The bridge to possible

Working with pxGrid 2.0 APIs

Viktor Bobrov, Sr. Technical Leader
@securityccie

Einar Nilsen-Nygaard, Principal Engineer
@einarnn

Agenda

- Introduction to pxGrid
- pxGrid Architecture
- pxGrid APIs
- Authentication
- REST API
- Websockets
- CLI Client

Benefits

- Simplify operations
- Enhance Controls
- Improve Response

Introduction to pxGrid

Context: Build, Summarize, Exchange

Visibility and Access Control

ISE builds context and applies access control restrictions to users and devices

Who
What
When
How
Where
Posture
Threat
Vulnerability

Security Groups

pxGrid/Cloud
API
Syslog

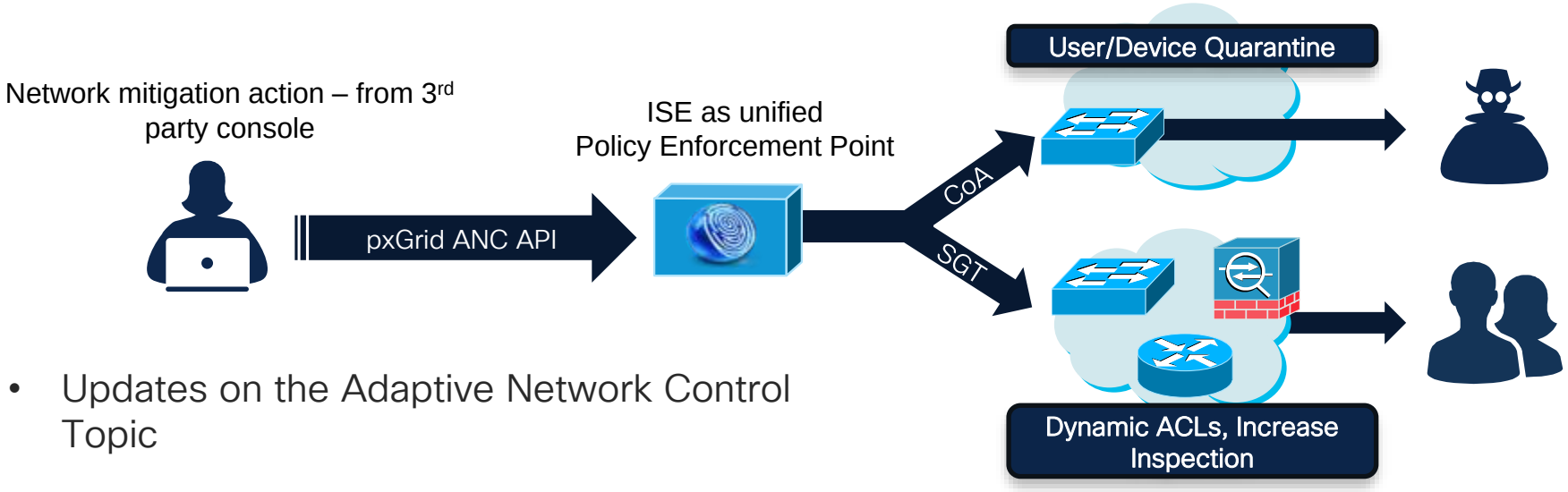
Context Reuse

By eco-system partners for analysis & control

Secure Network Analytics
Secure Firewall
DNAC
+ 3rd Party Partners

Adaptive Network Control (Rapid Threat Containment)

