

ExtraHop Python API

Version 5.0

Publication Date: 1/7/2016

Contents

Overview	8
Getting Started	9
Basic Script Structure	11
Best Practices	12
Functions	13
Getters	14
get_alerts	17
get_active_devices	19
get_activity_group_count	22
get_all_devices	24
get_all_flex_grids	27
get_applications	28
get_captures	29
get_custom_device	30
get_custom_device_criteria	31
get_custom_device_criterion	32
get_custom_devices	33
get_customization	35
get_customizations	36
get_device	37
get_device_activity	39
get_device_group_devices	41
get_device_groups	43
get_device_tags	48
get_devices	49
get_devices_with_activity	52
get_dynamic_group_devices	54
get_dynamic_groups_count	57

get_exstats	58
get_exstats_group	63
get_exstats_next	68
get_exstats_total	69
get_extrahop_version	74
get_flex_grid_data	75
get_running_config	77
get_ssl_decrypt_config	77
get_ssl_decrypt_configs	78
get_trigger	79
get_triggers	80
get_network	82
get_networks	83
get_packetcapture	84
get_packetcaptures	85
get_user	86
get_users	87
get_user_apikey	88
get_user_apikeys	89
Setters	90
add_device_group_members	93
add_flex_grid_rows	96
add_to_whitelist	96
apply_customization	97
apply_trigger	98
create_custom_device	99
create_custom_device_criterion	100
create_customization	101
delete_custom_device	102
delete_custom_device_criterion	103

delete_customization	104
delete_device_groups	105
delete_packetcapture	106
delete_triggers	107
device_tag_add	108
device_tag_delete	109
device_tag_remove	110
device_tag_rename	111
is_ecm	112
new_device_group	113
new_dynamic_device_group	116
register_license	117
remove_device_group_members	118
remove_flex_grid_rows	119
remove_flex_grids	120
remove_from_whitelist	120
remove_trigger	121
restart_service	122
save_flex_grid	123
save_ssl_decryption_key	124
set_running_config	125
set_trigger	126
update_custom_device	128
update_device	129
update_device_group	130
update_device_group_members	133
update_dynamic_device_group	134
update_network	135
update_user	136
Searchers	137

search_devices_by_activity	138
search_devices_by_any	141
search_devices_by_ip	144
search_devices_by_mac	146
search_devices_by_name	148
search_devices_by_tag	151
search_devices_by_type	154
search_devices_by_vendor	156
search_devices_by_vlan	159
Metrics	162
Application Metrics	163
AAA	164
Database	166
DNS	168
HTTP	170
IBMMQ	172
ICA	175
L4	178
LDAP	180
Memcache	182
NAS	184
SSL	186
Device Metrics	188
AAA	189
AMF	190
CIFS	192
Database	194
DNS	196
FIX	197
FTP	199

HTTP	202
IBMMQ	205
ICA	209
iSCSI	212
L2-L3	216
L4 TCP	218
LDAP	220
Memcache	222
MS-RPC	224
NFS	226
SMPP	229
SMTP	232
SSL	234
Capture Metrics	236
Custom Metrics	239
Health Metrics	240
Datatypes	243
count	244
dataset	246
snap	250
string	252
time	253
topn_count	254
topn_dset	258
topn_snap	262
topn_sset	265
topn_time	271
topn_tset	273
Examples	277
activity-stats.py	278

applications.py	279
bridge-devcount.py	281
count.py	282
custom-devices.py	284
dataset.py	285
device-groups.py	287
device-search-by-activity.py	289
device-search-by-name.py	289
device-tags.py	290
device-update-name.py	292
dns-queries.py	293
exstats-ecm.py	294
get-alert-definitions.py	295
get-alerts-fired.py	295
l7-proto-details.py	297
print-all-devices.py	298
print-trigger-runtime-log.py	299
pull-running-config.py	300
set-host-name.py	301
ssl-key-expiration.py	301
topn-dset.py	303
topn-sset.py	304
whitelist.py	305

Overview

The ExtraHop API allows structured access to data stored on the ExtraHop appliance. The API is based on exchanging JSON messages over a secure web service (HTTPS). This document is for use with ExtraHop firmware 4.0 and later.

Getting Started

This section provides instructions for running API scripts against an ExtraHop system using the ExtraHop Python API (PyHop).

Note: As of version 5.0.xxxx, you must use API key-based authentication. Username and password authentication is no longer supported.

With API key access, ExtraHop administrators use **Access Settings** in the ExtraHop Admin UI to grant API keys to authorize API users. API users then supply their key(s) when initializing the PyHop client object in their PyHop scripts.

As a convenience for operating across multiple PyHop scripts and ExtraHop appliances, API keys may be stored in a central configuration file having the following format:

```
# This is a comment
Host extrahop-east
Target extrahop.ny.example.com
ApiKey C7f8761f7b9b421f8b9bb64fdf376ca0

Host extrahop-west
Target 10.20.30.40
ApiKey 5e73afald08e13b76ffc32a961ac0d87
```

Configuration File Notes:

- The configuration file consists of one section for each ExtraHop appliance that will be accessed via PyHop.
- A **Host** line marks the beginning of a new section.
- When a configuration file is in use, the host argument in PyHop client calls will be matched against **Host** properties in the configuration file. When there is a match, the **Target** property in the same section will be interpreted as the ExtraHop hostname or IP to connect to. When there is no **Target** property in the same configuration file section, the **Host** property will be interpreted as the hostname.
- The **ApiKey** property specifies the API key that was allocated to the PyHop user by the ExtraHop appliance administrator.
- All configuration file content is case insensitive.

PyHop will look for the configuration file in `~/.extrahop`, however this default file location may be overridden. Scripts that initialize using `phyhop.Client()` can include the `'config'` property to specify an alternate file location. Scripts that initialize using `pyhop.make_client()` can use the `'-c'` command-line option to specify an alternate location.

To run API scripts against an ExtraHop system using PyHop, complete the following steps.

1. Ensure you are using ExtraHop firmware version 4.0 or later.
2. Ensure Python v2.7.3 or a later 2.7.x version (but not 3.x) is installed.

Note: If you have multiple versions of Python installed, run `"python --version"` on the command line to verify that the required version is the primary version on your path. Alternately, use the full path to the correct version when running PyHop scripts.

3. Download the ExtraHop Python API SDK (PyHop) from the ExtraHop Support Portal: <https://customer.extrahop.com/>.

Note: If you have upgraded your ExtraHop system from version 3.10 to 4.0, you must upgrade the pyhop client to version 4.0 to ensure proper functionality.

4. Install PyHop.

- If you are using Linux, run the following commands:

```
# unzip PyHop-4.x.xxxx.zip
```

```
# cd PyHop-4.x.xxxx
```

```
# sudo python setup.py install
```

- If you are using Windows, use the following process:
 - a. In Explorer, right-click on the PyHop.4.x.xxxx.zip file and click **Extract All**.
 - b. Open a command prompt and navigate to the directory your file was unzipped into. For instance:

```
cd c:\users\extrahop\Downloads\PyHop-4.x.xxxx\PyHop-4.x.xxxx
```

- c. Type the following at the command prompt and click **Enter** to run the SDK setup:

```
python setup.py install
```

5. Add the necessary directory to your PYTHONPATH. For instance, on Linux, change the directory to "pyhop", and run the command:

```
export PYTHONPATH=$(pwd) .
```

6. Run the "print all devices.py" example to ensure the setup was successful.

```
# python examples/print-all-devices.py -H <extrahop_hostname> -a <apikey>
```

Upon success, the script prints a list of all devices that have been discovered by the ExtraHop appliance.

Basic Script Structure

Use the following connection code to run commands against the ExtraHop appliance.

1. Import the pyhop client.

```
from pyhop import Client
```

2. Go to the ExtraHop Admin UI to generate and save an API key.
3. Establish a session with the ExtraHop appliance.

```
c = Client(host="1.2.3.4", apikey="d8e8c021257a493eb130a5f199732d75")
```

4. Use the Client object to call API commands against the ExtraHop appliance. The example below shows how to get the capture object and print its parameters.

```
# get capture object
result = c.get_captures(0)
if result is not None:
    capture = result[0]
    print "capture (oid,mod_time,idle):", capture.oid, capture.mod_time,
capture.idle
```

Best Practices

Note: As of release version 5.0.xxxx, you must authenticate using an API Key. Username and password authentication is no longer supported.

When using the API, it is important to understand that querying data uses resources on the device. If an inefficient query is issued, it could potentially interfere with the process of data capture and analysis. It is best practice to make your queries as specific as possible.

- Query for the exact metric and field.
- Query for the exact time.

Functions

This section defines the functions for the `pyhop.Client` object from the `pyhop` library.

Note: Users with the Administrator role can use getters, setters, searchers, and formatting functions. Users with the Operator role can use getters, searchers, and formatting functions.

Getters

Activity Groups

get_activity_group_count(*start, until*)
Get a number of members in an activity group.

Alerts

get_alerts(*since*)
Get alerts.

Applications

get_applications(*start, until*)
Get a list of applications and their attributes.

Captures

get_captures(*since*)
Get information about the capture process.

Configurations

get_running_config()
Get the running config.

Customizations

get_customization(*customization_oid*)
Get details about a specific customization.

get_customizations()
Get all customizations on the ExtraHop system.

Device Groups

get_device_group_devices(*start, until, group_oid, [{"offset":offset}, {"limit":limit}]*)
Get members of a specific group.

get_device_groups(*since*)
Get a list of all device groups.

get_dynamic_group_devices(*start, until, group_oid, [{"offset":offset}, {"limit":limit}]*)
Get members of a specific dynamic group.

get_dynamic_groups_count(*start, until*)
Get a number of members in a dynamic group.

Devices

get_active_devices(*offset, limit, start, until*)
Get active devices in a certain time frame.

get_all_devices()
Get the list of discovered devices.

get_custom_device(*device_oid*)
Get a specific custom device.

get_custom_device_criteria(*device_oid*)
Get all criteria for a custom device.

get_custom_device_criterion(*device_oid, criterion_oid*)
Get details about a specific criterion.

get_custom_devices()
Get a list of all custom devices on the ExtraHop system.

get_device(*start, until, device_oid*)
Get device with a certain oid.

get_device_activity(*device_oids*)
Get device activity log.

get_devices(*device_oids*)
Get specific devices.

get_devices_with_activity(*start, until, device_oids*)
Get active devices from a list of device object IDs that are active within a certain time-frame.

ExtraHop

get_extrahop_version()
Get the version of the ExtraHop system.

Flex-Grids

get_all_flex_grids(*start*)
Get a list of flex grids.

get_flex_grid_data(*flexgrid_oid*)
Get data associated with a specific flex grid.

Metrics

get_exstats(["*stat_name*", "*object_type*", "*object_id*", "*from_time*", "*until_time*"], ["*field_spec*"], [{"*cycle*": "*cycle*"}, {"*stat_key*": "*stat_key*"}, {"*stat_key_match_flags*: "*stat_key_match_flags*"}, {"*topn_max*: "*topn_max*"}]})
Get raw metrics for a specific capture or device.

get_exstats_group(["*stat_name*", "*object_type*", "*object_id*", "*from_time*", "*until_time*"], ["*field_spec*"], [{"*cycle*": "*cycle*"}, {"*stat_key*": "*stat_key*"}, {"*stat_key_match_flags*: "*stat_key_match_flags*"}, {"*topn_max*: "*topn_max*"}]})
Get raw metrics for a specific group of devices.

get_exstats_next(*xid*)
Get next chunked results returned by exstats.

get_exstats_total("*stat_name*", "*object_type*", "*object_spec*", "*field_spec*", [{"*cycle*": "*cycle*"}, {"*stat_key*": "*stat_key*"}, {"*stat_key_match_flags*: "*stat_key_match_flags*"}, {"*topn_max*: "*topn_max*"}]})
Get raw metrics for a specific capture or device and return a sum across those metrics.

Networks

get_network(*network_oid*)
Get details about a specific network.

get_networks()
Get all networks on the ExtraHop system.

Packet Captures

get_packetcapture(*capture_oid*)
Get details about a specific packet capture.

get_packetcaptures()
Get metadata about all packet captures on the ExtraHop system.

SSL Certificates

"get_ssl_decrypt_config" on page 77 (*ssl_id*)

Retrieve information about a single SSL certificate on the ExtraHop.

get_ssl_decrypt_configs()

Retrieve information about all SSL certificates on the ExtraHop

Tags

get_device_tags()

Get device tags.

Triggers

get_trigger(*trigger_oid*)

Get data associated with a specific trigger.

get_triggers(*start*)

Get a list of triggers.

Users

get_user("username")

Get details about a specific user.

get_user_apikey("username", "key_id")

Get a specific API key for a user.

get_user_apikeys("username")

Get all API keys associated with a user.

get_users()

Get a list of users on the ExtraHop system.

get_alerts

Get alerts.

```
get_alerts(since)
```

Parameter

since: Number

The start time, expressed in milliseconds. If the value is positive, it indicates the absolute time since the epoch (January 1, 1970). If the value is negative, it indicates the relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, it indicates the current capture time.

Return Value

A list of alerts.

Sample Usage

```
> r = c.get_alerts(0)
> import json
> print json.dumps(r, indent=2)
[
  {
    "description": "",
    "mod_time": 1361924315.0,
    "notify_snmp": true,
    "deleted": false,
    "assigned_objects": [
      {
        "object_type": "device",
        "id": 1,
        "object_id": 304
      }
    ],
    "field_op": "",
    "exclusion_intervals": [],
    "stat_name": "extrahop.device.http_server",
    "disabled": false,
    "operator": "> normal",
    "cluster_id": "no_cluster",
    "operand": "4",
    "additional_stats": [],
    "field_name": "rsp_error",
    "id": 23,
    "name": "test #2",
    "additional_emails": "",
    "apply_all": false,
    "severity": 3,
    "author": "Setup",
    "param": "{}",
    "interval_length": 30,
    "stat_key": "cluster_id: \"no_cluster\", name: \"test #2\"",
```

```
"emailgroups": [],
"param2": "{}",
"units": "period",
"field_name2": null,
"refire_interval": 0
},
{
"description": "",
"mod_time": 1360020766.0,
"notify_snmp": true,
"deleted": false,
"assigned_objects": [],
"field_op": "",
"exclusion_intervals": [],
"stat_name": "extrahop.capture.app",
"disabled": false,
"operator": ">",
"cluster_id": "no_cluster",
"operand": "300000",
"additional_stats": [],
"field_name": "bytes",
"id": 1,
"name": "tester",
"additional_emails": "",
"apply_all": false,
"severity": 3,
"author": "Setup",
"param": "{\"key1\": \"\"}",
"interval_length": 30,
"stat_key": "cluster_id: \"no_cluster\", name: \"tester\"",
"emailgroups": [],
"param2": "{}",
"units": "period",
"field_name2": null,
"refire_interval": 900
}
]
```

See Also:

get-alert-definitions.py on page 295

get_active_devices

Get a device in a specific time frame.

Note: This method has been deprecated. Please use the [search_devices_by_activity](#) on page 138 method instead.

Syntax:

```
get_active_devices(offset, limit, start, until)
```

Parameters

limit: Integer

The number of devices to return to support for pagination of results. For example, if you have 100 devices and you ask for `offset=0, limit=10`, you'll get the first 10 devices. If you ask for `offset=50, limit=50`, you'll get the last 50 devices.

offset: Integer

The offset of the devices to return to support for pagination of results. See **limit** above for more information.

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., `-180000` is three minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.get_active_devices(-1800000, 0, 0, 2)
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 702,
  "devices": [
    {
      "description": null,
      "mod_time": 1361900433.0,
      "macaddr": "78:2B:CB:32:C9:81",
      "vendor": "Dell",
```

```
"custom_type": "",
"devclass": "node",
"ipaddr4": null,
"ipaddr6": null,
"parent_oid": null,
"vlanid": 1010,
"ip_discover": false,
"default_name": "Dell 32C981",
"capture_oid": 0,
"id": 2345,
"device_id": "782bcb32c9811010",
"dhcp_name": "",
"user_mod_time": 1361900433.0,
"analysis_level": 0,
"name": "Dell 32C981",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1361900405041.0,
"oid": 2344,
"nb_name": ""
},
{
  "description": null,
  "mod_time": 1361900163.0,
  "macaddr": "78:2B:CB:32:C9:89",
  "vendor": "Dell",
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": "10.10.251.227",
  "ipaddr6": null,
  "parent_oid": 2342,
  "vlanid": 1010,
  "ip_discover": true,
  "default_name": "Dell 10.10.251.227",
  "capture_oid": 0,
  "id": 2344,
  "device_id": "fff4e3fb0a0a1010",
  "dhcp_name": "",
  "user_mod_time": 1361900163.0,
  "analysis_level": 0,
  "name": "Dell 10.10.251.227",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
  "tag_set": [],
  "tags": [],
  "discover_time": 1361900141041.0,
  "oid": 2343,
  "nb_name": ""
}
```

```
]
}
```

See Also:

get-alerts-fired.py on page 295

get_activity_group_count

Get the number of members of each activity group.

Syntax:

```
get_activity_group_count(start, until)
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is three minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

Return Value

An object with counts for each metric type.

Sample Usage

```
>>> r = c.get_activity_groups_count(-1800000, 0)
>>> import json
>>> print json.dumps(r, indent=2)
{
  "extrahop.device.nfs_client": 4,
  "extrahop.device.db_server": 1,
  "extrahop.device.ldap_client": 4,
  "extrahop.device.smtp_server": 9,
  "extrahop.device.ssl_client": 128,
  "extrahop.device.cifs_client": 50,
  "extrahop.device.tcp": 243,
  "extrahop.device.nfs_server": 2,
  "extrahop.device.ldap_server": 1,
  "extrahop.device.ftp_client": 5,
  "extrahop.device.http_server": 46,
  "extrahop.device.app": 372,
  "extrahop.device.ftp_server": 6,
  "extrahop.device.ica_server": 1,
  "extrahop.device.cifs_server": 16,
  "extrahop.device.ica_client": 1,
  "extrahop.device.db_client": 2,
  "extrahop.device.amf_client": 9,
  "extrahop.device.amf_server": 3,
  "extrahop.device.http_client": 112,
  "extrahop.device.dns_client": 215,
  "extrahop.device.net": 671,
  "extrahop.device.smtp_client": 26,
  "extrahop.device.ssl_server": 85,
```

```
"extrahop.device.dns_server": 22  
}
```

get_all_devices

Get the list of discovered devices.

Syntax:

```
get_all_devices()
```

Parameters

None.

Return Value

A list of device objects, each including the following fields:

capture_oid: Integer

The object ID of the capture process which discovered the device.

cdp_name: String

The Cisco Discovery Protocol name of the device.

devclass: String

The device class.

Possible Values:

- gateway'
- node
- pseudo
- remote'

device_id: UTF-8 String

The device ID, guaranteed to be unique on one ExtraHop device.

Sample Value: u'0015c5ec20d40000'

device_oid: Integer

The device object ID (unique).

dhcp_name: String

The DHCP name of the device.

discover_time: Time in UTC

The time at which the device was discovered.

Sample Value: 1279818934000.0

dns_name: String

The DNS name of the device.

ipaddr4: String

The IPv4 address of the device.

Sample Value: 10.10.248.29

ipaddr6: String

The IPv6 address of the device.

Sample Value: 2001:db8:1234:0000:0000:0000:0000:0000

macaddr: String

The MAC address of the device.

Sample Value: 74:86:7A:ED:79:48

mod_time: Time in UTC
The device modification time.

nb_name: String
The NetBIOS name of the device.

vlanid: Integer
The VLAN ID of the device.

Sample Usage

```
# get all devices
result = c.get_all_devices()
# print device with id 0
if result != None and result[0] != None:
d = result[0]
print "raw data:\n\n", d
print "sample device:", d.oid, d.macaddr, d.ipaddr4, d.dns_name, d.nb_name, d.mod_
time
```

Sample Output

```
>>> r = c.get_all_devices()
>>> import json
>>> print json.dumps(r, indent=2)
[
  {
    "description": null,
    "mod_time": 1361400676.0,
    "macaddr": "BC:AE:C5:50:F9:FF",
    "vendor": "ASUS",
    "custom_type": "",
    "devclass": "node",
    "ipaddr4": null,
    "ipaddr6": null,
    "parent_oid": null,
    "vlanid": 1010,
    "ip_discover": false,
    "default_name": "ASUS 50F9FF",
    "capture_oid": 0,
    "id": 1,
    "device_id": "bcaec550f9fff1010",
    "dhcp_name": "",
    "user_mod_time": 1358306852.0,
    "analysis_level": 0,
    "name": "ASUS 50F9FF",
    "custom_name": null,
    "cdp_name": "",
    "dns_name": "",
    "tag_set": [],
    "tags": [],
    "discover_time": 1354589328113.0,
    "oid": 0,
```

```
"nb_name": ""
},
{
  "description": null,
  "mod_time": 1361434711.0,
  "macaddr": "F4:6D:04:1B:C0:3F",
  "vendor": "ASUS",
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": null,
  "ipaddr6": null,
  "parent_oid": null,
  "vlanid": 1010,
  "ip_discover": false,
  "default_name": "ASUS 1BC03F",
  "capture_oid": 0,
  "id": 2,
  "device_id": "f46d041bc03f1010",
  "dhcp_name": "",
  "user_mod_time": 1358306852.0,
  "analysis_level": 0,
  "name": "ASUS 1BC03F",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
  "tag_set": [],
  "tags": [],
  "discover_time": 1354589328113.0,
  "oid": 1,
  "nb_name": ""
}
]
```

See Also:

print-all-devices.py on page 298

get_all_flex_grids

Get a list of flex grids.

Syntax:

```
get_all_flex_grids(start)
```

Parameter

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is three minutes from the current capture time). If the value is 0, it indicates the current capture time.

Return Value

A list of flex grid objects modified since the specified start time.

Sample Usage

```
>>> r = c.get_all_flex_grids(0)
>>> import json
>>> print json.dumps(r, indent=2)
[
  {
    "mod_time": 1360020702.0,
    "description": "This is my application flex grid",
    "object_type": "application",
    "name": "app grid",
    "oid": 1
  },
  {
    "mod_time": 1360020702.0,
    "description": "",
    "object_type": "device",
    "name": "my grid",
    "oid": 2
  }
]
```

get_applications

Get a list of applications and their attributes.

```
get_applications(start, until)
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

Return Value

A list of applications.

Sample Usage

```
>>> r = c.get_applications(-1800000, 0)
>>> import json
>>> print json.dumps(r, indent=2)
[
  {
    "mod_time": 1360020701.0,
    "description": "Built-in application based on all traffic",
    "application_id": "_default",
    "name": "Default",
    "deleted": false,
    "oid": 0,
    "display_name": "Default",
    "capture_oid": 0
  },
  {
    "mod_time": 1361924494.0,
    "description": null,
    "application_id": "myapp",
    "name": "myapp",
    "deleted": false,
    "oid": 7,
    "display_name": "myapp",
    "capture_oid": 0
  }
]
```

See Also:

"applications.py" on page 279

get_captures

Get information about the capture process. In this API, capture metrics refer to network metrics.

Syntax:

```
get_captures(since)
```

Parameter

since: Number

The start time, expressed in UTC. Returns all capture objects with a `mod_time` attribute greater than the **since** parameter provided here. If 0 is specified, all captures are returned, since they would have been previously modified.

Return Value

An array of capture objects with the following fields:

capture_oid: Integer

The capture object ID (unique).

idle: Boolean

Indicates whether the capture is currently processing traffic.

mod_time: Time in UTC

The capture modification time.

Sample Usage

```
# get capture object
result = c.get_captures(0)
```

for capture in result:

```
print "capture (capture_oid,mod_time,idle):", capture.capture_oid, capture.mod_
time, capture.idle
```

Sample Output (Single ExtraHop Appliance)

```
capture (capture_oid,mod_time,idle): 0 1393890951 False
```

Sample Output (ECM)

```
capture (capture_oid,mod_time,idle): 4294967296 1393890156 False
capture (capture_oid,mod_time,idle): 8589934592 1393888131 False
```

See Also:

"activity-stats.py" on page 278

"applications.py" on page 279

print-trigger-runtime-log.py on page 299

get_custom_device

Get a specific custom device.

Syntax:

```
get_custom_device(device_oid)
```

Parameters:

device_oid: Integer
The ID of the device you want returned.

Return Value:

A corresponding device object.

get_custom_device_criteria

Get all criteria for a custom device.

Syntax:

```
get_custom_device_criteria(device_oid)
```

Parameters:

device_oid: Integer
The unique object ID of the device.

Return Value:

A list of custom device criteria.

get_custom_device_criterion

Get details about a specific criterion.

Syntax:

```
get_custom_device_criterion(device_oid, criterion_oid)
```

Parameters:

criterion_oid: Integer
The unique object ID of the criterion.

device_oid: Integer
The unique object ID of the device.

Return Value:

The custom device criterion details.

get_custom_devices

Get a list of all custom devices on the ExtraHop system.

Syntax:

```
get_custom_devices()
```

Parameters

None.

Return Value

A list of device objects, each including the following fields:

capture_oid: Integer

The unique object ID of the capture process which discovered the device.

cdp_name: String

The Cisco Discovery Protocol name of the device.

devclass: String

The device class. Possible values are:

- gateway
- node
- pseudo
- remote

device_id: UTF-8 String

The ID of the device, guaranteed to be unique on one ExtraHop device.

Sample Value: u'0015c5ec20d40000'

device_oid: Integer

The unique object ID of the device.

dhcp_name: String

The DHCP name of the device.

discover_time: Integer

The time at which the device was discovered in UTC.

Sample Value: 1279818934000.0

dns_name: String

The DNS name of the device.

ipaddr4: String

The IPv4 address of the device.

Sample Value: 10.10.248.29

ipaddr6: String

The IPv6 address of the device.

Sample Value: 2001:db8:1234:0000:0000:0000:0000:0000

macaddr: String

The MAC address of the device.

Sample Value: 74:86:7A:ED:79:48

mod_time: Integer
The device modification time in UTC.

nb_name: String
The NetBIOS name of the device.

vlanid: Integer
The VLAN ID of the device.

get_customization

Get details about a specific customization.

Syntax:

```
get_customization(customization_oid)
```

Parameters:

customization_oid: Integer
The unique object ID of the customization.

Return Value:

Customization details.

get_customizations

Get all customizations on the ExtraHop system.

Syntax:

```
get_customizations()
```

Parameters:

None.

Return Value:

A list of customizations.

get_device

Get a device with a certain object ID.

Syntax:

```
get_device(start, until, device_oid)
```

Parameters

device_oid: Integer

The device object ID.

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

Return Value

A corresponding device object.

Sample Usage

```
>>> r=c.get_device(0, 0, 15)
>>> import json
>>> print json.dumps(r, indent=2)
{
  "description": null,
  "mod_time": 1361812035.0,
  "macaddr": "3C:07:54:32:D6:69",
  "vendor": "Apple",
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": null,
  "ipaddr6": null,
  "parent_oid": null,
  "vlanid": 1010,
  "ip_discover": false,
  "default_name": "Apple 32D669",
  "capture_oid": 0,
  "id": 16,
  "device_id": "3c075432d6691010",
  "dhcp_name": "",
  "user_mod_time": 1358306852.0,
  "analysis_level": 0,
  "name": "Apple 32D669",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
```

```
"tag_set": [],
"tags": [],
"discover_time": 1354589328113.0,
"activity": [
    "extrahop.device.app",
    "extrahop.device.tcp",
    "extrahop.device.ssl_client",
    "extrahop.device.dns_client",
    "extrahop.device.net"
],
"oid": 15,
"nb_name": ""
}
```

See Also:

"device-update-name.py" on page 292

get_device_activity

For every device specified in the input list, return the device activity log.

Syntax:

```
get_device_activity(device_oids)
```

get_device_activity(device_oids: List): Structure

Parameter

device_oids: List

A list of integers representing device object IDs.

Return Value

A structure specifying device activity including the following fields:

activity: List of key-value pairs

A list of key-value pairs specifying activity.

device_oid: Integer

The device object ID (unique).

from_time: Time in UTC

The time during which activity for the above metrics started.

Sample Value: 1279818934000.0

stat_name: String

The name of the metric for which activity was registered on this device.

Sample Value:extrahop.device.tcp

(Refer to **Metrics** on page 162 for a complete list.)

until_time: Time in UTC

The time during which activity for the above metrics ended.

Sample Value: 1279818934000.0

Sample Usage

```
device_oids = [0,1,2]
# query device activity for devices with oids 1, 2, 3
result = c.get_device_activity(device_oids)
for i in result:

    print "sample device activity", i
```

Sample Output

```
{'oid': 0,
 'activity':
  [{'from_time': 1279818934000.0, 'until_time': 1279818986000.0, 'stat_name':
u'extrahop.device.net'},
  {'from_time': 1279818934000.0, 'until_time': 1279818986000.0, 'stat_name':
u'extrahop.device.tcp'}]}
```

```
...  
}
```


get_device_group_devices

Get members of a certain group.

Syntax:

```
get_device_group_devices(start, until, group_oid, [{"offset":offset},
{"limit":limit}])
```

Parameters

group_oid: Integer

The group object ID.

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, it indicates the current capture time.

until: Number

The stop time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

Options

Optional support for pagination of results. For example, if you have 100 devices and you ask for offset=0, limit=10, the firsts 10 devices will be returned. If you ask for offset=50, limit=50, the last fifty devices will be returned.

limit: Integer

The number of devices to return.

offset: Integer

The offset of the devices to return.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.get_device_group_devices(-1800000, 0, 100, {"offset": 0, "limit": 1})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 3,
  "devices": [
    {
      "description": null,
      "mod_time": 1361434711.0,
      "macaddr": "F4:6D:04:1B:C0:3F",
```

```
"vendor": "ASUS",
"custom_type": "",
"devclass": "node",
"ipaddr4": null,
"ipaddr6": null,
"parent_oid": null,
"vlanid": 1010,
"ip_discover": false,
"default_name": "ASUS 1BC03F",
"capture_oid": 0,
"id": 2,
"device_id": "f46d041bc03f1010",
"dhcp_name": "",
"user_mod_time": 1358306852.0,
"analysis_level": 0,
"name": "ASUS 1BC03F",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1354589328113.0,
"activity": [
    "extrahop.device.net"
],
"oid": 1,
"nb_name": ""
}
]
```

get_device_groups

Get a list of all device groups.

Syntax

```
get_device_groups(since)
```

Parameter

since: Number

The since time, expressed in milliseconds. If the value is positive, it indicates the absolute time since the epoch (January 1, 1970). If the value is negative, it indicates the relative time from the current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, it indicates the current capture time.

Return Value

A list of device group objects.

Sample Usage

```
>>> r = c.get_device_groups(0)
>>> import json
>>> print json.dumps(r, indent=2)
[
  {
    "mod_time": 1359678765.0,
    "description": "description1",
    "name": "test1",
    "deleted": false,
    "oid": 1,
    "dynamic": false,
    "value": null,
    "field": null,
    "internal": false,
    "cluster_id": "no_cluster",
    "members": [
      1,
      2,
      0
    ],
    "member_set": [
      {
        "description": null,
        "mod_time": 1361434711.0,
        "macaddr": "F4:6D:04:1B:C0:3F",
        "vendor": "ASUS",
        "custom_type": "",
        "devclass": "node",
        "ipaddr4": null,
        "ipaddr6": null,
        "parent_oid": null,
        "vlanid": 1010,
        "ip_discover": false,
```

```
"default_name": "ASUS 1BC03F",
"capture_oid": 0,
"id": 2,
"device_id": "f46d041bc03f1010",
"dhcp_name": "",
"user_mod_time": 1358306852.0,
"analysis_level": 0,
"name": "ASUS 1BC03F",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1354589328113.0,
"oid": 1,
"nb_name": ""
},
{
  "description": null,
  "mod_time": 1361400676.0,
  "macaddr": "20:CF:30:57:36:3B",
  "vendor": "ASUS",
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": null,
  "ipaddr6": null,
  "parent_oid": null,
  "vlanid": 1010,
  "ip_discover": false,
  "default_name": "ASUS 57363B",
  "capture_oid": 0,
  "id": 3,
  "device_id": "20cf3057363b1010",
  "dhcp_name": "",
  "user_mod_time": 1358306852.0,
  "analysis_level": 0,
  "name": "ASUS 57363B",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
  "tag_set": [],
  "tags": [],
  "discover_time": 1354589328113.0,
  "oid": 2,
  "nb_name": ""
},
{
  "description": null,
  "mod_time": 1361400676.0,
  "macaddr": "BC:AE:C5:50:F9:FF",
  "vendor": "ASUS",
  "custom_type": "",
  "devclass": "node",
```

```
"ipaddr4": null,
"ipaddr6": null,
"parent_oid": null,
"vlanid": 1010,
"ip_discover": false,
"default_name": "ASUS 50F9FF",
"capture_oid": 0,
"id": 1,
"device_id": "bcaec550f9ff1010",
"dhcp_name": "",
"user_mod_time": 1358306852.0,
"analysis_level": 0,
"name": "ASUS 50F9FF",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1354589328113.0,
"oid": 0,
"nb_name": ""
}
]
},
{
"mod_time": 1359678776.0,
"description": "description3",
"name": "test3",
"deleted": false,
"oid": 3,
"dynamic": false,
"value": null,
"field": null,
"internal": false,
"cluster_id": "no_cluster",
"members": [
1,
2,
0
],
"member_set": [
{
"description": null,
"mod_time": 1361434711.0,
"macaddr": "F4:6D:04:1B:C0:3F",
"vendor": "ASUS",
"custom_type": "",
"devclass": "node",
"ipaddr4": null,
"ipaddr6": null,
"parent_oid": null,
"vlanid": 1010,
"ip_discover": false,
```

```
"default_name": "ASUS 1BC03F",
"capture_oid": 0,
"id": 2,
"device_id": "f46d041bc03f1010",
"dhcp_name": "",
"user_mod_time": 1358306852.0,
"analysis_level": 0,
"name": "ASUS 1BC03F",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1354589328113.0,
"oid": 1,
"nb_name": ""
},
{
  "description": null,
  "mod_time": 1361400676.0,
  "macaddr": "20:CF:30:57:36:3B",
  "vendor": "ASUS",
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": null,
  "ipaddr6": null,
  "parent_oid": null,
  "vlanid": 1010,
  "ip_discover": false,
  "default_name": "ASUS 57363B",
  "capture_oid": 0,
  "id": 3,
  "device_id": "20cf3057363b1010",
  "dhcp_name": "",
  "user_mod_time": 1358306852.0,
  "analysis_level": 0,
  "name": "ASUS 57363B",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
  "tag_set": [],
  "tags": [],
  "discover_time": 1354589328113.0,
  "oid": 2,
  "nb_name": ""
},
{
  "description": null,
  "mod_time": 1361400676.0,
  "macaddr": "BC:AE:C5:50:F9:FF",
  "vendor": "ASUS",
  "custom_type": "",
  "devclass": "node",
```

```
"ipaddr4": null,  
"ipaddr6": null,  
"parent_oid": null,  
"vlanid": 1010,  
"ip_discover": false,  
"default_name": "ASUS 50F9FF",  
"capture_oid": 0,  
"id": 1,  
"device_id": "bcaec550f9ff1010",  
"dhcp_name": "",  
"user_mod_time": 1358306852.0,  
"analysis_level": 0,  
"name": "ASUS 50F9FF",  
"custom_name": null,  
"cdp_name": "",  
"dns_name": "",  
"tag_set": [],  
"tags": [],  
"discover_time": 1354589328113.0,  
"oid": 0,  
"nb_name": ""  
}  
]  
}  
]
```

See Also:

"device-groups.py" on page 287

get_device_tags

Get device tags.

