simple-salesforce Documentation

Release 0.72.1

Nick Catalano, community contributors

Sep 09, 2021

Contents

1 Examples **3**

2 Record Management **5**

3 Queries **7**

4 Other Options **9**

5 Using Apex **11**

6 Additional Features **13**

7 API documentation **15**
7.1 simple_salesforce package 15

8 Release history **17**
8.1 v1.11.4 17
8.2 v1.11.3 17
8.3 v1.11.2 17
8.4 v1.11.1 17
8.5 v1.11.0 18
8.6 v1.10.1 18
8.7 v1.10.0 18
8.8 v1.0.0 18
8.9 v0.75 18
8.10 v0.72 19
8.11 v0.71 19
8.12 v0.70 19
8.13 v0.69 19
8.14 v0.68.2 19

9 Authors & License **21**

10 Indices and tables **23**

Simple Salesforce is a basic Salesforce.com REST API client built for Python 3.5, 3.6, 3.7 and 3.8. The goal is to provide a very low-level interface to the REST Resource and APEX API, returning a dictionary of the API JSON response.

You can find out more regarding the format of the results in the Official Salesforce.com REST API Documentation

Contents:

CHAPTER 1

Examples

There are two ways to gain access to Salesforce

The first is to simply pass the domain of your Salesforce instance and an access token straight to `Salesforce()`

For example:

```
from simple_salesforce import Salesforce
sf = Salesforce(instance='na1.salesforce.com', session_id='')
```

If you have the full URL of your instance (perhaps including the schema, as is included in the OAuth2 request process), you can pass that in instead using `instance_url`:

```
from simple_salesforce import Salesforce
sf = Salesforce(instance_url='https://na1.salesforce.com', session_id='')
```

There are also two means of authentication, one that uses username, password and security token and the other that uses IP filtering, username, password and organizationId

To login using the security token method, simply include the Salesforce method and pass in your Salesforce username, password and token (this is usually provided when you change your password):

```
from simple_salesforce import Salesforce
sf = Salesforce(username='myemail@example.com', password='password', security_token=
↪'token')
```

To login using IP-whitelist Organization ID method, simply use your Salesforce username, password and organizationId:

```
from simple_salesforce import Salesforce
sf = Salesforce(password='password', username='myemail@example.com', organizationId=
↪'OrgId')
```

If you'd like to enter a sandbox, simply add `domain='test'` to your `Salesforce()` call.

For example:

```
from simple_salesforce import Salesforce
sf = Salesforce(username='myemail@example.com.sandbox', password='password', security_
↪token='token', domain='test')
```

Note that specifying if you want to use a domain is only necessary if you are using the built-in username/password/security token authentication and is used exclusively during the authentication step.

If you'd like to keep track where your API calls are coming from, simply add `client_id='My App'` to your `Salesforce()` call.

```
from simple_salesforce import Salesforce
sf = Salesforce(username='myemail@example.com.sandbox', password='password', security_
↪token='token', client_id='My App', domain='test')
```

If you view the API calls in your Salesforce instance by Client Id it will be prefixed with `RestForce/`, for example `RestForce/My App`.

When instantiating a *Salesforce* object, it's also possible to include an instance of *requests.Session*. This is to allow for specialized session handling not otherwise exposed by simple_salesforce.

For example:

```
from simple_salesforce import Salesforce
import requests

session = requests.Session()
# manipulate the session instance (optional)
sf = Salesforce(
   username='user@example.com', password='password', organizationId='OrgId',
   session=session)
```

CHAPTER 2

Record Management

To create a new 'Contact' in Salesforce:

```
sf.Contact.create({'LastName':'Smith','Email':'example@example.com'})
```

This will return a dictionary such as `{u'errors': [], u'id': u'003e0000003GuNXAA0', u'success': True}`

To get a dictionary with all the information regarding that record, use:

```
contact = sf.Contact.get('003e0000003GuNXAA0')
```

To get a dictionary with all the information regarding that record, using a **custom** field that was defined as External ID:

```
contact = sf.Contact.get_by_custom_id('My_Custom_ID__c', '22')
```

To change that contact's last name from 'Smith' to 'Jones' and add a first name of 'John' use:

```
sf.Contact.update('003e0000003GuNXAA0',{'LastName': 'Jones', 'FirstName': 'John'})
```

To delete the contact:

```
sf.Contact.delete('003e0000003GuNXAA0')
```

To retrieve a list of deleted records between `2013-10-20` to `2013-10-29` (datetimes are required to be in UTC):

```
import pytz
import datetime
end = datetime.datetime.now(pytz.UTC)  # we need to use UTC as salesforce API␣
↪requires this!
sf.Contact.deleted(end - datetime.timedelta(days=10), end)
```

To retrieve a list of updated records between `2014-03-20` to `2014-03-22` (datetimes are required to be in UTC):

```
import pytz
import datetime
end = datetime.datetime.now(pytz.UTC) # we need to use UTC as salesforce API requires␣
↪this
sf.Contact.updated(end - datetime.timedelta(days=10), end)
```

Note that Update, Delete and Upsert actions return the associated Salesforce HTTP Status Code

Use the same format to create any record, including 'Account', 'Opportunity', and 'Lead'. Make sure to have all the required fields for any entry. The Salesforce API has all objects found under 'Reference -> Standard Objects' and the required fields can be found there.

