Search API

This chapter describes the Search API:

Overview

All search interfaces return a queryStatus, in addition to the actual search results. Search interfaces with count and offset arguments also return a queryId. The queryId is useful 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:

- DeviceDetailQueryResult getDeviceDetails(string query, list<string> deviceIds, string queryId, long count, long offset)
- DeletedDeviceQueryResult getDeletedDevices(string deviceType, date startTime, date endTime)
- DeviceQueryResult searchDevices(string query, list<string> fieldNames, string queryId, long count, long offset)
- EidsForIpAddressesResult findEidsForIpAddressesByDeviceType(string deviceType, list<string> ipAddresses)
- EidsForIpAddressesResult findEidsForIpAddresses(list<string> ipAddresses)
- GroupQueryResult getGroups(string groupType)
- MetricHistoryQueryResult getMetricHistory(string query, list<string> deviceIds, date startTime, date endTime, List<string> metricIds, string rollupInterval, string rollupFunction, string queryId, long count, long offset)
- UpdatedDeviceDetailQueryResult getUpdatedDeviceDetails(string query, date startTime, date endTime, string queryId, long count, long offset)

Using the Search API

Use this IoT FND server URL to access the Search API WSDL:

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

Provided are these 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','meshTxSpeed']
cgr1kStatus =
['doorStatus','numBBU','battery0Level','battery0Runtime','battery1Level','battery1Runtime','battery2Lev
el','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','meshTxSpeed']
cgr1kStatus =
['doorStatus','numBBU','battery0Level','battery0Runtime','battery1Level','battery1Runtime','battery2Lev
el','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','avg','',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","meshPathCost","meshRx
Speed","meshTxSpeed","meshTxDrops")
properties =
("sn","deviceType","pid","lng","lat","alt","geoHash","mapLevel","lastHeard","status","configInSync","ru
nningFirmwareVersion","hardwareId","vid","certSN","certL","certO","certCN","certST","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","meshEndpointCount","mesh
RxSpeed","meshTxSpeed"]
properties =
["sn","deviceType","pid","lng","lat","alt","geoHash","mapLevel","lastHeard","status","doorStatus","bbuP
resent","numBBU","configInSync","runningFirmwareVersion","hardwareId","vid","certSN","certL","certO","c
ertCN","certST","certOU","certC","meshPanidConfig","meshLinkKeyRefresh","meshLinkKeyExpiration","meshPr
efixConfig","powerSource","ip","hostname","domainName","meshPrefixLengthConfig","meshPrefix","meshPrefi
xLength","meshAddressConfig","meshAddress"]
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 IoT FND distribution provides a Search API client (search-client script) in the cgms-tools-*version*.x86_64.rpm package. The client is wrapped by a shell script (/opt/cgms-tools/bin/search-client) for communicating with the IoT FND API.

```
[root@localhost bin]# ./search-client
usage:
./search-client device <endpoint URL> <query> <field names> <count> <offset>
    ./search-client device <endpoint URL> <query Id> <field names>
    ./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 updatedDeviceDetail <endpoint URL> <query> <startTime> <endTime> <count> <offset>
    ./search-client updatedDeviceDetail <endpoint URL> <queryId>
    ./search-client deletedDevice <endpoint URL> <deviceType> <startTime> <endTime>
    ./search-client group <endpoint URL> <group type>
            <group type> can be deviceType, label, status, configGroup, firmwareGroup, tunnelGroup or
            subnetGroup
    ./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>
    ./search-client eidByAddresses <endPoint URL> <deviceIps>
    ./search-client ReprovisionByEids <endPoint URL> <groupName> <deviceEids> <interfaceName>
                                      <interfaceType>
```

Table 1 lists search-client script commands.

Table 1 search-client Commands

Command	Description
device	Calls the searchDevices API.
deviceDetail	Calls the getDeviceDetails API.
deletedDevice	Calls the getDeletedDevices API.
eidByAddresses	Calls the findEidsForIpAddresses API.
event	Calls the getUpdatedDeviceDetails API.
group	Calls the getGroups API.
issues	Calls the getUpdatedDeviceDetails API.
meshFirmwareUpgrade	Calls the getUpdatedDeviceDetails API.
metricHistory	Calls the getMetricHistory API.
updatedDeviceDetail	Calls the getUpdatedDeviceDetails API.
work order	Calls the getUpdatedDeviceDetails API.

Search API Method Calls

searchDevices

This call lets the client provide a search query string (using the query language) and returns a list of device details (properties or metrics).

Prototype

```
DeviceQueryResult searchDevices(string query, list<string> fieldNames, string queryId, long count,
                                long offset)
```

Parameters

Table 2 searchDevices Parameters

Parameter	Type	Description
query	string	Search query string.
fieldNames	string	List of property or metric names to retrieve for each device. For information about properties and metrics, see Property Field Names for All Devices, page 6 and Metrics Field Names, page 8.
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	long	Number of results to retrieve. Valid range is 1–40000.
offset	long	Position of the first result. Valid values are >= 0.

Results

Table 3 lists device details returned by this call.

Table 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.

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>
```

getGroups

This call lets the client 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 are any one of the following strings:

- deviceType
- status
- label
- configGroup
- firmwareGroup
- subnetGroup
- tunnelProvisioningGroup

Prototype

```
GroupQueryResult getGroups(string groupType)
```

Parameters

Table 4 getGroups Parameters

Parameter	Type	Description
groupType	string	Group type strings from the above list.

Results

Table 5 lists groups returned by this call.

Table 5 getGroups Results

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

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>
```

getDeviceDetails

This call lets the client retrieve the following information:

- Address list information:
  - Address
  - addressType
  - Key
  - prefixLength
- Assets list information:
  - Key
  - Name
  - Asset metrics
  - Asset properties
- ConfigGroup as a string
- EID as a string
- FirmwareGroup as a string
- Interface list information:
  - Interface key
  - Interface name
  - Interface index
  - Interface addresses list
  - Interface metrics
  - Interface properties
- Key as a long integer
- Labels list
- Metrics list
- Properties list
- SubnetGroup as a string
- Route list information:
  - Index
  - Key
  - Route metrics
  - Route properties

Devices are retrieved by specifying a query or a list. For example, the Route properties key-value pair information retrieved is:

```
destAddressType:2
nextHopAddressType:4
nextHopAddress : fe80:0:0:0:207:8108:3c:270b
destAddress:0:0:0:0:0:0:0:0
prefixLength:0
```

Prototype

```
DeviceDetailQueryResult getDeviceDetails(string query, list<string> deviceIds, string queryId,
                                         long count, long offset)
```

Parameters

Table 6 getDeviceDetails Parameters

Parameter	Type	Description
query	string	Search query.
deviceIds	list<string>	List of device EIDs to retrieve. When deviceIds is provided, the query parameter is ignored.
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	long	Number of results to retrieve. Valid range is 1–40000
offset	long	Position of the first result. Valid values are >= 0.

Results

Parameters returned by this call are listed in Table 7.

Table 7 getDeviceDetails Results

Parameter	Type	Description
addresses	list<address>	A list of addresses known to the device. See Table 8.
assets	list<asset>	A list of assets installed on the device. See Table 11.
configGroup	string	The name of the configuration group assigned to the device.
eid	string	The string ID of the device.
firmwareGroup	string	The name of the firmware group assigned to the device.
interfaces	list<interface>	A list of interfaces of the device. See Table 9.
labels	list<string>	The list of labels assigned to the device.
key	long	A long value associated with the device.
metrics	map<string, double>	Metric type-value pairs. See Metrics Field Names.
properties	map<string, string>	Property name-value pairs. See Property Field Names for All Devices.
routes	list<route>	A list of routes known to the device. See Table 10.
subnetGroup	string	The name of the subnet group assigned to the device.

