Testing and Configuration Guide for Cisco Platform Exchange Grid (pxGrid) 2.0

Author: John Eppich

Table of Contents

About this Document

This document is for Cisco Ecosystem Partners, Cisco Security Technical Alliances (CSTA) Partners and Cisco Internal Solutions implementing Cisco Platform Exchange Grid (pxGrid 2.0). This document accompanies the Cisco Devnet WebSockets & REST API Cisco pxGrid (2.0) development document.
https://developer.cisco.com/docs/pxgrid-api/

Cisco pxGrid 2.0 does not rely on "java" and "c" SDK's for development as was the case with Cisco pxGrid 1.0. Instead Cisco pxGrid 2.0 relies on WebSockets & REST API over the STOMP messaging protocol, which alleviates the dependency on SDK's for development. The developer may now use python for pxGrid application development.

This document explains how WebSockets & REST APIs are used in Cisco pxGrid 2.0, and STOMP is the underlying messaging protocol. Code examples are provided to help in code development.

Cisco pxGrid Context-In is a new feature in ISE 2.4, where Internet of Technology (IOT) solutions can provide their asset data for Cisco Identity Services Engine (ISE) to consume and incorporate with the ISE Profiling Policies. An IOT organization' security policy can then be written by assigning these ISE Profiling Policies to ISE Authorization Profiles.

An API_Simulator.JAR file is included to learn the details of Cisco pxGrid Context-In. This simulates a pxGrid client publishing asset information and ISE taking this context in via pxGrid. Coding examples for this JAR file are also provided.

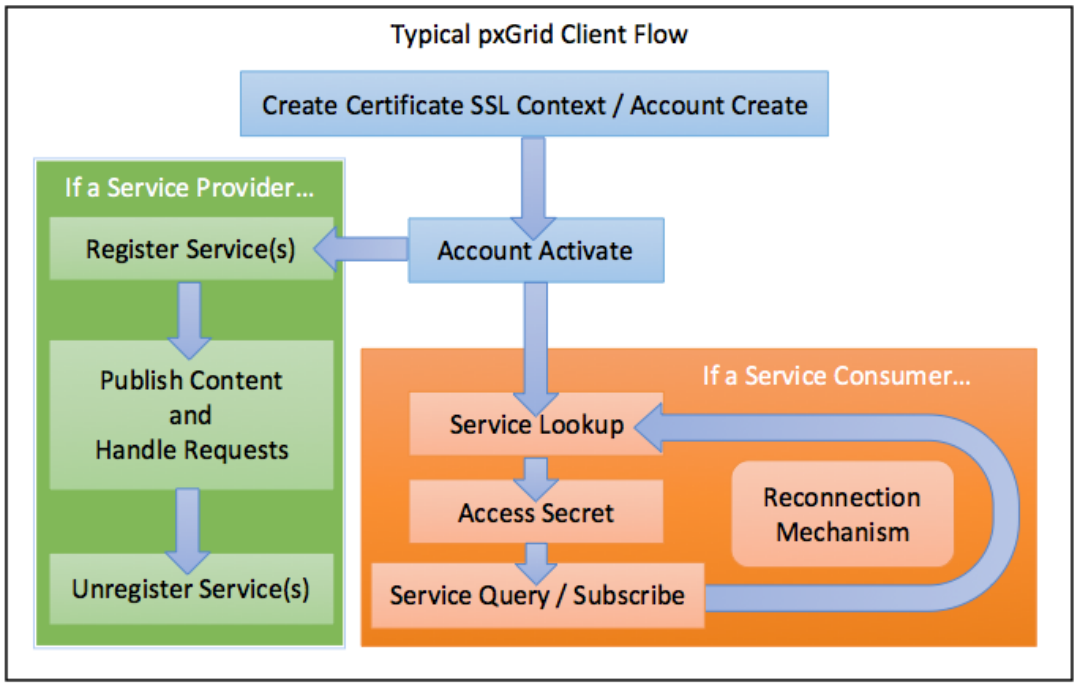
Chrome POSTMAN is a web-based REST Client that allows you to enter and monitor HTTP requests and responses. This will be used to explain the concepts of WebSockets & REST APIs.

Technical Overview

Cisco Platform Exchange Grid (pxGrid) 2.0 is based on a REpresentational State Transfer (REST) and WebSockets Model, over the Simple Text Oriented Protocol (STOMP) messaging protocol. WebSockets provide quick and scalable bi-directional data transfer, while REST provides quick extensible querying mechanisms – all over the same interface. pxGrid utilizes WebSockets for pubsub components while all one-shot queries (for both control and service data) is done via REST. All message bodies are formatted in JSON.

Cisco pxGrid will use port 8910 on ISE for pxGrid-related REST and WebSocket communications. Messages over Websockets will be sent and received in binary format, these messages should conform to the STOMP messaging protocol. For more information on supported commands on pxGrid please see: https://github.com/cisco-pxgrid/pxgrid-rest-ws/wiki

The following represents the typical pxGrid 2.0 client flow:



Create Certificate SSL Context / Account Create

All clients must authenticate to the ISE pxGrid controller either via certificate-based SSL authentication or username-password authentication.

Account Activate

All clients request to activate their accounts on the pxGrid server which is handled by the REST API.

Register/Unregister Service

Service providers will use these APIs to provide and update the necessary information (i.e. resource URLs) from which their services are accessible for other pxGrid clients.

Service Lookup

All clients can use this API to dynamically discover all available provider services and their locations.

Access Secret

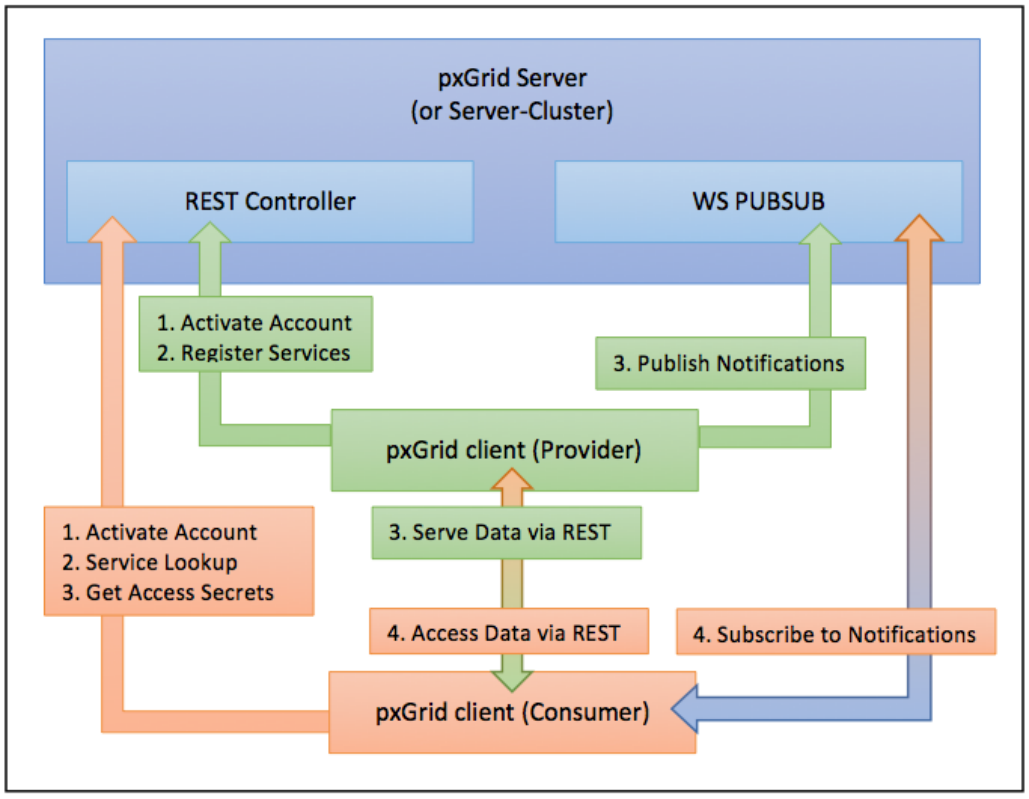
For every service returned that interests a particular client, that client must also query the pxGrid controller for an access secret in order to obtain the information provided by the service.

Service Query / Subscribe

With the access secret and service location information in hand, client can then perform REST-based queries or build WebSocket connections to receive information.

We will go through an example of authenticating and registering a pxGrid client using Chrome POSTMAN as the REST client. An account will be created and activated. The pxGrid client will request the session topic or service from the ISE pxGrid node via a service lookup. The service lookup will return the WebSockets URL (WsURL) resource and the ISE node that publishes this session topic. The pxGrid client will then obtain an access secret and retrieve the session information based on the WsURL resource.

The following is a visual representation with the REST and WebSocket architecture:



This representation shows the pxGrid client provider or publisher publishing notifications or topics to the pxGrid controller via WebSockets and another pxGrid client consumer subscribing to these notification or topics.

Cisco pxGrid Context-In follows the same pxGrid client flow and will be discussed in detail in the *Cisco pxGrid Context-in* Section.

Illustrating Cisco pxGrid Client Flow using Chrome Postman

Chrome POSTMAN will be used to illustrate Cisco pxGrid Client Flow using REST API. First Cisco pxGrid is enabled. Chrome POSTMAN can be downloaded from:
https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=en

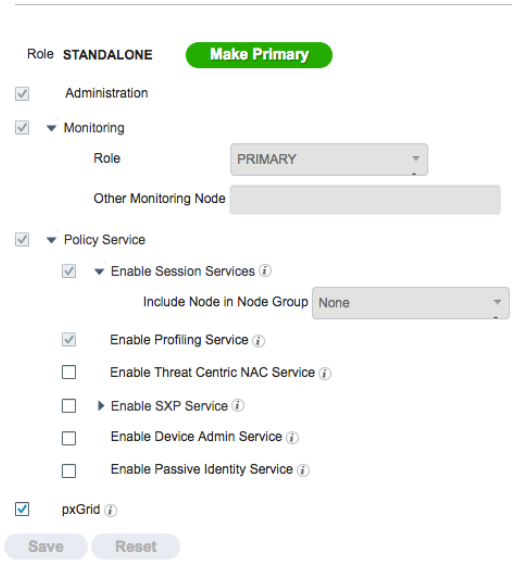
The following steps are as follows:

- Enabling pxGrid on the ISE Node
- Import the ISE identity certificates into your browser
- Creating the pxGrid client Account
- Enabling the Account
- Activating the Account
- Performing Services Lookup
- Obtaining the Access Secret
- Run GetSessionByIPAddress WebSockets REST API script

Please note that the pxGridcontol.java code sample uses the same logic. This is a good way to check your coding as well.

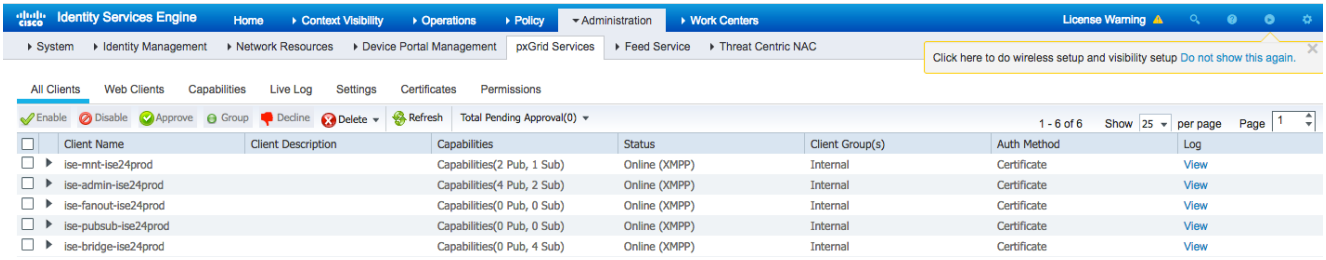
Enabling pxGrid

Step 1 Enable pxGrid
Select **Administration->System Deployment->Edit Node->Enable pxGrid**

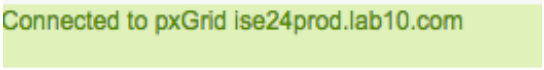


Step 2 Select **Save**

Step 3 Verify that the ISE published nodes appear
Select **Administration->pxGrid Services**



and you have connectivity

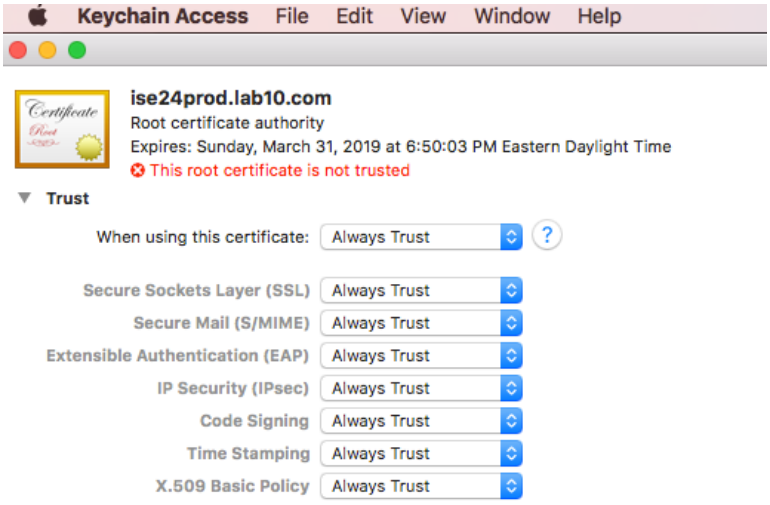


Exporting the ISE Identity Certificate into your Browser

Export the ISE identity certificate, into the PC's truststore. A MacBook Pro was used in this example.

References: https://github.com/cisco-pxgrid/pxgrid-rest-ws/wiki

Step 1 Import ISE identity certificate into Chrome browser and set to "always trust"



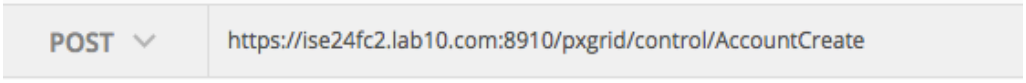
Step 2 Ensure that you have "Allowed password based account creation" enabled under **Administration->pxGrid Services->Settings->pxGrid Settings**

Step 3 Close out the browser and log back in again

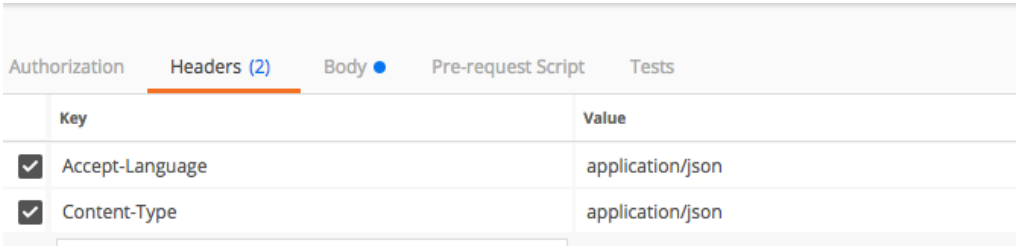
Creating the pxGrid client Account

The initial pxGrid client account will be created and returned username and password will be used to access other WebSocket REST calls. For example, this authentication will be used later to determine the ISE peer node for a service lookup that contains the session topic.

Step 1 Create an account and obtain the username and password that will be used for basic authentication for the other WebSocket REST API calls. In this exercise we will be using password-based authentication.

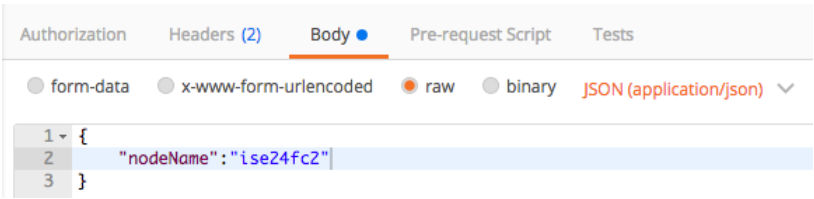


Step 2 Define the following Headers:

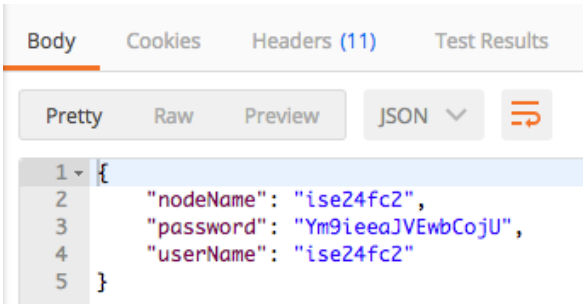


Step 3 The body should read:

Note: The nodeName provides the pxGrid client name. Please note that the nodename must be Fully Qualified Domain (FQDN) resolvable.



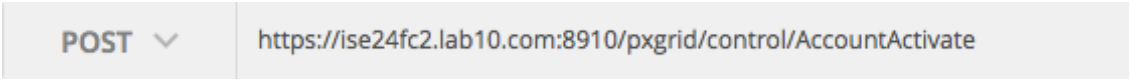
Step 4 You will receive the username and password that will be used for other WebSockets REST calls. The node name represents the pxGrid client node.



Activating the Account

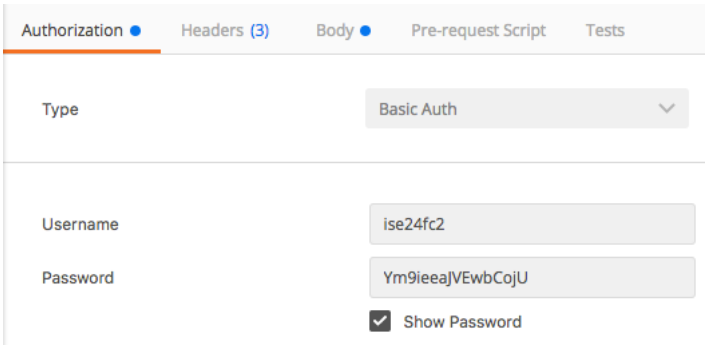
The account needs to be activated by the ISE or pxGrid admin before the pxGrid client can register to the ISE pxGrid node.

Step 1 Activate the account

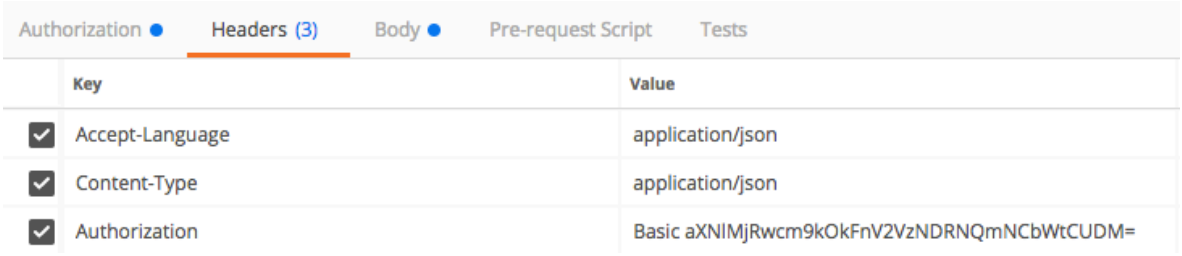


Step 2 Add the following authorization settings:

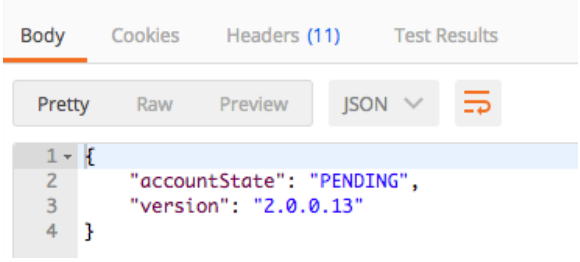
Note: The username and password are from step 6 above, or the results of the account creation.



Step 3 Add the following headers, the authorization header will appear after the username and password have been configured.

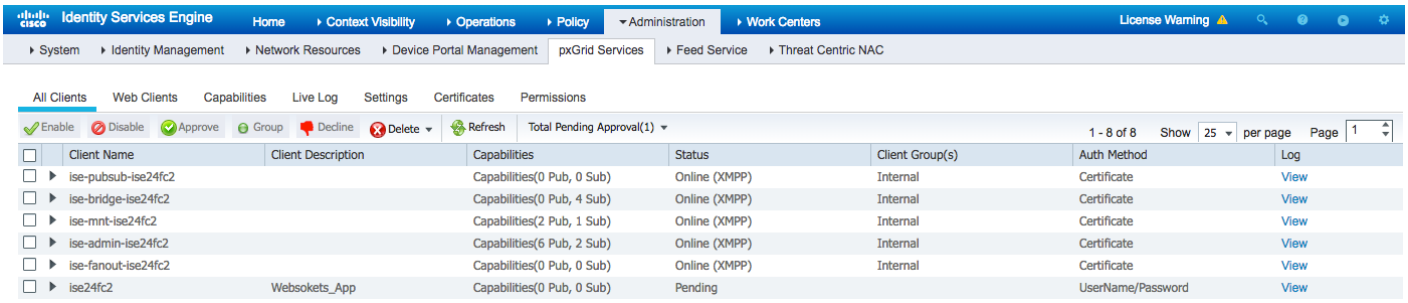


Step 4 You will be in the pending state until the ISE admin approves the account



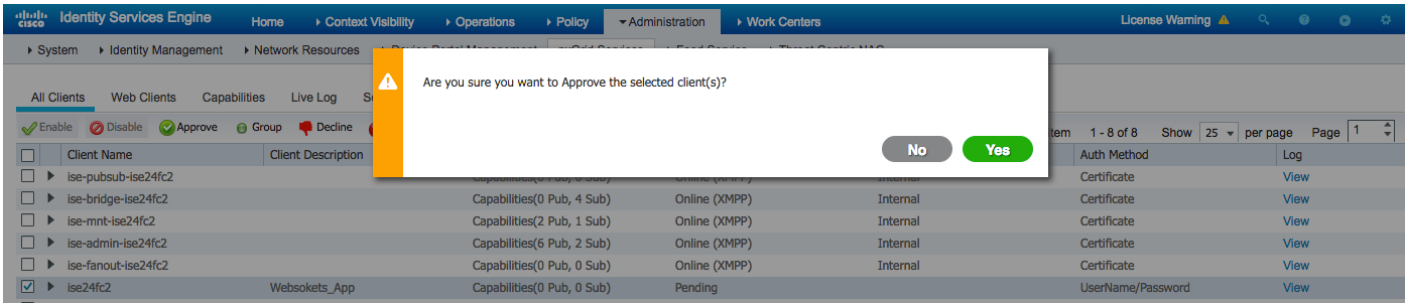
Step 5 In ISE, you will see pxGrid client name (nodename) in the pending state

Note: Regardless of enabled password-based authentication for client settings, the admin will still have to approve the pending client request.



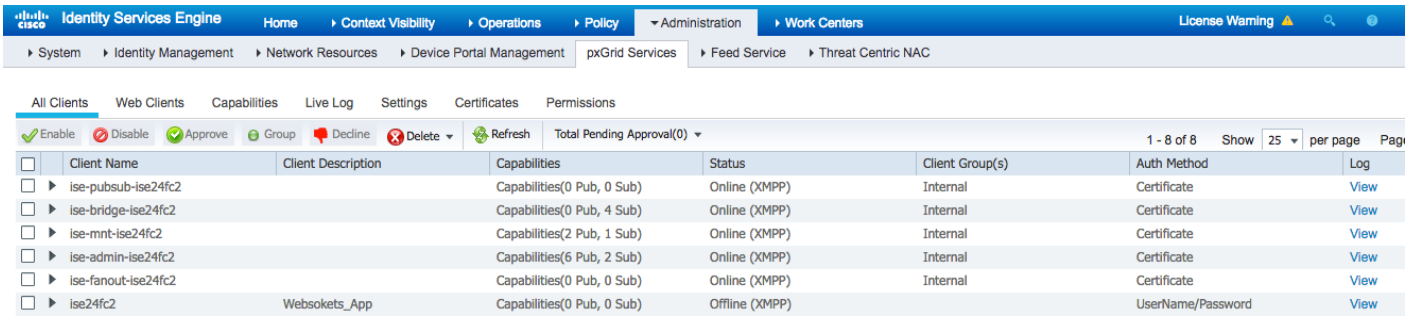
Step 6 Select "**ise24fc2**" and "**Approve**"

Step 7 You will see the following message:

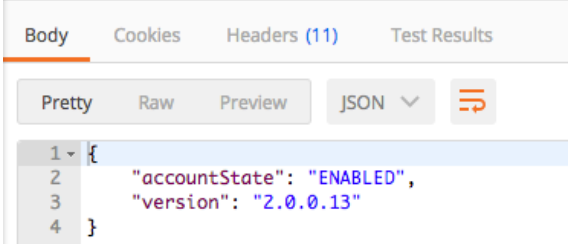


Step 8 Select "**Yes**"

Step 9 You should see:



Step 10 If you re-run REST client request again, the account will be enabled



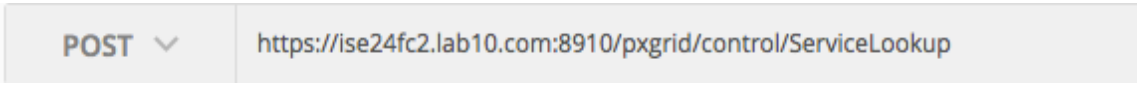
Performing Services Lookup

ServicesLookup is used to find a service and its properties. It determines what topics or services are available on the ISE peer node that publishes these topics. Examples of these topics are Session Directory, RADIUS failures, MDM, Profiler Configuration, System Health, TrustSec, TrustSec Configuration and TrustSec SXP. For more information, please see: https://github.com/cisco-pxgrid/pxgrid-rest-ws/wiki/pxGrid-Consumer

In the case of publishing Internet of Things (IOT) asset devices, this will return the services of the pxGrid client publishing the asset topic.

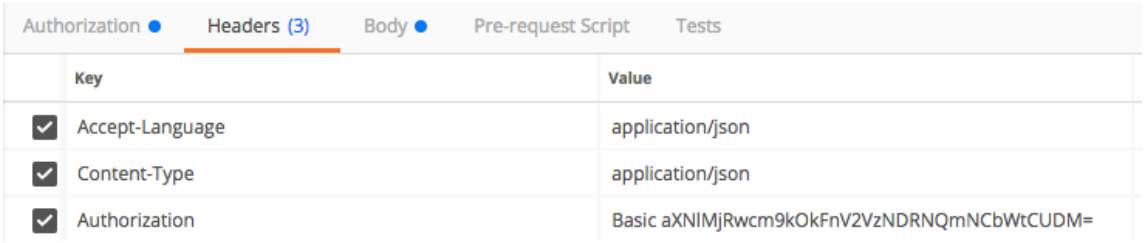
In this example, we are interested in seeing which node publishes the session information.

Step 1 Run Service Lookup, to see what services are available on all the ISE nodes

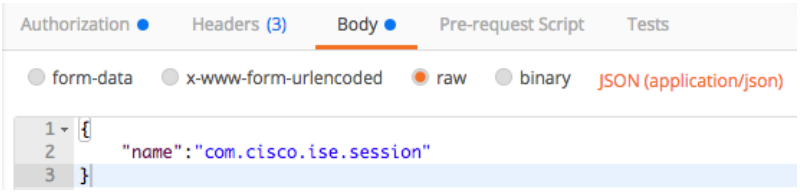


Step 2 The Headers should read:

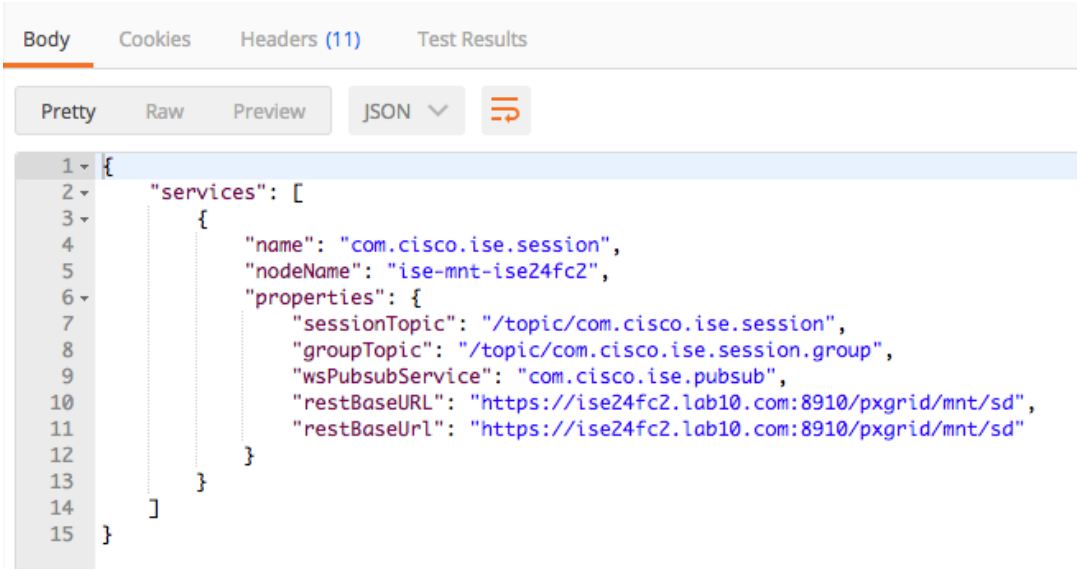
Note: The Authorization header should be the username and password from step 8, when creating the account



Step 3 We request session topic which can be found here: https://github.com/cisco-pxgrid/pxgrid-rest-ws/wiki/Session-Directory



Step 4 Note the nodeName or the ISE node that publishes the session information and the properties JSON example:

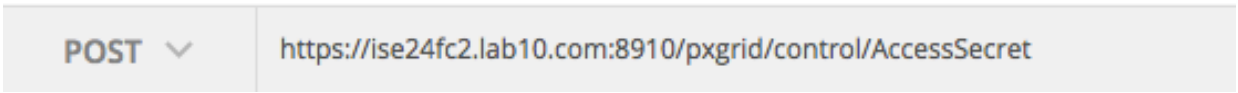


The **sessionTopic "/topic/com.cisco.com.ise.session"** contains the session attributes of the authenticated endpoint. The **restBaseUrl** https://ise24fc2.lab10.com:8910/pxgrid/mnt/sd will be used later on in the REST Client request to get all authenticated sessions.
The **wsPubsubService "com.cisco.ise.pubsub"** contains the WebSockets published service.

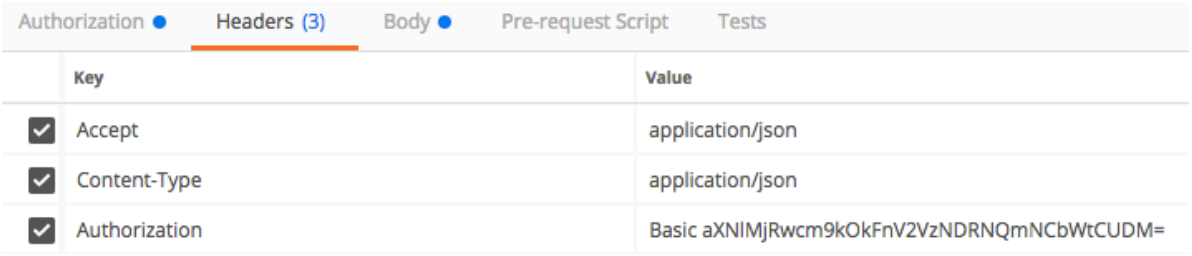
Obtaining the Access Secret

The Access Secret provides HTTP Basic Auth access to the ISE node that will publish the session information. This will be used in the REST Client API to obtain session information when querying based on an IP address.

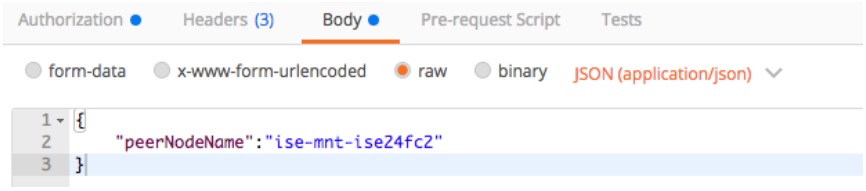
Step 1 Obtain access secret



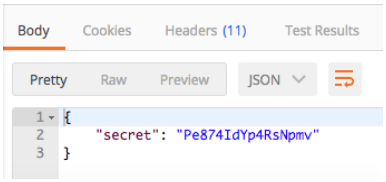
Step 2 The Headers are as follows



Step 3 Specify the ISE node that will publish the session information as provided by the returned service lookup results.



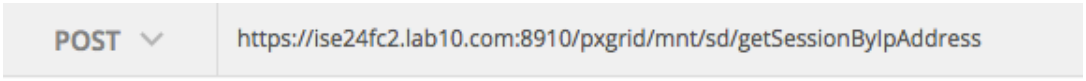
Step 4 The access secret is returned below:



Run GetSessionByIPAddress WebSockets REST API Script

The Get Session By IP RESTAPI returns available session information from an IP address query. For more information please see: https://github.com/cisco-pxgrid/pxgrid-rest-ws/wiki/Session-Directory

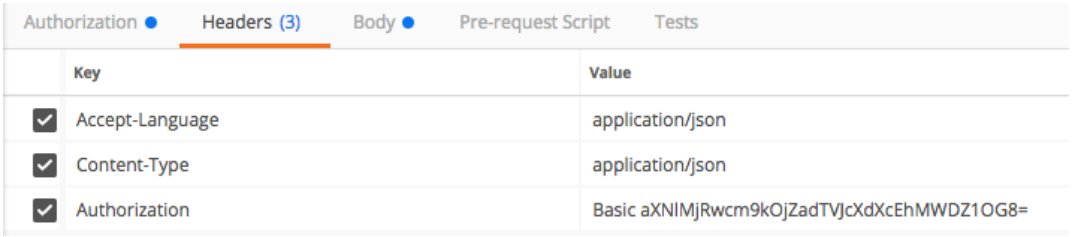
Step 1 Run the getSessionByIpAddress REST API script below:



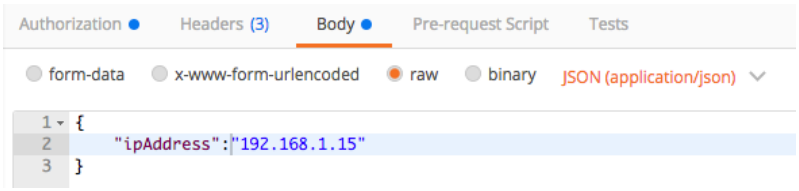
Note that the **restBaseUrl** https://ise24fc2.lab10.com:8910/pxgrid/mnt/sd is used in the script

Step 2 Please use the following header information

Note: The authorization header contains the username from step 8, creating the initial account, and the password contains the access secret password



Step 3 Submit the IP address you want to perform a query action on

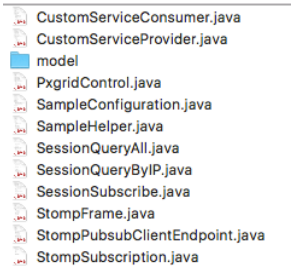


Step 4 Below are the returned session attributes for the endpoint IP address of 192.168.1.15

Java Code Examples

You can download Java code examples once registered to Cisco Devnet www.cisco.com/go/pxgridpartner from https://github.com/cisco-pxgrid/pxgrid-rest-ws

The code samples are as follows:



For this example, we will run the SessionSubscribe.java code, to see the available contextual information from an authenticated user session.

You can also develop code to subscribe and obtain contextual information from other topics: Radius Failure, these topics are Session Directory, RADIUS failures, MDM, Profiler Configuration, System Health, TrustSec, TrustSec Configuration and TrustSec SXP. For more information, please see: https://github.com/cisco-pxgrid/pxgrid-rest-ws/wiki/pxGrid-Consumer

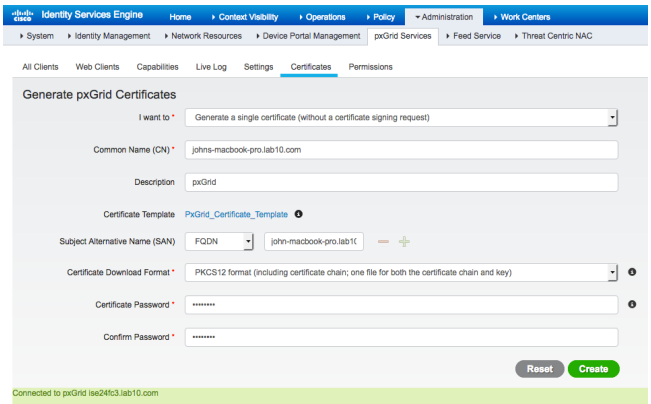
First, we will need to create certificates your system, which will be the pxGrid client. The pxGrid client will authenticate to the ISE pxGrid node using certificates or pre-shared keys, in this example we will use certificates generated from the ISE internal Certificate Authority (CA).

Generating pxGrid client Certificates from ISE Internal CA

In this example, we will create the pxGrid client certificate in PKCS12 format. You will also want to test in PEM format, when you go for your pxGrid certification.

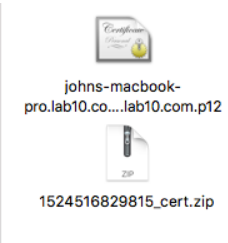
Step 1 Select **Administration->pxGrid Services->Certificates->** provide the following information

Note: CN name should be Fully Qualified Domain Name (FQDN) resolvable. PKCS12 format is not supported using Python libraries.