CHAPTER 3

Queries

It's also possible to write select queries in Salesforce Object Query Language (SOQL) and search queries in Salesforce Object Search Language (SOSL).

SOQL queries are done via:

```
sf.query("SELECT Id, Email FROM Contact WHERE LastName = 'Jones'")
```

If, due to an especially large result, Salesforce adds a `nextRecordsUrl` to your query result, such as `"nextRecordsUrl" : "/services/data/v26.0/query/01gD0000002HU6KIAW-2000"`, you can pull the additional results with either the ID or the full URL (if using the full URL, you must pass 'True' as your second argument)

```
sf.query_more("01gD0000002HU6KIAW-2000")
sf.query_more("/services/data/v26.0/query/01gD0000002HU6KIAW-2000", True)
```

As a convenience, to retrieve all of the results in a single local method call use

```
sf.query_all("SELECT Id, Email FROM Contact WHERE LastName = 'Jones'")
```

While `query_all` materializes the whole result into a Python list, `query_all_iter` returns an iterator, which allows you to lazily process each element separately

```
data = sf.query_all_iter("SELECT Id, Email FROM Contact WHERE LastName = 'Jones'")
for row in data:
  process(row)
```

Values used in SOQL queries can be quoted and escaped using `format_soql`:

```
sf.query(format_soql("SELECT Id, Email FROM Contact WHERE LastName = {}", "Jones"))
sf.query(format_soql("SELECT Id, Email FROM Contact WHERE LastName = {last_name}",␣
↪last_name="Jones"))
sf.query(format_soql("SELECT Id, Email FROM Contact WHERE LastName IN {names}",␣
↪names=["Smith", "Jones"]))
```

To skip quoting and escaping for one value while still using the format string, use `:literal`:

```
sf.query(format_soql("SELECT Id, Email FROM Contact WHERE Income > {:literal}",
↪"USD100"))
```

To escape a substring used in a LIKE expression while being able to use % around it, use `:like`:

```
sf.query(format_soql("SELECT Id, Email FROM Contact WHERE Name LIKE '{:like}%'",
↪"Jones"))
```

SOSL queries are done via:

```
sf.search("FIND {Jones}")
```

There is also 'Quick Search', which inserts your query inside the {} in the SOSL syntax. Be careful, there is no escaping!

```
sf.quick_search("Jones")
```

Search and Quick Search return `None` if there are no records, otherwise they return a dictionary of search results.

More details about syntax is available on the Salesforce Query Language Documentation Developer Website

CHAPTER 4

Other Options

To insert or update (upsert) a record using an external ID, use:

```
sf.Contact.upsert('customExtIdField__c/11999',{'LastName': 'Smith','Email':
↪'smith@example.com'})
```

To format an external ID that could contain non-URL-safe characters, use:

```
external_id = format_external_id('customExtIdField__c', 'this/that & the other')
```

To retrieve basic metadata use:

```
sf.Contact.metadata()
```

To retrieve a description of the object, use:

```
sf.Contact.describe()
```

To retrieve a description of the record layout of an object by its record layout unique id, use:

```
sf.Contact.describe_layout('39wmxcw9r23r492')
```

To retrieve a list of top level description of instance metadata, user:

```
sf.describe()

for x in sf.describe()["sobjects"]:
  print x["label"]
```

CHAPTER 5

Using Apex

You can also use this library to call custom Apex methods:

```
payload = {
  "activity": [
    {"user": "12345", "action": "update page", "time": "2014-04-21T13:00:15Z"}
  ]
}
result = sf.apexecute('User/Activity', method='POST', data=payload)
```

This would call the endpoint `https://<instance>.salesforce.com/services/apexrest/User/Activity` with `data=` as the body content encoded with `json.dumps`

You can read more about Apex on the Force.com Apex Code Developer's Guide

CHAPTER 6

Additional Features

There are a few helper classes that are used internally and available to you.

Included in them are `SalesforceLogin`, which takes in a username, password, security token, optional version and optional domain and returns a tuple of `(session_id, sf_instance)` where *session_id* is the session ID to use for authentication to Salesforce and `sf_instance` is the domain of the instance of Salesforce to use for the session.

