bandwidth_sdk Documentation

Release 1.0.6-stable

Bandwidth App Platform

July 14, 2020

Contents

bandwidth_sdk is a python library for working with Bandwidth Voice And Messaging APIs.

Complete original documentation of the API can be found here

Install the SDK with pip:

```
pip install bandwidth-sdk
```

The Bandwidth-Python API is broken up into logical pieces:

- Voice API
- Account API
- Messaging API

Before using the sdk you must initialize at least one client with your Bandwidth App Platform API credentials:

```
import bandwidth
voice_api = bandwidth.client('voice', 'u-user', 't-token', 's-secret')
messaging_api = bandwidth.client('messaging', 'u-user', 't-token', 's-secret')
account_api = bandwidth.client('account', 'u-user', 't-token', 's-secret')
```

Or import each individually for better IDE integration:

```
from bandwidth import messaging, voice, account
messaging_api = messaging.Client('u-user', 't-token', 's-secret')
voice_api = voice.Client('u-user', 't-token', 's-secret')
account_api = account.Client('u-user', 't-token', 's-secret')
```

CHAPTER 1

Voice API

- Phone Calls
- Conferences
- Recordings
- Transcriptions

CHAPTER 2

Messaging API

- Send MMS
- Send SMS
- Fetch Messages

CHAPTER 3

Account API

- Account
- Applications
- Search for numbers
- Register Domains and Endpoints
- Fetch Errors
- Upload/Download Media
- Order/update Phone Numbers

CHAPTER 4

Contents:

4.1 Quickstart Guide

4.1.1 Installation

To install the latest stable version with pip use the following command:

```
pip install bandwidth-sdk
```

If you want to install the bleeding edge version of the SDK from our github repository use the following command:

```
pip install -e git+https://github.com/bandwidth/python-bandwidth.git#egg=bandwidth_sdk
```

Note: This may have to be run as *root* or with *–user* flag if you are not using python virtual environment.

4.1.2 Client Initialization

Before using the sdk you must initialize a Client with your Bandwidth App Platform API credentials:

```
import bandwidth
voice_api = bandwidth.client('voice', 'u-user', 't-token', 's-secret')
messaging_api = bandwidth.client('messaging', 'u-user', 't-token', 's-secret')
account_api = bandwidth.client('account', 'u-user', 't-token', 's-secret')
```

Or import each individually for better IDE integration:

```
from bandwidth import messaging, voice, account
messaging_api = messaging.Client('u-user', 't-token', 's-secret')
voice_api = voice.Client('u-user', 't-token', 's-secret')
account_api = account.Client('u-user', 't-token', 's-secret')
```

4.1.3 Code Samples

Each of these code sample assumes that you have already initialized a client as described above.

Phone Numbers

Get available number via location search:

```
import bandwidth
account_api = bandwidth.client('account', 'u-user', 't-token', 's-secret')
numbers = account_api.search_available_local_numbers(area_code = '910', quantity = 3)
print(numbers)
## [   {   'city'          : 'WILMINGTON',
##         'national_number': '(910) 444-0230',
##         'number'        : '+19104440230',
##         'price'         : '0.35',
##         'rate_center'    : 'WILMINGTON',
##         'state'         : 'NC'},
##     {   'city'          : 'WILMINGTON',
##         'national_number': '(910) 444-0263',
##         'number'        : '+19104440263',
##         'price'         : '0.35',
##         'rate_center'    : 'WILMINGTON',
##         'state'         : 'NC'},
##     {   'city'          : 'WILMINGTON',
##         'national_number': '(910) 444-0268',
##         'number'        : '+19104440268',
##         'price'         : '0.35',
##         'rate_center'    : 'WILMINGTON',
##         'state'         : 'NC'}
## ]

my_number = api.order_phone_number(numbers[0]['number'])

print(my_number)
#n-rnd5ecson3da39fchqmrj3q
```

Calling

Create a call:

```
import bandwidth
voice_api = bandwidth.client('voice', 'u-user', 't-token', 's-secret')
call_id = voice_api.create_call(from_ = '+1234567890', to = '+1234567891', callback_url = "http://you
print(call_id)
## c-abc123

my_call = api.get_call(call_id)
print(my_call)
## {   'callback_url'         : 'http://yoursite.com/calls',
##     'direction'           : 'out',
##     'events'              : 'https://api.catapult.inetwork.com/v1/users/u-abc/calls/c-abc123/event
##     'from'                : '+1234567890',
##     'id'                  : 'c-abc123',
##     'recording_enabled'    : False,
##     'recording_file_format' : 'wav',
##     'recordings'          : 'https://api.catapult.inetwork.com/v1/users/u-abc/calls/c-abc123/reco
##     'start_time'           : '2017-01-26T16:10:11Z',
##     'state'               : 'started',
##     'to'                  : '+1234567891',
##     'transcription_enabled': False,
##     'transcriptions'      : 'https://api.catapult.inetwork.com/v1/users/u-abc/calls/c-abc123/trans
```

Retrieving list of calls:

```
import bandwidth
voice_api = bandwidth.client('voice', 'u-user', 't-token', 's-secret')
call_list = voice_api.list_calls(to = '+19192223333', size = 2)
print(list(call_list))
## [
##   {
##     'active_time'          : '2017-01-26T16:10:23Z',
##     'callback_url'         : 'http://yoursite.com/calls',
##     'chargeable_duration'  : 60,
##     'direction'           : 'out',
##     'endTime'             : '2017-01-26T16:10:33Z',
##     'events'              : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-abc123/ev
##     'from'                : '+17079311113',
##     'id'                  : 'c-abc123',
##     'recording_enabled'    : False,
##     'recording_file_format' : 'wav',
##     'recordings'          : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-abc123/re
##     'start_time'           : '2017-01-26T16:10:11Z',
##     'state'               : 'completed',
##     'to'                  : '+19192223333',
##     'transcription_enabled': False,
##     'transcriptions'      : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-abc123/ti
##   },
##   {
##     'active_time'          : '2016-12-29T23:50:35Z',
##     'chargeable_duration'  : 60,
##     'direction'           : 'out',
##     'endTime'             : '2016-12-29T23:50:41Z',
##     'events'              : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-xyz987/ev
##     'from'                : '+19194443333',
##     'id'                  : 'c-xyz987',
##     'recording_enabled'    : False,
##     'recording_file_format' : 'wav',
##     'recordings'          : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-xyz987/re
##     'start_time'           : '2016-12-29T23:50:15Z',
##     'state'               : 'completed',
##     'to'                  : '+19192223333',
##     'transcription_enabled': False,
##     'transcriptions'      : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-xyz987/ti
##   }
## ]
```

Messaging

Example: Send Text Message:

```
message_id = messaging_api.send_message(from_ = '+1234567980',
                                        to = '+1234567981',
                                        text = 'SMS message')
print(message_id)
# m-messageId
```

Example: Send Picture Message:

```
message_id = messaging_api.send_message(from_ = '+1234567980',
                                        to = '+1234567981',
                                        text = 'MMS message',
```

```
                                media=['http://cat.com/cat.png'])
print(message_id)
# m-messageId
```

Example: Bulk Send Picture or Text messages (or both):

```
results = messaging_api.send_messages([
    {'from': '+1234567980', 'to': '+1234567981', 'text': 'SMS message'},
    {'from': '+1234567980', 'to': '+1234567982', 'text': 'SMS message2'}
])
```

Example: Fetch information about single message:

```
my_message = messaging_api.get_message('m-na6cpyjf2qcpz6l3drhcx7y')
print(my_message)

## {
##     'callback_url'              :'https://yoursite.com/message',
##     'direction'                 :'in',
##     'from'                      :'+19193047864',
##     'id'                        :'m-messageId',
##     'media'                     :[],
##     'message_id'                :'m-messageId',
##     'skip_mms_carrier_validation':True,
##     'state'                     :'received',
##     'text'                      :'Hey there',
##     'time'                      :'2017-02-01T21:10:32Z',
##     'to'                        :'+19191234567'
## }
```

4.2 Messaging API

The Messaging API contains the methods to send sms and mms messages.

4.2.1 Client Initialization

Before using the sdk you must initialize a Client with your Bandwidth App Platform API credentials:

```
# Root import
import bandwidth
messaging_api = bandwidth.client('messaging', 'u-user', 't-token', 's-secret')

# OR for IDE goodness with auto completes
from bandwidth import messaging
messaging_api = messaging.Client('u-user', 't-token', 's-secret')
```

Example: Send Text Message:

```
message_id = messaging_api.send_message(from_ = '+1234567980',
                                to = '+1234567981',
                                text = 'SMS message')
print(message_id)
# m-messageId
```

Example: Send Picture Message:

```
message_id = messaging_api.send_message(from_ = '+1234567980',
                                to = '+1234567981',
                                text = 'MMS message',
                                media=['http://cat.com/cat.png'])
print(message_id)
# m-messageId
```

Example: Bulk Send Picture or Text messages (or both):

```
results = messaging_api.send_messages([
    {'from': '+1234567980', 'to': '+1234567981', 'text': 'SMS message'},
    {'from': '+1234567980', 'to': '+1234567982', 'text': 'SMS message2'}
])
```

Example: Fetch information about single message:

```
my_message = messaging_api.get_message('m-na6cpyjf2qcpz6l3drhcx7y')
print(my_message)

## {
##     'callback_url'             :'https://yoursite.com/message',
##     'direction'               :'in',
##     'from'                    :'+19193047864',
##     'id'                      :'m-messageId',
##     'media'                   :[],
##     'message_id'               :'m-messageId',
##     'skip_mms_carrier_validation':True,
##     'state'                   :'received',
##     'text'                    :'Hey there',
##     'time'                    :'2017-02-01T21:10:32Z',
##     'to'                      :'+19191234567'
## }
```

4.2.2 Messaging Methods

Messages

list_messages

Client.**list_messages**(*from_=None*, *to=None*, *from_date_time=None*, *to_date_time=None*, *direction=None*, *state=None*, *delivery_state=None*, *sort_order=None*, *size=None*, ***kwargs*)

Get a list of user's messages

Parameters

- **from_** (*str*) – The phone number to filter the messages that came from
- **to** (*str*) – The phone number to filter the messages that was sent to
- **from_date_time** (*str*) – The starting date time to filter the messages (must be in yyyy-MM-dd hh:mm:ss format, like 2014-05-25 12:00:00.)
- **to_date_time** (*str*) – The ending date time to filter the messages (must be in yyyy-MM-dd hh:mm:ss format, like 2014-05-25 12:00:00.)

- **direction** (*str*) – Filter by direction of message, in - a message that came from the telephone network to one of your numbers (an "inbound" message) or out - a message that was sent from one of your numbers to the telephone network (an "outbound" message)
- **state** (*str*) – The message state to filter. Values are 'received', 'queued', 'sending', 'sent', 'error'
- **delivery_state** (*str*) – The message delivery state to filter. Values are 'waiting', 'delivered', 'not-delivered'
- **sort_order** (*str*) – How to sort the messages. Values are 'asc' or 'desc'
- **size** (*str*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 1000)

Return type types.GeneratorType

Returns list of messages

Example: Search for all messages and are in error:

```
message_list = api.list_messages()

for message in message_list:
    if message['state'] == 'error':
        print(message['id'])
## m-it6ewpyiyadfe
## m-pjnqofcjyadfe
## m-t2gspvs6iadfe
## m-shuh6d6pyadfe
```

send_message

Client.**send_message**(*from_*, *to*, *text=None*, *media=None*, *receipt_requested=None*, *callback_url=None*, *callback_http_method=None*, *callback_timeout=None*, *fallback_url=None*, *tag=None*, ***kwargs*)

Send a message (SMS or MMS)

Parameters

- **from_** (*str*) – One of your telephone numbers the message should come from
- **to** (*str*) – The phone number the message should be sent to
- **text** (*str*) – The contents of the text message
- **media** (*list*) – For MMS messages, a media url to the location of the media or list of medias to be sent send with the message.
- **receipt_requested** (*str*) – Requested receipt option for outbound messages: 'none', 'all', 'error'
- **callback_url** (*str*) – The server URL where the events related to the outgoing message will be sent to
- **callback_http_method** (*str*) – Determine if the callback event should be sent via HTTP GET or HTTP POST. Values are get or post Default is post
- **callback_timeout** (*str*) – Determine how long should the platform wait for callbackUrl's response before timing out (milliseconds).