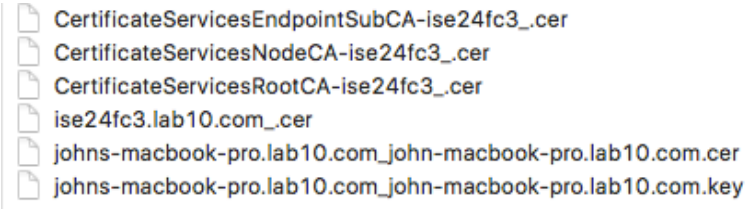
Step 2 Select **Create**

Step 3 Download the zipped file
You should see:



Step 4 Note, if using PEM format, when you unzip the file you will see the following:

Note: Please refer to https://communities.cisco.com/docs/DOC-71928 , using iSE 2.2 Internal CA to deploy to pxGrid clients (java keystores), for Productional ISE deployments, please refer to: https://communities.cisco.com/docs/DOC-68284



Converting Certificates to JKS format

The PKCS12 certificate is converted into the keystore filename and truststore filename.

```
openssl pkcs12 -export -out session.p12 -inkey Johns-Macbook-Pro.lab10.com_Johns-Macook-Pro.lab10.com.key -in
Johns-Macbook-Pro.lab10.com_Johns-Macook-Pro.lab10.com.cer -chain -CAfile CA1.cer
Enter pass phrase for Johns-Macbook-Pro.lab10.com_Johns-Macook-Pro.lab10.com.key:
Enter Export Password: Cisco123
Verifying - Enter Export Password: Cisco123

keytool -importkeystore -srckeystore session.p12 -destkeystore session1.jks -srcstoretype PKCS12
Enter destination keystore password:  Cisco123
Re-enter new password: Cisco123
Enter source keystore password:  Cisco123
Entry for alias 1 successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or cancelled

openssl x509 -outform der -in CA1.cer -out CA1.der
keytool -import -alias session1 -keystore rootsession.jks -file CA1.der
Enter keystore password:  Cisco123
Re-enter new password: Cisco123
Owner: CN=Certificate Services Endpoint Sub CA - ise24fc3
Issuer: CN=Certificate Services Node CA - ise24fc3
Serial number: 589713fe8d1d4c99b580aae99e862c4f
Valid from: Thu Apr 12 22:42:16 EDT 2018 until: Thu Apr 13 22:42:14 EDT 2028
Certificate fingerprints:
         MD5:  8E:B3:9F:92:B8:E4:80:51:64:68:4C:72:44:51:15:3F
         SHA1: 5D:EF:20:E1:9C:CA:5D:F7:15:28:FA:1D:4D:4F:A9:79:CD:E5:A6:FC
         SHA256:
FB:C5:84:4B:30:D3:8E:95:B9:FE:28:54:EC:60:A7:E4:4B:A7:6D:1C:8D:8C:0C:15:C0:4B:2C:37:4A:43:8F:0C
         Signature algorithm name: SHA256withRSA
         Version: 3
```

```
Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 10 B0 23 A7 DF 3C F7 9D   54 F7 B2 34 12 21 DE 91  ..#..<..T..4.!..
0010: A4 6D 33 CE                                        .m3.
]
[CN=Certificate Services Root CA - ise24fc3]
SerialNumber: [    6f56a636 30094fa4 b4b85ac9 4f5def5b]
]

#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#3: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  Key_CertSign
]

#4: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 33 47 E4 40 4B 5E 0C 08   77 DE A2 77 30 50 E9 3C  3G.@K^..w..w0P.<
0010: 12 78 92 39                                        .x.9
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore

keytool -import -alias session2 -keystore session1.jks -file Johns-Macbook-Pro.lab10.com_Johns-Macook-
Pro.lab10.com.cer
Enter keystore password:  Cisco123
Certificate already exists in keystore under alias <1>
Do you still want to add it? [no]:  yes
Certificate was added to keystore

keytool -import -alias session3 -keystore rootsession.jks -file CA1.cer
Enter keystore password:  Cisco123
Certificate already exists in keystore under alias <session1>
Do you still want to add it? [no]:  yes
Certificate was added to keystore

keytool -import -alias session4 -keystore rootsession.jks -file CertificateServicesRootCA-ise24fc3_.cer
Enter keystore password: Cisco123
Owner: CN=Certificate Services Root CA - ise24fc3
Issuer: CN=Certificate Services Root CA - ise24fc3
Serial number: 23471fb4679a4836b6023da18e312e3e
Valid from: Thu Apr 12 22:42:14 EDT 2018 until: Thu Apr 13 22:42:14 EDT 2028
Certificate fingerprints:
         MD5:  94:EA:6F:D5:E6:D6:A4:53:D2:69:7E:C6:6F:02:AB:2D
         SHA1: F8:8A:36:C8:45:F0:A5:01:32:32:E0:8D:59:E4:F9:A2:24:A6:71:47
         SHA256:
19:2F:41:EC:93:C7:EE:BB:CC:22:AB:44:24:FF:95:AF:E0:5F:5F:30:F9:D4:7C:84:43:91:93:A6:47:1C:67:97
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  Key_CertSign
```

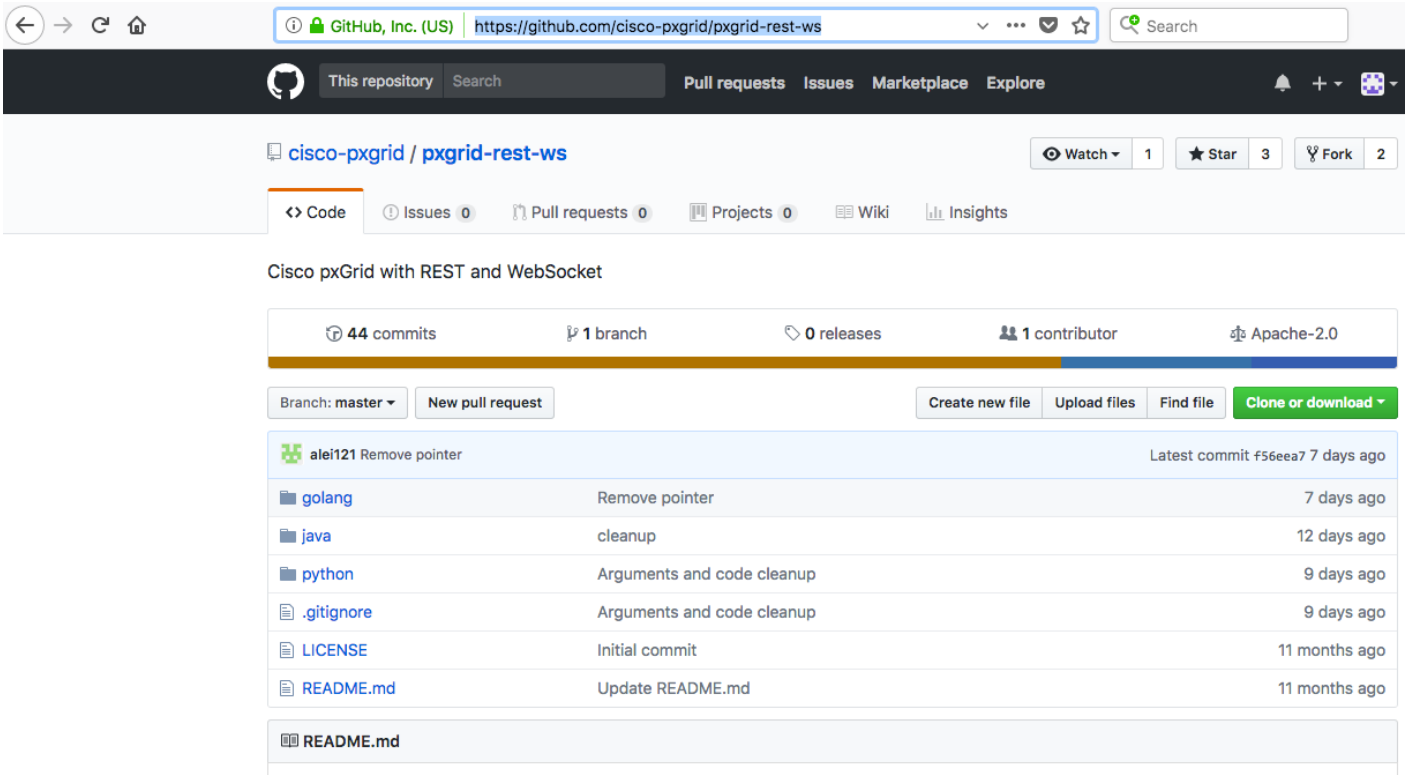
```
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: AE 42 CE AB 57 30 7C 75   F8 10 94 25 0E DC DF FA  .B..W0.u...%....
0010: 9E 7F 3A 57                                        ..:W
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Downloading Java Examples

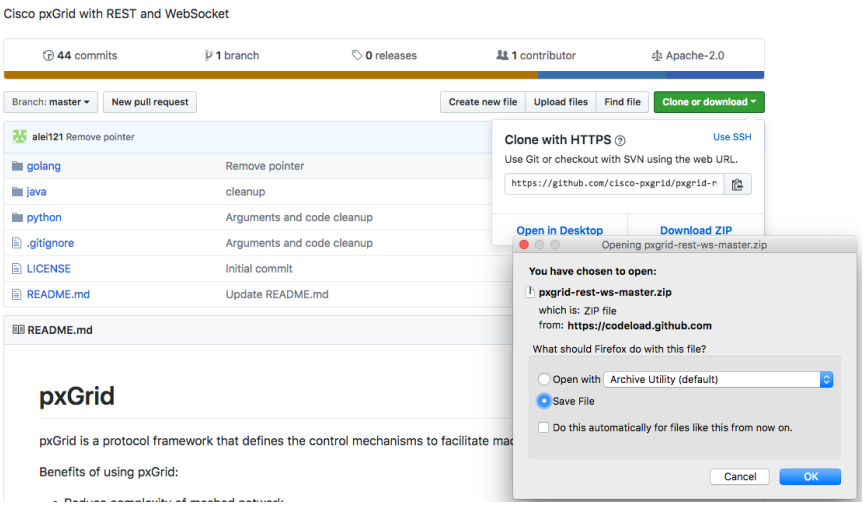
In this section, java code examples are downloaded and imported into an Eclipse project

Step 1 Register for devnet: www.cisco.com/go/pxgridpartner

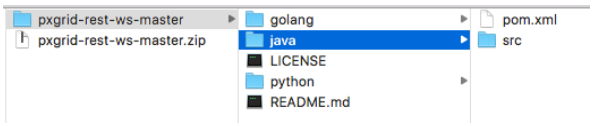
Step 2 Open browser to https://github.com/cisco-pxgrid/pxgrid-rest-ws



Step 3 Download the zipped file and save locally



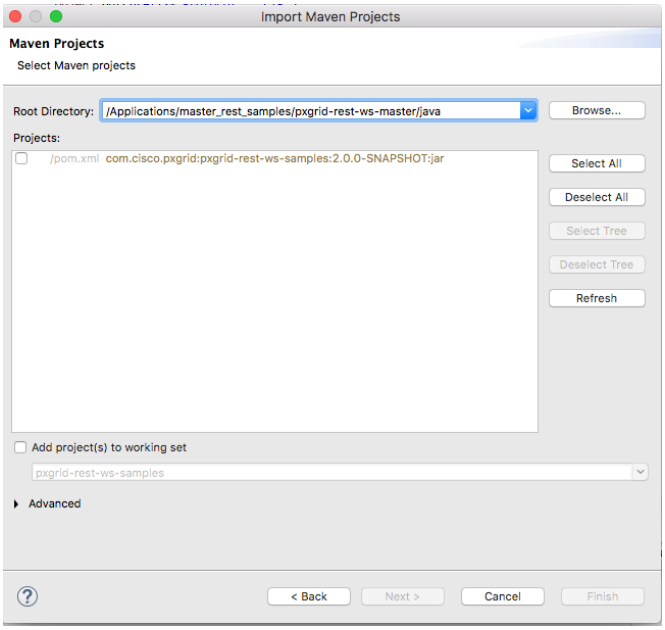
Step 4 Unzip the file
You should see:



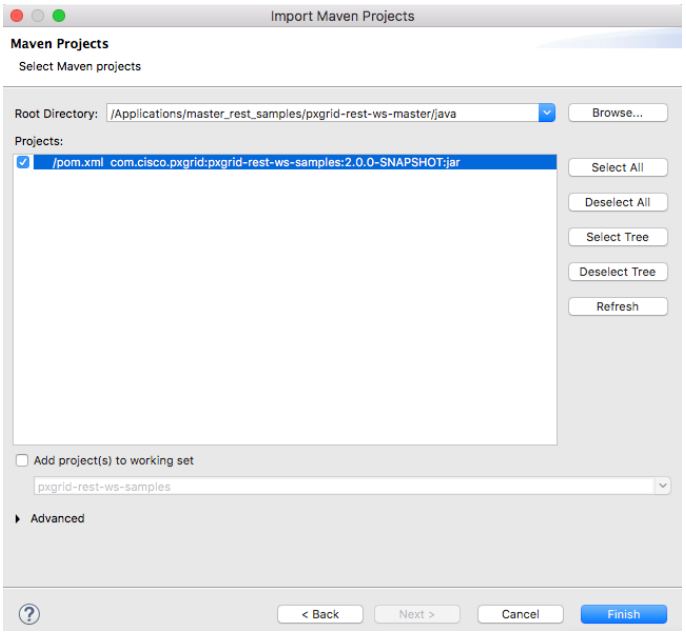
Step 5 Select **File->Import->Maven->Existing Maven Projects**

Step 6 Select **Next->enter root directory where the POM file is located**

Note: The POM.xml file will be enabled



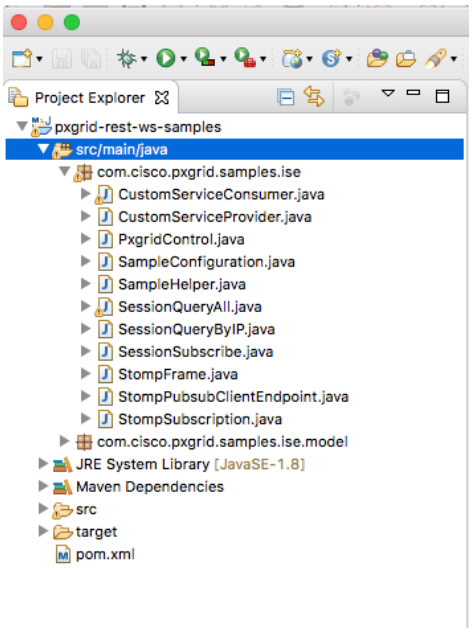
Step 7 Highlight the snapshot.jar file



Step 8 Select **Finish**

Step 9 You should see the imported jar file and the following samples:

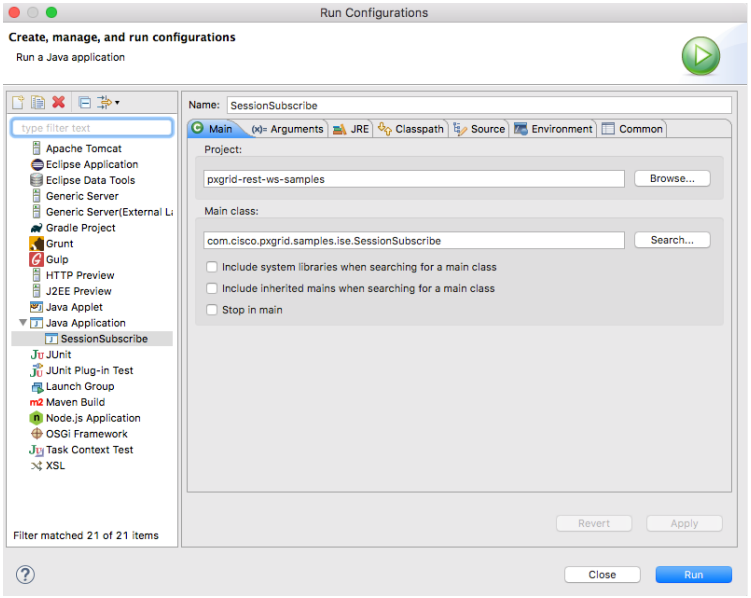
Note: If you do not see the scripts, select R**un->Run As->Maven Clean**



SessionSubscribe Coding Example

This is a common example of subscribing to the session directory service and receiving incoming session notifications in real-time

Step 1 Under **com.cisco.pxgrid.samples.ise->SessionSubscribe, Right-Click on "mainStringID:void->Run As->Run Configurations**
You will see



Step 2 Select **(x) Arguments,** and enter the following:

Note: The keystorefile name, keystorepassword, truststore filename and truststore password are from the earlier step in generating certificates.

```
-a ise24fc3.lab10.com -u mac03 -k /Applications/master_rest_samples/session1.jks -p Cisco123 -t
/Applications/master_rest_samples/rootsession.jks -q Cisco123
```

where the argument usage is:

```
usage: SessionSubscribe
 -a,--hostname <arg>              Host name (multiple accepted)
 -d,--description <arg>           Description (optional)
 -k,--keystorefilename <arg>      Keystore .jks filename (not required if
                                  password is specified)
 -p,--keystorepassword <arg>      Keystore password (not required if
                                  password is specified)
 -q,--truststorepassword <arg>    Truststore password
 -t,--truststorefilename <arg>    Truststore .jks filename
 -u,--nodename <arg>              Node name
 -w,--password <arg>              Password (not required if keystore is
                                  specified)
```

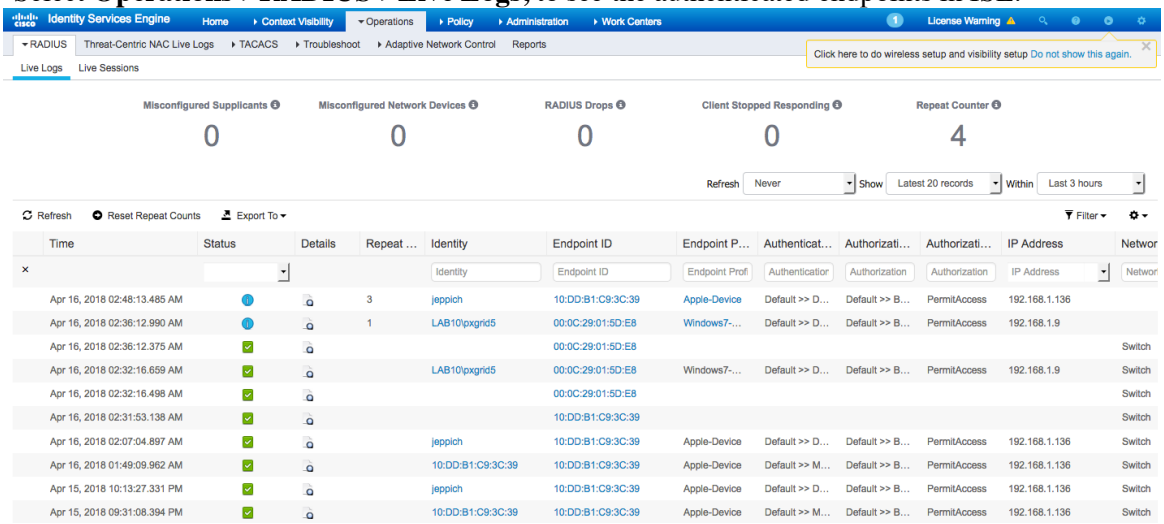
Step 3 Select **Apply**
Step 4 Select **Run**
Step 5 If you have active user authenticated sessions coming in, you will see the following

```
----- config ------
```

```
  hostname = ise24fc3.lab10.com
  nodename = mac03
  password = (not specified)
  description = (not specified)
  keystorefilename = /Applications/master_rest_samples/session1.jks
  keystorepassword = Cisco123
  truststorefilename = /Applications/master_rest_samples/rootsession.jks
  truststorepassword = Cisco123
--------------------
22:29:27.354 [main] INFO com.cisco.pxgrid.samples.ise.PxgridControl - AccountActivate request={}
22:29:32.701 [main] INFO com.cisco.pxgrid.samples.ise.PxgridControl - AccountActivate
response={"accountState":"ENABLED","version":"2.0.0.13"}
22:29:32.701 [main] INFO com.cisco.pxgrid.samples.ise.SessionSubscribe - pxGrid controller version=2.0.0.13
22:29:32.703 [main] INFO com.cisco.pxgrid.samples.ise.PxgridControl - ServiceLookup
request={"name":"com.cisco.ise.session"}
22:29:32.722 [main] INFO com.cisco.pxgrid.samples.ise.PxgridControl - ServiceLookup
response={"services":[{"name":"com.cisco.ise.session","nodeName":"ise-mnt-
ise24fc3","properties":{"sessionTopic":"/topic/com.cisco.ise.session","groupTopic":"/topic/com.cisco.ise.sess
ion.group","wsPubsubService":"com.cisco.ise.pubsub","restBaseURL":"https://ise24fc3.lab10.com:8910/pxgrid/mnt
/sd","restBaseUrl":"https://ise24fc3.lab10.com:8910/pxgrid/mnt/sd"}}]}
22:29:32.722 [main] INFO com.cisco.pxgrid.samples.ise.SessionSubscribe -
wsPubsubServiceName=com.cisco.ise.pubsub sessionTopic=/topic/com.cisco.ise.session
22:29:32.722 [main] INFO com.cisco.pxgrid.samples.ise.PxgridControl - ServiceLookup
request={"name":"com.cisco.ise.pubsub"}
22:29:32.733 [main] INFO com.cisco.pxgrid.samples.ise.PxgridControl - ServiceLookup
response={"services":[{"name":"com.cisco.ise.pubsub","nodeName":"ise-pubsub-
ise24fc3","properties":{"wsUrl":"wss://ise24fc3.lab10.com:8910/pxgrid/ise/pubsub"}}]}
22:29:32.733 [main] INFO com.cisco.pxgrid.samples.ise.SessionSubscribe -
wsUrl=wss://ise24fc3.lab10.com:8910/pxgrid/ise/pubsub
22:29:32.735 [main] INFO com.cisco.pxgrid.samples.ise.PxgridControl - AccessSecret
request={"peerNodeName":"ise-pubsub-ise24fc3"}
22:29:32.895 [main] INFO com.cisco.pxgrid.samples.ise.PxgridControl - AccessSecret
response={"secret":"36QiBjNQdWnR6lPf"}
22:29:34.805 [Grizzly(1)] INFO com.cisco.pxgrid.samples.ise.StompPubsubClientEndpoint - WS onOpen
22:29:34.833 [main] INFO com.cisco.pxgrid.samples.ise.StompPubsubClientEndpoint - STOMP CONNECT
host=ise24fc3.lab10.com
22:29:34.840 [main] INFO com.cisco.pxgrid.samples.ise.StompPubsubClientEndpoint - STOMP SUBSCRIBE
topic=/topic/com.cisco.ise.session
22:29:34.842 [Grizzly(2)] INFO com.cisco.pxgrid.samples.ise.StompPubsubClientEndpoint - STOMP CONNECTED
version=1.2
press <enter> to disconnect...
22:31:57.678 [Grizzly(1)] INFO com.cisco.pxgrid.samples.ise.SessionSubscribe -
Content={"sessions":[{"timestamp":"2018-04-
16T02:31:53.599Z","state":"STARTED","userName":"jeppich","callingStationId":"10:DD:B1:C9:3C:39","calledStatio
nId":"50:3D:E5:C4:05:8C","auditSessionId":"0A0000010000002D02D9EFBF","ipAddresses":["192.168.1.136"],"macAddr
ess":"10:DD:B1:C9:3C:39","nasIpAddress":"192.168.1.3","nasPortId":"GigabitEthernet1/0/12","nasPortType":"Ethe
rnet","endpointProfile":"Apple-Device","endpointOperatingSystem":"Apple Mac OS X 10.7.0 (Lion) - 10.10
(Yosemite) or iOS 4.1 - 8.3 (Darwin 10.0.0 -
14.5.0)","adNormalizedUser":"jeppich","adUserDomainName":"lab10.com","adUserNetBiosName":"LAB10","adUserResol
vedIdentities":"jeppich@lab10.com","adUserResolvedDns":"CN\u003dJohn
Eppich,CN\u003dUsers,DC\u003dlab10,DC\u003dcom","providers":["None"],"endpointCheckResult":"none","identitySo
urcePortStart":0,"identitySourcePortEnd":0,"identitySourcePortFirst":0,"isMachineAuthentication":"false","ser
viceType":"Framed","networkDeviceProfileName":"Cisco","radiusFlowType":"Wired802_1x","ssid":"50-3D-E5-C4-05-
8C","mdmRegistered":false,"mdmCompliant":false,"mdmDiskEncrypted":false,"mdmJailBroken":false,"mdmPinLocked":
false}]}
22:32:21.684 [Grizzly(1)] INFO com.cisco.pxgrid.samples.ise.SessionSubscribe -
Content={"sessions":[{"timestamp":"2018-04-
16T02:32:17.261Z","state":"STARTED","userName":"LAB10\\pxgrid5","callingStationId":"00:0C:29:01:5D:E8","calle
dStationId":"50:3D:E5:C4:05:96","auditSessionId":"0A0000010000001900003320D","ipAddresses":["192.168.1.9"],"ma
cAddress":"00:0C:29:01:5D:E8","nasIpAddress":"192.168.1.3","nasPortId":"GigabitEthernet1/0/22","nasPortType":
"Ethernet","endpointProfile":"Windows7-Workstation","endpointOperatingSystem":"Windows 7
Professional","adNormalizedUser":"pxgrid5","adUserDomainName":"lab10.com","adUserNetBiosName":"LAB10","adUser
ResolvedIdentities":"pxgrid5@lab10.com","adUserResolvedDns":"CN\u003dpxgrid5,CN\u003dUsers,DC\u003dlab10,DC\u
003dcom","providers":["None"],"endpointCheckResult":"none","identitySourcePortStart":0,"identitySourcePortEnd
":0,"identitySourcePortFirst":0,"isMachineAuthentication":"false","serviceType":"Framed","networkDeviceProfil
eName":"Cisco","radiusFlowType":"Wired802_1x","ssid":"50-3D-E5-C4-05-
96","mdmRegistered":false,"mdmCompliant":false,"mdmDiskEncrypted":false,"mdmJailBroken":false,"mdmPinLocked":
false}]}
22:36:17.720 [Grizzly(2)] INFO com.cisco.pxgrid.samples.ise.SessionSubscribe -
Content={"sessions":[{"timestamp":"2018-04-
16T02:36:12.99Z","state":"STARTED","userName":"LAB10\\pxgrid5","callingStationId":"00:0C:29:01:5D:E8","called
StationId":"50:3D:E5:C4:05:96","auditSessionId":"0A0000010000001900003320D","ipAddresses":["192.168.1.9"],"mac
Address":"00:0C:29:01:5D:E8","nasIpAddress":"192.168.1.3","nasPortId":"GigabitEthernet1/0/22","nasPortType":"
```

```
Ethernet","endpointProfile":"Windows7-Workstation","endpointOperatingSystem":"Windows 7
Professional","adNormalizedUser":"pxgrid5","adUserDomainName":"lab10.com","adUserNetBiosName":"LAB10","adUser
ResolvedIdentities":"pxgrid5@lab10.com","adUserResolvedDns":"CN\u003dpxgrid5,CN\u003dUsers,DC\u003dlab10,DC\u
003dcom","providers":["None"],"endpointCheckResult":"none","identitySourcePortStart":0,"identitySourcePortEnd
":0,"identitySourcePortFirst":0,"isMachineAuthentication":"false","serviceType":"Framed","networkDeviceProfil
eName":"Cisco","radiusFlowType":"Wired802_1x","ssid":"50-3D-E5-C4-05-
96","mdmRegistered":false,"mdmCompliant":false,"mdmDiskEncrypted":false,"mdmJailBroken":false,"mdmPinLocked":
false}]}
22:49:17.586 [Grizzly(2)] INFO com.cisco.pxgrid.samples.ise.SessionSubscribe -
Content={"sessions":[{"timestamp":"2018-04-
16T02:47:32.481Z","state":"DISCONNECTED","userName":"jeppich","callingStationId":"10:DD:B1:C9:3C:39","calledS
tationId":"50:3D:E5:C4:05:8C","auditSessionId":"0A0000010000002D02D9EFBF","ipAddresses":["192.168.1.136"],"ma
cAddress":"10:DD:B1:C9:3C:39","nasIpAddress":"192.168.1.3","nasPortId":"GigabitEthernet1/0/12","nasPortType":
"Ethernet","endpointProfile":"Apple-Device","endpointOperatingSystem":"Apple Mac OS X 10.7.0 (Lion) - 10.10
(Yosemite) or iOS 4.1 - 8.3 (Darwin 10.0.0 -
14.5.0)","adNormalizedUser":"jeppich","adUserDomainName":"lab10.com","adUserNetBiosName":"LAB10","adUserResol
vedIdentities":"jeppich@lab10.com","adUserResolvedDns":"CN\u003dJohn
Eppich,CN\u003dUsers,DC\u003dlab10,DC\u003dcom","providers":["None"],"endpointCheckResult":"none","identitySo
urcePortStart":0,"identitySourcePortEnd":0,"identitySourcePortFirst":0,"isMachineAuthentication":"false","ser
viceType":"Framed","networkDeviceProfileName":"Cisco","radiusFlowType":"Wired802_1x","ssid":"50-3D-E5-C4-05-
8C","mdmRegistered":false,"mdmCompliant":false,"mdmDiskEncrypted":false,"mdmJailBroken":false,"mdmPinLocked":
false}]}
22:49:17.588 [Grizzly(2)] INFO com.cisco.pxgrid.samples.ise.SessionSubscribe -
Content={"sessions":[{"timestamp":"2018-04-
16T02:48:13.485Z","state":"STARTED","userName":"jeppich","callingStationId":"10:DD:B1:C9:3C:39","calledStatio
nId":"50:3D:E5:C4:05:8C","auditSessionId":"0A0000010000002F02F84944","ipAddresses":["192.168.1.136"],"macAddr
ess":"10:DD:B1:C9:3C:39","nasIpAddress":"192.168.1.3","nasPortId":"GigabitEthernet1/0/12","nasPortType":"Ethe
rnet","endpointProfile":"Apple-Device","endpointOperatingSystem":"Apple Mac OS X 10.7.0 (Lion) - 10.10
(Yosemite) or iOS 4.1 - 8.3 (Darwin 10.0.0 -
14.5.0)","adNormalizedUser":"jeppich","adUserDomainName":"lab10.com","adUserNetBiosName":"LAB10","adUserResol
vedIdentities":"jeppich@lab10.com","adUserResolvedDns":"CN\u003dJohn
Eppich,CN\u003dUsers,DC\u003dlab10,DC\u003dcom","providers":["None"],"endpointCheckResult":"none","identitySo
urcePortStart":0,"identitySourcePortEnd":0,"identitySourcePortFirst":0,"isMachineAuthentication":"false","ser
viceType":"Framed","networkDeviceProfileName":"Cisco","radiusFlowType":"Wired802_1x","ssid":"50-3D-E5-C4-05-
8C","mdmRegistered":false,"mdmCompliant":false,"mdmDiskEncrypted":false,"mdmJailBroken":false,"mdmPinLocked":
false}]}
22:49:17.618 [Grizzly(2)] INFO com.cisco.pxgrid.samples.ise.StompPubsubClientEndpoint - WS onClose
closeReason code=VIOLATED_POLICY phrase=Did not receive a pong: too slow ...
```


Step 6 Select **Operations->RADIUS->Live Logs,** to see the authenticated endpoints in ISE.



Rest of Java Code Examples

These rest of the java code examples are explained

Sample Configuration

This code configures the –D PXGRID Hostname, -D PXGRID Username, -D PXGRID PASSWORD, -D PXGRID GROUP, -D PXGRID Description, -D PXGRID Keystore_Filename, -D PXGRID Keystore_Psssword, -D PXGRID Truststore Filename, and –D Truststore Password values. This also sets up the keystores.

```
package com.cisco.pxgrid.samples.ise.http;

import java.io.FileInputStream;
import java.io.IOException;
import java.net.Authenticator;
import java.net.PasswordAuthentication;
import java.net.Socket;
import java.security.GeneralSecurityException;
import java.security.KeyStore;
import java.security.Principal;
import java.security.PrivateKey;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.util.Collection;
import java.util.Enumeration;

import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.KeyManager;
import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.net.ssl.X509KeyManager;
import javax.net.ssl.X509TrustManager;

public class SampleConfiguration {
    protected final static String PROP_HOSTNAMES="PXGRID_HOSTNAMES";
    protected final static String PROP_USERNAME="PXGRID_USERNAME";
    protected final static String PROP_PASSWORD="PXGRID_PASSWORD";
    protected final static String PROP_GROUP="PXGRID_GROUP";
    protected final static String PROP_DESCRIPTION="PXGRID_DESCRIPTION";
    protected final static String PROP_KEYSTORE_FILENAME="PXGRID_KEYSTORE_FILENAME";
    protected final static String PROP_KEYSTORE_PASSWORD="PXGRID_KEYSTORE_PASSWORD";
    protected final static String PROP_TRUSTSTORE_FILENAME="PXGRID_TRUSTSTORE_FILENAME";
    protected final static String PROP_TRUSTSTORE_PASSWORD="PXGRID_TRUSTSTORE_PASSWORD";

    private String[] hostnames;
    private String username;
    private String password;
    private String[] groups;
    private String description;
    private SSLContext sslContext;

    private String keystoreFilename;
    private String keystorePassword;
    private String truststoreFilename;
    private String truststorePassword;

        public SampleConfiguration() throws GeneralSecurityException, IOException {
                load();
                print();
        }
```

```
    public String getUserName() {
                return username;
        }

    public void setUsername(String username) {
                this.username = username;
        }

        public String[] getGroups() {
                return groups;
        }

        public String getDescription() {
                return description;
        }

    public SSLContext getSSLContext() {
        return sslContext;
    }

    public String getPassword() {
                return password;
        }

    public String[] getHostnames() {
                return hostnames;
        }

    private void load() throws GeneralSecurityException, IOException {
        String hostnameProperty = System.getProperty(PROP_HOSTNAMES);
        username = System.getProperty(PROP_USERNAME);
        password = System.getProperty(PROP_PASSWORD);
        String group_property = System.getProperty(PROP_GROUP);
        description = System.getProperty(PROP_DESCRIPTION);

        keystoreFilename = System.getProperty(PROP_KEYSTORE_FILENAME);
        keystorePassword = System.getProperty(PROP_KEYSTORE_PASSWORD);
        truststoreFilename = System.getProperty(PROP_TRUSTSTORE_FILENAME);
        truststorePassword = System.getProperty(PROP_TRUSTSTORE_PASSWORD);

        if (hostnameProperty == null || hostnameProperty.isEmpty()) throw new
IllegalArgumentException("Missing " + PROP_HOSTNAMES);
        if (username == null || username.isEmpty()) throw new IllegalArgumentException("Missing " +
PROP_USERNAME);
        if (truststoreFilename == null || truststoreFilename.isEmpty()) throw new
IllegalArgumentException("Missing " + PROP_TRUSTSTORE_FILENAME);
        if (truststorePassword == null || truststorePassword.isEmpty()) throw new
IllegalArgumentException("Missing " + PROP_TRUSTSTORE_PASSWORD);

        hostnames = hostnameProperty.split(",");

        if (group_property != null && !group_property.isEmpty()) {
                        groups = group_property.split(",");
        }

        if (description != null) {
                 if (description.isEmpty()) description = null;
                 else description = description.trim();
        }

        sslContext = SSLContext.getInstance("TLSv1.2");
        sslContext.init(getKeyManagers(), getTrustManagers(), null);
    }

    public void setupAuth(HttpsURLConnection https) throws GeneralSecurityException, IOException {
                Authenticator.setDefault(new MyAuthenticator());
    }

    private class MyAuthenticator extends Authenticator {
        public PasswordAuthentication getPasswordAuthentication() {
            return (new PasswordAuthentication(username, password.toCharArray()));
        }
```

```
    }

        private KeyManager[] getKeyManagers() throws IOException, GeneralSecurityException {
                if (keystoreFilename == null || keystoreFilename.isEmpty())
                        return null;

                KeyStore ks = keystoreFilename.endsWith(".p12") ? KeyStore.getInstance("pkcs12") :
KeyStore.getInstance("JKS");
                FileInputStream in = new FileInputStream(keystoreFilename);
                ks.load(in, keystorePassword.toCharArray());
                in.close();
                KeyManagerFactory kmf =
KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
                kmf.init(ks, keystorePassword.toCharArray());
                KeyManager[] mngrs = kmf.getKeyManagers();

                if (mngrs == null || mngrs.length == 0) {
                        throw new GeneralSecurityException("no key managers found");
                }

                if (mngrs[0] instanceof X509KeyManager == false) {
                        throw new GeneralSecurityException("key manager is not for X509");
                }

                return new KeyManager[] { new SampleX509KeyManager((X509KeyManager) mngrs[0]) };
        }

        private TrustManager[] getTrustManagers() throws IOException, GeneralSecurityException {
                FileInputStream in = new FileInputStream(truststoreFilename);

                KeyStore ks = null;
                if(truststoreFilename.endsWith(".pem")) {
                        ks = KeyStore.getInstance("JKS");
                        ks.load(null, null);
                        CertificateFactory certFac = CertificateFactory.getInstance("X.509");
                        Collection<? extends Certificate> certs = certFac.generateCertificates(in);
                        int i = 0;
                        for(Certificate c : certs) {
                                ks.setCertificateEntry("trust-" + i, c);
                        }
                } else if(truststoreFilename.endsWith(".p12")) {
                        ks = KeyStore.getInstance("pkcs12");
                        ks.load(in, truststorePassword.toCharArray());
                } else {
                        ks = KeyStore.getInstance("JKS");
                        ks.load(in, truststorePassword.toCharArray());
                }

                in.close();

                Enumeration<String> e = ks.aliases();
                boolean hasCertEntries = false;

                while (e.hasMoreElements()) {
                        String alias = e.nextElement();

                        if (ks.isCertificateEntry(alias)) {
                                hasCertEntries = true;
                        }
                }

                if (hasCertEntries == false) {
                        e = ks.aliases();

                        while (e.hasMoreElements()) {
                                String alias = e.nextElement();

                                if (ks.isKeyEntry(alias)) {
                                        Certificate[] chain = ks.getCertificateChain(alias);

                                        for (int i = 0; i < chain.length; ++i) {
                                                ks.setCertificateEntry(alias + "." + i, chain[i]);
                                        }
```

```
                        }
                }
        }

        TrustManagerFactory tmf =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
        tmf.init(ks);

        TrustManager[] tms = tmf.getTrustManagers();

        if (tms == null || tms.length == 0) {
                throw new GeneralSecurityException("no trust managers found");
        }

        if (tms[0] instanceof X509TrustManager == false) {
                throw new GeneralSecurityException("trust manager is not for X509");
        }

        return new TrustManager[] { new SampleX509TrustManager((X509TrustManager) tms[0]) };
    }

    private static class SampleX509KeyManager implements X509KeyManager {

        private X509KeyManager mngr;

        public SampleX509KeyManager(X509KeyManager mngr) {
                this.mngr = mngr;
        }

        @Override
        public String chooseClientAlias(String[] arg0, Principal[] arg1, Socket arg2) {
                String alias = mngr.chooseClientAlias(arg0, arg1, arg2);

                if (alias == null) {
                        alias = mngr.chooseClientAlias(arg0, null, arg2);

                        if (alias == null) {
                                throw new RuntimeException("no client certificate found ...");
                        }
                }

                return alias;
        }

        @Override
        public String chooseServerAlias(String arg0, Principal[] arg1, Socket arg2) {
                throw new RuntimeException("Not implemented");
        }

        @Override
        public X509Certificate[] getCertificateChain(String arg0) {
                return mngr.getCertificateChain(arg0);
        }

        @Override
        public String[] getClientAliases(String arg0, Principal[] arg1) {
                return mngr.getClientAliases(arg0, null);
        }

        @Override
        public PrivateKey getPrivateKey(String arg0) {
                return mngr.getPrivateKey(arg0);
        }

        @Override
        public String[] getServerAliases(String arg0, Principal[] arg1) {
                throw new RuntimeException("Not implemented");
        }
    }

    private static class SampleX509TrustManager implements X509TrustManager {
        private X509TrustManager mngr;
```

```
                public SampleX509TrustManager(X509TrustManager mngr) {
                        this.mngr = mngr;
                }

                @Override
                public void checkClientTrusted(X509Certificate[] arg0, String arg1) throws
CertificateException {
                        throw new RuntimeException("not implemented");
                }

                @Override
                public void checkServerTrusted(X509Certificate[] arg0, String arg1) throws
CertificateException {
                        try {
                                mngr.checkServerTrusted(arg0, arg1);
                        } catch (CertificateException e) {
                                throw new CertificateException("Server certificate is not trusted:" +
arg0[0].getSubjectX500Principal(),
                                                e);
                        }
                }

                @Override
                public X509Certificate[] getAcceptedIssuers() {
                        return mngr.getAcceptedIssuers();
                }
        }

    private void print() {
        System.out.println("------- properties -------");
        System.out.print("  hostnames=");
        for (String hostname : hostnames) System.out.print(hostname + " ");
        System.out.println();
        System.out.println("  username=" + username);
        System.out.println("  password=" + password);
        System.out.print("  groups=");
        for (String group : groups) System.out.print(group + " ");
        System.out.println();
        System.out.println("  description=" + description);
        System.out.println("  keystoreFilename=" + keystoreFilename);
        System.out.println("  keystorePassword=" + keystorePassword);
        System.out.println("  truststoreFilename=" + truststoreFilename);
        System.out.println("  truststorePassword=" + truststorePassword);
        System.out.println("--------------------------");
    }
}
```

pxGrid Control

This code provides the pxGrid client with account creation on the ISE pxGrid node and service lookup request and access secret to the peer node, publishing the topic information. In the example, using API_Simulator, the peer node would reflect the pxGrid client.

```
package com.cisco.pxgrid.samples.ise.http;

import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.URL;
import java.util.Base64;
import java.util.Map;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLSession;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import com.cisco.pxgrid.model.AccessSecretRequest;
import com.cisco.pxgrid.model.AccessSecretResponse;
import com.cisco.pxgrid.model.AccountActivateRequest;
import com.cisco.pxgrid.model.AccountActivateResponse;
import com.cisco.pxgrid.model.AccountCreateRequest;
import com.cisco.pxgrid.model.AccountCreateResponse;
import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Authorization;
import com.cisco.pxgrid.model.AuthorizationRequest;
import com.cisco.pxgrid.model.AuthorizationResponse;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.model.ServiceLookupRequest;
import com.cisco.pxgrid.model.ServiceLookupResponse;
import com.cisco.pxgrid.model.ServiceRegisterRequest;
import com.cisco.pxgrid.model.ServiceRegisterResponse;
import com.google.gson.Gson;

/**
 * Using HTTPS for pxGrid control
 */
public class PxgridControl {
        private static Logger logger = LoggerFactory.getLogger(PxgridControl.class);
        private SampleConfiguration config;
    private String controllerVersion;

    public PxgridControl(SampleConfiguration config) {
        this.config = config;
        }

        private <T> T sendRequest(HttpsURLConnection https, Object request, Class<T> responseClass) throws
IOException {
                https.setRequestProperty("Content-Type", "application/json");
                https.setRequestProperty("Accept", "application/json");

                Gson gson = new Gson();

                OutputStreamWriter out = new OutputStreamWriter(https.getOutputStream());
                logger.info("Request={}", gson.toJson(request));
                gson.toJson(request, out);
                out.flush();

        InputStreamReader in = new InputStreamReader(https.getInputStream());
        T response = gson.fromJson(in, responseClass);
                logger.info("Response={}", gson.toJson(response));

        return response;
        }

    private HttpsURLConnection getHttpsURLConnection(String urlSuffix) throws IOException {
                String url = "https://" + config.getHostnames()[0] + ":8910/pxgrid/control/" + urlSuffix;
                URL conn = new URL(url);
                HttpsURLConnection https = (HttpsURLConnection) conn.openConnection();

                // SSL and Auth
                https.setSSLSocketFactory(config.getSSLContext().getSocketFactory());

                https.setRequestMethod("POST");

                String userPassword = config.getUserName() + ":" + config.getPassword();
                String encoded = Base64.getEncoder().encodeToString(userPassword.getBytes());
                https.setRequestProperty("Authorization", "Basic " + encoded);
                https.setHostnameVerifier(new HostnameVerifier() {
                        @Override
                        public boolean verify(String hostname, SSLSession session) {
                                return true;
                        }
                });
                https.setDoInput(true);
                https.setDoOutput(true);

                return https;
    }
```

```
    /**
     * Create new account
     *
     * @return password
     */
    public String accountCreate() throws IOException {
        HttpsURLConnection https = getHttpsURLConnection("AccountCreate");
                AccountCreateRequest request = new AccountCreateRequest();
                request.setNodeName(config.getUserName());
                AccountCreateResponse response = sendRequest(https, request, AccountCreateResponse.class);
                return response.getPassword();
    }

    public AccountState accountActivate() throws IOException {
                HttpsURLConnection https = getHttpsURLConnection("AccountActivate");
                AccountActivateRequest request = new AccountActivateRequest();
                request.setDescription(config.getDescription());
                AccountActivateResponse response = sendRequest(https, request, AccountActivateResponse.class);
                controllerVersion = response.getVersion();
                return response.getAccountState();
    }

        public void registerService(String name, Map<String, String> properties) throws IOException {
                HttpsURLConnection https = getHttpsURLConnection("ServiceRegister");
                ServiceRegisterRequest request = new ServiceRegisterRequest();
                request.setName(name);
                request.setProperties(properties);
                sendRequest(https, request, ServiceRegisterResponse.class);
        }

        public Service[] lookupService(String name) throws IOException {
                HttpsURLConnection https = getHttpsURLConnection("ServiceLookup");
                ServiceLookupRequest request = new ServiceLookupRequest();
                request.setName(name);
                ServiceLookupResponse response = sendRequest(https, request, ServiceLookupResponse.class);
                return response.getServices();
        }

        public String getAccessSecret(String peerNodeName) throws IOException  {
                HttpsURLConnection https = getHttpsURLConnection("AccessSecret");
                AccessSecretRequest request = new AccessSecretRequest();
                request.setPeerNodeName(peerNodeName);
                AccessSecretResponse response = sendRequest(https, request, AccessSecretResponse.class);
                return response.getSecret();
        }

        public boolean isAuthorized(String requestNodeName, String serviceName, String operation) throws
IOException {
                HttpsURLConnection https = getHttpsURLConnection("Authorization");
                AuthorizationRequest request = new AuthorizationRequest();
                request.setRequestNodeName(requestNodeName);
                request.setServiceName(serviceName);

                request.setServiceOperation(operation);

                AuthorizationResponse response = sendRequest(https, request, AuthorizationResponse.class);
                return (response.getAuthorization() == Authorization.PERMIT);
        }

        public String getControllerVersion() {
                return controllerVersion;
        }
}
```

SampleHelper

This code provides the initial WebSockets connection.

```
package com.cisco.pxgrid.samples.ise;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.time.OffsetDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Base64;
import java.util.Scanner;

import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLSocketFactory;

import org.apache.commons.io.IOUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.TypeAdapter;
import com.google.gson.stream.JsonReader;
import com.google.gson.stream.JsonToken;
import com.google.gson.stream.JsonWriter;

public class SampleHelper {
	private static Logger logger = LoggerFactory.getLogger(SampleHelper.class);

	public static HttpsURLConnection createHttpsURLConnection(String url, String user, String password,
			SSLSocketFactory sslSocketFactory) throws IOException {
		URL conn = new URL(url);
		HttpsURLConnection https = (HttpsURLConnection) conn.openConnection();
		https.setSSLSocketFactory(sslSocketFactory);
		String userPassword = user + ":" + password;
		String encoded = Base64.getEncoder().encodeToString(userPassword.getBytes());
		https.setRequestProperty("Authorization", "Basic " + encoded);
		return https;
	}

	public static String prompt(String msg) {
		System.out.println(msg);
		@SuppressWarnings("resource")
		Scanner scanner = new Scanner(System.in);
		String value = scanner.nextLine();
		if ("".equals(value))
			return null;
		return value;
	}

	public static OffsetDateTime promptDate(String msg) {
		String value = prompt(msg);
		if (value == null) return null;
		return OffsetDateTime.parse(value);
	}

	public static void postObjectAndPrint(String url, String user, String password, SSLSocketFactory ssl,
			Object postObject) throws IOException {
		Gson gson = new GsonBuilder().registerTypeAdapter(OffsetDateTime.class, new
OffsetDateTimeAdapter()).create();
		postStringAndPrint(url, user, password, ssl, gson.toJson(postObject));
	}

	public static void postStringAndPrint(String url, String user, String password, SSLSocketFactory ssl,
			String postData) throws IOException {
		logger.info("postData={}", postData);
		HttpsURLConnection httpsConn = SampleHelper.createHttpsURLConnection(url, user, password,
ssl);
		httpsConn.setRequestMethod("POST");
		httpsConn.setRequestProperty("Content-Type", "application/json");
		httpsConn.setRequestProperty("Accept", "application/json");
```

```
                httpsConn.setDoInput(true);
                httpsConn.setDoOutput(true);

                OutputStreamWriter osw = new OutputStreamWriter(httpsConn.getOutputStream());
                osw.write(postData);
                osw.flush();

                int status = httpsConn.getResponseCode();
                logger.info("Response status={}", status);

                if (status < HttpURLConnection.HTTP_BAD_REQUEST) {
                        try (InputStream in = httpsConn.getInputStream()) {
                                String content = IOUtils.toString(in, StandardCharsets.UTF_8);
                                System.out.println("Content: " + content);
                        }
                } else {
                        try (InputStream in = httpsConn.getErrorStream()) {
                                String content = IOUtils.toString(in, StandardCharsets.UTF_8);
                                System.out.println("Content: " + content);
                        }
                }
        }

        private static class OffsetDateTimeAdapter extends TypeAdapter<OffsetDateTime> {
                DateTimeFormatter formatter = DateTimeFormatter.ISO_OFFSET_DATE_TIME;

                @Override
                public void write(JsonWriter out, OffsetDateTime value) throws IOException {
                        if (value == null) {
                                out.nullValue();
                                return;
                        }
                        out.value(formatter.format(value));
                }

                @Override
                public OffsetDateTime read(JsonReader in) throws IOException {
                        if (in.peek() == JsonToken.NULL) {
                                in.nextNull();
                                return null;
                        }
                        return formatter.parse(in.nextString(), OffsetDateTime::from);
                }
        }
}
```

StompFrame

This code represents the parsing of STOMP frames.