Table 8 describes the Address attributes.

Table 8 Address Attributes

Attribute	Type	Description
key	long	A long value associated with the address.
addressType	string	Value is one of these address types: IPV4, IPV6, IPV4Z, IPV6Z or UNKNOWN.
address	string	The IP address.
prefixLength	integer	Subnet prefix length of the address.

Table 9 describes the Interface attributes.

Table 9 Interface Attributes

Attribute	Type	Description
key	long	A long value associated with the interface.
index	integer	Index value of the interface.
name	string	The name of the interface.
addresses	list<address>	List of Address (see Table 8) associated with the interface.
properties	map<string, string>	Properties of the interface in name-value pairs.
metrics	map<string, double>	Metrics of the interface.

Table 10 describes the Route attributes.

Table 10 Route Attributes

Attribute	Type	Description
key	long	A long value associated with the route.
index	integer	Index value of the route.
properties	map<string, string>	Properties of the route in name-value pairs.
metrics	map<string, double>	Metrics of the route.

Table 11 describes the Asset attribute parameters.

Table 11 Asset Attributes

Attribute	Type	Description
key	long	A long value associated with the asset.
name	string	The name of the asset.
properties	map<string, string>	Properties of the asset in name-value pairs.
metrics	map<string, double>	Metrics of the asset.

getDeviceDetails SOAP XML Request Format for a FAR

```
<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>
```

getDeviceDetails SOAP XML Request Format for a Mesh Device

```
<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>deviceType:cgmesh status:up</query>
         <!--Zero or more repetitions:-->
         <deviceIds></deviceIds>
         <!--Optional:-->
         <queryId></queryId>
         <!--Optional:-->
         <count>10</count>
         <!--Optional:-->
         <offset>0</offset>
      </sear:getDeviceDetails>
   </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
   <env:Header>
      <seam:conversationId xmlns:seam="http://www.jboss.org/seam/webservice">1125</seam:conversationId>
   </env:Header>
   <env:Body>
      <ns2:getDeviceDetailsResponse xmlns:ns2="http://search.nbapi.cgms.cisco.com/">
         <deviceDetailQueryResult>
            <queryId>E6FDA0F0D3E0ADFFF69E334462D1EF6A</queryId>
            <queryStatus>SUCCEEDED</queryStatus>
         </deviceDetailQueryResult>
      </ns2:getDeviceDetailsResponse>
   </env:Body>
</env:Envelope>
```

getDeletedDevices

This call retrieves information of devices deleted within a time period.

Prototype

```
DeletedDeviceQueryResult getDeletedDevices(string deviceType, date startTime, date endTime)
```

Parameters

Table 12 getDeletedDevices Parameters

Parameter	Type	Description
deviceType	string	The name of the device type to filter the results by. For example, cgr1000 only matches deleted devices against CGR devices.
startTime	date	Starting UTC date and time for the deletion interval. This is a mandatory parameter. It cannot be null. Format is YYYY-MM-DDThh:mm:ss (for example, 2015-03-25T00:00:00).
endTime	date	Ending UTC date and time for the deletion interval. This is a mandatory parameter. It cannot be null. Use it as the startTime for the next call without retrieving repeated records. Format is YYYY-MM-DDThh:mm:ss.

Results

Table 3 lists device details returned by this call.

Table 13 getDeletedDevices Results

Name	Type	Description
deleteDate	date	The date and time of device deletion.
deviceType	string	The device type of the deleted device.
eid	string	Device ID.

search-client Script Example

```
search-client deletedDevice https://kml-lnx2/nbapi/search?wsdl "cgr1000" "2015-03-25 12:00:00"
"2015-03-26 12:00:00"
```

search-client Script Results

```
URL: https://kml-lnx2/nbapi/search?wsdl
Mar 25, 2015 5:16:43 PM org.apache.cxf.service.factory.ReflectionServiceFactoryBean
buildServiceFromWSDL
INFO: Creating Service {http://search.nbapi.cgms.cisco.com/}SearchWebService from WSDL:
https://kml-lnx2/nbapi/search?wsdl
Mar 25, 2015 5:16:44 PM org.apache.cxf.service.factory.ReflectionServiceFactoryBean
buildServiceFromWSDL
INFO: Creating Service {http://search.nbapi.cgms.cisco.com/}SearchWebService from WSDL:
https://kml-lnx2/nbapi/search?wsdl
{"deleteDate":"Wed Mar 25 16:00:44 PDT 2015","deviceType":"cgr1000","eid":"CGR1240/K9+JSJ200201"}
{"deleteDate":"Wed Mar 25 16:00:44 PDT 2015","deviceType":"cgr1000","eid":"CGR1240/K9+JSJ200202"}

Elapsed Time: [169] ms
Status: Query succeeded
Rows: 2
```

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:getDeletedDevices>
         <deviceType>cgr1000</deviceType>
         <startTime>2015-03-25T00:00:00</startTime>
         <endTime>2015-03-26T00:00:00</endTime>
      </sear:getDeletedDevices>
   </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"/>
   <soap:Body>
      <ns2:getDeletedDevicesResponse xmlns:ns2="http://search.nbapi.cgms.cisco.com/">
         <deletedDeviceQueryResult>
            <queryStatus>SUCCEEDED</queryStatus>
            <deletedDevices>
               <deleteDate>2015-03-25T23:00:44Z</deleteDate>
               <deviceType>cgr1000</deviceType>
               <eid>CGR1240/K9+JSJ200201</eid>
            </deletedDevices>
            <deletedDevices>
               <deleteDate>2015-03-25T23:00:44Z</deleteDate>
               <deviceType>cgr1000</deviceType>
               <eid>CGR1240/K9+JSJ200202</eid>
            </deletedDevices>
         </deletedDeviceQueryResult>
      </ns2:getDeletedDevicesResponse>
   </soap:Body>
</soap:Envelope>
```

Example Python Request

```
#!/usr/local/bin/python
from suds.transport.https import HttpAuthenticated
from suds.client import Client
import logging
from datetime import timedelta,date,datetime,tzinfo
import requests
from requests.auth import HTTPBasicAuth
import datetime

#transport = HttpAuthenticated(username='root',password='PeterChen123!')
transport = HttpAuthenticated(username='root',password='Private123!')
#WSDL_URL= "https://172.27.126.110/nbapi/search/?wsdl"
WSDL_URL= "https://kml-lnx2/nbapi/search/?wsdl"
client = Client(WSDL_URL, faults=False,cachingpolicy=1,location=WSDL_URL,transport=transport)

print(client)
deviceType="cgr1000"
string_start_date = "2015-03-25 12:00:00.78200"
string_end_date = "2016-03-26 12:00:00.78200"
startTime = datetime.datetime.strptime(string_start_date, "%Y-%m-%d %H:%M:%S.%f")
endTime= datetime.datetime.strptime(string_end_date, "%Y-%m-%d %H:%M:%S.%f")
count=5
offset =0
searchthis = client.service.getDeletedDevices(deviceType,startTime,endTime)
print(searchthis)
```

Results