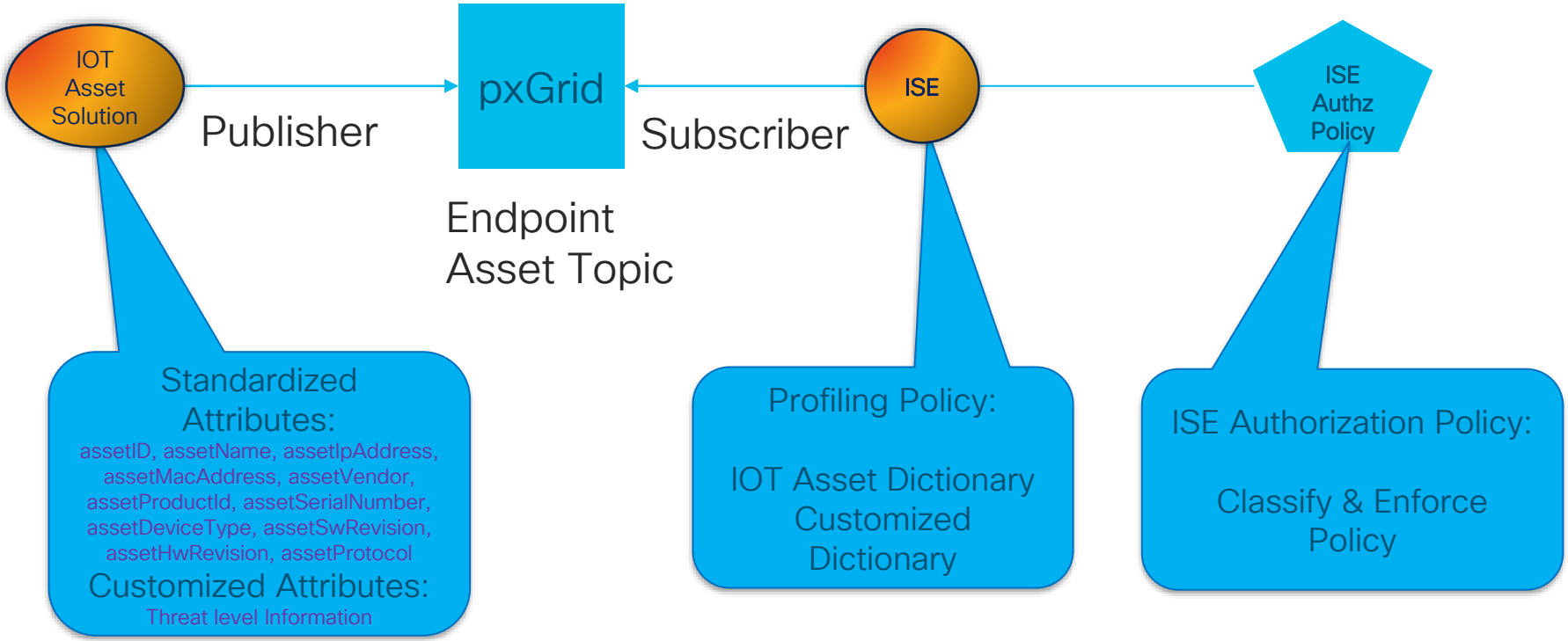
- Enforce mitigation actions based on an organization's security policy
- Uses ISE pxGgrid ANC policies



- Updates on the Adaptive Network Control Topic

pxGrid 2.0 Components

pxGrid Context-In (Culinda, Cylera, Ordr, Asimily, Armis, Nozomi, Radiflow)



ISE & pxGrid Partner Ecosystem

Benefits

- Simplify operations
- Enhance Controls
- Improve Response

Architecture

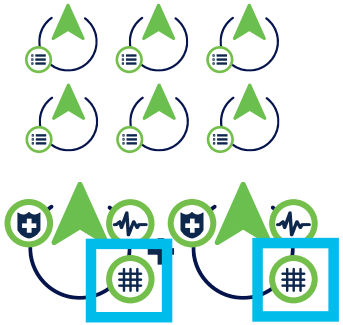
ISE Deployments and pxGrid Nodes

PAN MNT PSN pxGrid

Small HA Deployment

2 x (PAN+MNT+PSN)