```
package com.cisco.pxgrid.samples.ise;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.text.ParseException;
import java.util.HashMap;
import java.util.Map;

/**
 * This follows STOMP 1.2 specification to parse and generate STOMP frames:
 *   https://stomp.github.io/stomp-specification-1.2.html
 *
 * This single class is self-sufficient handle all STOMP frames.
 *
 * Note for WebSocket:
 * If input comes as WebSocket text type, (WS RFC says Text is UTF-8)
 * server side handling code like Spring TextMessage may convert the bytes to String as UTF-8
```

```
 * which maybe the wrong encoding as STOMP frame itself can use other encoding.
 *   e.g. A particular encoding may have bytes: FF FF FF FF FF FF FF FF FF FF... 10, that is completely out
of range for Unicode.
 * Unless STOMP body is also UTF-8, STOMP frame must be sent as binary
 *
 * @author Alan Lei
 */
public class StompFrame {
	public enum Command {
		CONNECT, STOMP, CONNECTED, SEND, SUBSCRIBE, UNSUBSCRIBE, ACK, NACK,
		BEGIN, COMMIT, ABORT, DISCONNECT, MESSAGE, RECEIPT, ERROR;

		private static Map<String, Command> mapOfStringToCommand = new HashMap<>();
		static {
			for (Command command : Command.values()) {
				mapOfStringToCommand.put(command.name(), command);
			}
		}

		public static Command get(String value) {
			return mapOfStringToCommand.get(value);
		}
	}

	private Command command;
	private Map<String, String> headers = new HashMap<>();
	private byte[] content;
	private final static int MAX_BUFFER_SIZE = 1024;

	public Command getCommand() {
		return command;
	}

	public void setCommand(Command command) {
		this.command = command;
	}

	public String getHeader(String name) {
		return headers.get(name);
	}

	public void setHeader(String name, String value) {
		headers.put(name, value);
	}

	public Map<String, String> getHeaders() {
		return headers;
	}

	public byte[] getContent() {
		return content;
	}

	public void setContent(byte[] content) {
		this.content = content;
	}

	public void write(OutputStream out) throws IOException {
		out.write(command.name().getBytes());
		out.write('\n');
		for (String name : headers.keySet()) {
			out.write(name.getBytes());
			out.write(':');
			out.write(headers.get(name).getBytes());
			out.write('\n');
		}
		out.write('\n');
		if (content != null) {
			out.write(content);
		}
		out.write(0);
	}
```

```
	private static String readLine(InputStream in) throws IOException, ParseException {
		byte[] line = new byte[MAX_BUFFER_SIZE];
		int index = 0;
		while (index < MAX_BUFFER_SIZE) {
			int b = in.read();
			if (b != -1) {
				if (b == '\n') {
					return new String(line, 0, index);
				}
				if (b != '\r') {
					line[index] = (byte)b;
					index++;
				}
			}
			else {
				// No line found
				return null;
			}
		}
		throw new ParseException("Line too long", MAX_BUFFER_SIZE);
	}

	/*
	 * Using InputStream instead of Reader because
	 * content-length is octet count instead of character count
	 */
	public static StompFrame parse(InputStream reader) throws IOException, ParseException {
		StompFrame stomp = new StompFrame();

		// Read Command
		String line = readLine(reader);
		Command command = Command.get(line);
		if (command == null) throw new ParseException("Unknown command: " + line, 0);
		stomp.setCommand(command);

		// Read Headers
		int contentLength = -1;
		while ((line = readLine(reader)) != null) {
			if (line.equals("")) break;
			int colon = line.indexOf(':');
			String name = line.substring(0, colon);
			String value = line.substring(colon + 1);
			stomp.setHeader(name, value);
			if (name.equals("content-length")) {
				contentLength = Integer.parseInt(value);
			}
		}

		// Read Content
		if (contentLength != -1) {
			// content-length is in octets
			byte[] content = new byte[contentLength];
			reader.read(content);
			stomp.setContent(content);
			if (reader.read() != 0) {
				throw new ParseException("Byte after STOMP Body not NULL", -1);
			}
		}
		else {
			// No content-length. Look for ending NULL byte.
			byte[] buffer = new byte[MAX_BUFFER_SIZE];
			int length = 0;
			while (length < MAX_BUFFER_SIZE) {
				int b = reader.read();
				if (b == -1) {
					throw new ParseException("Premature end of stream", -1);
				}
				if (b == 0) {
					if (length > 0) {
						byte[] content = new byte[length];
						System.arraycopy(buffer, 0, content, 0, length);
						stomp.setContent(content);
					}
```

```
                                        // More EOLs may follow, but ignored.
                                        return stomp;
                                }
                                buffer[length] = (byte)b;
                                length++;
                        }
                        throw new ParseException("Frame too long", -1);
                }
                return stomp;
        }

        @Override
        public String toString() {
                StringBuilder sb = new StringBuilder();
                sb.append("command=" + command);
                sb.append(", headers={");
                for (String name : headers.keySet()) {
                        sb.append("'" + name + "':");
                        sb.append("'" + headers.get(name) + "',");
                }
                sb.append("}");
                sb.append(", content.length=" + content.length);
                return sb.toString();
        }
}
```

StompSubscription

This code provides service or topic subscription using the STOMP messaging protocol

```
package com.cisco.pxgrid.samples.ise;

import java.util.concurrent.atomic.AtomicInteger;

public class StompSubscription {
        public static interface Handler {
                void handle(StompFrame message);
        }

        private static AtomicInteger currentSubscriptionID = new AtomicInteger();
        private String id = Integer.toString(currentSubscriptionID.getAndIncrement());
        private String topic;
        private Handler handler;

        public StompSubscription(String topic, Handler handler) {
                this.topic = topic;
                this.handler = handler;
        }

        public String getId() {
                return id;
        }

        public String getTopic() {
                return topic;
        }

        public Handler getHandler() {
                return handler;
        }

        public StompFrame getSubscribeMessage() {
                StompFrame message = new StompFrame();
                message.setCommand(StompFrame.Command.SUBSCRIBE);
                message.setHeader("destination", topic);
                message.setHeader("id", id);
                return message;
        }
```

```
}
```

StompPubSubClientEndpoint

This code provides service or topic subscription using the STOMP messaging protocol for the client endpoint or asset device.

```
package com.cisco.pxgrid.samples.ise;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.text.ParseException;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import javax.websocket.ClientEndpoint;
import javax.websocket.CloseReason;
import javax.websocket.Endpoint;
import javax.websocket.EndpointConfig;
import javax.websocket.MessageHandler;
import javax.websocket.Session;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.cisco.pxgrid.samples.ise.StompSubscription.Handler;

@ClientEndpoint
public class StompPubsubClientEndpoint extends Endpoint {
        private static Logger logger = LoggerFactory.getLogger(StompPubsubClientEndpoint.class);

        private volatile Session session;
        private Map<String, StompSubscription> mapOfIdToSubscription = new ConcurrentHashMap<>();

        public void connect(String hostname) throws IOException {
                logger.info("STOMP CONNECT host={}", hostname);
                StompFrame message = new StompFrame();
                message.setCommand(StompFrame.Command.CONNECT);
                message.setHeader("accept-version", "1.2");
                message.setHeader("host", hostname);
                send(message);
        }

        public void disconnect(String receipt) throws IOException {
                logger.info("STOMP DISCONNECT receipt={}", receipt);
                StompFrame message = new StompFrame();
                message.setCommand(StompFrame.Command.DISCONNECT);
                if (receipt != null) {
                        message.setHeader("receipt", receipt);
                }
                send(message);
        }

        public void subscribe(StompSubscription subscription) throws IOException {
                logger.info("STOMP SUBSCRIBE topic={}", subscription.getTopic());
                mapOfIdToSubscription.put(subscription.getId(), subscription);
                if (session != null) {
                        StompFrame message = subscription.getSubscribeMessage();
                        send(message);
                }
        }

        public void publish(String topic, byte[] content) throws IOException {
                logger.info("STOMP SEND topic={}", topic);
                StompFrame message = new StompFrame();
```

```
                message.setCommand(StompFrame.Command.SEND);
                message.setHeader("destination", topic);
                message.setHeader("content-length", Integer.toString(content.length));
                message.setContent(content);
                send(message);
        }

        private void send(StompFrame message) throws IOException {
                if (session != null) {
                        ByteArrayOutputStream baos = new ByteArrayOutputStream();
                        message.write(baos);
                        // Send as binary
                        session.getBasicRemote().sendBinary(ByteBuffer.wrap(baos.toByteArray()));
                }
        }

        public void waitForOpen() throws InterruptedException {
                synchronized (this) {
                        while (session == null) {
                                this.wait();
                        }
                }
        }

        private void onStompMessage(StompFrame stomp) {
                switch (stomp.getCommand()) {
                case CONNECTED:
                        String version = stomp.getHeader("version");
                        logger.info("STOMP CONNECTED version={}", version);
                        break;
                case RECEIPT:
                        String receiptId = stomp.getHeader("receipt-id");
                        logger.info("STOMP RECEIPT id={}", receiptId);
                        break;
                case MESSAGE:
                        String id = stomp.getHeader("subscription");
                        StompSubscription subscription = mapOfIdToSubscription.get(id);
                        Handler handler = subscription.getHandler();
                        if (handler != null) {
                                handler.handle(stomp);
                        }
                        break;
                case ERROR:
                        // Server will close connect on ERROR according to STOMP specification
                        logger.info("STOMP ERROR stomp={}", stomp);
                        break;
                default:
                        // Ignore others
                        break;
                }
        }

        private class TextHandler implements MessageHandler.Whole<String> {
                @Override
                public void onMessage(String message) {
                        try {
                                StompFrame stomp = StompFrame.parse(new
ByteArrayInputStream(message.getBytes()));
                                onStompMessage(stomp);
                        } catch (IOException | ParseException e) {
                                logger.error("onMessage", e);
                        }
                }
        }

        private class BinaryHandler implements MessageHandler.Whole<InputStream> {
                @Override
                public void onMessage(InputStream in) {
                        try {
                                StompFrame stomp = StompFrame.parse(in);
                                onStompMessage(stomp);
                        } catch (IOException | ParseException e) {
                                logger.error("onMessage", e);
```

```
                        }
                }
        }

        @Override
        public void onOpen(Session session, EndpointConfig cfg) {
                logger.info("WS onOpen");
                this.session = session;
                try {
                        session.addMessageHandler(new TextHandler());
                        session.addMessageHandler(new BinaryHandler());

                        for (StompSubscription subscription : mapOfIdToSubscription.values()) {
                                StompFrame message = subscription.getSubscribeMessage();
                                send(message);
                        }
                } catch (IOException e) {
                        logger.error("onOpen", e);
                }
                synchronized (this) {
                        this.notifyAll();
                }
        }

        @Override
        public void onClose(Session session, CloseReason closeReason) {
                logger.info("WS onClose closeReason code={} phrase={}", closeReason.getCloseCode(),
                                closeReason.getReasonPhrase());
                this.session = null;
        }

        @Override
        public void onError(Session session, Throwable thr) {
                logger.info("WS onError thr={}", thr.getMessage());
                this.session = null;
        }
}
```

SessionQueryAll

This code returns all available sessions

```
package com.cisco.pxgrid.samples.ise;

import java.time.OffsetDateTime;

import org.apache.commons.cli.ParseException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.cisco.pxgrid.samples.ise.model.AccountState;
import com.cisco.pxgrid.samples.ise.model.Service;

/**
 * Demonstrates how to query all sessions from ISE Session Directory service
 */
public class SessionQueryAll {
        private static Logger logger = LoggerFactory.getLogger(SessionQueryAll.class);

        private static class SessionQueryRequest {
                OffsetDateTime startTimestamp;
        }

        private static void downloadUsingAccessSecret(SampleConfiguration config) throws Exception {
                OffsetDateTime startTimestamp = SampleHelper.promptDate("Enter start time (ex. '2015-01-
31T13:00:00-07:00' or <enter> for no start time): ");

                PxgridControl https = new PxgridControl(config);

                // pxGrid ServiceLookup for session service
                Service[] services = https.serviceLookup("com.cisco.ise.session");
                if (services == null || services.length == 0) {
                        logger.warn("Service unavailabe");
                        return;
                }

                // Use first service
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/getSessions";
                logger.info("url={}", url);

                // pxGrid AccesssSecret for the node
                String secret = https.getAccessSecret(service.getNodeName());

                SessionQueryRequest request = new SessionQueryRequest();
                request.startTimestamp = startTimestamp;
                SampleHelper.postObjectAndPrint(url, config.getNodeName(), secret,
config.getSSLContext().getSocketFactory(), request);
        }

        public static void main(String [] args) throws Exception {
                // Parse arguments
                SampleConfiguration config = new SampleConfiguration();
                try {
                        config.parse(args);
                } catch (ParseException e) {
                        config.printHelp("SessionQueryAll");
                        System.exit(1);
                }

                // AccountActivate
                PxgridControl control = new PxgridControl(config);
                while (control.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                logger.info("pxGrid controller version={}", control.getControllerVersion());

                downloadUsingAccessSecret(config);
```

```
        }
}
```

SessionQueryByIP

This code returns available session attributes for a queried IP Address

```
package com.cisco.pxgrid.samples.ise;

import java.io.IOException;

import org.apache.commons.cli.ParseException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.cisco.pxgrid.samples.ise.model.AccountState;
import com.cisco.pxgrid.samples.ise.model.Service;

/**
 * Demonstrates how to query session by IP from ISE Session Directory service
 */
public class SessionQueryByIP {
        private static Logger logger = LoggerFactory.getLogger(SessionQueryByIP.class);

        private static void query(SampleConfiguration config, String ip) throws IOException {
                PxgridControl pxgrid = new PxgridControl(config);

                // pxGrid ServiceLookup for session service
                Service[] services = pxgrid.serviceLookup("com.cisco.ise.session");
                if (services == null || services.length == 0) {
                        System.out.println("Service unavailabe");
                        return;
                }

                // Use first service
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/getSessionByIpAddress";
                logger.info("url={}", url);

                // pxGrid AccessSecret for the node
                String secret = pxgrid.getAccessSecret(service.getNodeName());

                String postData = "{\"ipAddress\":\"" + ip + "\"}";
                SampleHelper.postStringAndPrint(url, config.getNodeName(), secret,
config.getSSLContext().getSocketFactory(), postData);
        }

        public static void main(String [] args) throws Exception {
                // Parse arguments
                SampleConfiguration config = new SampleConfiguration();
                try {
                        config.parse(args);
                } catch (ParseException e) {
                        config.printHelp("SessionQueryByIP");
                        System.exit(1);
                }

                // AccountActivate
                PxgridControl pxgrid = new PxgridControl(config);
                while (pxgrid.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                logger.info("pxGrid controller version={}", pxgrid.getControllerVersion());

                while (true) {
                        String ip = SampleHelper.prompt("IP address (or <enter> to disconnect): ");
                        if (ip == null)         break;
                        query(config, ip);
                }
```

```
        }
}
```

CustomServiceProvider

This code provides the pxGrid client to a publish a topic and the CustomServiceConsumer code will subscribe to this topic or service.

```
package com.cisco.pxgrid.samples.ise;

import java.io.IOException;
import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

import javax.websocket.Session;

import org.apache.commons.cli.ParseException;
import org.glassfish.tyrus.client.ClientManager;
import org.glassfish.tyrus.client.ClientProperties;
import org.glassfish.tyrus.client.SslEngineConfigurator;
import org.glassfish.tyrus.client.auth.Credentials;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.cisco.pxgrid.samples.ise.model.AccountState;
import com.cisco.pxgrid.samples.ise.model.Service;
import com.cisco.pxgrid.samples.ise.model.ServiceRegisterResponse;

/**
 * Demonstrate how to create a custom service that publishes data
 *
 * The flow of the application is as follows:
 * 1. Parse arguments for configurations
 * 2. Activate Account. This will then require ISE Admin to approve this new node.
 * 3. pxGrid ServiceRegister to register the new custom service
 * 4. Schedule periodic pxGrid ServiceReregister to signify the service is still alive
 * 5. pxGrid ServiceLookup for ISE pubsub service
 * 6. pxGrid get AccessSecret for the ISE pubsub node
 * 7. Establish WebSocket connection with the ISE pubsub node
 * 8. Establish STOMP connection for pubsub messaging
 * 9. Schedule periodic publish of data
 * 10. Wait for keyboard input for stopping the application
 */
public class CustomServiceProvider {
        private static Logger logger = LoggerFactory.getLogger(CustomServiceProvider.class);

        public static void main(String[] args) throws Exception {
                ScheduledExecutorService executor = Executors.newSingleThreadScheduledExecutor();

                // Parse arguments
                SampleConfiguration config = new SampleConfiguration();
                try {
                        config.parse(args);
                } catch (ParseException e) {
                        config.printHelp("CustomServiceProvider");
                        System.exit(1);
                }

                // AccountActivate
                PxgridControl control = new PxgridControl(config);
                while (control.accountActivate() != AccountState.ENABLED) {
                        Thread.sleep(60000);
                }
```

```
                logger.info("pxGrid controller version={}", control.getControllerVersion());

                // pxGrid ServiceRegister
                Map<String, String> sessionProperties = new HashMap<>();
                sessionProperties.put("wsPubsubService", "com.cisco.ise.pubsub");
                sessionProperties.put("customTopic", "/topic/com.example.custom");
                ServiceRegisterResponse response = control.serviceRegister("com.example.custom",
sessionProperties);
                String registrationId = response.getId();
                long reregisterTimeMillis = response.getReregisterTimeMillis();

                // Schedule pxGrid ServiceReregister
                executor.scheduleWithFixedDelay(() -> {
                        try {
                                control.serviceReregister(registrationId);
                        } catch (IOException e) {
                                logger.error("Reregister failure");
                        }
                }, reregisterTimeMillis, reregisterTimeMillis, TimeUnit.MILLISECONDS);

                // pxGrid ServiceLookup for pubsub service
                Service[] services = control.serviceLookup("com.cisco.ise.pubsub");
                if (services.length == 0) {
                        logger.info("Pubsub service unavailabe");
                        return;
                }
                // Use first service
                Service wsPubsubService = services[0];
                String wsURL = wsPubsubService.getProperties().get("wsUrl");
                logger.info("wsUrl={}", wsURL);

                // pxGrid AccessSecret
                String secret = control.getAccessSecret(wsPubsubService.getNodeName());

                // Setup WebSocket client
                ClientManager client = ClientManager.createClient();
                SslEngineConfigurator sslEngineConfigurator = new
SslEngineConfigurator(config.getSSLContext());
                client.getProperties().put(ClientProperties.SSL_ENGINE_CONFIGURATOR, sslEngineConfigurator);
                client.getProperties().put(ClientProperties.CREDENTIALS,
                                new Credentials(config.getNodeName(), secret.getBytes()));

                // WebSocket connect
                StompPubsubClientEndpoint endpoint = new StompPubsubClientEndpoint();
                URI uri = new URI(wsURL);
                Session session = client.connectToServer(endpoint, uri);

                // STOMP connect
                endpoint.connect(uri.getHost());

                // STOMP send periodically
                executor.scheduleWithFixedDelay(() -> {
                        try {
                                endpoint.publish("/topic/com.example.custom", "custom data".getBytes());
                        } catch (IOException e) {
                                logger.error("Publish failure");
                        }
                }, 0, 5, TimeUnit.SECONDS);

                SampleHelper.prompt("press <enter> to disconnect...");

                // pxGrid ServerUnregister
                control.unregisterService(registrationId);

                // Stop executor
                executor.shutdown();
                executor.awaitTermination(5, TimeUnit.SECONDS);

                // STOMP disconnect
                endpoint.disconnect("ID-123");
                // Wait for disconnect receipt
                Thread.sleep(3000);
```

```
                // Websocket close
                session.close();
        }
}
```

CustomServiceConsumer

This sample code provides consumer or pxGrid client subscription to a customer service or published topic

```
package com.cisco.pxgrid.samples.ise;

import java.net.URI;

import javax.net.ssl.SSLSession;
import javax.websocket.Session;

import org.apache.commons.cli.ParseException;
import org.glassfish.tyrus.client.ClientManager;
import org.glassfish.tyrus.client.ClientProperties;
import org.glassfish.tyrus.client.SslEngineConfigurator;
import org.glassfish.tyrus.client.auth.Credentials;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.cisco.pxgrid.samples.ise.model.AccountState;
import com.cisco.pxgrid.samples.ise.model.Service;

/**
 * Demonstrates how to subscribe a topic from a custom service
 *
 * The flow of the application is as follows:
 * 1. Parse arguments for configurations
 * 2. Activate Account. This will then require ISE Admin to approve this new node.
 * 3. pxGrid ServiceLookup for the custom service
 * 4. pxGrid ServiceLookup for ISE pubsub service
 * 5. pxGrid get AccessSecret for the ISE pubsub node
 * 6. Establish WebSocket connection with the ISE pubsub node
 * 7. Establish STOMP connection for pubsub messaging
 * 8. Subscribe to the topic in the custom service
 * 9. Wait for keyboard input for stopping the application
 */
public class CustomServiceConsumer {
        private static Logger logger = LoggerFactory.getLogger(CustomServiceProvider.class);

        // Subscribe handler class
        private static class MessageHandler implements StompSubscription.Handler {
                @Override
                public void handle(StompFrame message) {
                        System.out.println(new String(message.getContent()));
                }
        }

        public static void main(String [] args) throws Exception {
                // Parse arguments
                SampleConfiguration config = new SampleConfiguration();
                try {
                        config.parse(args);
                } catch (ParseException e) {
                        config.printHelp("CustomServiceConsumer");
                        System.exit(1);
                }

                // AccountActivate
                PxgridControl control = new PxgridControl(config);
                while (control.accountActivate() != AccountState.ENABLED) {
                        Thread.sleep(60000);
                }
                logger.info("pxGrid controller version={}", control.getControllerVersion());
```

```
                // pxGrid ServiceLookup for custom service
                Service[] services = control.serviceLookup("com.example.custom");
                if (services.length == 0) {
                        logger.info("Service unavailabe");
                        return;
                }

                // Use first service. Note that ServiceLookup randomize ordering of services
                Service customService = services[0];
                String wsPubsubServiceName = customService.getProperties().get("wsPubsubService");
                String customTopic = customService.getProperties().get("customTopic");
                logger.info("wsPubsubServiceName={} sessionTopic={}", wsPubsubServiceName, customTopic);

                // pxGrid ServiceLookup for pubsub service
                services = control.serviceLookup(wsPubsubServiceName);
                if (services.length == 0) {
                        logger.info("Pubsub service unavailabe");
                        return;
                }

                // Use first service
                Service wsPubsubService = services[0];
                String wsURL = wsPubsubService.getProperties().get("wsUrl");
                logger.info("wsUrl={}", wsURL);

                // pxGrid AccessSecret for the pubsub node
                String secret = control.getAccessSecret(wsPubsubService.getNodeName());

                // WebSocket config
                ClientManager client = ClientManager.createClient();
                SslEngineConfigurator sslEngineConfigurator = new
SslEngineConfigurator(config.getSSLContext());
                client.getProperties().put(ClientProperties.SSL_ENGINE_CONFIGURATOR, sslEngineConfigurator);
                client.getProperties().put(ClientProperties.CREDENTIALS,
                                new Credentials(config.getNodeName(), secret.getBytes()));

                // WebSocket connect
                StompPubsubClientEndpoint endpoint = new StompPubsubClientEndpoint();
                URI uri = new URI(wsURL);
                Session session = client.connectToServer(endpoint, uri);

                // STOMP connect
                endpoint.connect(uri.getHost());

                // Subscribe
                StompSubscription subscription = new StompSubscription(customTopic, new MessageHandler());
                endpoint.subscribe(subscription);

                SampleHelper.prompt("press <enter> to disconnect...");

                // STOMP disconnect
                endpoint.disconnect("ID-123");
                // Wait for disconnect receipt
                Thread.sleep(3000);

                session.close();
        }
}
```

Adaptive Network Control (ANC) Examples

The following coding examples represent the ANC operation that can be taken by the pxGrid client. The complete ANC configuration topic can be found: https://github.com/cisco-pxgrid/pxgrid-rest-ws/wiki/ANC-configuration

ANCSubscribe

The consumer or pxGrid client subscribes to the /topic/com.cisco.ise.config.anc.status topic or service

```
package com.cisco.pxgrid.samples.ise.anc;

import java.net.URI;

import org.glassfish.grizzly.ssl.SSLEngineConfigurator;
import org.glassfish.tyrus.client.ClientManager;
import org.glassfish.tyrus.client.ClientProperties;
import org.glassfish.tyrus.client.auth.Credentials;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.samples.ise.http.Console;
import com.cisco.pxgrid.samples.ise.http.PxgridControl;
import com.cisco.pxgrid.samples.ise.http.SampleConfiguration;
import com.cisco.pxgrid.samples.ise.http.StompFrame;
import com.cisco.pxgrid.samples.ise.http.StompPubsubClientEndpoint;
import com.cisco.pxgrid.samples.ise.http.StompSubscription;

/**
 * Demonstrates how to subscribe using REST/WS
 */
public class AncSubscribe {
        // Subscribe handler class
        private static class SubscriptionHandler implements StompSubscription.Handler {
                @Override
                public void handle(StompFrame message) {
                        System.out.println(new String(message.getContent()));
                }
        }

        public static void main(String [] args) throws Exception {
                // Read environment for config
                SampleConfiguration config = new SampleConfiguration();

                PxgridControl control = new PxgridControl(config);

                // AccountActivate
                while (control.accountActivate() != AccountState.ENABLED) {
                        Thread.sleep(60000);
                }
                Console.log("pxGrid controller version=" + control.getControllerVersion());

                // Session ServiceLookup
                Console.log("Looking up service com.cisco.ise.config.anc");
                Service[] services = control.lookupService("com.cisco.ise.config.anc");
                if (services.length == 0) {
                        Console.log("Session service unavailabe");
                        return;
                }

                Service sessionService = services[0];
                String wsPubsubServiceName = sessionService.getProperties().get("wsPubsubService");
                String statusTopic = sessionService.getProperties().get("statusTopic");
                Console.log("wsPubsubServiceName=" + wsPubsubServiceName + " statusTopic=" + statusTopic);
                // Pubsub ServiceLookup
                services = control.lookupService(wsPubsubServiceName);
                if (services.length == 0) {
```

```
                        Console.log("Pubsub service unavailabe");
                        return;
                }

                // Select first one for sample purpose. Should cycle through until connects.
                Service wsPubsubService = services[0];
                String wsURL = wsPubsubService.getProperties().get("wsUrl");
                Console.log("url=" + wsURL);

                // pxGrid AccessSecret
                String secret = control.getAccessSecret(wsPubsubService.getNodeName());

                // WebSocket config
                ClientManager client = ClientManager.createClient();
                SSLEngineConfigurator sslEngineConfigurator = new
SSLEngineConfigurator(config.getSSLContext());
                client.getProperties().put(ClientProperties.SSL_ENGINE_CONFIGURATOR, sslEngineConfigurator);
                client.getProperties().put(ClientProperties.CREDENTIALS,
                                new Credentials(config.getUserName(), secret.getBytes()));

                // WebSocket connect
                StompPubsubClientEndpoint endpoint = new StompPubsubClientEndpoint();
                URI uri = new URI(wsURL);
                javax.websocket.Session session = client.connectToServer(endpoint, uri);

                // STOMP connect
                endpoint.connect(uri.getHost());

                // Subscribe
                StompSubscription subscription = new StompSubscription(statusTopic, new
SubscriptionHandler());
                endpoint.subscribe(subscription);

                Console.log("press <enter> to disconnect...");
                System.in.read();

                // STOMP disconnect
                endpoint.disconnect("ID-123");
                // Wait for disconnect receipt
                Thread.sleep(3000);

                session.close();
        }
}
```

ANCGetPolicies

This code retrieves all ISE ANC Policies

```
package com.cisco.pxgrid.samples.ise.anc;

import java.io.IOException;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.samples.ise.http.Console;
import com.cisco.pxgrid.samples.ise.http.PxgridControl;
import com.cisco.pxgrid.samples.ise.http.SampleConfiguration;
import com.cisco.pxgrid.samples.ise.http.SampleHelper;

/**
 * Demonstrates how to query a session using IP address
 */
public class AncGetPolicies {

        private static void get(SampleConfiguration config) throws IOException {
                PxgridControl pxgrid = new PxgridControl(config);
                Service[] services = pxgrid.lookupService("com.cisco.ise.config.anc");
```

```
                if (services == null || services.length == 0) {
                        System.out.println("Service unavailabe");
                        return;
                }
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/getPolicies";
                String secret = pxgrid.getAccessSecret(service.getNodeName());
                SampleHelper.postStringAndPrint(url, config.getUserName(), secret,
config.getSSLContext().getSocketFactory(), "{}");
        }

        public static void main(String [] args) throws Exception {
                SampleConfiguration config = new SampleConfiguration();
                PxgridControl pxgrid = new PxgridControl(config);

                while (pxgrid.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                Console.log("pxGrid controller version=" + pxgrid.getControllerVersion());

                get(config);
        }
}
```

ANCGetPoliciesByName

This code retrieves the ISE ANC policies by policy name

```
package com.cisco.pxgrid.samples.ise.anc;

import java.io.IOException;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.samples.ise.http.Console;
import com.cisco.pxgrid.samples.ise.http.PxgridControl;
import com.cisco.pxgrid.samples.ise.http.SampleConfiguration;
import com.cisco.pxgrid.samples.ise.http.SampleHelper;

/**
 * Demonstrates how to query a session using IP address
 */
public class AncGetPolicyByName {
        private static class ByNameRequest {
                private String name;
                public String getName() {
                        return name;
                }
                public void setName(String name) {
                        this.name = name;
                }
        }
        private static void get(SampleConfiguration config, String name) throws IOException {
                PxgridControl pxgrid = new PxgridControl(config);
                Service[] services = pxgrid.lookupService("com.cisco.ise.config.anc");
                if (services == null || services.length == 0) {
                        System.out.println("Service unavailabe");
                        return;
                }
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/getPolicyByName";
                String secret = pxgrid.getAccessSecret(service.getNodeName());
                ByNameRequest request = new ByNameRequest();
                request.setName(name);
                SampleHelper.postObjectAndPrint(url, config.getUserName(), secret,
config.getSSLContext().getSocketFactory(), request);
        }

        public static void main(String [] args) throws Exception {
```

```
                SampleConfiguration config = new SampleConfiguration();
                PxgridControl pxgrid = new PxgridControl(config);

                while (pxgrid.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                Console.log("pxGrid controller version=" + pxgrid.getControllerVersion());

                String name = SampleHelper.prompt("Get policy name: ");
                get(config, name);
        }
}
```

ANCGetEndpoints

This code retrieves endpoints with the ANC policy applied

```
package com.cisco.pxgrid.samples.ise.anc;

import java.io.IOException;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.samples.ise.http.Console;
import com.cisco.pxgrid.samples.ise.http.PxgridControl;
import com.cisco.pxgrid.samples.ise.http.SampleConfiguration;
import com.cisco.pxgrid.samples.ise.http.SampleHelper;

/**
 * Demonstrates how to query a session using IP address
 */
public class AncGetEndpoints {

        private static void getEndpoints(SampleConfiguration config) throws IOException {
                PxgridControl pxgrid = new PxgridControl(config);
                Service[] services = pxgrid.lookupService("com.cisco.ise.config.anc");
                if (services == null || services.length == 0) {
                        System.out.println("Service unavailabe");
                        return;
                }
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/getEndpoints";
                String secret = pxgrid.getAccessSecret(service.getNodeName());
                SampleHelper.postStringAndPrint(url, config.getUserName(), secret,
config.getSSLContext().getSocketFactory(), "{}");
        }

        public static void main(String [] args) throws Exception {
                SampleConfiguration config = new SampleConfiguration();
                PxgridControl pxgrid = new PxgridControl(config);

                while (pxgrid.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                Console.log("pxGrid controller version=" + pxgrid.getControllerVersion());

                getEndpoints(config);
        }
}
```

ANCGetEndpointsByMAC

This code retrieves endpoints with an ISE ANC policy by MAC address

```
package com.cisco.pxgrid.samples.ise.anc;

import java.io.IOException;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.samples.ise.http.Console;
import com.cisco.pxgrid.samples.ise.http.PxgridControl;
import com.cisco.pxgrid.samples.ise.http.SampleConfiguration;
import com.cisco.pxgrid.samples.ise.http.SampleHelper;

/**
 * Demonstrates how to query a session using IP address
 */
public class AncGetEndpointByMac {
        private static class ByMacRequest {
                private String mac;
                public String getMac() {
                        return mac;
                }
                public void setMac(String mac) {
                        this.mac = mac;
                }
        }
        private static void get(SampleConfiguration config, String mac) throws IOException {
                PxgridControl pxgrid = new PxgridControl(config);
                Service[] services = pxgrid.lookupService("com.cisco.ise.config.anc");
                if (services == null || services.length == 0) {
                        System.out.println("Service unavailabe");
                        return;
                }
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/getEndpointByMacAddress";
                String secret = pxgrid.getAccessSecret(service.getNodeName());
                ByMacRequest request = new ByMacRequest();
                request.setMac(mac);
                SampleHelper.postObjectAndPrint(url, config.getUserName(), secret,
config.getSSLContext().getSocketFactory(), request);
        }

        public static void main(String [] args) throws Exception {
                SampleConfiguration config = new SampleConfiguration();
                PxgridControl pxgrid = new PxgridControl(config);

                while (pxgrid.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                Console.log("pxGrid controller version=" + pxgrid.getControllerVersion());

                String mac = SampleHelper.prompt("Get endpoint by mac: ");
                get(config, mac);
        }
}
```

ANCApplyByIP

This code retrieves endpoints with an ANCY policy by IP Address

```
package com.cisco.pxgrid.samples.ise.anc;

import java.io.IOException;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.samples.ise.http.Console;
import com.cisco.pxgrid.samples.ise.http.PxgridControl;
import com.cisco.pxgrid.samples.ise.http.SampleConfiguration;
import com.cisco.pxgrid.samples.ise.http.SampleHelper;

/**
 * Demonstrates how to query a session using IP address
 */
public class AncApplyByIp {
        private static class ApplyEndpointPolicyByIpRequest {
                private String policyName;
                private String ipAddress;
                public void setPolicyName(String policyName) {
                        this.policyName = policyName;
                }
                public void setIpAddress(String ipAddress) {
                        this.ipAddress = ipAddress;
                }

        }

        private static void apply(SampleConfiguration config, String policyName, String ip) throws
IOException {
                PxgridControl pxgrid = new PxgridControl(config);
                Service[] services = pxgrid.lookupService("com.cisco.ise.config.anc");
                if (services == null || services.length == 0) {
                        System.out.println("Service unavailabe");
                        return;
                }
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/applyEndpointByIpAddress";
                String secret = pxgrid.getAccessSecret(service.getNodeName());
                ApplyEndpointPolicyByIpRequest request = new ApplyEndpointPolicyByIpRequest();
                request.setPolicyName(policyName);
                request.setIpAddress(ip);
                SampleHelper.postObjectAndPrint(url, config.getUserName(), secret,
config.getSSLContext().getSocketFactory(), request);
        }


        public static void main(String [] args) throws Exception {
                SampleConfiguration config = new SampleConfiguration();
                PxgridControl pxgrid = new PxgridControl(config);

                while (pxgrid.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                Console.log("pxGrid controller version=" + pxgrid.getControllerVersion());

                String policyName = SampleHelper.prompt("Policy name: ");
                String ip = SampleHelper.prompt("IP address: ");
                apply(config, policyName, ip);
        }
}
```

ANCClearByIP

This code clears or unquarantines endpoints with an ISE ANC policy by IP Address

```
package com.cisco.pxgrid.samples.ise.anc;

import java.io.IOException;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.samples.ise.http.Console;
import com.cisco.pxgrid.samples.ise.http.PxgridControl;
import com.cisco.pxgrid.samples.ise.http.SampleConfiguration;
import com.cisco.pxgrid.samples.ise.http.SampleHelper;

/**
 * Demonstrates how to query a session using IP address
 */
public class AncClearByIp {
        private static class EndpointPolicyByIpRequest {
                private String ipAddress;
                public void setIpAddress(String ipAddress) {
                        this.ipAddress = ipAddress;
                }

        }

        private static void clear(SampleConfiguration config, String ip) throws IOException {
                PxgridControl pxgrid = new PxgridControl(config);
                Service[] services = pxgrid.lookupService("com.cisco.ise.config.anc");
                if (services == null || services.length == 0) {
                        System.out.println("Service unavailabe");
                        return;
                }
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/clearEndpointByIpAddress";
                String secret = pxgrid.getAccessSecret(service.getNodeName());
                EndpointPolicyByIpRequest request = new EndpointPolicyByIpRequest();
                request.setIpAddress(ip);
                SampleHelper.postObjectAndPrint(url, config.getUserName(), secret,
config.getSSLContext().getSocketFactory(), request);
        }


        public static void main(String [] args) throws Exception {
                SampleConfiguration config = new SampleConfiguration();
                PxgridControl pxgrid = new PxgridControl(config);

                while (pxgrid.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                Console.log("pxGrid controller version=" + pxgrid.getControllerVersion());

                String ip = SampleHelper.prompt("IP address: ");
                clear(config, ip);
        }
}
```

ANCCreatePolicy

This code creates an ISE ANC policy based on Quarantine, Shut_down, and Port_Bounce Actions

```
package com.cisco.pxgrid.samples.ise.anc;

import java.io.IOException;
import java.util.Arrays;
import java.util.List;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.samples.ise.http.Console;
import com.cisco.pxgrid.samples.ise.http.PxgridControl;
import com.cisco.pxgrid.samples.ise.http.SampleConfiguration;
import com.cisco.pxgrid.samples.ise.http.SampleHelper;

/**
 * Demonstrates how to query a session using IP address
 */
public class AncCreatePolicy {
        private static class Policy {
                public enum Action {
                        QUARANTINE, SHUT_DOWN, PORT_BOUNCE
                }
                private String name;
                private List<Action> actions;
                public String getName() {
                        return name;
                }
                public void setName(String name) {
                        this.name = name;
                }
                public List<Action> getActions() {
                        return actions;
                }
                public void setActions(List<Action> actions) {
                        this.actions = actions;
                }
        }

        private static void create(SampleConfiguration config, String policyName) throws IOException {
                PxgridControl pxgrid = new PxgridControl(config);
                Service[] services = pxgrid.lookupService("com.cisco.ise.config.anc");
                if (services == null || services.length == 0) {
                        System.out.println("Service unavailabe");
                        return;
                }
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/createPolicy";
                String secret = pxgrid.getAccessSecret(service.getNodeName());
                Policy policy = new Policy();
                policy.setName(policyName);
                policy.setActions(Arrays.asList(Policy.Action.QUARANTINE));
                SampleHelper.postObjectAndPrint(url, config.getUserName(), secret,
config.getSSLContext().getSocketFactory(), policy);
        }


        public static void main(String [] args) throws Exception {
                SampleConfiguration config = new SampleConfiguration();
                PxgridControl pxgrid = new PxgridControl(config);

                while (pxgrid.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                Console.log("pxGrid controller version=" + pxgrid.getControllerVersion());

                String policyName = SampleHelper.prompt("Create policy name: ");
                create(config, policyName);
        }
```

```
}
```

ANCDeletePolicy

This code deletes an ISE ANC policy

```
package com.cisco.pxgrid.samples.ise.anc;

import java.io.IOException;
import java.util.Arrays;
import java.util.List;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.samples.ise.http.Console;
import com.cisco.pxgrid.samples.ise.http.PxgridControl;
import com.cisco.pxgrid.samples.ise.http.SampleConfiguration;
import com.cisco.pxgrid.samples.ise.http.SampleHelper;

/**
 * Demonstrates how to query a session using IP address
 */
public class AncDeletePolicy {
        private static class Policy {
                private String id;
                public String getId() {
                        return id;
                }
                public void setId(String id) {
                        this.id = id;
                }
        }

        private static void delete(SampleConfiguration config, String id) throws IOException {
                PxgridControl pxgrid = new PxgridControl(config);
                Service[] services = pxgrid.lookupService("com.cisco.ise.config.anc");
                if (services == null || services.length == 0) {
                        System.out.println("Service unavailabe");
                        return;
                }
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/deletePolicyById";
                String secret = pxgrid.getAccessSecret(service.getNodeName());
                Policy policy = new Policy();
                policy.setId(id);
                SampleHelper.postObjectAndPrint(url, config.getUserName(), secret,
config.getSSLContext().getSocketFactory(), policy);
        }


        public static void main(String [] args) throws Exception {
                SampleConfiguration config = new SampleConfiguration();
                PxgridControl pxgrid = new PxgridControl(config);

                while (pxgrid.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                Console.log("pxGrid controller version=" + pxgrid.getControllerVersion());

                String id = SampleHelper.prompt("Create policy id: ");
                delete(config, id);
        }
}
```

ANCGetByOperationID

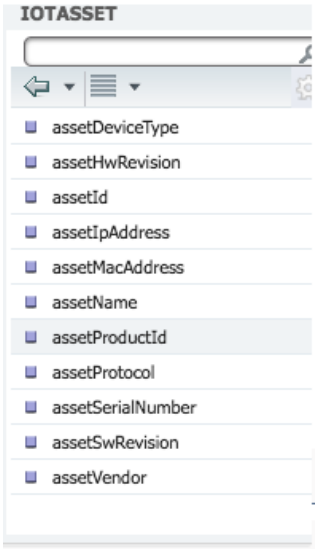
This code obtains the OperationID for retrieving the ANC policy.