Syntax:

```
get_device_tags()
```

Parameters

None.

Return Value

A list of tag objects.

Sample Usage

```
>>> r = c.get_device_tags()
>>> import json
>>> print json.dumps(r, indent=2)
[
  {
    "mod_time": 1361924015.0,
    "oid": 5,
    "cluster_id": "no_cluster",
    "name": "apple"
  },
  {
    "mod_time": 1360017703.0,
    "oid": 1,
    "cluster_id": "no_cluster",
    "name": "test"
  }
]
```


get_devices

Get specific devices.

Syntax:

```
get_devices(device_oids)
```

Parameter

device_oids: List

A list of integers representing device IDs.

Return Value

A list of device objects, each including the following fields:

capture_oid: Integer

The object ID of the capture process which discovered the device.

cdp_name: String

The Cisco Discovery Protocol name of the device.

devclass: String

The device class.

Possible Values:

- gateway
- node
- pseudo
- remote

device_id: UTF-8 String

The device ID, guaranteed to be unique on one ExtraHop device.

Sample Value: u'0015c5ec20d40000'

device_oid: Integer

The device object ID (unique).

dhcp_name: String

The DHCP name of the device.

discover_time: Integer

The time at which the device was discovered as a 64-bit integer. Time is expressed in UTC.

Sample Value: 1279818934000.0

dns_name: String

The DNS name of the device.

ipaddr4: String

The IPv4 address of the device.

Sample value: 10.10.248.29

ipaddr6: String

The IPv6 address of the device.

Sample value: 2001:db8:1234:0000:0000:0000:0000:0000

macaddr: String

The MAC address of the device.

Sample Value: 74:86:7A:ED:79:48

mod_time: Integer

The device modification time as a 64-bit integer. Time is expressed in UTC.

nb_name: String

The NetBIOS name of the device.

vlanid: Integer

The VLAN ID of the device.

Sample Usage

```
# get devices
result = c.get_devices()
# print device with oid 0
if result != None and result[0] != None:
d = result[0]
print "raw data:\n\n", d
print "sample device:", d.oid, d.macaddr, d.ipaddr4, d.dns_name, d.nb_name, d.mod_
time
```

Sample Output

```
>>> r = c.get_devices()
>>> import json
>>> print json.dumps(r, indent=2)
[
  {
    "description": null,
    "mod_time": 1361400676.0,
    "macaddr": "BC:AE:C5:50:F9:FF",
    "vendor": "ASUS",
    "custom_type": "",
    "devclass": "node",
    "ipaddr4": null,
    "ipaddr6": null,
    "parent_oid": null,
    "vlanid": 1010,
    "ip_discover": false,
    "default_name": "ASUS 50F9FF",
    "capture_oid": 0,
    "id": 1,
    "device_id": "bcaec550f9ff1010",
    "dhcp_name": "",
    "user_mod_time": 1358306852.0,
    "analysis_level": 0,
    "name": "ASUS 50F9FF",
    "custom_name": null,
    "cdp_name": "",
    "dns_name": "",
    "tag_set": [],
    "tags": [],
    "discover_time": 1354589328113.0,
    "oid": 0,
```

```
"nb_name": ""
},
{
  "description": null,
  "mod_time": 1361434711.0,
  "macaddr": "F4:6D:04:1B:C0:3F",
  "vendor": "ASUS",
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": null,
  "ipaddr6": null,
  "parent_oid": null,
  "vlanid": 1010,
  "ip_discover": false,
  "default_name": "ASUS 1BC03F",
  "capture_oid": 0,
  "id": 2,
  "device_id": "f46d041bc03f1010",
  "dhcp_name": "",
  "user_mod_time": 1358306852.0,
  "analysis_level": 0,
  "name": "ASUS 1BC03F",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
  "tag_set": [],
  "tags": [],
  "discover_time": 1354589328113.0,
  "oid": 1,
  "nb_name": ""
}
]
```

get_devices_with_activity

Get the devices from a list of device object IDs that are active within a certain timeframe.

Syntax:

```
get_devices_with_activity(start, until, device_oids)
```

Parameters

device_oids: Integers

The device object IDs.

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

Return Value

A list of devices similar to those returned by `get_devices()`, but including activity.

Sample Usage

```
>>> r = c.get_devices_with_activity(-1800000, 0, [0, 1])
>>> import json
>>> print json.dumps(r, indent=2)
[
  {
    "description": null,
    "mod_time": 1361400676.0,
    "macaddr": "BC:AE:C5:50:F9:FF",
    "vendor": "ASUS",
    "custom_type": "",
    "devclass": "node",
    "ipaddr4": null,
    "ipaddr6": null,
    "parent_oid": null,
    "vlanid": 1010,
    "ip_discover": false,
    "default_name": "ASUS 50F9FF",
    "capture_oid": 0,
    "id": 1,
    "device_id": "bcaec550f9ff1010",
    "dhcp_name": "",
    "user_mod_time": 1358306852.0,
    "analysis_level": 0,
    "name": "ASUS 50F9FF",
    "custom_name": null,
    "cdp_name": "",
```

```
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1354589328113.0,
"activity": [
  "extrahop.device.net"
],
"oid": 0,
"nb_name": ""
},
{
  "description": null,
  "mod_time": 1361434711.0,
  "macaddr": "F4:6D:04:1B:C0:3F",
  "vendor": "ASUS",
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": null,
  "ipaddr6": null,
  "parent_oid": null,
  "vlanid": 1010,
  "ip_discover": false,
  "default_name": "ASUS 1BC03F",
  "capture_oid": 0,
  "id": 2,
  "device_id": "f46d041bc03f1010",
  "dhcp_name": "",
  "user_mod_time": 1358306852.0,
  "analysis_level": 0,
  "name": "ASUS 1BC03F",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
  "tag_set": [],
  "tags": [],
  "discover_time": 1354589328113.0,
  "activity": [
    "extrahop.device.app",
    "extrahop.device.net"
  ],
  "oid": 1,
  "nb_name": ""
}
]
```

get_dynamic_group_devices

Get members of a certain group.

Syntax:

```
get_dynamic_group_devices(start, until, group_oid, [{"offset":offset,
"limit":limit})
```

Parameters

group_oid: Integer

The group object ID.

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates the absolute time since the epoch (January 1, 1970). If the value is negative, it indicates the relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The until time, expressed in milliseconds. If the value is positive, it indicates the absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

Options

Optional support for pagination of results. For example, if you have 100 devices and you ask for offset=0, limit=10, the firsts 10 devices will be returned. If you ask for offset=50, limit=50, the last fifty devices will be returned.

limit: Integer

The number of devices to return.

offset: Integer

The offset of the devices to return.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.get_dynamic_group_devices(-1800000, 0, 10, {"limit": 2})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 39,
  "devices": [
    {
      "description": null,
      "mod_time": 1361917464.0,
      "macaddr": "C8:2A:14:05:87:1B",
```

```
"vendor": "Apple",
"custom_type": "",
"devclass": "node",
"ipaddr4": null,
"ipaddr6": null,
"parent_oid": null,
"vlanid": 1010,
"ip_discover": false,
"default_name": "Apple 05871B",
"capture_oid": 0,
"id": 2349,
"device_id": "c82a1405871b1010",
"dhcp_name": "",
"user_mod_time": 1361917464.0,
"analysis_level": 0,
"name": "Apple 05871B",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1361917454041.0,
"activity": [
    "extrahop.device.app",
    "extrahop.device.net"
],
"oid": 2348,
"nb_name": ""
},
{
"description": null,
"mod_time": 1361809392.0,
"macaddr": "D4:9A:20:01:4B:62",
"vendor": "Apple",
"custom_type": "",
"devclass": "node",
"ipaddr4": null,
"ipaddr6": null,
"parent_oid": null,
"vlanid": 1010,
"ip_discover": false,
"default_name": "Apple 014B62",
"capture_oid": 0,
"id": 2193,
"device_id": "d49a20014b621010",
"dhcp_name": "",
"user_mod_time": 1360355335.0,
"analysis_level": 0,
"name": "Apple 014B62",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
```

```
"tags": [],  
"discover_time": 1360355335070.0,  
"activity": [  
    "extrahop.device.net"  
],  
"oid": 2192,  
"nb_name": ""  
    }  
]  
}
```

See Also:

"device-groups.py" on page 287

get_dynamic_groups_count

Get the number of members in dynamic groups.

Syntax:

```
get_dynamic_groups_count(start, until)
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, it indicates the current capture time.

until: Number

The until time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

Return Value

A JSON object (which gets turned into a Python dictionary), where each key is a dynamic group object ID string and the corresponding value is the device count.

Sample Usage

```
>>> r = c.get_dynamic_groups_count(c.tfrom, c.tuntil)
>>> import json
>>> print json.dumps(r, indent=2)
{
  "1": 219,
  "2": 64
}
```

get_exstats

A specialized request type to extract raw metrics from a specific capture or device.

If you need only totals, use **get_exstats_group()** or **get_exstats_total()**.

To extract one custom metric, filter the results using `custom_count=metric_name` to prevent pulling all custom metrics. You can also filter the results with a regex using `custom_count?some_regex` syntax. It is a best practice to use precise time intervals in order to limit the amount of data retrieved.

Syntax:

```
get_exstats(["stat_name", "object_type", ["object_id", from_time, until_time],
["field_spec"], [{"cycle": "cycle"}, {"stat_key": "stat_key"}, {"stat_key_match_
flags": stat_key_match_flags}, {"topn_max": topn_max}]])
```

Using `get_exstats` with no parameters will return every datapoint in the time range.

```
get_exstats()
```

Parameters

field_spec: String

A valid field names for which to retrieve the stats. (Refer to **Metrics** on page 162 for a complete list.)

Possible field spec for the `extrahop.capture.net` metric:

```
field_spec=["pkts", "bytes"]
```

object_spec: List of 3-Element Tuples

Each tuple consists of the object ID and a time range for which to retrieve the metrics.

from_time: Integer

The "from" time as a 64-bit integer, converted from relative specification in the request, expressed in milliseconds. If the value is positive or zero, it indicates absolute time since the epoch, Jan. 1, 1970. If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time).

Sample Value: 1230798324420.0

object_id: Integer

The object ID for the capture, device, or application object.

until_time: Integer

The end time as a 64-bit integer, expressed in milliseconds, during which activity for the above metrics ended. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, indicates the current capture time. Time is expressed in UTC.

Sample Value: 1279818934000.0

Example:

```
object_spec=[(capture.oid, -180000, 0)]
```

object_type: String

The type of object. Supported values are:

- application
- capture
- device

options: Dictionary

A dictionary of zero or more option name and value pairs. Option descriptions are:

cycle: String

Level of metric aggregation.

Possible values are:

- "fast" - Data aggregated at 30-second intervals.
- "medium" - Data aggregated at 5-minute intervals.
- "slow" - Data aggregated at 1-hour intervals.

Note: As metrics age, fast stats get aged out first, then medium, then slow. The speed at which this happens is determined dynamically based on the amount of available storage.

Example:

```
options={'cycle':"slow"}
```

stat_key: String

In rare cases, detailed metrics have top-level stat keys. One example of that is database metrics by database instance.

This example of pretty-printed metrics for database requests broken down by database instance:

```
cycle : fast
abs from : Wed Dec 31 23:55:24 2008 [1230796524000]
abs until : Thu Jan 1 00:25:24 2009 [1230798324000]
=====
time : Thu Jan 1 00:13:38 2009 [1230797618000]
object id : -1
stat key : jbossdb
--- req -----
key: addr 10.10.1.152 host evil.extrahop.net device 7
value type: count
99
```

stat_key_match_flags: Integer

Specifies what type of match to make with the stat_key. The default value is 1.

Possible Values:

- 1 : Exact match.
- 2 : Prefix match.

Example:

```
options={'stat_key': 'jbossdb', 'stat_key_match_flags': 1}
```

topn_max: Integer

The limit on the size of top-N stats. The default limit is 1000 entries per time cycle.

Possible Values:

- 0 : Uses the default limit of 1000 entries.
- 1+ : Limit to this number of results.

Example:

```
options={'topn_max':10000}
```

Note: Increasing this limit allows larger result sets to be returned, which may result in increased memory usage.

stat_name: String

The metric name for which to retrieve the stats. (Refer to **Metrics** on page 162 for a complete list.)

Return Value

A set of stats associated with device or capture and metrics specified. Specific result structure depends on the metric queried and the order may be slightly different from what is described below.

abs_from: Integer

The absolute "from" time as a 64-bit integer, converted from relative specification in the request, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch, Jan. 1, 1970. If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, it indicates the current capture time. Time is expressed in UTC.

Sample Value:1230798324420.0

abs_until: Integer

The absolute "until" time as a 64-bit integer, converted from relative specification in the request, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch, Jan. 1, 1970. If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, it indicates the current capture time. Time is expressed in UTC.

Sample Value:1230798324420.0

cycle: String

The level of metric aggregation.

Possible Values:

- "fast" : Data aggregated at 30-second intervals.
- "medium" : Data aggregated at 5-minute intervals.
- "slow" : Data aggregated at 1-hour intervals.

stats: List

A list of metric objects returned. This can include the following common fields as well as other names of metrics for fields queried in the `field_spec` parameter of the request.

application_oid: Integer

The object ID of the application queried.

Sample Value: 0

capture_oid: Integer

The object ID of the capture queried.

Sample Value: 0

device_oid: Integer

The object ID of the device queried.

Sample Value: 0

stat_key: String

The top-level stat key for the rare cases when a detailed metrics has a top-level stat key. One example of that is database metrics by database instance.

Sample Value: 'jbossdb'

time: Time in UTC

The time during which the above metrics were recorded.

Sample Value: 1230796801000.0

Sample Usage

```
import time
# get device packets_in, bytes_in
result = c.get_exstats ("extrahop.device.net",
                       "device",
                       [(0, -1800000, 0)],
                       ["pkts_in", "bytes_in"],
                       {'cycle':"fast"})

# print raw result
print result

# pretty-print results
for stat in result.stats:
    print time.ctime(long(stat.time) / 1000), stat.pkts_in, stat.bytes_in
```

Sample Raw Result

```
{'abs_from': 1230796524420.0,
 'abs_until': 1230798324420.0,
 'cycle': u'fast',
 'stats':
  [ {'stat_key': '',
     'oid': 0,
     'time': 1230796801000.0,
     'pkts_in': 17,
     'bytes_in': 1905}, ...]
}
```

Sample Pretty-Printed Result

```
Thu Jan 1 00:00:01 2009 17 1905
Thu Jan 1 00:00:03 2009 23 2726
Thu Jan 1 00:00:07 2009 59 7355
Thu Jan 1 00:00:14 2009 58 8602
```

See Also:

"activity-stats.py" on page 278

"applications.py" on page 279

"bridge-devcount.py" on page 281

"count.py" on page 282

get-alerts-fired.py on page 295

print-trigger-runtime-log.py on page 299

get_exstats_group

A specialized request type to extract raw metrics from a device group, grouped by device.

Syntax:

```
get_exstats_group(("stat_name", "object_type", ["object_id", from_time, until_time], ["field_spec"], [{"cycle": "cycle"}, {"stat_key": "stat_key"}, {"stat_key_match_flags": stat_key_match_flags}, {"topn_max": topn_max}]])
```

Parameters

field_spec: String

A set of field names for which to retrieve the stats. (Refer to **Metrics** on page 162 for a complete list.)

Example: A possible field spec for the extrahop.capture.net metric:

```
field_spec=["rsp", "status code"]
```

object_spec: List of 3-Element Tuples

Each tuple consists of the object ID and a time range for which to retrieve the metrics.

from_time: Integer

The "from" time as a 64-bit integer, converted from relative specification in the request, expressed in milliseconds. If the value is positive or zero, it indicates absolute time since the epoch, Jan. 1, 1970. If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time).

Sample Value:1230798324420.0

object_id: Integer

The object ID for the capture, device, or application object.

until_time: Integer

The "until" time as a 64-bit integer, converted from relative specification in the request, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch, Jan. 1, 1970. If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, it indicates the current capture time. Time is expressed in UTC.

Sample Value:1230798324420.0

Example:

```
object_spec=[(capture.oid, -180000, 0)]
```

object_type: String

The type of object. Supported values are:

- application
- capture
- device

options: Dictionary

A dictionary of zero or more option name and value pairs. Option descriptions are:

cycle: String

Level of metric aggregation.

Possible values are:

- "fast" - Data aggregated at 30-second intervals.
- "medium" - Data aggregated at 5-minute intervals.
- "slow" - Data aggregated at 1-hour intervals.

Note: As metrics age, fast stats get aged out first, then medium, then slow. The speed at which this happens is determined dynamically based on the amount of available storage.

Example:

```
options={'cycle':"slow"}
```

stat_key: String

In rare cases, detailed metrics have top-level stat keys. One example of that is database metrics by database instance.

This example of pretty-printed metrics for database requests broken down by database instance:

```
cycle : fast
abs from : Wed Dec 31 23:55:24 2008 [1230796524000]
abs until : Thu Jan 1 00:25:24 2009 [1230798324000]
=====
time : Thu Jan 1 00:13:38 2009 [1230797618000]
object id : -1
stat key : jbossdb
--- req -----
key: addr 10.10.1.152 host evil.extrahop.net device 7
value type: count
99
```

stat_key_match_flags: Integer

Specifies what type of match to make with the stat_key. The default value is 1.

Possible Values:

- 1 : Exact match.
- 2 : Prefix match.

Example:

```
options={'stat_key': 'jbossdb', 'stat_key_match_flags': 1}
```

topn_max: Integer

The limit on the size of top-N stats. The default limit is 1000 entries per time cycle.

Possible Values:

- 0 : Uses the default limit of 1000 entries.
- 1+ : Limit to this number of results.

Example:

```
options={'topn_max':10000}
```

Note: Increasing this limit allows larger result sets to be returned, which may result in increased memory usage.

stat_name: String

The metric name for which to retrieve the stats. (Refer to **Metrics** on page 162 for a complete list.)

Return Value

A set of group metrics associated with device or capture and metrics specified. Specific result structure depends on the metric queried. Order may be slightly different from what is described below.

abs_from: Time in UTC

The absolute "from" time in milliseconds converted from relative specification in the request. If the value is positive, it indicates absolute time since the epoch, Jan. 1, 1970. If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, indicates the current capture time.

Sample Value:1230798324420.0

abs_until: Time in UTC

Absolute "until" time in milliseconds converted from relative specification in the request. If the value is positive, it indicates absolute time since the epoch, Jan. 1, 1970. If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, indicates the current capture time.

Sample Value:1230798324420.0

cycle: String

The level of metric aggregation.

Values:

- fast - Data aggregated at 30-second intervals.
- medium - Data aggregated at 5-minute intervals
- slow - Data aggregated at 1-hour intervals

stats: List

A list of metric objects returned.

stat_key: String

In rare cases, detailed metrics have top-level stat keys. One example of that is database metrics by database instance.

Sample Value: 'jbossdb'

capture_oid, device_oid, or application_oid: Integer

Object ID of the capture, device, or application queried.

Sample Value: 0

time: Integer

Time in UTC during which the above metrics were recorded.

Sample Value: 1230796801000.0

Metrics for fields queried in the `field_spec` parameter of the request

Sample Usage

```
import time
# get pkts_in, bytes_in for an activity group 'TCP'
result = c.get_exstats_group ("extrahop.device.net",
    "activity_group",
    [("extrahop.device.tcp", -1800000, 0)],
    ["pkts_in","bytes_in"],
    {'cycle':"fast"})
# print raw result
print result
# pretty-print results
for stat in result.stats:
    print "time:", time.ctime(long(stat.time) / 1000)
    print "device:", stat.oid
    print "bytes_in", stat.bytes_in
    print "pkts_in", stat.pkts_in
```

Sample Raw Result

```
{'abs_until': 1230798324000.0,
 'cycle': u'fast',
 'stats': [
 {'pkts_in': 6,
  'stat_key': None,
  'oid': 4,
  'time': 1230798324000.0,
  'bytes_in': 478},
 {'pkts_in': 2461,
  'stat_key': None,
  'oid': 17,
  'time': 1230798324000.0,
  'bytes_in': 3062760}
 ...
 ]}
```

Sample Pretty-Printed Result

```
time: Thu Jan  1 00:25:24 2009
device: 4
bytes_in 478
pkts_in 6
time: Thu Jan  1 00:25:24 2009
device: 17
bytes_in 3062760
pkts_in 2461
```

See Also:

"count.py" on page 282

"dataset.py" on page 285

"dns-queries.py" on page 293

"exstats-ecm.py" on page 294

l7-proto-details.py on page 297

topn-dset.py on page 303

topn-sset.py on page 304

get_exstats_next

Get next chunked results returned by exstats. Use this function when querying for results against distributed ECM groups.

Syntax:

```
get_exstats_next(xid)
```

Parameter

xid: Integer

The ID of the chunked object to query against (returned by "get_exstats", "get_exstats_total", or "get_exstats_group" calls run against a distributed ECM group).

Return Value

An object representing chunked results returned by exstats.

Sample Usage

Refer to the **examples** directory in the Python SDK for an example.

get_exstats_total

A specialized request type to extract raw metrics from a specific capture, device, or device group (user-defined or activity) and return a sum across those metrics for a specified time period.

Syntax:

```
get_exstats_total("stat_name", "object_type", object_spec, field_spec,
[{"cycle": "cycle"}, {"stat_key": "stat_key"}, {"stat_key_match_flags": "stat_key_
match_flags"}, {"topn_max": "topn_max"}])
```

Parameters

field_spec: String

A valid field names for which to retrieve the stats. (Refer to **Metrics** on page 162 for a complete list.)

Possible field spec for the extrahop.capture.net metric:

```
field_spec=["pkts", "bytes"]
```

object_spec: List of 3-Element Tuples

Each tuple consists of the object ID and a time range for which to retrieve the metrics.

from_time: Integer

The "from" time as a 64-bit integer, converted from relative specification in the request, expressed in milliseconds. If the value is positive or zero, it indicates absolute time since the epoch, Jan. 1, 1970. If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time).

Sample Value: 1230798324420.0

object_id: Integer

The object ID for the capture, device, or application object.

until_time: Integer

The end time as a 64-bit integer, expressed in milliseconds, during which activity for the above metrics ended. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, indicates the current capture time. Time is expressed in UTC.

Sample Value: 1279818934000.0

Example:

```
object_spec=[(capture.oid, -180000, 0)]
```

object_type: String

The type of object, which is a 'capture', 'device', or 'application'.

stat_name: String

The metric name for which to retrieve the stats. (Refer to **Metrics** on page 162 for a complete list.)

options: Dictionary

A dictionary of zero or more option name and value pairs. Option descriptions are:

cycle: String

Level of metric aggregation.

Possible values are:

- "fast" - Data aggregated at 30-second intervals.
- "medium" - Data aggregated at 5-minute intervals.
- "slow" - Data aggregated at 1-hour intervals.

Note: As metrics age, fast stats get aged out first, then medium, then slow. The speed at which this happens is determined dynamically based on the amount of available storage.

Example:

```
options={'cycle':"slow"}
```

stat_key: String

In rare cases, detailed metrics have top-level stat keys. One example of that is database metrics by database instance.

This example of pretty-printed metrics for database requests broken down by database instance:

```
cycle : fast
abs from : Wed Dec 31 23:55:24 2008 [1230796524000]
abs until : Thu Jan 1 00:25:24 2009 [1230798324000]
=====
time : Thu Jan 1 00:13:38 2009 [1230797618000]
object id : -1
stat key : jbossdb
--- req -----
key: addr 10.10.1.152 host evil.extrahop.net device 7
value type: count
99
```

stat_key_match_flags: Integer

Specifies what type of match to make with the stat_key. The default value is 1.

Possible Values:

- 1 : Exact match.
- 2 : Prefix match.

Example:

```
options={'stat_key': 'jbossdb', 'stat_key_match_flags': 1}
```

topn_max: Integer

The limit on the size of top-N stats. The default limit is 1000 entries per time cycle.

Possible Values:

- 0 : Uses the default limit of 1000 entries..
- 1+ : Limit to this number of results.

Example:

```
options={'topn_max':10000}
```

Note: Increasing this limit allows larger result sets to be returned, which may result in increased memory usage.

stat_name: String

The metric name for which to retrieve the stats.

Return Value

A set of totals (metrics summed over a period of time) associated with device or capture and metrics specified. Specific result structure depends on the metric queried. Order may be slightly different from what is described below.

abs_from: Integer

The absolute "from" time as a 64-bit integer, converted from relative specification in the request, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch, Jan. 1, 1970. If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, it indicates the current capture time. Time is expressed in UTC.

Sample Value:1230798324420.0

abs_until: Integer

The absolute "until" time as a 64-bit integer, converted from relative specification in the request, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch, Jan. 1, 1970. If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from current capture time). If the value is 0, it indicates the current capture time. Time is expressed in UTC.

Sample Value:1230798324420.0

cycle: String

The level of metric aggregation.

Possible Values:

- "fast" : Data aggregated at 30-second intervals.
- "medium" : Data aggregated at 5-minute intervals.
- "slow" : Data aggregated at 1-hour intervals.

stats: List

A list of metric objects returned. This can include the following common fields as well as other names of metrics for fields queried in the `field_spec` parameter of the request.

application_oid: Integer

The object ID of the application queried.

Sample Value: 0

capture_oid: Integer

The object ID of the capture queried.

Sample Value: 0

device_oid: Integer

The object ID of the device queried.

Sample Value: 0

stat_key: String

The top-level stat key for the rare cases when a detailed metrics has a top-level stat key. One example of that is database metrics by database instance.

Sample Value: 'jbossdb'

time: Time in UTC

The time during which the above metrics were recorded.

Sample Value: 1230796801000.0

Sample Usage 1

```
# get total pkts_in, bytes_in for device with ID 0
print ">>> totals <<<"
result = c.get_exstats_total("extrahop.device.net",
    "device",
    [(0, -1800000, 0)],
    ["pkts_in", "bytes_in"],
    {'cycle':"fast"})
# print raw result
print result
# pretty-print results
for stat in result.stats:
    print "Total: ", time.ctime(long(stat.time) / 1000), stat.pkts_in, stat.bytes_in
```

Sample Raw Result 1

```
{ 'abs_from': 1230796524000.0,
  'abs_until': 1230796524000.0,
  'cycle': u'fast',
  'stats':
  [ {'stat_key': None,
    'oid': -1,
    'time': 1230798324420.0,
    'pkts_in': 38628,
    'bytes_in': 5497630} ]
```

Sample Pretty-Printed Result 1

```
Total: Thu Jan 1 00:25:24 2009 38628 5497630
```

Sample Usage 2

```
# get total pkts_in, bytes_in for activity group 'TCP'
print ">>> totals <<<"
result = c.get_exstats_total("extrahop.device.net",
    "activity_group",
    [(0, -1800000, 0)],
    ["pkts_in", "bytes_in"],
    options)
```



```
# print raw result
print result
# pretty-print results
for stat in result.stats:
    print "Total: ", time.ctime(long(stat.time) / 1000), stat.pkts_in, stat.bytes_in
```

Sample Raw Result 2

```
{ 'abs_from': 1230796524000.0,
  'abs_until': 1230796524000.0,
  'cycle': u'fast',
  'stats':
  [ {'stat_key': None,
    'oid': -1,
    'time': 1230798324000.0,
    'pkts_in': 1175878,
    'bytes_in': 571273572.0} ]
```

Sample Pretty-Printed Result 2

```
Total: Thu Jan 1 00:25:24 2009 1173752 571051328.0
```

See Also:

"count.py" on page 282

"dataset.py" on page 285

get_extrahop_version

Get the version of the ExtraHop system.

Syntax:

```
get_extrahop_version ()
```

Parameters:

None.

Return Value:

The ExtraHop system version number.

get_flex_grid_data

Get data associated with a specific flex grid.

Syntax:

```
get_flex_grid_data(flexgrid_oid)
```

Parameter

flexgrid_oid: Integer

The object ID of the flex grid to retrieve.

Return Value

An object representing flex grid rows and columns.

Sample Usage

```
>>> r=c.get_flex_grid_data(2)
>>> import json
>>> print json.dumps(r, indent=2)
{
  "rows": [
    {
      "table": 2,
      "object_type": "application",
      "id": 2,
      "object_id": 0,
      "target": {
        "mod_time": 1436940115515,
        "comment": "",
        "object_type": "application",
        "name": "DNS Dashboard",
        "oid": 2
      }
    },
    {
      "table": 2,
      "object_type": "application",
      "id": 3,
      "object_id": 17,
      "target": {
        "mod_time": 1436940115515,
        "comment": "",
        "object_type": "application",
        "name": "DNS Dashboard",
        "oid": 2
      }
    }
  ],
  "columns": [
    {
      "name": "Requests",
      "table": 2,
```

```
    "field_name": "req",
    "stat_class": "application",
    "stat_type": "dns",
    "id": 17,
    "use_binary_units": false
  },
  {
    "name": "Request Timeouts",
    "table": 2,
    "field_name": "req_timeout",
    "stat_class": "application",
    "stat_type": "dns",
    "id": 18,
    "use_binary_units": false
  },
  {
    "name": "Truncated Requests",
    "table": 2,
    "field_name": "req_trunc",
    "stat_class": "application",
    "stat_type": "dns",
    "id": 19,
    "use_binary_units": false
  }
]
}
```

get_running_config

Get the running config. To view and modify the running config, you must log in with the **setup** user account.

Syntax:

```
get_running_config()
```

Parameters

None.

Return Value

A Python dictionary.

Sample Usage

```
>>> running_config = c.get_running_config()
>>> print running_config['hostname']
{'hostname': u'extrahop'}
```

See Also:

pull-running-config.py on page 300

set-host-name.py on page 301

get_ssl_decrypt_config

Retrieve information about a single SSL certificate on the ExtraHop.

Syntax:

```
get_ssl_decrypt_config(ssl_id)
```

Parameters

ssl_id: String
The object ID of the SSL certificate to return details for.

Return Value

An object representing the SSL certificate requested which includes the following fields:

id: String
The unique object ID of the certificate.

name: String
The certificate name.

enabled: Boolean
True if the certificate is enabled.

cert_pem: string
The .pem file information of the public certificate.

Note: The private key is never given out after the initial user upload.

Sample Usage

```
get_ssl_decrypt_config("855349E09FCDF4D2D5A01D5DC1A94C7AEADF2D82")
```

get_ssl_decrypt_configs

Retrieve information about all SSL certificates on the ExtraHop.

Syntax:

```
get_ssl_decrypt_configs()
```

Parameters

None.

Return Value

A list of objects containing the information for all SSL certificates on the ExtraHop. Each object contains the following fields:

- id:** String
The unique object ID of the certificate.
- name:** String
The certificate name.
- enabled:** Boolean
True if the certificate is enabled.
- cert_pem:** string
The .pem file information of the public certificate.

Note: The private key is never given out after the initial user upload.

Sample Usage

```
get_ssl_decrypt_configs()
```

See Also:

ssl-key-expiration.py on page 301

get_trigger

Get data associated with a specific trigger.

Syntax:

```
get_trigger(trigger_oid)
```

Parameter

trigger_oid: Integer
The object ID of the trigger to retrieve.

Return Value

A specified trigger object.

Sample Usage

```
>>> r = c.get_trigger(8)
>>> import json
>>> print json.dumps(r, indent=2)
{
  "description": "",
  "deleted": false,
  "is_system_event": false,
  "disabled": false,
  "cluster_id": "no_cluster",
  "id": 8,
  "hints": "{\"payloadBytes\":0,\"packetCapture\":false}",
  "mod_time": 1361924466.0,
  "apply_all": true,
  "name": "app-test",
  "script": "Application(\"myapp\").commit();",
  "author": "Setup",
  "event": "HTTP_RESPONSE",
  "priority": 0,
  "debug": false,
  "prop_id": "no_cluster:app-test"
}
```

get_triggers

Get a list of triggers.

Syntax:

```
get_triggers(start)
```

Parameter

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

Return Value

A list of triggers modified since the specified start time.

Sample Usage

```
>>> r = c.get_triggers(1361394738)
>>> import json
>>> print json.dumps(r, indent=2)
[
  {
    "description": "",
    "deleted": false,
    "assigned_objects": [],
    "is_system_event": false,
    "disabled": false,
    "cluster_id": "no_cluster",
    "id": 8,
    "hints": "{\"payloadBytes\":0,\"packetCapture\":false}",
    "mod_time": 1361924466.0,
    "apply_all": true,
    "name": "app-test",
    "author": "Setup",
    "script": "Application(\"myapp\").commit();",
    "event": "HTTP_RESPONSE",
    "priority": 0,
    "debug": false,
    "prop_id": "no_cluster:app-test"
  },
  {
    "description": "",
    "deleted": false,
    "assigned_objects": [
      {
        "mod_time": 1361924904.0,
        "deleted": false,
        "object_type": "device",
        "id": 3,
        "object_id": 14
      }
    ]
  }
]
```



```
}
],
  "is_system_event": false,
  "disabled": false,
  "cluster_id": "no_cluster",
  "id": 9,
  "hints": "{\"packetCapture\":false}",
  "mod_time": 1361924904.0,
  "apply_all": false,
  "name": "flow-classify",
  "author": "Setup",
  "script": "if (Flow.server.port == 443) {\n    Flow.setApplication(\"HTTPS_
14\");\n}",
  "event": "FLOW_CLASSIFY",
  "priority": 0,
  "debug": false,
  "prop_id": "no_cluster:flow-classify"
}
]
```

get_network

Get details about a specific network.

Syntax:

```
get_network(network_oid)
```

Parameters:

network_oid: Integer
The unique object ID of the network.

Return Value:

Network details.

get_networks

Get all networks on the ExtraHop system.

Syntax:

```
get_networks ()
```

Parameters:

None.

Return Value:

A list of networks.

get_packetcapture

Get details about a specific packet capture.

Syntax;

```
get_packetcapture(capture_oid)
```

Parameters:

capture_oid: Integer
The unique object ID of the capture.

Return Value:

Packet capture details.

Sample Output:

```
[{
  "id": "6148120262589022215",
  "name": "Gettin a pkt cap",
  "tstart_usec": 1431631852130753,
  "tend_usec": 1431631933221447,
  "pkts": 20,
  "bytes": 1685,
  "device_id1": "f452148c61400000",
  "device_id2": "fff494090a0a0000",
  "port1": 80,
  "port2": 58073,
  "ipaddr1": "54.173.216.34",
  "ipaddr2": "10.10.9.148",
  "vlanid": 1010,
  "ipproto": 6,
  "l7proto": "HTTP",
  "snaplen": 100
}]
```

get_packetcaptures

Get metadata about all packet captures on the ExtraHop system.

Syntax:

```
get_packetcaptures ()
```

Parameters:

None.

Return Value:

A list of packet captures with metadata.

Sample Output:

```
[{
  "id": "6148120262589022215",
  "name": "Gettin a pkt cap",
  "tstart_usec": 1431631852130753,
  "tend_usec": 1431631933221447,
  "pkts": 20,
  "bytes": 1685,
  "device_id1": "f452148c61400000",
  "device_id2": "fff494090a0a0000",
  "port1": 80,
  "port2": 58073,
  "ipaddr1": "54.173.216.34",
  "ipaddr2": "10.10.9.148",
  "vlanid": 1010,
  "ipproto": 6,
  "l7proto": "HTTP",
  "snaplen": 100
}]
```

get_user

Get details about a specific user.

Syntax:

```
get_user("username")
```

Parameters:

username: String
The name of the user.

