CHAPTER 7

Search API

This chapter describes the Search API:

Overview

All search interfaces return a queryId and a queryStatus, in addition to the actual search results. The queryId may be used later for retrieving more results from the same search. The queryStatus contains an integer status code and a string message about the code.

The search API provides the following interfaces:

- DeviceQueryResult searchDevices(String query, List<String> fieldNames, String queryId, Long count, Long offset)
- GroupQueryResult getGroups(String groupType)
- DeviceDetailQueryResult getDeviceDetails(String query, List<String> deviceIds, String queryId, Long count, Long offset)
- MetricHistoryQueryResult getMetricHistory(String query, List<String> deviceIds, Date startTime, Date endTime, List<String> metricIds, String rollupInterval, String rollupFunction, String queryId, Long count, Long offset)

Using the Search API

In your CG-NMS NB API client application, use this CG-NMS server URL to access the Search API WSDL:

http://<*server_address*>/nbapi/search?wsdl

These are examples of client applications using the Search API:

Example 1 (Python-Suds Client)

```
from suds.client import Client
import logging
logging.basicConfig(level=logging.INFO)
logging.getLogger('suds.client').setLevel(logging.INFO)
url = "http://localhost/nbapi/search?wsdl"
cl = Client(url, username='root', password='Tree123!')

globalMetrics = ['uptime']
cgmeshMetrics =
['meshHops','meshLinkCost','meshPathCost','meshRssi','meshReverseRssi','meshRxSpeed','mesh
TxSpeed']
cgr1kStatus =
['doorStatus','numBBU','battery0Level','battery0Runtime','battery1Level','battery1Runtime'
,'battery2Level','battery2Runtime']
cgr1kMetrics = ['chassisTemp','ethernetTxDrops','ethernetRxSpeed','ethernetTxSpeed']
cgr1kCellIfMetrics = ['cellularRSSI','cellularRxSpeed','cellularTxSpeed']
cgr1kWiMAXIfMetrics = ['wimaxRSSI','wimaxRxSpeed','wimaxTxSpeed']
cgr1kMeshIfMetrics = ['meshEndpointCount','meshRxSpeed','meshTxSpeed']

# search FAN router devices
devices = []
result = cl.service.searchDevices('deviceType:cgr1000
status:down',['lng','lat','ip'],'',1000,0)
#print result
if result.queryStatus == "SUCCEEDED":
  print "eid,lng,lat,ip,"
  for d in result.devices:
    devices.append(d.eid)
    print str(d.eid)+",",
    for f in d.fields.entry:
      print str(f.value)+",",
    print ""
```

Example 2 (Python-Suds Client)

```
from suds.client import Client
import logging
logging.basicConfig(level=logging.INFO)
logging.getLogger('suds.client').setLevel(logging.INFO)
url = "http://localhost/nbapi/search?wsdl"
cl = Client(url, username='root', password='Tree123!')
#print cl

# search mesh end devices
```

```
devices = []
result = cl.service.searchDevices('deviceType:cgmesh',['lng','lat','ip'],'',10,0)
#print result
if result.queryStatus == "SUCCEEDED":
  print "eid,lng,lat,ip,"
  for d in result.devices:
    devices.append(d.eid)
    print str(d.eid)+",",
    for f in d.fields.entry:
      print str(f.value)+",",
    print ""

globalMetrics = ['uptime']
cgmeshMetrics =
['meshHops','meshLinkCost','meshPathCost','meshRssi','meshReverseRssi','meshRxSpeed','mesh
TxSpeed']
cgr1kStatus =
['doorStatus','numBBU','battery0Level','battery0Runtime','battery1Level','battery1Runtime'
,'battery2Level','battery2Runtime']
cgr1kMetrics = ['chassisTemp','ethernetTxDrops','ethernetRxSpeed','ethernetTxSpeed']
cgr1kCellIfMetrics = ['cellularRSSI','cellularRxSpeed','cellularTxSpeed']
cgr1kWiMAXIfMetrics = ['wimaxRSSI','wimaxRxSpeed','wimaxTxSpeed']
cgr1kMeshIfMetrics = ['meshEndpointCount','meshRxSpeed','meshTxSpeed']


# get device metric history
result =
cl.service.getMetricHistory('status:up',[str(devices[0]),str(devices[1]),str(devices[2])],
'2012-04-01 00:00:00','2012-04-01 23:59:59',cgmeshMetrics,'day','avg','',2,0)
#print result
if (result.queryStatus == "SUCCEEDED"):
  try:
    result.metricValues
  except:
    print "no metrics returned"
  else:
    print "eid,metric,value,timestamp,"
    for m in result.metricValues:
        print str(m.eid), ",", str(m.metricId),",", str(m.value),",", str(m.timestamp)

# search FAN router devices
devices = []
result = cl.service.searchDevices('deviceType:cgr1000',['lng','lat','ip'],'',10,0)
#print result
if result.queryStatus == "SUCCEEDED":
  print "eid,lng,lat,ip,"
  for d in result.devices:
    devices.append(d.eid)
    print str(d.eid)+",",
    for f in d.fields.entry:
      print str(f.value)+",",
    print ""

# get device metric history
for f in devices:
  result = cl.service.getMetricHistory('status:up',f,'2012-04-01 00:00:00','2012-04-01
23:59:59',cgr1kMetrics+cgr1kCellIfMetrics+cgr1kWiMAXIfMetrics+cgr1kMeshIfMetrics,'day','av
g','',0,0)
  #print result
  if (result.queryStatus == "SUCCEEDED"):
    try:
      result.metricValues
    except:
      print "no metrics returned"
```

```
    else:
      print "eid,metric,value,timestamp,"
      for m in result.metricValues:
        print str(m.eid), ",", str(m.metricId),",", str(m.value),",", str(m.timestamp)
```