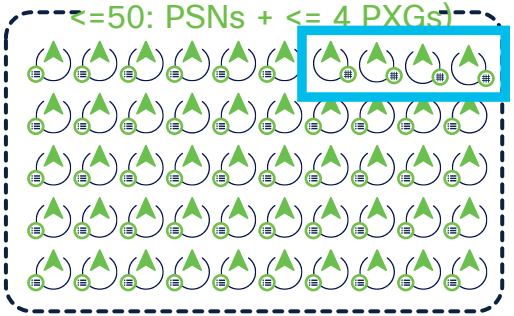
Small

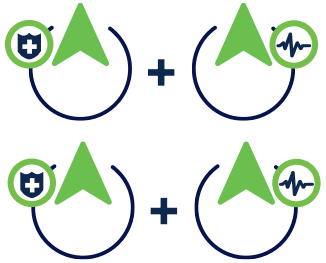


Medium Multi-node Deployment

2 x (PAN+MNT+PXG), <= 6 PSN

Medium



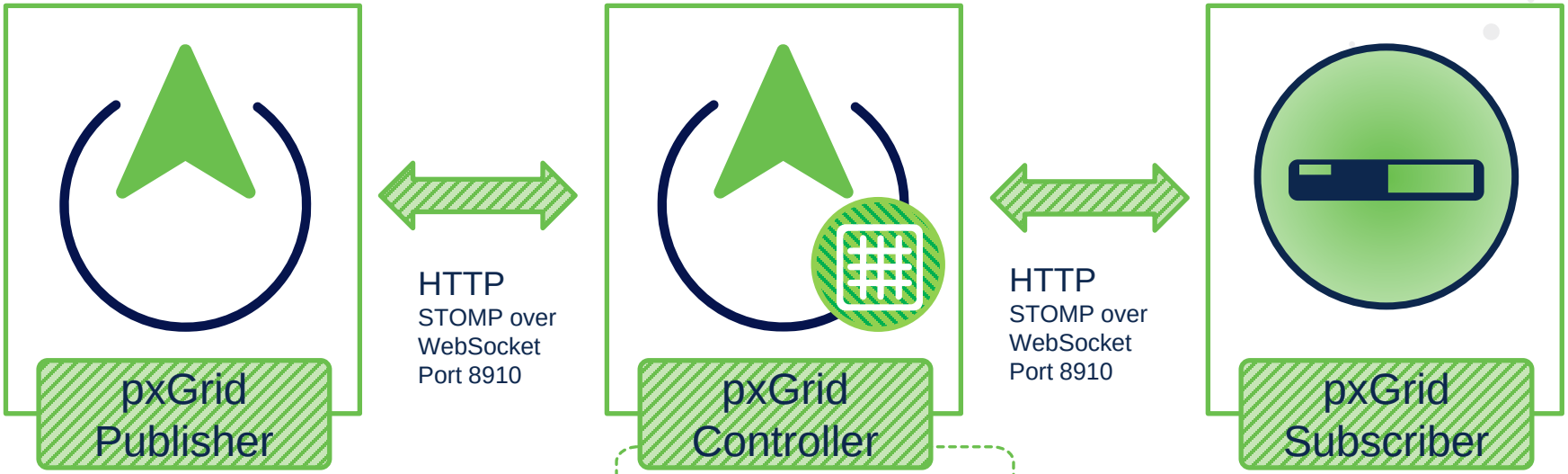


Large Deployment

2 PAN, 2 MNT, <=50: PSNs + <= 4 PXGs

Large

pxGrid 2.0 Components (ISE 3.1+)