Return Value:

User details.

get_users

Get a list of users on the ExtraHop system.

Syntax:

```
get_users ()
```

Parameters

None.

Return Value

A list of users.

get_user_apikey

Get a specific API key for a user.

Syntax:

```
get_user_apikey("username", "key_id")
```

Parameters:

key_id: String
The unique API key.

username: String
The name of the user.

Return Value:

An API key.

get_user_apikeys

Get all API keys associated with a user.

Syntax:

```
get_user_apikeys("username")
```

Parameters

username: String
The name of the user.

Return Value

A list of API keys.

Setters

Configurations

save_ssl_decryption_key(*"desc", cert, "key", enable*)
Upload SSL certificate.

set_running_config(*new_config, save*)
Set and optionally save the running config.

Customizations

apply_customization(*customization_oid*)
Apply a customization backup.

create_customization(*"name"*)
Create a new customizations backup.

delete_customization(*customization_oid*)
Delete a specific customization.

Device Groups

add_device_group_members(*group_oid, members*)
Add members to a device group.

delete_device_groups(*group_oids*)
Delete certain device groups.

new_device_group(*"name", "comment", members*)
Create a new device group (version 3.8 and later).

new_dynamic_device_group(*"name", "comment", "field", "value"*)
Create a new dynamic device group.

remove_device_group_members(*group_oid, members*)
Remove members from a device group.

update_device_group(*group_oid, "name", "comment", members*)
Update existing device groups.

update_device_group_members(*group_oid, members*)
Update device group members.

update_dynamic_device_group(*group_oid, "name", "comment", "field", "value"*)
Update a dynamic device group.

Devices

create_custom_device(*{parameters}*)
Create a new custom device.

create_custom_device_criterion(*device_oid, {criterion_details}*)
Create a new criterion for a custom device.

delete_custom_device(*device_oid*)
Delete a custom device from the ExtraHop system.

delete_custom_device_criterion(*device_oid, criterion_oid*)
Delete a specific criterion.

update_custom_device(*device_oid, updates*)
Update an existing custom device.

update_device({"oid": *oid*, ["custom_name": *custom_name*], ["comment": *comment*], ["custom_type": *custom_type*], ["vendor": *vendor*], ["default_name": *default_name*]})
Update an existing device.

ExtraHop

is_ecm()
Returns true if the ExtraHop system is an ECM.

register_license("product_key")
Automatically apply the ExtraHop license and restart the capture.

restart_service("service_name")
Restart system services.

Flex-Grids

add_flex_grid_rows(*flexgrid_oids*, "object_type", *object_ids*)
Add rows to specific flex grids (e.g., assign objects to flex grids).

remove_flex_grid_rows(*flexgrid_oid*, *object_ids*, "object_type")
Delete rows from a specific flex grid (e.g., remove objects from a flex grid).

remove_flex_grids(*flexgrid_oids*)
Delete rows from specific flex grids (e.g., remove objects from a flex grid).

save_flex_grid(*flexgrid_oid*, "name", "comment", "object_type", *columns*)
Create or modify a flex grid.

Networks

update_network(*network_oid*, *updates*)
Make updates to a specific network.

Packet Captures

delete_packetcapture(*capture_oid*)
Delete a specific packet capture.

Tags

device_tag_add("tag_name", *device_oids*)
Add device tag to specific devices.

device_tag_delete(*tag_names*)
Delete device tag.

device_tag_remove("tag_name", *device_oids*)
Remove device tag from specific devices.

device_tag_rename("old_name", "new_name")
Rename device tag.

Triggers

apply_trigger(*trigger_oid*, "object_type", *object_ids*)
Assign a specific trigger.

delete_triggers(*trigger_oids*)
Delete specific triggers.

remove_trigger(*trigger_oid*, "object_type", *object_ids*)
Remove assignments from a specific trigger.

set_trigger(*name*, *script*, *event*, [*trigger_oid*], [*comment*], [*priority*], [*debug*], [*novalidate*], [*disabled*], [*hints*], ["author"], [*apply_all*])

Add a specific trigger.

Users

update_user(*username, updates*)
Make updates to a specific user.

Whitelists

add_to_whitelist on page 96 (*device_oids*)
Add devices to the ExtraHop whitelist.

remove_from_whitelist on page 120 (*device_oids*)
Remove devices from the ExtraHop whitelist.

add_device_group_members

Add members to a device group.

Syntax:

```
add_device_group_members(group_oid, members)
```

Parameters

group_oid: Integer

The object ID of the device group to be updated.

members: List

A list of integers representing device object IDs.

Return Value

An updated device group.

Sample Usage

```
>>> r = c.add_device_group_members(12, [10, 11])
>>> import json
>>> print json.dumps(r, indent=2)
{
  "mod_time": 1361988235.0,
  "comment": "my group comment",
  "name": "new name",
  "deleted": false,
  "oid": 12,
  "dynamic": false,
  "value": null,
  "field": null,
  "internal": false,
  "cluster_id": "no_cluster",
  "members": [
    8,
    10,
    11,
    99
  ],
  "member_set": [
    {
      "comment": null,
      "mod_time": 1361400676.0,
      "macaddr": "00:1E:7A:B1:B5:00",
      "vendor": "Cisco",
      "custom_type": "",
      "devclass": "gateway",
      "ipaddr4": null,
      "ipaddr6": null,
      "parent_oid": null,
      "vlanid": 7,
      "ip_discover": false,
```

```
"default_name": "Cisco B1B500",
"capture_oid": 0,
"id": 9,
"device_id": "001e7ab1b5000007",
"dhcp_name": "",
"user_mod_time": 1358306852.0,
"analysis_level": 1,
"name": "Cisco B1B500",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1354589328113.0,
"oid": 8,
"nb_name": ""
},
{
  "comment": null,
  "mod_time": 1361400676.0,
  "macaddr": "00:18:71:87:36:09",
  "vendor": "HP",
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": null,
  "ipaddr6": null,
  "parent_oid": null,
  "vlanid": 7,
  "ip_discover": false,
  "default_name": "HP 873609",
  "capture_oid": 0,
  "id": 11,
  "device_id": "0018718736090007",
  "dhcp_name": "",
  "user_mod_time": 1358306852.0,
  "analysis_level": 0,
  "name": "HP 873609",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
  "tag_set": [],
  "tags": [],
  "discover_time": 1354589328113.0,
  "oid": 10,
  "nb_name": ""
},
{
  "comment": null,
  "mod_time": 1361400676.0,
  "macaddr": "00:50:56:00:DD:07",
  "vendor": "VMware",
  "custom_type": "",
  "devclass": "node",
```

```
"ipaddr4": null,
"ipaddr6": null,
"parent_oid": null,
"vlanid": 1010,
"ip_discover": false,
"default_name": "VMware 00DD07",
"capture_oid": 0,
"id": 12,
"device_id": "00505600dd071010",
"dhcp_name": "",
"user_mod_time": 1358306852.0,
"analysis_level": 0,
"name": "VMware 00DD07",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1354589328113.0,
"oid": 11,
"nb_name": ""
},
{
  "comment": null,
  "mod_time": 1361400677.0,
  "macaddr": "70:56:81:90:C5:B1",
  "vendor": null,
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": null,
  "ipaddr6": null,
  "parent_oid": null,
  "vlanid": 1010,
  "ip_discover": false,
  "default_name": "Device 70568190c5b11010",
  "capture_oid": 0,
  "id": 100,
  "device_id": "70568190c5b11010",
  "dhcp_name": "",
  "user_mod_time": 1358306852.0,
  "analysis_level": 0,
  "name": "Device 70568190c5b11010",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
  "tag_set": [],
  "tags": [],
  "discover_time": 1354589336113.0,
  "oid": 99,
  "nb_name": ""
}
]
}
```

add_flex_grid_rows

Add rows to specific flex grids (e.g., assign objects to a flex grid).

Syntax:

```
add_flex_grid_rows(flexgrid_oids, "object_type", object_ids)
```

Parameters

flex_grid_oids: List

A list of the object IDs of the flex grids to which rows will be added.

object_type: String

The type of object to be added (i.e., "device", "device_group", or "application").

object_ids: List

A list of the object IDs of the objects that will be added to the flex grid.

Return Value

None.

add_to_whitelist

Add devices to the ExtraHop whitelist.

Syntax:

```
add_to_whitelist(device_oids)
```

Parameters

device_oids: List

A list of the object IDs of one or more devices to add to the ExtraHop whitelist.

Return Value

The modified devices.

Sample Usage

```
device_group_oid_to_whitelist = 4
device_oids = set() # Unique set of oids

members = c.get_device_group_devices(-18000000, 0, device_group_oid_to_whitelist,
{})

for device in members.devices:
    device_oids.add(device.oid)

c.add_to_whitelist(list(device_oids))
```

See Also:

whitelist.py on page 305

apply_customization

Apply a customization backup.

Syntax:

```
apply_customization(customization_oid)
```

Parameters:

customization_oid: Integer
The unique object ID of the customization.

Return Value:

None.

apply_trigger

Assign a specific trigger.

Syntax:

```
apply_trigger(trigger_oid, "object_type", object_ids)
```

Parameters

object_ids: Integer

The object IDs of the objects to which the trigger will be assigned.

object_type: String

The type of object to which the trigger will be assigned (i.e., "device" or "device_group").

trigger_oid: Integer

The object ID of the trigger to be assigned.

Return Value

A trigger object.

create_custom_device

Create a new custom device.

Syntax:

```
create_custom_device({parameters})
```

Parameters:

parameters: Dictionary with the attribute names listed as keys.

Device object to replace the updated device. Include only the oid attribute (key) and the attributes to be changed. The object can have the following attributes:

author: String
(Optional) The custom device creator's name. The default value is "None".

comment: String
(Optional) A description of the custom device. The default value is "None".

device_id: Integer
The unique identifier of the device.

disabled: Boolean
False if the device is enabled.

name: String
The name of the custom device.

Return Value

Returns a dictionary representing the custom device that was created.

Sample Usage

```
cd = c.create_custom_device({'name': name,  
                             'device_id': name,  
                             'disabled': False,  
                             'author': "author",  
                             'comment': "a test custom device"})
```

See Also:

"custom-devices.py" on page 284

create_custom_device_criterion

Create a new criterion for a custom device.

Syntax:

```
create_custom_device_criterion(device_oid, {criterion_details})
```

Parameters:

device_oid: Integer

The unique object ID of the device.

criterion_details: Dictionary

Dictionary with the following optional criterion:

dst_port_max: Integer

The highest number in the destination port range.

dst_port_min: Integer

The lowest number in the destination port range.

ipaddr: String

The IP address of the device.

src_port_max: Integer

The highest number in the source port range.

src_port_min: Integer

The lowest number in the source port range.

vlan_max: Integer

The highest number in the VLAN range.

vlan_min: Integer

The lowest number in the VLAN range.

Return Value:

The custom device criterion.

See Also:

"custom-devices.py" on page 284

create_customization

Create a new customizations backup.

Syntax:

```
create_customization("name")
```

Parameters:

name: String
The name of the new customization backup.

Return Value:

A backup of customizations.

delete_custom_device

Delete a custom device from the ExtraHop system.

Syntax:

```
delete_custom_device(device_oid)
```

Parameters:

device_oid: Integer
The unique object ID of the device.

Return Value:

None.

delete_custom_device_criterion

Delete a specific criterion.

```
delete_custom_device_criterion(device_oid, criterion_oid)
```

Parameters:

criterion_oid: Integer
The unique object ID of the criterion.

device_oid: Integer
The unique object ID of the device.

Return Value:

None.

delete_customization

Delete a specific customization.

Syntax:

```
delete_customization(customization_oid)
```

Parameters:

customization_oid: Integer
The unique object ID of the customization.

Return Value

None.

delete_device_groups

Delete specific device groups.

Syntax:

```
delete_device_groups(group_oids)
```

Parameter

group_oids: List
The group object IDs of the groups to delete.

Return Value

None.

Sample Usage

```
>>> r = c.delete_device_groups([11])
>>> import json
>>> print json.dumps(r, indent=2)
null
```

See Also:

"device-groups.py" on page 287

delete_packetcapture

Delete a specific packet capture.

Syntax:

```
delete_packetcapture(capture_oid)
```

Parameters:

capture_oid: Integer
The unique object ID of the capture.

Return Value:

None.

delete_triggers

Delete specific triggers.

Syntax:

```
delete_triggers(trigger_oids)
```

Parameter

trigger_oids: List
A list of trigger object IDs to be deleted.

Return Value

None.

device_tag_add

Add device tag to specific devices.

Syntax:

```
device_tag_add("tag_name", device_oids)
```

Parameters

tag_name: String

The device tag name.

device_oids: List

A list of the object IDs of the devices to which to add the device tags.

Return Value

None.

Sample Usage

Refer to the **examples** directory in the Python SDK for an example.

See Also:

"device-tags.py" on page 290

device_tag_delete

Delete device tag.

Syntax:

```
device_tag_delete (tag_names)
```

Parameter

tag_names: List
A list of device tags to delete.

Return Value

None.

Sample Usage

Refer to the **examples** directory in the Python SDK for an example.

device_tag_remove

Remove device tag from specific devices.

Syntax:

```
device_tag_remove("tag_name", device_oids)
```

Parameters

tag_name: String

The device tag name.

device_oids: List

A list of object IDs of the devices from which to remove the device tags.

Return Value

None.

Sample Usage

Refer to the examples directory in the Python SDK for an example.

device_tag_rename

Rename device tag.

Syntax:

```
device_tag_rename("old_name", "new_name")
```

Parameters

old_name: String
The old device tag name.

new_name: String
The new device tag name.

Return Value

None.

Sample Usage

Refer to the **examples** directory in the Python SDK for an example.

is_ecm

Returns true if the ExtraHop system is an ECM.

Syntax:

```
is_ecm()
```

Parameters

None.

Return Value

True or false.

new_device_group

Create a new device group (version 3.8+).

Syntax:

```
new_device_group("name", "comment", members)
```

Parameters

comment: String
The comment.

members: List
A list of integers representing device group object IDs.

name: String
The device name.

Return Value

A new device group.

Sample Usage

```
>>> r = c.new_device_group("my group name", "my group comment", [0, 10, 20])
>>> import json
>>> print json.dumps(r, indent=2)
{
  "mod_time": 1361987813.0,
  "comment": "my group comment",
  "name": "my group name",
  "deleted": false,
  "oid": 11,
  "dynamic": false,
  "value": null,
  "field": null,
  "internal": false,
  "cluster_id": "no_cluster",
  "members": [
    0,
    10,
    20
  ],
  "member_set": [
    {
      "comment": null,
      "mod_time": 1361400676.0,
      "macaddr": "BC:AE:C5:50:F9:FF",
      "vendor": "ASUS",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": null,
      "ipaddr6": null,
      "parent_oid": null,
      "vlanid": 1010,
```

```
"ip_discover": false,
"default_name": "ASUS 50F9FF",
"capture_oid": 0,
"id": 1,
"device_id": "bcaec550f9ff1010",
"dhcp_name": "",
"user_mod_time": 1358306852.0,
"analysis_level": 0,
"name": "ASUS 50F9FF",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1354589328113.0,
"oid": 0,
"nb_name": ""
},
{
  "comment": null,
  "mod_time": 1361400676.0,
  "macaddr": "00:18:71:87:36:09",
  "vendor": "HP",
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": null,
  "ipaddr6": null,
  "parent_oid": null,
  "vlanid": 7,
  "ip_discover": false,
  "default_name": "HP 873609",
  "capture_oid": 0,
  "id": 11,
  "device_id": "0018718736090007",
  "dhcp_name": "",
  "user_mod_time": 1358306852.0,
  "analysis_level": 0,
  "name": "HP 873609",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
  "tag_set": [],
  "tags": [],
  "discover_time": 1354589328113.0,
  "oid": 10,
  "nb_name": ""
},
{
  "comment": null,
  "mod_time": 1361400676.0,
  "macaddr": "00:0D:DA:0C:BE:32",
  "vendor": "ALLIED TELESIS KK",
  "custom_type": "",
```

```
"devclass": "gateway",
"ipaddr4": null,
"ipaddr6": null,
"parent_oid": null,
"vlanid": 2,
"ip_discover": false,
"default_name": "ALLIED TELESIS KK 0CBE32",
"capture_oid": 0,
"id": 21,
"device_id": "000dda0cbe320002",
"dhcp_name": "",
"user_mod_time": 1358306852.0,
"analysis_level": 1,
"name": "ALLIED TELESIS KK 0CBE32",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1354589329113.0,
"oid": 20,
"nb_name": ""
}
]
}
```

See Also:

"device-groups.py" on page 287

new_dynamic_device_group

Create a new dynamic device group.

Syntax:

```
new_dynamic_device_group("name", "comment", "field", "value")
```

Parameters

comment: String

A comment about the device group.

field: String

Key that defines the criteria. Supported keys are:

- ip address
- mac address
- vendor
- vlan
- type
- tag

name: String

The name of the device group.

value: String

Value.

- If you want to use regex, enter a string. For instance, for regex on /vmware/, enter "vmware".
- for an explicit string, escape with a "\Q" in the front and "\E" in the back. For instance, for an explicit string "vmware", enter "\Qvmware\E".
- For "type", use activity types such as "http_server", "db_client", "router", etc.

Return Value

A new group object.

Sample Usage

Refer to the **examples** directory in the Python SDK for an example.

See Also:

"device-groups.py" on page 287

register_license

Automatically apply the ExtraHop license and restart the capture. (Version 3.10.18933+)

Syntax:

```
register_license("product_key")
```

Parameters

product_key: Array

The ExtraHop product key. If the product key is not specified, the ExtraHop system contacts the license server to access the product key.

Return Value

None.

Sample Usage

```
register_license('EXTR-EXTR-ABCD-EFGH')
```

remove_device_group_members

Remove members from a device group.

Syntax:

```
remove_device_group_members(group_oid, members)
```

Parameters

group_oid: Integer

The object ID of the device group to be removed.

members: List

A list of integers representing device group object IDs.

Return Value

An updated group.

Sample Usage

```
>>> group.oid
12
>>> group.members
[8, 10, 11, 99]
>>> group = c.remove_device_group_members(group.oid, [8, 11])
>>> group.oid
12
>>> group.members
[10, 99]
```

remove_flex_grid_rows

Delete rows from specific flex grids (e.g., remove objects from a flex grid).

Syntax:

```
remove_flex_grid_rows(flexgrid_oid, object_ids, "object_type")
```

Parameters

flexgrid_oid: Integer

The object ID of the flex grid from which rows will be removed.

object_ids: List

A list of object IDs that will be removed from the flex grid.

object_type: String

The type of object to be removed (i.e., "device", "device_group", or "application").

Return Value

None.

remove_flex_grids

Delete specific flex grids.

Syntax:

```
remove_flex_grids(flexgrid_oids)
```

Parameters

flexgrid_oids: List

A list of the object IDS of the flex grids to be deleted.

Return Value

None.

remove_from_whitelist

Remove devices from the ExtraHop whitelist.

Syntax:

```
remove_from_whitelist(device_oids)
```

Parameters

device_oids: List

A list of the object IDs of one or more devices to remove from the ExtraHop whitelist.

Return Value

The modified devices.

Sample Usage

```
device_group_oid_to_whitelist = 4
device_oids = set() # Unique set of oids

members = c.get_device_group_devices(-18000000, 0, device_group_oid_to_whitelist,
{})

for device in members.devices:
    device_oids.add(device.oid)

c.remove_from_whitelist(list(device_oids))
```

See Also:

whitelist.py on page 305

remove_trigger

Remove assignments from a specific trigger.

Syntax:

```
remove_trigger(trigger_oid, "object_type", object_ids)
```

Parameters

object_ids: Integer

The object IDs from which the trigger assignment will be removed.

object_type: String

The type of object from which the trigger assignment will be removed (i.e., "device" and "device_group").

trigger_oid: Integer

The object ID of the trigger from which assignments will be removed.

Return Value

A trigger object.

restart_service

Restart system services. (Version 3.10.18870+)

Syntax:

```
restart_service("service_name")
```

Parameters

service_name: String
The name of the service to restart.

Return Value

None.

Sample Usages

```
restart_service("exportal")
```

```
restart_service("webserver")
```

```
restart_service("system")
```

save_flex_grid

Create or modify a flex grid.

Syntax:

```
save_flex_grid(flexgrid_oid, "name", "comment", "object_type", columns)
```

Parameters

columns:

Dictionary with the following parameters:

name: String

The name of the column.

stat_class: String

The class of metric. Supported values are:

- application
- device
- device_group

stat_type: String

The type of metric (e.g., "net", "tcp", etc.).

field_name: String

The metric name (e.g., "bytes_in").

use_binary_units: Boolean

True if 1K=1024, otherwise false.

comment: String

A comment about the flex grid.

flexgrid_oid: Integer

The object ID of the flex grid to save. (The oid is "None" if creating a new flex grid.)

name: String

The name of the flex grid.

object_type: String

The type of assigned object (i.e., "device", "device_group", or "application").

Return Value

None.

Sample Usage

```
c.save_flex_grid(None, "samplegrid", "test", "device",
  [{'name': "Bytes In",
    'stat_class': "device",
    'stat_type': "net",
    'field_name': "bytes_in",
    'use_binary_units': False}]
)
```

save_ssl_decryption_key

Upload an SSL certificate.

Syntax:

```
save_ssl_decryption_key("desc", cert, "key", enable)
```

Parameters

cert: String
The PEM-encoded certificate.

desc: String
The certificate name.

enable: Boolean
A flag indicating whether the certification should be enabled or disabled.

key: String
The decryption key.

Return Value

None.

Sample Usage

Refer to the examples directory in the Python SDK for an example.

set_running_config

Set and optionally save the running config. To view and modify the running config, you must log in with the **setup** user account.

Syntax:

```
set_running_config(new_config, save)
```

Parameters

new_config: Python dictionary

The new running config to replace the existing running config.

save: Boolean

Saves the running config when set to `True`.

Return Value

None.

Sample Usage

The following example shows how to enable DHCP on the first network interface and then save the running config.

```
>>> running_config = c.get_running_config()
>>> running_config['net']['interfaces'][0] = {'dhcp': True}
>>> c.set_running_config(running_config, True)
```

See Also:

set-host-name.py on page 301

set_trigger

Create or modify a trigger. If the "trigger_id" parameter is set to "None", a new trigger is created. Otherwise, the trigger with the specified ID is modified.

Syntax:

```
set_trigger(name, script, event, [trigger_oid], [comment], [priority], [debug],
[novalidate], [disabled], [hints], ["author"], [apply_all])
```

Parameters

apply_all: Boolean

(Optional) Set to true to make the trigger a global trigger, applied to all objects by default. The default value is "False".

author: String

(Optional) The trigger author's name. The default value is "None".

comment: String

(Optional) A description of the trigger. The default value is "None".

debug: Boolean

(Optional) Represents whether debugging should be turned on. The default value is "False".

disabled: Boolean

(Optional) Set to true to disable the trigger. The default value is "False".

event: String

Event on which the trigger should fire (e.g., HTTP_RESPONSE). Refer to the *ExtraHop Application Inspection Triggers API* for a full list.

hints: JSON-formatted string

(Optional) The default value is "None". Additional guidance for specialized triggers:

- For an HTTP_RESPONSE event utilizing HTTP.payload, specify the maximum number of bytes using {"payloadBytes": 1024}. Parameter must be a non-negative integer.
- For protocol events using packet capture, specify whether to buffer packets by using "packetCapture": True. Parameter must be a boolean value.
- A string must be written as a parameter. Example: setTrigger(... json.dumps({"packetCapture": True, "payloadBytes": 1024}) ...)

name: String

Name of the trigger.

novalidate: Boolean

(Optional) The default value is "False".

priority: Integer

(Optional) Number 0-9 representing the priority, with 0 being the highest priority. The default value is "0".

script: String

The trigger script.

trigger_oid: Integer.

(Optional) The default value is "None". If this parameter is set to "None", a new trigger is created. Otherwise, the trigger with the specified object ID is modified.

Return Value

A trigger object.

update_custom_device

Update an existing custom device.

```
update_custom_device(device_oid, {updates})
```

Parameter

device_oid: Integer

The unique object ID of the device.

updates: Dictionary with the attribute names listed as keys.

Device object to replace the updated device. Include only the oid attribute (key) and the attributes to be changed. The object can have the following attributes:

author: String

(Optional) The custom device creator's name. The default value is "None".

comment: String

(Optional) A description of the custom device. The default value is "None".

device_id: Integer

The unique identifier of the device.

disabled: Boolean

False if the device is enabled.

name: String

The name of the custom device.

Return Value

None.

update_device

Update an existing device.

Syntax:

```
update_device({"oid": oid, ["custom_name": custom_name], ["comment": comment],
              ["custom_type": custom_type], ["vendor": vendor], ["default_name": default_name]})
```

Parameter

Device object to replace the updated device. Include only the oid attribute (key) and the attributes to be changed. The device object can have the following attributes:

- comment:** String
A new comment for the device.
- custom_name:** String
The new custom name for the device, if no such other name exists (must be non-null, e.g., some string or an empty string to unset the value).
- custom_type:** String
A new custom_type for the device.
- default_name:** String
A new default name for the device.
- oid:** Integer
The device object ID.
- vendor:** String
A new vendor for the device.

Return Value

None.

Sample Usage

```
r = c.update_device({"oid": 14, "custom_name": "foo", "comment": "my device"})
```

Sample Usage 2

```
>>> class MyObject: pass
...
>>> obj = MyObject()
>>> obj.oid = 14
>>> obj.custom_name = "bar"
>>> r = c.update_device(obj)
```

See Also:

"device-update-name.py" on page 292

update_device_group

Update existing device groups.

Syntax:

```
update_device_group(group_oid, "name", "comment", members)
```

Parameters

comment: String

A comment about the device group.

group_oid: Integer

The object ID of the device group to be updated.

members: List

A list of integers representing device group object IDs.

name: String

The name of the device group.

Return Value

An updated device group.

Sample Usage

```
>>> r = c.new_device_group("my group name", "my group description", [0, 10])
>>> r.oid
12
>>> r = c.update_device_group(12, "new name", "my group description", [8, 99])
>>> import json
>>> print json.dumps(r, indent=2)
{
  "mod_time": 1361988089.0,
  "comment": "my group comment",
  "name": "new name",
  "deleted": false,
  "oid": 12,
  "dynamic": false,
  "value": null,
  "field": null,
  "internal": false,
  "cluster_id": "no_cluster",
  "members": [
    8,
    99
  ],
  "member_set": [
    {
      "comment": null,
      "mod_time": 1361400676.0,
      "macaddr": "00:1E:7A:B1:B5:00",
      "vendor": "Cisco",
      "custom_type": "",

```

```
"devclass": "gateway",
"ipaddr4": null,
"ipaddr6": null,
"parent_oid": null,
"vlanid": 7,
"ip_discover": false,
"default_name": "Cisco B1B500",
"capture_oid": 0,
"id": 9,
"device_id": "001e7ab1b5000007",
"dhcp_name": "",
"user_mod_time": 1358306852.0,
"analysis_level": 1,
"name": "Cisco B1B500",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1354589328113.0,
"oid": 8,
"nb_name": ""
},
{
  "comment": null,
  "mod_time": 1361400677.0,
  "macaddr": "70:56:81:90:C5:B1",
  "vendor": null,
  "custom_type": "",
  "devclass": "node",
  "ipaddr4": null,
  "ipaddr6": null,
  "parent_oid": null,
  "vlanid": 1010,
  "ip_discover": false,
  "default_name": "Device 70568190c5b11010",
  "capture_oid": 0,
  "id": 100,
  "device_id": "70568190c5b11010",
  "dhcp_name": "",
  "user_mod_time": 1358306852.0,
  "analysis_level": 0,
  "name": "Device 70568190c5b11010",
  "custom_name": null,
  "cdp_name": "",
  "dns_name": "",
  "tag_set": [],
  "tags": [],
  "discover_time": 1354589336113.0,
  "oid": 99,
  "nb_name": ""
}
]
```

```
}
```

update_device_group_members

Update members of a device group.

Syntax:

```
update_device_group_members(group_oid, members)
```

Parameters

group_oid: Integer

The object ID of the device group to be updated.

members: List

A list of integers representing object IDs to replace in an existing device list.

Return Value

An updated group.

Sample Usage

```
>>> group.oid
12
>>> group.members
[10, 99]
>>> group = c.update_device_group_members(group.oid, [40, 50])
>>> group.oid
12
>>> group.members
[40, 50]
```

update_dynamic_device_group

Update a dynamic device group.

Syntax:

```
update_dynamic_device_group(group_oid, "name", "comment", "field", "value")
```

Parameters

comment: String

A comment about the device group.

field: String

A key that defines the criteria. Supported keys are:

- ip address
- mac address
- vendor
- vlan
- type
- tag

group_oid: Integer

The object ID of the device group to be updated.

name: String

The name of the device group.

value: String

Value.

- If you want to use regex, enter a string. For instance, for regex on /vmware/, enter "vmware".
- For an explicit string, escape with "\Q" before and "\E" after your string. For instance, for an explicit string "vmware", enter "\Qvmware\E".
- For 'type', use activity types such as "http_server", "db_client", "router", etc.

Return Value

An updated group object.

Sample Usage

Refer to the **examples** directory in the Python SDK for an example.

See Also:

"device-groups.py" on page 287

update_network

Make updates to a specific network.

Syntax:

```
update_network(network_oid, updates)
```

Parameters

network_oid: Integer

The unique object ID of the network.

updates: List

A dictionary including the following optional fields:

name: String

An optional author name for the network update.

comment: String

An optional comment about the network update.

Return Value

None.

update_user

Make updates to a specific user.

Syntax:

```
update_user(username, updates)
```

Parameters

updates: Dictionary

A dictionary with the following options:

enabled: Boolean

A flag indicating whether the field should be enabled or disabled.

name: String

The name of the field.

username: String

The name of the user.

Return Value

A list of user updates.

Searchers

Devices

search_devices_by_activity(*start, until, "value"*, [{"offset": *offset*}, {"limit": *limit*}, {"capture_oid": *capture_oid*}, {"devtype": "devtype"}])
Search devices with a certain type of activity.

search_devices_by_any(*start, until, "value"*, [{"offset": *offset*}, {"limit": *limit*}, {"capture_oid": *capture_oid*}, {"devtype": "devtype"}])
Search devices by any attribute (name, IP address, MAC address, vendor, tag, vlan).

search_devices_by_ip(*start, until, value*, [{"offset": *offset*}, {"limit": *limit*}, {"capture_oid": *capture_oid*}, {"devtype": "devtype"}])
Search devices by IP address.

search_devices_by_mac(*start, until, "value"*, [{"offset": *offset*}, {"limit": *limit*}, {"capture_oid": *capture_oid*}, {"devtype": "devtype"}])
Search devices by MAC address.

search_devices_by_name(*start, until, "value"*, [{"offset": *offset*}, {"limit": *limit*}, {"capture_oid": *capture_oid*}, {"devtype": "devtype"}])
Search devices by name.

search_devices_by_tag(*start, until, "value"*, [{"offset": *offset*}, {"limit": *limit*}, {"capture_oid": *capture_oid*}, {"devtype": "devtype"}])
Search devices by device tag.

search_devices_by_type(*start, until, "value"*, [{"offset": *offset*}, {"limit": *limit*}, {"capture_oid": *capture_oid*}, {"devtype": "devtype"}])
Search devices with a certain device type (server, router, etc.)

search_devices_by_vendor(*start, until, "value"*, [{"offset": *offset*}, {"limit": *limit*}, {"capture_oid": *capture_oid*}, {"devtype": "devtype"}])
Search devices by vendor.

search_devices_by_vlan(*start, until, "value"*, [{"offset": *offset*}, {"limit": *limit*}, {"capture_oid": *capture_oid*}, {"devtype": "devtype"}])
Search devices by VLAN tag.

search_devices_by_activity

Search for devices with a certain type of activity.

Syntax:

```
search_devices_by_activity(start, until, "value", [{"offset": offset}, {"limit": limit}, {"capture_oid": capture_oid}, {"devtype": "devtype"}])
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

value: String

The value string interpreted as a regular expression. The value is any full non-detail metric name, such as `extrahop.device.http_server` and `extrahop.device.db_server`.

Options

A dictionary of zero or more option name and value pairs.

devtype: String

The device type of devices to return. Possible values are:

- db_server
- gateway
- server
- firewall
- load_balancer
- file_server

limit: Integer

The number of devices to return to support the pagination of results. For example, if you have 100 devices and you ask for `offset=0` and `limit=10`, the first 10 devices will be returned. If you ask for `offset=50`, `limit=50`, the last 50 devices will be returned.

offset: Integer

The offset of the devices to return to support pagination. See the description under **limit** above.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.search_devices_by_activity(-1800000, 0, "activity", {"capture_oid": my_
capture_oid, "offset": 0, "limit": 50})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 702,
  "devices": [
    {
      "comment": null,
      "mod_time": 1361900433.0,
      "macaddr": "78:2B:CB:32:C9:81",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": null,
      "ipaddr6": null,
      "parent_oid": null,
      "vlanid": 1010,
      "ip_discover": false,
      "default_name": "Dell 32C981",
      "capture_oid": 0,
      "id": 2345,
      "device_id": "782bcb32c9811010",
      "dhcp_name": "",
      "user_mod_time": 1361900433.0,
      "analysis_level": 0,
      "name": "Dell 32C981",
      "custom_name": null,
      "cdp_name": "",
      "dns_name": "",
      "tag_set": [],
      "tags": [],
      "discover_time": 1361900405041.0,
      "activity": [
        "extrahop.device.net"
      ],
      "oid": 2344,
      "nb_name": ""
    },
    {
      "comment": null,
      "mod_time": 1361900163.0,
      "macaddr": "78:2B:CB:32:C9:89",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": "10.10.251.227",
      "ipaddr6": null,
      "parent_oid": 2342,
      "vlanid": 1010,
      "ip_discover": true,

```

```
"default_name": "Dell 10.10.251.227",
"capture_oid": 0,
"id": 2344,
"device_id": "fff4e3fb0a0a1010",
"dhcp_name": "",
"user_mod_time": 1361900163.0,
"analysis_level": 0,
"name": "Dell 10.10.251.227",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1361900141041.0,
"activity": [
  "extrahop.device.app",
  "extrahop.device.dns_client",
  "extrahop.device.net"
],
"oid": 2343,
"nb_name": ""
}
]
```

See Also:

"count.py" on page 282

"device-search-by-activity.py" on page 289

search_devices_by_any

Search devices by any attribute (name, IP address, MAC address, vendor, type, tag, VLAN).