```
package com.cisco.pxgrid.samples.ise.anc;

import java.io.IOException;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.samples.ise.http.Console;
import com.cisco.pxgrid.samples.ise.http.PxgridControl;
import com.cisco.pxgrid.samples.ise.http.SampleConfiguration;
import com.cisco.pxgrid.samples.ise.http.SampleHelper;

/**
 * Demonstrates how to query a session using IP address
 */
public class AncGetByOperationId {
        private static class GetOperationStatusRequest {
                private String operationId;
                public void setOperationId(String operationId) {
                        this.operationId = operationId;
                }
        }

        private static void getOperationStatus(SampleConfiguration config, String id) throws IOException {
                PxgridControl pxgrid = new PxgridControl(config);
                Service[] services = pxgrid.lookupService("com.cisco.ise.config.anc");
                if (services == null || services.length == 0) {
                        System.out.println("Service unavailabe");
                        return;
                }
                Service service = services[0];
                String url = service.getProperties().get("restBaseUrl") + "/getOperationStatus";
                String secret = pxgrid.getAccessSecret(service.getNodeName());
                GetOperationStatusRequest request = new GetOperationStatusRequest();
                request.setOperationId(id);
                SampleHelper.postObjectAndPrint(url, config.getUserName(), secret,
config.getSSLContext().getSocketFactory(), request);
        }


        public static void main(String [] args) throws Exception {
                SampleConfiguration config = new SampleConfiguration();
                PxgridControl pxgrid = new PxgridControl(config);

                while (pxgrid.accountActivate() != AccountState.ENABLED)
                        Thread.sleep(60000);
                Console.log("pxGrid controller version=" + pxgrid.getControllerVersion());

                String id = SampleHelper.prompt("Operation ID: ");
                getOperationStatus(config, id);
        }
}
```

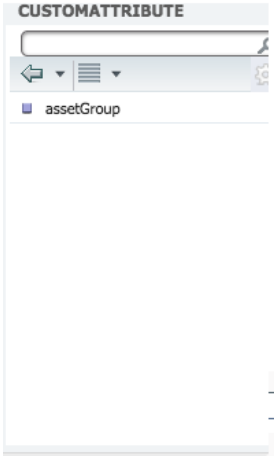
Cisco pxGrid Context-In

Cisco pxGrid Context-in provides ecosystem partners to publish device or endpoint asset context attributes into ISE based on the pxGrid client publishing to the asset topic or asset service, /topic/com.cisco.endpoint.asset. ISE will subscribe to this /topic/com.cisco.endpoint.asset topic. An ISE profiling policy will be created using these asset attributes. This ISE profile will be assigned an ISE logical profile to be used in an ISE authorization policy to provide secure network access for the device or endpoint asset. Custom attributes can also be defined in a profiling policy and can be used in an ISE authorization policy as well.

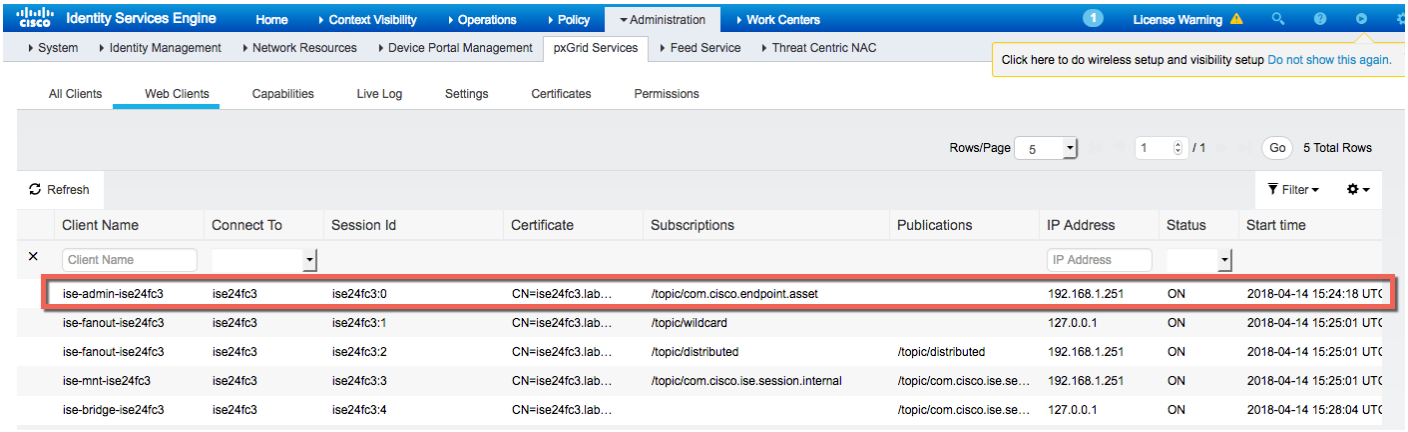
These attributes include: assetID, assetName, assetIpAddress, assetMacAddress, assetVendor, assetProductId, assetSerialNumber, assetDeviceType, assetSwRevision, assetHwRevision, assetProtocol and are defined in the IOT Assets profiling dictionary when creating the Asset's profiling policy. The asset attribute values are defined by the ecosystem partner in the custompublisher.java code. The API_Simulator Code listing is provided in the *API_Simulator* Context-In section.



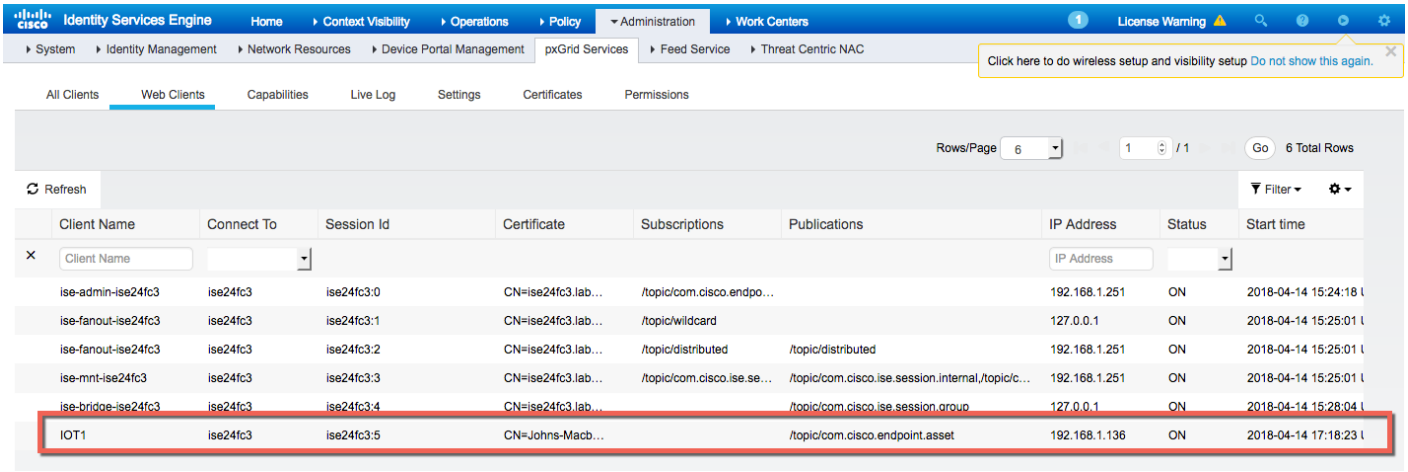
If the ecosystem partner has custom attributes that are not defined in the IOT Assets profiling dictionary, these can also be published. These custom attributes are defined in JSON array in the custompublisher.java code. These are also defined in the Custom Attributes profiling policies and defined in the ISE Identity Custom endpoint settings.



The Cisco ISE pxGrid node controller becomes the subscriber to the endpoint Asset topic and consumes this information.



The pxGrid client publishes to the Asset topic, /topic/com.cisco.endpoint.asset. IO1 is the registered pxGrid client publishing to this asset topic.



Enabling pxGrid as Subscriber for Profiling

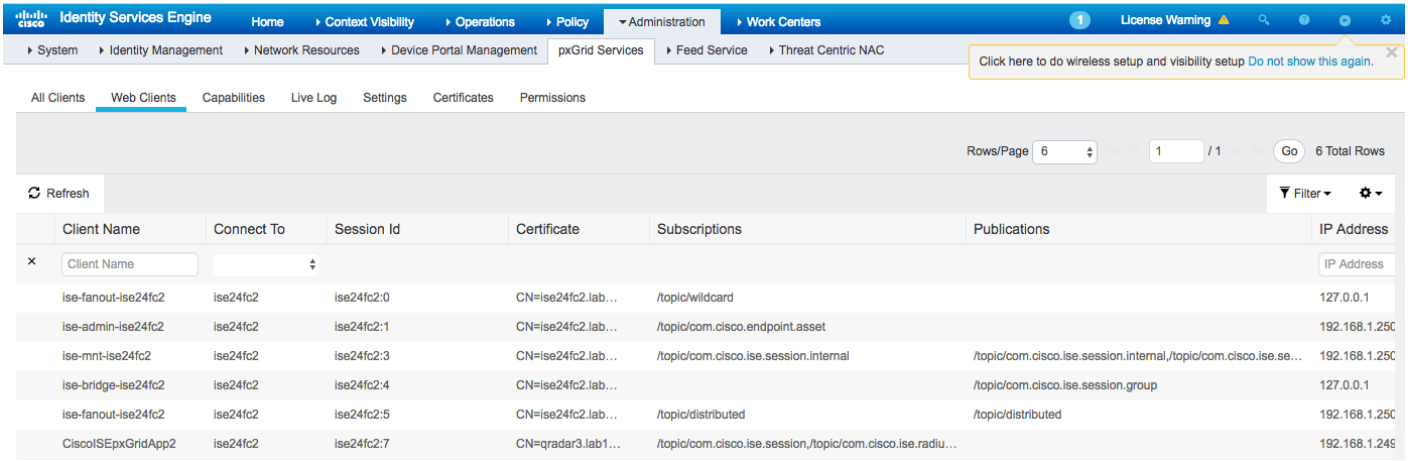
Step 1 Select **Administrator->System->Deployment>the ISE node->Edit the node->Profiling Configuration**

Step 2 Enable **pxGrid**

Step 3 Select **Save**

Step 4 Select **Administrator->pxGrid Services->Web Clients**

Note: The ISE admin node has subscribed to the endpoint asset topic



Running API_Simulator

The API Simulator is a tool that is used to provide pxGrid context-in simulating a pxGrid client asset device-publishing asset attributes to the ISE pxgrid node.

The *./run_publisher38 script* contains the command-line arguments to run the simulator, this also provides the access secret between the publisher and ISE pxGrid node to begin publishing content from the asset device or client endpoint in the script.

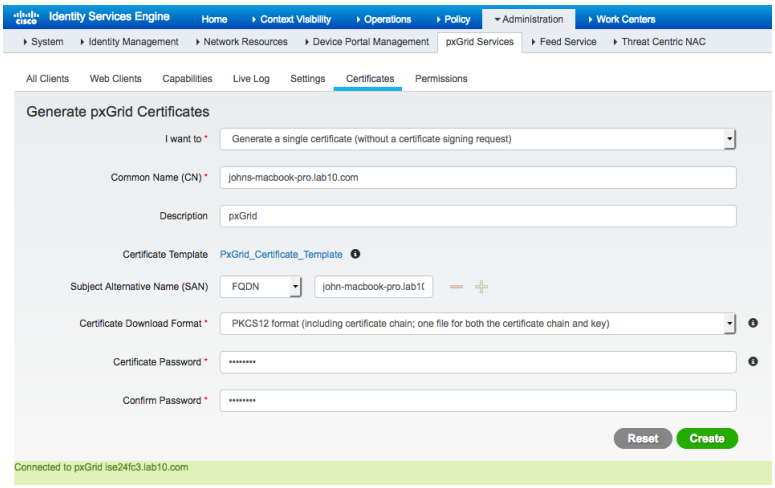
If you desire to modify the existing attributes in the script you must modify the custompublisher.java code. An example is provided using Maven.

A profiling policy will be created based on the asset attribute values from the client device. The profiling policy will then be assigned a Logical Profile that can be assigned in an ISE authorization condition rule that can be used in an ISE authorization policy to determine network access for the asset device. We will also illustrate the same example using custom attributes.

Step 1 Extract the contents of the API Simulator to a folder

Step 2 Create a PKCS12 certificate from **ISE->Administration->pxGrid Services->Certificates** and provide the following information:

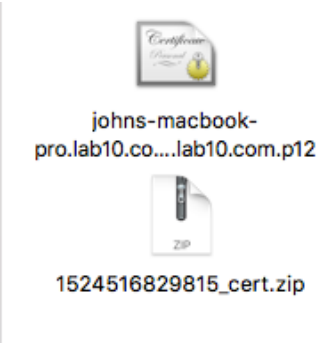
Note: CN name should be Fully Qualified Domain Name (FQDN) resolvable. PKCS12 format is not supported using Python libraries.



Step 3 Select **Create**

Step 4 Download the zipped file

You should see:



Step 5 Edit the ***./run_publisher38.sh*** file:

where:

-DPXGRID_HOSTNAMES= ISE pxGrid node (i.e. ise24fc3.lab10.com)
-DPXRID_USERNAME= pxGrid client name (i.e. IOT1)
-DPXGRID_GROUP= pxGrid client group (i.e. Session)
-DPXGRID_KEYSTORE_FILENAME= client keystore file (i.e. Johns-Macbook-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12)
-DPXGRID_KEYSTORE_PASSWORD=client keystore file password (i.e. Cisco123)
-DPXGRID_TRUSTSTORE_FILENAME= truststore or root keystore files (i.e. Johns-Macboo-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12)
-DPXGRID_KEYSTORE_PASSWORD=truststore or root keystore file password (i.e. Cisco123)

```
java -jar -DPXGRID_HOSTNAMES="ise24fc23.lab10.com" -DPXGRID_USERNAME="IOT1" -DPXGRID_GROUP="Session"  -
DPXGRID_KEYSTORE_FILENAME="Johns-Macbook-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12" -
DPXGRID_KEYSTORE_PASSWORD="Cisco123" -DPXGRID_TRUSTSTORE_FILENAME="Johns-Macbook-Pro.lab10.com_Johns-Macbook-
Pro.lab10.com.p12" -DPXGRID_TRUSTSTORE_PASSWORD="Cisco123" pxgrid-rest-ws-samples-2.0.0-SNAPSHOT.jar
```

Step 6 Type: **./run_publisher38.sh**, you will see the following:

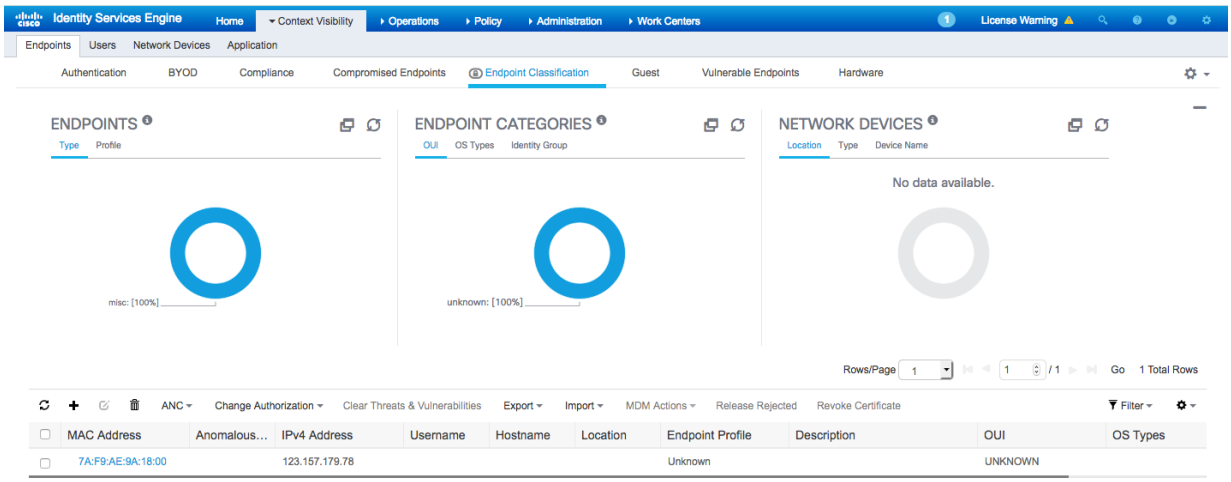
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.6.RELEASE)

2018-04-14 13:18:07.134  INFO 5546 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : Starting
CustomPublisher v2.0.0-SNAPSHOT on johns-macbook-pro.lab10.com with PID 5546
(/Applications/api_partner_fc3/api_simulator/pxgrid-rest-ws-samples-2.0.0-SNAPSHOT.jar started by jeppich in
/Applications/api_partner_fc3/api_simulator)
2018-04-14 13:18:07.167  INFO 5546 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : No active
profile set, falling back to default profiles: default
2018-04-14 13:18:07.611  INFO 5546 --- [           main] ationConfigEmbeddedWebApplicationContext :
Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@bebdb06:
startup date [Sat Apr 14 13:18:07 EDT 2018]; root of context hierarchy
2018-04-14 13:18:14.523  INFO 5546 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat
initialized with port(s): 8080 (http)
2018-04-14 13:18:14.649  INFO 5546 --- [           main] o.apache.catalina.core.StandardService   : Starting
service [Tomcat]
2018-04-14 13:18:14.672  INFO 5546 --- [           main] org.apache.catalina.core.StandardEngine  : Starting
Servlet Engine: Apache Tomcat/8.5.16
2018-04-14 13:18:15.101  INFO 5546 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]       :
Initializing Spring embedded WebApplicationContext
2018-04-14 13:18:15.102  INFO 5546 --- [ost-startStop-1] o.s.web.context.ContextLoader            : Root
WebApplicationContext: initialization completed in 7517 ms
2018-04-14 13:18:15.348  INFO 5546 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean  : Mapping
servlet: 'dispatcherServlet' to [/]
2018-04-14 13:18:15.386  INFO 5546 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'characterEncodingFilter' to: [/*]
2018-04-14 13:18:15.387  INFO 5546 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'hiddenHttpMethodFilter' to: [/*]
2018-04-14 13:18:15.387  INFO 5546 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'httpPutFormContentFilter' to: [/*]
2018-04-14 13:18:15.387  INFO 5546 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'requestContextFilter' to: [/*]
2018-04-14 13:18:16.062  INFO 5546 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking
for @ControllerAdvice:
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@bebdb06: startup date
[Sat Apr 14 13:18:07 EDT 2018]; root of context hierarchy
2018-04-14 13:18:16.251  INFO 5546 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/getAssets]}" onto public com.cisco.pxgrid.samples.ise.http.DeviceList
com.cisco.pxgrid.samples.ise.http.PublisherController.device()
2018-04-14 13:18:16.257  INFO 5546 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String,
java.lang.Object>>
org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2018-04-14 13:18:16.257  INFO 5546 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpServletReque
st,javax.servlet.http.HttpServletResponse)
2018-04-14 13:18:16.288  INFO 5546 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/webjars/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-14 13:18:16.288  INFO 5546 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-14 13:18:16.386  INFO 5546 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/**/favicon.ico] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-14 13:18:16.764  INFO 5546 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        :
Registering beans for JMX exposure on startup
2018-04-14 13:18:17.148  INFO 5546 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat
started on port(s): 8080 (http)
2018-04-14 13:18:17.159  INFO 5546 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : Started
CustomPublisher in 12.451 seconds (JVM running for 18.583)
------- properties -------
  hostnames=ise24fc3.lab10.com
```

```
  username=IOT1
  password=null
  groups=Session
  description=null
  keystoreFilename=Johns-Macbook-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12
  keystorePassword=Cisco123
  truststoreFilename=Johns-Macbook-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12
  truststorePassword=Cisco123
---------------------------
2018-04-14 13:18:19.919  INFO 5546 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={}
2018-04-14 13:18:20.590  INFO 5546 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"accountState":"ENABLED","version":"2.0.0.13"}
14-Apr-18 13:18:20.591 [main-1]: pxGrid controller version=2.0.0.13
2018-04-14 13:18:20.601  INFO 5546 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"name":"com.cisco.endpoint.asset","properties":{"wsPubsubService":"com.cisco.ise.pubsub","restBaseUR
L":"http://raghdasa-lnv1:8080","assetTopic":"/topic/com.cisco.endpoint.asset"}}
2018-04-14 13:18:20.735  INFO 5546 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={}
2018-04-14 13:18:20.742  INFO 5546 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"name":"com.cisco.ise.pubsub"}
2018-04-14 13:18:20.751  INFO 5546 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"services":[{"name":"com.cisco.ise.pubsub","nodeName":"ise-pubsub-
ise24fc3","properties":{"wsUrl":"wss://ise24fc3.lab10.com:8910/pxgrid/ise/pubsub"}}]}
14-Apr-18 13:18:20.751 [main-1]: wsUrl=wss://ise24fc3.lab10.com:8910/pxgrid/ise/pubsub
2018-04-14 13:18:20.757  INFO 5546 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"peerNodeName":"ise-pubsub-ise24fc3"}
2018-04-14 13:18:20.909  INFO 5546 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"secret":"tASXtHAbKDKbAODh"}
2018-04-14 13:18:22.072  INFO 5546 --- [     Grizzly(1)] c.c.p.s.i.h.StompPubsubClientEndpoint    : WS onOpen
2018-04-14 13:18:22.075  INFO 5546 --- [           main] c.c.p.s.i.h.StompPubsubClientEndpoint    : STOMP
CONNECT host=ise24fc3.lab10.com
press <enter> to start the publishing...2018-04-14 13:18:22.084  INFO 5546 --- [     Grizzly(2)]
c.c.p.s.i.h.StompPubsubClientEndpoint    : STOMP CONNECTED version=1.2
```

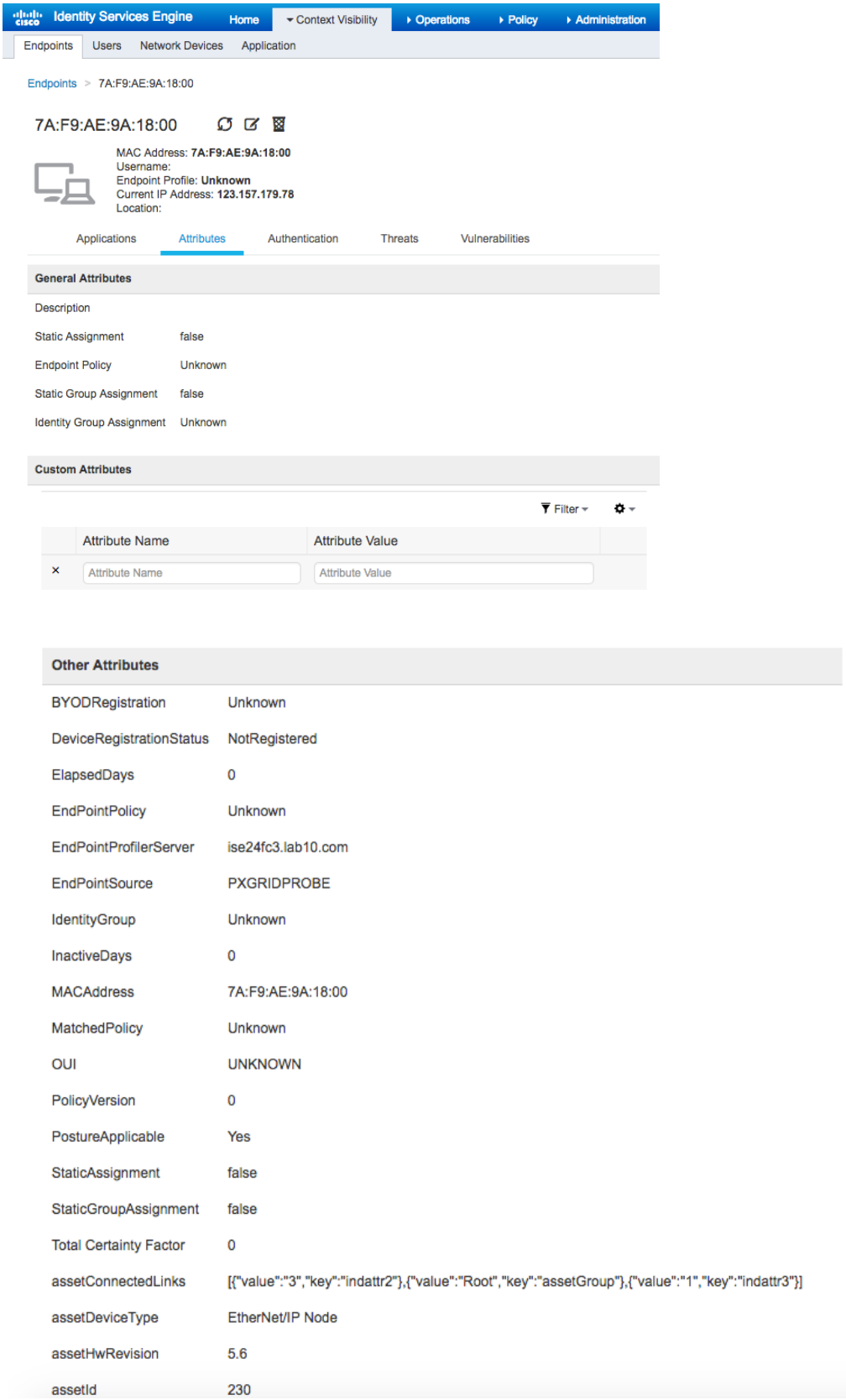
Viewing Asset Device in Context Visibility Screen

Step 1 Select **Context Visibility->Endpoints->Endpoint Classification**
You will see the client endpoint generated from the script as determined by the MAC address and the client IP address.

Note: if you do not see the endpoint, verify that the ISE admin node has subscribed to the endpoint asset topic. If the ISE Admin node has not subscribed to the ISE endpoint asset topic, ensure that pxGrid is enabled for Profiling under Administration->System->Deployment->edit the node and select Profiling Configuration.



Step 2 Click on the **MAC address,** and select **Attributes,** you will see the attributes:



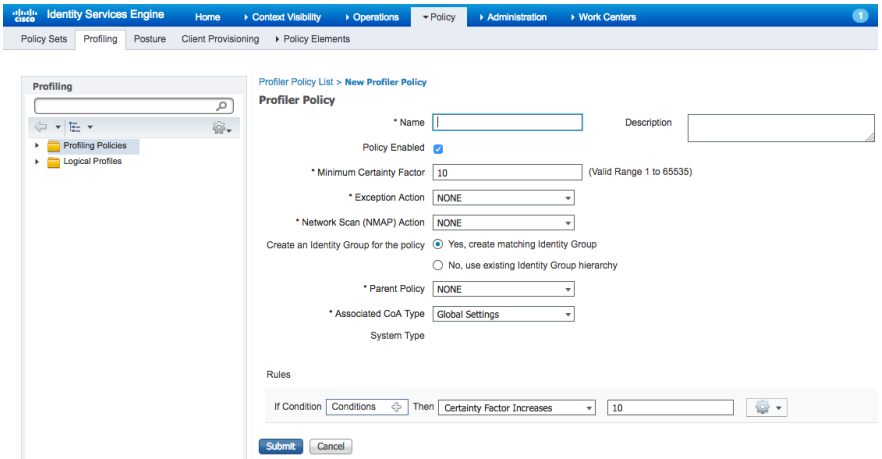
assetIpAddress	123.157.179.78
assetMacAddress	7a:f9:ae:9a:18:00
assetName	Abjergaryn - 45
assetProductId	IE2000
assetProtocol	CIP
assetSerialNumber	1212121213243
assetSwRevision	4.6
assetVendor	Cisco Systems
ip	123.157.179.78

Creating Profiling Policy Based on Asset Attributes

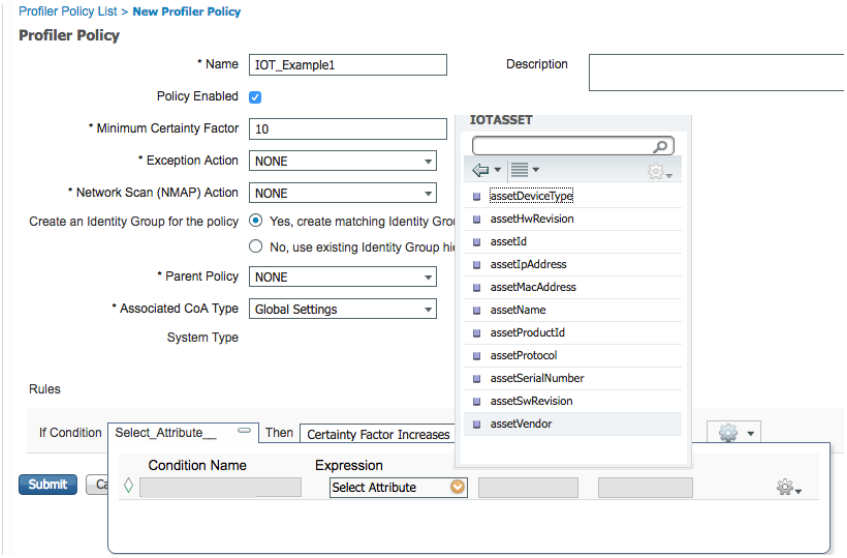
In this example, we define an IOT_Example1 profiling policy based on the attribute values of the client or asset device. These attribute values are mapped to the IOTASSETS dictionary attributes to define the policy. This profiling policy will then be assigned to an IOTDevices1 logical profile. This logical profile can then be added as an ISE authorization condition rule to be used in an ISE authorization policy to determine the asset's network access. A security group tag of IOT devices will also be created and added to the ISE authorization policy.

Note: When running the script, you will not see the RADIUS events that trigger this rule, since the script simulates endpoint. However, when you implement the code and test in your environment, this rule will be triggered.

Step 1 Select **Policy->Profiling->Profiling Policies->Add**
You will see the following:

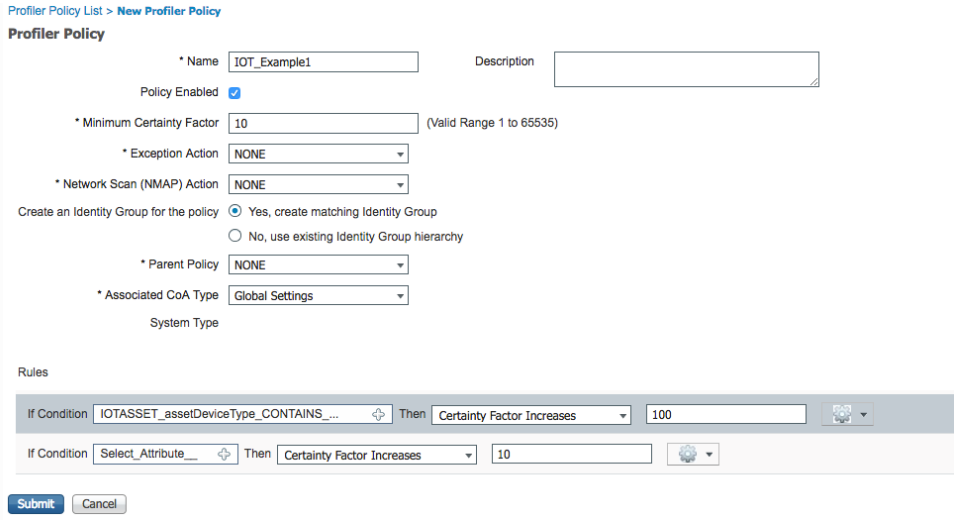


Step 2 Enter: **IOT_Example1** for Name
Step 3 Under **Rules->If Condition->Create New Condition (Advanced Option)**
Step 4 Under **Expression->Select Attribute->Iotas set**
Step 5 You should see:



Step 6 Select **assetDeviceType:contains:EtherNetVIP Node**, enter, increase certainty factor to 100

Step 7 Select **"Gear"->"Insert line below"**



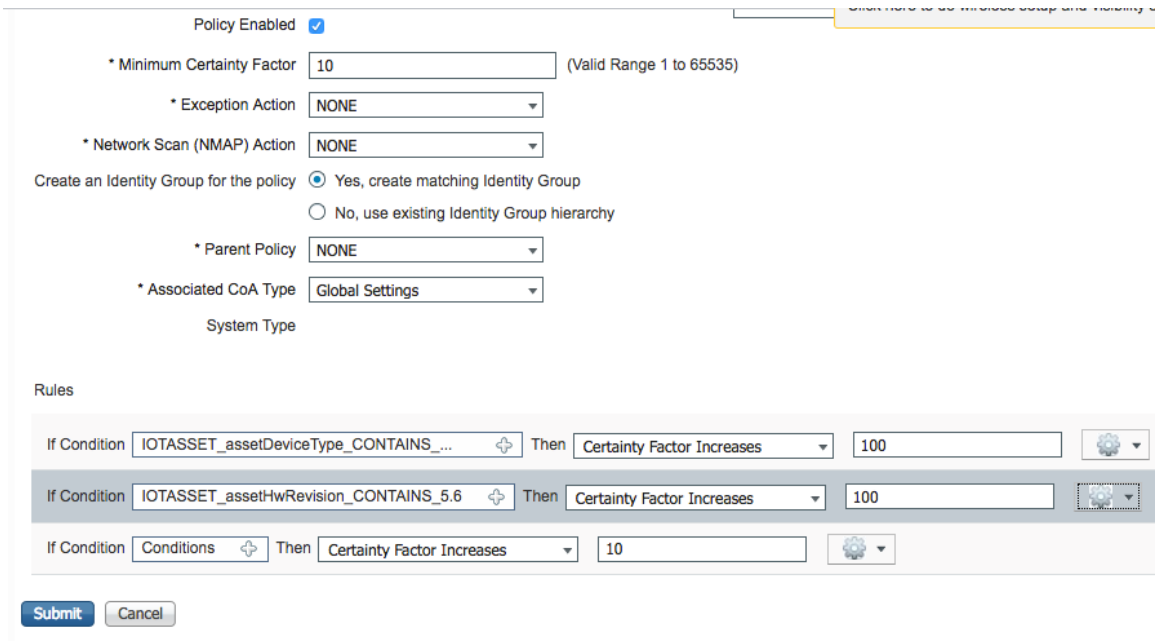
Step 8 Under **Rules->If Condition->Create New Condition(Advanced Option)**

Step 9 Under **Expression->Select Attribute->IOTAsset**

Step 10 Select **assetHwRevision:contains:5.6**, enter, **increase certainty factor to 100**

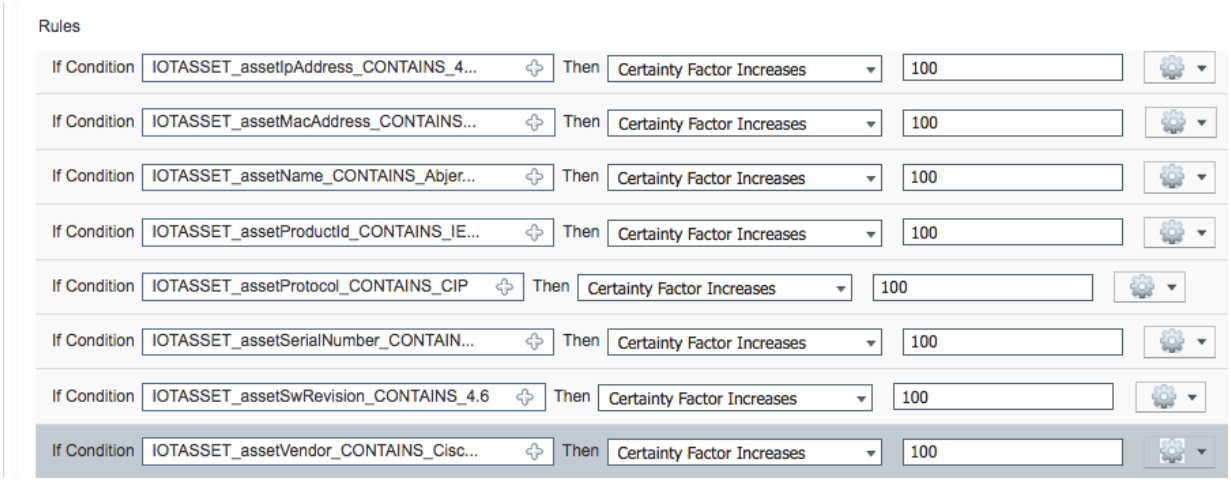
Step 11 Select **"Gear"->"Insert line below"**

Step 12 You should see:

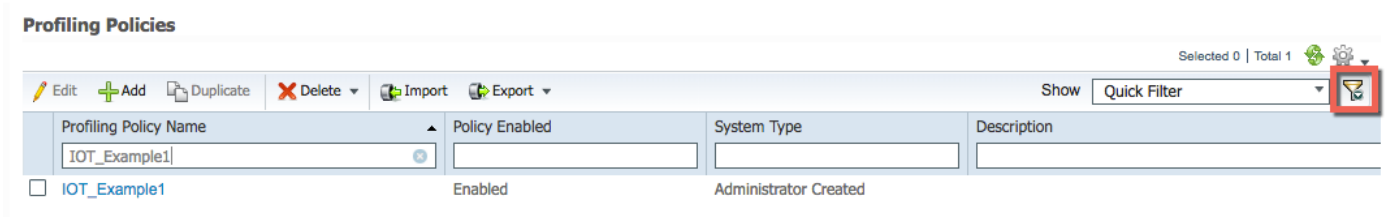


Step 13 Under **Rules->If Condition->Create New Condition(Advanced Option)**
Step 14 Under **Expression->Select Attribute->IOT Asset**
Step 15 Select **assetId**:contains:**230**, enter, increase certainty factor to 100
Step 16 Select **"Gear"->"Insert line below"**
Step 17 Under **Rules->If Condition->Create New Condition(Advanced Option)**
Step 18 Under **Expression->Select Attribute->IOT Asset**
Step 19 Select **assetIpAddress**:contains:**123.157.78.79**, enter, increase certainty factor to 100
Step 20 Select **"Gear"->"Insert line below**
Step 21 Under **Rules->If Condition->Create New Condition(Advanced Option)**
Step 22 Under **Expression->Select Attribute->IOT Asset**
Step 23 Select **assetMacAddress**:contains:**78:f9:ae:9a:18:00**, enter, increase certainty factor to 100
Step 24 Select "**Gear"->"Insert line below**
Step 25 Under **Rules->If Condition->Create New Condition(Advanced Option)**
Step 26 Under **Expression->Select Attribute->IOT Asset**
Step 27 Select **assetId:contains:230**, enter, increase certainty factor to 100
Step 28 Select **"Gear"->"Insert line below**
Step 29 Under **Rules->If Condition->Create New Condition(Advanced Option)**
Step 30 Under **Expression->Select Attribute->IOT Asset**
Step 31 Select **assetName:contains:Abjergaryn - 45** enter, increase certainty factor to 100
Step 32 Select **"Gear"->"Insert line below**
Step 33 Under **Rules->If Condition->Create New Condition(Advanced Option)**
Step 34 Under **Expression->Select Attribute->IOT Asset**
Step 35 Select **assetProductId**:contains:**IE2000**, enter, increase certainty factor to 100
Step 36 Select **"Gear"->"Insert line below**
Step 37 Under **Rules->If Condition->Create New Condition(Advanced Option)**
Step 38 Under **Expression->Select Attribute->IOT Asset**
Step 39 Select **assetProtocol:contains:CIP**, enter, increase certainty factor to 100
Step 40 Select **"Gear"->"Insert line below**
Step 41 Under **Rules->If Condition->Create New Condition(Advanced Option)**

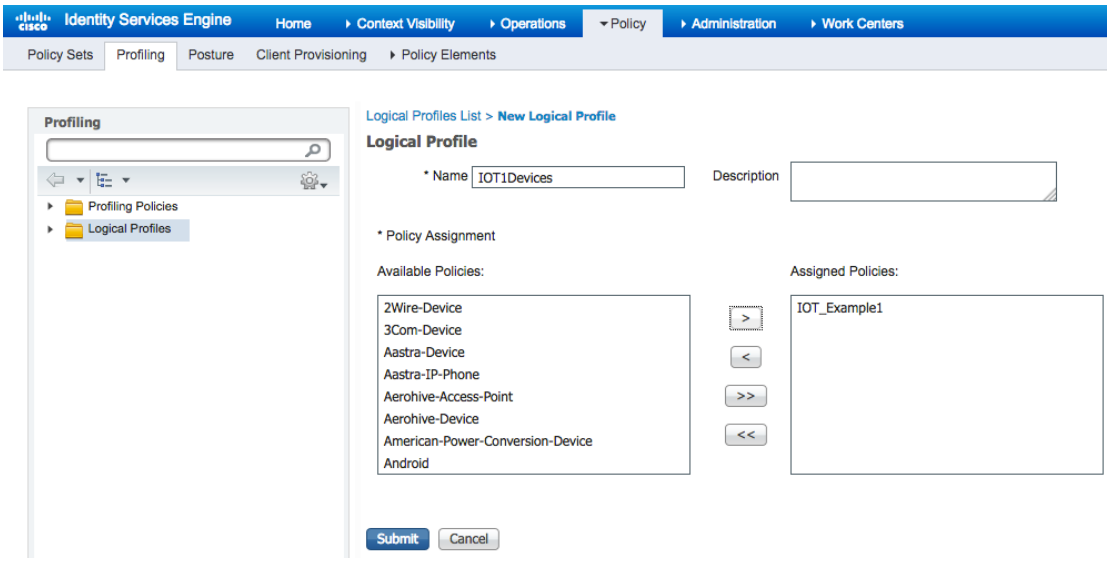
Step 42 Under **Expression->Select Attribute->IOT Asset**
Step 43 Select **assetSerialNumber**:contains: **1212121213243**, enter, increase certainty factor to 100
Step 44 Select **"Gear"->"Insert line below**
Step 45 Under **Rules->If Condition->Create New Condition(Advanced Option)**
Step 46 Under **Expression->Select Attribute->IOT Asset**
Step 47 Select **assetSwRevision:contains:4.6**, enter, increase certainty factor to 100
Step 48 Select **"Gear"->"Insert line below**
Step 49 Under **Rules->If Condition->Create New Condition(Advanced Option)**
Step 50 Under **Expression->Select Attribute->IOT Asset**
Step 51 Select **assetVendor:contains:Cisco Systems**, enter, increase certainty factor to 100
Step 52 Select **"Gear"->"Insert line below**
Step 53 You should see the following



Step 54 Select **Submit**
Step 55 Select Funnel, enter **IOT_Example1** under the **Profiling Policy Name**, you should see:

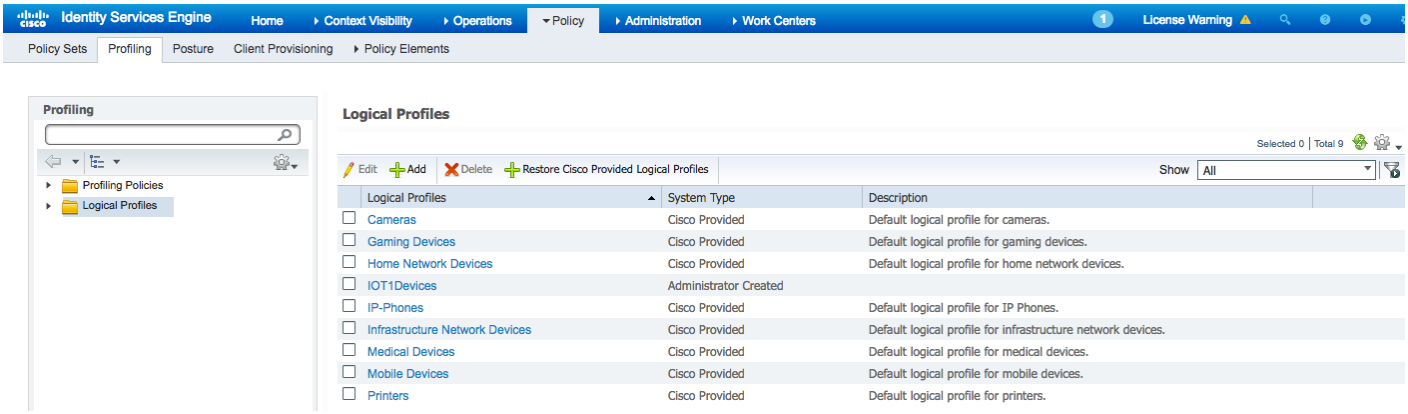


Step 56 Add IOT_Example1 to Logical Profile
Select **Profiling->Logical Profiles->Add->Name: IOT1Devices->Policy Assignment->Available Policies->IOT_Example1 and move to Assigned Polices**



Step 57 Select **Submit**

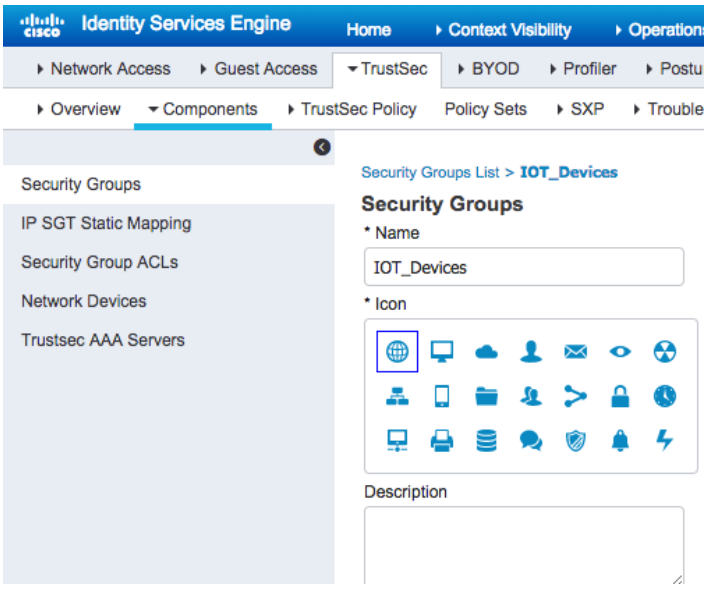
Step 58 You should see the logical profile IOTDevices



Creating Authorization Policy Based on Asset's Logical Profile

We will create an Authorization policy that determines the asset's network access. A Security Group Tag (SGT) will also be added to the Authorization Policy provided label to classify network traffic if Cisco's TrustSec Solution is used.