Example 3 (Python-Suds Client)

```
import sys
from suds.client import Client
import logging
logging.basicConfig(level=logging.INFO)
logging.getLogger('suds.client').setLevel(logging.DEBUG)
url = "http://localhost/nbapi/search?wsdl"
cl = Client(url, username='root', password='Tree123!')

# get config groups
groups =
['DeviceType','Status','ConfigGroup','FirmwareGroup','TunnelProvisioningGroup','Label']
for g in groups:
  result = cl.service.getGroups(g)
  #print result
  if result.queryStatus == "SUCCEEDED":
    print g
    for r in result.groups:
      print r.name,
      if (r.properties != None and r.properties != ""):
        for p in r.properties.entry:
          if (p.key == "description" or p.key == "deviceType"):
       print p.value,
      if r.memberCount != None:
        print "x " + str(r.memberCount) + ",",
      print ""
    print ""

# search devices
devices = []
result = cl.service.searchDevices('status:up',['lng','lat','ip'],'',10,0)
#print result
if result.queryStatus == "SUCCEEDED":
  print "eid,lng,lat,ip,"
  for d in result.devices:
    devices.append(d.eid)
    print str(d.eid)+",",
    for f in d.fields.entry:
      print str(f.value)+",",
    print ""

# fetch mesh endpoint node device details
metrics =
("uptime","nodeLocalTime","meshHops","meshRssi","meshReverseRssi","meshLinkCost","meshPath
Cost","meshRxSpeed","meshTxSpeed","meshTxDrops")
properties =
("sn","deviceType","pid","lng","lat","alt","geoHash","mapLevel","lastHeard","status","conf
igInSync","runningFirmwareVersion","hardwareId","vid","certSN","certL","certO","certCN","c
ertST","certOU","certC")
devices = []
result = cl.service.searchDevices('deviceType:cgmesh','','',10,0)
if result.queryStatus == "SUCCEEDED":
  for d in result.devices:
    devices.append(d.eid)
  print devices
if len(devices) != 0:
```

```
  fn = "./cgmeshDevDetails.csv"
  f = open(fn, 'w')
  #fetch device details
  print >>f, "eid,configGroup,firmwareGroup,",
  for p in properties:
    print >>f, p+",",
  for m in metrics:
    print >>f, m+",",
  print >>f, "interfaces"
  for d in devices:
    result = cl.service.getDeviceDetails('',d,'',0,0)
    #print result
    if result.queryStatus == "SUCCEEDED":
      print >>f,
d+","+result.deviceDetails[0].configGroup+","+result.deviceDetails[0].firmwareGroup+",",
      for p in properties:
        for j in result.deviceDetails[0].properties.entry:
      if j.key == p:
        try:
          j.value
        except:
          pass
        else:
          if j.value != "null":
            print >>f, j.value,
        print >>f, ",",
      for m in metrics:
        for j in result.deviceDetails[0].metrics.entry:
      if j.key == m:
        print >>f, j.value,
        print >>f, ",",
      for i in result.deviceDetails[0].interfaces:
        print >>f, i.name,
      print >>f, ""

# fetch FAN router details
metrics =
["uptime","chassisTemp","ethernetTxSpeed","ethernetRxSpeed","ethernetTxDrops","meshEndpoin
tCount","meshRxSpeed","meshTxSpeed"]
properties =
["sn","deviceType","pid","lng","lat","alt","geoHash","mapLevel","lastHeard","status","door
Status","bbuPresent","numBBU","configInSync","runningFirmwareVersion","hardwareId","vid","
certSN","certL","certO","certCN","certST","certOU","certC","meshPanidConfig","meshLinkKeyR
efresh","meshLinkKeyExpiration","meshPrefixConfig","powerSource","ip","hostname","domainNa
me","meshPrefixLengthConfig","meshPrefix","meshPrefixLength","meshAddressConfig","meshAddr
ess"]
devices = []
result = cl.service.searchDevices('deviceType:cgr1000','','',10,0)
#print result
if result.queryStatus == "SUCCEEDED":
  for d in result.devices:
    devices.append(d.eid)
  print devices
if len(devices) != 0:
  fn = "./cgr1kDevDetails.csv"
  f = open(fn, 'w')
  #fetch device details
  print >>f, "eid,configGroup,firmwareGroup,",
  for p in properties:
    print >>f, p+",",
  for m in metrics:
    print >>f, m+",",
  print >>f, "interfaces"
  for d in devices:
```

```
    result = cl.service.getDeviceDetails('',d,'',0,0)
    #print result
    if result.queryStatus == "SUCCEEDED":
      print >>f,
d+","+result.deviceDetails[0].configGroup+","+result.deviceDetails[0].firmwareGroup+",",
      for p in properties:
        for j in result.deviceDetails[0].properties.entry:
      if j.key == p:
        try:
          j.value
        except:
          pass
        else:
          if j.value != "null":
            print >>f, j.value,
        print >>f, ",",
      for m in metrics:
        for j in result.deviceDetails[0].metrics.entry:
      if j.key == m:
        print >>f, j.value,
        print >>f, ",",
      for i in result.deviceDetails[0].interfaces:
        print >>f, i.name,
      print >>f, ""
```