- **fallback_url** (*str*) – The server URL used to send the message events if the request to callbackUrl fails.
- **tag** (*str*) – Any string, it will be included in the callback events of the message.

Return type str

Returns id of created message

Example: Send Text Message:

```
id = api.send_message(
    from_ = '+1234567980',
    to = '+1234567981',
    text = 'SMS message'
    )
```

Example: Send Picture Message:

```
id = api.send_message(
    from_ = '+1234567980',
    to = '+1234567981',
    media = ['http://host/path/to/file']
    )
```

send_messages

Client.**send_messages**(*messages_data*)

Send some messages by one request

Parameters **messages_data** (*list*) – List of messages to send

Parameters of each message

from One of your telephone numbers the message should come from

to The phone number the message should be sent to

text The contents of the text message

media For MMS messages, a media url to the location of the media or list of medias to be sent send with the message.

receiptRequested Requested receipt option for outbound messages: 'none', 'all', 'error'

callbackUrl The server URL where the events related to the outgoing message will be sent to

callbackHttpMethod Determine if the callback event should be sent via HTTP GET or HTTP POST. Values are get or post Default is post

callbackTimeout Determine how long should the platform wait for callbackUrl's response before timing out (milliseconds).

fallbackUrl The server URL used to send the message events if the request to callbackUrl fails.

tag Any string, it will be included in the callback events of the message.

Return type list

Returns results of sent messages

Example: Bulk Send Picture or Text messages (or both):

```
    results = api.send_messages([
        {'from': '+1234567980', 'to': '+1234567981', 'text': 'SMS message'},
        {'from': '+1234567980', 'to': '+1234567982', 'text': 'SMS message2'}
    ])
```

get_message

Client.**get_message**(*id*)

Get information about a message

Parameters **id** (*str*) – id of a message

Return type dict

Returns message information

Example: Fetch information about single message:

```
    my_message = api.get_message('m-abc123')
    print(my_message)

    ## {
    ##     'callback_url'              :'https://yoursite.com/message',
    ##     'direction'                 :'in',
    ##     'from'                      :'+19193047864',
    ##     'id'                        :'m-messageId',
    ##     'media'                     :[],
    ##     'message_id'                :'m-messageId',
    ##     'skip_mms_carrier_validation':True,
    ##     'state'                     :'received',
    ##     'text'                      :'Hey there',
    ##     'time'                      :'2017-02-01T21:10:32Z',
    ##     'to'                        :'+19191234567'
    ## }
```

4.3 Voice API

The Voice API contains all the methods to create and manage:

- Phone Calls
- Conferences
- Recordings
- Transcriptions

4.3.1 Client Initialization

Before using the sdk you must initialize a Client with your Bandwidth App Platform API credentials:

```
# single import
import bandwidth
voice_api = bandwidth.client('voice', 'u-user', 't-token', 's-secret')

# OR for IDE goodness with auto completes
```

```
from bandwidth import voice
voice_api = voice.Client('u-user', 't-token', 's-secret')
```

Create a call:

```
import bandwidth
voice_api = bandwidth.client('voice', 'u-user', 't-token', 's-secret')
call_id = voice_api.create_call(from_ = '+1234567890', to = '+1234567891', callback_url = "http://you
print(call_id)
## c-abc123

my_call = api.get_call(call_id)
print(my_call)
## {   'callback_url'         : 'http://yoursite.com/calls',
##     'direction'           : 'out',
##     'events'              : 'https://api.catapult.inetwork.com/v1/users/u-abc/calls/c-abc123/event
##     'from'                : '+1234567890',
##     'id'                  : 'c-abc123',
##     'recording_enabled'    : False,
##     'recording_file_format' : 'wav',
##     'recordings'          : 'https://api.catapult.inetwork.com/v1/users/u-abc/calls/c-abc123/reco
##     'start_time'           : '2017-01-26T16:10:11Z',
##     'state'               : 'started',
##     'to'                  : '+1234567891',
##     'transcription_enabled': False,
##     'transcriptions'      : 'https://api.catapult.inetwork.com/v1/users/u-abc/calls/c-abc123/trans
```

Retrieving list of calls:

```
import bandwidth
voice_api = bandwidth.client('voice', 'u-user', 't-token', 's-secret')
call_list = voice_api.list_calls(to = '+19192223333', size = 2)
print(list(call_list))
## [
##   {
##     'active_time'          : '2017-01-26T16:10:23Z',
##     'callback_url'         : 'http://yoursite.com/calls',
##     'chargeable_duration'  : 60,
##     'direction'           : 'out',
##     'endTime'             : '2017-01-26T16:10:33Z',
##     'events'              : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-abc123/ev
##     'from'                : '+17079311113',
##     'id'                  : 'c-abc123',
##     'recording_enabled'    : False,
##     'recording_file_format' : 'wav',
##     'recordings'          : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-abc123/re
##     'start_time'           : '2017-01-26T16:10:11Z',
##     'state'               : 'completed',
##     'to'                  : '+19192223333',
##     'transcription_enabled': False,
##     'transcriptions'      : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-abc123/ti
##   },
##   {
##     'active_time'          : '2016-12-29T23:50:35Z',
##     'chargeable_duration'  : 60,
##     'direction'           : 'out',
##     'endTime'             : '2016-12-29T23:50:41Z',
##     'events'              : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-xyz987/ev
##     'from'                : '+19194443333',
```

```
##     'id'                  : 'c-xyz987',
##     'recording_enabled'    : False,
##     'recording_file_format' : 'wav',
##     'recordings'          : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-xyz987/re
##     'start_time'           : '2016-12-29T23:50:15Z',
##     'state'               : 'completed',
##     'to'                  : '+19192223333',
##     'transcription_enabled': False,
##     'transcriptions'      : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-xyz987/ti
##   }
## ]
```

4.3.2 Voice Methods

Calls

list_calls

Client.**list_calls**(*bridge_id=None*, *conference_id=None*, *from_=None*, *to=None*, *size=None*, *sort_order=None*, ***kwargs*)

Get a list of calls

Parameters

- **bridge_id** (*str*) – The id of the bridge for querying a list of calls history
- **conference_id** (*str*) – The id of the conference for querying a list of calls history
- **from_** (*str*) – The number to filter calls that came from
- **to** (*str*) – The number to filter calls that was called to
- **sort_order** (*str*) – How to sort the calls. Values are asc or desc If no value is specified the default value is desc
- **size** (*int*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 1000)

Return type types.GeneratorType

Returns list of calls

Example: Fetch calls from specific telephone number:

```
call_list = api.list_calls(from_ = '+19192223333', size = 1000)

total_chargeable_duration = 0

for call in call_list:
    total_chargeable_duration += call['chargeable_duration']

print(total_chargeable_duration)
## 240
```

Example: List Calls:

```
call_list = api.list_calls(to = '+19192223333', size = 2)
print(list(call_list))
## [
##   {
```

```
    ##     'active_time'          : '2017-01-26T16:10:23Z',
    ##     'callback_url'         : 'http://yoursite.com/calls',
    ##     'chargeable_duration'  : 60,
    ##     'direction'           : 'out',
    ##     'end_time'             : '2017-01-26T16:10:33Z',
    ##     'events'              : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-abc1
    ##     'from'                : '+17079311113',
    ##     'id'                  : 'c-abc123',
    ##     'recording_enabled'    : False,
    ##     'recording_file_format' : 'wav',
    ##     'recordings'          : 'https://api.../v1/users/u-abc123/calls/c-abc123/recordings',
    ##     'start_time'           : '2017-01-26T16:10:11Z',
    ##     'state'               : 'completed',
    ##     'to'                  : '+19192223333',
    ##     'transcription_enabled': False,
    ##     'transcriptions'      : 'https://api.../v1/users/u-abc123/calls/c-abc123/transcriptions'
    ##   },
    ##   {
    ##     'active_time'          : '2016-12-29T23:50:35Z',
    ##     'chargeable_duration'  : 60,
    ##     'direction'           : 'out',
    ##     'end_time'             : '2016-12-29T23:50:41Z',
    ##     'events'              : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-xyz9
    ##     'from'                : '+19194443333',
    ##     'id'                  : 'c-xyz987',
    ##     'recording_enabled'    : False,
    ##     'recording_file_format' : 'wav',
    ##     'recordings'          : 'https://api.../v1/users/u-abc123/calls/c-xyz987/recordings',
    ##     'start_time'           : '2016-12-29T23:50:15Z',
    ##     'state'               : 'completed',
    ##     'to'                  : '+19192223333',
    ##     'transcription_enabled': False,
    ##     'transcriptions'      : 'https://api.../v1/users/u-abc123/calls/c-xyz987/transcriptions'
    ##   }
    ## ]
```

create_call

Client.**create_call**(*from_*, *to*, *call_timeout=None*, *callback_url=None*, *callback_timeout=None*, *callback_http_method=None*, *fallback_url=None*, *bridge_id=None*, *conference_id=None*, *recording_enabled=False*, *recording_file_format=None*, *recording_max_duration=None*, *transcription_enabled=False*, *tag=None*, *sip_headers=None*, ***kwargs*)

Create a call

Parameters

- **from_** (*str*) – A Bandwidth phone number on your account the call should come from (required)
- **to** (*str*) – The number to call (required)
- **call_timeout** (*str*) – Determine how long should the platform wait for call answer before timing out in seconds.
- **callback_url** (*str*) – The full server URL where the call events related to the Call will be sent to.

- **callback_timeout** (*str*) – Determine how long should the platform wait for callbackUrl's response before timing out in milliseconds.
- **callback_http_method** (*str*) – Determine if the callback event should be sent via HTTP GET or HTTP POST. Values are "GET" or "POST" (if not set the default is POST).
- **fallback_url** (*str*) – The full server URL used to send the callback event if the request to callbackUrl fails.
- **bridge_id** (*str*) – The id of the bridge where the call will be added.
- **conference_id** (*str*) – Id of the conference where the call will be added. This property is required if you want to add this call to a conference.
- **recording_enabled** (*bool*) – Indicates if the call should be recorded after being created. Set to "true" to enable. Default is "false".
- **recording_file_format** (*str*) – The file format of the recorded call. Supported values are wav (default) and mp3.
- **recording_max_duration** (*str*) – Indicates the maximum duration of call recording in seconds. Default value is 1 hour.
- **transcription_enabled** (*bool*) – Recordings for this call is going to be automatically transcribed.
- **tag** (*str*) – A string that will be included in the callback events of the call.
- **sip_headers** (*str*) – Map of Sip headers prefixed by "X-". Up to 5 headers can be sent per call.