Step 1 Create IOT device SGT which will be used in the Authorization Policy
Select **WorkCenters->Trustsec->Components->Add->IOT Devices**

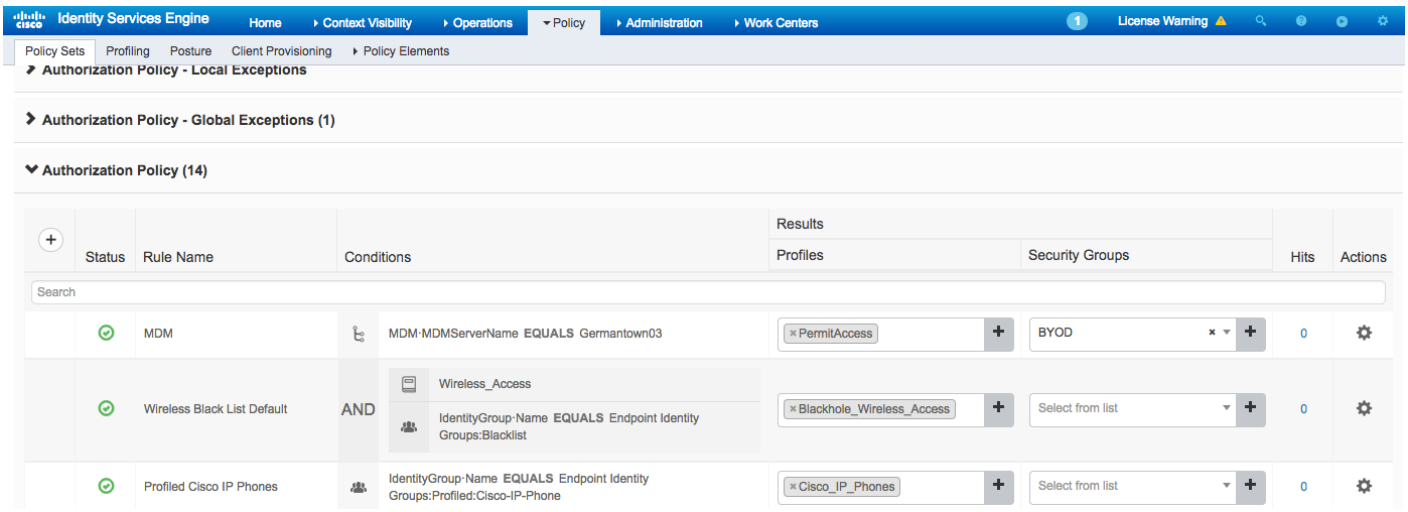


Step 2 Select **Save**

Step 3 Create ISE Authorization Profile Policy

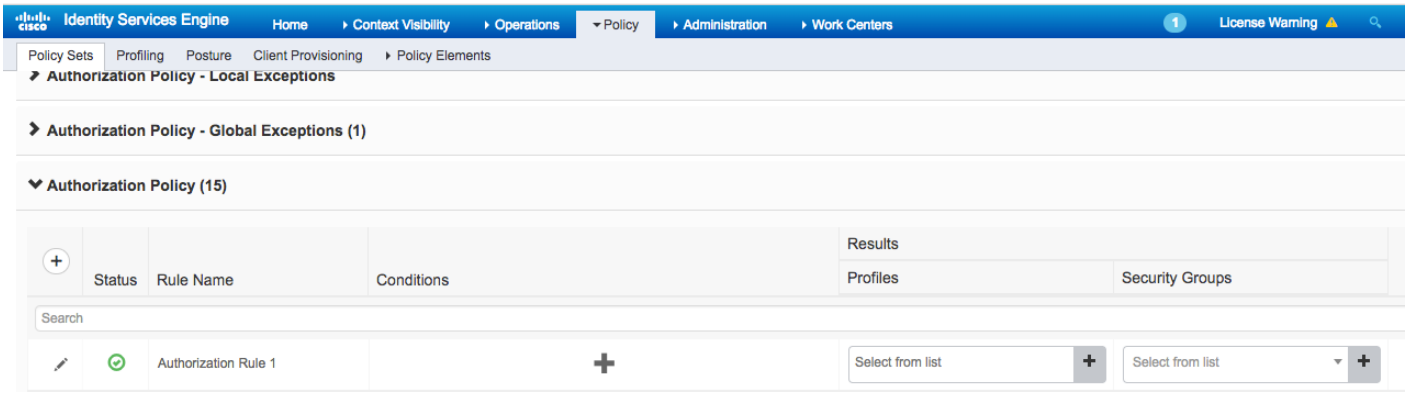
Step 4 Select **Policy->Policy Sets->View ">"->Authorization Policy**

Step 5 You should see the authorization policy:



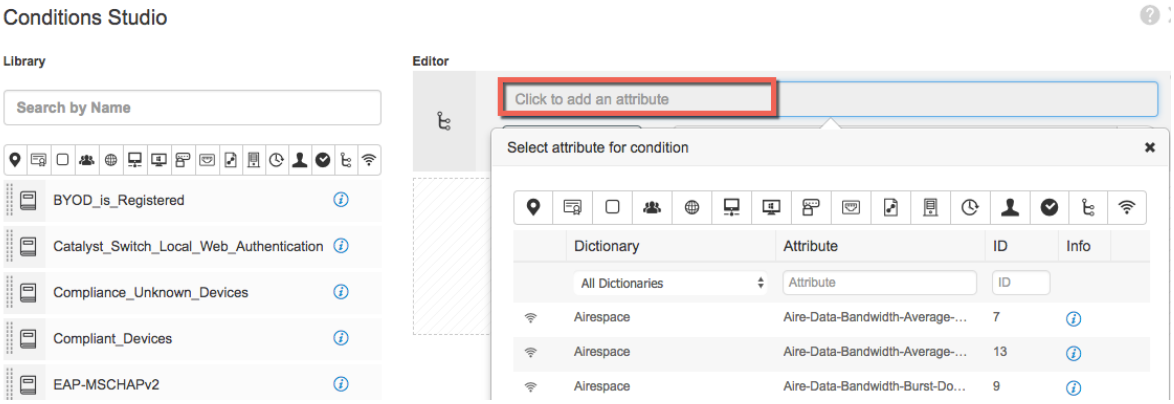
Step 6 Under **Actions**, **Click on "gear"** and "**Insert new rule above**

Step 7 You should see the following:

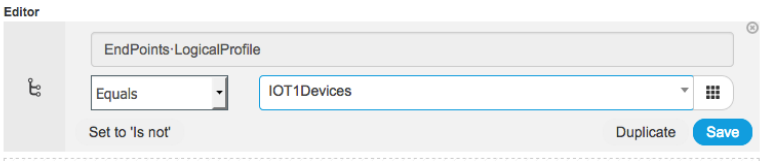


Step 8 Name Authorization Rule 1 to **IOT_Example_Rule**, **Under Condition**, select **"+"**

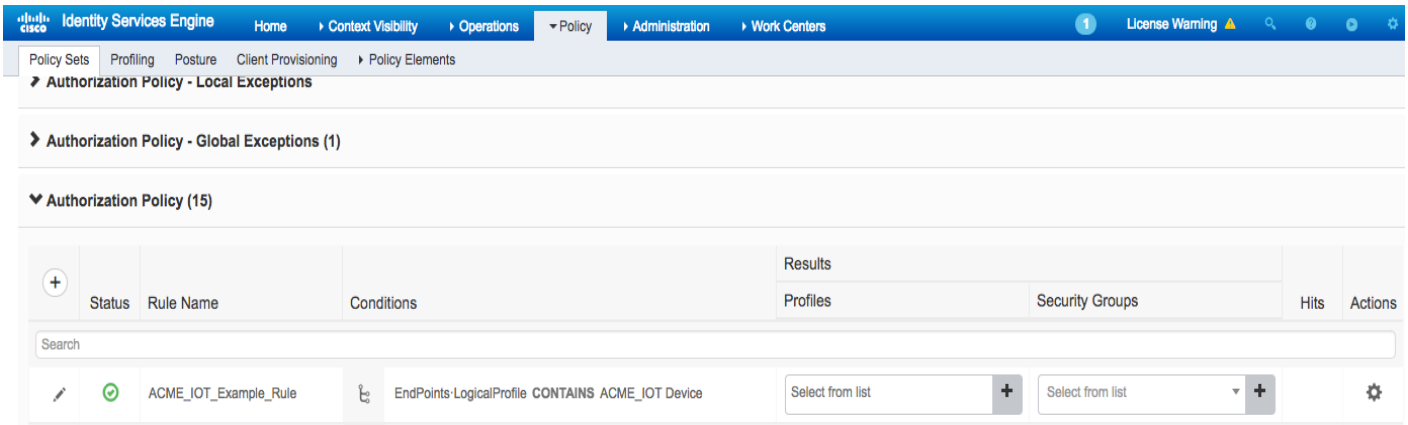
Step 9 Click on **Attribute**, you should see the following Dictionary Attributes



Step 10 Select **Endpoints->Logical Profile->Contains->IOT1Devices**



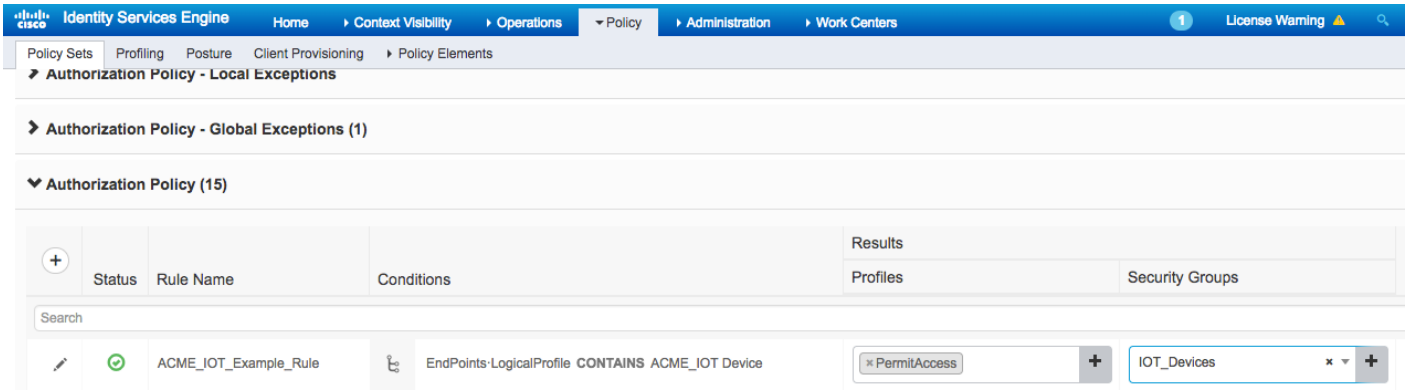
Step 11 Select **Use**
You should see:



Step 12 Under **Profiles**, select **Permit Access**

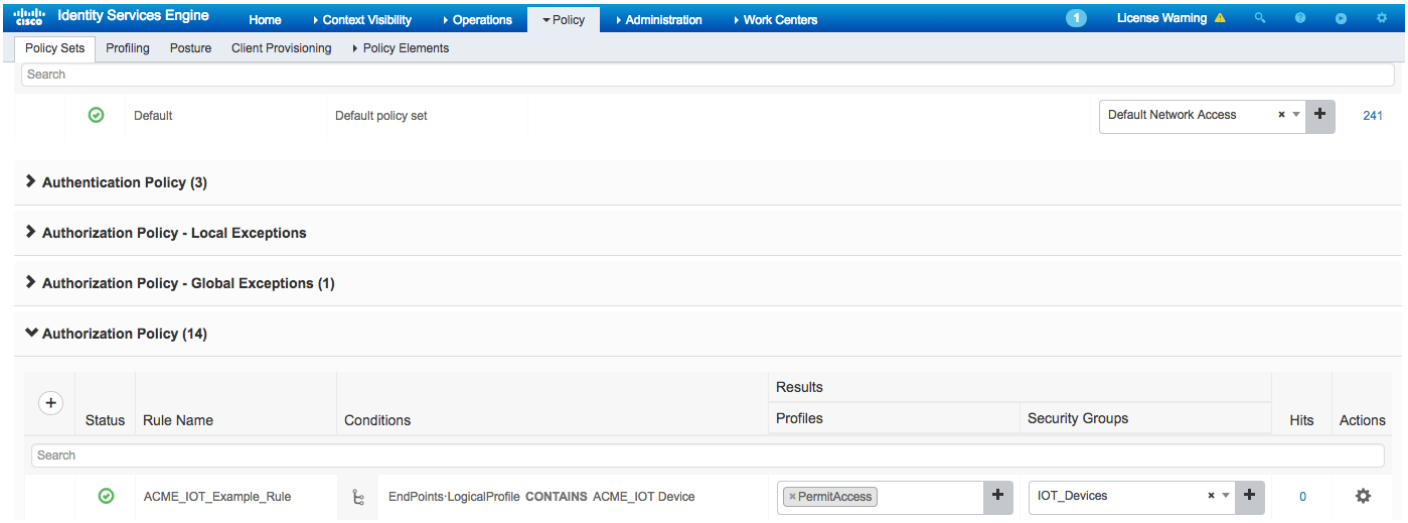
Step 13 Under **Security Groups**, select **IOT Devices**

You should see:



Step 14 Select **Save**

Step 15 You should see:



Verifying Asset as Defined by the Logical Profile

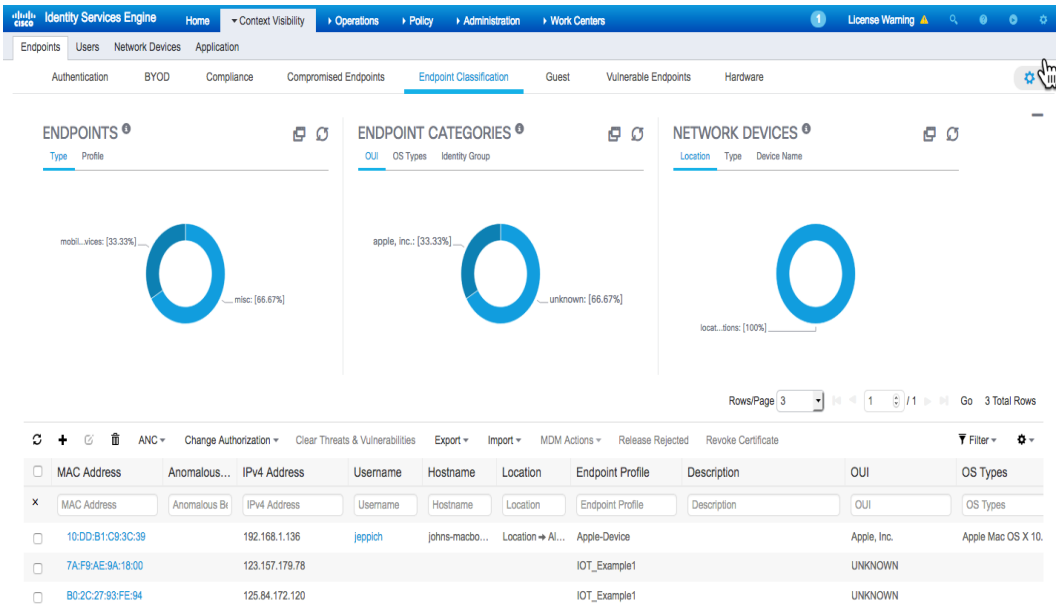
The Asset policy is now assigned to the logical profile and verified in the Context Visibility Endpoint Classification screen.

Step 1 Run script again, type: **./run_publisher38.sh**

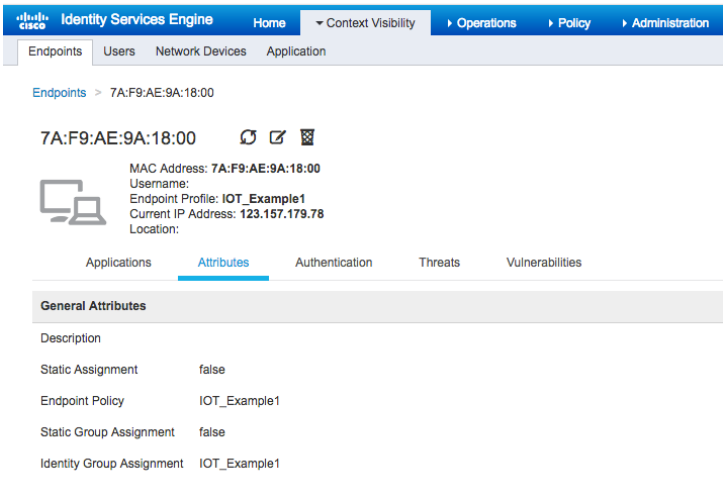
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.6.RELEASE)

2018-04-14 15:42:10.009  INFO 5926 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : Starting
CustomPublisher v2.0.0-SNAPSHOT on johns-macbook-pro.lab10.com with PID 5926
(/Applications/api_partner_fc3/api_simulator/pxgrid-rest-ws-samples-2.0.0-SNAPSHOT.jar started by jeppich in
/Applications/api_partner_fc3/api_simulator)
2018-04-14 15:42:10.042  INFO 5926 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : No active
profile set, falling back to default profiles: default
2018-04-14 15:42:10.508  INFO 5926 --- [           main] ationConfigEmbeddedWebApplicationContext :
Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@bebdb06:
startup date [Sat Apr 14 15:42:10 EDT 2018]; root of context hierarchy
2018-04-14 15:42:17.010  INFO 5926 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat
initialized with port(s): 8080 (http)
2018-04-14 15:42:17.136  INFO 5926 --- [           main] o.apache.catalina.core.StandardService   : Starting
service [Tomcat]
2018-04-14 15:42:17.158  INFO 5926 --- [           main] org.apache.catalina.core.StandardEngine  : Starting
Servlet Engine: Apache Tomcat/8.5.16
2018-04-14 15:42:17.587  INFO 5926 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]       :
Initializing Spring embedded WebApplicationContext
2018-04-14 15:42:17.588  INFO 5926 --- [ost-startStop-1] o.s.web.context.ContextLoader            : Root
WebApplicationContext: initialization completed in 7105 ms
2018-04-14 15:42:17.813  INFO 5926 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean  : Mapping
servlet: 'dispatcherServlet' to [/]
2018-04-14 15:42:17.828  INFO 5926 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'characterEncodingFilter' to: [/*]
2018-04-14 15:42:17.829  INFO 5926 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'hiddenHttpMethodFilter' to: [/*]
2018-04-14 15:42:17.829  INFO 5926 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'httpPutFormContentFilter' to: [/*]
2018-04-14 15:42:17.830  INFO 5926 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'requestContextFilter' to: [/*]
2018-04-14 15:42:18.471  INFO 5926 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking
for @ControllerAdvice:
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@bebdb06: startup date
[Sat Apr 14 15:42:10 EDT 2018]; root of context hierarchy
2018-04-14 15:42:18.653  INFO 5926 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/getAssets]}" onto public com.cisco.pxgrid.samples.ise.http.DeviceList
com.cisco.pxgrid.samples.ise.http.PublisherController.device()
2018-04-14 15:42:18.657  INFO 5926 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String,
java.lang.Object>>
org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2018-04-14 15:42:18.658  INFO 5926 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpServletReque
st,javax.servlet.http.HttpServletResponse)
2018-04-14 15:42:18.692  INFO 5926 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/webjars/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-14 15:42:18.692  INFO 5926 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-14 15:42:18.762  INFO 5926 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/**/favicon.ico] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
```

```
2018-04-14 15:42:19.129  INFO 5926 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        :
Registering beans for JMX exposure on startup
2018-04-14 15:42:19.434  INFO 5926 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat
started on port(s): 8080 (http)
2018-04-14 15:42:19.446  INFO 5926 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : Started
CustomPublisher in 11.951 seconds (JVM running for 18.438)
------- properties -------
  hostnames=ise24fc3.lab10.com
  username=IOT1
  password=null
  groups=Session
  description=null
  keystoreFilename=Johns-Macbook-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12
  keystorePassword=Cisco123
  truststoreFilename=Johns-Macbook-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12
  truststorePassword=Cisco123
--------------------------
2018-04-14 15:42:21.139  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={}
2018-04-14 15:42:21.209  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"accountState":"ENABLED","version":"2.0.0.13"}
14-Apr-18 15:42:21.210 [main-1]: pxGrid controller version=2.0.0.13
2018-04-14 15:42:21.219  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"name":"com.cisco.endpoint.asset","properties":{"wsPubsubService":"com.cisco.ise.pubsub","restBaseUR
L":"http://raghdasa-lnv1:8080","assetTopic":"/topic/com.cisco.endpoint.asset"}}
2018-04-14 15:42:21.369  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={}
2018-04-14 15:42:21.375  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"name":"com.cisco.ise.pubsub"}
2018-04-14 15:42:21.386  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"services":[{"name":"com.cisco.ise.pubsub","nodeName":"ise-pubsub-
ise24fc3","properties":{"wsUrl":"wss://ise24fc3.lab10.com:8910/pxgrid/ise/pubsub"}}]}
14-Apr-18 15:42:21.386 [main-1]: wsUrl=wss://ise24fc3.lab10.com:8910/pxgrid/ise/pubsub
2018-04-14 15:42:21.392  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"peerNodeName":"ise-pubsub-ise24fc3"}
2018-04-14 15:42:21.429  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"secret":"tASXtHAbKDKbAODh"}
2018-04-14 15:42:22.692  INFO 5926 --- [     Grizzly(1)] c.c.p.s.i.h.StompPubsubClientEndpoint    : WS onOpen
2018-04-14 15:42:22.695  INFO 5926 --- [           main] c.c.p.s.i.h.StompPubsubClientEndpoint    : STOMP
CONNECT host=ise24fc3.lab10.com
press <enter> to start the publishing...2018-04-14 15:42:22.703  INFO 5926 --- [     Grizzly(2)]
c.c.p.s.i.h.StompPubsubClientEndpoint    : STOMP CONNECTED version=1.2

2018-04-14 15:42:29.801  INFO 5926 --- [           main] c.c.p.s.i.h.StompPubsubClientEndpoint    : STOMP
SEND topic=/topic/com.cisco.endpoint.asset
command=SEND, headers={'content-length':'593','destination':'/topic/com.cisco.endpoint.asset',},
content.length=593
14-Apr-18 15:42:29.802 [main-1]:
{"assetHwRevision":"5.6","assetProtocol":"CIP","assetConnectedLinks":[{"value":"3","key":"indattr2"},{"value"
:"Root","key":"assetGroup"},{"value":"1","key":"indattr3"}],"assetVendor":"Cisco
Systems","assetSwRevision":"4.6","assetCustomAttributes":[{"value":"3","key":"indattr2"},{"value":"Root","key
":"assetGroup"},{"value":"1","key":"indattr3"}],"assetProductId":"IE2000","assetSerialNumber":"1212121213243"
,"assetMacAddress":"b0:2c:27:93:fe:94","assetId":"215","assetIpAddress":"125.84.172.120","assetName":"Abjerga
ryn - 49","assetDeviceType":"EtherNet\/IP Node"}
```

Step 2 Select->**Context Visibility->Endpoint->Endpoint Classification**
Note that both the original asset endpoint and the updated one are now assigned to the IOT_Example1 profile.



Step 3 If you click on either of the MAC addresses, you will see the logical profile

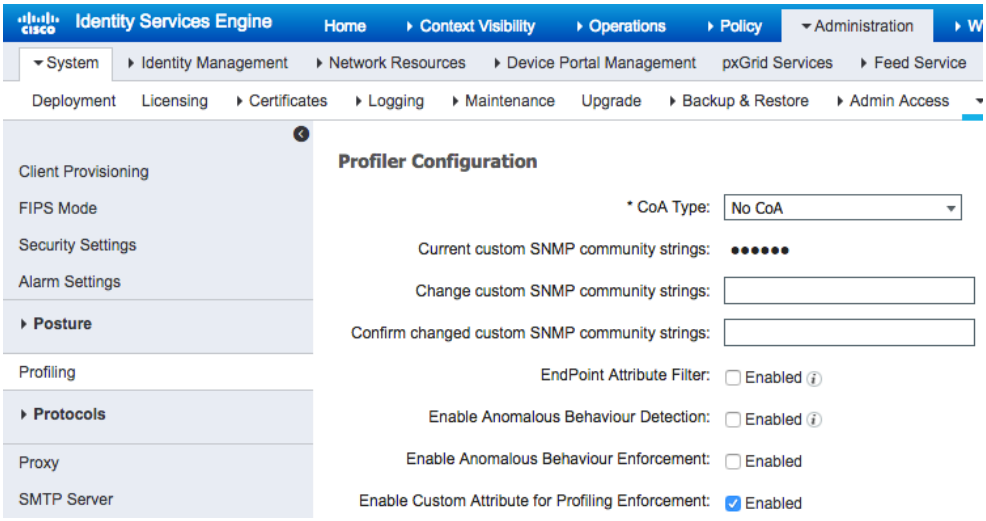


Creating Profiling Policy based on Asset Custom Attributes

A custom asset policy “CustomIOT” will be created. The custom attribute will be “assetgroup” and the value” will be “root”. The custom attribute endpoint setting must be enabled in ISE to use this feature.

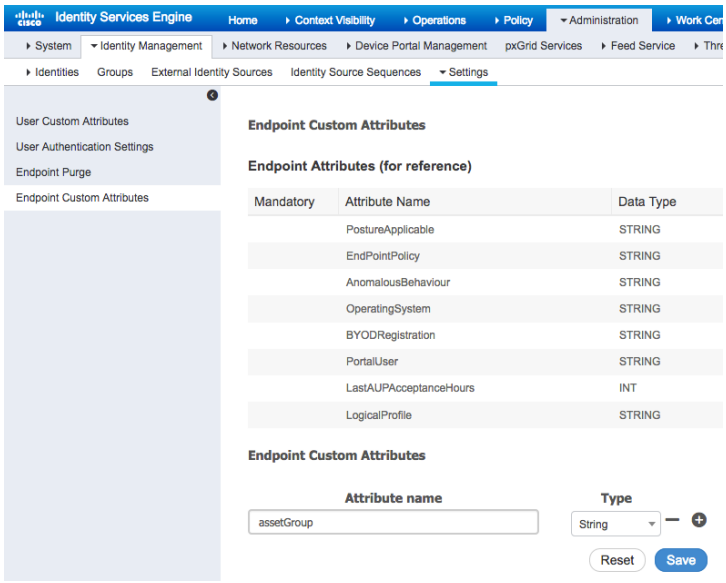
Enabling Custom Attribute Value

Step 1 Select **Administration->System->Settings->Profiling->enable->Enable**



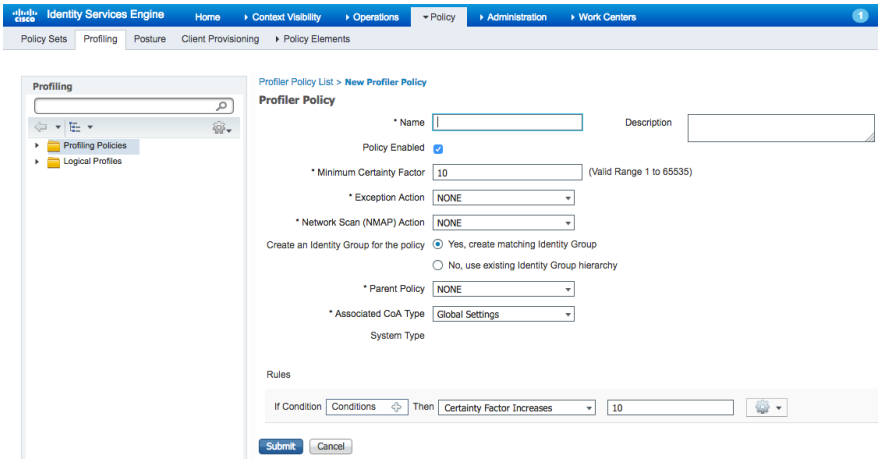
Step 2 Select **Save**

Step 3 Select->**Administration->Identity Management->Settings->Endpoint Custom Attributes**

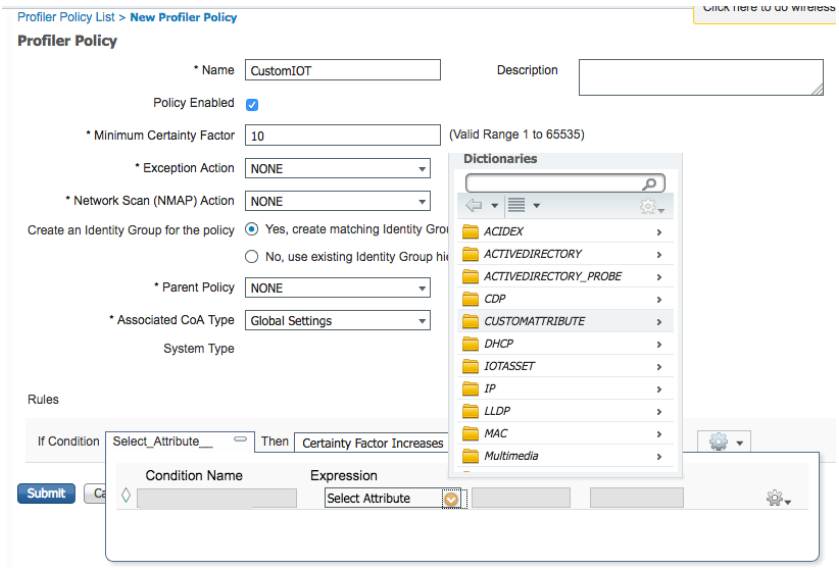


Step 4 Select **Save**

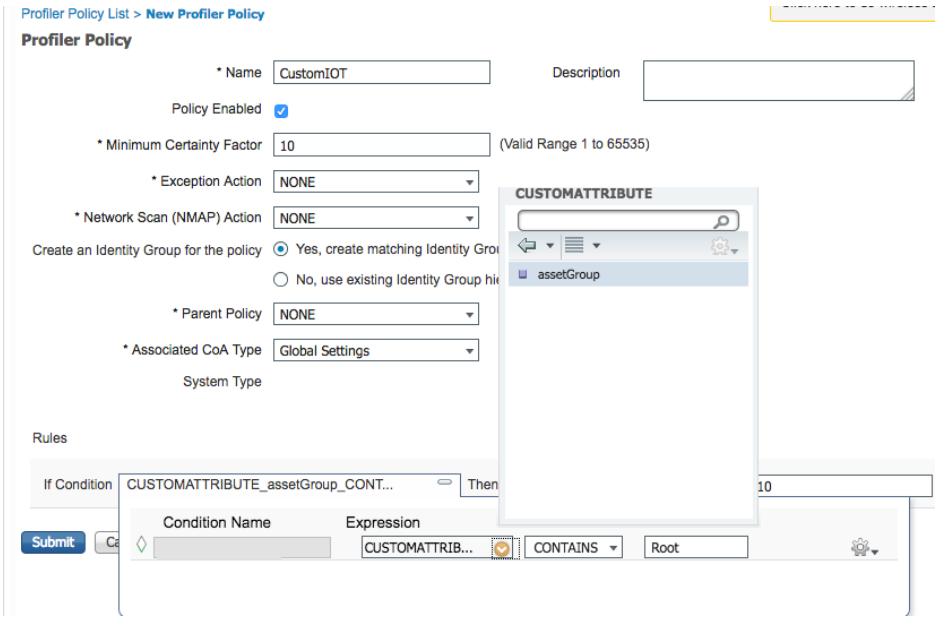
Step 5 Select **Policy-Profiling->Profiling Policies->Add**
You will see the following:



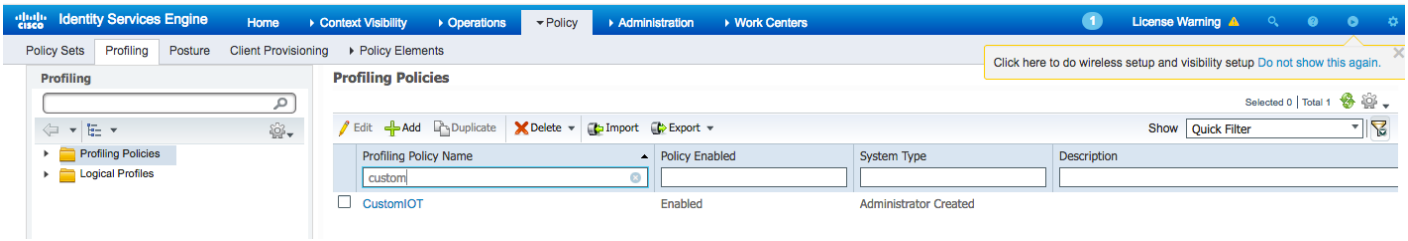
Step 6 Enter: **CustomIOT** for Name
Step 7 Under **Rules->If Condition->Create New Condition(Advanced Option)**
Step 8 Under **Expression->Select Attribute->CUSTOMATTRIBUTE**
Step 9 You should see:



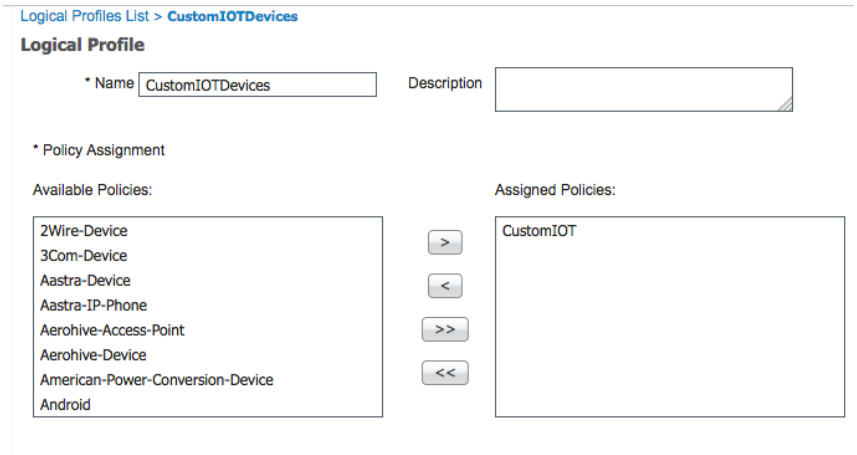
Step 10 Select **assetGroup:contains:Root**, enter, increase certainty factor to 100



Step 11 Select **Submit**
You should see the policy:

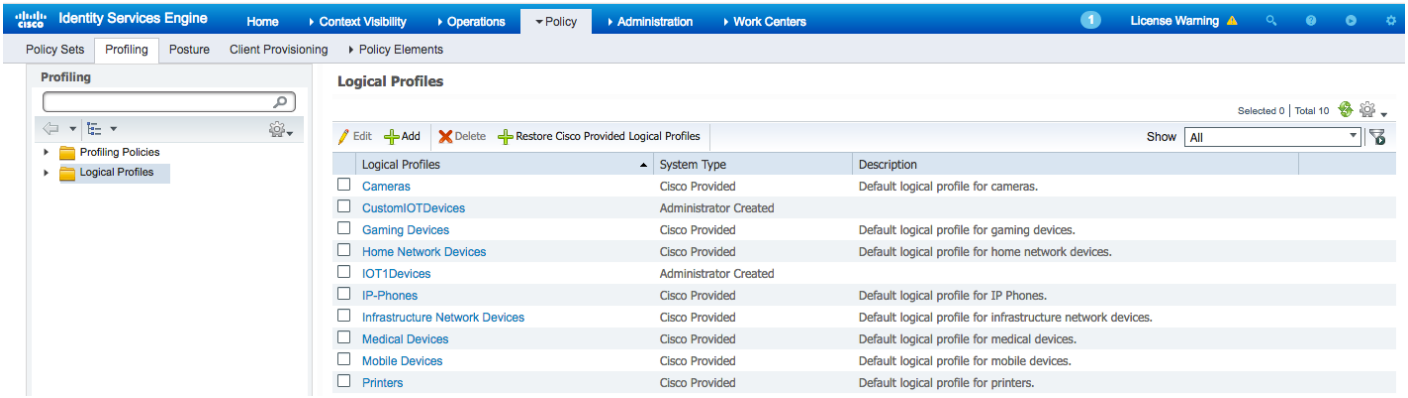


Step 12 Add CustomIOTDevices to Logical Profile
Select **Profiling->Logical Profiles->Add->Name: CustomIOTDevices->Policy Assignment->Available Policies->CustomIOT and move to Assigned Polices**



Step 13 Select **Submit**

Step 14 You should see the logical profile **CustomIOTDevices**



Step 15 Run script **./run_publisher38.sh**

```
Johns-MacBook-Pro-2:api_simulator jeppich$ ./run_publisher38.sh

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.6.RELEASE)

2018-04-15 00:16:17.854  INFO 7211 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : Starting
CustomPublisher v2.0.0-SNAPSHOT on johns-macbook-pro.lab10.com with PID 7211
(/Applications/api_partner_fc3/api_simulator/pxgrid-rest-ws-samples-2.0.0-SNAPSHOT.jar started by jeppich in
/Applications/api_partner_fc3/api_simulator)
2018-04-15 00:16:17.910  INFO 7211 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : No active
profile set, falling back to default profiles: default
2018-04-15 00:16:18.409  INFO 7211 --- [           main] ationConfigEmbeddedWebApplicationContext :
Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@bebdb06:
startup date [Sun Apr 15 00:16:18 EDT 2018]; root of context hierarchy
2018-04-15 00:16:25.288  INFO 7211 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat
initialized with port(s): 8080 (http)
2018-04-15 00:16:25.416  INFO 7211 --- [           main] o.apache.catalina.core.StandardService   : Starting
service [Tomcat]
2018-04-15 00:16:25.438  INFO 7211 --- [           main] org.apache.catalina.core.StandardEngine  : Starting
Servlet Engine: Apache Tomcat/8.5.16
2018-04-15 00:16:25.889  INFO 7211 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]       :
Initializing Spring embedded WebApplicationContext
2018-04-15 00:16:25.890  INFO 7211 --- [ost-startStop-1] o.s.web.context.ContextLoader            : Root
WebApplicationContext: initialization completed in 7506 ms
2018-04-15 00:16:26.113  INFO 7211 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean  : Mapping
servlet: 'dispatcherServlet' to [/]
2018-04-15 00:16:26.129  INFO 7211 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'characterEncodingFilter' to: [/*]
2018-04-15 00:16:26.130  INFO 7211 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'hiddenHttpMethodFilter' to: [/*]
2018-04-15 00:16:26.130  INFO 7211 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'httpPutFormContentFilter' to: [/*]
2018-04-15 00:16:26.130  INFO 7211 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'requestContextFilter' to: [/*]
2018-04-15 00:16:26.851  INFO 7211 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking
for @ControllerAdvice:
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@bebdb06: startup date
[Sun Apr 15 00:16:18 EDT 2018]; root of context hierarchy
2018-04-15 00:16:27.035  INFO 7211 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/getAssets]}" onto public com.cisco.pxgrid.samples.ise.http.DeviceList
com.cisco.pxgrid.samples.ise.http.PublisherController.device()
2018-04-15 00:16:27.039  INFO 7211 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String,
```

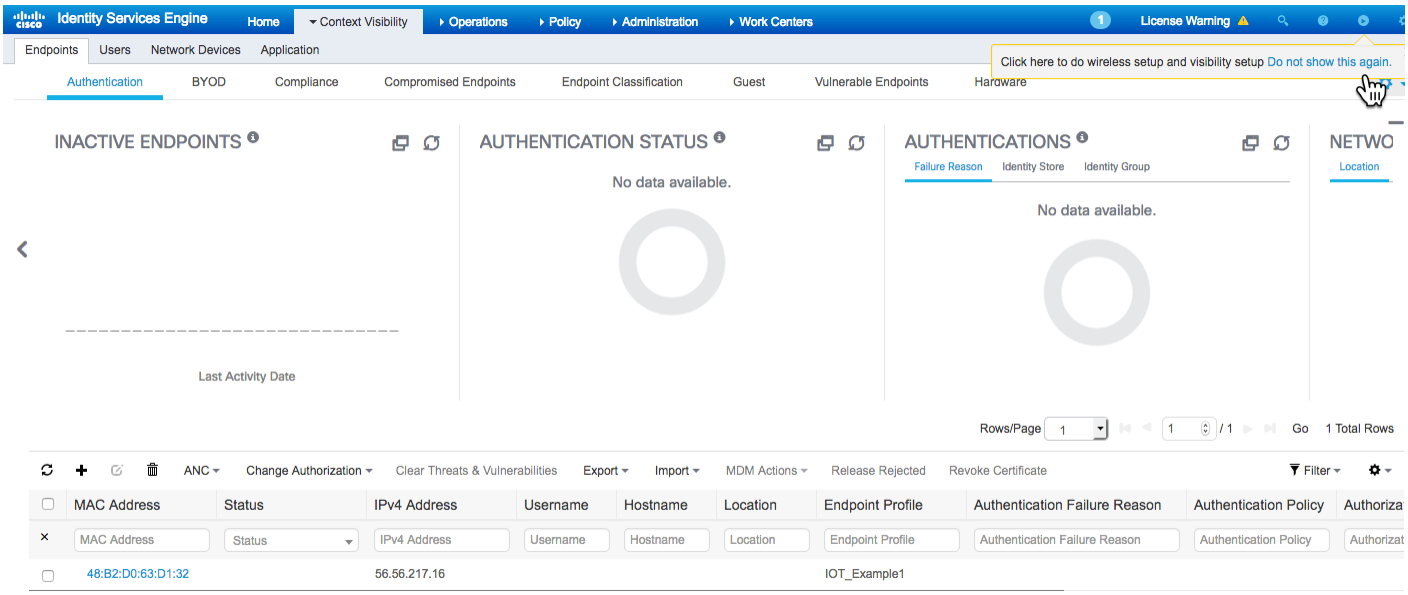
```
java.lang.Object>>
org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2018-04-15 00:16:27.039  INFO 7211 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpServletReque
st,javax.servlet.http.HttpServletResponse)
2018-04-15 00:16:27.080  INFO 7211 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/webjars/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-15 00:16:27.080  INFO 7211 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-15 00:16:27.153  INFO 7211 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/**/favicon.ico] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-15 00:16:27.508  INFO 7211 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        :
Registering beans for JMX exposure on startup
2018-04-15 00:16:27.847  INFO 7211 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat
started on port(s): 8080 (http)
2018-04-15 00:16:27.858  INFO 7211 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : Started
CustomPublisher in 13.385 seconds (JVM running for 21.849)
------- properties -------
  hostnames=ise24fc3.lab10.com
  username=IOT1
  password=null
  groups=Session
  description=null
  keystoreFilename=Johns-Macbook-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12
  keystorePassword=Cisco123
  truststoreFilename=Johns-Macbook-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12
  truststorePassword=Cisco123
--------------------------
2018-04-15 00:16:30.642  INFO 7211 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={}
2018-04-15 00:16:30.740  INFO 7211 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"accountState":"ENABLED","version":"2.0.0.13"}
15-Apr-18 00:16:30.741 [main-1]: pxGrid controller version=2.0.0.13
2018-04-15 00:16:30.750  INFO 7211 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"name":"com.cisco.endpoint.asset","properties":{"wsPubsubService":"com.cisco.ise.pubsub","restBaseUR
L":"http://raghdasa-lnv1:8080","assetTopic":"/topic/com.cisco.endpoint.asset"}}
2018-04-15 00:16:30.906  INFO 7211 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={}
2018-04-15 00:16:30.913  INFO 7211 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"name":"com.cisco.ise.pubsub"}
2018-04-15 00:16:30.924  INFO 7211 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"services":[{"name":"com.cisco.ise.pubsub","nodeName":"ise-pubsub-
ise24fc3","properties":{"wsUrl":"wss://ise24fc3.lab10.com:8910/pxgrid/ise/pubsub"}}]}
15-Apr-18 00:16:30.924 [main-1]: wsUrl=wss://ise24fc3.lab10.com:8910/pxgrid/ise/pubsub
2018-04-15 00:16:30.929  INFO 7211 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"peerNodeName":"ise-pubsub-ise24fc3"}
2018-04-15 00:16:30.955  INFO 7211 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"secret":"tASXtHAbKDKbAODh"}
2018-04-15 00:16:33.757  INFO 7211 --- [     Grizzly(1)] c.c.p.s.i.h.StompPubsubClientEndpoint    : WS onOpen
2018-04-15 00:16:33.760  INFO 7211 --- [           main] c.c.p.s.i.h.StompPubsubClientEndpoint    : STOMP
CONNECT host=ise24fc3.lab10.com
press <enter> to start the publishing...2018-04-15 00:16:33.768  INFO 7211 --- [     Grizzly(2)]
c.c.p.s.i.h.StompPubsubClientEndpoint    : STOMP CONNECTED version=1.2
```

Step 16 Press **Enter**

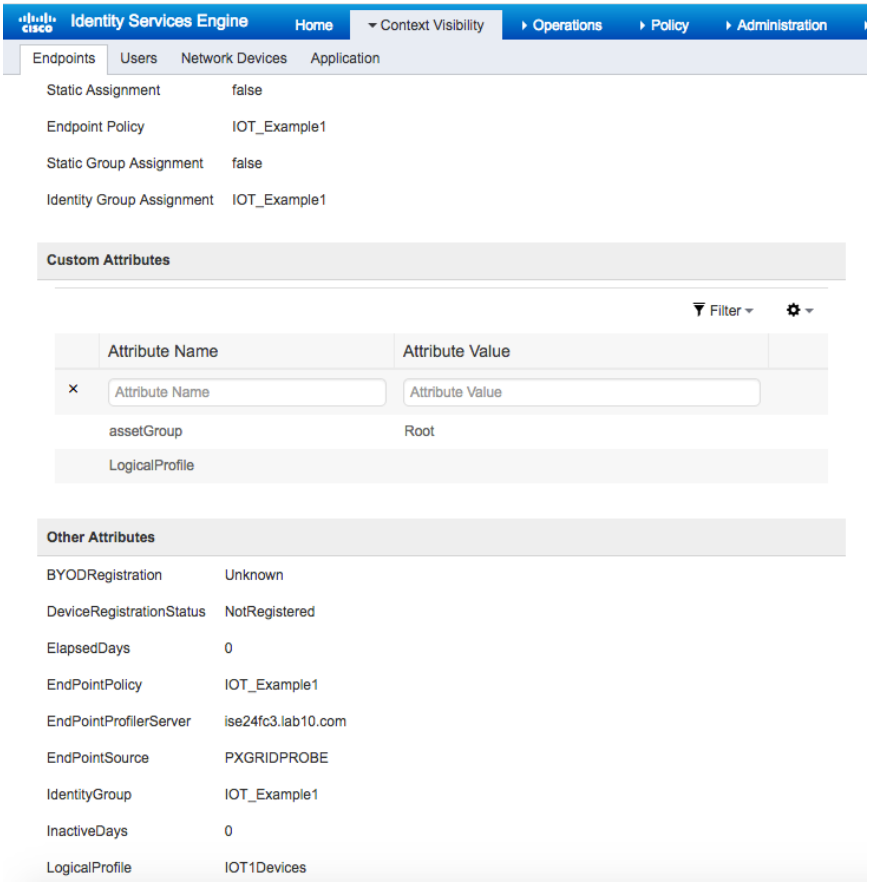
```
ClientEndpoint    : STOMP SEND topic=/topic/com.cisco.endpoint.asset
```

