AggCat Documentation

Release 0.2

Glen Zangirolami

August 13, 2013

CONTENTS

Ready to get your financial data? This quickstart is intended to get you setup to start using Intuit's Customer Account Data API.

CHAPTER

ONE

BEFORE USING THE API CLIENT

You must take a few steps on Intuit's website before using the API client:

1. Create a development account and login.
2. Create a new application in the Customer Account Data category. It's easiest to follow the instructions in the help documentation.
3. Create and upload self a generated x509 certificate to your application. It's easiest to use openssl to generate the certificate. *It's best to name your key the same as your application name*. **Don't lose these certificates!**
4. Gather the login details and store them somewhere. You will need *OAuth Consumer Key*, *Oauth Consumer Secret*, *SAML Identity Provider ID*

CHAPTER

TWO

INSTALLATION

```
pip install python-aggcat
```

CHAPTER

THREE

INITIALIZING THE API CLIENT

Assuming you have an *OAuth Consumer Key*, *Oauth Consumer Secret*, *SAML Identity Provider ID*, and a path to the x509 certificates you generated you are ready to start querying:

```
from aggcat import AggcatClient

client = AggcatClient(
    'oauth_consumer_key',
    'oauth_consumer_secret',
    'saml_identity_provider_id',
    'customer_id',
    '/path/to/x509/appname.key'
)
```

Note: `customer_id` (Integer) It can be any integer. You should try using the database primary key of a user in your system or some other unique identifier such as a guid. If you are just testing you can use whatever integer you want.

`objectify` (Boolean) This is a BETA functionality. It will objectify the XML returned from intuit into standard python objects so you don't have to mess with XML. Default: `True`

CHAPTER

FOUR

QUERYING THE API

Here are a few sample queries that don't require you to add an account

4.1 Getting all institutions

```
institutions = client.get_institutions()
```

Note: This query will take a very long time depending on your internet connection. It returns 18000+ institutions in XML format. Sux :(

If you are using the `objectify = True` keyword argument on the client you can access the institutions in a pythonic way

```
>>> institutions = client.get_institutions()
>>> len(institutions)
18716
>>> institutions[0].institution_name
'Carolina Foothills FCU Credit Card'
```

4.2 Searching for your institution

Currently finding an institution is somewhat of a manual process. Soon, there will be a helper method on the client that will have a better search. Patches welcome ;). This example searches for an institution that contains "chase" in any of the XML elements:

```
from aggcat import AggcatClient
from lxml import etree
from aggcat.utils import remove_namespaces

client = AggcatClient(
    'oauth_consumer_key',
    'oauth_consumer_secret',
    'saml_identity_provider_id',
    'customer_id',
    '/path/to/x509/appname.key'
)

search_string = 'Chase'
institutions = client.get_institutions()
```

```
xml = etree.fromstring(institutions.to_xml())
xml = etree.fromstring(remove_namespaces(xml))

for element in xml.xpath('./institution[contains(., "chase")]'):
    id = element.xpath('./institutionId')[0].text
    name = element.xpath('./institutionName')[0].text
    print id, name

13278 JP Morgan Chase Bank
13640 Quicken Visa
14554 Chase Bank Credit Card (Amazon.com)
14910 Chase e-Funds Card
14777 Fox Chase Bank - Business Banking
13718 Fox Chase Bank
14484 Chevy Chase Bank - Web Cash Manager
...
```

Note: This query will take a very long time depending on your internet connection. It returns 18000+ institutions in XML format. Sux :(

4.3 Getting the institution details

From the previous search example, we can use 13728 to get the institution details

```
institution_details = client.get_institution_details(13278)

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><InstitutionDetail xmlns="http://schema.intuit
```

If you are using the `objectify = True` keyword argument on the client you can access the institution parameters in a Pythonic way

```
>>> institution = client.get_institution_details(13278)
>>> institution
<Institutiondetail object @ 0x110389990>
>>> institution.institution_name
'JP Morgan Chase Bank'
>>> institution.home_url
'https://www.chase.com/'
>>> institution.currency_code
'USD'
```

CHAPTER

FIVE

USER'S GUIDE

5.1 API

5.1.1 Initializing the client

class aggcat.**AggcatClient**(*consumer_key*, *consumer_secret*, *saml_identity_provider_id*, *customer_id*, *private_key*, *objectify=True*)