Return type str

Returns id of created call

Example: Create an outbound phone call:

```
call_id = api.create_call(from_='+1234567890',
                          to='+1234567891',
                          callback_url='http://yoursite.com/calls')

print(call_id)
## c-abc123

my_call = api.get_call(call_id)
print(my_call)
## {   'callback_url'         : 'http://yoursite.com/calls',
##     'direction'            : 'out',
##     'events'               : 'https://api.catapult.inetwork.com/v1/users/u-abc/calls/c-abc123/
##     'from'                 : '+1234567890',
##     'id'                   : 'c-abc123',
##     'recording_enabled'    : False,
##     'recording_file_format' : 'wav',
##     'recordings'           : 'https://api.catapult.inetwork.com/v1/users/u-abc/calls/c-abc123/
##     'start_time'           : '2017-01-26T16:10:11Z',
##     'state'                : 'started',
##     'to'                   : '+1234567891',
##     'transcription_enabled': False,
##     'transcriptions'       : 'https://api.../v1/users/u-abc/calls/c-abc123/transcriptions'}
```

get_call

Client.**get_call**(*call_id*)

Get information about a call

Parameters **call_id** (*str*) – id of a call

Return type dict

Returns call information

Fetch and Print Call:

```
my_call = api.get_call(call_id)
print(my_call)
## {   'callback_url'         : 'http://yoursite.com/calls',
##     'direction'            : 'out',
##     'events'               : 'https://api.catapult.inetwork.com/v1/users/u-abc/calls/c-abc123/
##     'from'                 : '+1234567890',
##     'id'                   : 'c-abc123',
##     'recording_enabled'    : False,
##     'recording_file_format' : 'wav',
##     'recordings'           : 'https://api.catapult.inetwork.com/v1/users/u-abc/calls/c-abc123/
##     'start_time'           : '2017-01-26T16:10:11Z',
##     'state'                : 'started',
##     'to'                   : '+1234567891',
##     'transcription_enabled': False,
##     'transcriptions'       : 'https://api..../v1/users/u-abc/calls/c-abc123/transcriptions'}
```

update_call

Client.**update_call**(*call_id*, *state=None*, *recording_enabled=None*, *recording_file_format=None*, *transfer_to=None*, *transfer_caller_id=None*, *whisper_audio=None*, *callback_url=None*, ***kwargs*)

Update a call

Parameters

- **call_id** (*str*) – id of a call
- **state** (*str*) – The call state. Possible values: rejected to reject not answer, active to answer the call, completed to hangup the call, transferring to start and connect call to a new outbound call.
- **recording_enabled** (*bool*) – Indicates if the call should be recorded. Values true or false. You can turn recording on/off and have multiple recordings on a single call.
- **recording_file_format** (*str*) – The file format of the recorded call. Supported values are wav (default) and mp3.
- **transfer_to** (*str*) – Phone number or SIP address that the call is going to be transferred to.
- **transfer_caller_id** (*str*) – This is the caller id that will be used when the call is transferred.
- **whisper_audio** (*dict*) – Audio to be played to the caller that the call will be transferred to.
- **callback_url** (*str*) – The server URL where the call events for the new call will be sent upon transferring.

Update call with state = completed. (Hang up the call):

```
my_call = api.get_call(call_id)
my_call_state = my_call['state']
print(my_call_state)
## started

api.update_call(my_call['id'], state='completed')

my_call = api.get_call(my_call['id'])
print(my_call['state'])
## completed
```

play_audio_to_call

Client.**play_audio_to_call**(*call_id*, *file_url=None*, *sentence=None*, *gender=None*, *locale=None*, *voice=None*, *loop_enabled=None*, ***kwargs*)

Play audio to a call

Parameters

- **call_id** (*str*) – id of a call
- **file_url** (*str*) – The location of an audio file to play (WAV and MP3 supported).
- **sentence** (*str*) – The sentence to speak.
- **gender** (*str*) – The gender of the voice used to synthesize the sentence.
- **locale** (*str*) – The locale used to get the accent of the voice used to synthesize the sentence.
- **voice** (*str*) – The voice to speak the sentence.
- **loop_enabled** (*bool*) – When value is true, the audio will keep playing in a loop.

Play audio in file:

```
api.play_audio_to_call('call_id', file_url='http://host/path/file.mp3')
api.play_audio_to_call('call_id', sentence='Press 0 to complete call', gender='female')
# or with extension methods
api.play_audio_file_to_call('call_id', 'http://host/path/file.mp3')
api.speak_sentence_to_call('call_id', 'Hello')
```

send_dtmf_to_call

Client.**send_dtmf_to_call**(*call_id*, *dtmf_out*, ***kwargs*)

Send DTMF (phone keypad digit presses).

Parameters

- **call_id** (*str*) – id of a call
- **dtmf_out** (*str*) – String containing the DTMF characters to be sent in a call.

Example: Send Digits to call:

```
api.send_dtmf_to_cal('c-callId', '1234')
```

list_call_recordings

Client.**list_call_recordings**(*call_id*)

Get a list of recordings of a call

Parameters **call_id** (*str*) – id of a call

Return type types.GeneratorType

Returns list of recordings

Fetch all call recordings for a call:

```
list = api.get_call_recordings('callId')
```

list_call_transcriptions

Client.**list_call_transcriptions**(*call_id*)

Get a list of transcriptions of a call

Parameters **call_id** (*str*) – id of a call

Return type types.GeneratorType

Returns list of transcriptions

Get all transcriptions for calls:

```
list = api.get_call_transcriptions('callId')
```

list_call_events

Client.**list_call_events**(*call_id*)

Get a list of events of a call

Parameters **call_id** (*str*) – id of a call

Return type types.GeneratorType

Returns list of events

Fetch all events for calls:

```
list = api.get_call_events('callId')
```

get_call_event

Client.**get_call_event**(*call_id*, *event_id*)

Get an event of a call

Parameters

- **call_id** (*str*) – id of a call
- **event_id** (*str*) – id of an event

Return type dict

Returns data of event

Fetch information on a specific event:

```
data = api.get_call_event('callId', 'eventId')
```

create_call_gather

Client.**create_call_gather**(*call_id*, *max_digits=None*, *inter_digit_timeout=None*, *terminating_digits=None*, *tag=None*, ***kwargs*)

Create a gather for a call

Parameters

- **call_id** (*str*) – id of a call
- **max_digits** (*int*) – The maximum number of digits to collect, not including terminating digits (maximum 30).
- **inter_digit_timeout** (*int*) – Stop gathering if a DTMF digit is not detected in this many seconds (default 5.0; maximum 30.0).
- **terminating_digits** (*str*) – A string of DTMF digits that end the gather operation immediately if any one of them is detected
- **tag** (*str*) – A string you choose that will be included with the response and events for this gather operation.

Return type str

Returns id of create of gather

Create gather for only single digit:

```
gather_id = api.create_call_gather('callId', max_digits = 1)
```

get_call_gather

Client.**get_call_gather**(*call_id*, *gather_id*)

Get a gather of a call

Parameters

- **call_id** (*str*) – id of a call
- **gather_id** (*str*) – id of a gather

Return type dict

Returns data of gather

Get Gather information for a previously created gather:

```
data = api.get_call_gather('callId', 'gatherId')
```

update_call_gather

Client.**update_call_gather**(*call_id*, *gather_id*, *state=None*, ***kwargs*)

Update a gather of a call

Parameters

- **call_id** (*str*) – id of a call
- **gather_id** (*str*) – id of a gather
- **state** (*str*) – The only update allowed is state:completed to stop the gather.

End gather:

```
api.update_call_gather('callId', 'gatherId', state = 'completed')
```

answer_call

Client.**answer_call**(*call_id*)

Answer incoming call

Parameters **call_id** (*str*) – id of a call

Example: Answer incoming call:

```
api.answer_call('callId')
```

reject_call

Client.**reject_call**(*call_id*)

Reject incoming call

Parameters **call_id** (*str*) – id of a call

Example: Reject incoming call:

```
api.reject_call('callId')
```

hangup_call

Client.**hangup_call**(*call_id*)

Complete active call

Parameters **call_id** (*str*) – id of a call

Example: Hangup call:

```
api.hangup_call('callId')
```

enable_call_recording

Client.**enable_call_recording**(*call_id*)

Turn on call recording

Parameters **call_id** (*str*) – id of a call

Example: Enable Call Recording:

```
api.enable_call_recording('c-callId')
```

disable_call_recording

Client.**disable_call_recording**(*call_id*)

Turn off call recording

Parameters **call_id** (*str*) – id of a call

Example: Disable Call Recording:

```
api.disable_call_recording('c-callId')
```

toggle_call_recording

Client.**toggle_call_recording**(*call_id*)

Fetches the current call state and either toggles recording on or off

Parameters **call_id** (*str*) – id of the call to toggle

Example: Toggle the call recording:

```
my_call_id = api.create_call(to='+19192223333', from_='+18281114444')
my_call = api.get_call(my_call_id)
print(my_call['recording_enabled'])
## False

api.toggle_call_recording(my_call_id)
my_call = api.get_call(my_call_id)
print(my_call['recording_enabled'])
## True

api.toggle_call_recording(my_call_id)
my_call = api.get_call(my_call_id)
print(my_call['recording_enabled'])
## False
```

transfer_call

Client.**transfer_call**(*call_id*, *to*, *caller_id=None*, *whisper_audio=None*, *callback_url=None*, ***kwargs*)

Transfer a call

Parameters

- **call_id** (*str*) – id of a call
- **to** (*str*) – number that the call is going to be transferred to.
- **caller_id** (*str*) – caller id that will be used when the call is transferred
- **whisper_audio** (*dict*) – audio to be played to the caller that the call will be transferred to
- **callback_url** (*str*) – URL where the call events for the new call will be sent upon transferring

Returns str id of created call

Example: Transfer with whisper:

```
my_sentence = api.build_sentence(sentence = "Hello from Bandwidth",
                      gender="Female",
                      locale="en_UK",
                      voice="Bridget",
                      loop_enabled=True
                     )
my_call = api.get_call('c-callId')
api.transfer_call('c-callId', to = '+1234564890', caller_id = my_call['from'], whisper_audio = m
```

Example: Transfer with whisper audio playback:

```
my_audio = api.build_audio_playback('http://my_site.com/file.mp3', loop_enabled=True)
my_call = api.get_call('c-callId')
api.transfer_call('c-callId', to = '+1234564890', whisper_audio = my_audio )
```

Conferences

create_conference

Client.**create_conference**(*from_*, *callback_url=None*, *callback_timeout=None*, *callback_http_method=None*, *fallback_url=None*, *tag=None*, ***kwargs*)

Create a conference

Parameters

- **from_** (*str*) – The phone number that will host the conference (required)
- **callback_url** (*str*) – The full server URL where the conference events related to the Conference will be sent
- **callback_timeout** (*str*) – Determine how long should the platform wait for callbackUrl's response before timing out in milliseconds.
- **callback_http_method** (*str*) – Determine if the callback event should be sent via HTTP GET or HTTP POST. Values are "GET" or "POST" (if not set the default is POST).
- **fallback_url** (*str*) – The full server URL used to send the callback event if the request to callbackUrl fails or timesout
- **tag** (*str*) – A string that will be included in the callback events of the conference.

Return type str

Returns id of created conference

Example: create simple conference:

```
conference_id = api.create_conference('+12018994444')

print(conference_id)
## conf-ixaagbn5wcyskisiy
```

Example: create conference with extra parameters:

```
conference_id = api.create_conference(from_ = "+12018994444", callback_url = "http://google.com"
                                      callback_timeout= 2000, fallback_url = "http://yahoo.com")

print(conference_id)
## conf-ixaagbn5wcyskisiy
```

```
my_conf = api.get_conference(conference_id)

print(my_conf)
## {   'active_members'     : 0,
##     'callback_http_method': 'post',
##     'callback_timeout'   : 2000,
##     'callback_url'       : 'http://google.com',
##     'created_time'       : '2017-01-26T01:58:59Z',
##     'fallback_url'       : 'http://yahoo.com',
##     'from'               : '+12018994444',
##     'hold'               : False,
##     'id'                 : 'conf-ixaagbn5wcyskisiy',
##     'mute'               : False,
##     'state'              : 'created'}
```

get_conference

Client.**get_conference**(*conference_id*)

Get information about a conference

Parameters **conference_id** (*str*) – id of a conference

Return type dict

Returns conference information

Example: Create then fetch conference:

```
conference_id = api.create_conference(from_ = "+12018994444", callback_url = "http://google.com"
                                      callback_timeout= 2000, fallback_url = "http://yahoo.com")

print(conference_id)
## conf-ixaagbn5wcyskisiy

my_conf = api.get_conference(conference_id)

print(my_conf)
## {   'active_members'     : 0,
##     'callback_http_method': 'post',
##     'callback_timeout'   : 2000,
##     'callback_url'       : 'http://google.com',
##     'created_time'       : '2017-01-26T01:58:59Z',
##     'fallback_url'       : 'http://yahoo.com',
##     'from'               : '+12018994444',
##     'hold'               : False,
##     'id'                 : 'conf-ixaagbn5wcyskisiy',
##     'mute'               : False,
##     'state'              : 'created'}
```

update_conference

Client.**update_conference**(*conference_id*, *state=None*, *mute=None*, *hold=None*, *callback_url=None*, *callback_timeout=None*, *callback_http_method=None*, *fallback_url=None*, *tag=None*, ***kwargs*)

Update a conference

Parameters

- **conference_id** (*str*) – id of a conference
- **state** (*str*) – Conference state. Possible state values are: "completed" to terminate the conference.
- **mute** (*str*) – If "true", all member can't speak in the conference. If "false", all members can speak in the conference
- **hold** (*str*) – If "true", all member can't hear or speak in the conference. If "false", all members can hear and speak in the conference
- **callback_url** (*str*) – The full server URL where the conference events related to the conference will be sent
- **callback_timeout** (*str*) – Determine how long should the platform wait for callback-Url's response before timing out in milliseconds.
- **callback_http_method** (*str*) – Determine if the callback event should be sent via HTTP GET or HTTP POST. Values are "GET" or "POST" (if not set the default is POST).
- **fallback_url** (*str*) – The full server URL used to send the callback event if the request to callbackUrl fails.
- **tag** (*str*) – A string that will be included in the callback events of the conference.

