



BreakingPoint Cloud

Microsoft Azure DDoS Protection Validation

API Reference Guide

Release 1.1.0

Notices

Copyright Notice

© Keysight Technologies 2018

No part of this document may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Keysight Technologies, Inc. as governed by United States and international copyright laws.

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Keysight disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Keysight shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Keysight and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

U.S. Government Rights

The Software is "commercial computer software," as defined by Federal Acquisition Regulation ("FAR") 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement ("DFARS") 227.7202, the U.S. government

acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at

<http://www.keysight.com/find/sweula> or <https://support.ixiacom.com/support-services/warranty-license-agreements>.

The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data. 52.227-14 (June 1987) or DFAR 252.227-7015 (b) (2) (November 1995), as applicable in any technical data.

EULA

Use, download, and access to the BreakingPoint Cloud API is covered by the *Keysight Software End-User License Agreement* (found at www.keysight.com/find/sweula) and the terms contained in the *API Reference Guide*.

Contact Us

Ixia headquarters

26601 West Agoura Road
Calabasas, California 91302
+1 877 367 4942 – Toll-free North America
+1 818 871 1800 – Outside North America
+1.818.871.1805 – Fax
www.ixiacom.com/contact/info

Support

Global Support	+1 818 595 2599	support@ixiacom.com
APAC Support	+91 80 4939 6410	support-asiapac@ixiacom.com
EMEA Support	+40 21 301 5699	support-emea@ixiacom.com
Greater China Region	+400 898 0598	support-china@ixiacom.com
Hong Kong	+86 10 5732 3932	support-china@ixiacom.com
India Office	+91 80 4939 6410	support-india@ixiacom.com
Japan Head Office	+81 3 5326 1980	support-japan@ixiacom.com
Korea Office	+82 2 3461 0095	support-korea@ixiacom.com
Singapore Office	+656 494 8910	support-asiapac@ixiacom.com

CONTENTS

Contact Us	iii
Chapter 1 Introduction	1
Requirements	2
Download and install	3
Chapter 2 Functional organization	4
Chapter 3 Code samples	5
Connect to API	6
Manage Azure subscriptions	7
Start DDoS test	8
Stop a DDoS test	10
Get test profiles	11
Get billing data	12
Get data usage information	13
Increase data threshold	14
Get live test statistics	15
Get test statistics	16
Get test history	17
Chapter 4 API reference	19
Session object	20
Session properties	21
add_subscription()	23
start_test()	24

Test object	26
Test properties	27
test.refresh()	31
test.stop()	32
Profile object	33
Profile properties	34
validate_size()	35
Subscription object	36
Subscription properties	37
subscription.delete()	39
Billing object	40
get_cost()	41
get_stats()	42
Data Usage object	43
Data Usage common properties	44
Trial Mode object	45
Paid Mode object	47
INDEX	49

Chapter 1 Introduction

The BreakingPoint Cloud API provides functionality to:

- connect to BreakingPoint Cloud DDoS
- configure and run tests
- manage Azure subscriptions
- manage billing

It is provided in the form of a Python wrapper.

Topics in this section

Requirements	2
Download and install	3

See also the *User Guide* for a detailed description of the BreakingPoint Cloud web-based UI.

Requirements

The requirements for using the BreakingPoint Cloud API are as follows:

System and OS Requirements

- Personal computer running a Unix/Linux, OS X, or Windows (XP or later) operating system.
- HTTP connectivity between the personal computer and BreakingPoint Cloud.

Software Requirements

- Python (either of the following):
 - Python 2.7.14 or later version of 2.7 (<http://www.python.org/getit/>)
 - Python 3.6.4 or later version of 3.6 (<http://www.python.org/getit/>)
- OpenSSL 1.0.1 or later (<https://www.openssl.org/source/>)
- requests[security] 2.18.4 or later (<https://pypi.org/project/requests/>)
- requests-aws4auth 0.9 or later (<https://pypi.python.org/pypi/requests-aws4auth>)
- boto3 1.5.5 or later (<https://pypi.python.org/pypi/boto3>)

Download and install

To download and install the BreakingPoint Cloud Python API client:

1. From the gear menu (⚙️), click **Python API Client**.
The API client is immediately downloaded to your downloads folder.
2. Go to your downloads folder.
3. Open a console and execute:

```
pip install bpcddos.zip
```
4. To verify that the package is installed, enter the following commands:

```
python
>>> from ixia.bpcddos.session import Session
>>>
```

If the import works, then the API is properly installed and available for immediate use.