```
Suds ( https://fedorahosted.org/suds/ )  version: 0.6
```

```
Service ( SearchWebService ) tns="http://search.nbapi.cgms.cisco.com/"
   Prefixes (1)
      ns0 = "http://search.nbapi.cgms.cisco.com/"
   Ports (1):
      (SearchWebServicePort)
         Methods (8):
            findEidsForIpAddresses(xs:string[] ipAddresses)
            findEidsForIpAddressesByDeviceType(xs:string deviceType, xs:string[] ipAddresses)
            getDeletedDevices(xs:string deviceType, xs:dateTime startTime, xs:dateTime endTime)
            getDeviceDetails(xs:string query, xs:string[] deviceIds, xs:string queryId, xs:long count,
xs:long offset)
            getGroups(xs:string groupType)
            getMetricHistory(xs:string query, xs:string[] deviceIds, xs:dateTime startTime, xs:dateTime
endTime, xs:string[] metricIds, xs:string rollupInterval, xs:string rollupFunction, xs:string queryId,
xs:long count, xs:long offset)
            getUpdatedDeviceDetails(xs:string query, xs:dateTime startTime, xs:dateTime endTime,
xs:string queryId, xs:long count, xs:long offset)
            searchDevices(xs:string query, xs:string[] fieldNames, xs:string queryId, xs:long count,
xs:long offset)
         Types (36):
            Exception
            address
            asset
            deletedDevice
            deletedDeviceQueryResult
            device
            deviceDetail
            deviceDetailQueryResult
            deviceQueryResult
            eidsForIpAddressesResult
            findEidsForIpAddresses
            findEidsForIpAddressesByDeviceType
            findEidsForIpAddressesByDeviceTypeResponse
            findEidsForIpAddressesResponse
            getDeletedDevices
            getDeletedDevicesResponse
            getDeviceDetails
            getDeviceDetailsResponse
            getGroups
            getGroupsResponse
            getMetricHistory
            getMetricHistoryResponse
            getUpdatedDeviceDetails
            getUpdatedDeviceDetailsResponse
            group
            groupQueryResult
            interface
            metricHistoryQueryResult
            metricValue
            queryResult
            queryStatus
            route
            searchDevices
            searchDevicesResponse
            updatedDeviceDetail
            updatedDeviceDetailQueryResult


(200, (deletedDeviceQueryResult){
   queryStatus = "SUCCEEDED"
   deletedDevices[] =
      (deletedDevice){
         deleteDate = 2015-03-25 23:00:44+00:00
         deviceType = "cgr1000"
         eid = "CGR1240/K9+JSJ200201"
```

```
      },
      (deletedDevice){
         deleteDate = 2015-03-25 23:00:44+00:00
         deviceType = "cgr1000"
         eid = "CGR1240/K9+JSJ200202"
      },
})
```

getUpdatedDeviceDetails

This call lets the client retrieve the following information of devices updated within a specified time period:

- key
- eid
- configGroup
- event
- firmwareGroup
- issues
- Labels list
- meshDeviceOps
- workorder
- Updated Properties list
- Updated Metrics list
- Updated Interfaces list
- Updated Routes list
- Updated Addresses list
- Deleted Interfaces list
- Deleted Routes list
- Deleted Addresses list

Prototype

```
UpdatedDeviceDetailQueryResult getUpdatedDeviceDetails(string query, date startTime, date endTime,
                                                       string queryId, long count, long offset)
```

Parameters

Table 14 getUpdatedDeviceDetails Parameters

Parameter	Type	Description
query	string	Search query.
startTime	date	Starting UTC date and time for the updated interval. This is a mandatory parameter. It cannot be null. Format is YYYY-MM-DDThh:mm:ss (for example, 2015-03-25T00:00:00).
endTime	date	Ending UTC date and time for the updated interval. This is a mandatory parameter. It cannot be null. Format is YYYY-MM-DDThh:mm:ss. endTime is exclusive. It can be used as the startTime for the next call without retrieving repeated records.
queryId	string	The ID of the initial query used in subsequent calls to retrieve remaining records.
count	long	Number of results to retrieve. Valid range is 1–40000.
offset	long	Position of the first result. Valid values are >= 0.

This call returns the records of devices that match the specified deviceType and have delete time >= startTime and < endTime.

Results

Table 15 lists the device details returned by this call. The return message, UpdatedDeviceDetailQueryResult, is a list of UpdatedDeviceDetail records.

Table 15 getUpdatedDeviceDetails Results

Parameter	Type	Description
configGroup	string	The name of the configuration group assigned to the device.
deletedAddresses	list<string>	Deleted IP addresses.
deletedInterfaces	list<string>	Deleted interface names.
deletedRoutes	list<string>	Deleted route names.
eid	string	Device string ID.
event	string	Search for events based on device type, event name, event time, and event severity.
firmwareGroup	string	The name of the firmware group assigned to the device.
issues	string	Call searches for issues in the IoT FND database based on the provided criteria that return a list of those issues.
key	long	A long value associated with the device.
meshDeviceOps	string	Allows client applications to start and stop firmware uploads to a group of mesh devices, check their firmware upload status, obtain firmware information by firmware group, set up a backup firmware image, and schedule a firmware reload.
workorder	string	Provides seven primary action calls to cloud-based management services.
updatedAddresses	list<address>	The list of addresses of the device updated during the specified time interval.
updatedInterfaces	list<interface>	The interface of the device updated during the specified time interval.
updatedMetrics	map<string, double>	The metrics of the device updated during the specified time interval.
updatedProperties	map<string, string>	The properties of the device updated during the specified time interval.
updatedRoutes	list<route>	The routes of the device updated during the specified time interval.

See Table 8, Table 9, and Table 10 for information on the Address, Interface, and Route attributes.

Example