Syntax:

```
search_devices_by_any(start, until, "value", [{"offset": offset},  
["limit": limit], ["capture_oid": capture:oid], ["devtype": "devtype"]])
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

value: String

The value string interpreted as a regular expression. The value is any IP address, MAC address, etc.

Options

A dictionary of zero or more option name and value pairs.

capture_oid: Integer

The capture object ID.

devtype: String

The device type of devices to return. Possible values are:

- db_server
- gateway
- server
- firewall
- load_balancer
- file_server

limit: Integer

The number of devices to return to support the pagination of results. For example, if you have 100 devices and you ask for offset=0 and limit=10, the first 10 devices will be returned. If you ask for offset=50, limit=50, the last 50 devices will be returned.

offset: Integer

The offset of the devices to return to support pagination. See the description under **limit** above.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.search_devices_by_any(-1800000, 0, "apple", {"offset": 10, "limit": 2,
"capture_oid": my_capture_oid})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 702,
  "devices": [
    {
      "comment": null,
      "mod_time": 1361900433.0,
      "macaddr": "78:2B:CB:32:C9:81",
      "vendor": "Apple",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": null,
      "ipaddr6": null,
      "parent_oid": null,
      "vlanid": 1010,
      "ip_discover": false,
      "default_name": "Dell 32C981",
      "capture_oid": 0,
      "id": 2345,
      "device_id": "782bcb32c9811010",
      "dhcp_name": "",
      "user_mod_time": 1361900433.0,
      "analysis_level": 0,
      "name": "Dell 32C981",
      "custom_name": null,
      "cdp_name": "",
      "dns_name": "",
      "tag_set": [],
      "tags": [],
      "discover_time": 1361900405041.0,
      "activity": [
        "extrahop.device.net"
      ],
      "oid": 2344,
      "nb_name": ""
    },
    {
      "comment": null,
      "mod_time": 1361900163.0,
      "macaddr": "78:2B:CB:32:C9:89",
      "vendor": "Apple",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": "10.10.251.227",
      "ipaddr6": null,
      "parent_oid": 2342,
      "vlanid": 1010,
      "ip_discover": true,

```

```
"default_name": "Dell 10.10.251.227",
"capture_oid": 0,
"id": 2344,
"device_id": "fff4e3fb0a0a1010",
"dhcp_name": "",
"user_mod_time": 1361900163.0,
"analysis_level": 0,
"name": "Dell 10.10.251.227",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1361900141041.0,
"activity": [
    "extrahop.device.app",
    "extrahop.device.dns_client",
    "extrahop.device.net"
],
"oid": 2343,
"nb_name": ""
}
]
}
```

search_devices_by_ip

Search devices by IP address.

Syntax:

```
search_devices_by_ip(start, until, value, [{"offset": offset}, {"limit": limit}, {"capture_oid": capture_oid}, {"devtype": "devtype"}])
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

value: String

The value string interpreted as a regular expression. The value is the IP address.

Options

A dictionary of zero or more option name and value pairs.

capture_oid: Integer

The capture object ID.

devtype: String

The device type of devices to return. Possible values are:

- db_server
- gateway
- server
- firewall
- load_balancer
- file_server

limit: Integer

The number of devices to return to support the pagination of results. For example, if you have 100 devices and you ask for offset=0 and limit=10, the first 10 devices will be returned. If you ask for offset=50, limit=50, the last 50 devices will be returned.

offset: Integer

The offset of the devices to return to support pagination. See the description under **limit** above.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.search_devices_by_ip(-1800000, 0, "10.10.1.135", {"capture_oid": my_
capture_oid})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 702,
  "devices": [
    {
      "comment": null,
      "mod_time": 1361900433.0,
      "macaddr": "78:2B:CB:32:C9:81",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": 10.10.1.135,
      "ipaddr6": null,
      "parent_oid": null,
      "vlanid": 1010,
      "ip_discover": false,
      "default_name": "Dell 32C981",
      "capture_oid": 0,
      "id": 2345,
      "device_id": "782bcb32c9811010",
      "dhcp_name": "",
      "user_mod_time": 1361900433.0,
      "analysis_level": 0,
      "name": "Dell 32C981",
      "custom_name": null,
      "cdp_name": "",
      "dns_name": "",
      "tag_set": [],
      "tags": [],
      "discover_time": 1361900405041.0,
      "activity": [
        "extrahop.device.net"
      ],
      "oid": 2344,
      "nb_name": ""
    }
  ]
}
```

See Also:

"device-tags.py" on page 290

"device-update-name.py" on page 292

search_devices_by_mac

Search devices by MAC address.

Syntax:

```
search_devices_by_mac(start, until, "value", [{"offset": offset},
["limit": limit], ["capture_oid": capture_oid], ["devtype": "devtype"]}]
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

value: String

The value string interpreted as a regular expression. The value is the MAC address.

Options

A dictionary of zero or more option name and value pairs.

capture_oid: Integer

The capture object ID.

devtype: String

The device type of devices to return. Possible values are:

- db_server
- gateway
- server
- firewall
- load_balancer
- file_server

limit: Integer

The number of devices to return to support the pagination of results. For example, if you have 100 devices and you ask for offset=0 and limit=10, the first 10 devices will be returned. If you ask for offset=50, limit=50, the last 50 devices will be returned.

offset: Integer

The offset of the devices to return to support pagination. See the description under **limit** above.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.search_devices_by_mac(-1800000, 0, "78:2B:CB:32:C9:81", {"offset": 10,
"limit": 2, "capture_oid": my_capture_oid})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 702,
  "devices": [
    {
      "comment": null,
      "mod_time": 1361900433.0,
      "macaddr": "78:2B:CB:32:C9:81",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": null,
      "ipaddr6": null,
      "parent_oid": null,
      "vlanid": 1010,
      "ip_discover": false,
      "default_name": "Dell 32C981",
      "capture_oid": 0,
      "id": 2345,
      "device_id": "782bcb32c9811010",
      "dhcp_name": "",
      "user_mod_time": 1361900433.0,
      "analysis_level": 0,
      "name": "Dell 32C981",
      "custom_name": null,
      "cdp_name": "",
      "dns_name": "",
      "tag_set": [],
      "tags": [],
      "discover_time": 1361900405041.0,
      "activity": [
        "extrahop.device.net"
      ],
      "oid": 2344,
      "nb_name": ""
    },
  ]
}
```

search_devices_by_name

Search devices by name.

Syntax:

```
search_devices_by_name(start, until, "value", [{"offset": offset}, {"limit": limit}, {"capture_oid": capture_oid}, {"devtype": "devtype"}]}
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

value: String

The value string interpreted as a regular expression. The value is the device name.

Options

A dictionary of zero or more option name and value pairs.

capture_oid: Integer

The capture object ID.

devtype: String

The device type of devices to return. Possible values are:

- db_server
- gateway
- server
- firewall
- load_balancer
- file_server

limit: Integer

The number of devices to return to support the pagination of results. For example, if you have 100 devices and you ask for offset=0 and limit=10, the first 10 devices will be returned. If you ask for offset=50, limit=50, the last 50 devices will be returned.

offset: Integer

The offset of the devices to return to support pagination. See the description under **limit** above.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.search_devices_by_name(-1800000, 0, "device_name", {"offset": 10,
"limit": 2, "capture_oid": my_capture_oid})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 702,
  "devices": [
    {
      "comment": null,
      "mod_time": 1361900433.0,
      "macaddr": "78:2B:CB:32:C9:81",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": null,
      "ipaddr6": null,
      "parent_oid": null,
      "vlanid": 1010,
      "ip_discover": false,
      "default_name": "Dell 32C981",
      "capture_oid": 0,
      "id": 2345,
      "device_id": "782bcb32c9811010",
      "dhcp_name": "",
      "user_mod_time": 1361900433.0,
      "analysis_level": 0,
      "name": "Dell 32C981",
      "custom_name": null,
      "cdp_name": "",
      "dns_name": "",
      "tag_set": [],
      "tags": [],
      "discover_time": 1361900405041.0,
      "activity": [
        "extrahop.device.net"
      ],
      "oid": 2344,
      "nb_name": ""
    },
    {
      "comment": null,
      "mod_time": 1361900163.0,
      "macaddr": "78:2B:CB:32:C9:89",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": "10.10.251.227",
      "ipaddr6": null,
      "parent_oid": 2342,
      "vlanid": 1010,
      "ip_discover": true,

```

```
"default_name": "Dell 10.10.251.227",
"capture_oid": 0,
"id": 2344,
"device_id": "fff4e3fb0a0a1010",
"dhcp_name": "",
"user_mod_time": 1361900163.0,
"analysis_level": 0,
"name": "Dell 10.10.251.227",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1361900141041.0,
"activity": [
    "extrahop.device.app",
    "extrahop.device.dns_client",
    "extrahop.device.net"
],
"oid": 2343,
"nb_name": ""
}
]
}
```

See also:

"device-search-by-name.py" on page 289

whitelist.py on page 305

search_devices_by_tag

Search devices by device tag.

Syntax:

```
search_devices_by_tag(start, until, "value", [{"offset": offset}, {"limit": limit}, {"capture_oid": capture_oid}, {"devtype": "devtype"}]
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

value: String

The value string interpreted as a regular expression. The value is the device tag.

Options

A dictionary of zero or more option name and value pairs.

capture_oid: Integer

The capture object ID.

devtype: String

The device type of devices to return. Possible values are:

- db_server
- gateway
- server
- firewall
- load_balancer
- file_server

limit: Integer

The number of devices to return to support the pagination of results. For example, if you have 100 devices and you ask for offset=0 and limit=10, the first 10 devices will be returned. If you ask for offset=50, limit=50, the last 50 devices will be returned.

offset: Integer

The offset of the devices to return to support pagination. See the description under **limit** above.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.search_devices_by_tag(-1800000, 0, "my_tag", {"offset": 10, "limit": 2,
"capture_oid": my_capture_oid})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 702,
  "devices": [
    {
      "comment": null,
      "mod_time": 1361900433.0,
      "macaddr": "78:2B:CB:32:C9:81",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": null,
      "ipaddr6": null,
      "parent_oid": null,
      "vlanid": 1010,
      "ip_discover": false,
      "default_name": "Dell 32C981",
      "capture_oid": 0,
      "id": 2345,
      "device_id": "782bcb32c9811010",
      "dhcp_name": "",
      "user_mod_time": 1361900433.0,
      "analysis_level": 0,
      "name": "Dell 32C981",
      "custom_name": null,
      "cdp_name": "",
      "dns_name": "",
      "tag_set": [],
      "tags": ["my_tag"],
      "discover_time": 1361900405041.0,
      "activity": [
        "extrahop.device.net"
      ],
      "oid": 2344,
      "nb_name": ""
    },
    {
      "comment": null,
      "mod_time": 1361900163.0,
      "macaddr": "78:2B:CB:32:C9:89",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": "10.10.251.227",
      "ipaddr6": null,
      "parent_oid": 2342,
      "vlanid": 1010,
      "ip_discover": true,

```



```
"default_name": "Dell 10.10.251.227",
"capture_oid": 0,
"id": 2344,
"device_id": "fff4e3fb0a0a1010",
"dhcp_name": "",
"user_mod_time": 1361900163.0,
"analysis_level": 0,
"name": "Dell 10.10.251.227",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": ["my_tag"],
"discover_time": 1361900141041.0,
"activity": [
    "extrahop.device.app",
    "extrahop.device.dns_client",
    "extrahop.device.net"
],
"oid": 2343,
"nb_name": ""
}
]
}
```

search_devices_by_type

Search devices with a certain device type (server, router, etc.).

Syntax:

```
search_devices_by_type(start, until, "value", [{"offset": offset}, {"limit": limit}, {"capture_oid": capture_oid}, {"devtype": "devtype"}]
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

value: String

The value string interpreted as a regular expression. The value is the device type, such as gateway, server, etc.

Options

A dictionary of zero or more option name and value pairs.

capture_oid: Integer

The capture object ID.

devtype: String

The device type of devices to return. Possible values are:

- db_server
- gateway
- server
- firewall
- load_balancer
- file_server

limit: Integer

The number of devices to return to support the pagination of results. For example, if you have 100 devices and you ask for offset=0 and limit=10, the first 10 devices will be returned. If you ask for offset=50, limit=50, the last 50 devices will be returned.

offset: Integer

The offset of the devices to return to support pagination. See the description under **limit** above.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.search_devices_by_type(-1800000, 0, "dns_client", {"offset": 10, "limit":
1, "capture_oid": my_capture_oid})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 1,
  "devices": [
    {
      "comment": null,
      "mod_time": 1361900163.0,
      "macaddr": "78:2B:CB:32:C9:89",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": "10.10.251.227",
      "ipaddr6": null,
      "parent_oid": 2342,
      "vlanid": 1010,
      "ip_discover": true,
      "default_name": "Dell 10.10.251.227",
      "capture_oid": 0,
      "id": 2344,
      "device_id": "fff4e3fb0a0a1010",
      "dhcp_name": "",
      "user_mod_time": 1361900163.0,
      "analysis_level": 0,
      "name": "Dell 10.10.251.227",
      "custom_name": null,
      "cdp_name": "",
      "dns_name": "",
      "tag_set": [],
      "tags": [],
      "discover_time": 1361900141041.0,
      "activity": [
        "extrahop.device.app",
        "extrahop.device.dns_client",
        "extrahop.device.net"
      ],
      "oid": 2343,
      "nb_name": ""
    }
  ]
}
```

search_devices_by_vendor

Search devices by vendor.

Syntax:

```
search_devices_by_vendor(start, until, "value", [{"offset": offset}, {"limit": limit}, {"capture_oid": capture_oid}, {"devtype": "devtype"}]
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

value: String

The value string interpreted as a regular expression. The value is a vendor string.

Options

A dictionary of zero or more option name and value pairs.

capture_oid: Integer

The capture object ID.

devtype: String

The device type of devices to return. Possible values are:

- db_server
- gateway
- server
- firewall
- load_balancer
- file_server

limit: Integer

The number of devices to return to support the pagination of results. For example, if you have 100 devices and you ask for offset=0 and limit=10, the first 10 devices will be returned. If you ask for offset=50, limit=50, the last 50 devices will be returned.

offset: Integer

The offset of the devices to return to support pagination. See the description under **limit** above.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.search_devices_by_vendor(-1800000, 0, "apple", {"offset": 10, "limit": 2,
"capture_oid": my_capture_oid})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 702,
  "devices": [
    {
      "comment": null,
      "mod_time": 1361900433.0,
      "macaddr": "78:2B:CB:32:C9:81",
      "vendor": "Apple",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": null,
      "ipaddr6": null,
      "parent_oid": null,
      "vlanid": 1010,
      "ip_discover": false,
      "default_name": "Dell 32C981",
      "capture_oid": 0,
      "id": 2345,
      "device_id": "782bcb32c9811010",
      "dhcp_name": "",
      "user_mod_time": 1361900433.0,
      "analysis_level": 0,
      "name": "Dell 32C981",
      "custom_name": null,
      "cdp_name": "",
      "dns_name": "",
      "tag_set": [],
      "tags": [],
      "discover_time": 1361900405041.0,
      "activity": [
        "extrahop.device.net"
      ],
      "oid": 2344,
      "nb_name": ""
    },
    {
      "comment": null,
      "mod_time": 1361900163.0,
      "macaddr": "78:2B:CB:32:C9:89",
      "vendor": "Apple",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": "10.10.251.227",
      "ipaddr6": null,
      "parent_oid": 2342,
      "vlanid": 1010,
      "ip_discover": true,

```

```
"default_name": "Dell 10.10.251.227",
"capture_oid": 0,
"id": 2344,
"device_id": "fff4e3fb0a0a1010",
"dhcp_name": "",
"user_mod_time": 1361900163.0,
"analysis_level": 0,
"name": "Dell 10.10.251.227",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1361900141041.0,
"activity": [
    "extrahop.device.app",
    "extrahop.device.dns_client",
    "extrahop.device.net"
],
"oid": 2343,
"nb_name": ""
}
]
}
```

search_devices_by_vlan

Search devices by VLAN.

Syntax:

```
search_devices_by_vlan(start, until, "value", [{"offset": offset}, {"limit": limit}, {"capture_oid": capture_oid}, {"devtype": "devtype"}]
```

Parameters

start: Number

The start time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is negative, it indicates relative time from the current capture time (e.g., -180000 is 3 minutes from the current capture time). If the value is 0, it indicates the current capture time.

until: Number

The end time, expressed in milliseconds. If the value is positive, it indicates absolute time since the epoch (January 1, 1970). If the value is 0, it indicates the current capture time. Relative time is not supported.

value: String

The value string interpreted as a regular expression. The value is a vlan.

Options

A dictionary of zero or more option name and value pairs.

capture_oid: Integer

The capture object ID.

devtype: String

The device type of devices to return. Possible values are:

- db_server
- gateway
- server
- firewall
- load_balancer
- file_server

limit: Integer

The number of devices to return to support the pagination of results. For example, if you have 100 devices and you ask for offset=0 and limit=10, the first 10 devices will be returned. If you ask for offset=50, limit=50, the last 50 devices will be returned.

offset: Integer

The offset of the devices to return to support pagination. See the description under **limit** above.

Return Value

An object with the following properties:

devices: List

A list of device objects.

total: Integer

The total number of devices.

Sample Usage

```
>>> r = c.search_devices_by_vlan(-1800000, 0, "1010", {"offset": 10, "limit": 2,
"capture_oid": my_capture_oid})
>>> import json
>>> print json.dumps(r, indent=2)
{
  "total": 702,
  "devices": [
    {
      "comment": null,
      "mod_time": 1361900433.0,
      "macaddr": "78:2B:CB:32:C9:81",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": null,
      "ipaddr6": null,
      "parent_oid": null,
      "vlanid": 1010,
      "ip_discover": false,
      "default_name": "Dell 32C981",
      "capture_oid": 0,
      "id": 2345,
      "device_id": "782bcb32c9811010",
      "dhcp_name": "",
      "user_mod_time": 1361900433.0,
      "analysis_level": 0,
      "name": "Dell 32C981",
      "custom_name": null,
      "cdp_name": "",
      "dns_name": "",
      "tag_set": [],
      "tags": [],
      "discover_time": 1361900405041.0,
      "activity": [
        "extrahop.device.net"
      ],
      "oid": 2344,
      "nb_name": ""
    },
    {
      "comment": null,
      "mod_time": 1361900163.0,
      "macaddr": "78:2B:CB:32:C9:89",
      "vendor": "Dell",
      "custom_type": "",
      "devclass": "node",
      "ipaddr4": "10.10.251.227",
      "ipaddr6": null,
      "parent_oid": 2342,
      "vlanid": 1010,
      "ip_discover": true,

```



```
"default_name": "Dell 10.10.251.227",
"capture_oid": 0,
"id": 2344,
"device_id": "fff4e3fb0a0a1010",
"dhcp_name": "",
"user_mod_time": 1361900163.0,
"analysis_level": 0,
"name": "Dell 10.10.251.227",
"custom_name": null,
"cdp_name": "",
"dns_name": "",
"tag_set": [],
"tags": [],
"discover_time": 1361900141041.0,
"activity": [
    "extrahop.device.app",
    "extrahop.device.dns_client",
    "extrahop.device.net"
],
"oid": 2343,
"nb_name": ""
}
]
}
```

Metrics

The following metrics can be queried with `get_exstats`, `get_exstats_group`, and `get_exstats_total` functions. If you are looking for a specific top-level metric in the ExtraHop UI, you can find its name mousing over the metric. Refer to the **Datatypes** on page 243 section for a detailed description of the data types associated with each metric.

Application Metrics

- AAA
- Database
- DNS
- HTTP
- IBMMQ
- ICA
- L4
- Memcache
- NAS
- SSL

Device Metrics

- AAA
- AMF
- CIFS
- Database
- DNS
- FIX
- FTP
- HTTP
- IBMMQ
- ICA
- iSCSI
- L2-L3
- L4 TCP
- LDAP
- Memcache
- MS-RPC
- NFS
- SMPP
- SMTP
- SSL

Capture Metrics

Custom Metrics

Health Metrics

Application Metrics

AAA

Database

DNS

HTTP

IBMMQ

ICA

L4

LDAP

Memcache

NAS

SSL

AAA

Application AAA Metrics

176 extrahop.application.aaa application

```
field 0 req_pkts count
field 1 rsp_pkts count
field 2 req_l2_bytes count
field 3 rsp_l2_bytes count
field 4 req_rto count
field 5 rsp_rto count
field 6 req_bytes count
field 7 rsp_bytes count
field 8 rtt dataset
field 9 req count
field 10 rsp count
field 11 rsp_error count
field 12 aborts count
field 13 diameter_req count
field 14 radius_req count
field 15 method topn_count
field 16 tprocess dataset
```

177 extrahop.application.aaa_detail application

```
field 0 status_code topn_count
```

178 extrahop.application.aaa_client_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 aborts topn_count
field 11 diameter_req topn_count
field 12 radius_req topn_count
field 13 method topn_tset
field 14 tprocess topn_sset
```

193 extrahop.application.aaa_server_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
```

```
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 aborts topn_count
field 11 diameter_req topn_count
field 12 radius_req topn_count
field 13 method topn_tset
field 14 tprocess topn_sset
```

Database

Application Database Metrics

extrahop.application.db application

```
field 0 req_pkts count
field 1 rsp_pkts count
field 2 req_l2_bytes count
field 3 rsp_l2_bytes count
field 4 req_rto count
field 5 rsp_rto count
field 6 rtt dataset
field 7 req count
field 8 req_xfer dataset
field 9 rsp count
field 10 rsp_error count
field 11 tprocess dataset
field 12 rsp_ttlb dataset
field 13 rsp_xfer dataset
field 14 req_bytes count
field 15 rsp_bytes count
```

extrahop.application.db_client_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 tprocess topn_sset
```

extrahop.application.db_detail application

```
field 0 errors topn_count
field 1 user_rsp topn_count
field 2 user_rsp_error topn_count
```

extrahop.application.db_method_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 tprocess topn_sset
```

extrahop.application.db_server_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 tprocess topn_sset
```

DNS

Application DNS Metrics

extrahop.application.dns_application

```
field 0 req_pkts count
field 1 rsp_pkts count
field 2 req_l2_bytes count
field 3 rsp_l2_bytes count
field 4 req count
field 5 req_trunc count
field 6 req_timeout count
field 7 req_opcode topn_count
field 8 rsp count
field 9 rsp_trunc count
field 10 rsp_error count
field 11 rsp_rcode topn_count
field 12 tprocess dataset
```

extrahop.application.dns_client_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req topn_count
field 5 req_trunc topn_count
field 6 req_timeout topn_count
field 7 req_opcode topn_tset
field 8 rsp topn_count
field 9 rsp_trunc topn_count
field 10 rsp_error topn_count
field 11 rsp_rcode topn_tset
field 12 tprocess topn_sset
```

extrahop.application.dns_host_query_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req topn_count
field 5 req_trunc topn_count
field 6 req_timeout topn_count
field 7 req_opcode topn_tset
field 8 rsp topn_count
field 9 rsp_trunc topn_count
field 10 rsp_error topn_count
field 11 rsp_rcode topn_tset
field 12 tprocess topn_sset
```

extrahop.application.dns_server_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
```



```
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req topn_count
field 5 req_trunc topn_count
field 6 req_timeout topn_count
field 7 req_opcode topn_tset
field 8 rsp topn_count
field 9 rsp_trunc topn_count
field 10 rsp_error topn_count
field 11 rsp_rcode topn_tset
field 12 tprocess topn_sset
```

HTTP

Application HTTP Metrics

extrahop.application.http application

```
field 0 req_pkts count
field 1 rsp_pkts count
field 2 req_l2_bytes count
field 3 rsp_l2_bytes count
field 4 req_rto count
field 5 rsp_rto count
field 6 rtt dataset
field 7 req count
field 8 req_xfer dataset
field 9 rsp count
field 10 rsp_error count
field 11 tprocess dataset
field 12 rsp_xfer dataset
field 13 rsp_ttlb dataset
field 14 req_bytes count
field 15 rsp_bytes count
field 16 status_code topn_count
```

extrahop.application.http_client_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 tprocess topn_sset
field 11 status_code topn_tset
```

extrahop.application.http_detail application

```
field 0 referer topn_count
```

extrahop.application.http_server_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 tprocess topn_sset
```

```
field 11 status_code topn_tset

extrahop.application.http_uri_detail application
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 tprocess topn_sset
field 11 status_code topn_tset
```

IBMMQ

Application IBMMQ Metrics

194 extrahop.application.ibmmq application

```
field 0 req_pkts count
field 1 rsp_pkts count
field 2 req_l2_bytes count
field 3 rsp_l2_bytes count
field 4 req_rto count
field 5 rsp_rto count
field 6 req_bytes count
field 7 rsp_bytes count
field 8 rtt dataset
field 9 req count
field 10 rsp count
field 11 rsp_error count
field 12 rsp_warning count
field 13 server_msg count
field 14 client_msg count
field 15 server_to_server count
field 16 client_to_server count
field 17 method topn_count
field 18 put count
field 19 get count
```

195 extrahop.application.ibmmq_detail application

```
field 0 errors topn_count
field 1 warnings topn_count
```

196 extrahop.application.ibmmq_client_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 rsp_warning topn_count
field 11 server_msg topn_count
field 12 client_msg topn_count
field 13 server_to_server topn_count
field 14 client_to_server topn_count
field 15 method topn_tset
field 16 put topn_count
field 17 get topn_count
```

197 extrahop.application.ibmmq_server_addr_detail application

```
field 0 req_pkts topn_count
```

```
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 rsp_warning topn_count
field 11 server_msg topn_count
field 12 client_msg topn_count
field 13 server_to_server topn_count
field 14 client_to_server topn_count
field 15 method topn_tset
field 16 put topn_count
field 17 get topn_count
```

198 extrahop.application.ibmmq_queue_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 rsp_warning topn_count
field 11 server_msg topn_count
field 12 client_msg topn_count
field 13 server_to_server topn_count
field 14 client_to_server topn_count
field 15 method topn_tset
field 16 put topn_count
field 17 get topn_count
```

199 extrahop.application.ibmmq_channel_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 rsp_warning topn_count
field 11 server_msg topn_count
```

```
field 12 client_msg topn_count
field 13 server_to_server topn_count
field 14 client_to_server topn_count
field 15 method topn_tset
field 16 put topn_count
field 17 get topn_count
```

ICA

Application ICA Metrics

extrahop.application.ica application

```
field 0 client_pkts count
field 1 server_pkts count
field 2 client_l2_bytes count
field 3 server_l2_bytes count
field 4 client_rto count
field 5 server_rto count
field 6 client_bytes count
field 7 server_bytes count
field 8 rtt dataset
field 9 launches count
field 10 aborted count
field 11 encrypted count
field 12 client_msg count
field 13 server_msg count
field 14 client_cgp_msg count
field 15 server_cgp_msg count
field 16 load_time dataset
field 17 client_latency dataset
field 18 screen_updates count
field 19 channel_bytes_client topn_count
field 20 channel_bytes_server topn_count
field 21 app_bytes_client topn_count
field 22 app_bytes_server topn_count
field 23 network_latency dataset
```

extrahop.application.ica_app_detail application

```
field 0 client_pkts topn_count
field 1 server_pkts topn_count
field 2 client_l2_bytes topn_count
field 3 server_l2_bytes topn_count
field 4 client_rto topn_count
field 5 server_rto topn_count
field 6 rtt topn_sset
field 7 launches topn_count
field 8 aborted topn_count
field 9 encrypted topn_count
field 10 client_msg topn_count
field 11 server_msg topn_count
field 12 client_cgp_msg topn_count
field 13 server_cgp_msg topn_count
field 14 load_time topn_sset
field 15 client_latency topn_sset
field 16 screen_updates topn_count
field 17 channel_bytes_client topn_tset
field 18 channel_bytes_server topn_tset
field 19 network_latency topn_sset
```

extrahop.application.ica_client_addr_detail application

```
field 0 client_pkts topn_count
field 1 server_pkts topn_count
field 2 client_l2_bytes topn_count
field 3 server_l2_bytes topn_count
field 4 client_rto topn_count
field 5 server_rto topn_count
field 6 rtt topn_sset
field 7 launches topn_count
field 8 aborted topn_count
field 9 encrypted topn_count
field 10 client_msg topn_count
field 11 server_msg topn_count
field 12 client_cgp_msg topn_count
field 13 server_cgp_msg topn_count
field 14 load_time topn_sset
field 15 client_latency topn_sset
field 16 screen_updates topn_count
field 19 network_latency topn_sset
```

extrahop.application.ica_server_addr_detail application

```
field 0 client_pkts topn_count
field 1 server_pkts topn_count
field 2 client_l2_bytes topn_count
field 3 server_l2_bytes topn_count
field 4 client_rto topn_count
field 5 server_rto topn_count
field 6 rtt topn_sset
field 7 launches topn_count
field 8 aborted topn_count
field 9 encrypted topn_count
field 10 client_msg topn_count
field 11 server_msg topn_count
field 12 client_cgp_msg topn_count
field 13 server_cgp_msg topn_count
field 14 load_time topn_sset
field 15 client_latency topn_sset
field 16 screen_updates topn_count
field 19 network_latency topn_sset
```

extrahop.application.ica_user_detail application

```
field 0 client_pkts topn_count
field 1 server_pkts topn_count
field 2 client_l2_bytes topn_count
field 3 server_l2_bytes topn_count
field 4 client_rto topn_count
field 5 server_rto topn_count
field 6 rtt topn_sset
field 7 launches topn_count
field 8 aborted topn_count
```



```
field 9 encrypted topn_count
field 10 client_msg topn_count
field 11 server_msg topn_count
field 12 client_cgp_msg topn_count
field 13 server_cgp_msg topn_count
field 14 load_time topn_sset
field 15 client_latency topn_sset
field 16 screen_updates topn_count
field 19 network_latency topn_sset
```

L4

Application L4 Metrics

187 **extrahop.application.net application**

field 0 req_pkts count
field 1 rsp_pkts count
field 2 req_l2_bytes count
field 3 rsp_l2_bytes count
field 4 req_bytes count
field 5 rsp_bytes count
field 6 req_rto count
field 7 rsp_rto count
field 8 rtt dataset
field 9 turns count
field 10 req_xfer dataset
field 11 tprocess dataset
field 12 rsp_xfer dataset
field 13 req_size dataset
field 14 rsp_size dataset
field 15 connected count
field 16 closed count
field 17 aborted count
field 18 expired count
field 19 req_nagle count
field 20 rsp_nagle count
field 21 req_rcv_throttle count
field 22 rsp_rcv_throttle count
field 23 req_zwnd count
field 24 rsp_zwnd count
field 25 established snap

188 **extrahop.application.net_client_addr_detail application**

field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 turns topn_count
field 8 req_xfer topn_sset
field 9 tprocess topn_sset
field 10 rsp_xfer topn_sset
field 11 req_size topn_sset
field 12 rsp_size topn_sset
field 13 connected topn_count
field 14 closed topn_count
field 15 aborted topn_count
field 16 expired topn_count
field 17 req_nagle topn_count

```
field 18 rsp_nagle topn_count
field 19 req_rcv_throttle topn_count
field 20 rsp_rcv_throttle topn_count
field 21 req_zwnd topn_count
field 22 rsp_zwnd topn_count
field 23 established topn_snap
```

189 extrahop.application.net_server_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 turns topn_count
field 8 req_xfer topn_sset
field 9 tprocess topn_sset
field 10 rsp_xfer topn_sset
field 11 req_size topn_sset
field 12 rsp_size topn_sset
field 13 connected topn_count
field 14 closed topn_count
field 15 aborted topn_count
field 16 expired topn_count
field 17 req_nagle topn_count
field 18 rsp_nagle topn_count
field 19 req_rcv_throttle topn_count
field 20 rsp_rcv_throttle topn_count
field 21 req_zwnd topn_count
field 22 rsp_zwnd topn_count
field 23 established topn_snap
```

LDAP

LDAP Application Metrics

187 **extrahop.application.ldap application**

```
field 0 req_pkts count
field 1 rsp_pkts count
field 2 req_l2_bytes count
field 3 rsp_l2_bytes count
field 4 req_rto count
field 5 rsp_rto count
field 6 req_bytes count
field 7 rsp_bytes count
field 8 rtt dataset
field 9 req count
field 10 rsp count
field 11 rsp_error count
field 12 plain count
field 13 sasl count
field 14 tprocess dataset
field 15 msg_type topn_count
field 16 error_msg_short topn_count
```

189 **extrahop.application.ldap_client_addr_detail application**

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 plain topn_count
field 11 sasl topn_count
field 12 tprocess topn_sset
field 13 msg_type topn_tset
field 14 error_msg_short topn_tset
```

188 **extrahop.application.ldap_detail application**

```
field 0 error_msg topn_count
field 1 dn topn_count
field 2 binddn topn_count
```

190 **extrahop.application.ldap_server_addr_detail application**

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
```

```
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 rsp_error topn_count
field 10 plain topn_count
field 11 sasl topn_count
field 12 tprocess topn_sset
field 13 msg_type topn_tset
field 14 error_msg_short topn_tset
```

Memcache

Application Memcache Metrics

183 **extrahop.application.memcache** application

```
field 0 req_pkts count
field 1 rsp_pkts count
field 2 req_l2_bytes count
field 3 rsp_l2_bytes count
field 4 req_bytes count
field 5 rsp_bytes count
field 6 req_rto count
field 7 rsp_rto count
field 8 rtt dataset
field 9 req count
field 10 rsp count
field 11 noreply count
field 12 hit count
field 13 miss count
field 14 rsp_error count
field 15 method topn_count
field 16 status_code topn_count
```

184 **extrahop.application.memcache_detail** application

```
field 0 error_msg topn_count
field 1 miss topn_count
field 2 hit topn_count
```

185 **extrahop.application.memcache_client_addr_detail** application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 noreply topn_count
field 10 hit topn_count
field 11 miss topn_count
field 12 rsp_error topn_count
field 13 method topn_tset
field 14 status_code topn_tset
```

186 **extrahop.application.memcache_server_addr_detail** application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
```

```
field 6 rtt topn_sset
field 7 req topn_count
field 8 rsp topn_count
field 9 noreply topn_count
field 10 hit topn_count
field 11 miss topn_count
field 12 rsp_error topn_count
field 13 method topn_tset
field 14 status_code topn_tset
```

NAS

Application NAS Metrics

extrahop.application.nas application

```
field 0 req_pkts count
field 1 rsp_pkts count
field 2 req_l2_bytes count
field 3 rsp_l2_bytes count
field 4 req_rto count
field 5 rsp_rto count
field 6 req_bytes count
field 7 rsp_bytes count
field 8 rtt dataset
field 9 rsp count
field 10 rsp_error count
field 11 read count
field 12 write count
field 13 lock count
field 14 fs_info count
field 15 read_bytes count
field 16 write_bytes count
field 17 fs_info_bytes count
field 18 access_time dataset
```

extrahop.application.nas_client_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 req_bytes topn_count
field 7 rsp_bytes topn_count
field 8 rtt topn_sset
field 9 rsp topn_count
field 10 rsp_error topn_count
field 11 read topn_count
field 12 write topn_count
field 13 lock topn_count
field 14 fs_info topn_count
field 15 access_time topn_sset
```

extrahop.application.nas_detail application

```
field 0 errors topn_count
```

extrahop.application.nas_file_info_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
```



```
field 5 rsp_rto topn_count
field 6 req_bytes topn_count
field 7 rsp_bytes topn_count
field 8 rtt topn_sset
field 9 rsp topn_count
field 10 rsp_error topn_count
field 11 read topn_count
field 12 write topn_count
field 13 lock topn_count
field 14 fs_info topn_count
field 15 access_time topn_sset
```

extrahop.application.nas_server_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 req_bytes topn_count
field 7 rsp_bytes topn_count
field 8 rtt topn_sset
field 9 rsp topn_count
field 10 rsp_error topn_count
field 11 read topn_count
field 12 write topn_count
field 13 lock topn_count
field 14 fs_info topn_count
field 15 access_time topn_sset
```

extrahop.application.nas_user_info_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 req_bytes topn_count
field 7 rsp_bytes topn_count
field 8 rtt topn_sset
field 9 rsp topn_count
field 10 rsp_error topn_count
field 11 read topn_count
field 12 write topn_count
field 13 lock topn_count
field 14 fs_info topn_count
field 15 access_time topn_sset
```