Intuit Customer Data API client

Parameters

- **consumer_key** (*string*) – The OAuth consumer key given on the Intuit application page
- **consumer_secret** (*string*) – The OAuth consumer secret given on the Intuit application page
- **saml_identity_provider_id** (*string*) – The SAML identitity provider id given on the Intuit application page
- **customer_id** (*integer*) – It can be any integer. Try using the database primary key of a user in your system or some other unique identifier such as a guid. If you are just testing you can use whatever integer you want.
- **private_key** (*string*) – The absolute path to the generated x509 private key
- **objectify** (*boolean*) – (optional) Convert XML into pythonic object on every API call. Default: `True`

Returns `AggcatClient`

Assuming you have an *OAuth Consumer Key*, *Oauth Consumer Secret*, *SAML Identity Provider ID*, and a path to the x509 certificates generated, you are ready to start querying:

```
from aggcat import AggcatClient

client = AggcatClient(
    'oauth_consumer_key',
    'oauth_consumer_secret',
    'saml_identity_provider_id',
    'customer_id',
    '/path/to/x509/appname.key'
)
```

Note: `customer_id` (Integer) It can be any integer. Try using the database primary key of a user in your system or some other unique identifier such as a guid. If you are just testing you can use whatever integer you want.

`objectify` (Boolean) This is a BETA functionality. It will objectify the XML returned from intuit into standard python objects so you don't have to mess with XML. Default: `True`

5.1.2 Authorization & updating credentials

Getting credential fields

At some point in time you will need to allow a user to login to an institution so you can get their account information. `get_credential_fields()` is not part of Intuit's API, but aids you in getting the fields you need to present in your UI layer.

To get credential fields use `get_credential_fields()`. It will give you the fields in a `list` format pre-ordered. It's a precursor to `discover_and_add_accounts()`

`AggcatClient.`**`get_credential_fields`**(*institution_id*)

Get a dictionary of fields required to login to a specific institution.

Parameters **institution_id** (*integer*) – The institution's id. See *Searching for your institution*.

Returns `list` of credentials to present.

```
>>> client.get_credential_fields(100000)
>>> [{'description': 'Banking Userid',
      'displayFlag': 'true',
      'displayOrder': '1',
      'instructions': 'Enter banking userid (demo)',
      'mask': 'false',
      'name': 'Banking Userid',
      'status': 'Active',
      'valueLengthMax': '20',
      'valueLengthMin': '1'},
     {'description': 'Banking Password',
      'displayFlag': 'true',
      'displayOrder': '2',
      'instructions': 'Enter banking password (go)',
      'mask': 'true',
      'name': 'Banking Password',
      'status': 'Active',
      'valueLengthMax': '20',
      'valueLengthMin': '1'}]
```

The fields that are returned are already ordered by the *displayOrder* field. The *name* field goes into the html input tags name attribute. The *mask* field will let you know if it's a password field. The list above might convert to this html:

```
<form method="POST" action="/login-to-bank/">
    <div>Enter banking userid (demo): <input type="text" name="Banking Userid" /></div>
    <div>Enter banking password (go): <input type="password" name="Banking Password" /></div>
    <input type="submit" value="Login to your bank" />
</form>
```

Authenticating and adding accounts

Once you have gathered the username and password for a specific institution you well need to `discover_and_add_accounts()`. This particular endpoint will either return a list of accounts if authentication works or return a list of challanges that need to be answered.

AggcatClient.**discover_and_add_accounts**(*institution_id*, ***credentials*)

Attempt to add the account with the credentials given

Parameters

- **institution_id** (*integer*) – The institution's id. See *Searching for your institution*.
- **credentials** (*dictionary*) – Credentials