Chapter 2 Functional organization

The BreakingPoint Cloud API is organized as a set of objects accessible from the *Session* class. Each Python script using the API requires a *session* instance that:

1. authenticates the user of the BreakingPoint Cloud subscription
2. provides access to other BreakingPoint Cloud API objects that can perform specific tasks

Structure

The API is structured according the following hierarchy:

Session

Management

Subscriptions: Manages the list of Azure subscriptions that have been added to the user's BreakingPoint Cloud account.

Billing: Provides access to billing data.

Data Usage: Provides access to user account data usage (and data threshold settings for non-trial users).

Testing

Start and Stop: Starts and stops tests.

List: Lists the tests executed on the user's BreakingPoint Cloud account.

Profile (test profiles): Defines the attributes of a test (DDoS profile, test size).

Chapter 3 Code samples

This section provides a series of BreakingPoint Cloud API code samples, each of which demonstrates the coding for one of the actions that can be included in your test automation projects.

Code samples:

Connect to API	6
Manage Azure subscriptions	7
Start DDoS test	8
Stop a DDoS test	10
Get test profiles	11
Get billing data	12
Get data usage information	13
Increase data threshold	14
Get live test statistics	15
Get test statistics	16
Get test history	17

Connect to API

To connect to the API, import the `Session` module and initiate a session using the email address and password associated with your BreakingPoint Cloud subscription. For example:

```
from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)
```

Manage Azure subscriptions

Use `session.add_subscription` and `subscription.delete` to add Azure subscriptions to and remove Azure subscriptions from your BreakingPoint Cloud account.

! Important! To validate a newly-added Azure subscription, you need to either log in to the web-based UI and follow the steps outlined in the online BreakingPoint Cloud *User Guide*; or alternatively, open the `Subscription.authorization_uri` into a browser.

The following code sample demonstrates how to add and remove subscriptions:

```
from ixia.bpcddos.session import Session

# constants
__USER_EMAIL      = 'username@example.com'
__PASSWORD        = 'password'

__AZURE_SUBSCRIPTION_ID = 'valid-azure-subscription_id'
__ACCESS_URL_MESSAGE = 'Access this url to activate the subscription: {0}'

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    # add/replace subscription
    new_sub = session.add_subscription(__AZURE_SUBSCRIPTION_ID)
    sub_activation_url = new_sub.authorization_uri
    print(__ACCESS_URL_MESSAGE.format(sub_activation_url))

    # delete subscription
    subscription = [
        sub for sub in session.subscriptions if sub.id == __AZURE_SUBSCRIPTION_ID
    ][0]
    subscription.delete()
```

Start DDoS test

Use the `session.start_test` method to start a test. The following code sample demonstrates how to define the properties for and start a DDoS test:

```
import time

from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

__POLL_TIME     = 15

# valid public ip
__TARGET_IP     = 'valid-public-ip-in-your-azure-subscription'
__TARGET_PORT   = 80
__PROFILE_ID    = 'excessive-post'
__PROFILE_SIZE  = 'small'
__TEST_DURATION = 600

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    # start test using string params
    new_test = session.start_test(
        __TARGET_IP, __TARGET_PORT,
        __PROFILE_ID, __PROFILE_SIZE, __TEST_DURATION
    )
    # wait for the test to finish:
    while not new_test.stopped:
        time.sleep(__POLL_TIME)

    # start test using profile object params:
    profiles = session.profiles
    new_test = session.start_test(
        __TARGET_IP, __TARGET_PORT,
        profiles[0], profiles[0].sizes[0], __TEST_DURATION
    )
    # wait for the test to finish
    while not new_test.stopped:
        time.sleep(__POLL_TIME)

    # start test using mixed params:
    new_test = session.start_test(
```

```
        __TARGET_IP, __TARGET_PORT,  
        profiles[0], __PROFILE_SIZE, __TEST_DURATION  
    )  
    # wait for the test to finish  
    while not new_test.stopped:  
        time.sleep(__POLL_TIME)
```