```
search-client updatedDeviceDetails https://kml-lnx2/nbapi/search?wsdl "deviceCategory=router"
"2015-03-25 12:00:00" "2015-03-26 12:00:00" 5 0
search-client updatedDeviceDetails https://kml-lnx2/nbapi/search?wsdl "deviceCategory=router"
"2015-03-26 00:00:00" "2015-03-26 00:20:00" 2 0
```

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sear="http://search.nbapi.cgms.cisco.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <sear:getUpdatedDeviceDetails>
         <query>deviceCategory=router</query>
         <startTime>2015-03-25T23:00:00</startTime>
         <endTime>2015-03-26T00:00:00</endTime>
         <count>1</count>
         <offset>0</offset>
      </sear:getUpdatedDeviceDetails>
   </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"/>
   <soap:Body>
      <ns2:getUpdatedDeviceDetailsResponse xmlns:ns2="http://search.nbapi.cgms.cisco.com/">
         <updatedDeviceDetailQueryResult>
            <queryId>E1EA07CE9BE5536A883469707FCA602E</queryId>
            <queryStatus>SUCCEEDED</queryStatus>
            <updatedDeviceDetails>
               <configGroup>default-c800</configGroup>
               <eid>C819HGW-S-A-K9+FTX174685VB</eid>
               <firmwareGroup>default-c800</firmwareGroup>
               <key>270000</key>
               <updatedInterfaces>
                  <index>1</index>
                  <key>290009</key>
                  <metrics>
                     <entry>
                        <key>rxSpeed</key>
                        <value>3790.8790113658024</value>
                     </entry>
                     <entry>
                        <key>utilBytes</key>
                        <value>8.40631367E8</value>
                     </entry>
                     <entry>
                        <key>inBytes</key>
                        <value>7.97480016E8</value>
                     </entry>
                     <entry>
                        <key>txSpeed</key>
                        <value>210.51087003645023</value>
                     </entry>
                     <entry>
                        <key>outBytes</key>
                        <value>4.3151351E7</value>
                     </entry>
                     <entry>
                        <key>txDrops</key>
                        <value>0.0</value>
                     </entry>
                  </metrics>
                  <name>GigabitEthernet0</name>
                  <properties>
                     <entry>
                        <key>physAddress</key>
                        <value>c025.5c08.e3f5</value>
                     </entry>
                  </properties>
               </updatedInterfaces>
                        .
                        .
                        .

               <updatedMetrics>
                  <entry>
                     <key>cellularBandwidth</key>
                     <value>0.0</value>
                  </entry>
                  <entry>
                     <key>cellularTxSpeed</key>
                     <value>0.0</value>
                  </entry>
                  <entry>
```

```
            <key>ethernetTxSpeed</key>
            <value>516.6779073252414</value>
         </entry>
         <entry>
            <key>cellularBwPerCycle</key>
            <value>0.0</value>
         </entry>
         <entry>
            <key>ethernetTxDrops</key>
            <value>0.0</value>
         </entry>
         <entry>
            <key>uptime</key>
            <value>1650960.0</value>
         </entry>
         <entry>
            <key>cellularRxSpeed</key>
            <value>0.0</value>
         </entry>
         <entry>
            <key>cellularEcio</key>
            <value>-11.0</value>
         </entry>
         <entry>
            <key>ethernetRxSpeed</key>
            <value>3790.8790113658024</value>
         </entry>
         <entry>
            <key>cellConnectTime</key>
            <value>93.0</value>
         </entry>
         <entry>
            <key>cellularRssi</key>
            <value>-117.0</value>
         </entry>
      </updatedMetrics>
      <updatedProperties>
         <entry>
            <key>runningFirmwareImageId</key>
            <value>null</value>
         </entry>
         <entry>
            <key>lastUpdate</key>
            <value>2015-03-25 23:29:19.0</value>
         </entry>
         <entry>
            <key>lng</key>
            <value>-26.9007</value>
         </entry>
         <entry>
            <key>reloadFirmwareVersion</key>
            <value>null</value>
         </entry>
         <entry>
            <key>reloadDate</key>
            <value>null</value>
         </entry>
         <entry>
            <key>backupFirmwareVersion</key>
            <value>null</value>
         </entry>
         <entry>
            <key>slot4FirmwareImageId</key>
```

```
      <value>null</value>
   </entry>
   <entry>
      <key>deviceType</key>
      <value>c800</value>
   </entry>
   <entry>
      <key>alt</key>
      <value>21</value>
   </entry>
   <entry>
      <key>name</key>
      <value>C819HGW-S-A-K9+FTX174685VB</value>
   </entry>
   <entry>
      <key>configInSync</key>
      <value>false</value>
   </entry>
   <entry>
      <key>runningFirmwareVersion</key>
      <value>15.5(0.23)T</value>
   </entry>
   <entry>
      <key>hardwareId</key>
      <value>null</value>
   </entry>
   <entry>
      <key>vid</key>
      <value>null</value>
   </entry>
   <entry>
      <key>lat</key>
      <value>26.0197</value>
   </entry>
   <entry>
      <key>slot5FirmwareImageId</key>
      <value>null</value>
   </entry>
   <entry>
      <key>sn</key>
      <value>FTX174685VB</value>
   </entry>
   <entry>
      <key>backupFirmwareImageId</key>
      <value>null</value>
   </entry>
   <entry>
      <key>status</key>
      <value>up</value>
   </entry>
   <entry>
      <key>hostname</key>
      <value>kit-819</value>
   </entry>
   <entry>
      <key>pid</key>
      <value>C819HGW-S-A-K9</value>
   </entry>
   <entry>
      <key>lastHeard</key>
      <value>2015-03-25 23:34:18.0</value>
   </entry>
   <entry>
      <key>mapLevel</key>
      <value>1</value>
```

```
                    </entry>
                    <entry>
                       <key>ip</key>
                       <value>172.27.161.82</value>
                    </entry>
                    <entry>
                       <key>downloadFirmwareVersion</key>
                       <value>null</value>
                    </entry>
                    <entry>
                       <key>slot6FirmwareImageId</key>
                       <value>null</value>
                    </entry>
                    <entry>
                       <key>geoHash</key>
                       <value>eksu5bex8r8hrfk9942g4</value>
                    </entry>
                    <entry>
                       <key>downloadFirmwareImageId</key>
                       <value>null</value>
                    </entry>
                 </updatedProperties>
              </updatedDeviceDetails>
           </updatedDeviceDetailQueryResult>
        </ns2:getUpdatedDeviceDetailsResponse>
     </soap:Body>
</soap:Envelope>
```

Example SOAP Request using event

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"xmlns:even="http://event.nbapi.cgms.cisco.com/
">
 <soapenv:Header/>
 <soapenv:Body>
  <even:searchEvents>
   <!--Optional:-->
   <query>?</query>
   <!--Optional:-->
   <count>?</count>
   <!--Optional:-->
   <offset>?</offset>
  </even:searchEvents>
 </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Request using issues

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"xmlns:iss="http://issues.nbapi.cgms.cisco.com/
">
 <soapenv:Header/>
 <soapenv:Body>
  <iss:searchIssues>
   <!--Optional:-->?
   <query>deviceType:cgr1000 issueStatus:open</query>
   <!--Optional:-->
   <count>2</count>
   <!--Optional:-->
   <offset>0</offset>
  </iss:searchIssues>
 </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Request using meshDeviceOps

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:mes="http://meshDeviceOps.nbapi.cgms.cisco.com/">
 <soapenv:Header/>
 <soapenv:Body>
  <mes:getFirmwareImageInfoList>
   <!--Optional:-->
   <firmwareGroup>?</firmwareGroup>
  </mes:getFirmwareImageInfoList>
 </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:mes="http://meshDeviceOps.nbapi.cgms.cisco.com/">
 <soapenv:Header/>
 <soapenv:Body>
  <mes:getFirmwareUploadStatus>
   <!--Optional:-->
   <firmwareGroup>?</firmwareGroup>
  </mes:getFirmwareUploadStatus>
 </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:mes="http://meshDeviceOps.nbapi.cgms.cisco.com/">
 <soapenv:Header/>
 <soapenv:Body>
  <mes:scheduleReload>
   <!--Optional:-->
   <firmwareGroup>?</firmwareGroup>
   <!--Optional:-->
   <firmwareImageName>?</firmwareImageName>
   <!--Optional:-->
   <reloadGmtTime>?</reloadGmtTime>
  </mes:scheduleReload>?
 </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:mes="http://meshDeviceOps.nbapi.cgms.cisco.com/">
 <soapenv:Header/>
 <soapenv:Body>
  <mes:setBackupFirmwareImage>
   <!--Optional:-->
   <firmwareGroup>?</firmwareGroup>
   <!--Optional:-->
   <firmwareImageName>?</firmwareImageName>
  </mes:setBackupFirmwareImage>
 </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:mes="http://meshDeviceOps.nbapi.cgms.cisco.com/">
 <soapenv:Header/>
 <soapenv:Body>
  <mes:startUpload>
   <!--Optional:-->
   <firmwareGroup>?</firmwareGroup>
   <!--Optional:-->
   <firmwareImageName>?</firmwareImageName>
  </mes:startUpload>
 </soapenv:Body>
</soapenv:Envelope>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:mes="http://meshDeviceOps.nbapi.cgms.cisco.com/">
 <soapenv:Header/>
 <soapenv:Body>
  <mes:stopUpload>
   <!--Optional:-->
   <firmwareGroup>?</firmwareGroup>
  </mes:stopUpload>
 </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Request using queryId

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sear="http://search.nbapi.cgms.cisco.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <sear:getUpdatedDeviceDetails>
         <queryId>E1EA07CE9BE5536A883469707FCA602E</queryId>
      </sear:getUpdatedDeviceDetails>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Request using workorder