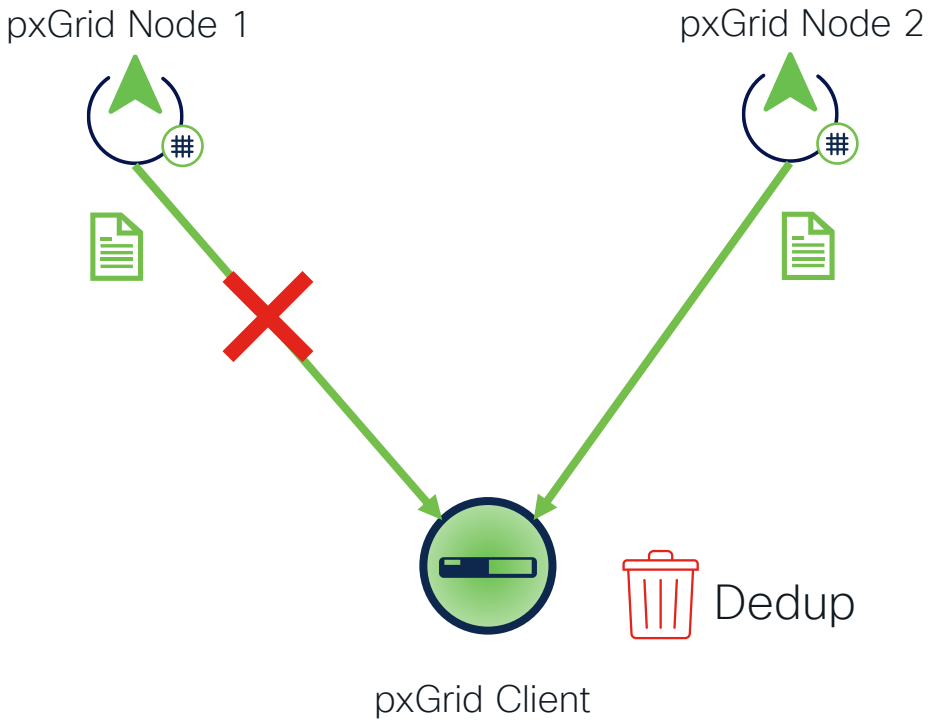


Listens on TCP ports:
7400: Connection for internal processes (internal only)
8910: service check (**all** nodes), pub-sub (pxGrid node)
pxGrid 2.0 nodes always active (not shown in CLI)

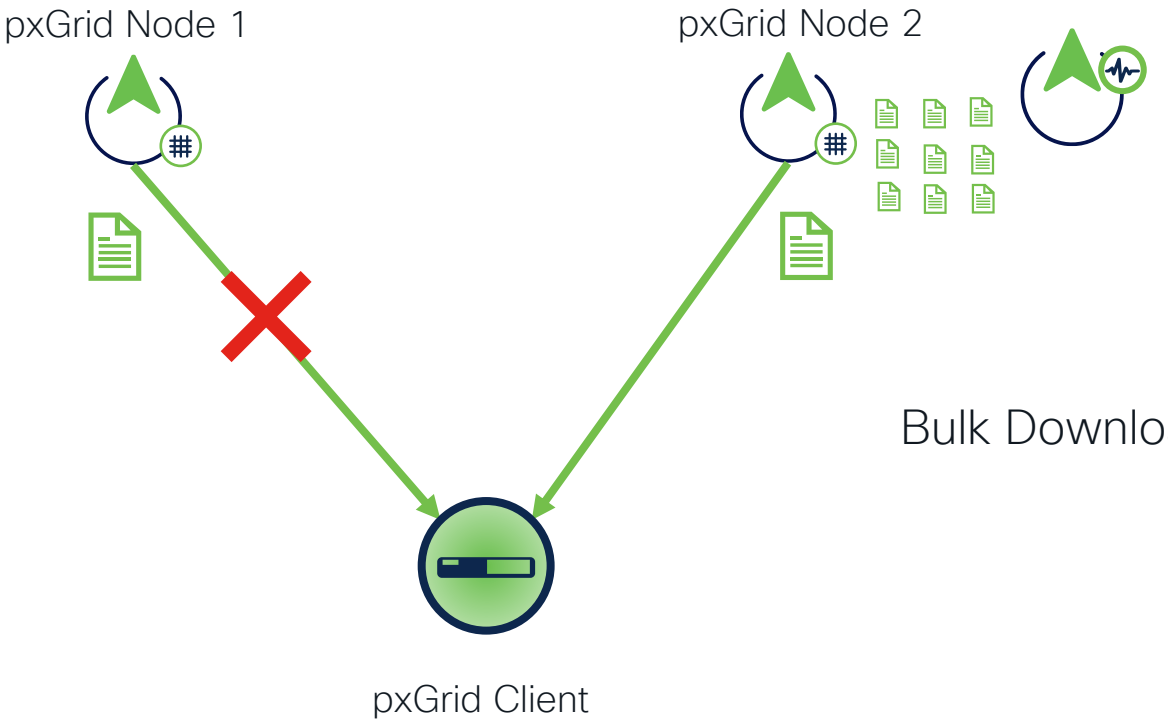
ISE versions and improvement for integrations