Returns AccgatResponse

If a *challenge response is not required* it will look similar to the example below:

```
>>> client.get_credential_fields(100000)
>>> [{'description': 'Banking Userid',
      'displayFlag': 'true',
      'displayOrder': '1',
      'instructions': 'Enter banking userid (demo)',
      'mask': 'false',
      'name': 'Banking Userid',
      'status': 'Active',
      'valueLengthMax': '20',
      'valueLengthMin': '1'},
     {'description': 'Banking Password',
      'displayFlag': 'true',
      'displayOrder': '2',
      'instructions': 'Enter banking password (go)',
      'mask': 'true',
      'name': 'Banking Password',
      'status': 'Active',
      'valueLengthMax': '20',
      'valueLengthMin': '1'}]
>>> r = client.discover_and_add_accounts(100000, **{
    'Banking Userid': 'direct',
    'Banking Password': 'anyvalue',
   })
>>> r
<AggCatResponse 201>
>>> r.content
<Accountlist object [<Creditaccount object @ 0x10d12d790>,<Loanaccount object @ 0x10d12d810> ...
```

If a *challenge response is required* we get a different result:

```
>>> r = client.discover_and_add_accounts(100000, **{
    'Banking Userid': 'direct',
    'Banking Password': 'anyvalue',
   })
>>> r
<AggCatResponse 401>
>>> r.content
<Challenges object [<Challenge object @ 0x10d12d5d0>,<Challenge object @ 0x10d12d110> ...] @ 0x1
```

Notice that `r.content` is now a `Challenges` object. Listing the challenges will give to the questions that must be asked to the user:

```
>>> for c in r.content:
        print c.text
Enter your first pet's name:
Enter your high school's name:
```

You might convert this into an html form

```
<form action="/login/confirm-challenge" method="POST>
    <div>Enter your first pet's name: <input type="text" name="response_1" /></></div>
    <div>Enter your high school's name: <input type="text" name="response_2" /></></div>
</form>

<input type="submit" value="Confirm Challenges" />
```

Responding to a challenge

If you attempted to authenticate and received back a `Challenge` response the you will need to answer that challenge by sending your responses to `confirm_challenge()`

`AggcatClient.`**`confirm_challenge`**(*institution_id*, *challenge_session_id*, *challenge_node_id*, *responses*)

Post challenge responses and add accounts

Parameters

- **institution_id** (*integer*) – The institution's id. See *Searching for your institution*.
- **challenge_session_id** (*string*) – The session id received from *AggcatResponse.headers* in `discover_and_add_accounts()`
- **challenge_node_id** (*string*) – The session id received from *AggcatResponse.headers* in `discover_and_add_accounts()`
- **responses** (*list*) – A list of responses, ex. ['Cats Name', 'First High School']

Returns An `AggcatReponse` with attribute `content` being an `AccountList`

When using `discover_and_add_accounts()` you might get a challenge response:

```
>>> r = client.discover_and_add_accounts(100000, **{
    'Banking Userid': 'direct',
    'Banking Password': 'anyvalue',
   })
>>> r
<AggCatResponse 401>
>>> r.content
<Challenges object [<Challenge object @ 0x10d12d5d0>,<Challenge object @ 0x10d12d110> ...] @ 0x1
>>> r.headers
{'challengenodeid': '10.136.17.82',
 'challengesessionid': 'c31c8a55-754e-4252-8212-b8143270f84f',
 'connection': 'close',
 'content-length': '275',
 'content-type': 'application/xml',
 'date': 'Mon, 12 Aug 2013 03:15:42 GMT',
 'intuit_tid': 'e41418d4-7b77-401e-a158-12514b0d84e3',
 'server': 'Apache/2.2.22 (Unix)',
 'status': '401',
 'via': '1.1 ipp-gateway-ap02'}
```

- The `challenge_session_id` parameter comes from `r.headers['challengesessionid']`
- The `challenge_node_id` parameter comes from `r.headers['challengenodeid']`

Now confirm the challenge:

```
>>> challenge_session_id = r.headers['challengesessionid']
>>> challenge_node_id = r.headers['challengenodeid']
>>> responses = ['Black Cat', 'Meow High School']
>>> accounts = r.confirm_challenge(
        1000000,
        challenge_session_id,
        challenge_node_id,
        responses
    )
>>> accounts
<AggCatResponse 201>
>>> accounts.content
<Accountlist object [<Creditaccount object @ 0x10d12d790>,<Loanaccount object @ 0x10d12d810> ...
```

Updating outdated login credentials

If you have changed your institution credentials you will need to let Intuit know that it needs to use new credentials. For example, let's assume you have a Bank of America or JP Morgan Chase account and changed your password from the banking web site. At this point Intuit will need to be notified of this update.