SSL

Application SSL Metrics

179 extrahop.application.ssl application

```
field 0 req_pkts count
field 1 rsp_pkts count
field 2 req_l2_bytes count
field 3 rsp_l2_bytes count
field 4 req_bytes count
field 5 rsp_bytes count
field 6 req_rto count
field 7 rsp_rto count
field 8 rtt dataset
field 9 connected count
field 10 resumed count
field 11 decrypted count
field 12 aborted count
field 13 renegotiated count
field 14 compressed count
field 15 numberofv2hello count
field 16 version topn_count
field 17 cipher topn_count
field 18 alerts topn_count
```

180 extrahop.application.ssl_client_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 connected topn_count
field 8 resumed topn_count
field 9 decrypted topn_count
field 10 aborted topn_count
field 11 renegotiated topn_count
field 12 compressed topn_count
field 13 numberofv2hello topn_count
field 14 version topn_tset
field 15 cipher topn_tset
field 16 alerts topn_tset
```

181 extrahop.application.ssl_server_addr_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
```

```
field 6 rtt topn_sset
field 7 connected topn_count
field 8 resumed topn_count
field 9 decrypted topn_count
field 10 aborted topn_count
field 11 renegotiated topn_count
field 12 compressed topn_count
field 13 numberofv2hello topn_count
field 14 version topn_tset
field 15 cipher topn_tset
field 16 alerts topn_tset
```

182 extrahop.application.ssl_certificate_detail application

```
field 0 req_pkts topn_count
field 1 rsp_pkts topn_count
field 2 req_l2_bytes topn_count
field 3 rsp_l2_bytes topn_count
field 4 req_rto topn_count
field 5 rsp_rto topn_count
field 6 rtt topn_sset
field 7 connected topn_count
field 8 resumed topn_count
field 9 decrypted topn_count
field 10 aborted topn_count
field 11 renegotiated topn_count
field 12 compressed topn_count
field 13 numberofv2hello topn_count
field 14 version topn_tset
field 15 cipher topn_tset
field 16 alerts topn_tset
field 17 cert_expiration topn_time
field 18 cert_subject_req_bytes topn_count
field 19 cert_subject_rsp_bytes topn_count
field 20 cert_ip_req_bytes topn_tset
field 21 cert_ip_rsp_bytes topn_tset
```

Device Metrics

AAA

AMF

CIFS

Database

DNS

FIX

FTP

HTTP

IBMMQ

ICA

iSCSI

L2-L3

L4 TCP

LDAP

Memcache

MS-RPC

NFS

SMPP

SMTP

SSL

AAA

Device AAA Metrics

extrahop.device.aaa_client device

- field 0 req count
- field 1 rsp count
- field 2 rsp_error count
- field 3 method topn_count
- field 4 status_code topn_count
- field 5 tprocess dataset
- field 6 aborts count
- field 7 diameter_req count
- field 8 radius_req count

extrahop.device.aaa_client_detail device

- field 0 req topn_count
- field 1 rsp topn_count
- field 2 rsp_error topn_count
- field 3 method topn_tset
- field 4 status_code topn_tset
- field 5 aborts topn_count

extrahop.device.aaa_server device

- field 0 req count
- field 1 rsp count
- field 2 rsp_error count
- field 3 method topn_count
- field 4 status_code topn_count
- field 5 tprocess dataset
- field 6 aborts count
- field 7 diameter_req count
- field 8 radius_req count

extrahop.device.aaa_server_detail device

- field 0 req topn_count
- field 1 rsp topn_count
- field 2 rsp_error topn_count
- field 3 method topn_tset
- field 4 status_code topn_tset
- field 5 aborts topn_count

AMF

Device AMF Metrics

extrahop.device.amf_client device

```
field 0 req count
field 1 rsp count
field 2 rsp_error count
field 3 req_no_len count
field 4 rsp_no_len count
field 5 req_size dataset
field 6 rsp_size dataset
field 7 req_xfer dataset
field 8 tprocess dataset
field 9 rsp_xfer dataset
```

extrahop.device.amf_client_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
field 3 req_no_len topn_count
field 4 rsp_no_len topn_count
field 5 req_size topn_sset
field 6 rsp_size topn_sset
field 8 tprocess topn_sset
```

extrahop.device.amf_server device

```
field 0 req count
field 1 rsp count
field 2 rsp_error count
field 3 req_no_len count
field 4 rsp_no_len count
field 5 req_size dataset
field 6 rsp_size dataset
field 7 req_xfer dataset
field 8 tprocess dataset
field 9 rsp_xfer dataset
```

extrahop.device.amf_server_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
field 3 req_no_len topn_count
field 4 rsp_no_len topn_count
field 5 req_size topn_sset
field 6 rsp_size topn_sset
field 8 tprocess topn_sset
```

extrahop.device.uri_amf_client_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
```

```
field 3 req_no_len topn_count
field 4 rsp_no_len topn_count
field 5 req_size topn_sset
field 6 rsp_size topn_sset
field 8 tprocess topn_sset
```

extrahop.device.uri_amf_server_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
field 3 req_no_len topn_count
field 4 rsp_no_len topn_count
field 5 req_size topn_sset
field 6 rsp_size topn_sset
field 8 tprocess topn_sset
```

CIFS

Device CIFS Metrics

extrahop.device.cifs_client device

```
field 0 method topn_count
field 1 req count
field 2 rsp count
field 3 rsp_error count
field 4 req_xfer dataset
field 5 tprocess dataset
field 6 rsp_xfer dataset
field 7 rsp_ttlb dataset
field 8 req_size dataset
field 9 rsp_size dataset
field 14 bytes_read count
field 15 bytes_write count
field 16 req_write count
field 17 req_read count
field 18 req_lock count
field 19 req_fsinfo count
field 20 access_time dataset
field 26 fsinfo_bytes count
```

extrahop.device.cifs_client_detail device

```
field 0 method topn_tset
field 1 req topn_count
field 2 rsp topn_count
field 3 rsp_error topn_count
field 4 req_size topn_sset
field 5 rsp_size topn_sset
field 6 error_msg topn_count
field 7 req_write topn_count
field 8 req_read topn_count
field 9 req_lock topn_count
field 10 req_fsinfo topn_count
field 11 user_info topn_count
field 12 user_info_bytes_in topn_count
field 13 user_info_bytes_out topn_count
field 14 file_info topn_count
field 15 file_info_bytes_in topn_count
field 16 file_info_bytes_out topn_count
field 17 method_info topn_count
field 18 method_info_bytes_in topn_count
field 19 method_info_bytes_out topn_count
field 20 file_info_access_time topn_sset
```

extrahop.device.cifs_server device

```
field 0 method topn_count
field 1 req count
field 2 rsp count
```



```
field 3 rsp_error count
field 4 req_xfer dataset
field 5 tprocess dataset
field 6 rsp_xfer dataset
field 7 rsp_ttlb dataset
field 8 req_size dataset
field 9 rsp_size dataset
field 14 bytes_read count
field 15 bytes_write count
field 16 req_write count
field 17 req_read count
field 18 req_lock count
field 19 req_fsinfo count
field 20 access_time dataset
field 26 fsinfo_bytes count
```

extrahop.device.cifs_server_detail device

```
field 0 method topn_tset
field 1 req topn_count
field 2 rsp topn_count
field 3 rsp_error topn_count
field 4 req_size topn_sset
field 5 rsp_size topn_sset
field 6 error_msg topn_count
field 7 req_write topn_count
field 8 req_read topn_count
field 9 req_lock topn_count
field 10 req_fsinfo topn_count
field 11 user_info topn_count
field 12 user_info_bytes_in topn_count
field 13 user_info_bytes_out topn_count
field 14 file_info topn_count
field 15 file_info_bytes_in topn_count
field 16 file_info_bytes_out topn_count
field 17 method_info topn_count
field 18 method_info_bytes_in topn_count
field 19 method_info_bytes_out topn_count
field 20 file_info_access_time topn_sset
```

Database

Device Database Metrics

extrahop.device.db_client device

```
field 0 method topn_count
field 1 req count
field 2 rsp count
field 3 rsp_error count
field 4 req_xfer dataset
field 5 tprocess dataset
field 6 rsp_xfer dataset
field 7 rsp_ttlb dataset
field 8 req_size dataset
field 9 rsp_size dataset
field 14 instance topn_count
field 17 req_abort count
field 18 rsp_abort count
```

extrahop.device.db_client_detail device

```
field 0 method topn_tset
field 1 req topn_count
field 2 rsp topn_count
field 3 rsp_error topn_count
field 4 req_size topn_sset
field 5 rsp_size topn_sset
field 7 req_abort topn_count
field 8 rsp_abort topn_count
field 9 method_info topn_count
field 10 method_info_tprocess topn_sset
field 11 error_msg topn_count
field 12 user_info_req topn_count
field 13 user_info_error topn_count
```

extrahop.device.db_server device

```
field 0 method topn_count
field 1 req count
field 2 rsp count
field 3 rsp_error count
field 4 req_xfer dataset
field 5 tprocess dataset
field 6 rsp_xfer dataset
field 7 rsp_ttlb dataset
field 8 req_size dataset
field 9 rsp_size dataset
field 14 instance topn_count
field 17 req_abort count
field 18 rsp_abort count
```

extrahop.device.db_server_detail device

```
field 0 method topn_tset
field 1 req topn_count
```

```
field 2 rsp topn_count
field 3 rsp_error topn_count
field 4 req_size topn_sset
field 5 rsp_size topn_sset
field 7 req_abort topn_count
field 8 rsp_abort topn_count
field 9 method_info topn_count
field 10 method_info_tprocess topn_sset
field 11 error_msg topn_count
field 12 user_info_req topn_count
field 13 user_info_error topn_count
```

extrahop.device.instance_db_client_detail device

```
field 0 method topn_tset
field 1 req topn_count
field 2 rsp topn_count
field 3 rsp_error topn_count
field 4 req_size topn_sset
field 5 rsp_size topn_sset
field 7 req_abort topn_count
field 8 rsp_abort topn_count
field 9 method_info topn_count
field 10 method_info_tprocess topn_sset
field 11 error_msg topn_count
field 12 user_info_req topn_count
field 13 user_info_error topn_count
```

extrahop.device.instance_db_server_detail device

```
field 0 method topn_tset
field 1 req topn_count
field 2 rsp topn_count
field 3 rsp_error topn_count
field 4 req_size topn_sset
field 5 rsp_size topn_sset
field 7 req_abort topn_count
field 8 rsp_abort topn_count
field 9 method_info topn_count
field 10 method_info_tprocess topn_sset
field 11 error_msg topn_count
field 12 user_info_req topn_count
field 13 user_info_error topn_count
```

DNS

Device DNS Metrics

extrahop.device.dns_client device3

```
field 0 req count
field 1 req_trunc count
field 2 req_timeout count
field 3 req_rectype topn_count
field 4 req_opcode topn_count
field 5 rsp count
field 6 rsp_trunc count
field 7 rsp_error count
field 8 rsp_rectype topn_count
field 9 rsp_rcode topn_count
field 10 host_query topn_count
field 11 host_error topn_count
field 12 tprocess dataset
```

extrahop.device.dns_client_detail device

```
field 0 req topn_count
field 1 req_opcode topn_tset
field 2 req_rectype topn_tset
field 3 rsp_error topn_count
field 4 rsp_rectype topn_tset
field 5 rsp_rcode topn_tset
field 6 tprocess topn_sset
```

extrahop.device.dns_server device

```
field 0 req count
field 1 req_trunc count
field 2 req_timeout count
field 3 req_rectype topn_count
field 4 req_opcode topn_count
field 5 rsp count
field 6 rsp_trunc count
field 7 rsp_error count
field 8 rsp_rectype topn_count
field 9 rsp_rcode topn_count
field 10 host_query topn_count
field 11 host_error topn_count
field 12 tprocess dataset
```

extrahop.device.dns_server_detail device

```
field 0 req topn_count
field 1 req_opcode topn_tset
field 2 req_rectype topn_tset
field 3 rsp_error topn_count
field 4 rsp_rectype topn_tset
field 5 rsp_rcode topn_tset
field 6 tprocess topn_sset
```

FIX

Device FIX Metrics

extrahop.device.fix_client device

```
field 0 version topn_count
field 1 msgtype topn_count
field 2 sender topn_count
field 3 target topn_count
field 4 possdupflag count
field 5 possresend count
field 6 body_size dataset
field 7 transactions count
field 8 tprocess dataset
field 9 tprocess_msgtype topn_dset
field 10 tr_msgtype topn_count
field 11 aborted count
field 12 rsp count
field 13 rsp_error count
field 14 req count
```

extrahop.device.fix_client_detail device

```
field 0 version topn_tset
field 1 msgtype topn_tset
field 2 sender topn_tset
field 3 target topn_tset
field 4 posdup topn_count
field 5 posresend topn_count
field 6 transactions topn_count
field 7 tprocess topn_sset
field 8 aborted topn_count
field 9 rsp topn_count
field 10 rsp_error topn_count
field 11 req topn_count
field 12 reject_reason topn_count
```

extrahop.device.fix_server device

```
field 0 version topn_count
field 1 msgtype topn_count
field 2 sender topn_count
field 3 target topn_count
field 4 possdupflag count
field 5 possresend count
field 6 body_size dataset
field 7 transactions count
field 8 tprocess dataset
field 9 tprocess_msgtype topn_dset
field 10 tr_msgtype topn_count
field 11 aborted count
field 12 rsp count
field 13 rsp_error count
```

field 14 req count

extrahop.device.fix_server_detail device

field 0 version topn_tset
field 1 msgtype topn_tset
field 2 sender topn_tset
field 3 target topn_tset
field 4 posdup topn_count
field 5 posresend topn_count
field 6 transactions topn_count
field 7 tprocess topn_sset
field 8 aborted topn_count
field 9 rsp topn_count
field 10 rsp_error topn_count
field 11 req topn_count
field 12 reject_reason topn_count

FTP

Device FTP Metrics

extrahop.device.ftp_client device

```
field 0 req count
field 1 rsp count
field 2 rsp_error count
field 3 data_req count
field 4 method topn_count
field 5 status_code topn_count
field 6 req_xfer dataset
field 7 tprocess dataset
field 8 rsp_xfer dataset
field 9 req_size dataset
field 10 rsp_size dataset
field 11 data_connect count
field 12 bytes_read count
field 13 bytes_write count
field 14 rsp_warning count
```

extrahop.device.ftp_client_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
field 3 method topn_tset
field 4 status_code topn_tset
field 6 tprocess topn_sset
field 8 req_size topn_sset
field 9 rsp_size topn_sset
field 10 file_req topn_count
field 11 file_bytes_in topn_count
field 12 file_bytes_out topn_count
field 13 file_error topn_count
field 14 error_msg topn_count
field 15 data_req topn_count
field 16 data_connect topn_count
field 17 rsp_warning topn_count
field 18 warning_msg topn_count
```

extrahop.device.ftp_server device

```
field 0 req count
field 1 rsp count
field 2 rsp_error count
field 3 data_req count
field 4 method topn_count
field 5 status_code topn_count
field 6 req_xfer dataset
field 7 tprocess dataset
field 8 rsp_xfer dataset
field 9 req_size dataset
```

```
field 10 rsp_size dataset
field 11 data_connect count
field 12 bytes_read count
field 13 bytes_write count
field 14 rsp_warning count
```

extrahop.device.ftp_server_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
field 3 method topn_tset
field 4 status_code topn_tset
field 6 tprocess topn_sset
field 8 req_size topn_sset
field 9 rsp_size topn_sset
field 10 file_req topn_count
field 11 file_bytes_in topn_count
field 12 file_bytes_out topn_count
field 13 file_error topn_count
field 14 error_msg topn_count
field 15 data_req topn_count
field 16 data_connect topn_count
field 17 rsp_warning topn_count
field 18 warning_msg topn_count
```

extrahop.device.user_ftp_client_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
field 3 method topn_tset
field 4 status_code topn_tset
field 6 tprocess topn_sset
field 8 req_size topn_sset
field 9 rsp_size topn_sset
field 10 file_req topn_count
field 11 file_bytes_in topn_count
field 12 file_bytes_out topn_count
field 13 file_error topn_count
field 15 data_req topn_count
field 16 data_connect topn_count
field 17 rsp_warning topn_count
```

extrahop.device.user_ftp_server_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
field 3 method topn_tset
field 4 status_code topn_tset
field 6 tprocess topn_sset
field 8 req_size topn_sset
field 9 rsp_size topn_sset
```



```
field 10 file_req topn_count
field 11 file_bytes_in topn_count
field 12 file_bytes_out topn_count
field 13 file_error topn_count
field 15 data_req topn_count
field 16 data_connect topn_count
field 17 rsp_warning topn_count
```

HTTP

Device HTTP Metrics

extrahop.device.host_http_client_detail device

```
field 0 req topn_count
field 1 pipelined topn_count
field 2 req_abort topn_count
field 3 method topn_tset
field 4 tprocess topn_sset
field 5 req_size topn_sset
field 6 rsp topn_count
field 7 rsp_abort topn_count
field 8 rsp_error topn_count
field 9 status_code topn_tset
field 11 chunked topn_count
field 12 compressed topn_count
field 13 authed topn_count
field 14 rsp_size topn_sset
```

extrahop.device.host_http_server_detail device

```
field 0 req topn_count
field 1 pipelined topn_count
field 2 req_abort topn_count
field 3 method topn_tset
field 4 tprocess topn_sset
field 5 req_size topn_sset
field 6 rsp topn_count
field 7 rsp_abort topn_count
field 8 rsp_error topn_count
field 9 status_code topn_tset
field 11 chunked topn_count
field 12 compressed topn_count
field 13 authed topn_count
field 14 rsp_size topn_sset
```

extrahop.device.http_client device

```
field 0 req count
field 1 pipelined count
field 2 req_abort count
field 3 method topn_count
field 4 req_xfer dataset
field 5 tprocess dataset
field 6 rsp_xfer dataset
field 7 rsp_ttlb dataset
field 8 req_size dataset
field 9 rsp count
field 10 rsp_abort count
field 11 rsp_error count
field 12 status_code topn_count
field 13 content_type topn_count
```

field 14 chunked count
field 15 compressed count
field 16 authed count
field 17 rsp_size dataset

extrahop.device.http_client_detail device

field 0 req topn_count
field 1 pipelined topn_count
field 2 req_abort topn_count
field 3 method topn_tset
field 4 tprocess topn_sset
field 5 req_size topn_sset
field 6 rsp topn_count
field 7 rsp_abort topn_count
field 8 rsp_error topn_count
field 9 status_code topn_tset
field 11 chunked topn_count
field 12 compressed topn_count
field 13 authed topn_count
field 14 rsp_size topn_sset
field 15 referer topn_count

extrahop.device.http_server device

field 0 req count
field 1 pipelined count
field 2 req_abort count
field 3 method topn_count
field 4 req_xfer dataset
field 5 tprocess dataset
field 6 rsp_xfer dataset
field 7 rsp_ttlb dataset
field 8 req_size dataset
field 9 rsp count
field 10 rsp_abort count
field 11 rsp_error count
field 12 status_code topn_count
field 13 content_type topn_count
field 14 chunked count
field 15 compressed count
field 16 authed count
field 17 rsp_size dataset

extrahop.device.http_server_detail device

field 0 req topn_count
field 1 pipelined topn_count
field 2 req_abort topn_count
field 3 method topn_tset
field 4 tprocess topn_sset
field 5 req_size topn_sset
field 6 rsp topn_count
field 7 rsp_abort topn_count

```
field 8 rsp_error topn_count
field 9 status_code topn_tset
field 11 chunked topn_count
field 12 compressed topn_count
field 13 authed topn_count
field 14 rsp_size topn_sset
field 15 referer topn_count
```

extrahop.device.uri_http_client_detail device

```
field 0 req topn_count
field 1 pipelined topn_count
field 2 req_abort topn_count
field 3 method topn_tset
field 4 tprocess topn_sset
field 5 req_size topn_sset
field 6 rsp topn_count
field 7 rsp_abort topn_count
field 8 rsp_error topn_count
field 9 status_code topn_tset
field 11 chunked topn_count
field 12 compressed topn_count
field 13 authed topn_count
field 14 rsp_size topn_sset
```

extrahop.device.uri_http_server_detail device

```
field 0 req topn_count
field 1 pipelined topn_count
field 2 req_abort topn_count
field 3 method topn_tset
field 4 tprocess topn_sset
field 5 req_size topn_sset
field 6 rsp topn_count
field 7 rsp_abort topn_count
field 8 rsp_error topn_count
field 9 status_code topn_tset
field 11 chunked topn_count
field 12 compressed topn_count
field 13 authed topn_count
field 14 rsp_size topn_sset
```

IBMMQ

Device IBMMQ Metrics

extrahop.device.channel_ibmmq_client_detail device

```
field 0 rsp topn_count
field 1 rsp_pcf topn_count
field 2 req topn_count
field 3 req_pcf topn_count
field 4 rsp_error topn_count
field 5 rsp_pcf_error topn_count
field 6 server_msg topn_count
field 7 client_msg topn_count
field 8 method topn_tset
field 9 method_pcf topn_tset
field 10 msg_format topn_tset
field 11 rsp_size topn_sset
field 12 req_size topn_sset
field 13 error topn_tset
field 14 error_pcf topn_tset
field 15 warning topn_tset
field 16 warning_pcf topn_tset
field 17 rsp_warning topn_count
field 18 rsp_warning_pcf topn_count
field 19 server_to_server topn_count
field 20 client_to_server topn_count
field 21 put topn_count
field 22 get topn_count
```

extrahop.device.channel_ibmmq_server_detail device

```
field 0 rsp topn_count
field 1 rsp_pcf topn_count
field 2 req topn_count
field 3 req_pcf topn_count
field 4 rsp_error topn_count
field 5 rsp_pcf_error topn_count
field 6 server_msg topn_count
field 7 client_msg topn_count
field 8 method topn_tset
field 9 method_pcf topn_tset
field 10 msg_format topn_tset
field 11 rsp_size topn_sset
field 12 req_size topn_sset
field 13 error topn_tset
field 14 error_pcf topn_tset
field 15 warning topn_tset
field 16 warning_pcf topn_tset
field 17 rsp_warning topn_count
field 18 rsp_warning_pcf topn_count
field 19 server_to_server topn_count
field 20 client_to_server topn_count
```

```
field 21 put topn_count
field 22 get topn_count
```

extrahop.device.ibmmq_client device

```
field 0 rsp count
field 1 rsp_pcf count
field 2 req count
field 3 req_pcf count
field 4 rsp_error count
field 5 rsp_pcf_error count
field 6 method topn_count
field 7 method_pcf topn_count
field 8 server_msg count
field 9 client_msg count
field 10 msg_format topn_count
field 11 rsp_size dataset
field 12 req_size dataset
field 13 error topn_count
field 14 error_pcf topn_count
field 15 warning topn_count
field 16 warning_pcf topn_count
field 17 rsp_warning count
field 18 rsp_pcf_warning count
field 19 server_to_server count
field 20 client_to_server count
field 21 put count
field 22 get count
```

extrahop.device.ibmmq_client_detail device

```
field 0 rsp topn_count
field 1 rsp_pcf topn_count
field 2 req topn_count
field 3 req_pcf topn_count
field 4 rsp_error topn_count
field 5 rsp_pcf_error topn_count
field 6 server_msg topn_count
field 7 client_msg topn_count
field 8 method topn_tset
field 9 method_pcf topn_tset
field 10 msg_format topn_tset
field 11 rsp_size topn_sset
field 12 req_size topn_sset
field 13 error topn_tset
field 14 error_pcf topn_tset
field 15 warning topn_tset
field 16 warning_pcf topn_tset
field 17 rsp_warning topn_count
field 18 rsp_warning_pcf topn_count
field 19 server_to_server topn_count
field 20 client_to_server topn_count
field 21 put topn_count
```

field 22 get topn_count

extrahop.device.ibmmq_server device

field 0 rsp count
field 1 rsp_pcf count
field 2 req count
field 3 req_pcf count
field 4 rsp_error count
field 5 rsp_pcf_error count
field 6 method topn_count
field 7 method_pcf topn_count
field 8 server_msg count
field 9 client_msg count
field 10 msg_format topn_count
field 11 rsp_size dataset
field 12 req_size dataset
field 13 error topn_count
field 14 error_pcf topn_count
field 15 warning topn_count
field 16 warning_pcf topn_count
field 17 rsp_warning count
field 18 rsp_pcf_warning count
field 19 server_to_server count
field 20 client_to_server count
field 21 put count
field 22 get count

extrahop.device.ibmmq_server_detail device

field 0 rsp topn_count
field 1 rsp_pcf topn_count
field 2 req topn_count
field 3 req_pcf topn_count
field 4 rsp_error topn_count
field 5 rsp_pcf_error topn_count
field 6 server_msg topn_count
field 7 client_msg topn_count
field 8 method topn_tset
field 9 method_pcf topn_tset
field 10 msg_format topn_tset
field 11 rsp_size topn_sset
field 12 req_size topn_sset
field 13 error topn_tset
field 14 error_pcf topn_tset
field 15 warning topn_tset
field 16 warning_pcf topn_tset
field 17 rsp_warning topn_count
field 18 rsp_warning_pcf topn_count
field 19 server_to_server topn_count
field 20 client_to_server topn_count
field 21 put topn_count
field 22 get topn_count

extrahop.device.queue_ibmmq_client_detail device

```
field 0 rsp topn_count
field 1 rsp_pcf topn_count
field 2 req topn_count
field 3 req_pcf topn_count
field 4 rsp_error topn_count
field 5 rsp_pcf_error topn_count
field 6 server_msg topn_count
field 7 client_msg topn_count
field 8 method topn_tset
field 9 method_pcf topn_tset
field 10 msg_format topn_tset
field 11 rsp_size topn_sset
field 12 req_size topn_sset
field 13 error topn_tset
field 14 error_pcf topn_tset
field 15 warning topn_tset
field 16 warning_pcf topn_tset
field 17 rsp_warning topn_count
field 18 rsp_warning_pcf topn_count
field 19 server_to_server topn_count
field 20 client_to_server topn_count
field 21 put topn_count
field 22 get topn_count
```

extrahop.device.queue_ibmmq_server_detail device

```
field 0 rsp topn_count
field 1 rsp_pcf topn_count
field 2 req topn_count
field 3 req_pcf topn_count
field 4 rsp_error topn_count
field 5 rsp_pcf_error topn_count
field 6 server_msg topn_count
field 7 client_msg topn_count
field 8 method topn_tset
field 9 method_pcf topn_tset
field 10 msg_format topn_tset
field 11 rsp_size topn_sset
field 12 req_size topn_sset
field 13 error topn_tset
field 14 error_pcf topn_tset
field 15 warning topn_tset
field 16 warning_pcf topn_tset
field 17 rsp_warning topn_count
field 18 rsp_warning_pcf topn_count
field 19 server_to_server topn_count
field 20 client_to_server topn_count
field 21 put topn_count
field 22 get topn_count
```


ICA

Device ICA Metrics

extrahop.device.ica_client device

field 0 launches count
field 1 load_time dataset
field 2 login_time dataset
field 3 client_msg count
field 4 server_msg count
field 5 channel_bytes_in topn_count
field 6 channel_bytes_out topn_count
field 7 aborted count
field 8 client_cgp_msg count
field 9 server_cgp_msg count
field 10 client_latency dataset
field 11 encrypted_sessions count
field 12 screen_updates count
field 13 network_latency dataset

extrahop.device.ica_client_detail device

field 0 launches topn_count
field 1 session_duration topn_sset
field 2 load_time topn_sset
field 3 login_time topn_sset
field 4 client_msg topn_count
field 5 server_msg topn_count
field 6 bytes_in topn_count
field 7 bytes_out topn_count
field 8 channel_bytes_in topn_tset
field 9 channel_bytes_out topn_tset
field 10 client_type topn_count
field 11 aborted topn_count
field 12 client_cgp_msg topn_count
field 13 server_cgp_msg topn_count
field 14 client_latency topn_sset
field 15 screen_updates topn_count
field 16 network_latency topn_sset

extrahop.device.ica_server device

field 0 launches count
field 1 load_time dataset
field 2 login_time dataset
field 3 client_msg count
field 4 server_msg count
field 5 channel_bytes_in topn_count
field 6 channel_bytes_out topn_count
field 7 aborted count
field 8 client_cgp_msg count
field 9 server_cgp_msg count
field 10 client_latency dataset

```
field 11 encrypted_sessions count
field 12 screen_updates count
field 13 network_latency dataset
```

extrahop.device.ica_server_detail device

```
field 0 launches topn_count
field 1 session_duration topn_sset
field 2 load_time topn_sset
field 3 login_time topn_sset
field 4 client_msg topn_count
field 5 server_msg topn_count
field 6 bytes_in topn_count
field 7 bytes_out topn_count
field 8 channel_bytes_in topn_tset
field 9 channel_bytes_out topn_tset
field 10 client_type topn_count
field 11 aborted topn_count
field 12 client_cgp_msg topn_count
field 13 server_cgp_msg topn_count
field 14 client_latency topn_sset
field 15 screen_updates topn_count
field 16 network_latency topn_sset
```

extrahop.device.user_ica_client_detail device

```
field 0 launches topn_count
field 1 session_duration topn_sset
field 2 load_time topn_sset
field 3 login_time topn_sset
field 4 client_msg topn_count
field 5 server_msg topn_count
field 6 bytes_in topn_count
field 7 bytes_out topn_count
field 8 channel_bytes_in topn_tset
field 9 channel_bytes_out topn_tset
field 10 client_type topn_count
field 11 aborted topn_count
field 12 client_cgp_msg topn_count
field 13 server_cgp_msg topn_count
field 14 client_latency topn_sset
field 15 screen_updates topn_count
```

extrahop.device.user_ica_server_detail device

```
field 0 launches topn_count
field 1 session_duration topn_sset
field 2 load_time topn_sset
field 3 login_time topn_sset
field 4 client_msg topn_count
field 5 server_msg topn_count
field 6 bytes_in topn_count
field 7 bytes_out topn_count
field 8 channel_bytes_in topn_tset
```

```
field 9 channel_bytes_out topn_tset
field 10 client_type topn_count
field 11 aborted topn_count
field 12 client_cgp_msg topn_count
field 13 server_cgp_msg topn_count
field 14 client_latency topn_sset
field 15 screen_updates topn_count
```

iSCSI

Device iSCSI Metrics

extrahop.device.iscsi_client device

```
field 0 rsp count
field 1 rsp_error count
field 2 read count
field 3 write count
field 4 bytes_read count
field 5 bytes_write count
field 6 opcodes topn_count
field 7 target topn_count
field 8 header_digest count
field 9 data_digest count
field 10 reject count
field 11 login_failed count
field 12 reject_reason topn_count
field 13 login_error topn_count
field 14 session count
```

extrahop.device.iscsi_client_detail device

```
field 0 rsp topn_count
field 1 rsp_error topn_count
field 2 read topn_count
field 3 write topn_count
field 4 bytes_read topn_count
field 5 bytes_write topn_count
field 6 opcodes topn_tset
field 7 header_digest topn_count
field 8 data_digest topn_count
field 9 reject topn_count
field 10 login_failed topn_count
field 11 reject_reason topn_tset
field 12 login_error topn_tset
field 13 session topn_count
field 14 rsp_init topn_count
field 15 bytes_read_init topn_count
field 16 bytes_write_init topn_count
field 17 opcode_init topn_count
field 18 rsp_error_init topn_count
```

extrahop.device.iscsi_server device

```
field 0 rsp count
field 1 rsp_error count
field 2 read count
field 3 write count
field 4 bytes_read count
field 5 bytes_write count
field 6 opcodes topn_count
field 7 target topn_count
```

```
field 8 header_digest count
field 9 data_digest count
field 10 reject count
field 11 login_failed count
field 12 reject_reason topn_count
field 13 login_error topn_count
field 14 session count
```

extrahop.device.iscsi_server_detail device

```
field 0 rsp topn_count
field 1 rsp_error topn_count
field 2 read topn_count
field 3 write topn_count
field 4 bytes_read topn_count
field 5 bytes_write topn_count
field 6 opcodes topn_tset
field 7 header_digest topn_count
field 8 data_digest topn_count
field 9 reject topn_count
field 10 login_failed topn_count
field 11 reject_reason topn_tset
field 12 login_error topn_tset
field 13 session topn_count
field 14 rsp_init topn_count
field 15 bytes_read_init topn_count
field 16 bytes_write_init topn_count
field 17 opcode_init topn_count
field 18 rsp_error_init topn_count
```

extrahop.device.iscsi_target_client device

```
field 0 rsp count
field 1 rsp_error count
field 2 read count
field 3 write count
field 4 bytes_read count
field 5 bytes_write count
field 6 opcodes topn_count
field 7 target topn_count
field 8 header_digest count
field 9 data_digest count
field 10 reject count
field 11 login_failed count
field 12 reject_reason topn_count
field 13 login_error topn_count
field 14 session count
```

extrahop.device.iscsi_target_client_detail device

```
field 0 rsp topn_count
field 1 rsp_error topn_count
field 2 read topn_count
field 3 write topn_count
```

```
field 4 bytes_read topn_count
field 5 bytes_write topn_count
field 6 opcodes topn_tset
field 7 header_digest topn_count
field 8 data_digest topn_count
field 9 reject topn_count
field 10 login_failed topn_count
field 11 reject_reason topn_tset
field 12 login_error topn_tset
field 13 session topn_count
field 14 rsp_init topn_count
field 15 bytes_read_init topn_count
field 16 bytes_write_init topn_count
field 17 opcode_init topn_count
field 18 rsp_error_init topn_count
```

extrahop.device.iscsi_target_server device

```
field 0 rsp count
field 1 rsp_error count
field 2 read count
field 3 write count
field 4 bytes_read count
field 5 bytes_write count
field 6 opcodes topn_count
field 7 target topn_count
field 8 header_digest count
field 9 data_digest count
field 10 reject count
field 11 login_failed count
field 12 reject_reason topn_count
field 13 login_error topn_count
field 14 session count
```

extrahop.device.iscsi_target_server_detail device

```
field 0 rsp topn_count
field 1 rsp_error topn_count
field 2 read topn_count
field 3 write topn_count
field 4 bytes_read topn_count
field 5 bytes_write topn_count
field 6 opcodes topn_tset
field 7 header_digest topn_count
field 8 data_digest topn_count
field 9 reject topn_count
field 10 login_failed topn_count
field 11 reject_reason topn_tset
field 12 login_error topn_tset
field 13 session topn_count
field 14 rsp_init topn_count
field 15 bytes_read_init topn_count
field 16 bytes_write_init topn_count
```

```
field 17 opcode_init topn_count  
field 18 rsp_error_init topn_count
```