Example: End conference:

```
api.update_conference('conferenceId', state='completed')
```

play_audio_to_conference

Client.**play_audio_to_conference**(*conference_id*, *file_url=None*, *sentence=None*, *gender=None*, *locale=None*, *voice=None*, *loop_enabled=None*, ***kwargs*)

Play audio to a conference

Parameters

- **conference_id** (*str*) – id of a conference
- **file_url** (*str*) – The location of an audio file to play (WAV and MP3 supported).
- **sentence** (*str*) – The sentence to speak.
- **gender** (*str*) – The gender of the voice used to synthesize the sentence.
- **locale** (*str*) – The locale used to get the accent of the voice used to synthesize the sentence.
- **voice** (*str*) – The voice to speak the sentence.
- **loop_enabled** (*str*) – When value is true, the audio will keep playing in a loop.

Example: Play audio file to conference:

```
api.play_audio_to_conference('conferenceId', file_url='http://host/path/file.mp3')
```

Example: Speak Sentence to conference:

```
api.play_audio_to_conference('conferenceId', sentence='Press 0 to complete call', gender='female
```

Example: Use Extensions methods:

```
    # or with extension methods
    api.play_audio_file_to_conference('conferenceId', 'http://host/path/file.mp3')
    api.speak_sentence_to_conference('conferenceId', 'Hello')
```

list_conference_members

Client.**list_conference_members**(*conference_id*)

Get a list of members of a conference

Parameters **conference_id** (*str*) – id of a conference

Return type types.GeneratorType

Returns list of recordings

Example: Fetch and list conference members:

```
    my_conf_id = api.create_conference(from_='+19192223333')
    print(my_conf)
    # conf-confId

    my_call_id = api.create_call(from_='+19192223333', to='+19192223334', conference_id= 'conf-confI
    print(my_call_id)
    # c-callId

    my_conf_member_id = api.create_conference_member(my_conf_id, call_id=my_call_id)
    print(my_conf_member_id)
    # member-memberId

    my_conference_members = list_conference_members(my_conf_id)

    print(list(my_conference_members))

    ## [
    ##    {
    ##        'added_time'  :'2017-01-30T22:01:11Z',
    ##        'call'        :'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId',
    ##        'hold'        :False,
    ##        'id'          :'member-memberId',
    ##        'join_tone'   :False,
    ##        'leaving_tone':False,
    ##        'mute'        :False,
    ##        'removed_time':'2017-01-30T22:01:21Z',
    ##        'state'       :'completed'
    ##    }
    ## ]
```

create_conference_member

Client.**create_conference_member**(*conference_id*, *call_id=None*, *join_tone=None*, *leaving_tone=None*, *mute=None*, *hold=None*, ***kwargs*)

Create a conference member for a conference

Parameters

- **conference_id** (*str*) – id of a conference

- **call_id** (*str*) – The callId must refer to an active call that was created using this conferenceId (required)
- **join_tone** (*bool*) – If "true", will play a tone when the member joins the conference. If "false", no tone is played when the member joins the conference.
- **leaving_tone** (*bool*) – If "true", will play a tone when the member leaves the conference. If "false", no tone is played when the member leaves the conference.
- **mute** (*bool*) – If "true", member can't speak in the conference. If "false", this members can speak in the conference (unless set at the conference level).
- **hold** (*bool*) – If "true", member can't hear or speak in the conference. If "false", member can hear and speak in the conference (unless set at the conference level).

Return type str

Returns id of create of conference member

Example: Create Conference and add member:

```
my_conf_id = api.create_conference(from_='+19192223333')
print(my_conf)
# conf-confId

my_call_id = api.create_call(from_='+19192223333', to='+19192223334', conference_id= 'conf-confI
print(my_call_id)
# c-callId

my_conf_member_id = api.create_conference_member(my_conf_id, call_id=my_call_id, join_tone=True)
print(my_conf_member_id)
# member-memberId

my_conf_member = api.get_conference_member(my_conf_id, my_member_id)
print(my_conf_member)

## {
##     'added_time': '2017-01-30T22:01:11Z',
##     'call'         : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId',
##     'hold'         : False,
##     'id'           : 'member-memberId',
##     'join_tone'     : False,
##     'leaving_tone'  : False,
##     'mute'         : False,
##     'removed_time'  : '2017-01-30T22:01:21Z',
##     'state'        : 'completed'
## }
```

get_conference_member

Client.**get_conference_member**(*conference_id*, *member_id*)

Get a conference member

Parameters

- **conference_id** (*str*) – id of a conference
- **member_id** (*str*) – id of a member

Return type dict

Returns data of conference member

Example: Create Conference and add member:

```
my_conf_id = api.create_conference(from_='+19192223333')
print(my_conf)
# conf-confId

my_call_id = api.create_call(from_='+19192223333', to='+19192223334', conference_id= 'conf-confI
print(my_call_id)
# c-callId

my_conf_member_id = api.create_conference_member(my_conf_id, call_id=my_call_id, join_tone=True)
print(my_conf_member_id)
# member-memberId

my_conf_member = api.get_conference_member(my_conf_id, my_member_id)
print(my_conf_member)

## {
##     'added_time': '2017-01-30T22:01:11Z',
##     'call'         : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId',
##     'hold'         : False,
##     'id'           : 'member-memberId',
##     'join_tone'     : True,
##     'leaving_tone'  : False,
##     'mute'         : False,
##     'removed_time'  : '2017-01-30T22:01:21Z',
##     'state'        : 'active'
## }
```

update_conference_member

Client.**update_conference_member**(*conference_id*, *member_id*, *join_tone=None*, *leaving_tone=None*, *mute=None*, *hold=None*, ***kwargs*)

Update a conference member

Parameters

- **conference_id** (*str*) – id of a conference
- **member_id** (*str*) – id of a conference member
- **join_tone** (*bool*) – If "true", will play a tone when the member joins the conference. If "false", no tone is played when the member joins the conference.
- **leaving_tone** (*bool*) – If "true", will play a tone when the member leaves the conference. If "false", no tone is played when the member leaves the conference.
- **mute** (*bool*) – If "true", member can't speak in the conference. If "false", this members can speak in the conference (unless set at the conference level).
- **hold** (*bool*) – If "true", member can't hear or speak in the conference. If "false", member can hear and speak in the conference (unless set at the conference level).

Example: update conference member:

```
my_conf_id = api.create_conference(from_='+19192223333')
print(my_conf)
# conf-confId
```

```
    my_call_id = api.create_call(from_='+19192223333', to='+19192223334', conference_id= 'conf-confI
    print(my_call_id)
    # c-callId

    my_conf_member_id = api.create_conference_member(my_conf_id, call_id=my_call_id, join_tone=True)
    print(my_conf_member_id)
    # member-memberId

    my_conf_member = api.get_conference_member(my_conf_id, my_member_id)
    print(my_conf_member)

    ## {
    ##     'added_time': '2017-01-30T22:01:11Z',
    ##     'call'         : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId',
    ##     'hold'         : False,
    ##     'id'           : 'member-memberId',
    ##     'join_tone'     : True,
    ##     'leaving_tone'  : False,
    ##     'mute'         : False,
    ##     'removed_time'  : '2017-01-30T22:01:21Z',
    ##     'state'        : 'active'
    ## }

    api.update_conference_member(my_conf_id, my_member_id, mute=True, hold=True)

    my_conf = api.get_conference_member(my_member_id)

    ## {
    ##     'added_time': '2017-01-30T22:01:11Z',
    ##     'call'         : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId',
    ##     'hold'         : True,
    ##     'id'           : 'member-memberId',
    ##     'join_tone'     : True,
    ##     'leaving_tone'  : False,
    ##     'mute'         : True,
    ##     'removed_time'  : '2017-01-30T22:01:21Z',
    ##     'state'        : 'active'
    ## }
```

play_audio_to_conference_member

Client.**play_audio_to_conference_member**(*conference_id*, *member_id*, *file_url=None*, *sentence=None*, *gender=None*, *locale=None*, *voice=None*, *loop_enabled=None*, ***kwargs*)

Play audio to a conference member

Parameters

- **conference_id** (*str*) – id of a conference
- **member_id** (*str*) – id of a conference member
- **file_url** (*str*) – The location of an audio file to play (WAV and MP3 supported).
- **sentence** (*str*) – The sentence to speak.
- **gender** (*str*) – The gender of the voice used to synthesize the sentence.

- **locale** (*str*) – The locale used to get the accent of the voice used to synthesize the sentence.
- **voice** (*str*) – The voice to speak the sentence.
- **loop_enabled** (*str*) – When value is true, the audio will keep playing in a loop.

Example: Play audio to specific conference member:

```
api.play_audio_to_conference_member('conferenceId', 'memberId', file_url='http://host/path/file.
api.play_audio_to_conference_member('conferenceId', 'memberId',
                                    sentence='Press 0 to complete call', gender='female')

# or with extension methods
api.play_audio_file_to_conference_member('conferenceId', 'memberId', 'http://host/path/file.mp3'
api.speak_sentence_to_conference_member('conferenceId', 'memberId', 'Hello')
```

speak_sentence_to_conference_member

Client.**speak_sentence_to_conference_member**(*conference_id*, *member_id*, *sentence*, *gender='female'*, *voice='susan'*, *locale='en_US'*, *tag=None*)

Speak sentence to a conference member

Parameters

- **conference_id** (*str*) – id of a conference
- **member_id** (*str*) – id of a conference member
- **sentence** (*str*) – sentence to say
- **gender** (*str*) – gender of voice
- **voice** (*str*) – voice name
- **locale** (*str*) – locale name
- **tag** (*str*) – A string that will be included in the callback events of the call.

Example: Speak sentence to specific conference member:

```
api.speak_sentence_to_conference_member('conferenceId', 'memberId', 'Hello')
```

play_audio_file_to_conference_member

Client.**play_audio_file_to_conference_member**(*conference_id*, *member_id*, *file_url*, *tag=None*)

Play audio file to a conference member

Parameters

- **conference_id** (*str*) – id of a conference
- **member_id** (*str*) – id of a conference member
- **file_url** (*str*) – URL to remote file to play
- **tag** (*str*) – A string that will be included in the callback events of the call.