Using the search-client Script

The CG-NMS distribution provides a Search API client (search-client script) contained in the package cgms-tools-*version*.x86_64.rpm. The client itself is wrapped by a shell script (/opt/cgms-tools/bin/search-client), which you can use directly for communicating with the CG-NMS API.

```
[root@localhost bin]# ./search-client
usage:
    ./search-client device <endpoint URL> <query> <field names> <count> <offset>
    ./search-client device <endpoint URL> <query Id>
    ./search-client deviceDetail <endpoint URL> <query> <count> <offset>
    ./search-client deviceDetail <endpoint URL> <query Id>
    ./search-client deviceDetail <endpoint URL> <device eid list>
    ./search-client group <endpoint URL> <group type>
    ./search-client metricHistory <endpoint URL> <query> <startTime> <endTime>
                         <field names> <rollupInterval> <rollupFunction> <count> <offset>
    ./search-client metricHistory <endpoint URL> <queryId>
    ./search-client metricHistory <endpoint URL> <device eid list> <startTime> <endTime>
                         <field names> <rollupInterval> <rollupFunction>
```

The search-client script defines four commands:

Table 7-1 search-client Commands

Command	Description
device	Calls the searchDevices API.
deviceDetail	Calls the getDeviceDetails API.
group	Calls the getGroups API.
metric	Calls the getMetricHistory API.

Search API Methods

searchDevices

The searchDevices call enables the client to provide a search query string (using the query language) and a list of device details (properties or metrics) to return.

searchDevices SOAP XML Request Format

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sear="http://search.nbapi.cgms.cisco.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <sear:searchDevices>
         <!--Optional:-->
         <query>deviceType:cgr1000</query>
         <!--Zero or more repetitions:-->
         <fieldNames></fieldNames>
         <!--Optional:-->
         <queryId></queryId>
         <!--Optional:-->
         <count>10</count>
         <!--Optional:-->
         <offset>0</offset>
   </sear:searchDevices> </soapenv:Body>
</soapenv:Envelope>
```

Parameters

Table 7-2 searchDevices Parameters

Parameter	Type	Description
query	string	Search query string.
fieldNames	string	List of property or metric names to return in the results for each device. For more information about those properties and metrics, see Property Field Names for All Devices, page 1-4 and Metrics Field Names, page 1-7
queryId	string	Query ID returned by the call. If available results for the query is more than count, the caller can use the returned queryId to call the same API repeatedly. When queryId is provided, all other parameters are ignored.
count	integer	Number of results to be retrieved.
offset	integer	Position of the first result.

Results

The searchDevices call returns a list of device details, described in Table 7-3.

Table 7-3 searchDevices Results

Name	Type	Description
eid	string	Device ID.
fields	map<String.String>	A list of fieldname-value pairs.
key	long	A long value associated with the device.

getGroups

The getGroups call enables the client to retrieve the following:

- List of groups for a specific group type within the system
- Current number of members in each group
- Properties assigned to the group

Group types can be one of the following strings:

- deviceType
- status
- label
- subnet
- config
- firmware

getGroups SOAP XML Request Format

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sear="http://search.nbapi.cgms.cisco.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <sear:getGroups>
         <!--Optional:-->
         <groupType>devicetype</groupType>
      </sear:getGroups>
   </soapenv:Body>
</soapenv:Envelope>
```

Parameters

Table 7-4 getGroups Parameters

Parameter	Type	Description
groupType	string	Group type name.