L2-L3

Device L2-L3 Metrics

```
extrahop.device.net device
  field 0 pkts_in count
  field 1 pkts_out count
  field 2 bytes_in count
  field 3 bytes_out count
  field 4 frame_cast_unicast_pkts count
  field 5 frame_cast_unicast_bytes count
  field 6 frame_cast_multicast_pkts count
  field 7 frame_cast_multicast_bytes count
  field 8 frame_cast_broadcast_pkts count
  field 9 frame_cast_broadcast_bytes count
  field 10 frame_size_64_in count
  field 11 frame_size_128_in count
  field 12 frame_size_256_in count
  field 13 frame_size_512_in count
  field 14 frame_size_1024_in count
  field 15 frame_size_1513_in count
  field 16 frame_size_1518_in count
  field 17 frame_size_jumbo_in count
  field 18 frame_size_64_out count
  field 19 frame_size_128_out count
  field 20 frame_size_256_out count
  field 21 frame_size_512_out count
  field 22 frame_size_1024_out count
  field 23 frame_size_1513_out count
  field 24 frame_size_1518_out count
  field 25 frame_size_jumbo_out count
  field 26 frame_type_ipv4_in count
  field 27 frame_type_ipv6_in count
  field 28 frame_type_arp_in count
  field 29 frame_type_ipx_in count
  field 30 frame_type_mpls_in count
  field 31 frame_type_lacp_in count
  field 32 frame_type_stp_in count
  field 33 frame_type_ieee8021x_in count
  field 34 frame_type_other_in count
  field 35 frame_type_ipv4_out count
  field 36 frame_type_ipv6_out count
  field 37 frame_type_arp_out count
  field 38 frame_type_ipx_out count
  field 39 frame_type_mpls_out count
  field 40 frame_type_lacp_out count
  field 41 frame_type_stp_out count
  field 42 frame_type_ieee8021x_out count
  field 43 frame_type_other_out count
  field 44 vlan_tagged_in count
```



```
field 45 vlan_tagged_out count
field 46 ip_type_pkts_in topn_count
field 47 ip_type_pkts_out topn_count
field 48 ip_type_bytes_in topn_count
field 49 ip_type_bytes_out topn_count
field 50 ip_fragments_in count
field 51 ip_fragments_out count
field 52 mcast_bytes topn_count
field 53 mcast_pkts topn_count
field 54 icmp_type_in topn_count
field 55 icmp_type_out topn_count
field 56 icmpv6_type_in topn_count
field 57 icmpv6_type_out topn_count
```

extrahop.device.net_detail device

```
field 0 pkts_in topn_count
field 1 pkts_out topn_count
field 2 bytes_in topn_count
field 3 bytes_out topn_count
field 4 ip_type_pkts_in topn_tset
field 5 ip_type_pkts_out topn_tset
field 6 ip_type_bytes_in topn_tset
field 7 ip_type_bytes_out topn_tset
field 8 ip_fragments_in topn_count
field 9 ip_fragments_out topn_count
field 10 icmp_type_in topn_tset
field 11 icmp_type_out topn_tset
field 12 icmpv6_type_in topn_tset
field 13 icmpv6_type_out topn_tset
```

extrahop.device.app device

```
field 0 bytes_in topn_count
field 1 bytes_out topn_count
field 2 pkts_in topn_count
field 3 pkts_out topn_count
```

extrahop.device.app_detail device

```
field 0 bytes_in topn_tset
field 1 bytes_out topn_tset
field 2 pkts_in topn_tset
field 3 pkts_out topn_tset
```

extrahop.device.app_turn device

```
field 0 turns topn_count
field 1 req_xfer topn_dset
field 2 tprocess topn_dset
field 3 rsp_xfer topn_dset
field 4 req_size topn_dset
field 5 rsp_size topn_dset
```

L4 TCP

Device TCP Metrics

```
extrahop.device.tcp device
  field 0 accepted count
  field 1 connected count
  field 2 closed count
  field 3 established snap
  field 4 expired count
  field 5 desync count
  field 6 aborted_in count
  field 7 reset_in count
  field 8 syn_in count
  field 9 stray_in count
  field 10 drop_in count
  field 11 zwnd_in count
  field 12 rcv_throttle_in count
  field 13 snd_throttle_in count
  field 14 no_tstamp_in count
  field 15 no_sack_in count
  field 16 rto_in count
  field 17 syndrop_in count
  field 18 ccbad_in count
  field 19 aborted_out count
  field 20 reset_out count
  field 21 syn_out count
  field 22 drop_out count
  field 23 tiny_out count
  field 24 nagle_out count
  field 25 zwnd_out count
  field 26 ss_out count
  field 27 rcv_throttle_out count
  field 28 snd_throttle_out count
  field 29 no_tstamp_out count
  field 30 no_sack_out count
  field 31 rto_out count
  field 32 retrans_out count
  field 33 outoforder_out count
  field 34 ccbad_out count
  field 35 rtt dataset
  field 36 bytes_in count
  field 37 bytes_out count

extrahop.device.tcp_app_detail device
  field 0 closed topn_count
  field 2 expired topn_count
  field 3 aborted_in topn_count
  field 4 aborted_out topn_count
  field 5 drop_in topn_count
  field 6 drop_out topn_count
```

```
field 7 zwnd_in topn_count
field 8 zwnd_out topn_count
field 9 rcv_throttle_in topn_count
field 10 rcv_throttle_out topn_count
field 11 snd_throttle_in topn_count
field 12 snd_throttle_out topn_count
field 13 rto_in topn_count
field 14 rto_out topn_count
field 15 syndrop_in topn_count
field 16 nagle_out topn_count
field 17 desync topn_count
```

extrahop.device.tcp_detail device

```
field 0 accepted topn_count
field 1 connected topn_count
field 2 closed topn_count
field 3 established topn_snap
field 4 expired topn_count
field 5 rtt topn_sset
field 6 aborted_in topn_count
field 7 aborted_out topn_count
field 8 drop_in topn_count
field 9 drop_out topn_count
field 10 zwnd_in topn_count
field 11 zwnd_out topn_count
field 12 rcv_throttle_in topn_count
field 13 rcv_throttle_out topn_count
field 14 snd_throttle_in topn_count
field 15 snd_throttle_out topn_count
field 16 rto_in topn_count
field 17 rto_out topn_count
field 18 syndrop_in topn_count
field 19 nagle_out topn_count
field 20 retrans_out topn_count
```

LDAP

Device LDAP Metrics

```
extrahop.device.ldap_client device  
field 0 bytes_in count  
field 1 bytes_out count  
field 2 msg_type topn_count  
field 3 plain count  
field 4 sasl count  
field 5 rsp count  
field 6 tprocess dataset  
field 7 tprocess_msgtype topn_dset  
field 8 rsp_error count  
field 10 error_msg_short topn_count  
field 11 req count
```

```
extrahop.device.ldap_client_detail device  
field 0 bytes_in topn_count  
field 1 bytes_out topn_count  
field 2 msg_type topn_tset  
field 3 plain topn_count  
field 4 sasl topn_count  
field 5 rsp topn_count  
field 6 tprocess topn_sset  
field 7 rsp_error topn_count  
field 8 error_msg_short topn_tset  
field 9 error_msg topn_count  
field 10 req topn_count
```

```
extrahop.device.ldap_server device  
field 0 bytes_in count  
field 1 bytes_out count  
field 2 msg_type topn_count  
field 3 plain count  
field 4 sasl count  
field 5 rsp count  
field 6 tprocess dataset  
field 7 tprocess_msgtype topn_dset  
field 8 rsp_error count  
field 10 error_msg_short topn_count  
field 11 req count
```

```
extrahop.device.ldap_server_detail device  
field 0 bytes_in topn_count  
field 1 bytes_out topn_count  
field 2 msg_type topn_tset  
field 3 plain topn_count  
field 4 sasl topn_count  
field 5 rsp topn_count  
field 6 tprocess topn_sset  
field 7 rsp_error topn_count
```

```
field 8 error_msg_short topn_tset  
field 9 error_msg topn_count  
field 10 req topn_count
```

Memcache

Device Memcache Metrics

extrahop.device.memcache_client device

```
field 0 req count
field 1 rsp count
field 2 noreply count
field 3 hit count
field 4 miss count
field 5 rsp_error count
field 6 method topn_count
field 7 status_code topn_count
field 8 access_time dataset
field 9 req_size dataset
field 10 rsp_size dataset
```

extrahop.device.memcache_client_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 noreply topn_count
field 3 hit topn_count
field 4 miss topn_count
field 5 rsp_error topn_count
field 6 method topn_tset
field 7 status_code topn_tset
field 8 access_time topn_sset
field 9 req_size topn_sset
field 10 rsp_size topn_sset
field 11 error_msg topn_count
```

extrahop.device.memcache_server device

```
field 0 req count
field 1 rsp count
field 2 noreply count
field 3 hit count
field 4 miss count
field 5 rsp_error count
field 6 method topn_count
field 7 status_code topn_count
field 8 access_time dataset
field 9 req_size dataset
field 10 rsp_size dataset
```

extrahop.device.memcache_server_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 noreply topn_count
field 3 hit topn_count
field 4 miss topn_count
field 5 rsp_error topn_count
```

```
field 6 method topn_tset
field 7 status_code topn_tset
field 8 access_time topn_sset
field 9 req_size topn_sset
field 10 rsp_size topn_sset
field 11 error_msg topn_count
```

MS-RPC

Device MS-RPC Metrics

extrahop.device.rpc_client device

```
field 0 bytes_in count
field 1 bytes_out count
field 2 pkts_in count
field 3 pkts_out count
field 4 naks count
field 5 failed_epm_binds count
field 6 orphaned count
field 7 fault count
field 8 cancel count
field 9 rsp count
field 10 rsp_interface topn_count
field 11 bytes_in_interface topn_count
field 12 bytes_out_interface topn_count
field 13 pkts_in_interface topn_count
field 14 pkts_out_interface topn_count
field 15 auth_used topn_count
field 16 frag_len dataset
field 17 frag_rsp count
```

extrahop.device.rpc_client_detail device

```
field 0 bytes_in topn_count
field 1 bytes_out topn_count
field 2 pkts_in topn_count
field 3 pkts_out topn_count
field 4 naks topn_count
field 5 failed_epm_binds topn_count
field 6 orphaned topn_count
field 7 fault topn_count
field 8 cancel topn_count
field 9 rsp topn_count
field 10 auth_used topn_tset
field 11 frag_len topn_sset
field 12 frag_rsp topn_count
```

extrahop.device.rpc_server device

```
field 0 bytes_in count
field 1 bytes_out count
field 2 pkts_in count
field 3 pkts_out count
field 4 naks count
field 5 failed_epm_binds count
field 6 orphaned count
field 7 fault count
field 8 cancel count
field 9 rsp count
field 10 rsp_interface topn_count
```



```
field 11 bytes_in_interface topn_count
field 12 bytes_out_interface topn_count
field 13 pkts_in_interface topn_count
field 14 pkts_out_interface topn_count
field 15 auth_used topn_count
field 16 frag_len dataset
field 17 frag_rsp count
```

extrahop.device.rpc_server_detail device

```
field 0 bytes_in topn_count
field 1 bytes_out topn_count
field 2 pkts_in topn_count
field 3 pkts_out topn_count
field 4 naks topn_count
field 5 failed_epm_binds topn_count
field 6 orphaned topn_count
field 7 fault topn_count
field 8 cancel topn_count
field 9 rsp topn_count
field 10 auth_used topn_tset
field 11 frag_len topn_sset
field 12 frag_rsp topn_count
```

NFS

Device NFS Metrics

extrahop.device.nfs_client device

```
field 0 rsp count
field 1 req count
field 2 rsp_error count
field 3 method topn_count
field 4 status topn_count
field 5 tprocess dataset
field 6 auth topn_count
field 7 auth_error topn_count
field 8 rpc_status topn_count
field 9 bytes_read count
field 10 bytes_write count
field 11 req_xfer dataset
field 12 rsp_xfer dataset
field 13 req_size dataset
field 14 rsp_size dataset
field 15 req_write count
field 16 req_read count
field 17 tcp count
field 18 udp count
field 19 retries count
field 20 aborts count
field 21 bytes_fsinfo count
field 22 version topn_count
```

extrahop.device.nfs_client_detail device

```
field 0 rsp topn_count
field 1 req topn_count
field 2 rsp_error topn_count
field 3 method topn_tset
field 4 status topn_tset
field 5 auth topn_tset
field 6 auth_error topn_tset
field 7 rpc_status topn_tset
field 8 file_bytes_in topn_count
field 9 file_bytes_out topn_count
field 10 file_count topn_count
field 11 method_bytes_in topn_count
field 12 method_bytes_out topn_count
field 13 method_count topn_count
field 14 error_count topn_count
field 15 rsp_size topn_sset
field 16 req_size topn_sset
field 17 req_write topn_count
field 18 req_read topn_count
field 19 tcp topn_count
field 20 udp topn_count
```

```
field 21 retries topn_count
field 22 aborts topn_count
field 23 file_info_access_time topn_sset
field 24 fsinfo_bytes topn_count
field 25 user_bytes_in topn_count
field 26 user_bytes_out topn_count
field 27 user_rsp topn_count
field 28 version topn_tset
```

extrahop.device.nfs_server device

```
field 0 rsp count
field 1 req count
field 2 rsp_error count
field 3 method topn_count
field 4 status topn_count
field 5 tprocess dataset
field 6 auth topn_count
field 7 auth_error topn_count
field 8 rpc_status topn_count
field 9 bytes_read count
field 10 bytes_write count
field 11 req_xfer dataset
field 12 rsp_xfer dataset
field 13 req_size dataset
field 14 rsp_size dataset
field 15 req_write count
field 16 req_read count
field 17 tcp count
field 18 udp count
field 19 retries count
field 20 aborts count
field 21 bytes_fsinfo count
field 22 version topn_count
```

extrahop.device.nfs_server_detail device

```
field 0 rsp topn_count
field 1 req topn_count
field 2 rsp_error topn_count
field 3 method topn_tset
field 4 status topn_tset
field 5 auth topn_tset
field 6 auth_error topn_tset
field 7 rpc_status topn_tset
field 8 file_bytes_in topn_count
field 9 file_bytes_out topn_count
field 10 file_count topn_count
field 11 method_bytes_in topn_count
field 12 method_bytes_out topn_count
field 13 method_count topn_count
field 14 error_count topn_count
field 15 rsp_size topn_sset
```

```
field 16 req_size topn_sset
field 17 req_write topn_count
field 18 req_read topn_count
field 19 tcp topn_count
field 20 udp topn_count
field 21 retries topn_count
field 22 aborts topn_count
field 23 file_info_access_time topn_sset
field 24 fsinfo_bytes topn_count
field 25 user_bytes_in topn_count
field 26 user_bytes_out topn_count
field 27 user_rsp topn_count
field 28 version topn_tset
```

SMPP

Device SMPP Metrics

```
extrahop.device.smpp_client device  
field 0 req count  
field 1 rsp count  
field 2 rsp_error count  
field 3 cmd_id_in_ts topn_count  
field 4 cmd_id_out_ts topn_count  
field 5 req_xfer dataset  
field 6 tprocess dataset  
field 7 rsp_ttlb dataset  
field 8 req_size dataset  
field 9 rsp_size dataset  
field 10 rsp_xfer dataset  
field 11 status_ts topn_count  
field 12 status_desc_ts topn_count
```

```
extrahop.device.smpp_client_detail device  
field 0 req topn_count  
field 1 rsp topn_count  
field 2 rsp_error topn_count  
field 3 cmd_id_in_ts topn_tset  
field 4 cmd_id_out_ts topn_tset  
field 6 tprocess topn_sset  
field 7 rsp_ttlb topn_sset  
field 8 req_size topn_sset  
field 9 rsp_size topn_sset  
field 11 status_ts topn_tset
```

```
extrahop.device.smpp_client_mdn_detail device  
field 0 req topn_count  
field 1 rsp topn_count  
field 2 rsp_error topn_count  
field 3 cmd_id_in_ts topn_tset  
field 4 cmd_id_out_ts topn_tset  
field 6 tprocess topn_sset  
field 7 rsp_ttlb topn_sset  
field 8 req_size topn_sset  
field 9 rsp_size topn_sset  
field 11 status_ts topn_tset
```

```
extrahop.device.smpp_client_sc_detail device  
field 0 req topn_count  
field 1 rsp topn_count  
field 2 rsp_error topn_count  
field 3 cmd_id_in_ts topn_tset  
field 4 cmd_id_out_ts topn_tset  
field 6 tprocess topn_sset  
field 7 rsp_ttlb topn_sset  
field 8 req_size topn_sset
```

```
field 9 rsp_size topn_sset
field 11 status_ts topn_tset
```

extrahop.device.smpserver device

```
field 0 req count
field 1 rsp count
field 2 rsp_error count
field 3 cmd_id_in_ts topn_count
field 4 cmd_id_out_ts topn_count
field 5 req_xfer dataset
field 6 tprocess dataset
field 7 rsp_ttlb dataset
field 8 req_size dataset
field 9 rsp_size dataset
field 10 rsp_xfer dataset
field 11 status_ts topn_count
field 12 status_desc_ts topn_count
```

extrahop.device.smpserver_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
field 3 cmd_id_in_ts topn_tset
field 4 cmd_id_out_ts topn_tset
field 6 tprocess topn_sset
field 7 rsp_ttlb topn_sset
field 8 req_size topn_sset
field 9 rsp_size topn_sset
field 11 status_ts topn_tset
```

extrahop.device.smpserver_mdn_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
field 3 cmd_id_in_ts topn_tset
field 4 cmd_id_out_ts topn_tset
field 6 tprocess topn_sset
field 7 rsp_ttlb topn_sset
field 8 req_size topn_sset
field 9 rsp_size topn_sset
field 11 status_ts topn_tset
```

extrahop.device.smpserver_sc_detail device

```
field 0 req topn_count
field 1 rsp topn_count
field 2 rsp_error topn_count
field 3 cmd_id_in_ts topn_tset
field 4 cmd_id_out_ts topn_tset
field 6 tprocess topn_sset
field 7 rsp_ttlb topn_sset
field 8 req_size topn_sset
```

```
field 9  rsp_size  topn_sset  
field 11 status_ts  topn_tset
```

SMTP

Device SMTP Metrics

extrahop.device.smtp_client device

```
field 0 bytes_in count
field 1 bytes_out count
field 5 sender topn_count
field 6 rcpt topn_count
field 7 domains topn_count
field 8 rsp count
field 9 rsp_error count
field 10 req count
field 12 encrypted_sessions count
field 13 sessions count
field 14 msg_len dataset
field 17 methods topn_count
```

extrahop.device.smtp_client_detail device

```
field 0 bytes_in topn_count
field 1 bytes_out topn_count
field 5 rsp topn_count
field 6 rsp_error topn_count
field 7 req topn_count
field 8 error topn_tset
field 9 encrypted_sessions topn_count
field 10 sessions topn_count
field 11 msg_len topn_sset
field 12 methods topn_tset
field 13 bytes_domain topn_count
field 14 bytes_sender topn_count
field 15 bytes_rcpt topn_count
field 16 msg_len_sender topn_sset
field 17 msg_len_rcpt topn_sset
field 18 errors topn_count
```

extrahop.device.smtp_server device

```
field 0 bytes_in count
field 1 bytes_out count
field 5 sender topn_count
field 6 rcpt topn_count
field 7 domains topn_count
field 8 rsp count
field 9 rsp_error count
field 10 req count
field 12 encrypted_sessions count
field 13 sessions count
field 14 msg_len dataset
field 17 methods topn_count
```

extrahop.device.smtp_server_detail device

```
field 0 bytes_in topn_count
```



```
field 1 bytes_out topn_count
field 5 rsp topn_count
field 6 rsp_error topn_count
field 7 req topn_count
field 8 error topn_tset
field 9 encrypted_sessions topn_count
field 10 sessions topn_count
field 11 msg_len topn_sset
field 12 methods topn_tset
field 13 bytes_domain topn_count
field 14 bytes_sender topn_count
field 15 bytes_rcpt topn_count
field 16 msg_len_sender topn_sset
field 17 msg_len_rcpt topn_sset
field 18 errors topn_count
```

SSL

Device SSL Metrics

extrahop.device.ssl_client device

```
field 0 version topn_count
field 1 content_type topn_count
field 2 record_size dataset
field 3 decrypted count
field 5 numberofv2hello count
field 6 compressed count
field 7 cipher topn_count
field 8 connected count
field 9 aborted count
field 10 renegotiated count
field 11 resumed count
field 12 cert_malformed count
field 15 alerts topn_count
field 16 keysize topn_count
```

extrahop.device.ssl_client_detail device

```
field 0 version topn_tset
field 1 content_type topn_tset
field 3 decrypted topn_count
field 5 numberofv2hello topn_count
field 6 compressed topn_count
field 7 cipher topn_tset
field 8 connected topn_count
field 9 aborted topn_count
field 10 renegotiated topn_count
field 11 resumed topn_count
field 12 alerts topn_tset
field 13 cert_subject topn_count
field 14 cert_expiration topn_time
field 15 cert_subject_bytes_in topn_count
field 16 cert_subject_bytes_out topn_count
field 17 keysize topn_tset
```

extrahop.device.ssl_server device

```
field 0 version topn_count
field 1 content_type topn_count
field 2 record_size dataset
field 3 decrypted count
field 5 numberofv2hello count
field 6 compressed count
field 7 cipher topn_count
field 8 connected count
field 9 aborted count
field 10 renegotiated count
field 11 resumed count
field 12 cert_malformed count
```

```
field 15 alerts topn_count
field 16 keysize topn_count
```

extrahop.device.ssl_server_detail device

```
field 0 version topn_tset
field 1 content_type topn_tset
field 3 decrypted topn_count
field 5 numberofv2hello topn_count
field 6 compressed topn_count
field 7 cipher topn_tset
field 8 connected topn_count
field 9 aborted topn_count
field 10 renegotiated topn_count
field 11 resumed topn_count
field 12 alerts topn_tset
field 13 cert_subject topn_count
field 14 cert_expiration topn_time
field 15 cert_subject_bytes_in topn_count
field 16 cert_subject_bytes_out topn_count
field 17 keysize topn_tset
```

Capture Metrics

In this API, capture metrics refer to network metrics.

extrahop.capture.net capture

```
field 0 pkts count
field 1 bytes count
field 2 frame_cast_unicast_pkts count
field 3 frame_cast_unicast_bytes count
field 4 frame_cast_multicast_pkts count
field 5 frame_cast_multicast_bytes count
field 6 frame_cast_broadcast_pkts count
field 7 frame_cast_broadcast_bytes count
field 8 frame_size_64 count
field 9 frame_size_128 count
field 10 frame_size_256 count
field 11 frame_size_512 count
field 12 frame_size_1024 count
field 13 frame_size_1513 count
field 14 frame_size_1518 count
field 15 frame_size_jumbo count
field 16 frame_type_ipv4 count
field 17 frame_type_ipv6 count
field 18 frame_type_arp count
field 19 frame_type_ipx count
field 20 frame_type_mpls count
field 21 frame_type_lacp count
field 22 frame_type_stp count
field 23 frame_type_ieee8021x count
field 24 frame_type_other count
field 25 vlan_tagged count
field 26 ip_type_pkts topn_count
field 27 ip_type_bytes topn_count
field 28 ip_fragments count
field 29 mcast_bytes topn_count
field 30 mcast_pkts topn_count
```

extrahop.vlan.app vlan

```
field 0 bytes topn_count
field 1 pkts topn_count
field 2 turns topn_count
```

extrahop.vlan.net vlan

```
field 0 pkts count
field 1 bytes count
field 2 frame_cast_unicast_pkts count
field 3 frame_cast_unicast_bytes count
field 4 frame_cast_multicast_pkts count
field 5 frame_cast_multicast_bytes count
field 6 frame_cast_broadcast_pkts count
field 7 frame_cast_broadcast_bytes count
```

```
field 8 frame_size_64 count
field 9 frame_size_128 count
field 10 frame_size_256 count
field 11 frame_size_512 count
field 12 frame_size_1024 count
field 13 frame_size_1513 count
field 14 frame_size_1518 count
field 15 frame_size_jumbo count
field 16 frame_type_ipv4 count
field 17 frame_type_ipv6 count
field 18 frame_type_arp count
field 19 frame_type_ipx count
field 20 frame_type_mpls count
field 21 frame_type_lacp count
field 22 frame_type_stp count
field 23 frame_type_ieee8021x count
field 24 frame_type_other count
field 25 ip_type_pkts topn_count
field 26 ip_type_bytes topn_count
field 27 ip_fragments count
field 28 mcast_bytes topn_count
field 29 mcast_pkts topn_count
```

extrahop.capture.tcp capture

```
field 0 desync count
```

extrahop.capture.app capture

```
field 0 bytes topn_count
field 1 pkts topn_count
field 2 turns topn_count
```

extrahop.capture.aaa capture

```
field 0 rsp count
field 1 rsp_error count
```

extrahop.capture.amf capture

```
field 0 req count
field 1 rsp count
field 2 rsp_error count
```

extrahop.capture.cifs capture

```
field 0 bytes_read count
field 1 bytes_write count
field 2 fsinfo_bytes count
```

extrahop.capture.db capture

```
field 0 req count
field 1 req_procedure count
field 2 rsp count
field 3 rsp_error count
```

extrahop.capture.dns capture

```
field 0 rsp count
```

```
field 1 rsp_error count

extrahop.capture.fix capture
field 0 rsp count
field 1 rsp_error count

extrahop.capture.ftp capture
field 0 rsp count
field 1 rsp_error count

extrahop.capture.http capture
field 0 req count
field 1 rsp count
field 2 rsp_error count

extrahop.capture.ibmmq capture
field 0 rsp count
field 1 rsp_error count

extrahop.capture.ica capture
field 0 client_msg count
field 1 server_msg count
field 2 launches count
field 3 aborted count

extrahop.capture.iscsi capture
field 0 rsp count
field 1 rsp_error count
field 2 bytes_read count
field 3 bytes_write count

extrahop.capture.ldap capture
field 0 rsp count
field 1 rsp_error count

extrahop.capture.memcache capture
field 0 req count
field 1 rsp count
field 2 rsp_error count

extrahop.capture.nfs capture
field 0 rsp count
field 1 rsp_error count
field 2 bytes_read count
field 3 bytes_write count
field 4 bytes_fsinfo count

extrahop.capture.smpp capture
field 0 rsp count
field 1 rsp_error count

extrahop.capture.smtp capture
field 0 rsp count
field 1 rsp_error count
```

Custom Metrics

Custom Metrics

```
extrahop.capture.custom capture
  field 0 custom_count topn_count
  field 1 custom_snap topn_snap
  field 2 custom_dset topn_dset
  field 3 custom_sset topn_sset
  field 4 custom_max topn_max

extrahop.capture.custom_detail capture
  field 0 custom_count topn_tset
  field 1 custom_snap topn_tset
  field 2 custom_dset topn_tset
  field 3 custom_sset topn_tset
  field 4 custom_max topn_tset

extrahop.application.custom application
  field 0 custom_count topn_count
  field 1 custom_snap topn_snap
  field 2 custom_dset topn_dset
  field 3 custom_sset topn_sset
  field 4 custom_max topn_max

extrahop.application.custom_detail application
  field 0 custom_count topn_tset
  field 1 custom_snap topn_tset
  field 2 custom_dset topn_tset
  field 3 custom_sset topn_tset
  field 4 custom_max topn_tset

extrahop.device.custom device
  field 0 custom_count topn_count
  field 1 custom_snap topn_snap
  field 2 custom_dset topn_dset
  field 3 custom_sset topn_sset
  field 4 custom_max topn_max

extrahop.device.custom_detail device
  field 0 custom_count topn_tset
  field 1 custom_snap topn_tset
  field 2 custom_dset topn_tset
  field 3 custom_sset topn_tset
  field 4 custom_max topn_tset

extrahop.capture.custom_debug_detail capture
  field 0 runtime_log topn_tset
  field 1 exec_time topn_sset
```

Health Metrics

Health Metrics

Packets

extrahop.capture.net capture

field 0 pkts count

Throughput

14 extrahop.capture.net capture

field 1 bytes count

TCP Desyncs

16 extrahop.capture.tcp capture

field 0 desync count

Trigger Executes

extrahop.capture.net capture

field 34 trigger_executes count

Total Trigger Cycles

extrahop.capture.net capture

field 35 trigger_cycles count

Capture Heap Allocation

extrahop.capture.net capture

field 33 heap_alloc snap

Trigger Heap Allocation

extrahop.capture.net capture

field 46 trigger_heap_alloc snap

External Timestamps

extrahop.capture.net capture

field 36 timestamped count

RPCAP Health

extrahop.capture.net capture

field 37 pktcap_bytes_written count

field 38 rpcap_encap_pkts topn_count


```
field 39 rpcap_encap_bytes topn_count
field 40 rpcapd_ifrecv topn_count
field 41 rpcapd_ifdrop topn_count
field 42 rpcapd_krnldrop topn_count
field 43 rpcapd_snd_tunnel_pkts topn_count
field 44 rpcap_rcv_tunnel_pkts topn_count
field 45 rpcap_rcv_tunnel_bytes topn_count
```

SSL Decryption Health

extrahop.capture.ssl capture

```
field 0 decrypted topn_count
field 1 unsupported topn_count
field 2 passthru topn_count
field 3 detached topn_count
```

Disk Read Throughput

extrahop.system.bridge capture

```
field 1 block_read_bytes count
```

Disk Write Throughput

extrahop.system.bridge capture

```
field 1 block_read_bytes count
field 2 block_write_bytes_fast count
```

Store Read Throughput

extrahop.system.bridge capture

```
field 16 store_read_bytes count
```

Store Write Throughput

extrahop.system.bridge capture

```
field 3 store_add_bytes_slow count
field 4 store_add_bytes_medium count
field 5 store_add_bytes_fast count
```

Working Set Size

extrahop.system.bridge capture

field 9 store_working_set_slow snap

field 10 store_working_set_medium snap

field 11 store_working_set_fast snap

Metric Size**extrahop.system.bridge capture**

field 17 metric_size dataset

field 18 metric_maxsize topn_max

Active Devices**extrahop.system.bridge capture**

field 13 l2_device_active snap

field 15 l3_device_active snap

Total Devices**extrahop.system.bridge capture**

field 12 l2_device snap

field 14 l3_device snap

Store Lookback**extrahop.system.bridge capture**

field 6 store_add_slow count

field 7 store_add_medium count

field 8 store_add_fast count

field 2 block_write_bytes_fast count

field 19 block_write_bytes_medium count

field 20 block_write_bytes_slow count

Datastore Heap Allocation**extrahop.system.bridge capture**

field 0 heap_alloc snap

Datatypes

The following datatypes are available in the ExtraHop datastore.

- **count**
- **dataset**
- **snap**
- **string**
- **time**
- **topn_count**
- **topn_dset**
- **topn_snap**
- **topn_sset**
- **topn_time**
- **topn_tset**

count

64-bit integer datatype.

Sample Metrics:

- Number of packets in the network capture.
- Number of requests to the HTTP server.
- Number of errors on the database server.

Sample Usages:

Can be queried with the following methods:

get_exstats

Sample Usage:

```
result = c.get_exstats ("extrahop.capture.net", "capture", [(capture.oid, -1800000, 0)], ["pkts","bytes"], options)
for stat in result.stats:
    print time.ctime(long(stat.time) / 1000), stat.pkts, stat.bytes
```

Sample Result:

```
Thu Jan  1 00:00:01 2009 17 1905
Thu Jan  1 00:00:03 2009 23 2726
Thu Jan  1 00:00:07 2009 59 7355
Thu Jan  1 00:00:14 2009 58 8602
```

get_exstats_total

Sample Usage:

```
# get total pkts_in, bytes_in for activity group 'TCP'
print ">>> totals <<<"
result = c.get_exstats_total ("extrahop.device.net", "activity_group",
                              [('extrahop.device.tcp', -1800000, 0)], ["pkts_in", "bytes_in"], options)
```

Sample Result:

```
Total:  Thu Jan  1 00:25:24 2009 1173752 571051328.0
```

get_exstats_group

Sample Usage:

```
# get pkts_in, bytes_in for an activity group 'TCP'
result = c.get_exstats_group ("extrahop.device.net", "activity_group",
                              [("extrahop.device.tcp", -1800000, 0)], ["pkts_in","bytes_in"],
                              options)
```

Sample Results:

```
time: Thu Jan 1 00:25:24 2009
device: 4
bytes_in 478
pkts_in 6
time: Thu Jan 1 00:25:24 2009
device: 17
bytes_in 3062760
pkts_in 2461
```

dataset

A frequency table. For each entry, freq is the number of times value has been seen (64-bit integer).

Sample Metrics:

- HTTP server request transfer time.
- HTTP server processing time.
- HTTP server response transfer time.

Field Options :

The following options can be used with any of the methods below:

- `summary:field_name - 5-number summary.`
- `summary595:field_name - 5th and 95th percentile.`
- `mean:field_name - mathematical mean.`

Sample Usages:

get_exstats

Sample Usage:

```
result = c.get_exstats ("extrahop.device.http_server", "device", [(10, -
1800000, 0)], ["tprocess"], {'cycle':"fast"})
print result
```

Sample Raw Results:

```
{'abs_from': 1230798324420.0
'abs_until': 1230798324420.0,
'cycle': u'fast',
'stats':
  [{'stat_key': '', 'oid': 10, 'time': 1230797172000.0,
'tprocess':
  [{'freq': 1, 'value': 0},
  {'freq': 1, 'value': 64},
  {'freq': 1, 'value': 75}]},
{'stat_key': '', 'oid': 10, 'time': 1230797202000.0,
'tprocess':
  [{'freq': 8, 'value': 0},
  {'freq': 14, 'value': 1},
  {'freq': 1, 'value': 20},
  {'freq': 1, 'value': 21},
  {'freq': 1, 'value': 27},
  {'freq': 1, 'value': 31}]},
...
}
```

Sample Pretty-Printed Results:

```

cycle      : fast
abs from   : Thu Jan  1 00:15:24 2009 [1230797724420]
abs until  : Thu Jan  1 00:25:24 2009 [1230798324420]
=====
time       : Thu Jan  1 00:15:42 2009 [1230797742000]
object id  : 10
stat key   :
--- tprocess -----
freq 1, value 0
freq 1, value 64
freq 1, value 75
=====
time       : Thu Jan  1 00:16:12 2009 [1230797772000]
object id  : 10
stat key   :
--- tprocess -----
freq 8, value 0
freq 14, value 1
freq 1, value 20
freq 1, value 21
freq 1, value 27
freq 1, value 31

```

get_exstats_total

Sample Usage:

```

result = c.get_exstats ("extrahop.device.http_server", "device", [(10, -
1800000, 0)], ["tprocess"], {'cycle':"fast"})

```

Sample Raw Results:

```

{'abs_from': 1230796524000.0,
 'abs_until': 1230798324000.0,
 'cycle': u'fast',
 'stats': [
   {'stat_key': None,
    'oid': -1,
    'time': 1230798324000.0,
    'tprocess': [
      {'freq': 3, 'value': 10},
      {'freq': 3, 'value': 11},
      {'freq': 7, 'value': 12},
      {'freq': 3, 'value': 13},
      ...
    ]
  ]
}

```

Sample Pretty-Printed Results:

```

cycle      : fast
abs from   : Sun Feb 13 16:14:07 2011 [1297642447292]
abs until  : Sun Feb 13 16:44:07 2011 [1297644247292]
=====
time       : Sun Feb 13 16:43:41 2011 [1297644221292]
object id  : -1
stat key   :
--- tprocess -----
freq 2, value 4
freq 1, value 5
freq 1, value 10
freq 1, value 11
...