`AggcatClient.`**`update_institution_login`**(*institution_id*, *login_id*, *refresh=True*, ***credentials*)

Update an instiutions login information

Parameters

- **institution_id** (*integer*) – The institution's id. See *Searching for your institution*.
- **login_id** (*string*) – Login id of the instiution. This can be retrieved from an account.
- **refresh** (*boolean*) – (optional) Setting this to `False` will only update the credentials in intuit and will not query against the instituion. This might cause issues because some institutions require you to re-answer the challenge questions. Default: `True`
- **credentials** (*dict*) – New credentials. See *Authorization & updating credentials*.

Returns `None` if update is valid or `AggcatResponse` if challenge occurs.

If a challenge response does not occur:

```
>>> r = ac.update_institution_login(100000, 83850162, **{
        'Banking Userid': 'direct',
        'Banking Password': 'anyvalue'
    })
>>> type(r)
NoneType
```

If a challenge response occurs:

```
>>> r = ac.update_institution_login(100000, 83850162, **{
        'Banking Userid': 'tfa_choice',
        'Banking Password': 'anyvalue'
    })
>>> r
<AggCatResponse 401>
>>> r.content
<Challenges object [<Challenge object @ 0x10d12d5d0>,<Challenge object @ 0x10d12d110> ...] @ 0x1
```

Notice that `r.content` is now a `Challenges` object. Listing the challenges will give to the questions that must be asked to the user:

```
>>> for c in r.content:
        print c.text
Enter your first pet's name:
Enter your high school's name:
```

You might convert this into an html form

```
<form action="/login/confirm-challenge" method="POST>
    <div>Enter your first pet's name: <input type="text" name="response_1" /></div>
    <div>Enter your high school's name: <input type="text" name="response_2" /></div>
</form>

<input type="submit" value="Confirm Challenges" />
```

Updating outdated challenge responses

AggcatClient.**update_challenge**(*login_id*, *challenge_session_id*, *challenge_node_id*, *responses*, *refresh=True*)

Update an instiutions challenge information

Parameters

- **login_id** (*string*) – Login id of the instiution. This can be retrieved from an account.
- **challenge_session_id** (*string*) – The session id received from *AggcatResponse.headers* in `discover_and_add_accounts()`
- **challenge_node_id** (*string*) – The session id received from *AggcatResponse.headers* in `discover_and_add_accounts()`
- **responses** (*list*) – A list of responses, ex. ['Cats Name', 'First High School']
- **refresh** (*boolean*) – (optional) Setting this to `False` will only update the credentials in intuit and will not query against the instituion. This might cause issues because some institutions require you to re-answer the challenge questions. Default: `True`

Returns `None`

When using `update_institution_login()` you might get a challenge response:

```
>>> r = client.update_institution_login(100000, 83850162, **{
    'Banking Userid': 'tfa_choice',
    'Banking Password': 'anyvalue',
   })
>>> r
<AggCatResponse 401>
>>> r.content
<Challenges object [<Challenge object @ 0x10d12d5d0>,<Challenge object @ 0x10d12d110> ...] @ 0x1
>>> r.headers
{'challengenodeid': '10.136.17.82',
 'challengesessionid': 'c31c8a55-754e-4252-8212-b8143270f84f',
 'connection': 'close',
 'content-length': '275',
 'content-type': 'application/xml',
 'date': 'Mon, 12 Aug 2013 03:15:42 GMT',
 'intuit_tid': 'e41418d4-7b77-401e-a158-12514b0d84e3',
 'server': 'Apache/2.2.22 (Unix)',
 'status': '401',
 'via': '1.1 ipp-gateway-ap02'}
```

- The `challenge_session_id` parameter comes from `r.headers['challengesessionid']`
- The `challenge_node_id` parameter comes from `r.headers['challengenodeid']`

Now update the challenge:

```
>>> challenge_session_id = r.headers['challengesessionid']
>>> challenge_node_id = r.headers['challengenodeid']
>>> responses = ['Black Cat', 'Meow High School']
>>> r.update_challenge(
        83850162,
        challenge_session_id,
        challenge_node_id,
        responses
    )
```

5.1.3 Working with Institutions

Getting all Institutions

`AggcatClient.`**`get_institutions`**`()`

Get a list of financial instituions

Returns `AccgatResponse`

```
>>> institutions = client.get_institutions()
>>> len(institutions)
18716
>>> institutions[0].institution_name
'Carolina Foothills FCU Credit Card'
```

Note: This call takes a very long time! Once you get your `institution_id` write it down so you don't forget it. Saving the output using `AggCatResponse.content.to_xml()` is a good idea.