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wor="http://workorder.nbapi.cgms.cisco.com/">
 <soapenv:Header/>
 <soapenv:Body>
  <wor:RequestSignedAuthorization>
   <!--Optional:-->
   <technicianUserName>?</technicianUserName>
   <!--Optional:-->
   <workOrderNumber>?</workOrderNumber>
   <!--Optional:-->
   <appVersion>?</appVersion>
  </wor:RequestSignedAuthorization>
 </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wor="http://workorder.nbapi.cgms.cisco.com/">
 <soapenv:Header/>
 <soapenv:Body>
  <wor:RequestSignedAuthorizationWithClientKey>
   <!--Optional:-->
   <technicianUserName>?</technicianUserName>
   <!--Optional:-->
   <workOrderNumber>?</workOrderNumber>
   <!--Optional:-->?
   <appVersion>?</appVersion>
   <!--Optional:-->
   <appKey>?</appKey>
  </wor:RequestSignedAuthorizationWithClientKey>
 </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wor="http://workorder.nbapi.cgms.cisco.com/">
 <soapenv:Header/>
 <soapenv:Body>
  <wor:RequestUserAuthentication>
```

```
    <!--Optional:-->
    <userAuthInfo>
     <!--Optional:-->
     <appVersion>?</appVersion>
     <!--Optional:-->
     <scriptVersion>?</scriptVersion>
    </userAuthInfo>
   </wor:RequestUserAuthentication>
  </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wor="http://workorder.nbapi.cgms.cisco.com/">
 <soapenv:Header/>
 <soapenv:Body>
  <wor:UploadServiceReport>
   <!--Zero or more repetitions:-->
   <serviceStatusReport>
    <!--Optional:-->
    <orderNumber>?</orderNumber>
    <!--Optional:-->
    <deviceId>?</deviceId>
    <!--Optional:-->
    <technicianUserName>?</technicianUserName>
    <!--Optional:-->
    <status>?</status>
   </serviceStatusReport>
  </wor:UploadServiceReport>
 </soapenv:Body>
</soapenv:Envelope>
```

Example Python Request

IoT FND 3.0 and NB API 3.0 work with both TLS1.0 and TLS1.2.

Note: If you use TLS1.2 with SUDS package, you must use the ActivePython package available at:

http://www.activestate.com/blog/2015/11/activepython-vs-open-source-python-whats-difference

Note: If you use TLS1.0 with SUDS package, you must use the regular Python package available at:

https://www.python.org/downloads/

```
#!/usr/local/bin/python
from suds.transport.https import HttpAuthenticated
from suds.client import Client
import logging
from datetime import timedelta,date,datetime,tzinfo
import requests
from requests.auth import HTTPBasicAuth
import datetime

transport = HttpAuthenticated(username='root',password='PeterChen123!')
WSDL_URL= "https://172.27.126.110/nbapi/search/?wsdl"
client = Client(WSDL_URL, faults=False,cachingpolicy=1,location=WSDL_URL,transport=transport)

print(client)
#query="deviceType:cgr1000"
query="deviceType:cgmesh"
#fieldNames = ["uptime","uplinkTxSpeed","uplinkTxDrops"]
string_start_date = "2014-09-28 20:30:55.78200"
string_end_date = "2015-03-25 20:30:55.78200"
startTime = datetime.datetime.strptime(string_start_date, "%Y-%m-%d %H:%M:%S.%f")
endTime= datetime.datetime.strptime(string_end_date, "%Y-%m-%d %H:%M:%S.%f")
```

```
#deviceId=["00173bab003c3500", "00173bab003c3501"]
count=5
offset =0
searchthis = client.service.getUpdatedDeviceDetails(query,startTime,endTime, "",count, offset)
print(searchthis)
```

Results

```
Suds ( https://fedorahosted.org/suds/ )  version: 0.6

Service ( SearchWebService ) tns="http://search.nbapi.cgms.cisco.com/"
   Prefixes (1)
      ns0 = "http://search.nbapi.cgms.cisco.com/"
   Ports (1):
      (SearchWebServicePort)
         Methods (8):
            findEidsForIpAddresses(xs:string[] ipAddresses)
            findEidsForIpAddressesByDeviceType(xs:string deviceType, xs:string[] ipAddresses)
            getDeletedDevices(xs:string deviceType, xs:dateTime startTime, xs:dateTime endTime)
            getDeviceDetails(xs:string query, xs:string[] deviceIds, xs:string queryId, xs:long count,
xs:long offset)
            getGroups(xs:string groupType)
            getMetricHistory(xs:string query, xs:string[] deviceIds, xs:dateTime startTime, xs:dateTime
endTime, xs:string[] metricIds, xs:string rollupInterval, xs:string rollupFunction, xs:string queryId,
xs:long count, xs:long offset)
            getUpdatedDeviceDetails(xs:string query, xs:dateTime startTime, xs:dateTime endTime,
xs:string queryId, xs:long count, xs:long offset)
            searchDevices(xs:string query, xs:string[] fieldNames, xs:string queryId, xs:long count,
xs:long offset)
         Types (36):
            Exception
            address
            asset
            deletedDevice
            deletedDeviceQueryResult
            device
            deviceDetail
            deviceDetailQueryResult
            deviceQueryResult
            eidsForIpAddressesResult
            findEidsForIpAddresses
            findEidsForIpAddressesByDeviceType
            findEidsForIpAddressesByDeviceTypeResponse
            findEidsForIpAddressesResponse
            getDeletedDevices
            getDeletedDevicesResponse
            getDeviceDetails
            getDeviceDetailsResponse
            getGroups
            getGroupsResponse
            getMetricHistory
            getMetricHistoryResponse
            getUpdatedDeviceDetails
            getUpdatedDeviceDetailsResponse
            group
            groupQueryResult
            interface
            metricHistoryQueryResult
            metricValue
            queryResult
            queryStatus
            route
            searchDevices
```

```
            searchDevicesResponse
            updatedDeviceDetail
            updatedDeviceDetailQueryResult


(200, (updatedDeviceDetailQueryResult){
   queryId = "60EF7F6ACFD8EE823AC2B8F83E545BC9"
   queryStatus = "SUCCEEDED"
   updatedDeviceDetails[] =
      (updatedDeviceDetail){
         configGroup = "default-cgmesh"
         eid = "00000000001"
         firmwareGroup = "default-cgmesh"
         key = 261628
         updatedMetrics = ""
         updatedProperties =
            (updatedProperties){
               entry[] =
                  (entry){
                     key = "runningFirmwareImageId"
                     value = "null"
                  },
                  (entry){
                     key = "lastUpdate"
                     value = "2015-03-12 21:28:48.0"
                  },
                  (entry){
                     key = "lng"
                     value = "1000.0"
                  },
                  (entry){
                     key = "reloadFirmwareVersion"
                     value = "null"
                  },
                  (entry){
                     key = "reloadDate"
                     value = "null"
                  },
                  (entry){
                     key = "function"
                     value = "meter"
                  },
                  (entry){
                     key = "backupFirmwareVersion"
                     value = "null"
                  },
                  (entry){
                     key = "slot4FirmwareImageId"
                     value = "null"
                  },
                  (entry){
                     key = "deviceType"
                     value = "cgmesh"
                  },
                  (entry){
                     key = "alt"
                     value = "null"
                  },
                  (entry){
                     key = "name"
                     value = "00000000001"
                  },
                  (entry){
                     key = "configInSync"
                     value = "false"
```