```

get_exstats_group

Sample Usage:

```

result = c.get_exstats_group ("extrahop.device.http_server", "activity_
group", [{"extrahop.device.http_server", -1800000, 0}], ["tprocess"],
{'cycle':"fast"})

```

Sample Raw Results:

```

{'abs_from': 1230796524000.0,
 'abs_until': 1230798324000.0,
 'cycle': u'fast',
 'stats': [{'stat_key': None, 'oid': 6,
'tprocess': [{'freq': 327, 'value': 0}, {'freq': 90, 'value': 1}, {'freq':
1, 'value': 2}], 'time': 1230798324000.0},
 {'stat_key': None, 'oid': 29,
 'tprocess': [{'freq': 22, 'value': 0}, {'freq': 23, 'value': 1}, {'freq':
7, 'value': 2}, ], 'time': 1230798324000.0}
...
}

```

"summary" metric option

Sample Usage:

```

# get a 5-number summary for device HTTP server processing time for device
with ID 10
result = c.get_exstats_total ("extrahop.device.http_server", "device", [(10,
-1800000, 0)], [":summary:tprocess"], options)

```

Sample Raw Results

```

{'abs_from': 1230796524000.0,
 'abs_until': 1230798324000.0,
 'cycle': u'fast',

```



```
'stats': [{'stat_key': None, 'oid': -1, 'time': 1230798324000.0,
':summary:tprocess': {'q1': 52.0, 'q3': 351.0, 'q2': 107.0, 'minimum': 0.0,
'maximum': 11680.0}}]
}
```

"summary595" metric option

Sample Usage:

```
# get a 5-number summary for device HTTP server processing time for device
with ID 10
result = c.get_exstats_total ("extrahop.device.http_server", "device", [(10,
-1800000, 0)], [":summary595:tprocess"], {'cycle':"fast"})
```

Sample Raw Results:

```
{'abs_from': 1230796524000.0,
'abs_until': 1230798324000.0,
'cycle': u'fast',
'stats': [{'stat_key': None, 'oid': -1, 'time': 1230798324000.0,
':summary595:tprocess': {'percentile5': 18.0, 'percentile95': 1230.0}}]
}
```

snap

64-bit integer datatype. The only difference in behavior when queried via `getExStatsTotal` function.

Sample Metrics:

TCP established connections (`extrahop.device.tcp -> established`).

Sample Usages:

get_exstats

Sample Usage:

```
# get metrics
result = c.get_exstats ("extrahop.device.tcp",
                        "device",
                        [(0, -1800000, 0)],
                        ["established"],
                        {'cycle':"fast"})

# pretty-print results
for stat in result.stats:
    print time.ctime(long(stat.time) / 1000), stat.established
```

Sample Results:

```
Thu Jan  1 00:00:01 2009 2
Thu Jan  1 00:00:02 2009 3
Thu Jan  1 00:00:05 2009 7
Thu Jan  1 00:00:11 2009 0
```

get_exstats_total

Note: For snap metrics, the `exStatsTotal` method returns the last value in the time period. For example, the total number of established TCP connections at the end of a time period will be equal to the last measurement for that metric in the selected time period.

Sample Usage:

```
# get metrics
result = c.get_exstats_total ( "extrahop.device.tcp",
                               "device",
                               [(0, -1800000, 0)],
                               ["established"],
                               {'cycle':"fast"})

# pretty-print results
for stat in result.stats:
    print "Total: ", time.ctime(long(stat.time) / 1000), stat.established
```

Sample Results:

```
Total:  Thu Jan  1 00:25:24 2009 0
```

get_exstats_group

Sample Usage:

```
result = c.get_exstats_group ( "extrahop.device.tcp",  
                              "activity_group",  
                              [{"extrahop.device.tcp", -1800000, 0}],  
                              ["established"],  
                              {'cycle':"fast"})
```

Sample Results

```
time: Thu Jan 1 00:00:01 2009  
device: 4  
established 0  
time: Thu Jan 1 00:17:41 2009  
device: 17  
established 1  
time: Thu Jan 1 00:10:26 2009  
device: 30  
established 0
```

string

UTF-8 string datatype.

Sample Metrics:

- Does not exist as a top-level metric.
- Only exists inside sets as a key. For example, CIFS/NFS file names are strings. (For details, refer to `topn_count`.)

time

Integer datatype representing a UTC time stamp.

Sample Metrics

- Does not exist as a standalone type.
- Inside any returned result, signifies time during which metrics were recorded.

topn_count

Set of *count* datatypes, a set of 64-bit integers, keyed by a structure that can be anything: string, IP address, file name, etc.

Sample Metrics:

- Number of times each file on a CIFS server was accessed.
- Number of times each URI on an HTTP server was accessed.
- Number of requests by status code on an HTTP server.
- Number of times each IP accessed a CIFS server.
- Number of times each IP accessed an HTTP server.

Sample Usages:

get_exstats

Sample Usage:

```
result = c.get_exstats("extrahop.device.cifs_server_detail",
                      "device",
                      [(22, -1800000, 0)],
                      ["file_info"],
                      {'cycle':"fast"})
print result
```

Sample Raw Result:

```
{'abs_from': 1230796524420.0,
 'abs_until': 1230798324420.0,
 'cycle': u'fast',
 'stats':
  [{ 'stat_key': '',
    'oid': 22,
    'time': 1230797563000.0},
  'file_info':
  [{'vtype': u'count',
   'key': {'key_type': u'string',
           'str': u'\\\\\\TITANIUM\\SHARE\\tools\\captures\\ldap.cap'},
   'value': 4}]
  ]},
 ...
```

Sample Usage:

```
result = send_exstats_request(stat_name="extrahop.device.cifs_server_
detail",
                             object_type="device",
                             object_spec=[(22, -1800000, 0)],
                             field_spec=["req_fsinfo"])
print result
```

Sample Results:

```
{'abs_from': 1230796524420.0,
'abs_until': 1230798324420.0,
'cycle': u'fast',
'stats':
  [{'req_fsinfo':
    [{'vtype': u'count', 'stat_key': '', 'oid': 22, 'time': 1230797503000.0,
'key': {'key_type': u'ipaddr', 'host': u'database.extrahop.net', 'device_
oid': 29, 'addr': 10.10.248.29},
'value': 1}}],
{'req_fsinfo':
  [{'vtype': u'count', 'stat_key': '', 'oid': 22, 'time': 1230797533000.0,
'key': {'key_type': u'ipaddr', 'host': u'database.extrahop.net', 'device_
oid': 29, 'addr': 10.10.248.29},
'value': 1}}],
....
}
```

get_exstats_total

Sample Usage:

```
result = c.get_exstats_total("extrahop.device.cifs_server_detail",
                             "device",
                             [(22, -1800000, 0)],
                             ["req_fsinfo"],
                             {'cycle':"fast"})
print result
```

Sample Raw Results:

```
{'abs_from': 1230796524420.0,
'abs_until': 1230798324420.0,
'cycle': u'fast',
'stats':
  [{'stat_key': '',
'oid': -1,
'time': 1230797803000.0,
'req_fsinfo':
  [{'vtype': u'count',
'key':
  {'key_type': u'ipaddr',
'host': u'database.extrahop.net',
'device_oid': 29,
'addr': 10.10.248.29},
'value': 68}]
}]
}
```

get_exstats_group

Sample Usage:

```
# get rsp, status_code for an activity group 'HTTP Client'
print ">>> ExStatsGroup <<<"
result = send_exstatsgroup_request(stat_name="extrahop.device.http_client",
                                   object_type="activity_group",
                                   object_spec=[("extrahop.device.http_client", -1800000,
0)],
                                   field_spec=["rsp","status_code"])

#print raw result
print result

# pretty-print result
for stat in result.stats:
    print "time:", time.ctime(long(stat.time) / 1000)
    print "device:", stat.oid
    print "rsp count", stat.rsp
    print "status codes:"
    for x in stat.status_code:
        print "\t", x.key.str, x.value
```

Sample Raw Result:

```
{'abs_from': 1230796524000.0,
'abs_until': 1230798324000.0,
'cycle': u'fast',
'stats': [
{'stat_key': None,
'oid': 1,
'time': 1230798324000.0,
'rsp': 80,
'status_code':
[{'vtype': u'count', 'key': {'key_type': u'string', 'str': u'200'},
'value': 78},
{'vtype': u'count', 'key': {'key_type': u'string', 'str': u'404'},
'value': 2}]
},
{'stat_key': None,
'oid': 3,
'time': 1230798324000.0,
'rsp': 22278,
'status_code':
[{'vtype': u'count', 'key': {'key_type': u'string', 'str': u'302'},
'value': 2483},
{'vtype': u'count', 'key': {'key_type': u'string', 'str': u'404'},
'value': 16947},
{'vtype': u'count', 'key': {'key_type': u'string', 'str': u'200'},
'value': 2802},
{'vtype': u'count', 'key': {'key_type': u'string', 'str': u'304'},
'value': 44},
{'vtype': u'count', 'key': {'key_type': u'string', 'str': u'500'},
'value': 2}],
},
],
}
```



```
...  
}
```

Sample Pretty-Printed Result:

```
time: Thu Jan  1 00:25:24 2009  
device: 1  
rsp count 80  
status codes:  
  200 78  
  404 2  
time: Thu Jan  1 00:25:24 2009  
device: 3  
rsp count 22278  
status codes:  
  302 2483  
  404 16947  
  200 2802  
  304 44  
  500 2
```

topn_dset

Set of datasets.

Sample Metrics:

Application page per-protocol processing times.

Field Options

The following options can be used with any of the methods below:

- `:summary:field_name` - 5-number summary.
- `:summary595:field_name` - 5th and 95th percentile.
- `:mean:field_name` - mathematical mean.

Sample Usages:

get_exstats

Sample Usage:

```
result = c.get_exstats ("extrahop.device.app",
                        "device",
                        [(0, -1800000, 0)],
                        ["tprocess"],
                        {'cycle':"fast"})
print result
```

Sample Raw Result:

```
{'abs_from': 1230798324420.0,
 'abs_until': 1230798324420.0,
 'cycle': u'fast',
 'stats':
  [{'stat_key': '', 'oid': 10, 'time': 1230796839000.0},
   'tprocess':
    [{'vtype': u'dset',
     'key': {'key_type': u'string', 'str': u'SSH'},
     'value': [{'freq': 22, 'value': 0},
               {'freq': 6, 'value': 1},
               {'freq': 2, 'value': 2},
               {'freq': 1, 'value': 3}]}],
    {'vtype': u'dset',
     'key': {'key_type': u'string', 'str': u'DNS'},
     'value': [{'freq': 1, 'value': 1}]}],
   {'stat_key': '', 'oid': 10, 'time': 1230796869000.0}
  'tprocess':
    [{'vtype': u'dset',
     'key': {'key_type': u'string', 'str': u'SSH'},
     'value': [{'freq': 55, 'value': 0},
               {'freq': 4, 'value': 1}, ...
    ]}]
}
```

Sample Pretty-Printed Result:

```

=====
time      : Thu Jan  1 00:19:09 2009 [1230797949000]
object id : 10
stat key  :
--- tprocess -----
key: MySQL
value type: dset
freq 85, value 0
freq 299, value 1
freq 262, value 2
freq 7, value 3
key: HTTP
value type: dset
freq 1, value 2138
=====

```

```

=====
time      : Thu Jan  1 00:19:39 2009 [1230797979000]
object id : 10
stat key  :
--- tprocess -----
key: MySQL
value type: dset
freq 61, value 0
freq 275, value 1
freq 198, value 2
freq 5, value 3
freq 1, value 4
key: HTTP
value type: dset
freq 1, value 1621
=====

```

get_exstats_group

Sample Usage:

```

result = c.get_exstats_group ("extrahop.device.app",
                             "activity_group",
                             [(extrahop.device.tcp, -1800000, 0)],
                             ["tprocess"],
                             options)

```

Sample Raw Result:

```

{'abs_from': 1230796524000.0,
 'abs_until': 1230798324000.0,
 'cycle': u'fast',
 'stats': [
   {'stat_key': '', 'oid': 17, 'time': 1230798306000.0,
    'tprocess': [{'vtype': u'dset',
                  'key': {'key_type': u'string', 'str': u'NFS'},
                  'value': [{'freq': 134, 'value': 0},
                             {'freq': 107, 'value': 1}]}]}]

```

```

}}
}},
  {'stat_key': '', 'oid': 22, 'time': 1230798300000.0,
   'tprocess': [{'vtype': u'dset',
                  'key': {'key_type': u'string', 'str': u'SSH'},
                  'value': [{'freq': 9, 'value': 0},
                             {'freq': 16, 'value': 1},
                             {'freq': 2, 'value': 2}]}]},
  {'vtype': u'dset',
   'key': {'key_type': u'string', 'str': u'DNS'},
   'value': [{'freq': 4, 'value': 0}]}]},
...
}

```

"summary" metric option

Sample Usage:

```

# get a 5-number summary for device App server processing time for device
with ID 10
result = c.get_exstats ("extrahop.device.app",
                       "device",
                       [(10, -1800000, 0)],
                       [":summary:tprocess"],
                       {'cycle':"fast"})

```

Sample Raw Result:

```

{'abs_from': 1230796524000.0,
 'abs_until': 1230798324000.0,
 'cycle': u'fast',
 'stats':
  [{'stat_key': '', 'oid': 10, 'time': 1230796826000.0,
   ':summary:tprocess':
    [{'vtype': u'dset', 'key': {'key_type': u'string', 'str': u'SSH'},
     'value': {'q1': 1.0, 'q3': 9.0, 'q2': 2.0, 'minimum': 1.0, 'maximum':
14.0}},
     {'vtype': u'dset', 'key': {'key_type': u'string', 'str': u'DNS'},
     'value': {'q1': 1.0, 'q3': 1.0, 'q2': 1.0, 'minimum': 1.0, 'maximum':
1.0}}]}]},
  {'stat_key': '', 'oid': 10, 'time': 1230796850000.0,
   ':summary:tprocess':
    [{'vtype': u'dset', 'key': {'key_type': u'string', 'str': u'SSH'},
     'value': {'q1': 0.0, 'q3': 0.0, 'q2': 0.0, 'minimum': 0.0, 'maximum':
5.0}}]}]},
  {'stat_key': '', 'oid': 10, 'time': 1230796880000.0,
   ':summary:tprocess':
    [{'vtype': u'dset', 'key': {'key_type': u'string', 'str': u'SSH'},
     'value': {'q1': 0.0, 'q3': 0.0, 'q2': 0.0, 'minimum': 0.0, 'maximum': 3.0}},
     {'vtype': u'dset', 'key': {'key_type': u'string', 'str': u'NTP'},

```

```
'value': {'q1': 173.0, 'q3': 178.5, 'q2': 178.0, 'minimum': 168.0,
'maximum': 179.0}}}],
...
}
```

"summary595" metric option

Sample Usage:

```
# get a 5/95 summary for device App server processing time for device with
ID 10
result = c.get_exstats ("extrahop.device.app",
                        "device",
                        [(10, -1800000, 0)],
                        [":summary595:tprocess"],
                        {'cycle':"fast"})
```

Sample Raw Result:

```
{'abs_from': 1230796524000.0,
'abs_until': 1230798324000.0,
'cycle': u'fast',
'stats':
[{'stat_key': '', 'oid': 10, 'time': 1230796826000.0,
':summary595:tprocess':
  [{'vtype': u'dset', 'key': {'key_type': u'string', 'str': u'SSH'},
'value': {'percentile5': 1.0, 'percentile95': 13.25}},
  {'vtype': u'dset', 'key': {'key_type': u'string', 'str': u'DNS'},
'value': {'percentile5': 1.0, 'percentile95': 1.0}}]}
{'stat_key': '', 'oid': 10, 'time': 1230796850000.0,
':summary595:tprocess':
  [{'vtype': u'dset', 'key': {'key_type': u'string', 'str': u'SSH'},
'value': {'percentile5': 0.0, 'percentile95': 2.0}}]},
{'stat_key': '', 'oid': 10, , 'time': 1230796880000.0,
':summary595:tprocess':
  [{'vtype': u'dset', 'key': {'key_type': u'string', 'str': u'SSH'},
'value': {'percentile5': 0.0, 'percentile95': 1.0}},
  {'vtype': u'dset', 'key': {'key_type': u'string', 'str': u'NTP'},
'value': {'percentile5': 169.0, 'percentile95': 178.90000000000001}}]},
...
}
```

topn_snap

Set of snapshot metrics.

Sample Metrics

TCP established connections detail.

Sample Usages

get_exstats

Sample_Usage:

```
result = c.get_exstats ("extrahop.device.tcp_detail",
                        "device",
                        [(0, -1800000, 0)],
                        ["established"],
                        {'cycle':"fast"})
print result
```

Sample Raw Result:

```
{'abs_from': 1230798324420.0,
 'abs_until': 1230798324420.0,
 'cycle': u'fast',
 'stats':
  [{'established':
   [{'vtype': u'snap', 'stat_key': '', 'oid': 10, 'time': 1230796837000.0,
    'key': {'key_type': u'ipaddr', 'host': u'krypton.extrahop.net', 'device_
oid': 9, 'addr': 10.10.248.29},
    'value': 1}]}
 ],
  {'established':
   [{'vtype': u'snap', 'stat_key': '', 'oid': 10, 'time': 1230797227000.0
    'key': {'key_type': u'ipaddr', 'host': u'iron.extrahop.net', 'device_oid':
8, 'addr': 10.10.248.29},
    'value': 2},
   ...
 ]}
```

Sample Pretty-Printed Results:

```
=====
time      : Thu Jan  1 00:23:07 2009 [1230798187000]
object id : 10
stat key  :
--- established -----
key: addr 10.10.1.132 host quark.extrahop.net device 3
value type: snap
10
key: addr 10.10.1.118 host lead.extrahop.net device 18
value type: snap
1
```

```
=====
time      : Thu Jan  1 00:23:37 2009 [1230798217000]
object id : 10
stat key  :
--- established -----
key: addr 10.10.1.118 host lead.extrahop.net device 18
value type: snap
1
```

get_exstats_total

Sample Usage:

```
result = c.get_exstats_total ("extrahop.device.tcp_detail",
                              "device",
                              [(0, -1800000, 0)],
                              ["established"],
                              {'cycle':"fast"})
```

Sample Result:

```
cycle      : fast
abs from   : Wed Dec 31 23:55:24 2008 [1230796524000]
abs until  : Thu Jan  1 00:25:24 2009 [1230798324000]
=====
time      : Thu Jan  1 00:23:55 2009 [1230798235000]
object id : -1
stat key  :
--- established -----
key: addr 10.10.1.132 host quark.extrahop.net device 3
value type: snap
1
key: addr 10.10.1.133 host iron.extrahop.net device 8
value type: snap
1
key: addr 10.10.1.154 host krypton.extrahop.net device 9
value type: snap
1
key: addr 10.10.1.118 host lead.extrahop.net device 19
value type: snap
1
```

get_exstats_group

Sample Usage:

```
result = c.get_exstats_group ("extrahop.device.tcp_detail",
                              "device",
                              [(0, -1800000, 0)],
                              ["established"],
                              {'cycle':"fast"})
```

Sample Result:

```
cycle      : fast
abs from   : Wed Dec 31 23:55:24 2008 [1230796524000]
abs until  : Thu Jan  1 00:25:24 2009 [1230798324000]
=====
time       : Thu Jan  1 00:23:55 2009 [1230798235000]
object id  : 10
stat key   :
--- established -----
key: addr 10.10.1.132 host quark.extrahop.net device 3
value type: snap
1
key: addr 10.10.1.154 host krypton.extrahop.net device 9
value type: snap
1
=====
time       : Thu Jan  1 00:25:01 2009 [1230798301000]
object id  : 11
stat key   :
--- established -----
key: addr 172.25.11.122 device 17
value type: snap
1
key: addr 172.25.11.117 device 13
value type: snap
2
```


topn_sset

Set of samplesets.

Sample Metrics

- HTTP server processing time detail (timing shown when clicking HTTP server **Responses**)
- Database server processing time detail (timing shown when clicking Database server **Methods**)
- CIFS server access time detail (timing shown when clicking CIFS **Files**)
- TCP round-trip times (RTTs) (timing shown when clicking TCP **Accepted** or **Connected**)

Additional Calculations

Mean:

```
# Mean Calculation
def mean(count, sum):
    if count < 1: return 0.0
    return sum / float(count)
```

Sigma:

```
# Sigma Calculation
def sigma(count, sum, sum2):
    if count < 1: return 0.0
    return math.sqrt((sum2 / count) - ((sum / float(count)) ** 2))
```

Sample Usages

get_exstats

Sample Usage:

```
result = c.get_exstats ("extrahop.device.uri_http_server_detail",
                        "device",
                        [(10, -1800000, 0)],
                        ["tprocess"],
                        {'cycle':"fast"})
print result
```

Sample Raw Result:

```
{'abs_from': 1230798324420.0,
 'abs_until': 1230798324420.0,
 'cycle': u'fast',
 'stats':
  [{'stat_key': '',
   'oid': 10,
   'time': 1230797202000.0,
   'tprocess':
    [{'vtype': u'sset',
     'key': {'key_type': u'ipaddr', 'host': u'quark.extrahop.net', 'device_
```

```
oid': 3, 'addr': 10.10.248.29},
  'value': {'count': 20.0, 'sum': 832.0, 'sum2': 96370.0}},
  {'vtype': u'sset',
   'key': {'key_type': u'ipaddr', 'host': u'iron.extrahop.net', 'device_oid':
8, 'addr': 10.10.248.29},
   'value': {'count': 15.0, 'sum': 226.0, 'sum2': 23640.0}}}],
...
}
```

Sample Pretty-Printed Result:

```
=====
time      : Thu Jan  1 00:20:42 2009 [1230798042000]
object id : 10
stat key  :
--- tprocess -----
key: addr 10.10.1.133 host iron.extrahop.net device 8
value type: sset
count 267, sum 290475, sum2 451334385, mean 1087.921348, sigma 711.911578
key: addr 10.10.1.132 host quark.extrahop.net device 3
value type: sset
count 259, sum 277212, sum2 421802746, mean 1070.316602, sigma 694.985181
=====
time      : Thu Jan  1 00:21:12 2009 [1230798072000]
object id : 10
stat key  :
--- tprocess -----
key: addr 10.10.1.133 host iron.extrahop.net device 8
value type: sset
count 86, sum 91240, sum2 137009566, mean 1060.930233, sigma 683.784710
key: addr 10.10.1.132 host quark.extrahop.net device 3
value type: sset
count 405, sum 305158, sum2 349833224, mean 753.476543, sigma 544.112890
```

Sample Usage: HTTP processing time details by URI for device with ID 10

```
result = c.get_exstats ("extrahop.device.uri_http_server_detail",
                        "device",
                        [(10, -1800000, 0)],
                        ["tprocess"],
                        {'cycle':"fast"})
print result
```

Sample Raw Result:

```
{'abs_until': 1230798324420.0,
 'abs_until': 1230798324420.0,
 'cycle': u'fast',
 'stats':
  [{'stat_key': '', 'oid': 10, 'time': 1230797165000.0,
```

```
'tprocess':
  [{'vtype': u'sset',
'key': {'key_type': u'string', 'str': u'xenon:8080/seam-dvd/checkout'},
'value': {'count': 1.0, 'sum': 75.0, 'sum2': 5625.0}}],
{'stat_key': '', 'oid': 10, 'time': 1230797195000.0,
'tprocess':
  [{'vtype': u'sset',
'key': {'key_type': u'string', 'str': u'xenon:8080/seam-dvd/img/bg.gif'},
'value': {'count': 1.0, 'sum': 1.0, 'sum2': 1.0}},
{'vtype': u'sset',
'key': {'key_type': u'string', 'str': u'xenon:8080/seam-dvd/checkout'},
'value': {'count': 4.0, 'sum': 192.0, 'sum2': 11022.0}},
{'vtype': u'sset',
'key': {'key_type': u'string', 'str': u'xenon:8080/seam-dvd/newuser/account'},
'value': {'count': 1.0, 'sum': 144.0, 'sum2': 20736.0}},
{'vtype': u'sset',
'key': {'key_type': u'string', 'str': u'xenon:8080/seam-dvd/home'},
'value': {'count': 1.0, 'sum': 31.0, 'sum2': 961.0}}
...
}
```

Sample Pretty-Printed Result:

```
cycle      : fast
abs from   : Wed Dec 31 23:55:24 2008 [1230796524000]
abs until  : Thu Jan  1 00:25:24 2009 [1230798324000]
=====
time       : Thu Jan  1 00:06:05 2009 [1230797165000]
object id  : 10
stat key   :
--- tprocess -----
key: xenon:8080/seam-dvd/checkout
value type: sset
count 1, sum 75, sum2 5625, mean 75.000000, sigma 0.000000
=====
time       : Thu Jan  1 00:06:35 2009 [1230797195000]
object id  : 10
stat key   :
--- tprocess -----
key: xenon:8080/seam-dvd/img/bg.gif
value type: sset
count 1, sum 1, sum2 1, mean 1.000000, sigma 0.000000
key: xenon:8080/seam-dvd/checkout
value type: sset
count 4, sum 192, sum2 11022, mean 48.000000, sigma 21.248529
key: xenon:8080/seam-dvd/newuser/account
value type: sset
count 1, sum 144, sum2 20736, mean 144.000000, sigma 0.000000
key: xenon:8080/seam-dvd/home
```

get_exstats_total

Sample Usage:

```
print ">>> ExStatsTotal <<<"
result = c.get_exstats_total ("extrahop.device.uri_http_server_detail",
                             "device",
                             [(10, -1800000, 0)],
                             ["tprocess"],
                             {'cycle':"fast"})
```

Sample Result:

```
cycle      : fast
abs from   : Wed Dec 31 23:55:24 2008 [1230796524000]
abs until  : Thu Jan  1 00:25:24 2009 [1230798324000]
=====
time       : Thu Jan  1 00:23:35 2009 [1230798215000]
object id  : -1
stat key   :
--- tprocess -----
key: xenon:8080/seam-dvd/dvd/192
value type: sset
count 1, sum 1672, sum2 2795584, mean 1672.000000, sigma 0.000000
key: xenon:8080/seam-dvd/dvd/383
value type: sset
count 12, sum 7846, sum2 8405232, mean 653.833333, sigma 522.434658
key: xenon:8080/seam-dvd/dvd/184
value type: sset
count 2004, sum 1831642, sum2 2582282844, mean 913.993014, sigma 673.187243
key: xenon:8080/seam-dvd/dvd/181
value type: sset
count 5, sum 2934, sum2 2300236, mean 586.800000, sigma 340.166077
key: xenon:8080//seam-dvd/dvd/149
value type: sset
count 5000, sum 3693690, sum2 5086334320, mean 738.738000, sigma 686.682628
key: xenon:8080/seam-dvd/dvd/247
value type: sset
count 16, sum 8813, sum2 8620075, mean 550.812500, sigma 485.139441
key: xenon:8080/seam-dvd/home
value type: sset
```

get_exstats_group

Sample Usage:

```
print ">>> ExStatsTotal <<<"
result = c.get_exstats_group ("extrahop.device.uri_http_server_detail",
                              "activity_group",
                              [("extrahop.device.http_server", -1800000, 0)],
                              ["tprocess"],
                              {'cycle':"fast"})
```

Sample Raw Result:

```
{'abs_until': 1230798324000.0,
 'cycle': u'fast',
 'stats':
  [{'stat_key': '',
   'oid': 6,
   'time': 1230797602000.0,
   'tprocess':
    [{'vtype': u'sset',
     'key': {'key_type': u'string', 'str': u'neon/gui/gui.swf'},
     'value': {'count': 3.0, 'sum': 1.0, 'sum2': 1.0}},
    {'vtype': u'sset',
     'key': {'key_type': u'string', 'str': u'neon/favicon.ico'},
     'value': {'count': 2.0, 'sum': 0.0, 'sum2': 0.0}},
    {'vtype': u'sset',
     'key': {'key_type': u'string', 'str': u'neon/bridge'},
     'value': {'count': 436.0, 'sum': 1052.0, 'sum2': 45178.0}}}],
  {'stat_key': '',
   'oid': 21,
   'time': 1230797818000.0,
   'tprocess':
    [{'vtype': u'sset',
     'key': {'key_type': u'string', 'str':
u'tax.dsprod.drugstore.com:8080/TaxServer/'},
     'value': {'count': 407.0, 'sum': 13903.0, 'sum2': 5317579.0}}
    ]
  }
}
```

Sample Pretty-Printed Result:

```
cycle      : slow
abs from   : Wed Dec 31 16:30:00 1969 [1800000]
abs until  : Thu Jan  1 00:25:24 2009 [1230798324000]
=====
time       : Thu Jan  1 00:17:52 2009 [1230797872000]
object id  : 6
stat key   :
--- tprocess -----
key: neon/gui/gui.swf
value type: sset
count 2, sum 1, sum2 1, mean 0.500000, sigma 0.500000
key: neon/bridge
value type: sset
count 318, sum 472, sum2 19842, mean 1.484277, sigma 7.758424
key: neon/favicon.ico
value type: sset
count 2, sum 0, sum2 0, mean 0.000000, sigma 0.000000
=====
time       : Thu Jan  1 00:17:52 2009 [1230797872000]
object id  : 21
stat key   :
--- tprocess -----
key: tax.dsprod.drugstore.com:8080/TaxServer/
```

```
value type: sset  
count 407, sum 13903, sum2 5317579, mean 0.500000, sigma 0.500000
```

topn_time

Set of times.

Sample Metrics

SSL certificate expiration detail.

Sample Usages

get_exstats

Sample Usage:

```
result = c.get_exstats ("extrahop.device.ssl_server_detail",
                        "device",
                        [(0, -1800000, 0)],
                        ["cert_expiration"],
                        {'cycle':"fast"})
print result
```

Sample Raw Result:

```
{'abs_from': 1230798324420.0,
 'abs_until': 1230798324420.0,
 'cycle': u'fast',
 'stats':
  [{'stat_key': '',
   'oid': 0,
   'time': 1230796908000.0,
   'cert_expiration':
    [{'vtype': u'time',
     'key': {'key_type': u'string', 'str': u'mail.google.com:RSA_1024'},
     'value': 1241247600.0},
     {'vtype': u'time',
      'key': {'key_type': u'string', 'str': u'voyager.commandcode.com:RSA_1024'},
      'value': 1266739200.0},
     {'vtype': u'time',
      'key': {'key_type': u'string', 'str': u'*.mail.google.com:RSA_1024'},
      'value': 1262332800.0},
     {'vtype': u'time', 'key': {'key_type': u'string', 'str':
u'www.google.com:RSA_1024'},
      'value': 1241247600.0}]}]}
...
}
```

Sample Pretty-Printed Result:

```
=====
time      : Thu Jan  1 00:20:18 2009 [1230798018000]
object id : 0
stat key  :
--- cert_expiration -----
```

```
key: imap.gmail.com:RSA_1024
value type: time
1240988400.0
=====
time      : Thu Jan  1 00:20:48 2009 [1230798048000]
object id : 0
stat key  :
=====
time      : Thu Jan  1 00:21:18 2009 [1230798078000]
object id : 0
stat key  :
--- cert_expiration -----
key: voyager.commandcode.com:RSA_1024
value type: time
1266739200.0
```

get_exstats_group

Sample Usage:

```
result = c.get_exstats_group ("extrahop.device.ssl_server_detail",
                              "activity_group",
                              [{"extrahop.device.ssl_server", -1800000, 0}],
                              ["cert_expiration"],
                              {'cycle':"fast"})
```

Sample Raw Result:

```
>>> ExStatsGroup <<<
{'abs_until': 1230798324000.0,
 'cycle': u'fast',
 'stats':
  [{'stat_key': '', 'oid': 14, 'time': 1230796937000.0,
   'cert_expiration':
    [{'vtype': u'time',
     'key': {'key_type': u'string', 'str': u'pubs.acs.org:RSA_1024'},
     'value': 1273820400.0}]},
   {'stat_key': '', 'oid': 11, 'time': 1230796901000.0,
    'cert_expiration':
     [{'vtype': u'time',
      'key': {'key_type': u'string', 'str': u'pubs.acs.org:RSA_1024'},
      'value': 1273820400.0}]},
   {'stat_key': '', 'oid': 16, 'time': 1230796896000.0,
    'cert_expiration':
     [{'vtype': u'time',
      'key': {'key_type': u'string', 'str': u'pubs.acs.org:RSA_1024'},
      'value': 1273820400.0}]},
   ...
  ]
}
```


topn_tset

Set of top-N sets.

Sample Metrics

- URI detail by HTTP status code
- IP detail by HTTP status code
- URI detail by HTTP method
- IP detail by HTTP method

Sample Usages

get_exstats

Sample Usage:

```
result = c.get_exstats ("extrahop.device.uri_http_server_detail",
                        "device",
                        [(10, -1800000, 0)],
                        ["status_code"],
                        {'cycle':"fast"})
print result
```

Sample Raw Result:

```
{'abs_from': 1230798324420.0,
 'abs_until': 1230798324420.0,
 'cycle': u'fast',
 'stats':
  [{'status_code':
    [{'vtype': u'tset',
      'stat_key': '',
      'oid': 10,
      'time': 1230797171000.0,
      'key': {'key_type': u'string', 'str': u'500'},
      'value':
        [{'vtype': u'count',
          'key': {'key_type': u'string', 'str': u'xenon:8080/seam-dvd/checkout'},
          'value': 1}]}]}],
 'status_code':
  [{'vtype': u'tset',
    'key': {'key_type': u'string', 'str': u'200'},
    'value':
      [{'vtype': u'count',
        'key': {'key_type': u'string', 'str': u'xenon:8080/seam-dvd/checkout'},
        'value': 2},
       {'vtype': u'count',
        'key': {'key_type': u'string', 'str': u'xenon:8080/seam-dvd/dvd/383'},
        'value': 2},
       {'vtype': u'count',
        'key': {'key_type': u'string', 'str': u'xenon:8080/seam-dvd/dvd/112'},
        'value': 1}],
  ]}
```

```
...
}
```

Sample Pretty-Printed Result:

```
=====
time      : Thu Jan  1 00:21:41 2009 [1230798101000]
object id : 10
stat key  :
--- status_code -----
key: 302
value type: tset
key: xenon:8080//seam-dvd/dvd/149
value type: count
219
key: 200
value type: tset
key: xenon:8080//seam-dvd/dvd/149
value type: count
237
=====
time      : Thu Jan  1 00:22:11 2009 [1230798131000]
object id : 10
stat key  :
--- status_code -----
key: 302
value type: tset
key: xenon:8080//seam-dvd/dvd/149
value type: count
236
key: 200
value type: tset
key: xenon:8080//seam-dvd/dvd/149
value type: count
266
```

get_exstats_total

Sample Usage:

```
...
```

Sample Raw Result:

```
{'abs_until': 1230798324000.0,
 'cycle': u'fast',
 'stats': [{ 'stat_key': '', 'oid': 6, 'time': 1230798324000.0,
 'status_code':
  [{'vtype': u'tset',
 'key': {'key_type': u'string', 'str': u'200'}},
```

```
'value':
  [{'vtype': u'count',
    'key': {'key_type': u'string', 'str': u'xenon:8080/seam-
dvd/dvd/181'},
    'value': 5},
  {'vtype': u'count',
    'key': {'key_type': u'string', 'str': u'xenon:8080/seam-dvd/debug'},
    'value': 8},
  ...
```

Sample Pretty-Printed Result:

```
cycle      : slow
abs from   : Wed Dec 31 16:30:00 1969 [1800000]
abs until  : Thu Jan  1 00:25:24 2009 [1230798324000]
=====
time       : Thu Jan  1 00:25:24 2009 [1230798324000]
object id  : -1
stat key   : None
--- status_code -----
key: 200
value type: tset
key: neon/gui/lookout.html
value type: count
1
key: neon/bridge
value type: count
318
key: neon/gui/gui.swf
value type: count
1
```

get_exstats_group

Sample Usage:

```
result = c.get_exstats_group ("extrahop.device.uri_http_server_detail",
                             "activity_group",
                             [("extrahop.device.http_server", -1800000, 0)],
                             ["status_code"],
                             {'cycle':"fast"})
```

Sample Raw Result:

```
{'abs_from': 1230798324420.0,
 'abs_until': 1230798324000.0,
 'cycle': u'fast',
 'stats': [{'stat_key': '', 'oid': 6, 'time': 1230798250000.0,
```

```
'status_code':
  [{'vtype': u'tset',
'key': {'key_type': u'string', 'str': u'200'},
'value':
  [{'vtype': u'count',
    'key': {'key_type': u'string', 'str': u'neon/gui/lookout.html'},
    'value': 1},
  {'vtype': u'count',
    'key': {'key_type': u'string', 'str': u'neon/bridge'},
    'value': 318} ],
'key': {'key_type': u'string', 'str': u'404'},
'value':
  [{'vtype': u'count',
    'key': {'key_type': u'string', 'str': u'neon/favicon.ico'},
    'value': 2} ] ]}
...
}
```

Sample Pretty-Printed Result:

```
cycle      : slow
abs from   : Wed Dec 31 16:30:00 1969 [1800000]
abs until  : Thu Jan  1 00:25:24 2009 [1230798324000]
=====
time       : Thu Jan  1 00:17:52 2009 [1230797872000]
object id  : 6
stat key   :
--- status_code -----
key: 200
value type: tset
key: neon/gui/lookout.html
value type: count
1
key: neon/bridge
value type: count
318
key: 404
value type: tset
key: neon/favicon.ico
value type: count
2
=====
time       : Thu Jan  1 00:17:52 2009 [1230797872000]
object id  : 0
stat key   :
--- status_code -----
key: 200
value type: tset
key: blah.html
value type: count
1
```

Examples

The examples listed below are available in this documentation but are also included in the Examples directory in the Python SDK.