	ISE 2.4-2.7	ISE 3.0	ISE 3.1	ISE 3.2
pxGrid version	v1+v2	v1+v2	v2	v2
pxGrid Loss Detection	NA	Yes	Yes	Yes
API Gateway	NA	Yes	Yes	Yes
pxGrid Cloud (ERS API+pxGrid)	NA	NA	Yes	Yes
pxGrid Direct	NA	NA	NA	Yes

pxGrid HA Active/Active



pxGrid HA Active/Passive



Bulk Download from MnT

pxGrid Client HA Options Active-Active

Client uses pxGrid service discovery to determine at least two nodes (3 recommended) that is capable of servicing API requests and two nodes capable of delivering session notifications and any other events

Client registers for session notifications with two pxGrid nodes. Each event received twice under normal conditions, and client deduplicates events. (ISE 2.4-2.7, 3.x adds data loss mechanism)

Client performs initial session directory sync

Under single ISE pxGrid node failure client will lose connection, but still have active connection

If >2 pxGrid nodes in deployment, client can redo service discovery register with alternate node to maintain redundancy

pxGrid Client HA Options
Active-Standby

Client uses pxGrid service discovery to determine a node that is capable of servicing API requests and node capable of delivering session notifications and any other events

Client registers for session notifications with one pxGrid node.

Client performs initial session directory sync

On failure of pxGrid connection, client can discover and attach to alternate node and perform both event subscription and initial sync again

- Initial sync for sessions may be optimized by using the timestamp of the last received event to restrict the scope of session directory query

Client can use RFC-defined WebSocket Ping to check connection liveness and trigger failover logic

Active-Active vs Active-Standby Pros & Cons

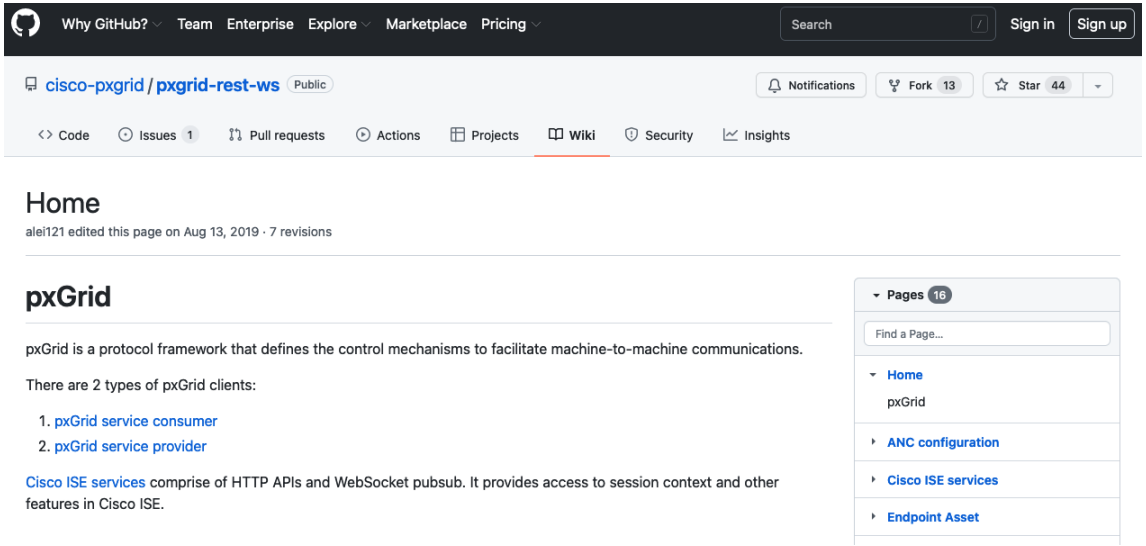
Active-Active Pros	Active-Active Cons	Active-Standby Pros	Active-Standby Cons
No events missed by client	Twice number of events delivered to client	Simpler code in client (Basic sync process required by Active-Active as well)	Client needs to spend time resyncing on pxGrid node failure
No resync delay on pxGrid node failure	Client needs to deduplicate events (use loss detection in 3.x)	Less load on client	Resync will load ISE MNT node