Searching for your institution

Currently finding an institution is somewhat of a manual process. Soon, there will be a helper method on the client that will have a better search. Patches welcome ;). This example searches for an institution that contains "chase" in any of the XML elements:

```
from aggcat import AggcatClient
from lxml import etree
from aggcat.utils import remove_namespaces

client = AggcatClient(
    'oauth_consumer_key',
    'oauth_consumer_secret',
    'saml_identity_provider_id',
    'customer_id',
    '/path/to/x509/appname.key'
)

search_string = 'Chase'
institutions = client.get_institutions()
```

```
xml = etree.fromstring(institutions.to_xml())
xml = etree.fromstring(remove_namespaces(xml))

for element in xml.xpath('./institution[contains(., "chase")]'):
    id = element.xpath('./institutionId')[0].text
    name = element.xpath('./institutionName')[0].text
    print id, name

13278 JP Morgan Chase Bank
13640 Quicken Visa
14554 Chase Bank Credit Card (Amazon.com)
14910 Chase e-Funds Card
14777 Fox Chase Bank - Business Banking
13718 Fox Chase Bank
14484 Chevy Chase Bank - Web Cash Manager
...
```

Note: This query will take a very long time depending on your internet connection. It returns 18000+ institutions in XML format. Sux :(

Getting Institution details

`AggcatClient.`**`get_institution_details`**`(`*institution_id*`)`

Get the details of a finanical institution

Parameters **institution_id** (*integer*) – The institution's id. See *Searching for your institution*.

Returns `AccgatResponse`

```
>>> r = client.get_institution_details()
>>> r
<AggCatResponse 200>
>>> r.content
<Institutiondetail object @ 0x110371f50>
>>> r.content.institution_name
'JP Morgan Chase Bank'
>>> r.content.address.address1
'P O Box 36520'
```

5.1.4 Accounts & Account transactions

Here is the meat of the API. This is what you came for, the ability to aggregate your financials. Here be dragons or not. :)

Customer accounts

`AggcatClient.`**`get_customer_accounts`**`()`

Get a list of all current customer accounts

Returns `AggcatResponse`

This endpoint assumes that the customer accounts we are getting are for the one set in *Initializing the client*:

```
>>> r = ac.get_customer_accounts()
>>> r.content
<Accountlist object [<Investmentaccount object @ 0x1104779d0>,<Creditaccount object @ 0x110477d5
>>> for account in r.content:
        print account.account_nickname, account. account_number
My Retirement 0000000001
My Visa 4100007777
My Mortgage 1000000001
My Brokerage 0000000000
My Roth IRA 2000004444
My Savings 1000002222
My Line of Credit 8000006666
My CD 2000005555
My Auto Loan 8000008888
My Checking 1000001111
```

Note: Attributes on accounts very depending on account type. See account reference in the Intuit documentation. Also note that when the XML gets objectified XML attributes like `accountId` get converted to `account_id`

Login Accounts

`AggcatClient.`**`get_login_accounts`**`(`*`login_id`*`)`

Get a list of account belonging to a login

Parameters **login_id** (*integer*) – Login id of the instiution. This can be retrieved from an account.

Returns `AggcatResponse`

You may have multiple logins. For example, a Fidelity Account and a Bank of America. This will allow you to get onlt the accounts for a specified login:

```
>>> client.get_login_accounts(83850162)
<AggCatResponse 200>
>>> r.content
<Accountlist object [<Investmentaccount object @ 0x1090c9bd0>,<Loanaccount object @ 0x1090c9890>
>>> len(r.content)
10
>>> r.content[0]
<Investmentaccount object @ 0x1090c9bd0>
```

Note: Attributes on accounts very depending on account type. See account reference in the Intuit documentation. Also note that when the XML gets objectified XML attributes like `accountId` get converted to `account_id`

Single Account Details

`AggcatClient.`**`get_account`**`(`*`account_id`*`)`

Get the details of an account

Parameters **account_id** (*integer*) – the id of an account retrieved from `get_login_accounts()` or `get_customer_accounts()`.