```
command=SEND, headers={'content-length':'591','destination':'/topic/com.cisco.endpoint.asset',},
content.length=591
15-Apr-18 00:19:00.175 [main-1]:
{"assetHwRevision":"5.6","assetProtocol":"CIP","assetConnectedLinks":[{"value":"3","key":"indattr2"},{"value"
:"Root","key":"assetGroup"},{"value":"1","key":"indattr3"}],"assetVendor":"Cisco
Systems","assetSwRevision":"4.6","assetCustomAttributes":[{"value":"3","key":"indattr2"},{"value":"Root","key
":"assetGroup"},{"value":"1","key":"indattr3"}],"assetProductId":"IE2000","assetSerialNumber":"1212121213243"
,"assetMacAddress":"48:b2:d0:63:d1:32","assetId":"260","assetIpAddress":"56.56.217.16","assetName":"Abjergary
n - 47","assetDeviceType":"EtherNet\/IP Node"}
```

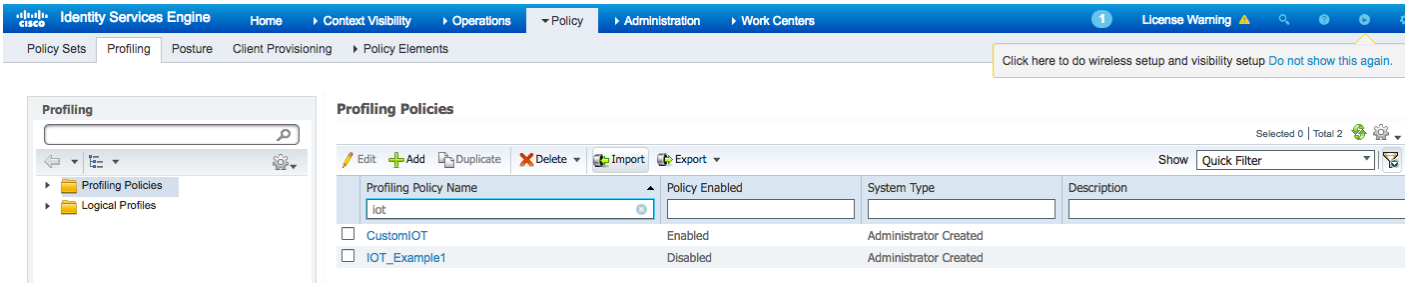

Step 17 Select **Context Visibility->Endpoints-Authentication or Endpoint Classification**
You will see the asset device



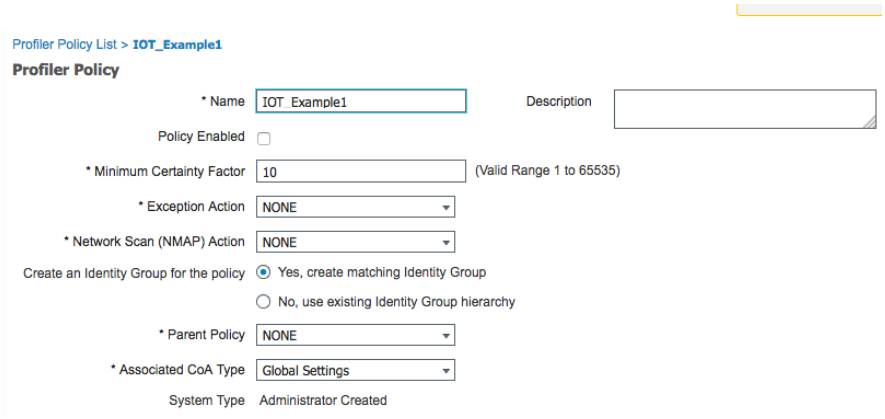
Step 18 Select **MAC address** of the endpoint devices, then **Attributes**
You will see "**AssetGroup**" under "**Attribute Name**" and "**Root**" under "**Attribute Value**". Note the Logical Profile still shows IOT_Example1.



Step 19 Select **Policy->Profiling->Profiling Policies->Quick filter->"iot"**
You will see both profiling policies

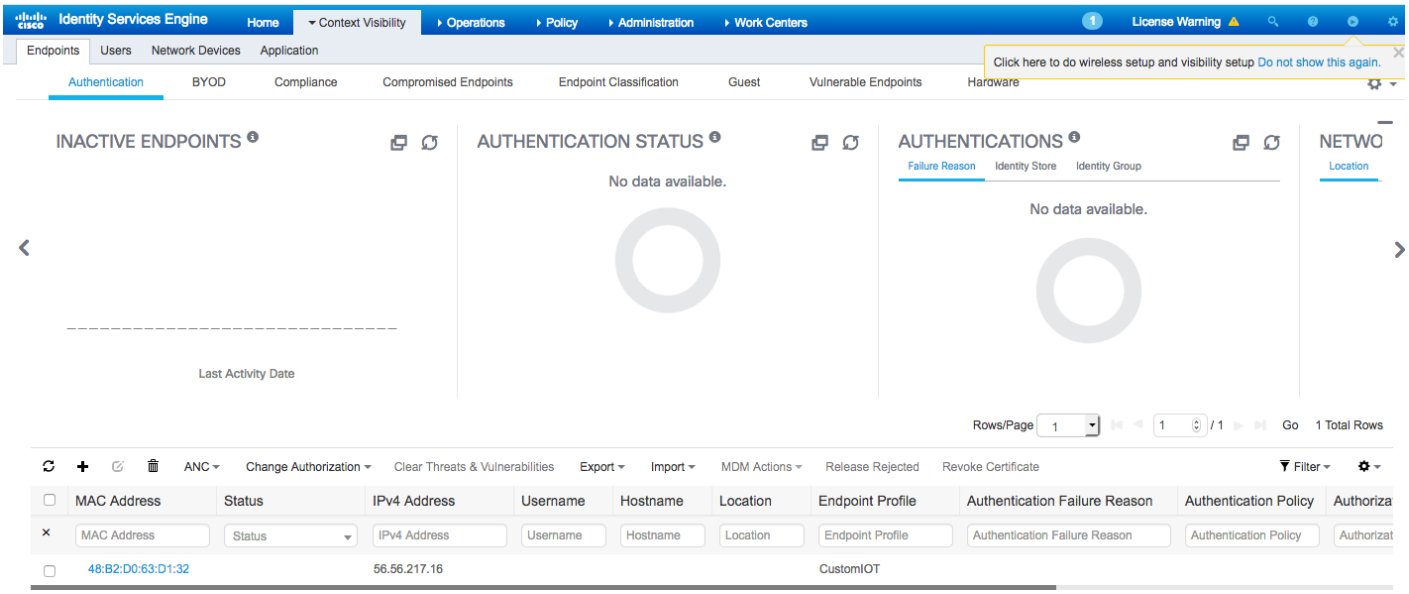


Step 20 Select->**IOT_Example1->uncheck** policy enabled



Step 21 Select **Save**

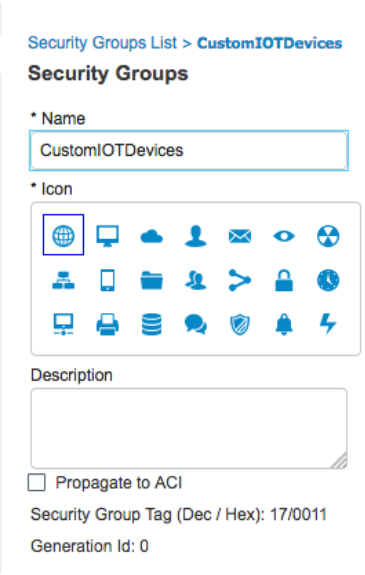
Step 22 Select-> **Context Visibility->Endpoints or Endpoint Classification**
You will now see the proper endpoint profile and logical profile.



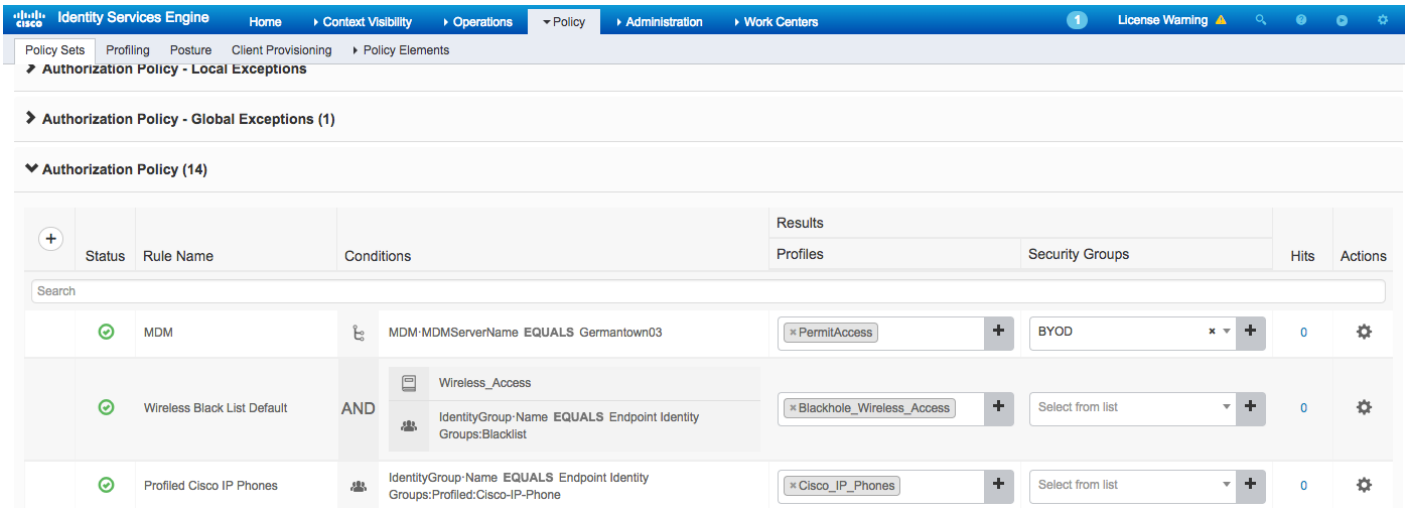
Creating Authorization Policy Based on Asset's Logical Profile for Custom Attributes

An authorization policy will be created that determines the asset's network access. A Security Group Tag (SGT) will also be added to the Authorization Policy provided label to classify network traffic if Cisco's TrustSec Solution is used.

Step 1 Create IOT device SGT which will be used in the Authorization Policy
Select **WorkCenters->Trustsec->Components->Add->CustomIOTDevices**

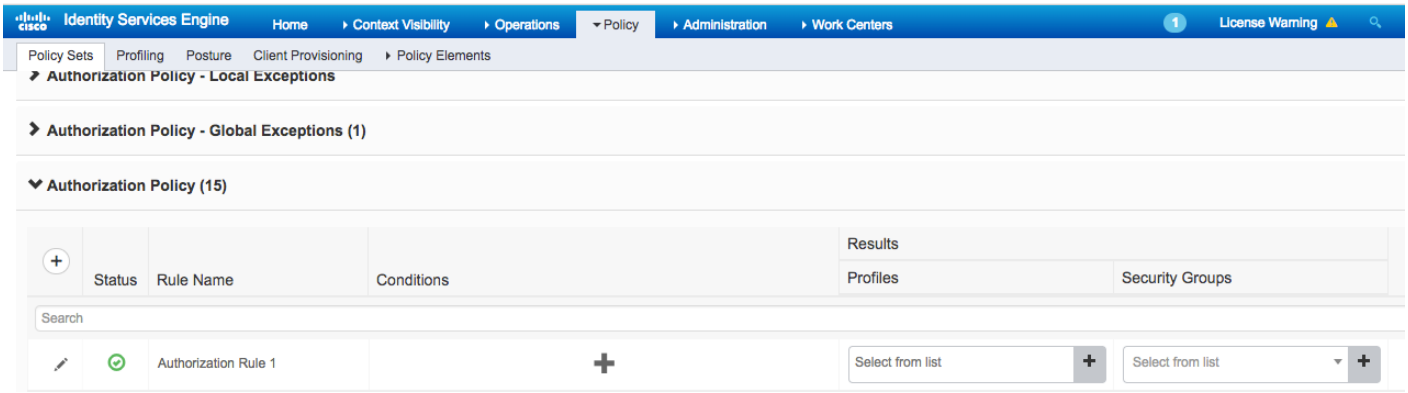


Step 2 Select **Save**
Step 3 Create ISE Authorization Profile Policy
Step 4 Select **Policy->Policy Sets->View ">"->Authorization Policy**
Step 5 You should see:



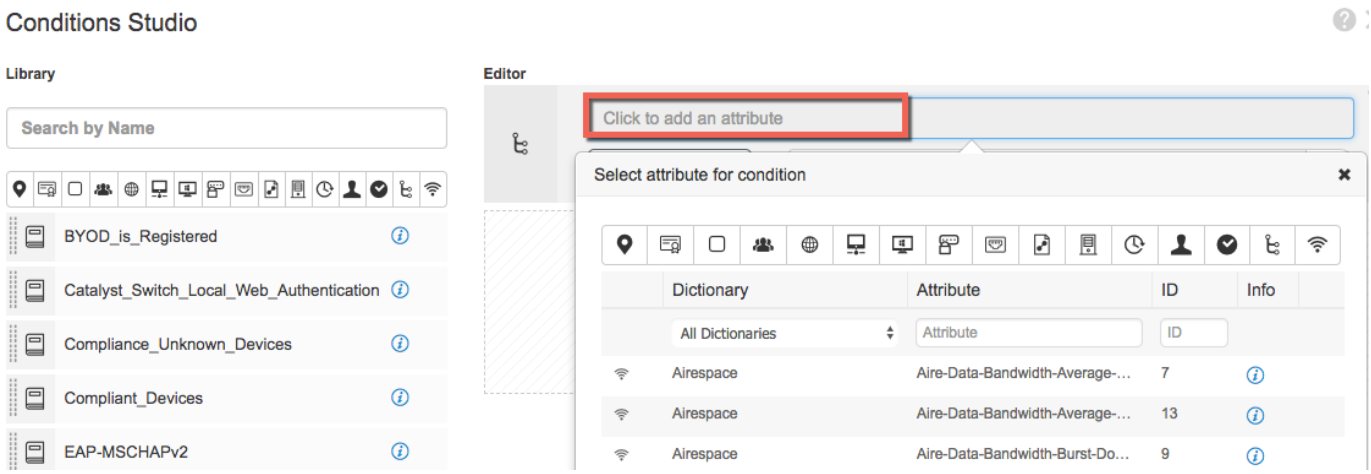
Step 6 Under **Actions**, **Click on "gear"** and "**Insert new rule above**

Step 7 You should see the following:

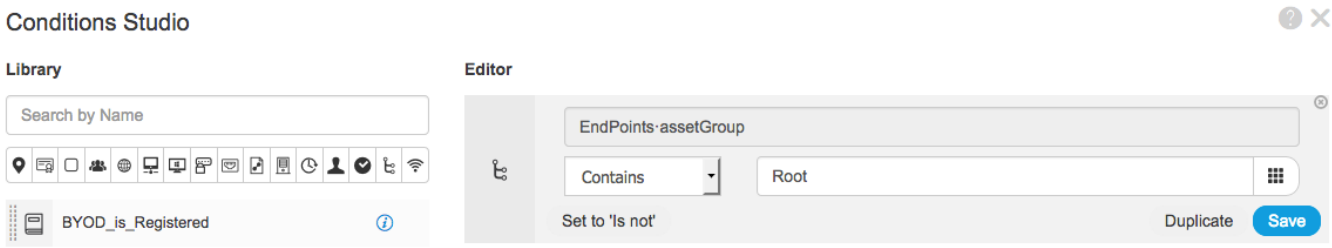


Step 8 Name Authorization Rule 1 to **CustomIOTDevices**, **Under Condition**, select **"+"**

Step 9 Click on **Attribute** , you should see the following Dictionary Attributes



Step 10 Select **Endpoints->assetGroup->Contains->Root**

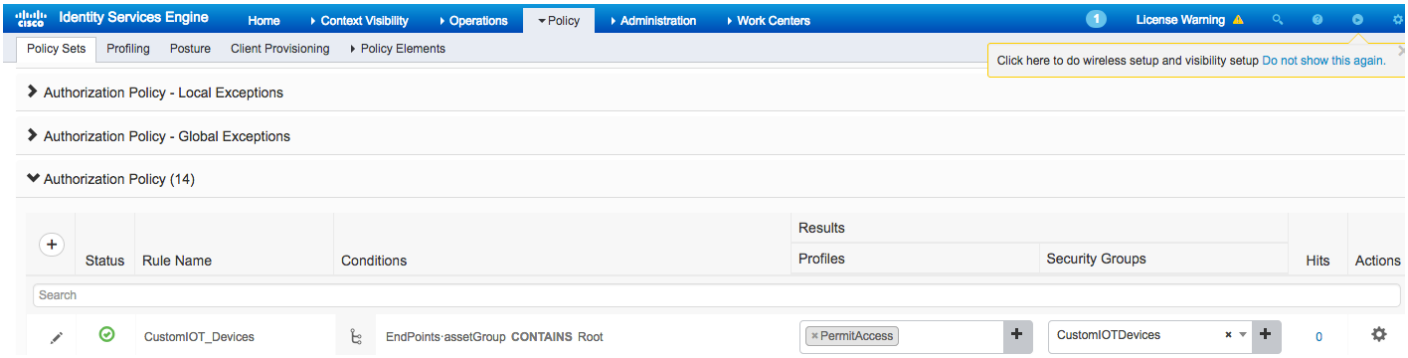


Step 11 Select **Use**

Step 12 Under **Profiles**, select **Permit Access**

Step 13 Under **Security Groups**, select **CustomIOTDevices**

You should see:



Step 14 Select **Save**

Verifying Asset as Defined by the Logical Profile for Custom Attributes

The custom asset policy is now assigned to the logical profile. This is verified in the Context Visibility Endpoint Classification screen.

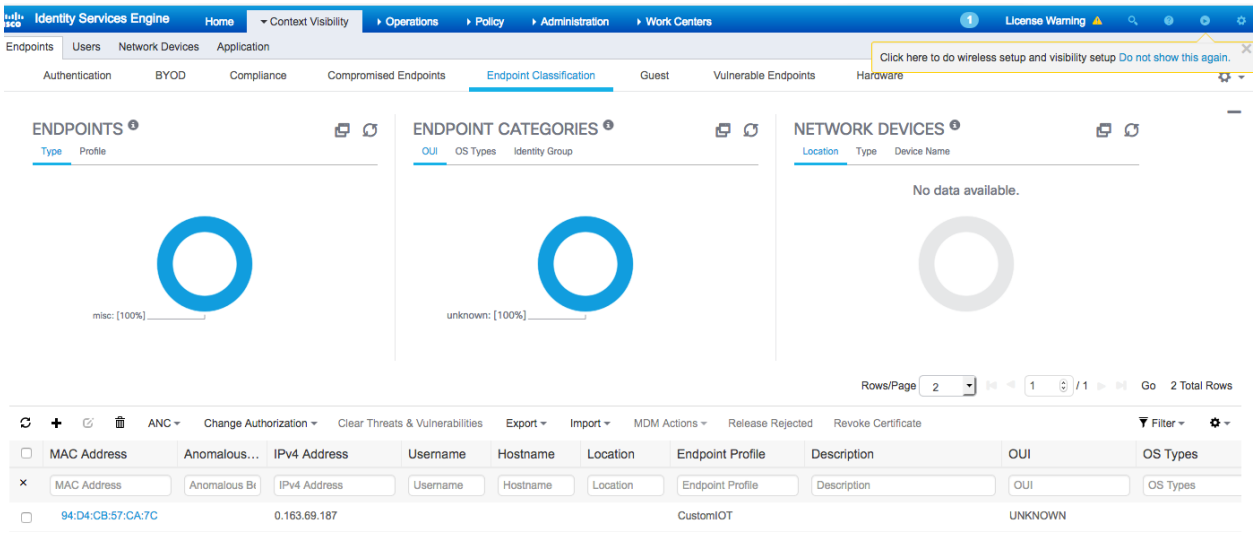
Step 1 Run script again, type: **./run_publisher38.sh**

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.6.RELEASE)

2018-04-14 15:42:10.009  INFO 5926 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : Starting
CustomPublisher v2.0.0-SNAPSHOT on johns-macbook-pro.lab10.com with PID 5926
(/Applications/api_partner_fc3/api_simulator/pxgrid-rest-ws-samples-2.0.0-SNAPSHOT.jar started by jeppich in
/Applications/api_partner_fc3/api_simulator)
2018-04-14 15:42:10.042  INFO 5926 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : No active
profile set, falling back to default profiles: default
2018-04-14 15:42:10.508  INFO 5926 --- [           main] ationConfigEmbeddedWebApplicationContext :
Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@bebdb06:
startup date [Sat Apr 14 15:42:10 EDT 2018]; root of context hierarchy
2018-04-14 15:42:17.010  INFO 5926 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat
initialized with port(s): 8080 (http)
2018-04-14 15:42:17.136  INFO 5926 --- [           main] o.apache.catalina.core.StandardService   : Starting
service [Tomcat]
2018-04-14 15:42:17.158  INFO 5926 --- [           main] org.apache.catalina.core.StandardEngine  : Starting
Servlet Engine: Apache Tomcat/8.5.16
2018-04-14 15:42:17.587  INFO 5926 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]       :
Initializing Spring embedded WebApplicationContext
2018-04-14 15:42:17.588  INFO 5926 --- [ost-startStop-1] o.s.web.context.ContextLoader            : Root
WebApplicationContext: initialization completed in 7105 ms
2018-04-14 15:42:17.813  INFO 5926 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean  : Mapping
servlet: 'dispatcherServlet' to [/]
2018-04-14 15:42:17.828  INFO 5926 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'characterEncodingFilter' to: [/*]
2018-04-14 15:42:17.829  INFO 5926 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'hiddenHttpMethodFilter' to: [/*]
```

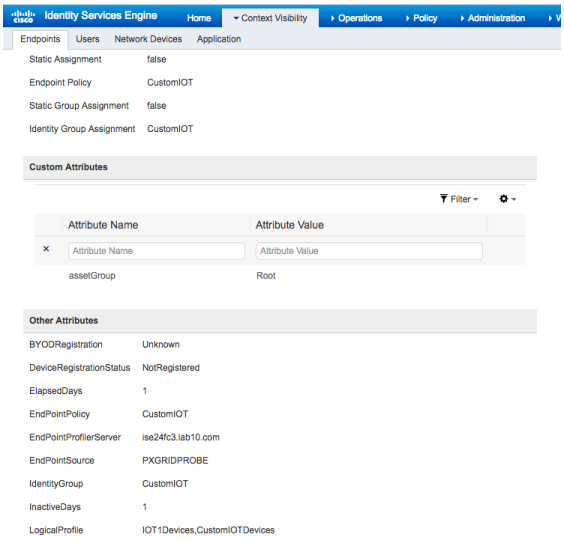
```
2018-04-14 15:42:17.829  INFO 5926 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'httpPutFormContentFilter' to: [/*]
2018-04-14 15:42:17.830  INFO 5926 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping
filter: 'requestContextFilter' to: [/*]
2018-04-14 15:42:18.471  INFO 5926 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking
for @ControllerAdvice:
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@bebdb06: startup date
[Sat Apr 14 15:42:10 EDT 2018]; root of context hierarchy
2018-04-14 15:42:18.653  INFO 5926 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/getAssets]}" onto public com.cisco.pxgrid.samples.ise.http.DeviceList
com.cisco.pxgrid.samples.ise.http.PublisherController.device()
2018-04-14 15:42:18.657  INFO 5926 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String,
java.lang.Object>>
org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2018-04-14 15:42:18.658  INFO 5926 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped
"{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpServletReque
st,javax.servlet.http.HttpServletResponse)
2018-04-14 15:42:18.692  INFO 5926 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/webjars/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-14 15:42:18.692  INFO 5926 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-14 15:42:18.762  INFO 5926 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped
URL path [/**/favicon.ico] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-04-14 15:42:19.129  INFO 5926 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        :
Registering beans for JMX exposure on startup
2018-04-14 15:42:19.434  INFO 5926 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat
started on port(s): 8080 (http)
2018-04-14 15:42:19.446  INFO 5926 --- [           main] c.c.p.samples.ise.http.CustomPublisher   : Started
CustomPublisher in 11.951 seconds (JVM running for 18.438)
------- properties -------
  hostnames=ise24fc3.lab10.com
  username=IOT1
  password=null
  groups=Session
  description=null
  keystoreFilename=Johns-Macbook-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12
  keystorePassword=Cisco123
  truststoreFilename=Johns-Macbook-Pro.lab10.com_Johns-Macbook-Pro.lab10.com.p12
  truststorePassword=Cisco123
--------------------------
2018-04-14 15:42:21.139  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={}
2018-04-14 15:42:21.209  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"accountState":"ENABLED","version":"2.0.0.13"}
14-Apr-18 15:42:21.210 [main-1]: pxGrid controller version=2.0.0.13
2018-04-14 15:42:21.219  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"name":"com.cisco.endpoint.asset","properties":{"wsPubsubService":"com.cisco.ise.pubsub","restBaseUR
L":"http://raghdasa-lnv1:8080","assetTopic":"/topic/com.cisco.endpoint.asset"}}
2018-04-14 15:42:21.369  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={}
2018-04-14 15:42:21.375  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"name":"com.cisco.ise.pubsub"}
2018-04-14 15:42:21.386  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"services":[{"name":"com.cisco.ise.pubsub","nodeName":"ise-pubsub-
ise24fc3","properties":{"wsUrl":"wss://ise24fc3.lab10.com:8910/pxgrid/ise/pubsub"}}]}
14-Apr-18 15:42:21.386 [main-1]: wsUrl=wss://ise24fc3.lab10.com:8910/pxgrid/ise/pubsub
2018-04-14 15:42:21.392  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Request={"peerNodeName":"ise-pubsub-ise24fc3"}
2018-04-14 15:42:21.429  INFO 5926 --- [           main] c.c.p.samples.ise.http.PxgridControl     :
Response={"secret":"tASXtHAbKDKbAODh"}
2018-04-14 15:42:22.692  INFO 5926 --- [     Grizzly(1)] c.c.p.s.i.h.StompPubsubClientEndpoint    : WS onOpen
2018-04-14 15:42:22.695  INFO 5926 --- [           main] c.c.p.s.i.h.StompPubsubClientEndpoint    : STOMP
CONNECT host=ise24fc3.lab10.com
press <enter> to start the publishing...2018-04-14 15:42:22.703  INFO 5926 --- [     Grizzly(2)]
c.c.p.s.i.h.StompPubsubClientEndpoint    : STOMP CONNECTED version=1.2

2018-04-14 15:42:29.801  INFO 5926 --- [           main] c.c.p.s.i.h.StompPubsubClientEndpoint    : STOMP
SEND topic=/topic/com.cisco.endpoint.asset
```

```
command=SEND, headers={'content-length':'593','destination':'/topic/com.cisco.endpoint.asset',},
content.length=593
14-Apr-18 15:42:29.802 [main-1]:
{"assetHwRevision":"5.6","assetProtocol":"CIP","assetConnectedLinks":[{"value":"3","key":"indattr2"},{"value"
:"Root","key":"assetGroup"},{"value":"1","key":"indattr3"}],"assetVendor":"Cisco
Systems","assetSwRevision":"4.6","assetCustomAttributes":[{"value":"3","key":"indattr2"},{"value":"Root","key
":"assetGroup"},{"value":"1","key":"indattr3"}],"assetProductId":"IE2000","assetSerialNumber":"1212121213243"
,"assetMacAddress":"b0:2c:27:93:fe:94","assetId":"215","assetIpAddress":"125.84.172.120","assetName":"Abjerga
ryn - 49","assetDeviceType":"EtherNet\/IP Node"}
```


Step 2 Select->**Context Visibility->Endpoint->Endpoint Classification**
Note that the client endpoint is now assigned to the CustomIOT profile.



Step 3 If you click on the MAC address, you will see the logical profile of CustomIOT devices. You may also see IOTDevices if you did not remove the CutomIOT device policy from IOT devices logical profile.



Editing Script Values

The API_Simulator CustomPublisher.java code needs to be modified to modify the asset values is the script. In this example, Maven is used.

Step 1 Copy the api_simulator/pxgrid-rest-ws folder over to your maven repository.

```
/Users/jeppich/maven/repo/pxgrid-rest-ws
```

Step 2 Go to the java folder, and see the pom.xml file and the src folder

```
johns-macbook-pro:pxgrid-rest-ws jeppich$ cd java
johns-macbook-pro:java jeppich$ ls
pom.xml                 src
run_publisher.sh        target
```

Step 3 Go to the http folder

```
/Users/jeppich/maven/repo/pxgrid-rest-ws/java/src/main/java/com/cisco/pxgrid/samples/ise/http
```

Step 4 You will see the sample code

```
Console.java                    SampleHelper.java
CustomPublisher.java            SessionQueryAll.java
CustomSubscriber.java           SessionQueryByIP.java
DeviceList.java                 SessionSubscribe.java
Devices.java                    StompFrame.java
PublisherController.java        StompPubsubClientEndpoint.java
PxgridControl.java              StompSubscription.java
SampleConfiguration.java
```

Step 5 Edit the CustomPublisher.java file, to change any of the values, keep the prefix name (i.e.) **object.put(prefix+"Vendor",** and change the value name (i.e. "**Cisco Systems**"); in the section of the code below. The (prefix + ".") values cannot be changed as they represent the values if the IOTASSETS dictionary.

```
package com.cisco.pxgrid.samples.ise.http;

import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Map;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.SSLSession;
import java.util.Random;
import org.glassfish.tyrus.client.ClientManager;
import org.glassfish.tyrus.client.ClientProperties;
import org.glassfish.tyrus.client.SslEngineConfigurator;
import org.glassfish.tyrus.client.auth.Credentials;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import java.util.concurrent.ThreadLocalRandom;
```

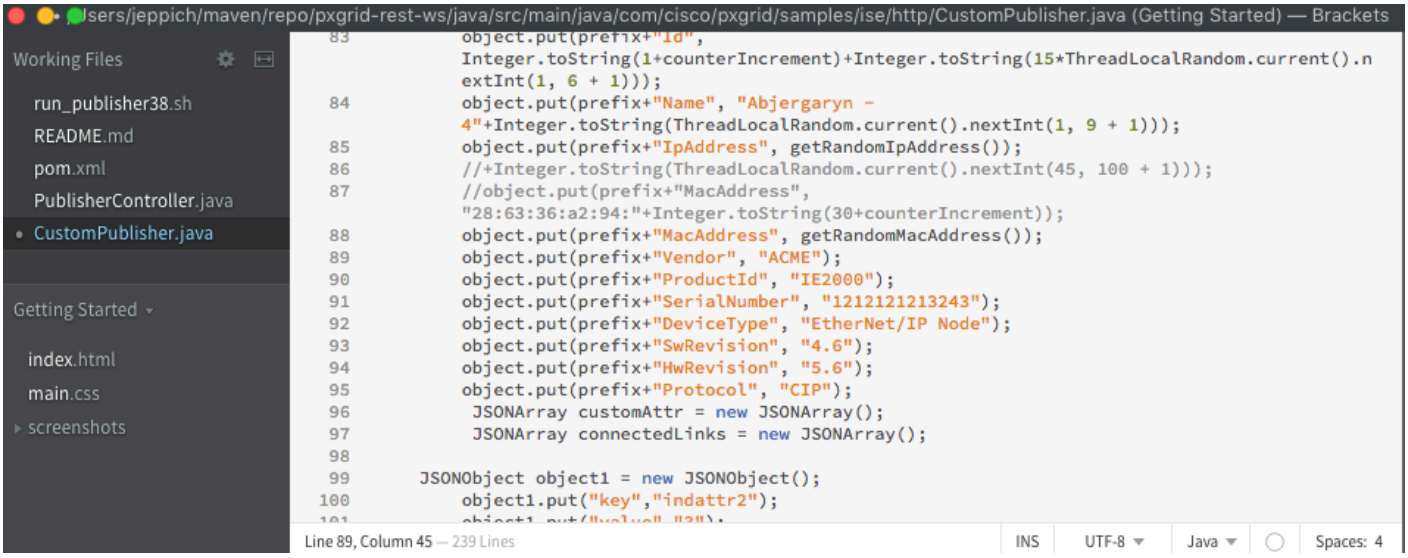
```
import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 * Sample creation of a Dynamic Service by a client, also publishes. Another Client may subscribe to this
service and receive
 * notifications when this service is published to.
 *
 * USE CASE for Dynamic Services: My company that requires all devices that connect to our network to be
specially authenticated so they need
 * more information. Using pxGrid dynamic services, I can set up a service that broadcasts the requirements
to all devices that connect and tell
 * them to send this information over to allow for authentication.
 *
 * @author anirvenk
 */

@SpringBootApplication
public class CustomPublisher {
        private static final String SERVICE_NAME = "com.cisco.endpoint.asset";
        private static final String TOPIC_PATH = "/topic/com.cisco.endpoint.asset";
        private static final String PUBSUBSERVICE = "com.cisco.ise.pubsub";
        public static int counter = 0;
        private static String prefix = "asset";
        private static int counterIncrement = 1;

        /**
         * Static class that is used to create a sample Service object.
         */
        public static Service createService() {
                Service service = new Service();
                service.setName(SERVICE_NAME);
                service.setNodeName("dynamic capability");
                Map<String, String> properties = new HashMap<String, String>();
                properties.put("wsPubsubService", PUBSUBSERVICE);
                properties.put("assetTopic", TOPIC_PATH);
                properties.put("restBaseURL", "http://raghdasa-lnv1:8080");
                service.setProperties(properties);
                return service;
        }
        public static String getRandomIpAddress(){
                Random r = new Random();
                return r.nextInt(256) + "." + r.nextInt(256) + "." + r.nextInt(256) + "." + r.nextInt(256);
        }
    public static String getRandomMacAddress(){
        Random rand = new Random();
        byte[] macAddr = new byte[6];
        rand.nextBytes(macAddr);

        macAddr[0] = (byte)(macAddr[0] & (byte)254);  //zeroing last 2 bytes to make it unicast and locally
adminstrated

        StringBuilder sb = new StringBuilder(18);
        for(byte b : macAddr){

                if(sb.length() > 0)
                sb.append(":");

                        sb.append(String.format("%02x", b));
                }


        return sb.toString();
    }
        public static JSONObject createJsonObject() {
                JSONObject object = new JSONObject();
                //change prefix to asset later
                object.put(prefix+"Id",
Integer.toString(1+counterIncrement)+Integer.toString(15*ThreadLocalRandom.current().nextInt(1, 6 + 1)));
```

```
                object.put(prefix+"Name", "Abjergaryn -
4"+Integer.toString(ThreadLocalRandom.current().nextInt(1, 9 + 1)));
                object.put(prefix+"IpAddress", getRandomIpAddress());
                //+Integer.toString(ThreadLocalRandom.current().nextInt(45, 100 + 1)));
                //object.put(prefix+"MacAddress", "28:63:36:a2:94:"+Integer.toString(30+counterIncrement));
                object.put(prefix+"MacAddress", getRandomMacAddress());
                object.put(prefix+"Vendor", "Cisco Systems");
                object.put(prefix+"ProductId", "IE2000");
                object.put(prefix+"SerialNumber", "1212121213243");
                object.put(prefix+"DeviceType", "EtherNet/IP Node");
                object.put(prefix+"SwRevision", "4.6");
                object.put(prefix+"HwRevision", "5.6");
                object.put(prefix+"Protocol", "CIP");
                 JSONArray customAttr = new JSONArray();
                 JSONArray connectedLinks = new JSONArray();

    JSONObject object1 = new JSONObject();
                object1.put("key","indattr2");
                object1.put("value","3");
                customAttr.add(object1);
                 JSONObject object2 = new JSONObject();
                 object2.put("key","assetGroup");
                 object2.put("value","Root");
                 customAttr.add(object2);
                 JSONObject object3 = new JSONObject();
                 object3.put("key","indattr3");
          object3.put("value","1");
          customAttr.add(object3);
          connectedLinks.add(object1);
                 connectedLinks.add(object2);
                 connectedLinks.add(object3);
                object.put("assetCustomAttributes",customAttr);
                object.put("assetConnectedLinks",connectedLinks);
                counterIncrement++;
                return object;
        }

        public static void main(String[] args) throws Exception {
                // setting up the environment from passed in arguments
         SpringApplication.run(CustomPublisher.class, args);

                SampleConfiguration config = new SampleConfiguration();

                //creates PxGridControl object with the environment
                PxgridControl control = new PxgridControl(config);

                // AccountActivate
                while (control.accountActivate() != AccountState.ENABLED) {
                        Thread.sleep(45000);
                }
                Console.log("pxGrid controller version=" + control.getControllerVersion());

                //creating new service.
                Service service = createService();

                //registers the service that we created above
                control.registerService(service.getName(), service.getProperties());

                Service[] list_of_services;

                //below lookup should find main pxGrid server.
                list_of_services = control.lookupService(PUBSUBSERVICE);
                if (list_of_services.length == 0) {
                        Console.log("service isn't there");
                        return;
                }

                // takes the main pxGrid server node
                Service wsPubsubService = list_of_services[0];

                //get wsURL so we can get the URI later from it via REST query.
                String wsURL = wsPubsubService.getProperties().get("wsUrl");
                Console.log("wsUrl=" + wsURL);
```

```
                // pxGrid AccessSecret
                String secret = control.getAccessSecret(wsPubsubService.getNodeName());

                //setting up client manager which will use ssl connection for authentication.
                ClientManager client = ClientManager.createClient();
                SslEngineConfigurator sslEngineConfigurator = new
SslEngineConfigurator(config.getSSLContext());
                sslEngineConfigurator.setHostnameVerifier(new HostnameVerifier() {
                        @Override
                        public boolean verify(String hostname, SSLSession session) {
                                return true;
                        }
                });
                client.getProperties().put(ClientProperties.SSL_ENGINE_CONFIGURATOR, sslEngineConfigurator);
                client.getProperties().put(ClientProperties.CREDENTIALS,
                                new Credentials(config.getUserName(), secret.getBytes()));

                // WebSocket connect
                StompPubsubClientEndpoint endpoint = new StompPubsubClientEndpoint();

                //get URI, connect pxGrid client to the pxGrid server so that we can publish to dynamic
service
                URI uri = new URI(wsURL);

                javax.websocket.Session session = client.connectToServer(endpoint, uri);

                // STOMP connect
                endpoint.connect(uri.getHost());

                /*
                 * publishing to the dynamic service. This message "dynamic topic publish" will be received by
all subscribers to the service.
                 * only triggers when key is pressed. This is to make it so we can subscribe another client to
the service by running
                 * DynamicServiceSubscribe.java to see if it receives the published info.
                 */
                SampleHelper.prompt("press <enter> to start the publishing...");

                //for multi publishing
                //for(int i = 0; i < 20000; i++) {
                        JSONArray deviceArr = new JSONArray();
                        JSONObject device_object = new JSONObject();
                        JSONObject device = createJsonObject();
                        deviceArr.add(device);
                        device_object.put("asset", device);
                        device_object.put("opType","UPDATE");
                        byte[] array = device_object.toJSONString().getBytes(StandardCharsets.UTF_8);
                        endpoint.publish(TOPIC_PATH, array);
                        Console.log(device.toJSONString());
            //          Console.log(Integer.toString(i));
                //}

                /*JSONArray deviceArr = new JSONArray();
                JSONObject device_object = new JSONObject();

                JSONObject device1 = createJsonObject();
                JSONObject device2 = createJsonObject();
                JSONObject device3 = createJsonObject();
                JSONObject device4 = createJsonObject();

                device2.put(prefix+"SwRevision", "7.8");
                device3.put(prefix+"SwRevision", "3.3");
                device3.put(prefix+"HwRevision", "2.5");
                device4.put(prefix+"SwRevision", "3.5");
                deviceArr.add(device1);
                deviceArr.add(device2);
                deviceArr.add(device3);
                deviceArr.add(device4);


                device_object.put("asset", device1);
                device_object.put("opType","UPDATE");
```

```
                byte[] array = device_object.toJSONString().getBytes(StandardCharsets.UTF_8);
                endpoint.publish(TOPIC_PATH, array);
                */

                //SampleHelper.prompt("press <enter> to disconnect...");

                // STOMP disconnect
                //endpoint.disconnect("ID-123");
                // Wait for disconnect receipt
                //Thread.sleep(3000);

                //session.close();
        }
}
```

Step 6 Change the Vendor Name to "ACME" and Save the file



Step 7 Go to folder

```
johns-macbook-pro:java jeppich$ ls
pom.xml                 src
run_publisher.sh        target
johns-macbook-pro:java jeppich$
```