Stop a DDoS test

Use `test.stop` to stop a test that is running. The following code sample demonstrates how to use `test.stop` to stop a running test (interrupt its run):

```
import time

from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

# valid public ip
__TARGET_IP     = 'valid-public-ip-in-your-azure-subscription'
__TARGET_PORT   = 80
__PROFILE_NAME   = 'HTTP Excessive Get'
__TEST_SIZE     = 'small'
__TEST_DURATION  = 600
__TEST_WAIT     = 10

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    # start test using string parameters
    new_test = session.start_test(
        __TARGET_IP, __TARGET_PORT,
        __PROFILE_NAME, __TEST_SIZE, __TEST_DURATION
    )
    time.sleep(__TEST_WAIT)

    # stop test
    new_test.stop()

    # wait until the test is stopped
    while not new_test.stopped:
        time.sleep(__TEST_WAIT)
```

Get test profiles

The following code sample demonstrates how to use `session.profiles` to get test profile data:

```
from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

__PROFILE_NAME = 'HTTP Excessive Get'

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    # get test profiles
    profiles = session.profiles

    # List profile names and ids:
    for p in profiles:
        print(p.name, p.id)

    # find profile
    httpProfile = [p for p in profiles if p.name == __PROFILE_NAME][0]
```


Get billing data

Use `session.billing` to get billing statistics and costs for one or more billing cycles. The following code sample gets the billing data for the last billing cycle as well as for the prior six billing cycles:

```
from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    # get billing data for the last billing cycle
    print(session.billing.get_stats(end_year=2018, end_month=1))
    print(session.billing.get_cost(end_year=2018, end_month=1))

    # get billing data for the previous billing cycles
    print(session.billing.get_stats(end_year=2018, end_month=1, billing_cycles=6))
    print(session.billing.get_cost(end_year=2018, end_month=1, billing_cycles=6))
```

Get data usage information

Use `session.data_usage` to get data usage information for the BreakingPoint Cloud user. The following code sample exits if the account is in *paid-mode*, and displays the current data usage values:

```
from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    data_usage = session.data_usage
    if not data_usage.trial_mode:
        print('Account is in paid mode')
        exit(1)

    # trial expired
    print(data_usage.trial_expired)

    # consumed data
    print(repr(data_usage.consumed))
    # pretty print
    print(data_usage.consumed)

    # remaining data
    print(repr(data_usage.remaining))
    # pretty print
    print(data_usage.remaining)

    # quota
    print(repr(data_usage.quota))
    # pretty print
    print(data_usage.quota)
```

Increase data threshold

Use `data_usage.increase_threshold()` to increase the current BreakingPoint Cloud data usage threshold. The following code sample exits if the account is in *trial-mode*. If the account is in *paid-mode*, it checks to see if there is no more than 1 GB left until the data threshold is reached and, if so, increases the threshold by 250GB:

```
from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    data_usage = session.data_usage
    if data_usage.trial_mode:
        print('Account is in trial mode')
        exit(1)

    # print consumed value in bytes
    print(repr(data_usage.consumed))
    # pretty print
    print(data_usage.consumed)

    # print threshold value in bytes
    print(repr(data_usage.threshold))
    # pretty print
    print(data_usage.threshold)

    # if there is 1 GB left until the data threshold is reached
    # increase the threshold
    if data_usage.threshold - data_usage.consumed < 1000000000:
        try:
            data_usage.increase_threshold()
        except Exception as e:
            # handle exception
            print(e.message)

    # print threshold value in bytes
    print(repr(data_usage.threshold))

    # pretty print
    print(data_usage.threshold)
```

Get live test statistics

The following code sample demonstrates how to use `test.stats` to get and print the statistics while the test is running:

```
import time

from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

__POLL_TIME     = 15

# valid public ip
__TARGET_IP     = 'valid-public-ip-in-your-azure-subscription'
__TARGET_PORT   = 80
__PROFILE_ID    = 'excessive-post'
__TEST_SIZE     = 'small'
__TEST_DURATION = 600

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    # start test using string params
    new_test = session.start_test(
        __TARGET_IP, __TARGET_PORT,
        __PROFILE_ID, __TEST_SIZE, __TEST_DURATION
    )

    # wait for the test to finish:
    while not new_test.stopped:
        # print test stats (auto-refresh) - live stats
        print(new_test.stats)
        time.sleep(__POLL_TIME)
```