pxGrid APIs

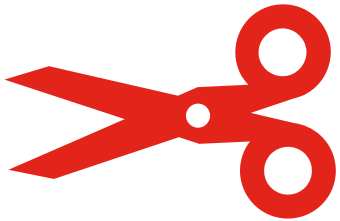
pxGrid 2.0 Components
https://cs.co/pxGrid-github/wiki

pxGrid 2.0 Services

- SessionDirectory
- RadiusFailure
- Profiler Configuration
- System Health
- MDM
- ANC Status
- TrustSec
- TrustSec Configuration
- TrustSec SXP
- Endpoint Asset



Python Code Disclaimer



We show code snippets.
Full examples are available on GitHub.

For Reference indicates repetitive content w
won't spend much time on

REST API Calls

- TCP/8910
- Atomic operations
- Used to retrieve and post data
- All requests are POST even when no data is posted.
- pxGrid allows clients to discover where to request data from
- Data requested directly from publishing nodes. Eg. Admin and MnT

Pub/Sub

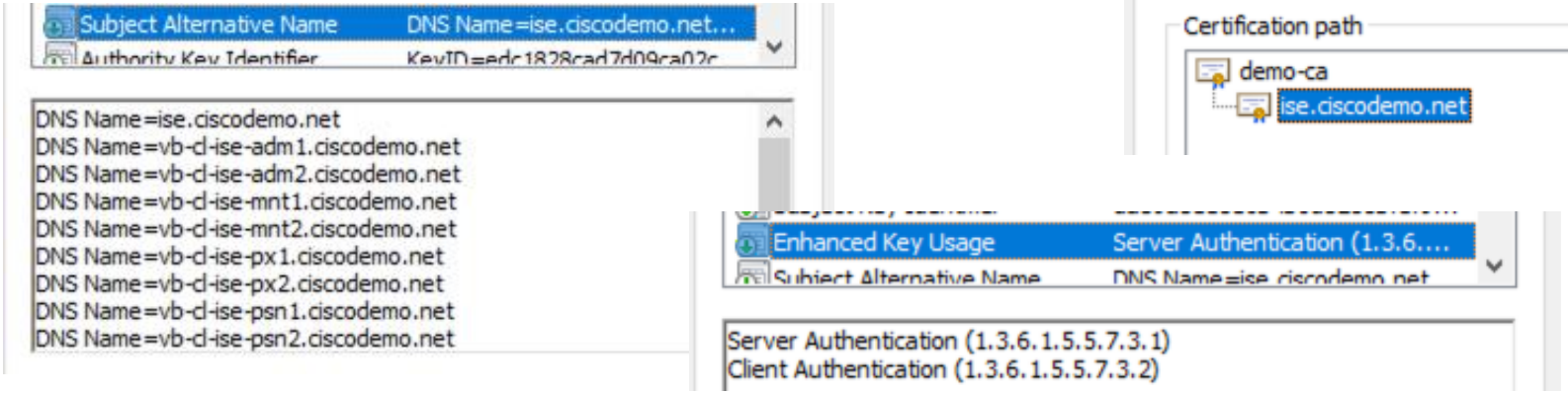
- Also TCP/8910, upgraded to a websocket
- STOMP used over the websocket
- Publishers send data to topics
- Subscribers subscribe to topics and receive data
- pxGrid allows clients to discover where subscribers can connect to a topic
- All websocket connections are to pxGrid nodes