Step 8 Run mvn install to recompile

```
johns-macbook-pro:java jeppich$ mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] --------------< com.cisco.pxgrid:pxgrid-rest-ws-samples >---------------
[INFO] Building Publisher 2.0.0-SNAPSHOT
```

```
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ pxgrid-rest-ws-samples ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 40 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ pxgrid-rest-ws-samples ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 40 source files to /Users/jeppich/maven/repo/pxgrid-rest-ws/java/target/classes
[WARNING] /Users/jeppich/maven/repo/pxgrid-rest-
ws/java/src/main/java/com/cisco/pxgrid/samples/ise/http/CustomPublisher.java:
/Users/jeppich/maven/repo/pxgrid-rest-
ws/java/src/main/java/com/cisco/pxgrid/samples/ise/http/CustomPublisher.java uses unchecked or unsafe
operations.
[WARNING] /Users/jeppich/maven/repo/pxgrid-rest-
ws/java/src/main/java/com/cisco/pxgrid/samples/ise/http/CustomPublisher.java: Recompile with -Xlint:unchecked
for details.
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ pxgrid-rest-ws-samples ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/jeppich/maven/repo/pxgrid-rest-ws/java/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ pxgrid-rest-ws-samples ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.18.1:test (default-test) @ pxgrid-rest-ws-samples ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.6:jar (default-jar) @ pxgrid-rest-ws-samples ---
[INFO] Building jar: /Users/jeppich/maven/repo/pxgrid-rest-ws/java/target/pxgrid-rest-ws-samples-2.0.0-
SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:1.5.6.RELEASE:repackage (default) @ pxgrid-rest-ws-samples ---
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ pxgrid-rest-ws-samples ---
[INFO] Installing /Users/jeppich/maven/repo/pxgrid-rest-ws/java/target/pxgrid-rest-ws-samples-2.0.0-
SNAPSHOT.jar to /Users/jeppich/.m2/repository/com/cisco/pxgrid/pxgrid-rest-ws-samples/2.0.0-SNAPSHOT/pxgrid-
rest-ws-samples-2.0.0-SNAPSHOT.jar
[INFO] Installing /Users/jeppich/maven/repo/pxgrid-rest-ws/java/pom.xml to
/Users/jeppich/.m2/repository/com/cisco/pxgrid/pxgrid-rest-ws-samples/2.0.0-SNAPSHOT/pxgrid-rest-ws-samples-
2.0.0-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 02:20 min
[INFO] Finished at: 2018-04-21T16:13:14-04:00
[INFO] ------------------------------------------------------------------------
johns-macbook-pro:java jeppich$
```

Step 9 Replace the **/Applications/api_partner_fc3/api_simulator/snapshot.jar file** with the one that was just created. (**i.e. /Users/jeppich/.m2/repository/com/cisco/pxgrid/pxgrid-rest-ws-samples/2.0.0-SNAPSHOT**)

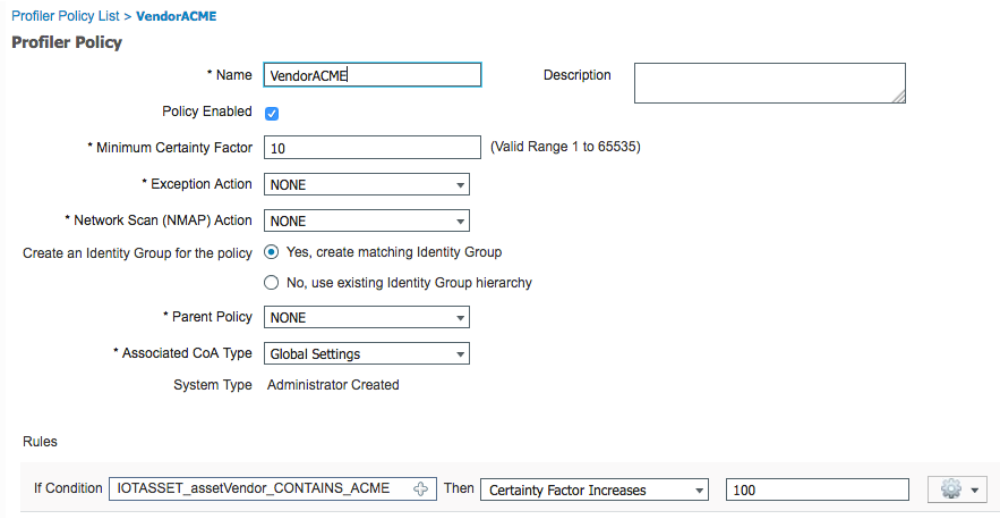
```
cp pxgrid-rest-ws-samples-2.0.0-SNAPSHOT.jar /Applications/api_partner_fc3/api_simulator
```

Step 10 Run the script, you should see “ACME” in the Asset Vendor

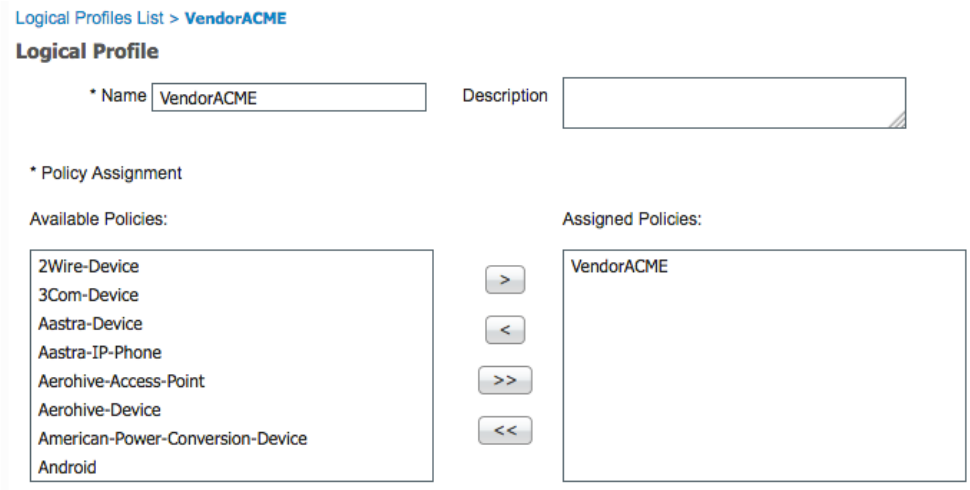
```
ClientEndpoint    : STOMP SEND topic=/topic/com.cisco.endpoint.asset
command=SEND, headers={'content-length':'581','destination':'/topic/com.cisco.endpoint.asset',},
content.length=581
```

```
21-Apr-18 16:35:37.330 [main-1]:
{"assetHwRevision":"5.6","assetProtocol":"CIP","assetConnectedLinks":[{"value":"3","key":"indattr2"},{"value"
:"Root","key":"assetGroup"},{"value":"1","key":"indattr3"}],"assetVendor":"ACME","assetSwRevision":"4.6","ass
etCustomAttributes":[{"value":"3","key":"indattr2"},{"value":"Root","key":"assetGroup"},{"value":"1","key":"i
ndattr3"}],"assetProductId":"IE2000","assetSerialNumber":"1212121213243","assetMacAddress":"00:66:cd:b6:17:d6
","assetId":"275","assetIpAddress":"115.86.71.7","assetName":"Abjergaryn -
41","assetDeviceType":"EtherNet\/IP Node"}
```


Step 11 You can now create a Profiling Policy based on IOTASSET assetvendor attribute using

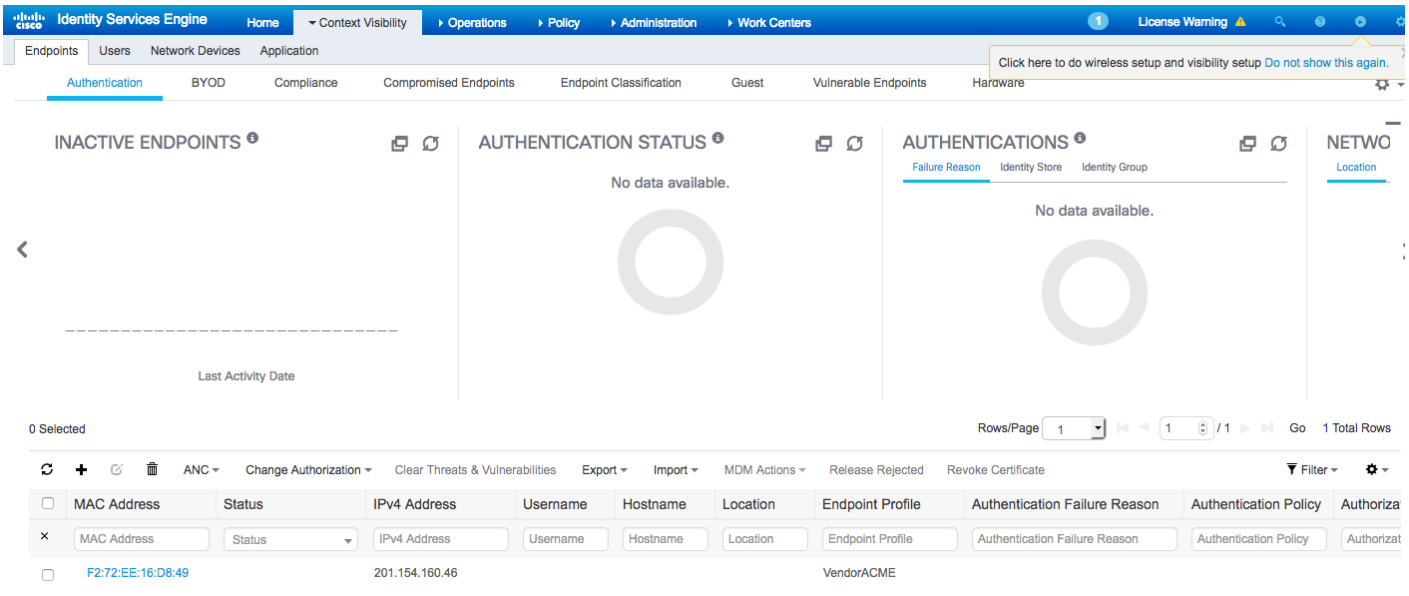


Step 12 You can also create a logical profile VendorACME



Step 13 Select **Context Visibility->Endpoints->Authentications,** you should see "VendorACME" for the EndpointProfile

Note: If you see, "CustomIOTDevices", disable "CustomIOTDevices" profiling policy for this exercise.



Editing/Adding Customer Values

Step 1 Copy the api_simulator/ pxgrid-rest-ws folder over to your maven repository

```
/Users/jeppich/maven/repo/pxgrid-rest-ws
```

Step 2 Go to the java folder, and see the pom.xml file and the src folder

```
johns-macbook-pro:pxgrid-rest-ws jeppich$ cd java
johns-macbook-pro:java jeppich$ ls
pom.xml                 src
run_publisher.sh        target
```

Step 3 Go to the http folder

```
/Users/jeppich/maven/repo/pxgrid-rest-ws/java/src/main/java/com/cisco/pxgrid/samples/ise/http
```

Step 4 You will see the sample code

```
Console.java                    SampleHelper.java
CustomPublisher.java            SessionQueryAll.java
CustomSubscriber.java           SessionQueryByIP.java
```

```
DeviceList.java                 SessionSubscribe.java
Devices.java                    StompFrame.java
PublisherController.java        StompPubsubClientEndpoint.java
PxgridControl.java              StompSubscription.java
SampleConfiguration.java
```

Step 5 Edit the CustomPublisher.java file, replace **object1.put ("key", "inadttr2")** with **object1.put ("key", "CVSS")"**

Step 6 Also replace **object1.put("value","3")** with **object1.put ("value","7")** in the highlighted fields below.

These fields will represent the custom attribute values of "key" and "value" that will be added to the ISE identity custom attribute screen and also when creating the CUSTOMATTRIBUTE profiling policy.

```
package com.cisco.pxgrid.samples.ise.http;

import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Map;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.SSLSession;
import java.util.Random;
import org.glassfish.tyrus.client.ClientManager;
import org.glassfish.tyrus.client.ClientProperties;
import org.glassfish.tyrus.client.SslEngineConfigurator;
import org.glassfish.tyrus.client.auth.Credentials;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import java.util.concurrent.ThreadLocalRandom;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 * Sample creation of a Dynamic Service by a client, also publishes. Another Client may subscribe to this
service and receive
 * notifications when this service is published to.
 *
 * USE CASE for Dynamic Services: My company that requires all devices that connect to our network to be
specially authenticated so they need
 * more information. Using pxGrid dynamic services, I can set up a service that broadcasts the requirements
to all devices that connect and tell
 * them to send this information over to allow for authentication.
 *
 * @author anirvenk
 */

@SpringBootApplication
public class CustomPublisher {
        private static final String SERVICE_NAME = "com.cisco.endpoint.asset";
        private static final String TOPIC_PATH = "/topic/com.cisco.endpoint.asset";
        private static final String PUBSUBSERVICE = "com.cisco.ise.pubsub";
        public static int counter = 0;
        private static String prefix = "asset";
        private static int counterIncrement = 1;

        /**
         * Static class that is used to create a sample Service object.
         */
        public static Service createService() {
                Service service = new Service();
                service.setName(SERVICE_NAME);
                service.setNodeName("dynamic capability");
                Map<String, String> properties = new HashMap<String, String>();
                properties.put("wsPubsubService", PUBSUBSERVICE);
                properties.put("assetTopic", TOPIC_PATH);
```

```
                properties.put("restBaseURL", "http://raghdasa-lnv1:8080");
                service.setProperties(properties);
                return service;
        }
        public static String getRandomIpAddress(){
                Random r = new Random();
                return r.nextInt(256) + "." + r.nextInt(256) + "." + r.nextInt(256) + "." + r.nextInt(256);
        }
    public static String getRandomMacAddress(){
        Random rand = new Random();
        byte[] macAddr = new byte[6];
        rand.nextBytes(macAddr);

        macAddr[0] = (byte)(macAddr[0] & (byte)254);  //zeroing last 2 bytes to make it unicast and locally
adminstrated

        StringBuilder sb = new StringBuilder(18);
        for(byte b : macAddr){

                if(sb.length() > 0)
                sb.append(":");

                        sb.append(String.format("%02x", b));
                }


        return sb.toString();
    }
        public static JSONObject createJsonObject() {
                JSONObject object = new JSONObject();
                //change prefix to asset later
                object.put(prefix+"Id",
Integer.toString(1+counterIncrement)+Integer.toString(15*ThreadLocalRandom.current().nextInt(1, 6 + 1)));
                object.put(prefix+"Name", "Abjergaryn -
4"+Integer.toString(ThreadLocalRandom.current().nextInt(1, 9 + 1)));
                object.put(prefix+"IpAddress", getRandomIpAddress());
                //+Integer.toString(ThreadLocalRandom.current().nextInt(45, 100 + 1)));
                //object.put(prefix+"MacAddress", "28:63:36:a2:94:"+Integer.toString(30+counterIncrement));
                object.put(prefix+"MacAddress", getRandomMacAddress());
                object.put(prefix+"Vendor", "Cisco Systems");
                object.put(prefix+"ProductId", "IE2000");
                object.put(prefix+"SerialNumber", "1212121213243");
                object.put(prefix+"DeviceType", "EtherNet/IP Node");
                object.put(prefix+"SwRevision", "4.6");
                object.put(prefix+"HwRevision", "5.6");
                object.put(prefix+"Protocol", "CIP");
                 JSONArray customAttr = new JSONArray();
                 JSONArray connectedLinks = new JSONArray();

    JSONObject object1 = new JSONObject();
                object1.put("key","CVSS");    //  original (object1.put("key", "indattr2" );
                object1.put("value","7");         //  original (object1.put("value", "3");
                customAttr.add(object1);
                 JSONObject object2 = new JSONObject();
                 object2.put("key","assetGroup");
                 object2.put("value","Root");
                 customAttr.add(object2);
                 JSONObject object3 = new JSONObject();
                 object3.put("key","indattr3");
          object3.put("value","1");
          customAttr.add(object3);
          connectedLinks.add(object1);
                 connectedLinks.add(object2);
                 connectedLinks.add(object3);
                object.put("assetCustomAttributes",customAttr);
                object.put("assetConnectedLinks",connectedLinks);
                counterIncrement++;
                return object;
        }

        public static void main(String[] args) throws Exception {
                // setting up the environment from passed in arguments
         SpringApplication.run(CustomPublisher.class, args);
```

```
                SampleConfiguration config = new SampleConfiguration();

                //creates PxGridControl object with the environment
                PxgridControl control = new PxgridControl(config);

                // AccountActivate
                while (control.accountActivate() != AccountState.ENABLED) {
                        Thread.sleep(45000);
                }
                Console.log("pxGrid controller version=" + control.getControllerVersion());

                //creating new service.
                Service service = createService();

                //registers the service that we created above
                control.registerService(service.getName(), service.getProperties());

                Service[] list_of_services;

                //below lookup should find main pxGrid server.
                list_of_services = control.lookupService(PUBSUBSERVICE);
                if (list_of_services.length == 0) {
                        Console.log("service isn't there");
                        return;
                }

                // takes the main pxGrid server node
                Service wsPubsubService = list_of_services[0];

                //get wsURL so we can get the URI later from it via REST query.
                String wsURL = wsPubsubService.getProperties().get("wsUrl");
                Console.log("wsUrl=" + wsURL);

                // pxGrid AccessSecret
                String secret = control.getAccessSecret(wsPubsubService.getNodeName());

                //setting up client manager which will use ssl connection for authentication.
                ClientManager client = ClientManager.createClient();
                SslEngineConfigurator sslEngineConfigurator = new
SslEngineConfigurator(config.getSSLContext());
                sslEngineConfigurator.setHostnameVerifier(new HostnameVerifier() {
                        @Override
                        public boolean verify(String hostname, SSLSession session) {
                                return true;
                        }
                });
                client.getProperties().put(ClientProperties.SSL_ENGINE_CONFIGURATOR, sslEngineConfigurator);
                client.getProperties().put(ClientProperties.CREDENTIALS,
                                new Credentials(config.getUserName(), secret.getBytes()));

                // WebSocket connect
                StompPubsubClientEndpoint endpoint = new StompPubsubClientEndpoint();

                //get URI, connect pxGrid client to the pxGrid server so that we can publish to dynamic
service
                URI uri = new URI(wsURL);

                javax.websocket.Session session = client.connectToServer(endpoint, uri);

                // STOMP connect
                endpoint.connect(uri.getHost());

                /*
                 * publishing to the dynamic service. This message "dynamic topic publish" will be received by
all subscribers to the service.
                 * only triggers when key is pressed. This is to make it so we can subscribe another client to
the service by running
                 * DynamicServiceSubscribe.java to see if it receives the published info.
                 */
                SampleHelper.prompt("press <enter> to start the publishing...");

                //for multi publishing
```

```
                //for(int i = 0; i < 20000; i++) {
                        JSONArray deviceArr = new JSONArray();
                        JSONObject device_object = new JSONObject();
                        JSONObject device = createJsonObject();
                        deviceArr.add(device);
                        device_object.put("asset", device);
                        device_object.put("opType","UPDATE");
                        byte[] array = device_object.toJSONString().getBytes(StandardCharsets.UTF_8);
                        endpoint.publish(TOPIC_PATH, array);
                        Console.log(device.toJSONString());
            //          Console.log(Integer.toString(i));
                //}

                /*JSONArray deviceArr = new JSONArray();
                JSONObject device_object = new JSONObject();

                JSONObject device1 = createJsonObject();
                JSONObject device2 = createJsonObject();
                JSONObject device3 = createJsonObject();
                JSONObject device4 = createJsonObject();

                device2.put(prefix+"SwRevision", "7.8");
                device3.put(prefix+"SwRevision", "3.3");
                device3.put(prefix+"HwRevision", "2.5");
                device4.put(prefix+"SwRevision", "3.5");
                deviceArr.add(device1);
                deviceArr.add(device2);
                deviceArr.add(device3);
                deviceArr.add(device4);


                device_object.put("asset", device1);
                device_object.put("opType","UPDATE");
                byte[] array = device_object.toJSONString().getBytes(StandardCharsets.UTF_8);
                endpoint.publish(TOPIC_PATH, array);
                */

                //SampleHelper.prompt("press <enter> to disconnect...");

                // STOMP disconnect
                //endpoint.disconnect("ID-123");
                // Wait for disconnect receipt
                //Thread.sleep(3000);

                //session.close();
        }
}
```

Step 7 Go to folder

```
johns-macbook-pro:java jeppich$ ls
pom.xml                 src
run_publisher.sh        target
johns-macbook-pro:java jeppich$
```

Step 8 Run mvn install to recompile

```
johns-macbook-pro:java jeppich$ mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] --------------< com.cisco.pxgrid:pxgrid-rest-ws-samples >---------------
[INFO] Building Publisher 2.0.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ pxgrid-rest-ws-samples ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 40 resources
```

```
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ pxgrid-rest-ws-samples ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 40 source files to /Users/jeppich/maven/repo/pxgrid-rest-ws/java/target/classes
[WARNING] /Users/jeppich/maven/repo/pxgrid-rest-
ws/java/src/main/java/com/cisco/pxgrid/samples/ise/http/CustomPublisher.java:
/Users/jeppich/maven/repo/pxgrid-rest-
ws/java/src/main/java/com/cisco/pxgrid/samples/ise/http/CustomPublisher.java uses unchecked or unsafe
operations.
[WARNING] /Users/jeppich/maven/repo/pxgrid-rest-
ws/java/src/main/java/com/cisco/pxgrid/samples/ise/http/CustomPublisher.java: Recompile with -Xlint:unchecked
for details.
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ pxgrid-rest-ws-samples ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/jeppich/maven/repo/pxgrid-rest-ws/java/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ pxgrid-rest-ws-samples ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.18.1:test (default-test) @ pxgrid-rest-ws-samples ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.6:jar (default-jar) @ pxgrid-rest-ws-samples ---
[INFO] Building jar: /Users/jeppich/maven/repo/pxgrid-rest-ws/java/target/pxgrid-rest-ws-samples-2.0.0-
SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:1.5.6.RELEASE:repackage (default) @ pxgrid-rest-ws-samples ---
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ pxgrid-rest-ws-samples ---
[INFO] Installing /Users/jeppich/maven/repo/pxgrid-rest-ws/java/target/pxgrid-rest-ws-samples-2.0.0-
SNAPSHOT.jar to /Users/jeppich/.m2/repository/com/cisco/pxgrid/pxgrid-rest-ws-samples/2.0.0-SNAPSHOT/pxgrid-
rest-ws-samples-2.0.0-SNAPSHOT.jar
[INFO] Installing /Users/jeppich/maven/repo/pxgrid-rest-ws/java/pom.xml to
/Users/jeppich/.m2/repository/com/cisco/pxgrid/pxgrid-rest-ws-samples/2.0.0-SNAPSHOT/pxgrid-rest-ws-samples-
2.0.0-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 19.717 s
[INFO] Finished at: 2018-04-21T21:47:59-04:00
[INFO] ------------------------------------------------------------------------
johns-macbook-pro:java jeppich$
```

Step 9 Replace the **/Applications/api_partner_fc3/api_simulator/snapshot.jar** file with the one that was just created. (**i.e /Users/jeppich/.m2/repository/com/cisco/pxgrid/pxgrid-rest-ws-samples/2.0.0-SNAPSHOT**)

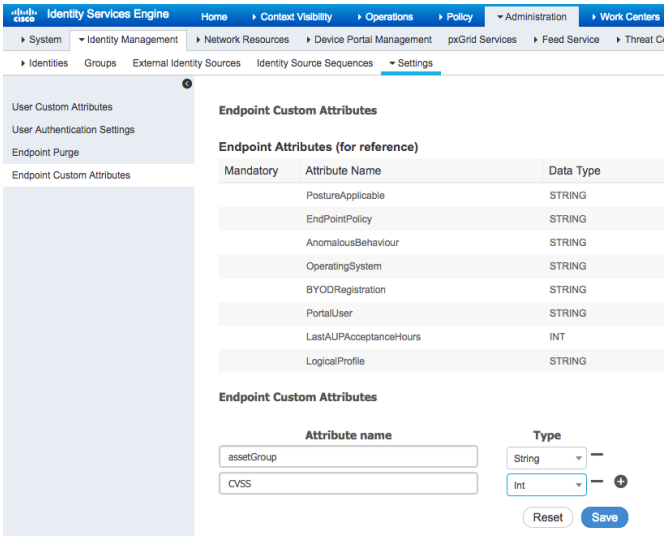
```
cp pxgrid-rest-ws-samples-2.0.0-SNAPSHOT.jar /Applications/api_partner_fc3/api_simulator
```

Step 10 Run the script, you should see **"value":"7", "key":"CVSS"** in the output

```
ClientEndpoint    : STOMP SEND topic=/topic/com.cisco.endpoint.asset
command=SEND, headers={'content-length':'576','destination':'/topic/com.cisco.endpoint.asset',},
content.length=576
21-Apr-18 21:56:49.171 [main-1]:
{"assetHwRevision":"5.6","assetProtocol":"CIP","assetConnectedLinks":[{"value":"7","key":"CVSS"},{"value":"Ro
ot","key":"assetGroup"},{"value":"1","key":"indattr3"}],"assetVendor":"ACME","assetSwRevision":"4.6","assetCu
stomAttributes":[{"value":"7","key":"CVSS"},{"value":"Root","key":"assetGroup"},{"value":"1","key":"indattr3"
}],"assetProductId":"IE2000","assetSerialNumber":"1212121213243","assetMacAddress":"b4:54:0a:e8:c1:fd","asset
```

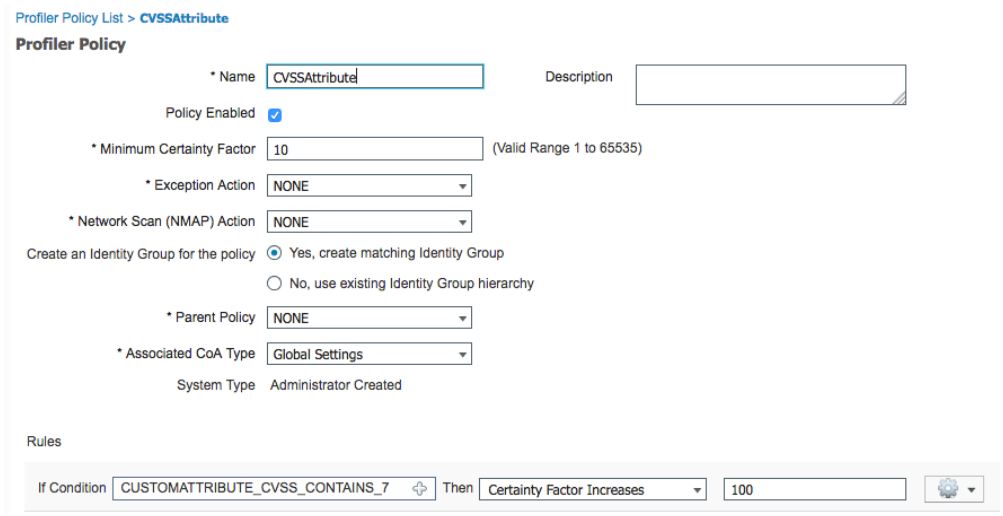
Id":"275","assetIpAddress":"117.65.215.140","assetName":"Abjergaryn - 43","assetDeviceType":"EtherNet\/IP Node"}

Step 11 Select **Administration->System->Settings->Endpoint Custom Attributes**, and add **CVSS** as the **attribute name** and **int** as the **Type**



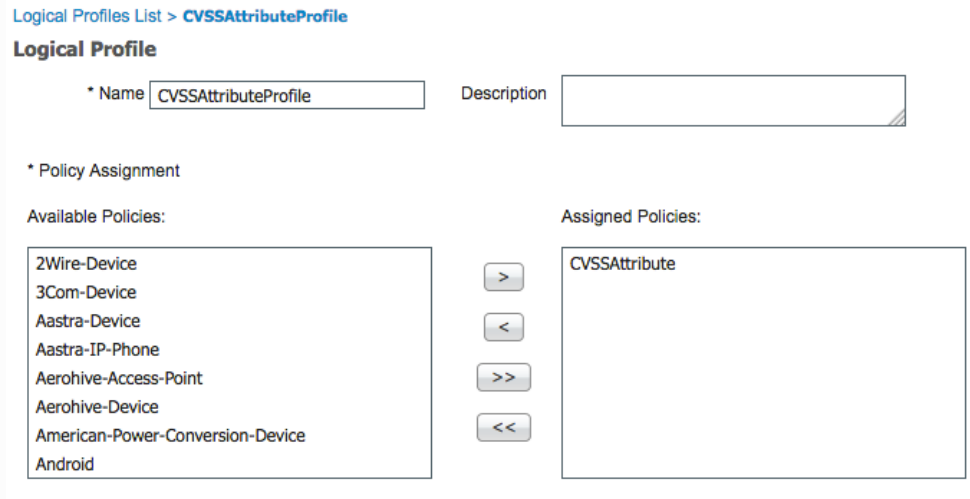
Step 12 Select **Save**

Step 13 Create CVSSAttribute Profiling Policy that will be used to define the CUSTOMATTRIBUTE



Step 14 Create CVSSAttribute Logical Profile is also created.

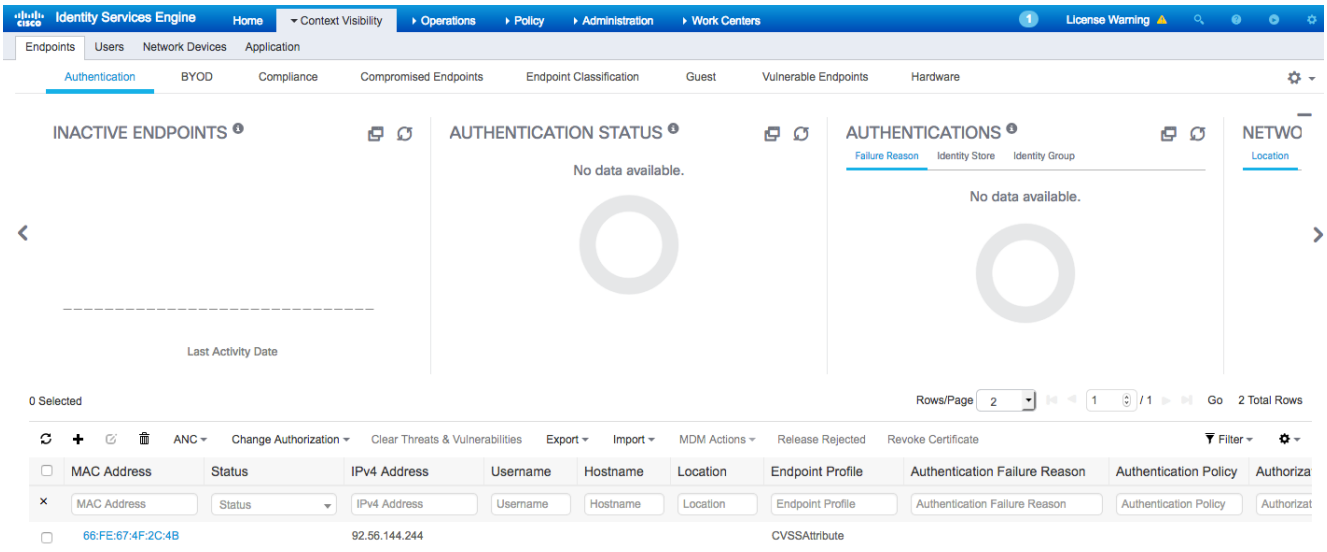
Note: Disable VendorACME policy if enabled



Step 15 Run the script, you should see **"value":"7", "key":"CVSS"** in the output

```
ClientEndpoint     : STOMP SEND topic=/topic/com.cisco.endpoint.asset
command=SEND, headers={'content-length':'575','destination':'/topic/com.cisco.endpoint.asset',},
content.length=575
21-Apr-18 22:25:59.684 [main-1]:
{"assetHwRevision":"5.6","assetProtocol":"CIP","assetConnectedLinks":[{"value":"7","key":"CVSS"},{"value":"Ro
ot","key":"assetGroup"},{"value":"1","key":"indattr3"}],"assetVendor":"ACME","assetSwRevision":"4.6","assetCu
stomAttributes":[{"value":"7","key":"CVSS"},{"value":"Root","key":"assetGroup"},{"value":"1","key":"indattr3"
}],"assetProductId":"IE2000","assetSerialNumber":"1212121213243","assetMacAddress":"66:fe:67:4f:2c:4b","asset
Id":"230","assetIpAddress":"92.56.144.244","assetName":"Abjergaryn - 44","assetDeviceType":"EtherNet\/IP
Node"}
```

Step 16 Select **Context Visibility->Endpoints->Authentication**, you will see CVSSAttribute Endpoint Profile



API_Simulator Context-In Code

SampleConfiguration

This code configures the command-line arguments to run the inside the **./run_publisher** script.

```
package com.cisco.pxgrid.samples.ise.http;

import java.io.FileInputStream;
import java.io.IOException;
import java.net.Authenticator;
import java.net.PasswordAuthentication;
import java.net.Socket;
import java.security.GeneralSecurityException;
import java.security.KeyStore;
import java.security.Principal;
import java.security.PrivateKey;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.util.Collection;
import java.util.Enumeration;

import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.KeyManager;
import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.net.ssl.X509KeyManager;
import javax.net.ssl.X509TrustManager;

public class SampleConfiguration {
    protected final static String PROP_HOSTNAMES="PXGRID_HOSTNAMES";
    protected final static String PROP_USERNAME="PXGRID_USERNAME";
    protected final static String PROP_PASSWORD="PXGRID_PASSWORD";
    protected final static String PROP_GROUP="PXGRID_GROUP";
    protected final static String PROP_DESCRIPTION="PXGRID_DESCRIPTION";
    protected final static String PROP_KEYSTORE_FILENAME="PXGRID_KEYSTORE_FILENAME";
    protected final static String PROP_KEYSTORE_PASSWORD="PXGRID_KEYSTORE_PASSWORD";
    protected final static String PROP_TRUSTSTORE_FILENAME="PXGRID_TRUSTSTORE_FILENAME";
    protected final static String PROP_TRUSTSTORE_PASSWORD="PXGRID_TRUSTSTORE_PASSWORD";

    private String[] hostnames;
    private String username;
    private String password;
    private String[] groups;
    private String description;
    private SSLContext sslContext;

    private String keystoreFilename;
    private String keystorePassword;
    private String truststoreFilename;
    private String truststorePassword;

        public SampleConfiguration() throws GeneralSecurityException, IOException {
                load();
                print();
        }

    public String getUserName() {
                return username;
        }

    public void setUsername(String username) {
                this.username = username;
        }
```

```
        public String[] getGroups() {
                return groups;
        }

        public String getDescription() {
                return description;
        }

    public SSLContext getSSLContext() {
        return sslContext;
    }

    public String getPassword() {
                return password;
        }

    public String[] getHostnames() {
                return hostnames;
        }

    private void load() throws GeneralSecurityException, IOException {
        String hostnameProperty = System.getProperty(PROP_HOSTNAMES);
        username = System.getProperty(PROP_USERNAME);
        password = System.getProperty(PROP_PASSWORD);
        String group_property = System.getProperty(PROP_GROUP);
        description = System.getProperty(PROP_DESCRIPTION);

        keystoreFilename = System.getProperty(PROP_KEYSTORE_FILENAME);
        keystorePassword = System.getProperty(PROP_KEYSTORE_PASSWORD);
        truststoreFilename = System.getProperty(PROP_TRUSTSTORE_FILENAME);
        truststorePassword = System.getProperty(PROP_TRUSTSTORE_PASSWORD);

        if (hostnameProperty == null || hostnameProperty.isEmpty()) throw new
IllegalArgumentException("Missing " + PROP_HOSTNAMES);
        if (username == null || username.isEmpty()) throw new IllegalArgumentException("Missing " +
PROP_USERNAME);
        if (truststoreFilename == null || truststoreFilename.isEmpty()) throw new
IllegalArgumentException("Missing " + PROP_TRUSTSTORE_FILENAME);
        if (truststorePassword == null || truststorePassword.isEmpty()) throw new
IllegalArgumentException("Missing " + PROP_TRUSTSTORE_PASSWORD);

        hostnames = hostnameProperty.split(",");

        if (group_property != null && !group_property.isEmpty()) {
                        groups = group_property.split(",");
        }

        if (description != null) {
                 if (description.isEmpty()) description = null;
                 else description = description.trim();
        }

        sslContext = SSLContext.getInstance("TLSv1.2");
        sslContext.init(getKeyManagers(), getTrustManagers(), null);
    }

    public void setupAuth(HttpsURLConnection https) throws GeneralSecurityException, IOException {
                Authenticator.setDefault(new MyAuthenticator());
    }

    private class MyAuthenticator extends Authenticator {
        public PasswordAuthentication getPasswordAuthentication() {
            return (new PasswordAuthentication(username, password.toCharArray()));
        }
    }

        private KeyManager[] getKeyManagers() throws IOException, GeneralSecurityException {
                if (keystoreFilename == null || keystoreFilename.isEmpty())
                        return null;

                KeyStore ks = keystoreFilename.endsWith(".p12") ? KeyStore.getInstance("pkcs12") :
KeyStore.getInstance("JKS");
                FileInputStream in = new FileInputStream(keystoreFilename);
```

```
                ks.load(in, keystorePassword.toCharArray());
                in.close();
                KeyManagerFactory kmf =
KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
                kmf.init(ks, keystorePassword.toCharArray());
                KeyManager[] mngrs = kmf.getKeyManagers();

                if (mngrs == null || mngrs.length == 0) {
                        throw new GeneralSecurityException("no key managers found");
                }

                if (mngrs[0] instanceof X509KeyManager == false) {
                        throw new GeneralSecurityException("key manager is not for X509");
                }

                return new KeyManager[] { new SampleX509KeyManager((X509KeyManager) mngrs[0]) };
        }

        private TrustManager[] getTrustManagers() throws IOException, GeneralSecurityException {
                FileInputStream in = new FileInputStream(truststoreFilename);

                KeyStore ks = null;
                if(truststoreFilename.endsWith(".pem")) {
                        ks = KeyStore.getInstance("JKS");
                        ks.load(null, null);
                        CertificateFactory certFac = CertificateFactory.getInstance("X.509");
                        Collection<? extends Certificate> certs = certFac.generateCertificates(in);
                        int i = 0;
                        for(Certificate c : certs) {
                                ks.setCertificateEntry("trust-" + i, c);
                        }
                } else if(truststoreFilename.endsWith(".p12")) {
                        ks = KeyStore.getInstance("pkcs12");
                        ks.load(in, truststorePassword.toCharArray());
                } else {
                        ks = KeyStore.getInstance("JKS");
                        ks.load(in, truststorePassword.toCharArray());
                }

                in.close();

                Enumeration<String> e = ks.aliases();
                boolean hasCertEntries = false;

                while (e.hasMoreElements()) {
                        String alias = e.nextElement();

                        if (ks.isCertificateEntry(alias)) {
                                hasCertEntries = true;
                        }
                }

                if (hasCertEntries == false) {
                        e = ks.aliases();

                        while (e.hasMoreElements()) {
                                String alias = e.nextElement();

                                if (ks.isKeyEntry(alias)) {
                                        Certificate[] chain = ks.getCertificateChain(alias);

                                        for (int i = 0; i < chain.length; ++i) {
                                                ks.setCertificateEntry(alias + "." + i, chain[i]);
                                        }
                                }
                        }
                }

                TrustManagerFactory tmf =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
                tmf.init(ks);

                TrustManager[] tms = tmf.getTrustManagers();
```

```
                if (tms == null || tms.length == 0) {
                        throw new GeneralSecurityException("no trust managers found");
                }

                if (tms[0] instanceof X509TrustManager == false) {
                        throw new GeneralSecurityException("trust manager is not for X509");
                }

                return new TrustManager[] { new SampleX509TrustManager((X509TrustManager) tms[0]) };
        }

        private static class SampleX509KeyManager implements X509KeyManager {

                private X509KeyManager mngr;

                public SampleX509KeyManager(X509KeyManager mngr) {
                        this.mngr = mngr;
                }

                @Override
                public String chooseClientAlias(String[] arg0, Principal[] arg1, Socket arg2) {
                        String alias = mngr.chooseClientAlias(arg0, arg1, arg2);

                        if (alias == null) {
                                alias = mngr.chooseClientAlias(arg0, null, arg2);

                                if (alias == null) {
                                        throw new RuntimeException("no client certificate found ...");
                                }
                        }

                        return alias;
                }

                @Override
                public String chooseServerAlias(String arg0, Principal[] arg1, Socket arg2) {
                        throw new RuntimeException("Not implemented");
                }

                @Override
                public X509Certificate[] getCertificateChain(String arg0) {
                        return mngr.getCertificateChain(arg0);
                }

                @Override
                public String[] getClientAliases(String arg0, Principal[] arg1) {
                        return mngr.getClientAliases(arg0, null);
                }

                @Override
                public PrivateKey getPrivateKey(String arg0) {
                        return mngr.getPrivateKey(arg0);
                }

                @Override
                public String[] getServerAliases(String arg0, Principal[] arg1) {
                        throw new RuntimeException("Not implemented");
                }
        }

        private static class SampleX509TrustManager implements X509TrustManager {
                private X509TrustManager mngr;

                public SampleX509TrustManager(X509TrustManager mngr) {
                        this.mngr = mngr;
                }

                @Override
                public void checkClientTrusted(X509Certificate[] arg0, String arg1) throws
CertificateException {
                        throw new RuntimeException("not implemented");
                }
```

```
                @Override
                public void checkServerTrusted(X509Certificate[] arg0, String arg1) throws
CertificateException {
                        try {
                                mngr.checkServerTrusted(arg0, arg1);
                        } catch (CertificateException e) {
                                throw new CertificateException("Server certificate is not trusted:" +
arg0[0].getSubjectX500Principal(),
                                                e);
                        }
                }

                @Override
                public X509Certificate[] getAcceptedIssuers() {
                        return mngr.getAcceptedIssuers();
                }
        }

    private void print() {
        System.out.println("------- properties -------");
        System.out.print("  hostnames=");
        for (String hostname : hostnames) System.out.print(hostname + " ");
        System.out.println();
        System.out.println("  username=" + username);
        System.out.println("  password=" + password);
        System.out.print("  groups=");
        for (String group : groups) System.out.print(group + " ");
        System.out.println();
        System.out.println("  description=" + description);
        System.out.println("  keystoreFilename=" + keystoreFilename);
        System.out.println("  keystorePassword=" + keystorePassword);
        System.out.println("  truststoreFilename=" + truststoreFilename);
        System.out.println("  truststorePassword=" + truststorePassword);
        System.out.println("--------------------------");
    }
}
```

PxgridControl

This code provides the pxGrid client with account creation on the ISE pxGrid node and service lookup request and access secret to the peer node.