Get test statistics

The following code sample demonstrates how to use `test.stats` to get and print the test statistics at the end of the test:

```
import time

from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

__POLL_TIME     = 15

# valid public ip
__TARGET_IP     = 'valid-public-ip-in-your-azure-subscription'
__TARGET_PORT   = 80
__PROFILE_ID    = 'excessive-post'
__TEST_SIZE     = 'small'
__TEST_DURATION = 600

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    # start test using string params
    new_test = session.start_test(
        __TARGET_IP, __TARGET_PORT,
        __PROFILE_ID, __TEST_SIZE, __TEST_DURATION
    )

    # see all test parameters & data
    print(new_test.data)

    # see test configuration
    print(new_test.config)

    # get test profile
    new_test_profile = new_test.profile
    print(new_test_profile.data)

    # wait for the test to finish:
    while not new_test.stopped:
        time.sleep(__POLL_TIME)

    # see test stats at the end of the test
    print(new_test.stats)
```

Get test history

The following code sample demonstrates how to display your test history:

```
import time

from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

__POLL_TIME     = 15

# valid public ip
__TARGET_IP     = 'valid-public-ip-in-your-azure-subscription'
__TARGET_PORT   = 80
__PROFILE_ID    = 'excessive-post'
__PROFILE_SIZE  = 'small'
__TEST_DURATION = 600

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    # start test using string params
    new_test = session.start_test(
        __TARGET_IP, __TARGET_PORT,
        __PROFILE_ID, __PROFILE_SIZE, __TEST_DURATION
    )
    test_id = new_test.ID
    print(new_test.Status)
    print(new_test.id)

    # wait for the test to finish:
    while not new_test.stopped:
        time.sleep(__POLL_TIME)

    # find your test:
    tests = session.tests
    old_test = [test for test in tests if test.id == test_id][0]

    # see all test parameters & data
    print(old_test.data)

    # see test configuration
    print(old_test.config)
```

```
# get test profile
old_test_profile = old_test.profile
print(old_test_profile.data)

# see test stats
print(old_test.stats)
```

Chapter 4 API reference

This section includes a complete reference for every public method and property associated with the objects provided by the BreakingPoint Cloud API.

Objects:

Session object	20
Session properties	21
add_subscription()	23
start_test()	24
Test object	26
Test properties	27
test.refresh()	31
test.stop()	32
Profile object	33
Profile properties	34
validate_size()	35
Subscription object	36
Subscription properties	37
subscription.delete()	39
Billing object	40
get_cost()	41
get_stats()	42
Data Usage object	43
Data Usage common properties	44
Trial Mode object	45
Paid Mode object	47

Session object

The *Session* object provides the methods and properties required for managing Azure subscriptions, running BreakingPoint Cloud DDoS tests, and billing.

Topics in this section:

Session properties	21
add_subscription()	23
start_test()	24

Session properties

Session objects have the following properties:

- test
- profiles
- subscriptions
- billing
- data_usage

Each is described below.

test	
Returns:	Test object (A list containing the previously executed tests and tests that are currently running from the account)
Usage:	<pre>session = Session('user@example.com', 'pass') tests = session.tests for test in tests: print(test.data)</pre>
Reference:	Test object on page 26

profile	
Returns:	Profile object (a list of available test profiles) The result is cached in the Session instance.
Usage:	<pre>session = Session('user@example.com', 'pass') profiles = session.profiles for profile in profiles: print(profile.id, profile.name, profile.sizes)</pre>
Reference:	Profile object on page 33

subscriptions	
Returns:	Subscription object (a list of Azure subscriptions associated with the account)

subscriptions

Usage:	<pre> session = Session('user@example.com', 'pass') subscriptions = session.subscriptions for subscription in subscriptions: print(subscriptions.id, subscriptions.username, subscription.valid, subscription.enabled) </pre>
Reference:	Subscription object on page 36

billing

Returns:	Billing object
Usage:	<pre> session = Session('user@example.com', 'pass') billing = session.billing stats = billing.get_stats(end_year=2018, end_month=1, billing_cycles=1) cost = billing.get_cost(end_year=2018, end_month=1, billing_cycles=1) </pre>
Reference:	Billing object on page 40

data_usage

Returns:	TrialMode object if the account is in trial mode, PaidMode object otherwise
Usage:	<pre> session = Session('user@example.com', 'pass') data_usage = session.data_usage if data_usage.trial_mode: print(type(data_usage)) print(data_usage.trial_expired) else: print(type(data_usage)) print(data_usage.threshold) </pre>
Reference:	Data Usage object on page 43

add_subscription()

Any IP address that you designate as the target of a test must be owned by an Azure subscription that you can access. Use the `session.add_subscription()` call to add an Azure subscription ID to your BreakingPoint Cloud account.