"activity-stats.py" on the next page

This script queries captures of a specified ExtraHop for server/client activity.

"applications.py" on page 279

This script queries count type statistics for layer 2 bytes as well as their capture time.

"bridge-devcount.py" on page 281

This script queries captures for the total number of devices, as well as active devices. Additionally, this script counts the type of specific devices.

"count.py" on page 282

This script provides counts of the number of incoming packets and bytes for the selected device.

"custom-devices.py" on page 284

This script creates a custom device with specified criteria.

"dataset.py" on page 285

This script queries for some dataset type statistics.

"device-groups.py" on page 287

This script creates, prints and deletes device groups.

"device-search-by-activity.py" on page 289

This script gets a list of devices with HTTP server activity for the last 30 minutes and then prints out a list of devices with MAC, IP Address and name for each device.

"device-search-by-name.py" on page 289

This script searches for L3 devices containing the string "apple" in the last 30 minutes.

"device-tags.py" on page 290

This script adds a tag "test" to devices with the IP address of "10.10.1.254".

"device-update-name.py" on page 292

This script updates a device's custom name.

"dns-queries.py" on page 293

This script retrieves all host queries in the last 30 minutes and writes the results to a CSV file with a unique name, disambiguated by date.

"exstats-ecm.py" on page 294

This script gets exstats for client HTTP server responses on an ECM.

"get-alert-definitions.py" on page 295

This script prints out a list of all the defined alerts.

"get-alerts-fired.py" on page 295

This script gets a list of active devices, fetches alerts with the name defined below, and fetches metadata defined below for the band of time defined by the offset.

"l7-proto-details.py" on page 297

This script gets L7 metrics for the number of packets, the number of bytes, and the port number of captures with the last 30 minutes.

"print-all-devices.py" on page 298

This script prints out a list of all the discovered devices on the ExtraHop as well as some of its network properties.

"print-trigger-runtime-log.py" on page 299

This script prints the trigger runtime log. In order to print to the log, debugging must be enabled for the trigger.

"pull-running-config.py" on page 300

This script prints the current running configuration.

"set-host-name.py" on page 301

This script changes the hostname within the running config.

"ssl-key-expiration.py" on page 301

This script prints the name, common name and expiration of any SSL key on the ExtraHop that is expiring within 'expiration-days' or which has already expired.

"topn-dset.py" on page 303

This script queries topn dset type statistics.

"topn-sset.py" on page 304

This script queries topn sset type statistics.

"whitelist.py" on page 305

This script adds devices matching the search string and whitelists them. After whitelisting them, it removes them from the whitelist.

activity-stats.py

This script queries captures of a specified ExtraHop for server/client activity.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script queries captures of a specified extrahop device for
server/client activity.
"""

# Imports -----

import time
from pyhop import pyhop

client = pyhop.make_client()

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TFROM = 0
TUNTIL = 0
ONE_SECOND = 1000

ALL_CAPTURES = 0
FIRST_CAPTURE = 0

# Stat Key, this value should be changed to your extrahop device ID.
STAT_KEY = "fff41510a8c00000"
```

```
# Code -----

# set options for get_exstats
options = {"cycle": "slow",
          "stat_key": STAT_KEY}

# Use the first capture found. Note: On an ECM there may be multiple
# captures associated to different nodes
capture = client.get_captures(ALL_CAPTURES) [FIRST_CAPTURE]

# get set of stats from the specified device or capture with the
# specified metrics
result = client.get_exstats("extrahop.capture.device_activity",
                            "capture",
                            [(capture.oid, TFROM, TUNTIL)],
                            [],
                            options)

# Print Device Stats
for stat in result.stats:
    print (">>> device {0} {1}".format(stat.stat_key,
                                       time.ctime(stat.time / ONE_SECOND)))

    # each stat type has a key in the metric
    for key in sorted(stat.keys()):
        if key in ("time", "oid", "stat_key"):
            continue

        # If there is activity, print it out
        if stat[key] > 0:
            print (key, stat[key])

client.close()
```

See Also:

"get_captures" on page 29

"get_exstats" on page 58

applications.py

This script queries count type statistics for layer 2 bytes as well as their capture time.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script queries count type statistics for layer 2 bytes as well as
their capture times.
"""
```

```
# Imports -----

import time
from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TFROM = -180000
TUNTIL = 0

ALL_CAPTURES = 0
ONE_SECOND = 1000

# Code -----

client = pyhop.make_client()

# get all capture objects
result = client.get_captures(ALL_CAPTURES)
if not result:
    print "No captures available."
    exit()

# Use the first capture found.
capture = result[0]
print("capture (oid,mod_time,idle):", capture.oid,
      capture.mod_time, capture.idle)

result = client.get_applications(TFROM, TUNTIL)
if not result:
    print "No applications available."
    exit()

# print each application in result
for apps in result:
    print apps.oid, apps.name, apps.comment

object_spec = ((apps.oid, TFROM, TUNTIL),)
field_spec = ["req_l2_bytes", "rsp_l2_bytes"]
options = {}

# get set of stats from the specified device or capture with the
# specified metrics
result = client.get_exstats("extrahop.application.net", "application",
                            object_spec, field_spec, options)

# for each stat, print the value of its associated stat_key
for stat in result.stats:
    print("Time: {0} - req L2 bytes: {1} - rsp L2 bytes: {2}"
          .format(time.ctime(long(stat.time) / ONE_SECOND),
```



```
        stat.req_l2_bytes,  
        stat.rsp_l2_bytes))  
  
client.close()
```

See Also:

"get_applications" on page 28

"get_captures" on page 29

"get_exstats" on page 58

bridge-devcount.py

This script queries captures for the total number of devices, as well as active devices. Additionally, this script counts the type of specific devices.

```
#!/usr/bin/env python2  
#-*- Mode: python -*-  
  
"""  
Copyright 2015, ExtraHop Networks, Inc  
  
This script queries captures for the total number of devices, as well  
as active devices. Additionally, this script counts the type of  
specific devices.  
"""  
  
# Imports -----  
  
from pyhop import pyhop  
  
# Constants -----  
  
# Object ID of the ExtraHop. Running this script on an ECM requires the  
# ExtraHop Object ID of interest.  
OBJECT_ID = 0  
  
# Using a negative number indicates time (in ms) relative to the current  
# capture time. Setting TUNTIL to '0' represents using the current  
# capture time however, setting TFROM to '0' represents UNIX time 0.  
TUNTIL = 0  
  
# Code -----  
client = pyhop.make_client()  
object_spec = (OBJECT_ID, client.tfrom, TUNTIL),  
  
fields = ("l2_device gateway_device pseudo_device l3_device "  
         "l2_device_active gateway_device_active pseudo_device_active "  
         "l3_device_active")  
  
field_spec = fields.split()
```

```
# Get stats from the specified device or capture with the
# specified metrics
result = client.get_exstats("extrahop.system.bridge",
                            "capture",
                            object_spec,
                            field_spec,
                            {})

# print the total amount of devices and active devices
for stat in result.stats:
    print "~" * 50
    # total number of devices
    stat.total = (stat.l2_device + stat.gateway_device +
                 stat.pseudo_device + stat.l3_device)

    # total number of active devices
    stat.total_active = (stat.l2_device_active +
                        stat.gateway_device_active +
                        stat.pseudo_device_active +
                        stat.l3_device_active)

    keys = stat.keys()
    keys.sort()
    for key in keys:
        print key, stat[key]

client.close()
```

See Also:

"get_exstats" on page 58

count.py

This script provides counts of the number of incoming packets and bytes for the selected device.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script provides counts of the number of incoming packets and bytes
for the selected device.
"""

# Imports -----

import sys
import time
from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
```

```
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TUNTIL = 0
TFROM = -180000
ONE_SECOND = 1000

NO_ACTIVE_DEVICES = 0
# Code -----

client = pyhop.make_client()
options = {}

# Get some active devices.
result = client.search_devices_by_activity(TFROM,
                                          TUNTIL,
                                          "extrahop.device.tcp",
                                          options)

# Check if there are active devices
if len(result.devices) == NO_ACTIVE_DEVICES:
    print "No active devices"
    sys.exit(0)
else:
    print (len(result.devices), "active devices")

# Get incoming packets, in bytes, for the first device.
device = result.devices[0]
print ("Fetching >>> stats for device {0} {1}"
      .format(device.oid, device.name))
result = client.get_exstats("extrahop.device.net",
                           "device",
                           [(device.oid, TFROM, TUNTIL)],
                           ["pkts_in", "bytes_in"],
                           options)

print ">>> ExStats - Pretty-Printed <<<"
for stat in result.stats:
    print (time.ctime(long(stat.time) / ONE_SECOND),
          stat.pkts_in,
          stat.bytes_in)

# Get the total number of incoming packets in Bytes
print "Fetching >>> totals for device", device.oid, device.name
result = client.get_exstats_total("extrahop.device.net",
                                  "device",
                                  [(device.oid, TFROM, TUNTIL)],
                                  ["pkts_in", "bytes_in"],
                                  options)

print ">>> ExStatsTotal - Pretty-Printed <<<"
for stat in result.stats:
    print ("Total: ", time.ctime(long(stat.time) / ONE_SECOND),
          stat.pkts_in,
```

```
stat.bytes_in)

# Get incoming packets, in bytes, for 'TCP' activity group
print "Fetching >>> device totals for TCP activity group"
result = client.get_exstats_group("extrahop.device.net",
                                  "activity_group",
                                  [("extrahop.device.tcp", TFROM, TUNTIL)],
                                  ["pkts_in", "bytes_in"],
                                  options)

print ">>> ExStatsGroup - Pretty-Printed <<<"
for stat in result.stats:
    print ("time: {0} \tdevice: {1} \tbytes_in: {2} \tpkts_in: {3}"
           .format(time.ctime(long(stat.time) / ONE_SECOND),
                   stat.oid,
                   stat.bytes_in,
                   stat.pkts_in))

# Get total packet count, in bytes for activity group 'TCP'
print "Fetching >>> totals for TCP activity group"
result = client.get_exstats_total("extrahop.device.net",
                                  "activity_group",
                                  [('extrahop.device.tcp', TFROM, TUNTIL)],
                                  ["pkts_in", "bytes_in"],
                                  options)

print ">>> ExStatsTotal for Group - Pretty-Printed<<<"
for stat in result.stats:
    print ("Group Total: {0} {1} {2}"
           .format(time.ctime(long(stat.time) / ONE_SECOND),
                   stat.pkts_in,
                   stat.bytes_in))

client.close()
```

See Also:

"get_exstats" on page 58

"get_exstats_group" on page 63

"get_exstats_total" on page 69

"search_devices_by_activity" on page 138

custom-devices.py

This script creates a custom device with specified criteria.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc
```

This script creates a custom device with specified criteria.

```
"""
# Imports -----

import pyhop

# Code -----

c = pyhop.make_client()

name = "testdev2"
# name:          a unique friendly name
# device_id:     a unique string id for this dev
#                usually the same as its name
# disabled:      if the custom device should start off disabled
# author (opt):  who created the device
# comment (opt): a comment to remember it by
cd = c.create_custom_device({'name': name,
                             'device_id': name,
                             'disabled': False,
                             'author': "eball",
                             'comment': "a test custom device"})

# All criteria are optional - use the ones wanted
criteria = {'dst_port_max': 6000,
            'dst_port_min': 512,
            'ipaddr': "10.10.2.0/24",
            'src_port_max': 6000,
            'src_port_min': 512,
            'vlan_max': 2000,
            'vlan_min': 0}

# Multiple criteria may be added to the same custom device
c.create_custom_device_criterion(cd['id'], criteria)
c.close()
```

See Also:

"create_custom_device" on page 99

"create_custom_device_criterion" on page 100

dataset.py

This script queries for some dataset type statistics.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc
```

```
This script queries for some dataset type stats.
"""

# Imports -----

from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TUNTIL = 0
TFROM = -180000

# Code -----

client = pyhop.make_client()

stat_type = "extrahop.device.http_server"
object_type = "activity_group"
object_spec = [("extrahop.device.http_server", TFROM, TUNTIL)]
opts = {}

# get HTTP processing time for an activity group 'HTTP Server'
result = client.get_exstats_group(stat_type,
                                  object_type,
                                  object_spec,
                                  ["tprocess"],
                                  opts)

for stat in result.stats:
    print ">>> device", stat.oid, "tprocess dataset:"
    for datapoint in stat.tprocess:
        print "value:", datapoint.value, "freq:", datapoint.freq

# get a 5-number summaries
result = client.get_exstats_group(stat_type,
                                  object_type,
                                  object_spec,
                                  [":summary:tprocess"],
                                  opts)

for stat in result.stats:
    print ">>> device", stat.oid, "tprocess 5-number summary:"
    s = stat[":summary:tprocess"]
    print ("min:", s.min, "q1:", s.q1, "mean:",
           s.q2, "q3:", s.q3, "max:", s.max)

# get a 5th/95th percentiles for the entire group
result = client.get_exstats_total(stat_type,
                                  object_type,
                                  object_spec,
                                  [":summary595:tprocess"],
                                  opts)
```

```
print ">>> group tprocess 5-number summary:"
s = result.stats[0][":summary595:tprocess"]
print "5th percentile:", s.pc5, "95th percentile:", s.pc95

client.close()
```

See Also:

"get_exstats_group" on page 63

"get_exstats_total" on page 69

device-groups.py

This script creates, prints and deletes device groups.

```
#!/usr/bin/python
#-*- Mode: python -*-
# Copyright 2012-2015, ExtraHop Networks, Inc

from pyhop import pyhop

def print_device_groups():
    print "----- Printing a list of all groups -----"
    result = c.get_device_groups(-180000)
    for group in result:
        if group.deleted == False:
            if group.dynamic:
                members = c.get_dynamic_group_devices(-180000, 0,
                                                       group.oid, {})

                print("Name:", group.name,
                      ", Oid:", group.oid,
                      ", Type: dynamic",
                      ", # Members:", members.total,
                      ", Field:", group.field,
                      ", Value:", group.value)
            else:
                print("Name:", group.name,
                      ", Oid:", group.oid,
                      ", Type: static",
                      ", # Members:", len(group.members))

# What this script does:
# - Creates, prints and delete device groups

c = pyhop.make_client()

# create a dummy static device group with 3 members
result = c.new_device_group('dummy static', 'test', [0, 1, 2])
# get oid of the new device group
dummy_oid = result.oid
print "Added a dummy static group with oid", dummy_oid
```

```
# print a list of existing device groups
print_device_groups()

# update a dummy static device group to have 4 members
c.update_device_group(dummy_oid, 'dummy static - updated',
                      'test - updated', [0, 1, 2, 3])
print "Updated a dummy static group with oid", dummy_oid

# print a list of existing device groups
print_device_groups()

# delete the dummy static device group
c.delete_device_groups([dummy_oid])
print "Deleted a dummy static group with oid", dummy_oid

# print a list of existing device groups
print_device_groups()

# create a dummy static device group for all VMWare devices, based
# on regex 'vmware'
result = c.new_dynamic_device_group('dummy dynamic', 'test',
                                    'vendor', 'vmware')

# get oid of the new device group
dummy_oid = result.oid
print "Added a dummy dynamic group with oid", dummy_oid

# print a list of existing device groups
print_device_groups()

# update a dummy dynamic device group for all devices on vlan 4,
# based on exact number '4'
c.update_dynamic_device_group(dummy_oid, 'dummy dynamic - updated',
                              'test - updated', 'vlan', '\Q4\E')
print "Updated a dummy dynamic group with oid", dummy_oid

# print a list of existing device groups
print_device_groups()

# delete the dummy dynamic device group
c.delete_device_groups([dummy_oid])
print "Deleted a dummy dynamic group with oid", dummy_oid

# print a list of existing device groups
print_device_groups()

c.close()
```

See Also:

"delete_device_groups" on page 105

"get_device_groups" on page 43

"get_dynamic_group_devices" on page 54

"new_device_group" on page 113

"new_dynamic_device_group" on page 116

"update_dynamic_device_group" on page 134

device-search-by-activity.py

This script gets a list of devices with HTTP server activity for the last 30 minutes and then prints out a list of devices with MAC, IP Address and name for each device.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script gets a list of devices with HTTP server activity for the
last 30 minutes. This script then prints out a list of devices with:
MAC, IP address and name.
"""

# Imports -----

from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TUNTIL = 0
TFROM = -180000

# Code -----
client = pyhop.make_client()

# Get the active devices.
result = client.\
    search_devices_by_activity(TFROM,
                              TUNTIL,
                              "extrahop.device.http_server",
                              {})

print "Total devices: ", result.total
for device in result.devices:
    print ("{0} - {1} - {2}".format(device.macaddr,
                                   device.ipaddr4,
                                   device.name))

client.close()
```

See Also:

"search_devices_by_activity" on page 138

device-search-by-name.py

This script searches for L3 devices containing the string "apple" in the last 30 minutes.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script searches for L3 devices containing the string "apple" in
the last 30 minutes.
"""
# Imports -----

from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TFROM = -1800000
TUNTIL = 0

SEARCH_STRING = "apple"

# Code -----

client = pyhop.make_client()

# Grab the capture object
result = client.get_captures(TFROM)
if result is not None:
    capture = result[0]
    print("capture (oid,mod_time,idle):", capture.oid,
          capture.mod_time, capture.idle)

# Set options for our search.
options = {"capture_oid": capture.oid,
          "devtype": "l3"}

# Search for an L3 device containing our search string.
result = client.search_devices_by_name(TFROM,
                                       TUNTIL,
                                       SEARCH_STRING,
                                       options)

print "total:", result.total
for device in result.devices:
    print ("{0} - {1}".format(device.name, device.macaddr))

client.close()
```

See Also:

"search_devices_by_name" on page 148

device-tags.py

This script adds a tag "test" to devices with the IP address of "10.10.1.254".

```
Thi#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

Adds a tag "test" to devices with IP "10.10.1.254"
"""

# Imports -----

import string
import sys
from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TFROM = -1800000
TUNTIL = 0

SEARCH_STRING = "10.10.1.254"

# Code -----

c = pyhop.make_client()

# get device by IP
options = {}
result = c.search_devices_by_ip(TFROM,
                                TUNTIL,
                                SEARCH_STRING,
                                options)

if len(result.devices) == 0:
    print "No matching devices found."
    sys.exit(0)

# Grab device IDs
object_ids = [d.oid for d in result.devices]
for d in result.devices:
    print "device", d.oid, d.name, d.ipaddr4
    print "  tags:", string.join(d.tags, ",")

# add a tag
print "Adding tag", "'test'", "to device(s)", object_ids
c.device_tag_add("test", object_ids)

c.close()
```

See Also:

"device_tag_add" on page 108

"search_devices_by_ip" on page 144

device-update-name.py

This script updates a device's custom name.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script updates a device's custom name.
"""
# Imports -----

from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TFROM = -1800000
TUNTIL = 0
DEVICE_IP = "10.10.1.135"

# Code -----

client = pyhop.make_client()

result = client.search_devices_by_ip(TFROM, TUNTIL, DEVICE_IP, {})

# parse through result
print "Total devices: ", result.total
for device in result.devices:

    # Print some device metrics as well as its current custom name
    print device.macaddr, device.ipaddr4, device.custom_name

    print "Updating device with custom name:", device.custom_name
    client.update_device({"oid": device.oid,
                        "custom_name": "my custom name"})
    d = client.get_device(TFROM, TUNTIL, device.oid)

    print "New custom name:", d.custom_name

client.close()
```

See Also:

get_device on page 37

"search_devices_by_ip" on page 144

"update_device" on page 129

dns-queries.py

This script retrieves all host queries in the last 30 minutes and writes the results to a CSV file with a unique name, disambiguated by date.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script retrieves all host queries in the last 30 minutes and
write the results to a CSV file with a unique name disambiguated by
date.
"""

# Imports -----

import datetime
import csv
from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. For the PyHop API, get functions using TUNTIL and TFROM
# Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TFROM = -1800000
TUNTIL = 0

# Code -----

client = pyhop.make_client()

r = client.\
    get_exstats_group("extrahop.device.dns_server",
                      "activity_group",
                      [("extrahop.device.dns_server", TFROM, TUNTIL)],
                      ["host_query"],
                      {})

# write out result to CSV
now = datetime.datetime.now()
ofile = csv.writer(
    open('extrahop_export_' + now.strftime("%Y-%m-%d_%H%M") + '.csv',
        'wb'),
    delimiter=',', quotechar='|', quoting=csv.QUOTE_MINIMAL)
ofile.writerow(['name', 'count'])

# Check if such a device exists (there could be more than one).
for stat in r.stats:
    for query in stat.host_query:
        ofile.writerow([query.key.str, query.value])
```

```
client.close()
```

See Also:

"get_exstats_group" on page 63

exstats-ecm.py

This script gets exstats for client HTTP server responses on an ECM.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script gets exstats for client HTTP server responses on an ECM.
"""

# Imports -----

from pyhop import pyhop

# Code -----

client = pyhop.make_client()

# Grab HTTP client server information
http_server = client.activity_groups.http_server
name_map = {}

# Create a dictionary of device IDs to device names.
for d in http_server.members:
    name_map[d.oid] = d.name

results = http_server.get_exstats_group(field_spec=["rsp"])

# Print out the number of http responses for each device.
for stat in results.stats:
    print stat.oid, name_map[stat.oid], "=>", stat.rsp

# print out any per-node errors
if "errors" in results:
    print ">>> errors:"
    for e in results.errors:
        print "-", e

client.close()
```

See Also:

"get_exstats_group" on page 63

get-alert-definitions.py

This script prints out a list of all the defined alerts.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script prints out a list of all the defined alerts.
"""

# Imports -----

import json
from pyhop import pyhop

# Constants -----
ALL_CAPTURES = 0

# Code -----

client = pyhop.make_client()
result = client.get_alerts(ALL_CAPTURES)

print json.dumps(result, indent=2)
client.close()
```

See Also:

get_alerts on page 17

get-alerts-fired.py

This script gets a list of active devices, fetches alerts with the name defined below, and fetches metadata defined below for the band of time defined by the offset.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script does the following:
- Gets a list of active devices
- Fetches alerts with the name defined below
- Fetches metadata defined below for band of
  time defined by offset
"""

# Imports -----

from pyhop import pyhop
```

```
# Constants -----

# Setting TUNTIL to '0' represents using the current capture time.
TUNTIL = 0

# polling interval in minutes
POLL_INT = 10

# before/after time offset for metadata in minutes
META_OFFSET = 1

# milliseconds in a minute
MILLI_IN_MIN = 60000

# Alert to match
ALERT_TO_MATCH = "Database Errors > 0"

# max number of devices to poll
MAX_POLL_DEVICES = 1000

OFFSET = 0

# Code -----

# correlating metric
corr_metric = "extrahop.device.db_server_detail"
corr_fields = ["error_msg", "rsp_error"]

client = pyhop.make_client()

# get active devices
result = client.get_active_devices(OFFSET,
                                   MAX_POLL_DEVICES,
                                   -MILLI_IN_MIN * POLL_INT,
                                   TUNTIL)
print "Number of active devices:", result.total

for device in result.devices:
    # get alerts for device
    print "Getting alerts for device:", device.oid
    result = client.\
        get_exstats("extrahop.device.alert",
                    "device",
                    [(device.oid, -MILLI_IN_MIN * POLL_INT, TUNTIL)],
                    [],
                    {'cycle': "slow"})

    if result.stats is None:
        exit

    for stat in result.stats:
        if stat.alert_name == ALERT_TO_MATCH:
```



```
print("Getting metadata for", ALERT_TO_MATCH,
      "on device", device.oid)
# get metadata
metadata = client.get_exstats(
    corr_metric,
    "device",
    [(device.oid,
      stat.stat_time-MILLI_IN_MIN * META_OFFSET,
      stat.stat_time+MILLI_IN_MIN * META_OFFSET)],
    corr_fields,
    {})
# do something with result, e.g. print
print metadata
client.close()
```

See Also:

get_active_devices on page 19

get_exstats on page 58

[l7-proto-details.py](#)

This script gets L7 metrics for the number of packets, the number of bytes, and the port number of captures with the last 30 minutes.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script gets L7 metrics for the number packets, the number of bytes,
and port number of captures within the last 30 seconds
"""

# Imports -----

from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TFROM = -180000
TUNTIL = 0

# Object ID of the ExtraHop. Running this script on an ECM requires the
# ExtraHop Object ID of interest.
OBJECT_ID = 0

STATKEY_FIELD = 0
```

```
# Code -----

client = pyhop.make_client()

options = {}
result = {}

# get device pkts, bytes in for first device
r = client.get_exstats_group("extrahop.capture.app",
                             "capture",
                             [(OBJECT_ID, TFROM, TUNTIL)],
                             ["pkts", "bytes"],
                             {})

if not r.stats:
    print "No stats available."
    exit()

# Store the number of packets
for pkt in r.stats[STATKEY_FIELD].pkts:
    print pkt
    # result[pkt.key.str] = {"pkts": pkt.value, "bytes": 0}

# Store the number of bytes
for byte in r.stats[0].bytes:
    if byte.key.str in result:
        result[byte.key.str]["bytes"] = byte.value
    else:
        result[byte.key.str] = {"pkts": 0, "bytes": byte.value}

# Sort and print our results.
keys = result.keys()
keys.sort()
for k in keys:
    print k, "=> packets", result[k]["pkts"], \
          "bytes", result[k]["bytes"]

client.close()
```

See Also:

[get_exstats_group](#) on page 63

[print-all-devices.py](#)

This script prints out a list of all the discovered devices on the ExtraHop as well as some of its network properties.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc
```

This script prints out a list of all the discovered devices on the ExtraHop as well as some of its network properties

```
"""  
  
# Imports -----  
  
from pyhop import make_client  
  
# Code -----  
  
client = make_client()  
  
# get all devices  
result = client.get_all_devices()  
  
# print out each device and its network properties  
for device in result:  
    print ("device:", device.oid, device.macaddr, device.ipaddr4,  
          device.dns_name, device.nb_name, device.mod_time)  
  
client.close()
```

See Also:

[get_all_devices](#) on page 24

[print-trigger-runtime-log.py](#)

This script prints the trigger runtime log. In order to rprint to the log, debugging must be enabled for the trigger.

```
#!/usr/bin/env python2  
#-*- Mode: python -*-  
  
"""  
Copyright 2015, ExtraHop Networks, Inc  
  
This script prints the trigger runtime log. In order to print to the  
log, debugging must be enabled for the trigger.  
"""  
  
# Imports -----  
  
import json  
from pyhop import pyhop  
  
# Constants -----  
  
# Using a negative number indicates time (in ms) relative to the current  
# capture time. Setting TUNTIL to '0' represents using the current  
# capture time however, setting TFROM to '0' represents UNIX time 0.  
TFROM = -1800000  
TUNTIL = 0
```

```
# Object ID of the ExtraHop. Running this script on an ECM requires the
# ExtraHop Object ID of interest.
OBJECT_ID = 0

# Code -----

client = pyhop.make_client()

# Simply grabbing the first capture object for the sake of this
# example Note: On an ECM there may be multiple captures associated to
# different nodes
capture = client.get_captures(OBJECT_ID)[0]
for line in capture:
    print line

print "Fetching trigger runtime log"
result = client.get_exstats("extrahop.capture.custom_debug_detail",
                            "capture",
                            [(capture.oid, TFROM, TUNTIL)],
                            [],
                            {})

print "Dumping trigger runtime log in JSON format"
print json.dumps(result.stats, indent=2)

client.close()
```

See Also:

get_captures on page 29

get_exstats on page 58

pull-running-config.py

This script prints the current running configuration.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script prints the current running configuration.
"""

# Imports -----

from pyhop import pyhop

# Code -----

client = pyhop.make_client()
```

```
my_running_config = client.get_running_config()
print "-- Current Running Configuration --"
print my_running_config
```

See Also:

[get_running_config](#) on page 77

[set-host-name.py](#)

This script changes the hostname within the running config.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script changes the hostname within the running config.
"""

# Imports -----

from pyhop import pyhop

# Code -----

client = pyhop.make_client()

my_running_cfg = client.get_running_config()

my_running_cfg['hostname'] = {'hostname': "Py-Host"}

client.set_running_config(my_running_cfg)
client.close()

new_running_cfg = client.get_running_config()
print new_running_cfg['hostname']
```

See Also:

["get_running_config"](#) on page 77

["set_running_config"](#) on page 125

[ssl-key-expiration.py](#)

This script prints the name, common name and expiration of any SSL key on the ExtraHop that is expiring within 'expiration-days' or which has already expired.

```
#!/usr/bin/python
#-*- Mode: python -*-
# Copyright 2012-2015, ExtraHop Networks, Inc
```

```
import argparse
import re
from subprocess import Popen, PIPE, STDOUT
from datetime import datetime
from pyhop import pyhop

# What this script does:
# Print the name, common name, and expiration of any SSL key
# on the ExtraHop that is expiring within `expiration-days`
# or which has already expired.
#
# Run with '-h' for usage information

parser = argparse.ArgumentParser(
    description='Print information about SSL keys expiring soon')
parser.add_argument('host', help='Hostname of the ExtraHop')
parser.add_argument('apikey', help='ApiKey to use when connecting to ExtraHop')
parser.add_argument('-d', '--days', type=int, dest='days', default=30,
                    help='Number of days before ticket expiration')

args = parser.parse_args()

now = datetime.now()
timefmt_in = "%b %d %H:%M:%S %Y %Z"
command = ['openssl', 'x509',
           '-noout',
           '-subject',
           '-enddate',
           '-sha1', '-fingerprint'
          ]

def findmatch(section, string):
    m = re.search('/' + section + '=(^[^/]+)', string)
    return m.group(1) if m else None

c = pyhop.Client(host=args.host, apikey=args.apikey)
for key in c.get_ssl_decrypt_configs():
    p = Popen(command, stdout=PIPE, stdin=PIPE, stderr=STDOUT)
    result = p.communicate(input=key['cert_pem'])[0].strip().split("\n")

    if len(result) != 3:
        print("Failed to parse certificate with name " +
              "%s" and id '%s'" % (key['name'], key['id']))
    else:
        subject_cn = findmatch("CN", result[0])
        enddate = (datetime.strptime(result[1].split('=')[1], timefmt_in))
        fingerprint = result[2].split('=')[1]

        delta = enddate - now
        if delta.days <= args.days:
            print("Key expiring in %d days:" % delta.days)
            print("  Fingerprint: %s" % fingerprint)
```

```
print(" Subject CN: %s" % subject_cn)
print(" End Date: %s" % enddate)
```

See Also:

[get_ssl_decrypt_configs](#) on page 78

[topn-dset.py](#)

This script queries topn dset type statistics.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script queries topN dset type stats.
"""

# Imports -----

from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TFROM = -1800000
TUNTIL = 0

# Code -----

client = pyhop.make_client()

stat_name = "extrahop.device.app_turn"
object_type = "activity_group"
object_spec = [("extrahop.device.tcp", TFROM, TUNTIL)]
field_spec = [":summary:tprocess"]
opts = {}

result = client.get_exstats_group(stat_name,
                                  object_type,
                                  object_spec,
                                  field_spec,
                                  opts)

# Print out the devices and their stats of interest.
for stat in result.stats:
    print ">>> Device:", stat.oid, "<" * 40
    for x in stat[field_spec[0]]:
        print x.key.str, x.value.min, x.value.q2, x.value.max
```

```
client.close()
```

See Also:

get_exstats_group on page 63

topn-sset.py

This script queries topn sset type statistics.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script queries topN sset type stats.
"""

# Imports -----

import math
from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
TFROM = -1800000
TUNTIL = 0

# Class Definitions -----

class SampleSet():
    def __init__(self, sset):
        self.count = sset.count
        self.sum = sset.sum
        self.sum2 = sset.sum2

    @property
    def mean(self):
        if self.count < 1:
            return 0.0
        return self.sum / float(self.count)

    @property
    def sigma(self):
        if self.count < 1:
            return 0.0
        sigma2 = ((self.sum2 / self.count) -
                  ((self.sum / float(self.count)) ** 2))
```



```
        return math.sqrt(max(sigma2, 0))

# Code -----

client = pyhop.make_client()

stat_name = "extrahop.device.uri_http_server_detail"
object_type = "activity_group"
object_spec = [("extrahop.device.http_server", TFROM, TUNTIL)]
field_spec = ["tprocess"]
opts = {}

result = client.get_exstats_group(stat_name,
                                  object_type,
                                  object_spec,
                                  field_spec,
                                  opts)

for stat in result.stats:
    print ">>> Device:", stat.oid, "<" * 40
    for x in stat[field_spec[0]]:
        sset = SampleSet(x.value)
        print x.key.str, "=>", "mean:", sset.mean, "sigma:", sset.sigma

client.close()
```

See Also:

[get_exstats_group](#) on page 63

[whitelist.py](#)

This script adds devices matching the search string and whitelists them. After whitelisting them, it removes them from the whitelist.

```
#!/usr/bin/env python2
#-*- Mode: python -*-

"""
Copyright 2015, ExtraHop Networks, Inc

This script adds devices matching the search string and whitelists them.
After whitelisting them, it then removes them from the whitelist.
"""

# Imports -----

from pyhop import pyhop

# Constants -----

# Using a negative number indicates time (in ms) relative to the current
# capture time. Setting TUNTIL to '0' represents using the current
# capture time however, setting TFROM to '0' represents UNIX time 0.
```

```
TFROM = -1800000
TUNTIL = 0

SEARCH_STRING = "apple"

# Function Definitions -----

def show_apple_whitelist():
    print ">>> apple whitelisted devices:"
    options = {"devtype": "whitelist"}
    result = client.\
        search_devices_by_name(TFROM, TUNTIL, SEARCH_STRING, options)

    print "total:", result.total
    for d in result.devices:
        print d.name, d.macaddr

# Code -----
client = pyhop.make_client()

show_apple_whitelist()

print ">>> apple L3 devices:"
options = {"devtype": "l3"}
result = client.\
    search_devices_by_name(-TFROM, TUNTIL, SEARCH_STRING , options)

print "total:", result.total
for d in result.devices:
    print d.name, d.macaddr

print ">>> adding all L3 apple devices to white list:"
oids = [d.oid for d in result.devices]
result = client.add_to_whitelist(oids)

show_apple_whitelist()

print ">>> removing all L3 apple devices to white list:"
result = client.remove_from_whitelist(oids)

show_apple_whitelist()

client.close()
```

See Also:

add_to_whitelist on page 96

"remove_from_whitelist" on page 120

search_devices_by_name on page 148