ISE Topology for this Session

Authentication

Certificates – ISE Certificates

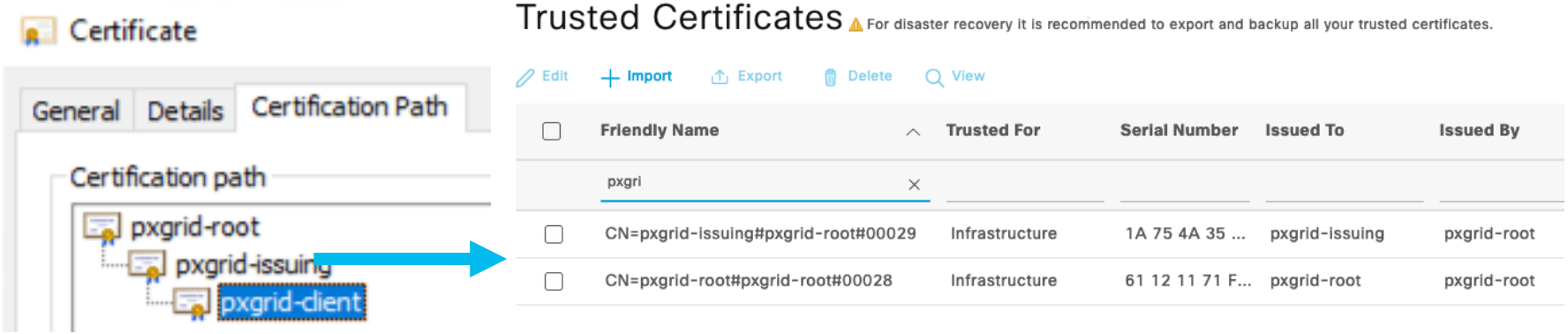
- Admin and MnT nodes need the **pxGrid certificate**. Not just pxGrid nodes!
- Recommended to have certificates from Corporate CA
- Certificates must have Client and Server Authentication EKUs



Certificates – Client Certificates

- Not needed if using password authentication
- FQDN is not validated by ISE
- You can even use the ISE pxGrid certificate on the client
- Client and ISE certificates do not need to be from the same CA chain
- Client certificate also must have Client and Server EKUs

Certificates - Trust

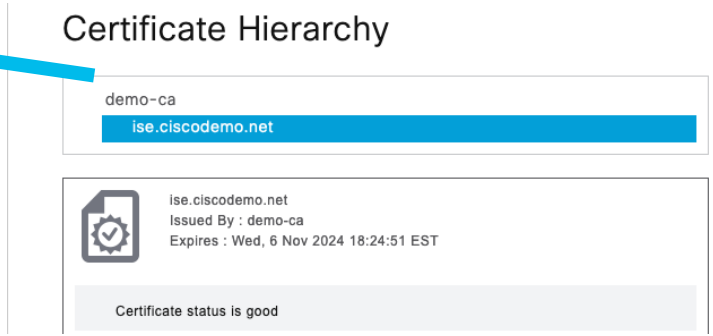


ISE must trust client roots. N/A when using password authentication

$ curl https://vb-cl-ise-px1.ciscodemo.net:8910 --cacert demo-ca.cer

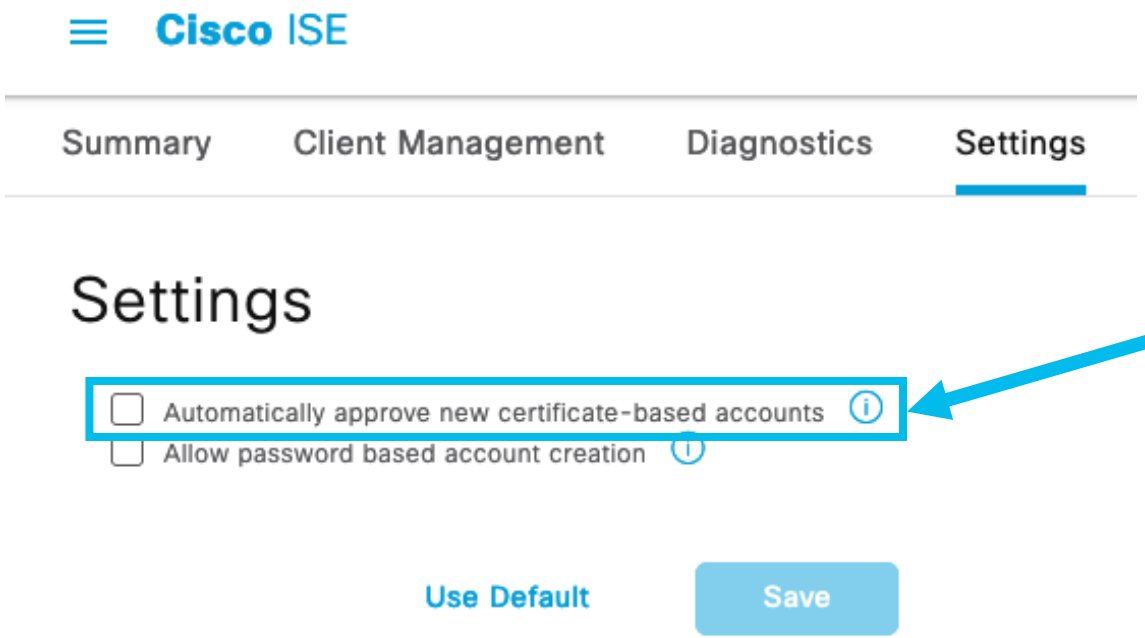
$ curl -k https://vb-cl-ise-px1.ciscodemo.net:8910