Example: Play an audio file to specific member:

```
api.play_audio_file_to_conference_member('conferenceId', 'memberId', 'http://host/path/file.mp3'
```

remove_conference_member

Client.**remove_conference_member**(*conference_id*, *member_id*)

Remove a conference member

Parameters

- **conference_id** (*str*) – id of a conference
- **member_id** (*str*) – id of a conference member

Example: Remove Member from Conference:

```
my_conf = api.get_conference('conferenceId')
my_conf_members = list(api.list_conference_members(my_conf['id']))
print(my_conf_members)

## [{ 'added_time'  : '2017-01-30T23:17:11Z',
##    'call'       : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId',
##    'hold'       : False,
##    'id'         : 'member-memberId',
##    'join_tone'   : False,
##    'leaving_tone': False,
##    'mute'       : False,
##    'state'      : 'active'},
##  { 'added_time'  : '2017-01-30T23:17:14Z',
##    'call'       : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId2',
##    'hold'       : False,
##    'id'         : 'member-memberId2',
##    'join_tone'   : False,
##    'leaving_tone': False,
##    'mute'       : False,
##    'state'      : 'active'}]

api.remove_conference_member(my_conf['id'], my_conf_members[1]['id'])

my_conf_members = list(api.list_conference_members(my_conf['id']))
print(my_conf_members)

## [{ 'added_time'  : '2017-01-30T23:17:11Z',
##    'call'       : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId',
##    'hold'       : False,
##    'id'         : 'member-memberId',
##    'join_tone'   : False,
##    'leaving_tone': False,
##    'mute'       : False,
##    'state'      : 'active'},
##  { 'added_time'  : '2017-01-30T23:17:14Z',
##    'call'       : 'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId2',
##    'hold'       : False,
##    'id'         : 'member-memberId2',
##    'join_tone'   : False,
##    'leaving_tone': False,
##    'mute'       : False,
##    'state'      : 'completed'}]
```

hold_conference_member

Client.**hold_conference_member**(*conference_id*, *member_id*, *hold*)
Hold or unhold a conference member

Parameters

- **conference_id** (*str*) – id of a conference
- **member_id** (*str*) – id of a conference member
- **hold** (*bool*) – hold (if true) or unhold (if false) a member

Example: Put specific conference member on hold:

```
api.hold_conference_member('conferenceId', 'memberId', True)
```

mute_conference_member

Client.**mute_conference_member**(*conference_id*, *member_id*, *mute*)
Mute or unmute a conference member

Parameters

- **conference_id** (*str*) – id of a conference
- **member_id** (*str*) – id of a conference member
- **mute** (*bool*) – mute (if true) or unmute (if false) a member

Example: Mute specific conference member:

```
api.mute_conference_member('conferenceId', 'memberId', True)
```

terminate_conference

Client.**terminate_conference**(*conference_id*)
Terminate of current conference

Parameters **conference_id** (*str*) – id of a conference

Example: End the Conference:

```
api.terminate_conference('conferenceId')
```

hold_conference

Client.**hold_conference**(*conference_id*, *hold*)
Hold or unhold a conference

Parameters

- **conference_id** (*str*) – id of a conference
- **hold** (*bool*) – hold (if true) or unhold (if false) a conference

Example: Put entire confernce on hold, where no one can hear:

```
    api.hold_conference('conferenceId', True)
```

mute_conference

Client.**mute_conference**(*conference_id*, *mute*)

Mute or unmute a conference

Parameters

- **conference_id** (*str*) – id of a conference
- **mute** (*bool*) – mute (if true) or unmute (if false) a conference

Example: Mute the entire Conference, where no one can speak:

```
    api.mute_conference('conferenceId', True)
```

Bridges

list_bridges

Client.**list_bridges**(*size=None*, ***kwargs*)

Get a list of bridges

Parameters **size** (*int*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 1000)

Return type types.GeneratorType

Returns list of bridges

Example: List bridges 1000 at a time:

```
    bridges = api.list_bridges(size=1000)

    for bridge in bridges:
        print(bridge["id"])

    ## brg-6mv7pi22i
    ## brg-3emq7olua
    ## brg-bbufdc7yq
    ## brg-dvpvd7cuy
    ## brg-5ws2buzmq
```

create_bridge

Client.**create_bridge**(*call_ids=None*, *bridge_audio=None*, ***kwargs*)

Create a bridge

Parameters

- **bridge_audio** (*bool*) – Enable/Disable two way audio path (default = true)
- **call_ids** (*str*) – The first of the call ids in the bridge. If either of the call ids is not provided the bridge is logically created and it can be used to place calls later.

Return type str

Returns id of created bridge

Example: Create bridge with 2 calls and audio:

```
bridge_id = api.create_bridge(call_ids = ['callId1', 'callId2'], bridge_audio = True)

print(bridge_id)
# brg-123
```

get_bridge

Client.**get_bridge**(*bridge_id*)

Gets information about a bridge

Parameters **bridge_id** (*str*) – id of a bridge

Return type dict

Returns bridge information

Example: Fetch single bridge by ID:

```
my_bridge = api.get_bridge('brg-bridgeId')
print(my_bridge)

## {   'bridge_audio': True,
##     'calls'       : 'https://api.catapult.inetwork.com/v1/users/u-123/bridges/brg-bridgeId/cal
##     'created_time': '2017-01-26T01:15:09Z',
##     'id'          : 'brg-bridgeId',
##     'state'       : 'created'}

print(my_bridge["state"])
## created
```

update_bridge

Client.**update_bridge**(*bridge_id*, *call_ids=None*, *bridge_audio=None*, ***kwargs*)

Update a bridge

Parameters

- **bridge_id** (*str*) – id of a bridge
- **bridge_audio** (*bool*) – Enable/Disable two way audio path (default = true)
- **call_ids** (*str*) – The first of the call ids in the bridge. If either of the call ids is not provided the bridge is logically created and it can be used to place calls later.

Example: stop bridging audio:

```
my_bridge = api.get_bridge('brg-bridgeId')

print(my_bridge["bridge_audio"])
## True

api.update_bridge(my_bridge['id'], call_ids = ['callId1', 'callId2'], bridge_audio = False)
my_bridge = api.get_bridge(my_bridge['id'])

print(my_bridge["bridge_audio"])
## False
```

list_bridge_calls

Client.**list_bridge_calls**(*bridge_id*)

Get a list of calls of a bridge

Parameters **bridge_id** (*str*) – id of a bridge

Return type types.GeneratorType

Returns list of calls

Example: Fetch all calls that were in a bridge:

```
call_list = api.get_bridge_calls('bridgeId')

print(list(call_list))
## [
##     {
##         "active_time"      : "2013-05-22T19:49:39Z",
##         "direction"       : "out",
##         "from"            : "{fromNumber}",
##         "id"              : "{callId1}",
##         "bridge_id"        : "{bridgeId}",
##         "start_time"       : "2013-05-22T19:49:35Z",
##         "state"           : "active",
##         "to"              : "{toNumber1}",
##         "recording_enabled": false,
##         "events"          : "https://api.catapult.inetwork.com/v1/users/{userId}/calls/{call1
##         "bridge"          : "https://api.catapult.inetwork.com/v1/users/{userId}/bridges/{bri
##     },
##     {
##         "active_time"      : "2013-05-22T19:50:16Z",
##         "direction"       : "out",
##         "from"            : "{fromNumber}",
##         "id"              : "{callId2}",
##         "bridge_id"        : "{bridgeId}",
##         "start_time"       : "2013-05-22T19:50:16Z",
##         "state"           : "active",
##         "to"              : "{toNumber2}",
##         "recording_enabled": false,
##         "events"          : "https://api.catapult.inetwork.com/v1/users/{userId}/calls/{call1
##         "bridge"          : "https://api.catapult.inetwork.com/v1/users/{userId}/bridges/{bri
##     }
## ]
```

play_audio_to_bridge

Client.**play_audio_to_bridge**(*bridge_id*, *file_url=None*, *sentence=None*, *gender=None*, *locale=None*, *voice=None*, *loop_enabled=None*, ***kwargs*)

Play audio to a bridge

Parameters

- **bridge_id** (*str*) – id of a bridge
- **file_url** (*str*) – The location of an audio file to play (WAV and MP3 supported).
- **sentence** (*str*) – The sentence to speak.
- **gender** (*str*) – The gender of the voice used to synthesize the sentence.

- **locale** (*str*) – The locale used to get the accent of the voice used to synthesize the sentence.
- **voice** (*str*) – The voice to speak the sentence.
- **loop_enabled** (*bool*) – When value is true, the audio will keep playing in a loop.

Examples: Play either file for speak sentence:

```
api.play_audio_to_bridge('bridgeId', file_url='http://host/path/file.mp3')
api.play_audio_to_bridge('bridgeId', sentence='Press 0 to complete call', gender='female')

# or with extension methods
api.play_audio_file_to_bridge('bridgeId', 'http://host/path/file.mp3')
api.speak_sentence_to_bridge('bridgeId', 'Hello')
```

Recordings

list_recordings

Client.**list_recordings**(*size=None*, ***kwargs*)

Get a list of call recordings

Parameters **size** (*int*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 1000)

Return type types.GeneratorType

Returns list of recordings

Example: List all recordings:

```
recording_list = api.list_recordings(size=1000)
print(recording_list)
## [
##     {
##         'call'     :'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId',
##         'end_time'  :'2017-01-30T17:58:45Z',
##         'id'       :'rec-recordingId',
##         'media'    :'https://api.catapult.inetwork.com/v1/users/u-abc123/media/c-callId-1.wav
##         'media_name':'c-callId-1.wav',
##         'start_time':'2017-01-30T17:58:34Z',
##         'state'    :'complete'
##     },
##     {
##         'call'     :'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId2',
##         'end_time'  :'2017-01-30T17:53:30Z',
##         'id'       :'rec-recordingId2',
##         'media'    :'https://api.catapult.inetwork.com/v1/users/u-abc123/media/c-callId2-1.wa
##         'media_name':'c-callId2-1.wav',
##         'start_time':'2017-01-30T17:53:20Z',
##         'state'    :'complete'
##     }
## ]
```

get_recording

Client.**get_recording**(*recording_id*)

Gets information about a recording

Parameters **recording_id** (*str*) – id of a recording

Return type dict

Returns recording information

Example: Fetch recording information:

```
my_recording = api.get_recording('recordingId2')
print(my_recording)

## {
##     'call'     :'https://api.catapult.inetwork.com/v1/users/u-abc123/calls/c-callId2',
##     'end_time'  :'2017-01-30T17:53:30Z',
##     'id'       :'rec-recordingId2',
##     'media'    :'https://api.catapult.inetwork.com/v1/users/u-abc123/media/c-callId2-1.wav',
##     'media_name':'c-callId2-1.wav',
##     'start_time':'2017-01-30T17:53:20Z',
##     'state'    :'complete'
## }
```

Transcriptions

list_transcriptions

Client.**list_transcriptions**(*recording_id*, *size=None*, ***kwargs*)

Get a list of transcriptions

Parameters

- **recording_id** (*str*) – id of a recording
- **size** (*int*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 1000)

Return type types.GeneratorType

Returns list of transcriptions

Example: Print off all transcriptions for a recording:

```
transcriptions_list = api.list_transcriptions('recordingId')
print(list(transcriptions_list))
## [
##     {
##         'chargeable_duration': 60,
##         'id': '{transcription-id}',
##         'state': 'completed',
##         'time': '2014-10-09T12:09:16Z',
##         'text': '{transcription-text}',
##         'text_size': 3627,
##         'text_url': '{url-to-full-text}'
##     },
##     {
##         'chargeable_duration': 60,
```

```
    ##         'id': '{transcription-id}',
    ##         'state': 'completed',
    ##         'text': '{transcription-text}',
    ##         'time': '2014-10-09T14:04:44Z',
    ##         'text_size': 72,
    ##         'text_url': '{url-to-full-text}'
    ##     }
    ## ]
```

create_transcription

`Client.`**`create_transcription`**(*recording_id*)

Create a transcirption for given recording

Parameters **`recording_id`** (*str*) – id of a recording

Return type str

Returns id of created transcription

Example: Create new transcription from existing recording:

```
    transcription_id = api.create_transcirption('recordingId')
```

get_transcription

`Client.`**`get_transcription`**(*recording_id*, *transcription_id*)

Get information about a transcription

Parameters

- **`recording_id`** (*str*) – id of a recording
- **`id`** (*str*) – id of a transcription