Interface

```
session.add_subscription(self, subscription_id)
```

Arguments

Argument	Description
<code>subscription_id</code>	A string value containing the ID of an Azure subscription that you can access.

Returns

A Subscription object/class instance.

Usage

```
session = Session('user@example.com', 'pass')
sub = session.add_subscription('azure-subscription-id')
print ('Visit this url: {0}'.format(sub.authorization_uri))
```

Example

```
from ixia.bpcddos.session import Session

# constants
__USER_EMAIL    = 'username@example.com'
__PASSWORD      = 'password'

__AZURE_SUBSCRIPTION_ID = 'valid-azure-subscription_id'
__ACCESS_URL_MESSAGE = 'Access this url to activate the subscription: {0}'

if __name__ == '__main__':
    session = Session(__USER_EMAIL, __PASSWORD)

    # add/replace subscription
    new_sub = session.add_subscription(__AZURE_SUBSCRIPTION_ID)
    sub_activation_url = new_sub.authorization_uri
    print(__ACCESS_URL_MESSAGE.format(sub_activation_url))
```

start_test()

Use the `session.start_test()` call to start the test run using the provided configuration values.

Interface

```
session.start_test(self, target_ip, target_port, test_profile, test_size, test_duration)
```

Arguments

Argument	Description
<code>target_ip(str)</code>	A string value representing an IPv4 address. When you start the test, BreakingPoint Cloud will verify that the IP address that you entered is owned by the Azure account that is executing the attack.
<code>target_port(int)</code>	A string value representing a TCP/UDP port number.
<code>test_profile(str or ProfileObject)</code>	A value that identifies the DDoS test profile to use for the test.
<code>test_size(str or ProfileObject.size)</code>	A value that identifies the DDoS test size. Each Test Size profile is a set combination of these values: <ul style="list-style-type: none">• Data volume and rate• The number of source IPs (simulated bots) from which the test data will originate
<code>test_duration(int)</code>	An integer that specifies the number of seconds that the test should run.

Returns

A Test object.

Usage

```
session = Session('user@example.com', 'pass')
profile = session.profiles[0]
# valid public ip
target_ip = 'valid-public-ip-in-your-azure-subscription'
ddos_test = session.start_test(
    target_ip, 80,
    profile, profile.sizes[0], 600
)
ddos_test2 = session.start_test(
    target_ip, 80,
    'http-excessive-get', 'small', 600
)
ddos_test3 = session.start_test(
```

```
|    target_ip, 80,  
|    'TCP SYNACK Flood', '125K pps, 65Mbps and 2 source IPs', 600  
| )
```

Test object

The *Test* object provides the methods and properties to stop BreakingPoint Cloud tests, get test run statistics, and test configuration information.

You can use:

- `test.data` to retrieve the data currently stored for the object (it does not trigger a test data refresh).
- `test.attribute_name` or `test['attribute_name']` to retrieve relevant data from a refreshed test data set (it triggers a test data refresh).

Example:

```
print test.Status
print test['Status']
```

Use `test.data.keys()` to see all available keys/attribute names. (Note that some keys/attribute names depend on the test status).

Topics in this section:

Test properties	27
test.refresh()	31
test.stop()	32

Test properties

Test objects have the following properties:

- id
- failed
- stopped
- config
- profile
- stats

Each is described below.

id	
Returns:	A string containing the ID of the test.
Usage:	<pre> from ixia.bpcddos.session import Session session = Session('user', 'pass') profile = session.profiles[0] target_ip = 'valid-public-ip-in-your-azure-subscription' test = session.start_test(target_ip, 80, profile, profile.sizes[0], 600) print(test.id) </pre>

failed	
Returns:	A Boolean indicating whether or not the current test has failed to start: True if the test has failed, False otherwise.
Usage:	<pre> from ixia.bpcddos.session import Session session = Session('user', 'pass') profile = session.profiles[0] target_ip = 'valid-public-ip-in-your-azure-subscription' test = session.start_test(target_ip, 80, profile, profile.sizes[0], 600) print(test.failed) </pre>

stopped

Returns:	A Boolean indicating whether or not the current test has stopped: True if the test is stopped, False otherwise.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') profile = session.profiles[0] target_ip = 'valid-public-ip-in-your-azure-subscription' test = session.start_test(target_ip, 80, profile, profile.sizes[0], 600) print(test.stopped)</pre>