```
},
(entry){
   key = "runningFirmwareVersion"
   value = "null"
},
(entry){
   key = "hardwareId"
   value = "null"
},
(entry){
   key = "vid"
   value = "null"
},
(entry){
   key = "lat"
   value = "1000.0"
},
(entry){
   key = "slot5FirmwareImageId"
   value = "null"
},
(entry){
   key = "sn"
   value = "null"
},
(entry){
   key = "backupFirmwareImageId"
   value = "null"
},
(entry){
   key = "status"
   value = "unheard"
},
(entry){
   key = "hostname"
   value = "null"
},
(entry){
   key = "pid"
   value = "null"
},
(entry){
   key = "lastHeard"
   value = "null"
},
(entry){
   key = "mapLevel"
   value = "16"
},
(entry){
   key = "ip"
   value = "2.2.2.11"
},
(entry){
   key = "downloadFirmwareVersion"
   value = "null"
},
(entry){
   key = "slot6FirmwareImageId"
   value = "null"
},
(entry){
  key = "geoHash"
```

```
                value = "null"
             },
             (entry){
                key = "downloadFirmwareImageId"
                value = "null"
             },
       }
},
(updatedDeviceDetail){
   configGroup = "default-cgmesh"
   eid = "00000000002"
   firmwareGroup = "default-cgmesh"
   key = 261627
   updatedMetrics = ""
   updatedProperties =
      (updatedProperties){
         entry[] =
            (entry){
               key = "runningFirmwareImageId"
               value = "null"
            },
            (entry){
               key = "lastUpdate"
               value = "2015-03-12 21:28:48.0"
            },
            (entry){
               key = "lng"
               value = "1000.0"
            },
            (entry){
               key = "reloadFirmwareVersion"
               value = "null"
            },
            (entry){
               key = "reloadDate"
               value = "null"
            },
            (entry){
               key = "function"
               value = "meter"
            },
            (entry){
               key = "backupFirmwareVersion"
               value = "null"
            },
            (entry){
               key = "slot4FirmwareImageId"
               value = "null"
            },
            (entry){
               key = "deviceType"
               value = "cgmesh"
            },
            (entry){
               key = "alt"
               value = "null"
            },
            (entry){
               key = "name"
               value = "00000000002"
            },
            (entry){
               key = "configInSync"
               value = "false"
            },
```

```
(entry){
   key = "runningFirmwareVersion"
   value = "null"
},
(entry){
   key = "hardwareId"
   value = "null"
},
(entry){
   key = "vid"
   value = "null"
},
(entry){
   key = "lat"
   value = "1000.0"
},
(entry){
   key = "slot5FirmwareImageId"
   value = "null"
},
(entry){
   key = "sn"
   value = "null"
},
(entry){
   key = "backupFirmwareImageId"
   value = "null"
},
(entry){
   key = "status"
   value = "unheard"
},
(entry){
   key = "hostname"
   value = "null"
},
(entry){
   key = "pid"
   value = "null"
},
(entry){
   key = "lastHeard"
   value = "null"
},
(entry){
   key = "mapLevel"
   value = "16"
},
(entry){
   key = "ip"
   value = "2.2.2.12"
},
(entry){
   key = "downloadFirmwareVersion"
   value = "null"
},
(entry){
   key = "slot6FirmwareImageId"
   value = "null"
},
(entry){
   key = "geoHash"
   value = "null"
```

```
            },
            (entry){
               key = "downloadFirmwareImageId"
               value = "null"
            },
         }
      },
      (updatedDeviceDetail){
         configGroup = "default-cgmesh"
         eid = "0007810800a80bfe"
         firmwareGroup = "default-cgmesh"
         key = 20127
         updatedAddresses[] =
            (address){
               address = "fe80:0:0:0:0:0:0:1"
               addressType = "IPV6"
               key = 20001004
               prefixLength = 64
            },
            (address){
               address = "2010:dead:beef:cafe:0:8108:a8:bfe"
               addressType = "IPV6"
               key = 20001005
               prefixLength = 64
            },
            (address){
               address = "fe80:0:0:0:207:8108:a8:bfe"
               addressType = "IPV6"
               key = 20001006
               prefixLength = 64
            },
            (address){
               address = "0:0:0:0:0:0:0:1"
               addressType = "IPV6"
               key = 20001007
               prefixLength = 128
            },
         updatedInterfaces[] =
            (interface){
               addresses[] =
                  (address){
                     address = "0:0:0:0:0:0:0:1"
                     addressType = "IPV6"
                     key = 20001007
                     prefixLength = 128
                  },
               index = 1
               key = 50004
               metrics = ""
               name = "lo"
               properties = ""
            },
            (interface){
               addresses[] =
                  (address){
                     address = "2010:dead:beef:cafe:0:8108:a8:bfe"
                     addressType = "IPV6"
                     key = 20001005
                     prefixLength = 64
                  },
                  (address){
                     address = "fe80:0:0:0:207:8108:a8:bfe"
                     addressType = "IPV6"
                     key = 20001006
                     prefixLength = 64
```

```
            },
         index = 2
         key = 50005
         metrics = ""
         name = "lowpan"
         properties =
            (properties){
               entry[] =
                  (entry){
                     key = "physAddress"
                     value = "0007810800a80bfe"
                  },
            }
      },
      (interface){
         addresses[] =
            (address){
               address = "fe80:0:0:0:0:0:0:1"
               addressType = "IPV6"
               key = 20001004
               prefixLength = 64
            },
         index = 3
         key = 50006
         metrics = ""
         name = "ppp"
         properties =
            (properties){
               entry[] =
                  (entry){
                    key = "physAddress"
                     value = "0007810800a80bfe"
                  },
            }
      },
   updatedMetrics = ""
   updatedProperties =
      (updatedProperties){
         entry[] =
            (entry){
               key = "runningFirmwareImageId"
               value = "null"
            },
            (entry){
               key = "lastUpdate"
               value = "2015-03-12 21:28:48.0"
            },
            (entry){
               key = "meshPanid"
               value = "26930"
            },
            (entry){
               key = "lng"
               value = "-88.66548868"
            },
            (entry){
               key = "reloadFirmwareVersion"
               value = "null"
            },
            (entry){
               key = "activeLinkType"
               value = "RF"
            },
```

```
(entry){
   key = "reloadDate"
   value = "1970-01-01 00:00:00.0"
},
(entry){
   key = "backupFirmwareVersion"
   value = "null"
},
(entry){
   key = "slot4FirmwareImageId"
   value = "null"
},
(entry){
   key = "deviceType"
   value = "cgmesh"
},
(entry){
   key = "alt"
   value = "null"
},
(entry){
   key = "name"
   value = "0007810800a80bfe"
},
(entry){
   key = "configInSync"
   value = "true"
},
(entry){
   key = "runningFirmwareVersion"
   value = "5.5.62"
},
(entry){
   key = "hardwareId"
   value = "null"
},
(entry){
   key = "vid"
   value = "3.1"
},
(entry){
   key = "lat"
   value = "43.56901132"
},
(entry){
   key = "slot5FirmwareImageId"
   value = "null"
},
(entry){
   key = "sn"
   value = "0007810800A80BFE"
},
(entry){
   key = "backupFirmwareImageId"
   value = "null"
},
(entry){
   key = "status"
   value = "outage"
},
(entry){
   key = "hostname"
   value = "null"
},
(entry){
```