Returns `AggcatResponse`

```
>>> r = client.get_account(400004540560)
<AggCatResponse 200>
>>> r.content
<Investmentaccount object @ 0x1091cfc10>
>>> r.content.account_id
'400004540560'
```

Note: Attributes on accounts very depending on type. See account reference in the Intuit documentation. Also note that when the XML gets objectified XML attributes like `accountId` get converted to `account_id`

Transactions

This might hurt or be candy to your eyes depending on how good of a budgeter you are. heh!

`AggcatClient.`**`get_account_transactions`**(*account_id*, *start_date*, *end_date=None*)

Get specific account transactions from a date range

Parameters

- **account_id** (*integer*) – the id of an account retrieved from `get_login_accounts()` or `get_customer_accounts()`.
- **start_date** (*string*) – the date you want the transactions to start in the format YYYY-MM-DD
- **end_date** (*string*) – (optional) the date you want the transactions to end in the format YYYY-MM-DD

Returns `AggcatResponse`

```
>>> r = ac.get_account_transactions(400004540560, '2013-08-10', '2013-08-12')
>>> r
<AggCatResponse 200>
>>> r.content
<Transactionlist object [<Investmenttransaction object @ 0x10a380710>,<Investmenttransaction obj
>>> len(r.content)
3
>>> for t in r.content:
        print t.id, t.description, t.total_amount, t.currency_type
400189790351 IRA debit 222 -8.1 USD
400190413930 IRA debit 224 -8.12 USD
400190413931 IRA credit 223 8.11 USD
```

Note: Attributes on transactions very depending on transaction type. See transaction reference in the Intuit documentation. When the XML gets objectified XML attributes like `totalAmount` get converted to `total_amount`. This endpoint is not ordered by the date of transaction.

Investement Positions

`AggcatClient.`**`get_investment_positions`**(*account_id*)

Get the investment positions of an account

Parameters **account_id** (*integer*) – the id of an account retrieved from `get_login_accounts()` or `get_customer_accounts()`.

Returns AggcatResponse

```
>>> r = client.get_investment_positions(400004540560)
<AggCatResponse 200>
>>> r.content
<Investmentpositions object @ 0x10a25ebd0>
```

Note: This endpoint has needs to be tested with an account that actually returns data here

---

Updating Account Type

Certain banks do not present account types in any of the API data or data that comes back from screen scraping banks accounts without APIs. In those cases the account types are noted as "Other". Most likely, you will know what the account actually is so you can update the type here.

AggcatClient.**update_account_type**(*account_id*, *account_name*, *account_type*)

Update an account's type

Parameters

- **account_id** (*integer*) – the id of an account retrieved from get_login_accounts() or get_customer_accounts().
- **account_name** (*string*) – See possible values for account names and types below
- **account_type** (*string*) – See possible values for account names and types below

Returns None

```
>>> client.update_account_type(400004540560, 'investment', '403b')
```

Possible values for account names and types

Name	Types
banking	checking, savings, moneymrkt, cd, cashmanagement, overdraft
credit	creditcard, lineofcredit, other
loan	loan, auto, commercial, constr, consumer, homeequity, military, mortgage, smb, student
investment	taxable, 401k, brokerage, ira, 403b, keogh, trust, tda, simple, normal, sarsep, ugma, other

Deleting An Account

AggcatClient.**delete_account**(*account_id*)

Delete an account

Parameters **account_id** (*integer*) – the id of an account retrieved from get_login_accounts() or get_customer_accounts().

Returns *None*

```
>>> r = client.delete_account(400004540560)
```

Deleting a customer

AggcatClient.**delete_customer**()

Delete the current customer

Returns *None*

Warning: This deletes all information about the customer. You will have to re-authenticate and rediscover accounts. This endpoint assumes that the customer account getting deleted is the one set in *Initializing the client*. Once this endpoint has been called **do not call** any other endpoints with this customer because the system will automatically create them again!

```
>>> client.delete_customer()
```

CHAPTER

SIX

RELEASE NOTES

0.2

- Cleanup
- Made end_date an optional parameter in *get_account_transactions* to reflect intuit
- Added *requirements.pip* file do that docs build correctly on readthedocs.org

0.1

- Initial Release