config

Returns:	A dict with the test's attack parameters: <ul style="list-style-type: none">• AttackSize(str)• TargetIpAddress(str)• Duration(str)• TargetPortNumber(str)• AttackProfile(str)
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') profile = session.profiles[0] target_ip = 'valid-public-ip-in-your-azure-subscription' test = session.start_test(target_ip, 80, profile, profile.sizes[0], 600) print(test.config['AttackSize']) print(test.config['TargetIpAddress']) print(test.config['Duration']) print(test.config['TargetPortNumber']) print(test.config['AttackProfile'])</pre>

profile

Returns:	Profile Object: the profile used by the test. it raises an exception if the profile ID no longer exists.
----------	---

profile**Usage:**

```
from ixia.bpcddos.session import Session
session = Session('user', 'pass')
profile = session.profiles[0]
target_ip = 'valid-public-ip-in-your-azure-subscription'
test = session.start_test(
    target_ip, 80,
    profile, profile.sizes[0], 600
)
test_profile = test.profile
print(test_profile.id == test.config['AttackProfile'])
```

stats**Returns:**

A dict object containing the test statistics:

- TxFrameRate
- Attacks
- AttackRate
- TxDataRate
- TxFrames
- TxData

The test.data is auto-refreshed on call.

stats**Usage:**

```
import time
from ixia.bpcddos.session import Session
session = Session('user', 'pass')
profile = session.profiles[0]
target_ip = 'valid-public-ip-in-your-azure-subscription'
test = session.start_test(
    target_ip, 80,
    profile, profile.sizes[0], 600
)
stats_snapshot = test.stats
print(stats_snapshot['TxFrameRate'])
print(stats_snapshot['Attacks'])
print(stats_snapshot['AttackRate'])
print(stats_snapshot['TxDataRate'])
print(stats_snapshot['TxFrames'])
print(stats_snapshot['TxData'])
# next set of stats, assuming the test is running
time.sleep(10)
stats_snapshot2 = test.stats
print(stats_snapshot2['TxFrameRate'])
print(stats_snapshot2['Attacks'])
print(stats_snapshot2['AttackRate'])
print(stats_snapshot2['TxDataRate'])
print(stats_snapshot2['TxFrames'])
print(stats_snapshot2['TxData'])
```

test.refresh()

Use test.refresh() to retrieve the latest test data, if the test is still running.

Interface

test.refresh()

Arguments

None.

Returns

None.

Usage

```
import time
from ixia.bpcddos.session import Session
session = Session('user', 'pass')
profile = session.profiles[0]
target_ip = 'valid-public-ip-in-your-azure-subscription'
test = session.start_test(
    target_ip, 80,
    profile, profile.sizes[0], 600
)
# print the test data snapshot taken at the start of the test
print(test.data)
time.sleep(10)
test.refresh()
# print the test data snapshot after 10 seconds
print(test.data)
```

test.stop()

Use test.stop() to explicitly stop a test that is running.

Interface

test.stop(self)

Arguments

None

Returns

Latest test data.

Usage

```
from ixia.bpcddos.session import Session
session = Session('user', 'pass')
profile = session.profiles[0]
target_ip = 'valid-public-ip-in-your-azure-subscription'
test = session.start_test(
    target_ip, 80,
    profile, profile.sizes[0], 600
)
test.stop()
```

Profile object

The *Profile* object provides the methods and properties used to validate the test profile size and retrieve information about that test profile.

Profile properties	34
validate_size()	35

Profile properties

Profile objects have the following properties:

- id
- name
- sizes

Each is described below.

id	
Returns:	str: A string value containing the ID of the profile.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') profile = session.profiles[0] profile_id = profile.id</pre>

name	
Returns:	str: A string value containing the name of the profile.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') profile = session.profiles[0] profile_name = profile.name</pre>

sizes	
Returns:	[_Size,]: A list of Size named tuples.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') profile = session.profiles[0] size_name = profile.sizes[0].name size_type = profile.sizes[0].type</pre>

validate_size()

Each BreakingPoint Cloud test that you configure designates a test profile size. The available sizes are dependent upon the test profile selected for the test. Use `validate_size()` call to verify that a particular size is valid for the current the test profile.