```
                key = "pid"
                value = "OWCM"
             },
             (entry){
                key = "lastHeard"
                value = "2015-02-24 18:59:52.0"
             },
             (entry){
                key = "mapLevel"
                value = "16"
             },
             (entry){
                key = "ip"
                value = "2010:dead:beef:cafe:0:8108:a8:bfe"
             },
             (entry){
                key = "downloadFirmwareVersion"
                value = "null"
             },
            (entry){
                key = "slot6FirmwareImageId"
                value = "null"
             },
             (entry){
                key = "geoHash"
                value = "dp8zy77zbk7u02v6jxzgn"
             },
             (entry){
                key = "downloadFirmwareImageId"
                value = "null"
             },
             (entry){
                key = "previousMeshPanid"
                value = "2015"
             },
      }
   updatedRoutes[] =
      (route){
         index = 1
         key = 100005
         metrics = ""
         properties =
            (properties){
               entry[] =
                  (entry){
                     key = "destAddressType"
                     value = "2"
                  },
                  (entry){
                     key = "nextHopAddressType"
                     value = "4"
                  },
                  (entry){
                     key = "nextHopAddress"
                     value = "fe80:0:0:0:207:8108:bf:a053"
                  },
                  (entry){
                     key = "destAddress"
                     value = "0:0:0:0:0:0:0:0"
                  },
                  (entry){
                     key = "prefixLength"
                     value = "0"
```

```
                    },
                }
            },
        },
    (updatedDeviceDetail){
       configGroup = "default-cgmesh"
       eid = "0007810800a80d40"
       firmwareGroup = "default-cgmesh"
       key = 20126
       updatedAddresses[] =
          (address){
             address = "2010:dead:beef:cafe:0:8108:a8:d40"
             addressType = "IPV6"
             key = 20001009
             prefixLength = 64
          },
          (address){
             address = "fe80:0:0:0:0:0:0:1"
             addressType = "IPV6"
             key = 20001010
             prefixLength = 64
          },
          (address){
             address = "0:0:0:0:0:0:0:1"
             addressType = "IPV6"
             key = 20001011
             prefixLength = 128
          },
          (address){
             address = "fe80:0:0:0:207:8108:a8:d40"
             addressType = "IPV6"
             key = 20001008
             prefixLength = 64
          },
       updatedInterfaces[] =
          (interface){
             addresses[] =
                (address){
                   address = "0:0:0:0:0:0:0:1"
                   addressType = "IPV6"
                  key = 20001011
                   prefixLength = 128
                },
             index = 1
             key = 50007
             metrics = ""
             name = "lo"
             properties = ""
          },
          (interface){
             addresses[] =
                (address){
                   address = "2010:dead:beef:cafe:0:8108:a8:d40"
                   addressType = "IPV6"
                   key = 20001009
                   prefixLength = 64
                },
                (address){
                   address = "fe80:0:0:0:207:8108:a8:d40"
                   addressType = "IPV6"
                   key = 20001008
                   prefixLength = 64
                },
             index = 2
             key = 50008
```

```
            metrics = ""
            name = "lowpan"
            properties =
               (properties){
                  entry[] =
                     (entry){
                        key = "physAddress"
                        value = "0007810800a80d40"
                     },
               }
         },
         (interface){
            addresses[] =
               (address){
                  address = "fe80:0:0:0:0:0:0:1"
                  addressType = "IPV6"
                  key = 20001010
                  prefixLength = 64
               },
            index = 3
            key = 50009
            metrics = ""
            name = "ppp"
            properties =
               (properties){
                  entry[] =
                     (entry){
                        key = "physAddress"
                        value = "0007810800a80d40"
                     },
               }
         },
      updatedMetrics = ""
      updatedProperties =
         (updatedProperties){
            entry[] =
               (entry){
                  key = "runningFirmwareImageId"
                  value = "null"
               },
               (entry){
                  key = "lastUpdate"
                  value = "2015-03-12 21:28:48.0"
               },
               (entry){
                  key = "meshPanid"
                  value = "26930"
               },
               (entry){
                  key = "lng"
                  value = "-88.96548868"
               },
               (entry){
                  key = "reloadFirmwareVersion"
                  value = "null"
               },
               (entry){
                  key = "activeLinkType"
                  value = "RF"
               },
               (entry){
                  key = "reloadDate"
                  value = "1970-01-01 00:00:00.0"
```

```
},
(entry){
   key = "backupFirmwareVersion"
   value = "null"
},
(entry){
   key = "slot4FirmwareImageId"
   value = "null"
},
(entry){
   key = "deviceType"
   value = "cgmesh"
},
(entry){
   key = "alt"
   value = "null"
},
(entry){
   key = "name"
   value = "0007810800a80d40"
},
(entry){
   key = "configInSync"
   value = "true"
},
(entry){
   key = "runningFirmwareVersion"
   value = "5.5.42"
},
(entry){
   key = "hardwareId"
   value = "null"
},
(entry){
   key = "vid"
   value = "3.1"
},
(entry){
   key = "lat"
   value = "43.46901132"
},
(entry){
   key = "slot5FirmwareImageId"
   value = "null"
},
(entry){
   key = "sn"
   value = "0007810800A80D40"
},
(entry){
   key = "backupFirmwareImageId"
   value = "null"
},
(entry){
   key = "status"
   value = "outage"
},
(entry){
   key = "hostname"
   value = "null"
},
(entry){
   key = "pid"
   value = "OWCM"
},
```

```
            (entry){
               key = "lastHeard"
               value = "2015-02-23 23:27:00.0"
            },
            (entry){
               key = "mapLevel"
               value = "16"
            },
            (entry){
               key = "ip"
               value = "2010:dead:beef:cafe:0:8108:a8:d40"
            },
            (entry){
               key = "downloadFirmwareVersion"
               value = "5.5.59"
            },
            (entry){
               key = "slot6FirmwareImageId"
               value = "null"
            },
            (entry){
               key = "geoHash"
               value = "dp8xr9333g58c96cp1ptr"
            },
            (entry){
               key = "downloadFirmwareImageId"
               value = "null"
            },
            (entry){
               key = "previousMeshPanid"
               value = "2015"
            },
      }
   updatedRoutes[] =
      (route){
         index = 1
         key = 100006
         metrics = ""
         properties =
            (properties){
               entry[] =
                  (entry){
                     key = "destAddressType"
                     value = "2"
                  },
                  (entry){
                     key = "nextHopAddressType"
                     value = "4"
                  },
                  (entry){
                     key = "nextHopAddress"
                     value = "fe80:0:0:0:207:8108:cc:e50b"
                  },
                  (entry){
                     key = "destAddress"
                     value = "0:0:0:0:0:0:0:0"
                  },
                  (entry){
                     key = "prefixLength"
                     value = "0"
                  },
            }
      },
```

```
},
(updatedDeviceDetail){
   configGroup = "default-cgmesh"
   eid = "0007810800bf38e7"
   firmwareGroup = "default-cgmesh"
   key = 20130
   updatedAddresses[] =
      (address){
         address = "fe80:0:0:0:207:8108:bf:38e7"
         addressType = "IPV6"
         key = 104001023
         prefixLength = 64
      },
      (address){
         address = "fe80:0:0:0:0:0:0:1"
         addressType = "IPV6"
         key = 104001024
         prefixLength = 64
      },
      (address){
         address = "2011:dead:beef:cafe:0:8108:bf:38e7"
         addressType = "IPV6"
         key = 104001025
         prefixLength = 64
      },
      (address){
         address = "0:0:0:0:0:0:0:1"
         addressType = "IPV6"
         key = 104001026
         prefixLength = 128
      },
   updatedInterfaces[] =
      (interface){
         addresses[] =
            (address){
               address = "0:0:0:0:0:0:0:1"
               addressType = "IPV6"
               key = 104001026
               prefixLength = 128
            },
         index = 1
         key = 250016
         metrics = ""
         name = "lo"
         properties = ""
      },
      (interface){
         addresses[] =
            (address){
               address = "fe80:0:0:0:207:8108:bf:38e7"
               addressType = "IPV6"
               key = 104001023
               prefixLength = 64
            },
            (address){
               address = "2011:dead:beef:cafe:0:8108:bf:38e7"
               addressType = "IPV6"
               key = 104001025
               prefixLength = 64
            },
         index = 2
         key = 250017
         metrics = ""
         name = "lowpan"
         properties =
```