For example, to use SalesforceLogin for a sandbox account you'd use:

```
from simple_salesforce import SalesforceLogin
session_id, instance = SalesforceLogin(
    username='myemail@example.com.sandbox',
    password='password',
    security_token='token',
    domain='test')
```

Simply leave off the final domain if you do not wish to use a sandbox.

Also exposed is the `SFType` class, which is used internally by the `__getattr__()` method in the `Salesforce()` class and represents a specific SObject type. `SFType` requires `object_name` (i.e. `Contact`), `session_id` (an authentication ID), `sf_instance` (hostname of your Salesforce instance), and an optional `sf_version`

To add a Contact using the default version of the API you'd use:

```
from simple_salesforce import SFType
contact = SFType('Contact','sessionid','na1.salesforce.com')
contact.create({'LastName':'Smith','Email':'example@example.com'})
```

To use a proxy server between your client and the SalesForce endpoint, use the proxies argument when creating SalesForce object. The proxy argument is the same as what requests uses, a map of scheme to proxy URL:

```
proxies = {
  "http": "http://10.10.1.10:3128",
  "https": "http://10.10.1.10:1080",
}
SalesForce(instance='na1.salesforce.com', session_id='', proxies=proxies)
```

All results are returned as JSON converted OrderedDict to preserve order of keys from REST responses.

CHAPTER 7

API documentation

7.1 simple_salesforce package

7.1.1 Subpackages

simple_salesforce.tests package

Submodules

simple_salesforce.tests.test_api module

simple_salesforce.tests.test_bulk module

simple_salesforce.tests.test_format module

simple_salesforce.tests.test_login module

simple_salesforce.tests.test_util module

Module contents

7.1.2 Submodules

simple_salesforce.api module

simple_salesforce.bulk module

simple_salesforce.exceptions module

simple_salesforce.format module

simple_salesforce.login module

simple_salesforce.messages module

simple_salesforce.metadata module

simple_salesforce.util module

7.1.3 Module contents

CHAPTER 8

Release history

8.1 v1.11.4

Changes as of 2021-09-09

Other - [#494] Raise exception when bulk query returns a failure status - [#503] Fix lint warnings - [#497] Support non-standard https port number

8.2 v1.11.3

8.2.1 Bugs

- [431] Fix timezone handling

8.3 v1.11.2

8.3.1 Bugs

- [469] Fix bulk call results returning nested list

8.4 v1.11.1

8.4.1 Features

- [445] Added wrapper for Tooling API
- [451] Support JWT without file for private key

8.4.2 Bugs

- [454] Fixed typo in metadata file
- [443] Fix to prevent silent failure by preventing NaN in payload

8.5 v1.11.0

- [375] Added file based metadata deployment

8.6 v1.10.1

8.6.1 Other

- [405] Update readme to remove syntax error in bulk
- [394] Add format_soql and format_external_id functions
- [393] Updated readme for JWT authentication
- Update readme to remove python 3.3/3.4

8.7 v1.10.0

8.7.1 Features

- [316] Added support for bulk multi-batch processing records
- [349] Added support for bulk concurrency mode

8.8 v1.0.0

8.8.1 Other

- [362] Increased default Salesforce API Version to 42.0
- [360] Remove depreciated interfaces
- [358] Removed support for Python 2.6, 2.7, 3.3, and 3.4
- [359] Make the minimum version of requests v 2.22.0, allowing us to remove requests[security]
- Changed “Beta” classifier to “Production/Stable”

8.9 v0.75

8.9.1 Features

- [305] Support for JWT Bearer Token workflow

- [354] Ability to load large results lazily (query_all_iter)

8.10 v0.72

8.10.1 Bugs

- [134] query_all changed to be non-recursive due to recursion limit being hit for large result sets.

8.11 v0.71

8.11.1 Features

- [131] Added the ability to pass custom headers through to requests

8.12 v0.70

8.12.1 Features

- [98] Requests session objects are now shared between SFType instances and the parent Salesforce object

8.13 v0.69

8.13.1 Features

- [103] Require requests[secure] to allow for system openssl, which will allow for TLSv1.2 in Python < 2.7.9 assuming system openssl is newer than Python's.

8.13.2 Other

- Replaced httpretty with responses for unit tests due to SSL-related errors in httpretty.

8.14 v0.68.2

8.14.1 Other

- [85] Added tox support, updated travis-ci config
- Added CHANGES file
- [97] _call_salesforce is now used consistently across API

CHAPTER 9

Authors & License

This package is released under an open source Apache 2.0 license. Simple-Salesforce was originally written by Nick Catalano but most newer features and bugfixes come from community contributors. Pull requests submitted to the GitHub Repo are highly encouraged!

Authentication mechanisms were adapted from Dave Wingate's RestForce and licensed under a MIT license

The latest build status can be found at Travis CI

CHAPTER 10

Indices and tables

- genindex
- modindex
- search