Interface

```
validate_size(self, size)
```

Arguments

Argument	Description										
size:	<p>A string that specifies the test size. Each <code>size</code> specifies the number of source IP addresses that will be used in the test, plus a corresponding data volume and rate. Every profile has its own specific combination of test size properties. As a point of reference, the following chart shows the number of source IP addresses that are used based on the <code>size</code> string:</p> <table> <tr> <th>Size</th><th>Number of source IPs</th></tr> <tr> <td>small</td><td>4</td></tr> <tr> <td>medium</td><td>8</td></tr> <tr> <td>large</td><td>16</td></tr> <tr> <td>xlarge</td><td>32</td></tr> </table>	Size	Number of source IPs	small	4	medium	8	large	16	xlarge	32
Size	Number of source IPs										
small	4										
medium	8										
large	16										
xlarge	32										

Returns

Boolean: True if the size is supported by the profile, False otherwise.

Usage

```
from ixia.bpcddos.session import Session
session = Session('user', 'pass')
profile = session.profiles[0]
size_id_exists = profile.validate_size('small')
size_name_exists = profile.validate_size('10K sessions/sec, 725 Mbps and 4 source
IPs')
```


Subscription object

The *Subscription* object methods and attributes needed to manage Azure subscription data within your BreakingPoint Cloud account.

- Subscription properties37**
- subscription.delete()39**

Subscription properties

Subscription objects have the following properties:

- username
- id
- valid
- RefreshedOn
- RegisteredOn
- authorization_uri

Each is described below.

username	
Returns:	str: A string containing the username associated with the subscription.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') subscription = session.subscriptions[0] print(subscription.username)</pre>

id	
Returns:	str: A string containing the Azure subscription ID.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') subscription = session.subscriptions[0] print(subscription.id)</pre>

enabled	
Returns:	Boolean: True if the subscription is enabled, False otherwise.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') subscription = session.subscriptions[0] print(subscription.enabled)</pre>

valid

Returns:	Boolean: True if the subscription is valid, False otherwise.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') subscription = session.subscriptions[0] print(subscription.valid)</pre>

RefreshedOn

Returns:	datetime.datetime: The timestamp when the subscription was last refreshed (in python format).
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') subscription = session.subscriptions[0] print(subscription.RefreshedOn)</pre>

RegisteredOn

Returns:	datetime.datetime: The timestamp when the subscription was added (in python format).
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') subscription = session.subscriptions[0] print(subscription.RegisteredOn)</pre>

authorization_uri

Returns:	str: the URI used for authorizing this subscription.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') subscription = session.subscriptions[0] print(subscription.authorization_uri)</pre>

subscription.delete()

Use the `subscription.delete()` method to remove an Azure subscription from your BreakingPoint Cloud subscription. Once removed, the Azure subscription is no longer available for test activities (unless you add it again).

Interface

`subscription.delete()`

Arguments

None.

Returns

A dict object.

Example

```
# delete subscription
subscription = [
    sub for sub in session.subscriptions if sub.id == __AZURE_SUBSCRIPTION_ID
][0]
subscription.delete()
```

Billing object

The *Billing* object provides the methods and attributes needed to get BreakingPoint Cloud test billing data (usage statistics and costs).

get_cost()	41
get_stats()	42

get_cost()

Use the `session.billing.get_cost()` call to get billing costs for one or more billing cycles.

Interface

```
session.billing.get_cost(self, end_year, end_month, billing_cycles=1)
```

Arguments

Argument	Description
<code>end_year(int):</code>	A four-digit integer that specifies the billing year for the request.
<code>end_month(int):</code>	An integer that specifies the billing month for the request.
<code>billing_cycles(int):</code>	An integer that specifies the number of billing cycles to include in the request.

Returns

A list with invoice data.

Usage

```
from ixia.bpcddos.session import Session
session = Session('user', 'pass')
billing = session.billing
billing_data = billing.get_cost(year=2018, month=3, billing_cycles=1)
for invoice in billing_data:
    print(invoice['creationData'], invoice['amount'], invoice['downloadObjectUrl'])
```

get_stats()

Use the `session.billing.get_stats()` call to get usage statistics for one or more billing cycles.

Interface

```
session.billing.get_stats(self, end_year, end_month, billing_cycles=1)
```

Arguments

Argument	Description
<code>end_year(int):</code>	A four-digit integer that specifies the billing year for the request.
<code>end_month(int):</code>	An integer that specifies the billing month for the request.
<code>billing_cycles (int):</code>	An integer that specifies the number of billing cycles to include in the request.

Returns

a dict object containing the request billing data.