Results

This method returns a list of groups, described in Table 7-5.

Table 7-5 GetGroups Results

Name	Type	Description
name	string	Name of the group.
memberCount	integer	Number of devices in the group.
properties	Map<String, String>	Property name-value pairs.

getDeviceDetails

The getDeviceDetails call enables the client to retrieve the following:

- All device properties
- Current metrics
- Assigned labels
- Interface information
- Addressing information
- Routing information about a device

Devices can be retrieved by specifying a query or by specifying a list.

getDeviceDetails SOAP XML Request Format

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sear="http://search.nbapi.cgms.cisco.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <sear:getDeviceDetails>
         <!--Optional:-->
         <query>far_test2</query>
         <!--Zero or more repetitions:-->
         <deviceIds>+JSJ1522003G</deviceIds>
         <!--Optional:-->
         <queryId></queryId>
         <!--Optional:-->
         <count>5</count>
         <!--Optional:-->
         <offset>0</offset>
      </sear:getDeviceDetails>
   </soapenv:Body>
</soapenv:Envelope>
```

Parameters

Table 7-6 getDeviceDetails Parameters

Parameter	Type	Description
query	string	Search query.
deviceIds	list<String>	List of device IDs for which the result will be retrieved.
queryId	string	ID returned by the call.
count	integer	Number of results to be retrieved.
offset	integer	Position of the first result.

Results

This interface returns a list of device details, described in Table 7-7.

Table 7-7 getDeviceDetails Results

Parameter	Type	Description
eid	string	String ID of the device.
properties	Map<String, String>	PropertyName-value pairs.
metrics	Map<String, Double>	MetricType-value pairs.
labels	List<String>	List of labels assigned to the device.
subnetGroup	string	Name of the subnet group assigned to the device.
configGroup	string	Name of the configuration group assigned to the device.
firmwareGroup	string	Name of the firmware group assigned to the device.
interfaces	List<interface>	List of interfaces of the device.
routes	List<Route>	List of routes known to the device.
assets	List<Asset>	List of assets installed on the device.
addresses	List<Address>	List of addresses known to the device

Note For detailed information on Interface, Route, Asset, and Address, see the Cisco 1000 Series Connected Grid Router software configuration guides, at www.cisco.com/go/cgr1000-docs.

getMetricHistory

The getMetricHistory call enables the client to retrieve the historical metric values saved in the CG-NMS database. In the call, you can specify:

- Single device
- List of devices
- Query that returns devices

You can also specify a rollup interval from the following:

- none
- hour
- day

And a rollup function from the following:

- avg
- max
- min

getMetricHistory SOAP XML Request Format

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sear="http://search.nbapi.cgms.cisco.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <sear:getMetricHistory>
         <!--Optional:-->
         <query>deviceType:cgr1000</query>
         <!--Zero or more repetitions:-->
        <deviceIds>far_test3</deviceIds>
         <!--Optional:-->
         <startTime>null</startTime>
         <!--Optional:-->
         <endTime>null</endTime>
         <!--Zero or more repetitions:-->
         <metricIds>1</metricIds>
         <!--Optional:-->
         <rollupInterval>none</rollupInterval>
         <!--Optional:-->
         <rollupFunction>min</rollupFunction>
         <!--Optional:-->
         <queryId></queryId>
         <!--Optional:-->
         <count>5</count>
         <!--Optional:-->
         <offset>1</offset>
      </sear:getMetricHistory>
   </soapenv:Body>
</soapenv:Envelope>
```

Parameters

Table 7-8 getMetricHistory Parameters

Parameter	Type	Description
query	string	Search query string
deviceIds	list<String>	List of device IDs for which the result will be retrieved.
startTime	date	Starting UTC date and time for the metric history interval; use null if not defined.
endTime	date	Ending UTC date and time for the metric history interval; use null if not defined.
metricIds	List<String>	List of metric type names, desired to be filled in the result for each device.
rollupInterval	string	Can be one of *none*, *hour*, or *day*
rollupFunction	string	Can be one of *avg*, *max*, or *min*
queryId	string	ID returned by the call
count	integer	Number of results to be retrieved
offset	integer	Position of the first result

Results

This interface returns a list of metric values, as defined in Table 7-9.

Table 7-9 getMetricHistory Results

Parameter	Type	Description
eid	string	String ID of the device
metricId	string	Metric type name
timestamp	date	UTC timestamp of the metric value
value	double	Value of the metric