```
            (properties){
               entry[] =
                  (entry){
                     key = "physAddress"
                     value = "0007810800bf38e7"
                  },
            }
      },
      (interface){
         addresses[] =
            (address){
               address = "fe80:0:0:0:0:0:0:1"
               addressType = "IPV6"
               key = 104001024
               prefixLength = 64
            },
         index = 3
         key = 250018
         metrics = ""
         name = "ppp"
         properties =
            (properties){
               entry[] =
                  (entry){
                     key = "physAddress"
                     value = "0007810800bf38e7"
                  },
            }
      },
   updatedMetrics =
      (updatedMetrics){
         entry[] =
            (entry){
               key = "nodeLocalTime"
               value = 1427113644.0
            },
      }
   updatedProperties =
      (updatedProperties){
         entry[] =
            (entry){
               key = "runningFirmwareImageId"
               value = "null"
            },
            (entry){
               key = "lastUpdate"
               value = "2015-03-23 12:27:24.0"
            },
            (entry){
               key = "meshPanid"
               value = "118"
            },
            (entry){
               key = "lng"
               value = "-87.71548868"
            },
            (entry){
               key = "reloadFirmwareVersion"
               value = "null"
            },
            (entry){
               key = "reloadDate"
               value = "null"
```

```
},
(entry){
   key = "backupFirmwareVersion"
   value = "null"
},
(entry){
   key = "slot4FirmwareImageId"
   value = "null"
},
(entry){
   key = "deviceType"
   value = "cgmesh"
},
(entry){
   key = "alt"
   value = "null"
},
(entry){
   key = "name"
   value = "0007810800bf38e7"
},
(entry){
   key = "configInSync"
   value = "false"
},
(entry){
   key = "runningFirmwareVersion"
   value = "5.5.42"
},
(entry){
   key = "hardwareId"
   value = "null"
},
(entry){
   key = "vid"
   value = "3.1"
},
(entry){
   key = "lat"
   value = "42.16901532"
},
(entry){
   key = "slot5FirmwareImageId"
   value = "null"
},
(entry){
   key = "sn"
   value = "0007810800BF38E7"
},
(entry){
   key = "backupFirmwareImageId"
   value = "null"
},
(entry){
   key = "status"
   value = "down"
},
(entry){
   key = "hostname"
   value = "null"
},
(entry){
   key = "pid"
   value = "OWCM"
},
```

```
                    (entry){
                       key = "lastHeard"
                       value = "2015-03-23 12:26:00.0"
                    },
                    (entry){
                       key = "mapLevel"
                       value = "16"
                    },
                    (entry){
                       key = "ip"
                       value = "2011:dead:beef:cafe:0:8108:bf:38e7"
                    },
                    (entry){
                       key = "downloadFirmwareVersion"
                       value = "null"
                    },
                    (entry){
                       key = "meshSsid"
                       value = "sjklin-internal"
                    },
                    (entry){
                       key = "slot6FirmwareImageId"
                       value = "null"
                    },
                    (entry){
                       key = "meterCert"
                       value = "host/smartmeter"
                    },
                    (entry){
                       key = "geoHash"
                       value = "dp3xguxs2fpne6gfgtjyw"
                    },
                    (entry){
                       key = "downloadFirmwareImageId"
                       value = "null"
                    },
                    (entry){
                       key = "meshTxPower"
                       value = "-34"
                    },
                    (entry){
                       key = "previousMeshPanid"
                       value = "null"
                    },
                    (entry){
                       key = "meshSecMode"
                       value = "1"
                    },
                }
        },
})
```

getMetricHistory

This call lets the client retrieve the historical metric values saved in the IoT FND database by specifying the following:

- 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

Prototype

```
MetricHistoryQueryResult getMetricHistory(string query, list<string> deviceIds, date startTime,
                                          date endTime, list<string> metricIds, string rollupInterval,
                                          string rollupFunction, string queryId, long count,
                                          long offset)
```

Parameters

Table 16 getMetricHistory Parameters

Parameter	Type	Description
query	string	Search query string
deviceIds	list<string>	List of device EIDs to retrieve.
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 to retrieve for each device.
rollupInterval	string	Valid values are *none*, *hour*, or *day*.
rollupFunction	string	Valid values are *avg*, *max*, or *min*.
queryId	string	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	long	Number of results to retrieve. Valid range is 1–40000
offset	long	Position of the first result. Valid values are >= 0.

Results

This interface returns a list of metric values, as defined in Table 17.

Table 17 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.

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>
```

findEidsForIpAddresses

This call retrieves the EID of devices based on the IP address, and allows the Collection Engine to communicate to IoT FND which devices to migrate.

Prototype

```
EidsForIpAddressesResult findEidsForIpAddresses(list<string> ipAddresses)
```

Parameters

Table 18 findEidsForIpAddresses Parameters

Parameter	Type	Description
ipAddresses	list<string>	The list of IP addresses with EIDs corresponding to devices to search for. **Note:** The list is limited to a maximum of 1000 IP addresses. An error returns if more than 1000 are specified. IoT FND does not support inputs as GRE Tunnel IPv6 addresses.

Results

This call always returns a response of type EidsForIpAddressesResult. The queryStatus value defines any errors.

On success, the eidMap contains correlations between the IP address and the EID of the device with that IP address. If an IP address is not mapped to a device, those IP addresses are included in the invalidMappings value.

Table 19 describes the parameters in the response.

Table 19 findEidsForIpAddresses Results

Parameter	Type	Description
eidMap	map<string,string>	Map of IP addresses-to-EIDs, where the address maps to a single CGR.
invalidMappings	list<string>	List of IP addresses that did not correspond to any CGR.

findEidsForIpAddressesByDeviceType

This call retrieves the EID of devices based on the device type, and allows the Collection Engine to communicate to IoT FND which devices to migrate.

Prototype

```
EidsForIpAddressesResult findEidsForIpAddressesByDeviceType(string deviceType,
                                                            list<string> ipAddresses)
```

Parameters

Table 20 findEidsForIpAddressesByDeviceType Parameters

Parameter	Type	Description
deviceType	string	The name of the device type with EIDs corresponding to devices to search for. For example, cgr1000 only matches IP addresses against CGR devices.
ipAddresses	list<string>	Returns a list of uplink IP addresses the collection engine uses to track CGR devices. WAN or IPSEC Tunnel. **Note:** The list is limited to a maximum of 1000 IP addresses. An error returns if more than 1000 are specified. IoT FND does not support inputs as GRE Tunnel IPv6 addresses.

Results

This call always returns a response of type EidsForIpAddressesResult. The queryStatus value defines any errors.

On success, the eidMap contains correlations between the IP address and the EID of the device with that IP address. If an IP address is not mapped to a device of the specified deviceType, those IP addresses are included in the invalidMappings value.

Table 21 describes the parameters in the response.

Table 21 findEidsForIpAddressesByDeviceType Results

Parameter	Type	Description
eidMap	map<string,string>	Map of IP addresses-to-EIDs, where the address maps to a single CGR.
invalidMappings	list<string>	List of IP addresses that did not correspond to any CGR.