Return type dict

Returns application information

Example: Fetch a single transcription on a recording:

```
    my_transcription = api.get_transcription('recordingId', 'transcriptionId')
    print(my_transcription)
    ## {
    ##     'chargeable_duration': 11,
    ##     'id'                 : '{transcriptionId}',
    ##     'state'              : 'completed',
    ##     'text'               : 'Hey there, I was calling to talk about plans for this saturday. ',
    ##     'text_size'           : 63,
    ##     'text_url'            : 'https://api.catapult.inetwork.com/.../media/{transcriptionId}',
    ##     'time'               : '2014-12-23T23:08:59Z'
    ## }
```

4.4 Account Api

The account API contains the methods to interact with account features:

- Account
- Applications
- Search for numbers
- Register Domains and Endpoints
- Fetch Errors
- Upload/Download Media
- Order/update Phone Numbers

4.4.1 Client Initialization

Before using the sdk you must initialize a Client with your Bandwidth App Platform API credentials:

```
# Single import
import bandwidth
account_api = bandwidth.client('account', 'u-user', 't-token', 's-secret')

# OR for IDE goodness with auto completes
from bandwidth import account
account_api = account.Client('u-user', 't-token', 's-secret')
```

4.4.2 Code Samples

Each of these samples assumes you have already have a bandwidth account

Phone Numbers

Get available number via location search:

```
from bandwidth import account
account_api = account.Client('u-user', 't-token', 's-secret')

numbers = account_api.search_available_local_numbers(area_code = '910', quantity = 3)

print(numbers)

## [   {   'city'          : 'WILMINGTON',
##         'national_number': '(910) 444-0230',
##         'number'        : '+19104440230',
##         'price'         : '0.35',
##         'rate_center'    : 'WILMINGTON',
##         'state'         : 'NC'},
##     {   'city'          : 'WILMINGTON',
##         'national_number': '(910) 444-0263',
##         'number'        : '+19104440263',
##         'price'         : '0.35',
##         'rate_center'    : 'WILMINGTON',
##         'state'         : 'NC'},
##     {   'city'          : 'WILMINGTON',
##         'national_number': '(910) 444-0268',
##         'number'        : '+19104440268',
##         'price'         : '0.35',
```

```
##          'rate_center'    : 'WILMINGTON',
##          'state'          : 'NC'}
## ]

print(numbers[0]["number"])
## +19104440230
```

4.4.3 Account Methods

Account

get_account

Client.**get_account**()

Get an Account object

Return type dict

Returns account data

Example:

```
    data = api.get_account()
```

get_account_transactions

Client.**list_account_transactions**(*max_items=None*, *to_date=None*, *from_date=None*, *trans_type=None*, *size=None*, *number=None*, ***kwargs*)

Get the transactions from the user's account

Parameters

- **max_items** (*str*) – Limit the number of transactions that will be returned.
- **to_date** (*str*) – Return only transactions that are newer than the parameter. Format: "yyyy-MM-dd'T'HH:mm:ssZ"
- **from_date** (*str*) – Return only transactions that are older than the parameter. Format: "yyyy-MM-dd'T'HH:mm:ssZ"
- **trans_type** (*str*) – Return only transactions that are this type.
- **size** (*int*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 1000)
- **number** (*str*) – Search transactions by phone number

Return type types.GeneratorType

Returns list of transactions

Example: Get transactions:

```
    list = api.get_account_transactions(type = 'charge')
```

Example: Get transactions by date:

```
    list = api.get_account_transactions(type = 'charge')
```

Example: Get transactions filtering by date:

```
list = api.get_account_transactions(type = 'charge')
```

Example: Get transactions limiting result:

```
list = api.get_account_transactions(type = 'charge')
```

Example: Get transactions of payment type:

```
list = api.get_account_transactions(type = 'charge')
```

Applications

list_applications

Client.**list_applications**(*size=None*, ***kwargs*)

Get a list of user's applications

Parameters **size** (*int*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 1000)

Return type types.GeneratorType

Returns list of applications

Example: Fetch and print all applications:

```
apps = api.list_applications()
print(list(apps))
```

Example: Iterate over all applications to find specific name:

```
apps = api.list_applications(size=20)

app_name = ""
while app_name != "MyAppName":
    my_app = next(apps)
    app_name = my_app["name"]

print(my_app)


## {   'auto_answer': True,
##     'callback_http_method': 'get',
##     'id': 'a-asdf',
##     'incoming_call_url': 'https://test.com/callcallback/',
##     'incoming_message_url': 'https://test.com/msgcallback/',
##     'name': 'MyAppName'
## }
```

create_application

Client.**create_application**(*name*, *incoming_call_url=None*, *incoming_call_url_callback_timeout=None*, *incoming_call_fallback_url=None*, *incoming_message_url=None*, *incoming_message_url_callback_timeout=None*, *incoming_message_fallback_url=None*, *callback_http_method=None*, *auto_answer=None*, ***kwargs*)

Creates an application that can handle calls and messages for one of your phone number.

Parameters

- **name** (*str*) – A name you choose for this application (required).
- **incoming_call_url** (*str*) – A URL where call events will be sent for an inbound call.
- **incoming_call_url_callback_timeout** (*str*) – Determine how long should the platform wait for inconmingCallUrl's response before timing out in milliseconds.
- **incoming_call_fallback_url** (*str*) – The URL used to send the callback event if the request to incomingCallUrl fails.
- **incoming_message_url** (*str*) – A URL where message events will be sent for an inbound SMS message
- **incoming_message_url_callback_timeout** (*str*) – Determine how long should the platform wait for incomingMessageUrl's response before timing out in milliseconds.
- **incoming_message_fallback_url** (*str*) – The URL used to send the callback event if the request to incomingMessageUrl fails.
- **callback_http_method** (*str*) – Determine if the callback event should be sent via HTTP GET or HTTP POST. (If not set the default is HTTP POST)
- **auto_answer** (*str*) – Determines whether or not an incoming call should be automatically answered. Default value is 'true'.

Return type str

Returns id of created application

Example: Create Application:

```
my_app_id = api.create_application( name                 = "MyFirstApp",
                                    incoming_call_url    = "http://callback.com/calls",
                                    incoming_message_url  = "http://callback.com/messages",
                                    callback_http_method = "GET")

print(my_app_id)
## a-1232asf123

my_app = api.get_application(my_app_id)
print(my_app)
## {   'auto_answer'        : True,
##     'callback_http_method': 'get',
##     'id'                 : 'a-1232asf123',
##     'incoming_call_url'   : 'http://callback.com/calls',
##     'incoming_message_url': 'http://callback.com/messages',
##     'incoming_sms_url'    : 'http://callback.com/messages',
##     'name'               : 'MyFirstApp2'
## }
```

```
print(my_app["id"])
## a-1232asf123
```

update_application

Client.**update_application**(*app_id*, *name=None*, *incoming_call_url=None*, *incoming_call_url_callback_timeout=None*, *incoming_call_fallback_url=None*, *incoming_message_url=None*, *incoming_message_url_callback_timeout=None*, *incoming_message_fallback_url=None*, *callback_http_method=None*, *auto_answer=None*, ***kwargs*)

Updates an application that can handle calls and messages for one of your phone number.

Parameters

- **app_id** (*str*) – The Id of the application to upate (a-123)
- **name** (*str*) – A name you choose for this application (required).
- **incoming_call_url** (*str*) – A URL where call events will be sent for an inbound call.
- **incoming_call_url_callback_timeout** (*str*) – Determine how long should the platform wait for inconmingCallUrl's response before timing out in milliseconds.
- **incoming_call_fallback_url** (*str*) – The URL used to send the callback event if the request to incomingCallUrl fails.
- **incoming_message_url** (*str*) – A URL where message events will be sent for an inbound SMS message
- **incoming_message_url_callback_timeout** (*str*) – Determine how long should the platform wait for incomingMessageUrl's response before timing out in milliseconds.
- **incoming_message_fallback_url** (*str*) – The URL used to send the callback event if the request to incomingMessageUrl fails.
- **callback_http_method** (*str*) – Determine if the callback event should be sent via HTTP GET or HTTP POST. (If not set the default is HTTP POST)
- **auto_answer** (*str*) – Determines whether or not an incoming call should be automatically answered. Default value is 'true'.

Return type str

Returns id of created application

Example: Update Existing Application:

```
my_app_id = api.create_application( name                  = "MyFirstApp",
                                    incoming_call_url     = "http://callback.com/calls",
                                    incoming_message_url  = "http://callback.com/messages",
                                    callback_http_method = "GET")

print(my_app_id)
## a-1232asf123

my_app = api.get_application(my_app_id)
print(my_app)
{   'auto_answer'        : True,
    'callbackHttpMethod': 'get',
    'id'                 : 'a-1232asf123',
```

```
        'incomingCallUrl'    : 'http://callback.com/calls',
        'incomingMessageUrl': 'http://callback.com/messages',
        'incomingSmsUrl'     : 'http://callback.com/messages',
        'name'               : 'MyFirstApp'
    }

    api.update_application(my_app_id, name = "My Updated App")

    my_app = api.get_application(my_app_id)
    print(my_app)
    {    'autoAnswer'         : True,
        'callbackHttpMethod': 'get',
        'id'                 : 'a-1232asf123',
        'incomingCallUrl'    : 'http://callback.com/calls',
        'incomingMessageUrl': 'http://callback.com/messages',
        'incomingSmsUrl'     : 'http://callback.com/messages',
        'name'               : 'My Updated App'
    }
```

get_application

Client.**get_application**(*app_id*)

Gets information about an application

Parameters **app_id** (*str*) – id of an application

Return type dict

Returns application information

Example: Fetch single application:

```
    my_app = api.get_application(my_app_id)
    print(my_app)
    ## {    'auto_answer': True,
    ##      'callback_http_method': 'get',
    ##      'id': 'a-1232asf123',
    ##      'incoming_call_url': 'http://callback.com/calls',
    ##      'incoming_message_url': 'http://callback.com/messages',
    ##      'incoming_sms_url': 'http://callback.com/messages',
    ##      'name': 'MyFirstApp2'
    ## }

    print(my_app["id"])
    ## a-1232asf123
```

delete_application

Client.**delete_application**(*app_id*)

Remove an application

Parameters **app_id** (*str*) – id of an application

Example: Delete single application:

```
    api.delete_application('a-appId')
```

```
try :
    api.get_application('a-appId')
except CatapultException as err:
    print(err.message)
## The application a-appId could not be found
```

Available Numbers

search_available_local_numbers

Client.**search_available_local_numbers**(*city=None*, *state=None*, *zip_code=None*, *area_code=None*, *local_number=None*, *in_local_calling_area=None*, *quantity=None*, *pattern=None*, ***kwargs*)

Searches for available local or toll free numbers.

Parameters

- **city** (*str*) – A city name
- **state** (*str*) – A two-letter US state abbreviation
- **zip_code** (*str*) – A 5-digit US ZIP code
- **area_code** (*str*) – A 3-digit telephone area code
- **local_number** (*str*) – It is defined as the first digits of a telephone number inside an area code for filtering the results. It must have at least 3 digits and the areaCode field must be filled.
- **in_local_calling_area** (*str*) – Boolean value to indicate that the search for available numbers must consider overlayed areas.
- **quantity** (*int*) – The maximum number of numbers to return (default 10, maximum 5000)
- **pattern** (*str*) – A number pattern that may include letters, digits, and the wildcard characters

Return type list

Returns list of numbers

Example: Search for 3 910 numbers:

```
numbers = api.search_available_local_numbers(area_code = '910', quantity = 3)

print(numbers)

## [   {   'city'          : 'WILMINGTON',
##         'national_number': '(910) 444-0230',
##         'number'        : '+19104440230',
##         'price'         : '0.35',
##         'rate_center'    : 'WILMINGTON',
##         'state'         : 'NC'},
##     {   'city'          : 'WILMINGTON',
##         'national_number': '(910) 444-0263',
##         'number'        : '+19104440263',
##         'price'         : '0.35',
##         'rate_center'    : 'WILMINGTON',
##         'state'         : 'NC'},
```

```
##     {   'city'          : 'WILMINGTON',
##         'national_number': '(910) 444-0268',
##         'number'        : '+19104440268',
##         'price'         : '0.35',
##         'rate_center'    : 'WILMINGTON',
##         'state'         : 'NC'}
## ]

print(numbers[0]["number"])
## +19104440230
```

search_available_toll_free_numbers

Client.**search_available_toll_free_numbers**(*quantity=None*, *pattern=None*, ***kwargs*)

Searches for available local or toll free numbers.

