptions

class `stac.exceptions.`**`StacError`**
Base for exceptions raised by the Stac library

class `stac.exceptions.`**`NoMatchingVersionsError`**(**args*, ***kwargs*)
Raised when there is no version or versions matching given criteria

4.4 Changelog

4.4.1 1.1.0 - 2016-04-04

- Add optional parameter to *`stac.client.ArtifactoryClient`* and implementations to allow remote repositories to be searched for the latest version of an artifact.

4.4.2 1.0.1 - 2016-03-09

- Change *`stac.exceptions.NoMatchingVersionsError`* to be a subclass of the base *`stac.exceptions.StacError`* exception.
- Change `stac.util.get_logger()` to get the `stac` named logger.

4.4.3 1.0.0 - 2016-02-09

- This is the first stable release of Stac. From this point on, all breaking changes will only be made in major version releases. This release is functionally the same as the `0.3.1` release.

4.4.4 0.3.1 - 2016-01-25

- Fix instance where `GenericArtifactoryClient` would not correctly handle artifacts without a `.` in the name.

4.4.5 0.3.0 - 2015-12-24

- **Breaking change** - Rename `MavenArtifactoryClient` to `GenericArtifactoryClient` and move all Maven- specific logic to a URL generator class that can be injected into it. Users creating the client via `new_maven_client` shouldn't notice any changes.

4.4.6 0.2.0 - 2015-12-23

- **Breaking change** - `get_latest_version` and `get_latest_versions` methods in the client now return version numbers only. Callers can use the `get_version_url` method to construct artifact URLs if desired.

4.4.7 0.1.1 - 2015-12-22

- Gracefully handle the case when we are looking for the latest SNAPSHOT version but there have not been any integration deploys to a repository. Fixes #1.

4.4.8 0.1.0 - 2015-12-21

- Initial release

Index