Google Domain Shared Contacts Client Documentation

Release 0.1.0

Robert Joyal

Jun 07, 2020

Contents

Contents:

CHAPTER 1

Google Domain Shared Contacts Client

A Python client to CRUD Google Domain Shared Contacts

- Free software: MIT license
- Documentation: https://domain-shared-contacts-client.readthedocs.io.

1.1 Features

- Command-line client and Python library:

```
$ pip install domain_shared_contacts_client
$ domain_shared_contacts_client --help
Usage: domain_shared_contacts_client [OPTIONS] DOMAIN ADMIN CREDENTIALS
                                     COMMAND [ARGS]...

  Command line utility to interact with the Shared Contacts API.

Options:
  --help  Show this message and exit.

Commands:
  create  a single contact
  delete  a single contact
  list    all contacts
  read    a single contact
  update  a single contact
```

- **List, create, read, update and delete basic information for Google Domain Shared Contacts**

  - **name**
    - given_name
    - family_name
    - full_name
  - **organization**
    - name
    - department
    - title
    - job_description
    - symbol
  - **email (list)**
    - address
    - primary
  - **phone_number (list)**
    - text
    - primary
  - **structured_postal_address (list)**
    - street
    - city
    - region
    - postcode
    - country
    - primary

1.2 Limitations

- Python 2.7 only
- Updates a limited set of the ContactEntry attributes
- Email address, phone number and postal address types are hard-coded to gdata.data.WORK_REL

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

[~ Dependencies scanned by PyUp.io ~]

CHAPTER 2

Installation

2.1 Prerequisite

Install gdata-python-client.

```
$ git clone git@github.com:google/gdata-python-client.git
$ cd gdata-python-client
$ python setup.py install
```

2.2 Stable release

To install Google Domain Shared Contacts Client, run this command in your terminal:

```
$ pip install domain_shared_contacts_client
```

This is the preferred method to install Google Domain Shared Contacts Client, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.3 From sources

The sources for Google Domain Shared Contacts Client can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/rjoyal/domain_shared_contacts_client
```

Or download the tarball:

```
$ curl  -OL https://github.com/rjoyal/domain_shared_contacts_client/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

Use Google Domain Shared Contacts Client in a project to list, create, read, update and delete contacts:

```
import domain_shared_contacts_client
import json

# Create the client. Authenticates to the Google Data API using the service account␣
↪credentials provided
client = domain_shared_contacts_client.Client(
    domain='example.com',
    admin='admin@example.com',
    credentials='/path/to/credentials.json')

# Fetch the list of contacts currently available
contacts = client.get_contacts()
print json.dumps(contacts, default=contacts_helper.convert_contacts)
```

Fetch contacts using the CLI:

```
$ domain_shared_contacts_client \
    example.com \
    admin@example.com \
    /path/to/client_secret.json \
    list
```

Example new_contact.json:

```
{
    "name": {
        "given_name": "John",
        "family_name": "Doe",
        "full_name": "John Doe"
    },
    "email": [
        {
```

(continues on next page)

(continued from previous page)

```
            "address": "jdoe@somewhere.com",
            "primary": "true"
        }
    ],
    "organization": {
        "name": "Example Pty Ltd",
        "title": "Senior Manager",
        "job_description": "Manage development activities",
        "department": "Software Engineering",
        "symbol": "EXPL"
    },
    "phone_number": [
        {
            "text": "(888)555-1212",
            "primary": "true"
        }
    ],
    "structured_postal_address": [
        {
            "street": "1 Example St.",
            "city": "Springfield",
            "region": "IL",
            "postcode": "62701",
            "country": "United States",
            "primary": "true"
        }
    ]
}
```

Create a new contact:

```
saved_contact = client.create_contact('/path/to/new_contact.json')
print json.dumps(saved_contact)
```

Create a new contact using the CLI:

```
$ domain_shared_contacts_client \
    example.com \
    admin@example.com \
    /path/to/client_secret.json \
    create /path/to/new_contact.json
{"id": "http://www.google.com/m8/feeds/contacts/example.com/base/2ba2136d0e101978"}
```

Read a contact:

```
contact = client.read_contact(saved_contact['id'])
print json.dumps(contact, default=contacts_helper.convert_contact)
```

Read a contact using the CLI:

```
$ domain_shared_contacts_client \
    example.com \
    admin@example.com \
    /path/to/client_secret.json \
    read 2ba2136d0e101978
```

Example update_contact.json:

```
{
    "name": {
        "given_name": "Joe",
        "full_name": "Joe Doe"
    },
    "email": [
        {
            "address": "jdoe@somewhereelse.com",
            "primary": "true"
        },
        {
            "address": "jdoe@somewhere.com",
            "primary": "false"
        }
    ]
}
```

Update a contact:

```
contact = client.update_contact(saved_contact['id'], '/path/to/updated_contact.json')
print json.dumps(contact, default=contacts_helper.convert_contact)
```

Update a contact using the CLI:

```
$ domain_shared_contacts_client \
    example.com \
    admin@example.com \
    /path/to/client_secret.json \
    update /path/to/updated_contact.json
```

Delete a contact:

```
result = client.delete_contact(saved_contact['id'])
print json.dumps(result)
```

Delete a contact using the CLI:

```
$ domain_shared_contacts_client
    example.com \
    admin@example.com \
    /path/to/client_secret.json \
    delete 2ba2136d0e101978
{"status": "OK"}
```

This package assumes the following:

- You have a Google Apps domain account
- You are able to login as the Domain Admin for the domain account
- You have created a Service Account and enabled G-Suite Domain-wide Delegation for that account
- You have created a key for the service account and downloaded it in JSON format
  - This will be provided in the 'credentials' parameter to instantiate a Client
- You have granted your service account authority to make API calls on your behalf
  - Go to the domain Admin Console

- Select Security from the list of controls. If you don't see Security listed, select More controls from the gray bar at the bottom of the page, then select Security from the list of controls. If you can't see the controls, make sure you're signed in as an administrator for the domain.
- Select Show more and then Advanced settings from the list of options.
- Select Manage API client access in the Authentication section.
- In the Client Name field enter the service account's Client ID. You can find your service account's client ID in the Service accounts page.
- In the One or More API Scopes field enter the list of scopes that your application should be granted access to. In our case, that is http://www.google.com/m8/feeds/contacts/
- Click Authorize
- Your application now has the authority to make API calls as users in your domain (to "impersonate" users).

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/rjoyal/domain_shared_contacts_client/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

Google Domain Shared Contacts Client could always use more documentation, whether as part of the official Google Domain Shared Contacts Client docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/rjoyal/domain_shared_contacts_client/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *domain_shared_contacts_client* for local development.

1. Fork the *domain_shared_contacts_client* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/domain_shared_contacts_client.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv domain_shared_contacts_client
$ cd domain_shared_contacts_client/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 domain_shared_contacts_client tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/rjoyal/domain_shared_contacts_client/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_domain_shared_contacts_client
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search