Parameters

- **quantity** (*int*) – The maximum number of numbers to return (default 10, maximum 5000)
- **pattern** (*str*) – A number pattern that may include letters, digits, and the wildcard characters

Return type list

Returns list of numbers

Example: Search for 3 toll free numbers with pattern 456:

```
numbers = api.search_available_toll_free_numbers(pattern = '*456', quantity = 3)

print(numbers)

## [   {   'national_number': '(844) 489-0456',
##         'number'        : '+18444890456',
##         'pattern_match'  : '           456',
##         'price'         : '0.75'},
##     {   'national_number': '(844) 498-2456',
##         'number'        : '+18444982456',
##         'pattern_match'  : '           456',
##         'price'         : '0.75'},
##     {   'national_number': '(844) 509-4566',
##         'number'        : '+18445094566',
##         'pattern_match'  : '          456 ',
##         'price'         : '0.75'}]

print(numbers[0]["number"])
## +18444890456
```

search_and_order_local_numbers

Client.**search_and_order_local_numbers**(*city=None*, *state=None*, *zip_code=None*, *area_code=None*, *local_number=None*, *in_local_calling_area=None*, *quantity=None*, ***kwargs*)

Searches and orders for available local numbers.

Parameters

- **city** (*str*) – A city name
- **state** (*str*) – A two-letter US state abbreviation
- **zip_code** (*str*) – A 5-digit US ZIP code
- **area_code** (*str*) – A 3-digit telephone area code
- **local_number** (*str*) – It is defined as the first digits of a telephone number inside an area code for filtering the results. It must have at least 3 digits and the areaCode field must be filled.
- **in_local_calling_area** (*str*) – Boolean value to indicate that the search for available numbers must consider overlayed areas.
- **quantity** (*int*) – The maximum number of numbers to return (default 10, maximum 5000)

Return type list

Returns list of ordered numbers

Example: Search _and_ order a single number:

```
ordered_numbers = api.search_and_order_available_numbers(zip = '27606', quantity = 1)

print(ordered_numbers)

## [   {   'city'          : 'RALEIGH',
##         'id'            : 'n-abc',
##         'location'      : 'https://api.catapult.inetwork.com/v1/users/u-12/phoneNumbers/n-abc
##         'national_number': '(919) 222-4444',
##         'number'        : '+19192224444',
##         'price'         : '0.35',
##         'state'         : 'NC'}]
```

search_and_order_toll_free_numbers

Client.**search_and_order_toll_free_numbers**(*quantity*, ***kwargs*)

Searches for available local or toll free numbers.

Query parameters for toll free numbers :param int quantity: The maximum number of numbers to return (default 10, maximum 5000)

Return type list

Returns list of numbers

Example: Search then order a single toll-free number:

```
numbers = api.search_and_order_toll_free_numbers(quantity = 1)

print(numbers)

## [   {   'location'      : 'https://api.catapult.inetwork.com/v1/users/u-123/phoneNumbers/n-ab
##         'national_number': '(844) 484-1048',
##         'number'        : '+18444841048',
##         'price'         : '0.75'}]

print(numbers[0]["number"])
## +18444841048
```

Domains

list_domains

Client.**list_domains**(*size=None*, ***kwargs*)

Get a list of domains

Parameters **size** (*int*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 100)

Return type types.GeneratorType

Returns list of domains

Example: Fetch domains and print:

```
domain_list = api.list_domains(size=10)
print(list(domain_list))

## [{   'endpoints_url': 'https://api.catapult.inetwork.com/v1/users/u-abc123/domains/endpoints'
##      'id'           : 'rd-domainId',
##      'name'         : 'siplearn1'},
## {   'endpoints_url' : 'https://api.catapult.inetwork.com/v1/users/u-abc123/domains/endpoints'
##      'id'           : 'rd-domainId2',
##      'name'         : 'siplearn2'}]
```

Example: Search for domain based on name:

```
domain_list = api.list_domains(size=100)

domain_name = ''

while domain_name != 'My Prod Site':
    my_domain = next(domain_list)
    domain_name = my_domain['name']

print(my_domain)
## {   'description' : 'Python Docs Example',
##      'endpoints_url': 'https://api.catapult.inetwork.com/v1/users/u-abc123/domains/rd-domainId
##      'id'          : 'rd-domainId',
##      'name'        : 'My Prod Site'}
```

create_domain

Client.**create_domain**(*name*, *description=None*, ***kwargs*)

Create a domain

Parameters

- **name** (*str*) – The name is a unique URI to be used in DNS lookups
- **description** (*str*) – String to describe the domain

Return type str

Returns id of created domain

Example: Create Domain:

```
    domain_id = api.create_domain(name='qwerty', description='Python Docs Example')

    print(domain_id)
    # rd-domainId
```

get_domain

Client.**get_domain**(*domain_id*)

Get information about a domain

Parameters **domain_id** (*str*) – id of the domain

Return type dict

Returns domain information

Example: Create then fetch domain:

```
    domain_id = api.create_domain(name='qwerty', description='Python Docs Example')

    print(domain_id)
    # rd-domainId

    my_domain = api.get_domain(domain_id)

    print(my_domain)
    ## {   'description' : 'Python Docs Example',
    ##     'endpoints_url': 'https://api.catapult.inetwork.com/v1/users/u-abc123/domains/rd-domainId
    ##     'id'          : 'rd-domainId',
    ##     'name'        : 'qwerty'}
```

delete_domain

Client.**delete_domain**(*domain_id*)

Delete a domain

Parameters **domain_id** (*str*) – id of a domain

Example: Delete domain 'domainId':

```
    api.delete_domain('domainId')
```

Endpoints

list_domain_endpoints

Client.**list_domain_endpoints**(*domain_id*, *size=None*, ***kwargs*)

Get a list of domain's endpoints

Parameters

- **domain_id** (*str*) – id of a domain
- **size** (*int*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 1000)

Return type types.GeneratorType

Returns list of endpoints

Example: List and iterate over:

```
endpoint_list = api.list_domain_endpoints('rd-domainId', size=1000)

for endpoint in endpoint_list:
    print(endpoint['id'])
##re-endpointId1
##re-endpointId2
```

Example: List and print all:

```
endpoint_list = api.list_domain_endpoints('rd-domainId', size=1000)

print(list(endpoint_list))

## [
##     {
##         'application_id':'a-appId',
##         'credentials'  :{
##             'realm'    :'creds.bwapp.bwsip.io',
##             'username' :'user1'
##         },
##         'description'  :"Your SIP Account",
##         'domain_id'     :'rd-domainId',
##         'enabled'      :True,
##         'id'           :'re-endpointId1',
##         'name'         :'User1_endpoint',
##         'sip_uri'       :'sip:user1@creds.bwapp.bwsip.io'
##     },
##     {
##         'application_id':'a-appId',
##         'credentials'  :{
##             'realm'    :'creds1.bwapp.bwsip.io',
##             'username' :'user2'
##         },
##         'description'  :"Your SIP Account",
##         'domain_id'     :'rd-domainId',
##         'enabled'      :True,
##         'id'           :'re-endpointId2',
##         'name'         :'User2_endpoint',
##         'sip_uri'       :'sip:user2@creds.bwapp.bwsip.io'
##     }
## ]
```

create_domain_endpoint

Client.**create_domain_endpoint**(*domain_id*, *name*, *password*, *description=None*, *application_id=None*, *enabled=True*, ***kwargs*)

Create a domain endpoint

Parameters

- **domain_id** (*str*) – id of a domain
- **name** (*str*) – The name of endpoint
- **description** (*str*) – String to describe the endpoint

- **application_id** (*str*) – Id of application which will handle calls and messages of this endpoint
- **enabled** (*bool*) – When set to true, SIP clients can register as this device to receive and make calls. When set to false, registration, inbound, and outbound calling will not succeed.
- **password** (*str*) – Password of created SIP account

Return type str

Returns id of endpoint

Example: Create Endpoint on Domain 'rd-domainId':

```
endpoint_id = api.create_domain_endpoint('rd-domainId',
                                         endpoint_name='User3_endpoint',
                                         password='AtLeast6Chars')
print(endpoint_id)
# re-endpointId3

my_endpoint = api.get_domain_endpoint(endpoint_id)
print(my_endpoint)

## {
##     'credentials' :{
##         'realm'   :'qwerty.bwapp.bwsip.io',
##         'username':'User3_endpoint'
##     },
##     'domain_id'    :'rd-domainId',
##     'enabled'     :True,
##     'id'          :'re-endpointId3',
##     'name'        :'User3_endpoint',
##     'sip_uri'      :'sip:user5@qwerty.bwapp.bwsip.io'
## }
```

get_domain_endpoint

Client.**get_domain_endpoint**(*domain_id*, *endpoint_id*)

Get information about an endpoint

Parameters

- **domain_id** (*str*) – id of a domain
- **endpoint_id** (*str*) – id of a endpoint

Return type dict

Returns call information

Example: Create Endpoint on Domain 'rd-domainId' then fetch the endpoint:

```
endpoint_id = api.create_domain_endpoint('rd-domainId',
                                         endpoint_name='User3_endpoint',
                                         password='AtLeast6Chars')
print(endpoint_id)
# re-endpointId3

my_endpoint = api.get_domain_endpoint(endpoint_id)
print(my_endpoint)
```

```
## {
##     'credentials' :{
##         'realm'   :'qwerty.bwapp.bwsip.io',
##         'username':'User3_endpoint'
##     },
##     'domain_id'    :'rd-domainId',
##     'enabled'     :True,
##     'id'          :'re-endpointId3',
##     'name'        :'User3_endpoint',
##     'sip_uri'      :'sip:user5@qwerty.bwapp.bwsip.io'
## }
```

update_domain_endpoint

Client.**update_domain_endpoint**(*domain_id*, *endpoint_id*, *password=None*, *description=None*, *application_id=None*, *enabled=None*, ***kwargs*)

Update information about an endpoint

Parameters

- **domain_id** (*str*) – id of a domain
- **endpoint_id** (*str*) – id of a endpoint
- **description** (*str*) – String to describe the endpoint
- **application_id** (*str*) – Id of application which will handle calls and messages of this endpoint
- **enabled** (*bool*) – When set to true, SIP clients can register as this device to receive and make calls. When set to false, registration, inbound, and outbound calling will not succeed.
- **password** (*str*) – Password of created SIP account

Example: Update password and disable the endpoint:

```
my_endpoint = api.get_domain_endpoint('rd-domainId', 're-endpointId')
print(my_endpoint)

## {
##     'credentials' :{
##         'realm'   :'qwerty.bwapp.bwsip.io',
##         'username':'user5'
##     },
##     'domain_id'    :'rd-domainId',
##     'enabled'     :True,
##     'id'          :'re-endpointId',
##     'name'        :'user3',
##     'sip_uri'      :'sip:user5@qwerty.bwapp.bwsip.io'
## }

api.update_domain_endpoint('rd-domainId', 're-endpointId', enabled=False, password='abc123')
my_endpoint = api.get_domain_endpoint('rd-domainId', 're-endpointId')
print(my_endpoint)

## {
##     'credentials' :{
##         'realm'   :'qwerty.bwapp.bwsip.io',
##         'username':'user5'
##     },
```

```
    ##     'domain_id'    :'rd-domainId',
    ##     'enabled'     :False,
    ##     'id'          :'re-endpointId',
    ##     'name'        :'user3',
    ##     'sip_uri'      :'sip:user5@qwerty.bwapp.bwsip.io'
    ## }
```

delete_domain_endpoint

Client.**delete_domain_endpoint**(*domain_id*, *endpoint_id*)

Remove an endpoint

Parameters

- **domain_id** (*str*) – id of a domain
- **endpoint_id** (*str*) – id of a endpoint

Example: Delete and try to fetch endpoint:

```
my_endpoint = api.get_domain_endpoint('rd-domainId', 're-endpointId')
print(my_endpoint)
## {
##     'credentials' :{
##         'realm'   :'qwerty.bwapp.bwsip.io',
##         'username':'user5'
##     },
##     'domain_id'    :'rd-domainId',
##     'enabled'     :False,
##     'id'          :'re-endpointId3ndpointId',
##     'name'        :'user3',
##     'sip_uri'      :'sip:user5@qwerty.bwapp.bwsip.io'
## }
api.delete_domain_endpoint(d, e)

try:
    my_endpoint = api.get_domain_endpoint(d, e)
except Exception as e:
    print(e)
## CatapultException(404, "The endpoint 're-endpointId' could not be found")
```

create_domain_endpoint_auth_token

Client.**create_domain_endpoint_auth_token**(*domain_id*, *endpoint_id*, *expires=3600*, ***kwargs*)

Create auth token for an endpoint

Parameters

- **domain_id** (*str*) – id of a domain
- **endpoint_id** (*str*) – id of a endpoint
- **expires** (*int*) – Duration of valid token.