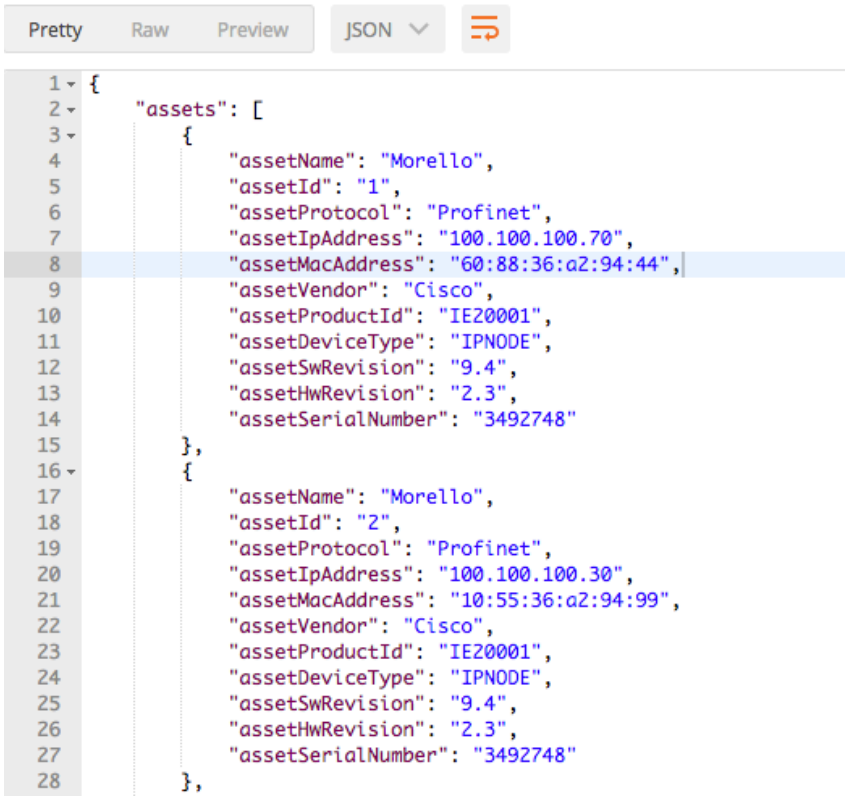
```
package com.cisco.pxgrid.samples.ise.http;

import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.URL;
import java.util.Base64;
import java.util.Map;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLSession;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.cisco.pxgrid.model.AccessSecretRequest;
import com.cisco.pxgrid.model.AccessSecretResponse;
import com.cisco.pxgrid.model.AccountActivateRequest;
import com.cisco.pxgrid.model.AccountActivateResponse;
import com.cisco.pxgrid.model.AccountCreateRequest;
import com.cisco.pxgrid.model.AccountCreateResponse;
import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Authorization;
import com.cisco.pxgrid.model.AuthorizationRequest;
import com.cisco.pxgrid.model.AuthorizationResponse;
```

```
import com.cisco.pxgrid.model.Service;
import com.cisco.pxgrid.model.ServiceLookupRequest;
import com.cisco.pxgrid.model.ServiceLookupResponse;
import com.cisco.pxgrid.model.ServiceRegisterRequest;
import com.cisco.pxgrid.model.ServiceRegisterResponse;
import com.google.gson.Gson;

/**
 * Using HTTPS for pxGrid control
 */
public class PxgridControl {
        private static Logger logger = LoggerFactory.getLogger(PxgridControl.class);
        private SampleConfiguration config;
    private String controllerVersion;

    public PxgridControl(SampleConfiguration config) {
        this.config = config;
        }

        private <T> T sendRequest(HttpsURLConnection https, Object request, Class<T> responseClass) throws
IOException {
                https.setRequestProperty("Content-Type", "application/json");
                https.setRequestProperty("Accept", "application/json");

                Gson gson = new Gson();

                OutputStreamWriter out = new OutputStreamWriter(https.getOutputStream());
                logger.info("Request={}", gson.toJson(request));
                gson.toJson(request, out);
                out.flush();

        InputStreamReader in = new InputStreamReader(https.getInputStream());
        T response = gson.fromJson(in, responseClass);
                logger.info("Response={}", gson.toJson(response));

        return response;
        }

    private HttpsURLConnection getHttpsURLConnection(String urlSuffix) throws IOException {
                String url = "https://" + config.getHostnames()[0] + ":8910/pxgrid/control/" + urlSuffix;
                URL conn = new URL(url);
                HttpsURLConnection https = (HttpsURLConnection) conn.openConnection();

                // SSL and Auth
                https.setSSLSocketFactory(config.getSSLContext().getSocketFactory());

                https.setRequestMethod("POST");

                String userPassword = config.getUserName() + ":" + config.getPassword();
                String encoded = Base64.getEncoder().encodeToString(userPassword.getBytes());
                https.setRequestProperty("Authorization", "Basic " + encoded);
                https.setHostnameVerifier(new HostnameVerifier() {
                        @Override
                        public boolean verify(String hostname, SSLSession session) {
                                return true;
                        }
                });
                https.setDoInput(true);
                https.setDoOutput(true);

                return https;
    }


    /**
     * Create new account
     *
     * @return password
     */
    public String accountCreate() throws IOException {
        HttpsURLConnection https = getHttpsURLConnection("AccountCreate");
                AccountCreateRequest request = new AccountCreateRequest();
                request.setNodeName(config.getUserName());
```

```
                AccountCreateResponse response = sendRequest(https, request, AccountCreateResponse.class);
                return response.getPassword();
    }

    public AccountState accountActivate() throws IOException {
                HttpsURLConnection https = getHttpsURLConnection("AccountActivate");
                AccountActivateRequest request = new AccountActivateRequest();
                request.setDescription(config.getDescription());
                AccountActivateResponse response = sendRequest(https, request, AccountActivateResponse.class);
                controllerVersion = response.getVersion();
                return response.getAccountState();
    }

        public void registerService(String name, Map<String, String> properties) throws IOException {
                HttpsURLConnection https = getHttpsURLConnection("ServiceRegister");
                ServiceRegisterRequest request = new ServiceRegisterRequest();
                request.setName(name);
                request.setProperties(properties);
                sendRequest(https, request, ServiceRegisterResponse.class);
        }

        public Service[] lookupService(String name) throws IOException {
                HttpsURLConnection https = getHttpsURLConnection("ServiceLookup");
                ServiceLookupRequest request = new ServiceLookupRequest();
                request.setName(name);
                ServiceLookupResponse response = sendRequest(https, request, ServiceLookupResponse.class);
                return response.getServices();
        }

        public String getAccessSecret(String peerNodeName) throws IOException  {
                HttpsURLConnection https = getHttpsURLConnection("AccessSecret");
                AccessSecretRequest request = new AccessSecretRequest();
                request.setPeerNodeName(peerNodeName);
                AccessSecretResponse response = sendRequest(https, request, AccessSecretResponse.class);
                return response.getSecret();
        }

        public boolean isAuthorized(String requestNodeName, String serviceName, String operation) throws
IOException {
                HttpsURLConnection https = getHttpsURLConnection("Authorization");
                AuthorizationRequest request = new AuthorizationRequest();
                request.setRequestNodeName(requestNodeName);
                request.setServiceName(serviceName);

                request.setServiceOperation(operation);

                AuthorizationResponse response = sendRequest(https, request, AuthorizationResponse.class);
                return (response.getAuthorization() == Authorization.PERMIT);
        }

        public String getControllerVersion() {
                return controllerVersion;
        }
}
```

PublisherController

This code publishes the “assets” topic and looks like the following:



Pretty Raw Preview JSON


```
{
    "assets": [
        {
            "assetName": "Morello",
            "assetId": "1",
            "assetProtocol": "Profinet",
            "assetIpAddress": "100.100.100.70",
            "assetMacAddress": "60:88:36:a2:94:44",
            "assetVendor": "Cisco",
            "assetProductId": "IE20001",
            "assetDeviceType": "IPNODE",
            "assetSwRevision": "9.4",
            "assetHwRevision": "2.3",
            "assetSerialNumber": "3492748"
        },
        {
            "assetName": "Morello",
            "assetId": "2",
            "assetProtocol": "Profinet",
            "assetIpAddress": "100.100.100.30",
            "assetMacAddress": "10:55:36:a2:94:99",
            "assetVendor": "Cisco",
            "assetProductId": "IE20001",
            "assetDeviceType": "IPNODE",
            "assetSwRevision": "9.4",
            "assetHwRevision": "2.3",
            "assetSerialNumber": "3492748"
        },
```


```
package com.cisco.pxgrid.samples.ise.http;

import java.util.concurrent.ThreadLocalRandom;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class PublisherController {

        public Devices DeviceGenerator(int i) {
                return new Devices(Integer.toString(i+1), "Morello",
                                "100.100.100."+10*ThreadLocalRandom.current().nextInt(1, 9 + 1),
(10*ThreadLocalRandom.current().nextInt(1, 9 + 1))+":"+(11*ThreadLocalRandom.current().nextInt(1, 9 +
1))+":36:a2:94:"+
                                                (11*ThreadLocalRandom.current().nextInt(1, 9 + 1)), "Cisco",
"IE20001", "3492748", "IPNODE", "9.4", "2.3", "Profinet");
        }

    @RequestMapping("/getAssets")
    public DeviceList device() {
                Devices[] device_list = new Devices[5];
                for(int i = 0; i < 5; i++) {
                        device_list[i] = DeviceGenerator(i);
                }
         return new DeviceList(device_list);
    }
}
```

Custom Publisher

The CustomPublisher Java code publishes the asset device attributes by calling the service name “com.cisco.endpoint.asset”, topic path “/topic/com.cisco.endpoint.asset”, PUBSERVCE “com.cisco.ise.pubsub.

This will return the wsPubsubServce: “com.cisco.ise.pubsub”, restbaseURL http://raghdasa-lnv1:8080, and assetTopic, “/topic/com.cisco.endpoint.asset”.

```
package com.cisco.pxgrid.samples.ise.http;

import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Map;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.SSLSession;
import java.util.Random;
import org.glassfish.tyrus.client.ClientManager;
import org.glassfish.tyrus.client.ClientProperties;
import org.glassfish.tyrus.client.SslEngineConfigurator;
import org.glassfish.tyrus.client.auth.Credentials;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import java.util.concurrent.ThreadLocalRandom;

import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 * Sample creation of a Dynamic Service by a client, also publishes. Another Client may subscribe to this
service and receive
 * notifications when this service is published to.
 *
 * USE CASE for Dynamic Services: My company that requires all devices that connect to our network to be
specially authenticated so they need
 * more information. Using pxGrid dynamic services, I can set up a service that broadcasts the requirements
to all devices that connect and tell
 * them to send this information over to allow for authentication.
 *
 * @author anirvenk
 */

@SpringBootApplication

public class CustomPublisher {
        private static final String SERVICE_NAME = "com.cisco.endpoint.asset";
        private static final String TOPIC_PATH = "/topic/com.cisco.endpoint.asset";
        private static final String PUBSUBSERVICE = "com.cisco.ise.pubsub";
        public static int counter = 0;
        private static String prefix = "asset";
        private static int counterIncrement = 1;

        /**
         * Static class that is used to create a sample Service object.
         */
        public static Service createService() {
                Service service = new Service();
                service.setName(SERVICE_NAME);
                service.setNodeName("dynamic capability");
                Map<String, String> properties = new HashMap<String, String>();
                properties.put("wsPubsubService", PUBSUBSERVICE);
                properties.put("assetTopic", TOPIC_PATH);
                properties.put("restBaseURL", "http://raghdasa-lnv1:8080");
                service.setProperties(properties);
                return service;
        }
        public static String getRandomIpAddress(){
                Random r = new Random();
                return r.nextInt(256) + "." + r.nextInt(256) + "." + r.nextInt(256) + "." + r.nextInt(256);
        }
    public static String getRandomMacAddress(){
        Random rand = new Random();
        byte[] macAddr = new byte[6];
        rand.nextBytes(macAddr);
```

```
        macAddr[0] = (byte)(macAddr[0] & (byte)254);  //zeroing last 2 bytes to make it unicast and locally
adminstrated

        StringBuilder sb = new StringBuilder(18);
        for(byte b : macAddr){

                if(sb.length() > 0)
                sb.append(":");

                        sb.append(String.format("%02x", b));
                }


        return sb.toString();
    }
        public static JSONObject createJsonObject() {
                JSONObject object = new JSONObject();
                //change prefix to asset later
                object.put(prefix+"Id",
Integer.toString(1+counterIncrement)+Integer.toString(15*ThreadLocalRandom.current().nextInt(1, 6 + 1)));
                object.put(prefix+"Name", "Abjergaryn -
4"+Integer.toString(ThreadLocalRandom.current().nextInt(1, 9 + 1)));
                object.put(prefix+"IpAddress", getRandomIpAddress());
                //+Integer.toString(ThreadLocalRandom.current().nextInt(45, 100 + 1)));
                //object.put(prefix+"MacAddress", "28:63:36:a2:94:"+Integer.toString(30+counterIncrement));
                object.put(prefix+"MacAddress", getRandomMacAddress());
                object.put(prefix+"Vendor", "ACME");
                object.put(prefix+"ProductId", "IE2000");
                object.put(prefix+"SerialNumber", "1212121213243");
                object.put(prefix+"DeviceType", "EtherNet/IP Node");
                object.put(prefix+"SwRevision", "4.6");
                object.put(prefix+"HwRevision", "5.6");
                object.put(prefix+"Protocol", "CIP");
                 JSONArray customAttr = new JSONArray();
                 JSONArray connectedLinks = new JSONArray();

    JSONObject object1 = new JSONObject();
                object1.put("key","CVSS");
                object1.put("value","7");
                customAttr.add(object1);
                 JSONObject object2 = new JSONObject();
                 object2.put("key","assetGroup");
                 object2.put("value","Root");
                 customAttr.add(object2);
                 JSONObject object3 = new JSONObject();
                 object3.put("key","indattr3");
          object3.put("value","1");
          customAttr.add(object3);
          connectedLinks.add(object1);
                 connectedLinks.add(object2);
                 connectedLinks.add(object3);
                object.put("assetCustomAttributes",customAttr);
                object.put("assetConnectedLinks",connectedLinks);
                counterIncrement++;
                return object;
        }

        public static void main(String[] args) throws Exception {
                // setting up the environment from passed in arguments
         SpringApplication.run(CustomPublisher.class, args);

                SampleConfiguration config = new SampleConfiguration();

                //creates PxGridControl object with the environment
                PxgridControl control = new PxgridControl(config);

                // AccountActivate
                while (control.accountActivate() != AccountState.ENABLED) {
                        Thread.sleep(45000);
                }
                Console.log("pxGrid controller version=" + control.getControllerVersion());
```

```
                //creating new service.
                Service service = createService();

                //registers the service that we created above
                control.registerService(service.getName(), service.getProperties());

                Service[] list_of_services;

                //below lookup should find main pxGrid server.
                list_of_services = control.lookupService(PUBSUBSERVICE);
                if (list_of_services.length == 0) {
                        Console.log("service isn't there");
                        return;
                }

                // takes the main pxGrid server node
                Service wsPubsubService = list_of_services[0];

                //get wsURL so we can get the URI later from it via REST query.
                String wsURL = wsPubsubService.getProperties().get("wsUrl");
                Console.log("wsUrl=" + wsURL);

                // pxGrid AccessSecret
                String secret = control.getAccessSecret(wsPubsubService.getNodeName());

                //setting up client manager which will use ssl connection for authentication.
                ClientManager client = ClientManager.createClient();
                SslEngineConfigurator sslEngineConfigurator = new
SslEngineConfigurator(config.getSSLContext());
                sslEngineConfigurator.setHostnameVerifier(new HostnameVerifier() {
                        @Override
                        public boolean verify(String hostname, SSLSession session) {
                                return true;
                        }
                });
                client.getProperties().put(ClientProperties.SSL_ENGINE_CONFIGURATOR, sslEngineConfigurator);
                client.getProperties().put(ClientProperties.CREDENTIALS,
                                new Credentials(config.getUserName(), secret.getBytes()));

                // WebSocket connect
                StompPubsubClientEndpoint endpoint = new StompPubsubClientEndpoint();

                //get URI, connect pxGrid client to the pxGrid server so that we can publish to dynamic
service
                URI uri = new URI(wsURL);

                javax.websocket.Session session = client.connectToServer(endpoint, uri);

                // STOMP connect
                endpoint.connect(uri.getHost());

                /*
                 * publishing to the dynamic service. This message "dynamic topic publish" will be received by
all subscribers to the service.
                 * only triggers when key is pressed. This is to make it so we can subscribe another client to
the service by running
                 * DynamicServiceSubscribe.java to see if it receives the published info.
                 */
                SampleHelper.prompt("press <enter> to start the publishing...");

                //for multi publishing
                //for(int i = 0; i < 20000; i++) {
                        JSONArray deviceArr = new JSONArray();
                        JSONObject device_object = new JSONObject();
                        JSONObject device = createJsonObject();
                        deviceArr.add(device);
                        device_object.put("asset", device);
                        device_object.put("opType","UPDATE");
                        byte[] array = device_object.toJSONString().getBytes(StandardCharsets.UTF_8);
                        endpoint.publish(TOPIC_PATH, array);
                        Console.log(device.toJSONString());
            //          Console.log(Integer.toString(i));
                //}
```

```
                /*JSONArray deviceArr = new JSONArray();
                JSONObject device_object = new JSONObject();

                JSONObject device1 = createJsonObject();
                JSONObject device2 = createJsonObject();
                JSONObject device3 = createJsonObject();
                JSONObject device4 = createJsonObject();

                device2.put(prefix+"SwRevision", "7.8");
                device3.put(prefix+"SwRevision", "3.3");
                device3.put(prefix+"HwRevision", "2.5");
                device4.put(prefix+"SwRevision", "3.5");
                deviceArr.add(device1);
                deviceArr.add(device2);
                deviceArr.add(device3);
                deviceArr.add(device4);


                device_object.put("asset", device1);
                device_object.put("opType","UPDATE");
                byte[] array = device_object.toJSONString().getBytes(StandardCharsets.UTF_8);
                endpoint.publish(TOPIC_PATH, array);
                */

                //SampleHelper.prompt("press <enter> to disconnect...");

                // STOMP disconnect
                //endpoint.disconnect("ID-123");
                // Wait for disconnect receipt
                //Thread.sleep(3000);

                //session.close();
        }
}
```

CustomSubscriber

This code sets up the subscriber to the published topic, in the case, the ISE pxGrid node.

```
package com.cisco.pxgrid.samples.ise.http;

import java.net.URI;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.SSLSession;

import org.glassfish.tyrus.client.ClientManager;
import org.glassfish.tyrus.client.ClientProperties;
import org.glassfish.tyrus.client.SslEngineConfigurator;
import org.glassfish.tyrus.client.auth.Credentials;
import com.cisco.pxgrid.model.AccountState;
import com.cisco.pxgrid.model.Service;

/**
 * Demonstrates how to subscribe to a Dynamic Service using REST/WS.
 * Works hand in hand with Dynamic Service.java class.
 */
public class CustomSubscriber {
        // Subscribe handler class
        private static class SessionHandler implements StompSubscription.Handler {
                @Override
                public void handle(StompFrame message) {
                        System.out.println(new String(message.getContent()));
                }
        }
```

```
	public static void main(String [] args) throws Exception {
		// Read environment for config
		SampleConfiguration config = new SampleConfiguration();

		PxgridControl control = new PxgridControl(config);

		// AccountActivate
		while (control.accountActivate() != AccountState.ENABLED) {
			Thread.sleep(60000);
		}
		Console.log("pxGrid controller version=" + control.getControllerVersion());

		// dynamic service lookup
		Console.log("Looking up service com.custom.ise.dynamic");
		Service[] services = control.lookupService("com.custom.ise.dynamic");
		if (services.length == 0) {
			Console.log("dynamic service unavailabe");
			return;
		}

		//gets dynamic service.
		Service service = services[0];
		String wsPubsubServiceName = service.getProperties().get("wsPubsubService");

		//sets topic to be used later as the service to subscribe to.
		String topic = service.getProperties().get("iotTopic");
		Console.log("wsPubsubServiceName=" + wsPubsubServiceName + " iotTopic=" + topic);

		// Pubsub ServiceLookup
		services = control.lookupService(wsPubsubServiceName);
		if (services.length == 0) {
			Console.log("Pubsub service unavailabe");
			return;
		}

		// Select first one which ends up being the pxGrid server node. Should cycle through until
connects.
		Service wsPubsubService = services[0];
		String wsURL = wsPubsubService.getProperties().get("wsUrl");
		Console.log("wsUrl=" + wsURL);

		// pxGrid AccessSecret
		String secret = control.getAccessSecret(wsPubsubService.getNodeName());

		//setting up client manager which will use ssl connection for authentication.
		ClientManager client = ClientManager.createClient();
		SslEngineConfigurator sslEngineConfigurator = new
SslEngineConfigurator(config.getSSLContext());
		sslEngineConfigurator.setHostnameVerifier(new HostnameVerifier() {
			@Override
			public boolean verify(String hostname, SSLSession session) {
				return true;
			}
		});
		client.getProperties().put(ClientProperties.SSL_ENGINE_CONFIGURATOR, sslEngineConfigurator);
		client.getProperties().put(ClientProperties.CREDENTIALS,
				new Credentials(config.getUserName(), secret.getBytes()));

		// WebSocket connect
		StompPubsubClientEndpoint endpoint = new StompPubsubClientEndpoint();
		URI uri = new URI(wsURL);

		//connects the pxGrid client to the pxGrid server
		javax.websocket.Session session = client.connectToServer(endpoint, uri);

		// STOMP connect
		endpoint.connect(uri.getHost());

		// Subscribes to the specific topic of the dynamic service
		StompSubscription subscription = new StompSubscription(topic, new SessionHandler());
		endpoint.subscribe(subscription);
```

```
                SampleHelper.prompt("press <enter> to disconnect...");

                // STOMP disconnect
                endpoint.disconnect("ID-123");
                // Wait for disconnect receipt
                Thread.sleep(3000);

                session.close();
        }
}
```

Device List

This code returns the list of devices as called in the PublishController Code

```
package com.cisco.pxgrid.samples.ise.http;

import com.fasterxml.jackson.annotation.JsonProperty;

public class DeviceList {
        Devices[] list_of_devices;

        public DeviceList(Devices[] device) {
                this.list_of_devices = device;
        }

        @JsonProperty("assets")
        public Devices[] getDeviceList() {
                return list_of_devices;
        }
}
```

Devices

This code defines the asset and custom attributes

```
package com.cisco.pxgrid.samples.ise.http;

import com.fasterxml.jackson.annotation.JsonProperty;
import org.json.simple.JSONObject;
public class Devices {

        private String assetId;
        private String assetIpAddress;
        private String assetMacAddress;
        private String assetName;
        private String assetVendor;
        private String assetSerialNumber;
        private String assetProductId;
        private String assetDeviceType;
        private String assetSwRevision;
        private String assetHwRevision;
        private String assetProtocol;
//  private JSONObject CustomAttributes;
        public Devices(String assetId, String assetName, String assetIpAddress, String assetMacAddress,
String assetVendor, String assetProductId,
                        String assetSerialNumber, String assetDeviceType, String assetSwRevision, String
assetHwRevision, String assetProtocol) {
                this.assetId = assetId;
                this.assetName = assetName;
```

```
                this.assetIpAddress = assetIpAddress;
                this.assetMacAddress = assetMacAddress;
                this.assetVendor = assetVendor;
                this.assetProductId = assetProductId;
                this.assetSerialNumber = assetSerialNumber;
                this.assetDeviceType = assetDeviceType;
                this.assetSwRevision = assetSwRevision;
                this.assetHwRevision = assetHwRevision;
                this.assetProtocol = assetProtocol;
        /*      this.CustomAttributes= new JSONObject();
                this.CustomAttributes.put("test1","value1");
                this.CustomAttributes.put("test2","value2");*/
        }
  /*@JsonProperty("CustomAttributes")
        public JSONObject getCustomAttributes() {
          return CustomAttributes;
        }*/
        @JsonProperty("assetId")
        public String getId() {
                return assetId;
        }

        @JsonProperty("assetIpAddress")
        public String getIpAddress() {
                return assetIpAddress;
        }

        @JsonProperty("assetMacAddress")
        public String getMacAddress() {
                return assetMacAddress;
        }

        @JsonProperty("assetName")
        public String getName() {
                return assetName;
        }

        @JsonProperty("assetVendor")
        public String getVendor() {
                return assetVendor;
        }

        @JsonProperty("assetSerialNumber")
        public String getSerialNumber() {
                return assetSerialNumber;
        }

        @JsonProperty("assetProductId")
        public String getProductId() {
                return assetProductId;
        }

        @JsonProperty("assetDeviceType")
        public String getDeviceType() {
                return assetDeviceType;
        }

        @JsonProperty("assetSwRevision")
        public String getSwRevision() {
                return assetSwRevision;
        }

        @JsonProperty("assetHwRevision")
        public String getHwRevision() {
                return assetHwRevision;
        }

        @JsonProperty("assetProtocol")
        public String getProtocol() {
                return assetProtocol;
        }
}
```

Console.java

This code prints out the time zone

```
package com.cisco.pxgrid.samples.ise.http;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

public class Console {
        public static void log(String message) {
                SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-yy HH:mm:ss.SSS");
                formatter.setTimeZone(TimeZone.getDefault());
                String date = formatter.format(new Date());

                String prepend = date + " [" + Thread.currentThread().getName()
                                                + "-" + Thread.currentThread().getId() + "]: ";
                System.out.println(prepend + message);
        }
}
```

Sample Helper. Java

This code helps with the initial WebSockets Connection

```
package com.cisco.pxgrid.samples.ise.http;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.text.ParseException;
import java.time.OffsetDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Base64;
import java.util.Scanner;

import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLSocketFactory;

import org.apache.commons.io.IOUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
```

```
import com.google.gson.TypeAdapter;
import com.google.gson.stream.JsonReader;
import com.google.gson.stream.JsonToken;
import com.google.gson.stream.JsonWriter;

public class SampleHelper {
        private static Logger logger = LoggerFactory.getLogger(SampleHelper.class);

        public static HttpsURLConnection createHttpsURLConnection(String url, String user, String password,
SSLSocketFactory sslSocketFactory) throws IOException {
                URL conn = new URL(url);
                HttpsURLConnection https = (HttpsURLConnection) conn.openConnection();
                https.setSSLSocketFactory(sslSocketFactory);
                String userPassword = user + ":" + password;
                String encoded = Base64.getEncoder().encodeToString(userPassword.getBytes());
                https.setRequestProperty("Authorization", "Basic " + encoded);
                return https;
        }

        public static String prompt(String msg) {
         System.out.print(msg);
         @SuppressWarnings("resource")
                Scanner scanner = new Scanner(System.in);
         String value = scanner.nextLine();
         if ("".equals(value)) return null;
         return value;
    }

    public static OffsetDateTime promptDate(String msg) throws ParseException {
        String value = prompt(msg);
        if (value == null) return null;
        return OffsetDateTime.parse(value);
        }

        public static void postObjectAndPrint(String url, String user, String password, SSLSocketFactory ssl,
Object postObject) throws IOException {
                Gson gson = new GsonBuilder()
                     .registerTypeAdapter(OffsetDateTime.class, new OffsetDateTimeAdapter())
                     .create();
                postStringAndPrint(url, user, password, ssl, gson.toJson(postObject));
        }

        public static void postStringAndPrint(String url, String user, String password, SSLSocketFactory ssl,
String postData) throws IOException {
                logger.info("postData={}", postData);
                HttpsURLConnection httpsConn = SampleHelper.createHttpsURLConnection(url, user, password,
ssl);
        httpsConn.setRequestMethod("POST");
        httpsConn.setRequestProperty("Content-Type", "application/json");
        httpsConn.setRequestProperty("Accept", "application/json");
        httpsConn.setDoInput(true);
        httpsConn.setDoOutput(true);

                OutputStreamWriter osw = new OutputStreamWriter(httpsConn.getOutputStream());
                osw.write(postData);
                osw.flush();

                int status = httpsConn.getResponseCode();
        logger.info("Response status={}", status);

                if (status < HttpURLConnection.HTTP_BAD_REQUEST) {
                        try (InputStream in = httpsConn.getInputStream()) {
                                String content = IOUtils.toString(in, StandardCharsets.UTF_8);
                        System.out.println("Content: "  + content);
                 }
                }
                else {
                        try (InputStream in = httpsConn.getErrorStream()) {
                                String content = IOUtils.toString(in, StandardCharsets.UTF_8);
                        System.out.println("Content: "  + content);
                 }
                }
        }
```

```
	private static class OffsetDateTimeAdapter extends TypeAdapter<OffsetDateTime> {
		DateTimeFormatter formatter = DateTimeFormatter.ISO_OFFSET_DATE_TIME;
		@Override
		public void write(JsonWriter out, OffsetDateTime value) throws IOException {
			if (value == null) {
				out.nullValue();
				return;
			}
			out.value(formatter.format(value));
		}

		@Override
		public OffsetDateTime read(JsonReader in) throws IOException {
			if (in.peek() == JsonToken.NULL) {
				in.nextNull();
				return null;
			}
			return formatter.parse(in.nextString(), OffsetDateTime::from);
		}
	}
}
```

Stompframe.java

This code handles the STOMP message frame

```
package com.cisco.pxgrid.samples.ise.http;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.text.ParseException;
import java.util.HashMap;
import java.util.Map;

/**
 * This follows STOMP 1.2 specification to parse and generate STOMP frames:
 *   https://stomp.github.io/stomp-specification-1.2.html
 *
 * This single class is self-sufficient handle all STOMP frames.
 *
 * Note for WebSocket:
 * If input comes as WebSocket text type, (WS RFC says Text is UTF-8)
 * server side handling code like Spring TextMessage may convert the bytes to String as UTF-8
 * which maybe the wrong encoding as STOMP frame itself can use other encoding.
 *   e.g. A particular encoding may have bytes: FF FF FF FF FF FF FF FF FF FF... 10, that is completely out
of range for Unicode.
 * Unless STOMP body is also UTF-8, STOMP frame must be sent as binary
 *
 * @author Alan Lei
 */
public class StompFrame {
	public enum Command {
		CONNECT, STOMP, CONNECTED, SEND, SUBSCRIBE, UNSUBSCRIBE, ACK, NACK,
		BEGIN, COMMIT, ABORT, DISCONNECT, MESSAGE, RECEIPT, ERROR;

		private static Map<String, Command> mapOfStringToCommand = new HashMap<>();
		static {
			for (Command command : Command.values()) {
				mapOfStringToCommand.put(command.name(), command);
			}
		}

		public static Command get(String value) {
			return mapOfStringToCommand.get(value);
		}
```

```
}

private Command command;
private Map<String, String> headers = new HashMap<>();
private byte[] content;
private final static int MAX_BUFFER_SIZE = 1024;

public Command getCommand() {
	return command;
}

public void setCommand(Command command) {
	this.command = command;
}

public String getHeader(String name) {
	return headers.get(name);
}

public void setHeader(String name, String value) {
	headers.put(name, value);
}

public Map<String, String> getHeaders() {
	return headers;
}

public byte[] getContent() {
	return content;
}

public void setContent(byte[] content) {
	this.content = content;
}

public void write(OutputStream out) throws IOException {
	out.write(command.name().getBytes());
	out.write('\n');
	for (String name : headers.keySet()) {
		out.write(name.getBytes());
		out.write(':');
		out.write(headers.get(name).getBytes());
		out.write('\n');
	}
	out.write('\n');
	if (content != null) {
		out.write(content);
	}
	out.write(0);
}

private static String readLine(InputStream in) throws IOException, ParseException {
	byte[] line = new byte[MAX_BUFFER_SIZE];
	int index = 0;
	while (index < MAX_BUFFER_SIZE) {
		int b = in.read();
		if (b != -1) {
			if (b == '\n') {
				return new String(line, 0, index);
			}
			if (b != '\r') {
				line[index] = (byte)b;
				index++;
			}
		}
		else {
			// No line found
			return null;
		}
	}
	throw new ParseException("Line too long", MAX_BUFFER_SIZE);
}
```

```
/*
 * Using InputStream instead of Reader because
 * content-length is octet count instead of character count
 */
public static StompFrame parse(InputStream reader) throws IOException, ParseException {
        StompFrame stomp = new StompFrame();

        // Read Command
        String line = readLine(reader);
        Command command = Command.get(line);
        if (command == null) throw new ParseException("Unknown command: " + line, 0);
        stomp.setCommand(command);

        // Read Headers
        int contentLength = -1;
        while ((line = readLine(reader)) != null) {
                if (line.equals("")) break;
                int colon = line.indexOf(':');
                String name = line.substring(0, colon);
                String value = line.substring(colon + 1);
                stomp.setHeader(name, value);
                if (name.equals("content-length")) {
                        contentLength = Integer.parseInt(value);
                }
        }

        // Read Content
        if (contentLength != -1) {
                // content-length is in octets
                byte[] content = new byte[contentLength];
                reader.read(content);
                stomp.setContent(content);
                if (reader.read() != 0) {
                        throw new ParseException("Byte after STOMP Body not NULL", -1);
                }
        }
        else {
                // No content-length. Look for ending NULL byte.
                byte[] buffer = new byte[MAX_BUFFER_SIZE];
                int length = 0;
                while (length < MAX_BUFFER_SIZE) {
                        int b = reader.read();
                        if (b == -1) {
                                throw new ParseException("Premature end of stream", -1);
                        }
                        if (b == 0) {
                                if (length > 0) {
                                        byte[] content = new byte[length];
                                        System.arraycopy(buffer, 0, content, 0, length);
                                        stomp.setContent(content);
                                }
                                // More EOLs may follow, but ignored.
                                return stomp;
                        }
                        buffer[length] = (byte)b;
                        length++;
                }
                throw new ParseException("Frame too long", -1);
        }
        return stomp;
}

@Override
public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("command=" + command);
        sb.append(", headers={");
        for (String name : headers.keySet()) {
                sb.append("'" + name + "':");
                sb.append("'" + headers.get(name) + "',");
        }
        sb.append("}");
        sb.append(", content.length=" + content.length);
```

```
                return sb.toString();
        }
}
```

StompSubscription

This code handles subscription to the STOMP Messaging protocol.

```
package com.cisco.pxgrid.samples.ise;

import java.util.concurrent.atomic.AtomicInteger;

public class StompSubscription {
        public static interface Handler {
                void handle(StompFrame message);
        }

        private static AtomicInteger currentSubscriptionID = new AtomicInteger();
        private String id = Integer.toString(currentSubscriptionID.getAndIncrement());
        private String topic;
        private Handler handler;

        public StompSubscription(String topic, Handler handler) {
                this.topic = topic;
                this.handler = handler;
        }

        public String getId() {
                return id;
        }

        public String getTopic() {
                return topic;
        }

        public Handler getHandler() {
                return handler;
        }

        public StompFrame getSubscribeMessage() {
                StompFrame message = new StompFrame();
                message.setCommand(StompFrame.Command.SUBSCRIBE);
                message.setHeader("destination", topic);
                message.setHeader("id", id);
                return message;
        }
}
```

StompPubSubClientEndpoint

This code handles endpoint client subscription to the STOMP messaging protocol.

```
package com.cisco.pxgrid.samples.ise.http;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.text.ParseException;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
```

```
import javax.websocket.ClientEndpoint;
import javax.websocket.CloseReason;
import javax.websocket.Endpoint;
import javax.websocket.EndpointConfig;
import javax.websocket.MessageHandler;
import javax.websocket.Session;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.cisco.pxgrid.samples.ise.http.StompSubscription.Handler;

@ClientEndpoint
public class StompPubsubClientEndpoint extends Endpoint {
        private static Logger logger = LoggerFactory.getLogger(StompPubsubClientEndpoint.class);

        private volatile Session session;
        private Map<String, StompSubscription> mapOfIdToSubscription = new ConcurrentHashMap<>();

        public void connect(String hostname) throws IOException {
        logger.info("STOMP CONNECT host=" + hostname);
                StompFrame message = new StompFrame();
        message.setCommand(StompFrame.Command.CONNECT);
        message.setHeader("accept-version", "1.2");
        message.setHeader("host", hostname);
        send(message);
        }

        public void disconnect(String receipt) throws IOException {
                logger.info("STOMP DISCONNECT receipt=" + receipt);
                StompFrame message = new StompFrame();
        message.setCommand(StompFrame.Command.DISCONNECT);
        if (receipt != null) {
                message.setHeader("receipt", receipt);
        }
        send(message);
        }

        public void subscribe(StompSubscription subscription) throws IOException {
                logger.info("STOMP SUBSCRIBE topic=" + subscription.getTopic());
                mapOfIdToSubscription.put(subscription.getId(), subscription);
                if (session != null) {
                        StompFrame message = subscription.getSubscribeMessage();
                send(message);
                }
        }

        public void publish(String topic, byte[] content) throws IOException {
                logger.info("STOMP SEND topic=" + topic);
                StompFrame message = new StompFrame();
        message.setCommand(StompFrame.Command.SEND);
        message.setHeader("destination", topic);
        message.setHeader("content-length", Integer.toString(content.length));
        message.setContent(content);
        System.out.println(message);
        send(message);
        }

        private void send(StompFrame message) throws IOException {
                if (session != null) {
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                message.write(baos);
                // Send as binary
                session.getBasicRemote().sendBinary(ByteBuffer.wrap(baos.toByteArray()));
                }
        }

        public void waitForOpen() throws InterruptedException {
                synchronized (this) {
                        while (session == null) {
                                this.wait();
                        }
                }
```

```
}

private void onStompMessage(StompFrame stomp) {
        switch (stomp.getCommand()) {
        case CONNECTED:
                String version = stomp.getHeader("version");
                logger.info("STOMP CONNECTED version={}", version);
                break;
        case RECEIPT:
                String receiptId = stomp.getHeader("receipt-id");
                logger.info("STOMP RECEIPT id={}", receiptId);
                break;
        case MESSAGE:
                String id = stomp.getHeader("subscription");
                StompSubscription subscription = mapOfIdToSubscription.get(id);
                Handler handler = subscription.getHandler();
                if (handler != null) {
                        handler.handle(stomp);
                }
                break;
        case ERROR:
                // Server will close connect on ERROR according to STOMP specification
                logger.info("STOMP ERROR stomp={}", stomp);
                break;
        default:
                // Ignore others
                break;
        }
}

private class TextHandler implements MessageHandler.Whole<String> {
        @Override
        public void onMessage(String message) {
        try {
                StompFrame stomp = StompFrame.parse(new ByteArrayInputStream(message.getBytes()));
                onStompMessage(stomp);
        } catch (IOException | ParseException e) {
                logger.error("onMessage", e);
        }
        }
}

private class BinaryHandler implements MessageHandler.Whole<InputStream> {
        @Override
        public void onMessage(InputStream in) {
        try {
                        StompFrame stomp = StompFrame.parse(in);
                        onStompMessage(stomp);
                } catch (IOException | ParseException e) {
                logger.error("onMessage", e);
                }
        }
}

@Override
public void onOpen(Session session, EndpointConfig cfg) {
        logger.info("WS onOpen");
        this.session = session;
        try {
        session.addMessageHandler(new TextHandler());
        session.addMessageHandler(new BinaryHandler());

        for (StompSubscription subscription : mapOfIdToSubscription.values()) {
                StompFrame message = subscription.getSubscribeMessage();
                                send(message);
                }
        } catch (IOException e) {
                logger.error("onOpen", e);
        }
        synchronized (this) {
                this.notifyAll();
        }
}
```

```
	@Override
	public void onClose(Session session, CloseReason closeReason) {
		logger.info("WS onClose closeReason code={} phrase={}", closeReason.getCloseCode(),
closeReason.getReasonPhrase());
		this.session = null;
	}

	@Override
	public void onError(Session session, Throwable thr) {
		logger.info("WS onError thr={}", thr.getMessage());
		this.session = null;
	}
}
```

Endpoint Asset Configuration

Service: com.cisco.endpoint.asset

This is endpoint asset service topic

Service properties

Name	Description	Example
restBaseUrl	The base URL for REST APIs	https://[ind-host1]:8910/pxgrid/ind/asset
wsPubsubService	The WebSocket Pubsub service name	com.cisco.ise.pubsub
assetTopic	Topic for asset events	/topic/com.cisco.endpoint.asset

assetTopic WS Stomp message:

```
{
  "asset": Asset object,
  "opType": operation type
}
```

"opType" will be one of the following strings: CREATE/UPDATE/DELETE

Sample Response:


```
{
    "asset": {
        "assetId": 20101,
        "assetName": "switchxbx208",
        "assetIpAddress": "172.27.162.172",
        "assetMacAddress": "00:1b:1b:0f:19:23",
        "assetVendor": "Siemens",
        "assetProductId": "6GK5 208-0BA10-2AA3",
        "assetSerialNumber": "VPB9559430",
        "assetDeviceType": "Switch",
        "assetSwRevision": "",
        "assetHwRevision": "",
        "assetProtocol": "PROFINET",
        "assetConnectedLinks": [
            {
                "assetId": "105",
                "assetName": "IE5k-162-162.cisco.com",
                "assetIpAddress": "172.27.162.162",
                "assetPortName": "GigabitEthernet1/4",
                "assetDeviceType": "Switch"
            }
        ],
        "assetCustomAttributes": [ ]
    },
    "opType": "CREATE"
}
```

POST [restBaseUrl]/getAssets

This is used to get all Endpoint devices.

Request URL: [restBaseUrl]/getAssets

Request Method: POST

Content-Type: application/json

Accept: application/json

Authorization: Basic [nodeName]:[secret]

Label	Description
[restBaseUrl]	Obtain by ServiceLookup of com.cisco.endpoint.asset
[nodeName]	pxGrid node name
[secret]	Obtain via AccessSecret

Request:

```
{
  "limit": integer (optional),
  "offset": integer (optional)
}
```

Parameter	Description	Default Value
limit	Number of records per page	500
offset	Zero based page index	0

Response:

```
{
 "assets": [
```

```
    array of Asset objects
  ]
}
```

"Asset" Object

Name	Type	Description
assetId	Long	IND DB id for the asset
assetName	String	Asset name
assetIpAddress	String	IP address
assetMacAddress	String	MAC address
assetVendor	String	Manufacturer
assetProductId	String	Product Code
assetSerialNumber	String	Serial Number
assetDeviceType	String	Device Type
assetSwRevision	String	S/W Revision number
assetHwRevision	String	H/W Revision number
assetProtocol	String	
assetConnectedLinks		Array of Link object
assetCustomAttributes		Array of name-value pair in the form of Pair object

Asset Custom Attributes

Key Name	Value Type	Description	Supported IND Version
assetGroup	String	Fully qualified group name assigned to the Asset	1.3, 1.4
assetTag	String	Security Tag assigned to the Asset	1.4

"Link" Object

Name	Type	Description
assetId	Long	IND DB id for the connected asset
assetName	String	Connected Asset name
assetIpAddress	String	Connected Asset IP address
assetPortName	String	Name of the interface that the asset is connected to
assetDeviceType	String	Device Type

"Pair" Object

Name	Type	Description
key	String	Custom attribute name
value	String	Custom attribute value

Sample Response:

```
{
    "assets": [
        {
            "assetId": "118",
            "assetName": "172.27.162.154",
            "assetIpAddress": "172.27.162.154",
            "assetMacAddress": "ec:e5:55:7d:4a:2c",
            "assetVendor": "Hirschmann, a Belden brand",
            "assetProductId": "Hirschmann RSR30-L2",
            "assetSerialNumber": "8210988",
            "assetDeviceType": "Switch",
            "assetSwRevision": "",
            "assetHwRevision": "",
            "assetProtocol": "CIP",
            "assetConnectedLinks": [ ],
            "assetCustomAttributes": [ ]
        },
        {
            "assetId": "106",
            "assetName": "172.27.162.171",
            "assetIpAddress": "172.27.162.171",
            "assetMacAddress": "e4:90:69:a1:14:27",
            "assetVendor": "Rockwell Automation/Allen-Bradley",
            "assetProductId": "1734-AENTR/B Ethernet Adapter",
            "assetSerialNumber": "1615510289",
            "assetDeviceType": "EtherNet/IP Node",
            "assetSwRevision": "",
            "assetHwRevision": "",
            "assetProtocol": "CIP",
            "assetConnectedLinks": [
                {
                    "assetId": "109",
                    "assetName": "IE5k-162-165.cisco.com",
                    "assetIpAddress": "172.27.162.165",
                    "assetPortName": "GigabitEthernet1/9",
                    "assetDeviceType": "Switch"
                }
            ],
            "assetCustomAttributes": [ ]
        }
    ]
}
```


POST [restBaseUrl]/getSyncStatus

This is used to get the sync status of the IND Pxgrid Service.

Request URL: [restBaseUrl]/getSyncStatus

Request Method: POST

Content-Type: application/json

Accept: application/json

Authorization: Basic [nodeName]:[secret]

Label	Description
[restBaseUrl]	Obtain by ServiceLookup of com.cisco.endpoint.asset
[nodeName]	pxGrid node name
[secret]	Obtain via AccessSecret

Request:

```
{
```

```
}
```

Response:

```
{
IN_SYNC
}
```

Response will be one of the following strings: DISABLED/IN_SYNC/OUT_OF_SYNC