Usage

```
from ixia.bpcddos.session import Session
session = Session('user', 'pass')
billing = session.billing
billing_data = billing.get_stats(year=2018, month=3, billing_cycles=1)
```

Data Usage object

The *Data Usage* object provides the methods and attributes needed to get information about the BreakingPoint Cloud user's current data usage.

Data Usage common properties	44
Trial Mode object	45
Paid Mode object	47

Data Usage common properties

Data Usage objects have the following common property: trial_mode.

trial_mode	
Returns:	Boolean: True if the user is in trial mode, False otherwise.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') data_usage = session.data_usage print(data_usage.trial_mode)</pre>

Trial Mode object

The *Data Usage* Trial Mode object provides the methods and attributes needed to get or change the user's BreakingPoint Cloud-subscription data, while in trial mode.

Trial Mode properties

Data Usage Trial Mode objects have the following properties:

- trial_expired
- consumed
- remaining
- quota

Each is described below.

trial_expired	
Returns:	Boolean: True if the trial has expired, False otherwise.
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') data_usage = session.data_usage print(data_usage.trial_expired)</pre>

consumed	
Returns:	<p>For Python 2: long: The current consumed data value in bytes.</p> <p>For Python 3: int: The current consumed data value in bytes.</p>
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') data_usage = session.data_usage if data_usage.quota - data_usage.consumed == 0 : ... print('Trial quota reached') print(data_usage.consumed) '75.000 GB'</pre>

remaining	
Returns:	<p>For Python 2:</p> <p>long: The total amount of data (in bytes) that remains available for the trial user. It is calculated as (quota – consumed).</p> <p>For Python 3:</p> <p>int: The total amount of data (in bytes) that remains available for the trial user. It is calculated as (quota – consumed).</p>
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') data_usage = session.data_usage if data_usage.remaining == 0 : ... print('Trial quota reached') # pretty print print(data_usage.remaining) '20.000 GB'</pre>

quota	
Returns:	<p>For Python 2:</p> <p>long: The total amount of data (in bytes) that one trial user can consume.</p> <p>For Python 3:</p> <p>int: The total amount of data (in bytes) that one trial user can consume.</p>
Usage:	<pre>from ixia.bpcddos.session import Session session = Session('user', 'pass') data_usage = session.data_usage if data_usage.quota - data_usage.consumed == 0 : ... print('Trial quota reached') # pretty print print(data_usage.quota) '100.000 GB'</pre>

Paid Mode object

The *Data Usage* Paid Mode object provides the methods and attributes needed to get or change the user's BreakingPoint Cloud-subscription data, while in paid mode.

Paid Mode properties

Data Usage Paid Mode objects have the following properties:

- threshold
- consumed

Each is described below.

threshold	
Returns:	For Python 2: long: The current threshold value in bytes. For Python 3: int: The current threshold value in bytes.
Usage:	<pre> from ixia.bpcddos.session import Session session = Session('user', 'pass') data_usage = session.data_usage if data_usage.threshold - data_usage.consumed > 100000: ... data_usage.increase_threshold() # pretty print print(data_usage.threshold) '250.000 GB' </pre>

consumed	
Returns:	For Python 2: long: The current consumed data value in bytes. For Python 3: int: The current consumed data value in bytes.
Usage:	<pre> from ixia.bpcddos.session import Session session = Session('user', 'pass') data_usage = session.data_usage if data_usage.threshold - data_usage.consumed > 100000: ... data_usage.increase_threshold() # pretty print print(data_usage.consumed) '123.251 GB' </pre>

increase_threshold()

Use `increase_threshold()` to increase the data usage threshold for your BreakingPoint Cloud account.

Interface

`increase_threshold(self)`

Arguments

None

Returns

A Success message upon success.

Raises

Exception if the increase threshold operation failed.

Usage

```
from ixia.bpcddos.session import Session
session = Session('user', 'pass')
data_usage = session.data_usage
```

INDEX

A

add_subscription() 23
API, connect to 6
Azure subscriptions, manage 36

B

billing, object 40
boto3 2

C

code samples 5

D

data_usage, object 43

G

get_cost() 41
get_stats() 42

I

increase_threshold() 48
installation 3

P

profile, object 33
Python versions supported 2

R

requests library 2

requirements 2

S

session, object 20
start_test() 24
subscription, object 36
subscription.delete() 39

T

test, object 26
test.refresh() 31
test.stop() 32

V

validate_size() 35