Example: Create token:

```
    token = api.create_domain_endpoint_auth_token('domainId', 'endpointId', 5000)
```

Errors

list_errors

Client.**list_errors**(*size=None*, ***kwargs*)

Get a list of errors

Parameters **size** (*int*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 1000)

Return type types.GeneratorType

Returns list of calls

Example: List all errors:

```
    error_list = api.list_errors()

    print(list(error_list))

    # [{
    #     'category':'unavailable',
    #     'code'    :'number-allocator-unavailable',
    #     'details':[
    #         {
    #             'id'   :'ued-eh3zn3dxgiin4y',
    #             'name' :'requestPath',
    #             'value':'availableNumbers/local'
    #         },
    #         {
    #             'id'   :'ued-3fsdqiq',
    #             'name' :'remoteAddress',
    #             'value':'216.82.234.65'
    #         },
    #         {
    #             'id'   :'ued-2r4t47bwi',
    #             'name' :'requestMethod',
    #             'value':'GET'
    #         }
    #     ],
    #     'id'     :'ue-upvfv53xzca',
    #     'message':'Cannot connect to the number allocator',
    #     'time'   :'2016-03-28T18:31:33Z'
    # },
    # {
    #     'category':'unavailable',
    #     'code':'number-allocator-unavailable',
    #     'details':[
    #         {
    #             'id':'ued-kntwx7vyotalci',
    #             'name':'requestPath',
    #             'value':'availableNumbers/local'
    #         },
    #         {
    #             'id':'ued-b24vxpfskldq',
    #             'name':'remoteAddress',
```

```
    #             'value':'216.82.234.65'
    #         },
    #         {
    #             'id':'ued-ww5rcgl7zm2ydi',
    #             'name':'requestMethod',
    #             'value':'GET'
    #         }
    #     ],
    #     'id':'ue-pok2vg7kyuzaqq',
    #     'message':'Cannot connect to the number allocator',
    #     'time':'2016-03-28T18:31:33Z'
    # }]
```

get_error

`Client.get_error(error_id)`

Get information about an error

Parameters **id** – id of an error

Return type dict

Returns error information

Example: Get information of specific error:

```
    error = api.get_error('ue-errorId')
    print(error)

    ## {
    ##     'category':'unavailable',
    ##     'code'    :'number-allocator-unavailable',
    ##     'details' :[
    ##         {
    ##             'id'      :'ued-kntvyotalci',
    ##             'name'    :'requestPath',
    ##             'value'   :'availableNumbers/local'
    ##         },
    ##         {
    ##             'id'      :'ued-b2dq',
    ##             'name'    :'remoteAddress',
    ##             'value'   :'216.82.234.65'
    ##         },
    ##         {
    ##             'id'      :'ued-wzm2ydi',
    ##             'name'    :'requestMethod',
    ##             'value'   :'GET'
    ##         }
    ##     ],
    ##     'id'      :'ue-errorId',
    ##     'message' :'Cannot connect to the number allocator',
    ##     'time'    :'2016-03-28T18:31:33Z'
    ## }
```

Media

list_media_files

Client.**list_media_files**()

Gets a list of user's media files.

Return type types.GeneratorType

Returns list of media files

Example: list media files and save any with the name *dog* in file name:

```
media_list = api.list_media_files()
for media in media_list:
    if 'dog' in media['media_name'].lower():
        stream, content_type = api.download_media_file(media['media_name'])
        with io.open(media['media_name'], 'wb') as file:
            file.write(stream.read())
```

upload_media_file

Client.**upload_media_file**(*media_name*, *content=None*, *content_type='application/octet-stream'*, *file_path=None*)

Upload a file

Parameters

- **media_name** (*str*) – name of file on bandwidth server
- **content** (*str|buffer|bytearray|stream|file*) – content of file to upload (file object, string or buffer). Don't use together with file_path
- **content_type** (*str*) – mime type of file
- **file_path** (*str*) – path to file to upload. Don't use together with content

Example: Upload text file:

```
api.upload_media_file('file1.txt', 'content of file', 'text/plain')

# with file path
api.upload_media_file('file1.txt', file_path='/path/to/file1.txt')
```

download_media_file

Client.**download_media_file**(*media_name*)

Download a file

Parameters **media_name** (*str*) – name of file on bandwidth server

:rtype (stream, str) :returns stream to file to download and mime type

Example: list media files and save any with the name *dog* in file name:

```
media_list = api.get_media_files()
for media in media_list:
    if 'dog' in media['media_name'].lower():
        stream, content_type = api.download_media_file(media['media_name'])
        with io.open(media['media_name'], 'wb') as file:
            file.write(stream.read())
```

Numbers

get_number_info

Client.**get_number_info**(*number*)

Gets CNAM information about phone number

Parameters **number** (*str*) – phone number to get information

Return type dict

Returns CNAM information

Example: Get Number information:

```
data = api.get_number_info('+1234567890')
print(data)
## {   'created': '2017-02-10T09:11:50Z',
##     'name'       : 'RALEIGH, NC',
##     'number'     : '+1234567890',
##     'updated'    : '2017-02-10T09:11:50Z'}
```

list_phone_numbers

Client.**list_phone_numbers**(*application_id=None*, *state=None*, *name=None*, *city=None*, *number_state=None*, *size=None*, ***kwargs*)

Get a list of user's phone numbers

Parameters

- **application_id** (*str*) – Used to filter the retrieved list of numbers by an associated application ID.
- **state** (*str*) – Used to filter the retrieved list of numbers allocated for the authenticated user by a US state.
- **name** (*str*) – Used to filter the retrieved list of numbers allocated for the authenticated user by it's name.
- **city** (*str*) – Used to filter the retrieved list of numbers allocated for the authenticated user by it's city.
- **number_state** (*str*) – Used to filter the retrieved list of numbers allocated for the authenticated user by the number state.
- **size** (*str*) – Used for pagination to indicate the size of each page requested for querying a list of items. If no value is specified the default value is 25. (Maximum value 1000)

Return type types.GeneratorType

Returns list of phone numbers

Example: List all phone numbers:

```
number_list = api.list_phone_numbers(size=1000)
print(list(number_list))
## [
##     {
##         'city'          :'RALEIGH',
##         'created_time'   :'2017-02-06T18:41:37Z',
##         'id'            :'n-n123',
```

```
    ##         'name'          :'demo name',
    ##         'national_number':'(919) 555-5346',
    ##         'number'        :'+19195555346',
    ##         'number_state'   :'enabled',
    ##         'price'         :'0.35',
    ##         'state'         :'NC'
    ##     },
    ##     {
    ##         'city'          :'RALEIGH',
    ##         'created_time'   :'2017-02-06T18:41:56Z',
    ##         'id'            :'n-n1234',
    ##         'name'          :'demo name',
    ##         'national_number':'(919) 555-5378',
    ##         'number'        :'+19195555378',
    ##         'number_state'   :'enabled',
    ##         'price'         :'0.35',
    ##         'state'         :'NC'
    ##     }
    ## ]
```

order_phone_number

Client.**order_phone_number**(*number=None*, *name=None*, *application_id=None*, *fallback_number=None*, ***kwargs*)

Allocates a number so user can use it to make and receive calls and send and receive messages.

Parameters

- **number** (*str*) – An available telephone number you want to use
- **name** (*str*) – A name you choose for this number.
- **application_id** (*str*) – The unique id of an Application you want to associate with this number.
- **fallback_number** (*str*) – Number to transfer an incoming call when the callback/fallback events can't be delivered.

Return type str

Returns id of created phone number

Example: Order Number:

```
number_id = api.order_phone_number(number='+1234567890')
print(number_id)
# n-asdf123
```

get_phone_number

Client.**get_phone_number**(*number_id*)

Get information about a phone number

Parameters **number_id** (*str*) – id of a phone number

Return type dict

Returns number information

Example: Search, order, and fetch Number information:

```
available_numbers = api.search_available_local_numbers(city='Raleigh', state='NC')

number_id = api.order_phone_number(available_numbers[0]['number'])
print(number_id)
# n-123

my_number = api.get_phone_number(number_id)
print(my_number)
## {
##     'city'           :'RALEIGH',
##     'created_time'    :'2017-02-06T18:27:14Z',
##     'id'             :'n-123',
##     'national_number':'(919) 561-5039',
##     'number'         :'+19195615039',
##     'number_state'    :'enabled',
##     'price'          :'0.35',
##     'state'          :'NC'
## }
```

update_phone_number

Client.**update_phone_number**(*number_id*, *name=None*, *application_id=None*, *fallback_number=None*, ***kwargs*)

Update information about a phone number

Parameters

- **number_id** (*str*) – id of a phone number
- **name** (*str*) – A name you choose for this number.
- **application_id** (*str*) – The unique id of an Application you want to associate with this number.
- **fallback_number** (*str*) – Number to transfer an incoming call when the callback/fallback events can't be delivered.

Example: Update number information:

```
my_number = api.get_phone_number(number_id)
print(my_number)
## {
##     'city'           :'RALEIGH',
##     'created_time'    :'2017-02-06T18:27:14Z',
##     'id'             :'n-123',
##     'national_number':'(919) 561-5039',
##     'number'         :'+19195615039',
##     'number_state'    :'enabled',
##     'price'          :'0.35',
##     'state'          :'NC'
## }

api.update_phone_number(number_id, name='demo name')

my_number = api.get_phone_number(number_id)
print(my_number)
## {
```

```
    ##     'id'            :'n-123',
    ##     'number'        :'+19195615039',
    ##     'national_number':'(919) 561-5039',
    ##     'name'          :'demo name',
    ##     'created_time'   :'2017-02-06T18:41:56Z',
    ##     'city'          :'RALEIGH',
    ##     'state'         :'NC',
    ##     'price'         :'0.35',
    ##     'number_state'   :'enabled'
    ## }
```

delete_phone_number

Client.**delete_phone_number**(*number_id*)

Remove a phone number

Parameters **number_id** (*str*) – id of a phone number

Example: Delete phone number (release) from account:

```
api.delete_phone_number('numberId')
```

4.5 Test Suite

To run the test suite use the provided commands in the makefile:

```
make req && make test
```

or:

```
make req && make local_test
```

4.6 Contributing and Reporting Bugs

4.6.1 Submitting Bug Reports

To file a bug report or request a feature submit a new issue to our GitHub Issue Tracker

4.6.2 Contributing

Fork our GitHub repository , create a new topic branch with your feature/fix, and send pull request with a comment that describes your changes.

All pull requests should pass our test suite and include tests for any new features/fixes. More information about the test suite can be found here

In addition to adding tests, pull requests should include updated documentation. Documentation is located in the /docs/source directory. Scripts are included to generate html documentation locally.

On linux systems:

```
cd python-bandwidth
make html_docs
```

On Windows:

```
cd python-bandwidth/docs
make.bat html
```

For more information on the format of the docs refer to the official Sphinx Docs

CHAPTER 5

Indices and tables

- genindex
- modindex

M

O

P

R

S

T

U