Client should trust ISE's root
Ignoring trust **in the lab** works too



Client Approval

All clients must be approved before being able to use pxGrid



Clients can be auto-approve
Still requires client certs

Authentication Flow – Certificate

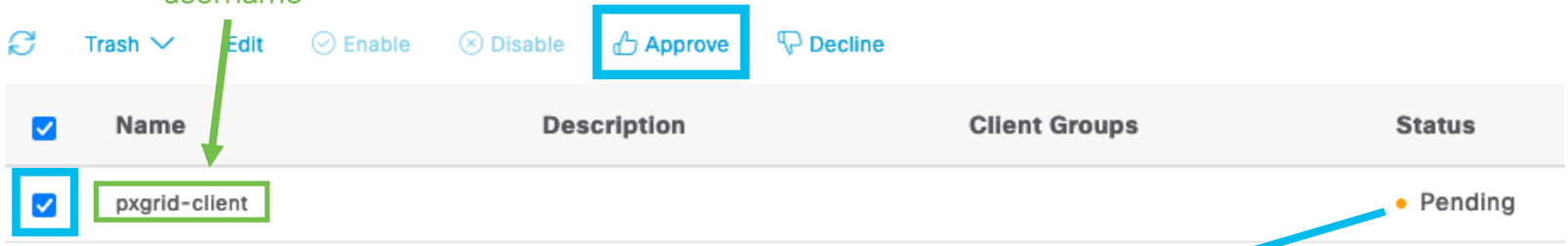
* Once account is approved, steps can be skipped. They verify that account is still ENABLED

Authentication With Certificate

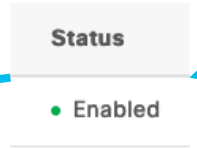
```
$ curl -s https://pxGrid-client:none@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccountActivate \
--cacert ../demo-ca.cer --cert pxGrid-client.crt --key pxGrid-client.key -d '{}' \
-H 'Content-Type: application/json' | jq –M

{
  "accountState": "PENDING",
  "version": "2.0"
}
```

Password is not checked

username




```
{
  "accountState": "ENABLED",
  "version": "2.0"
}
```



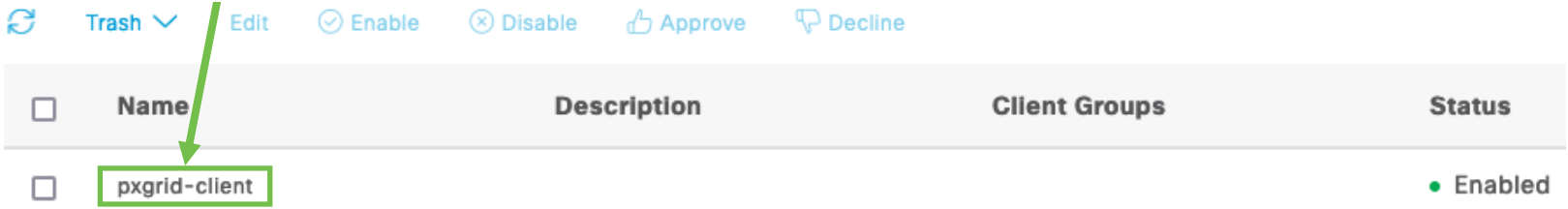
Accounts can be approved manually or using an API:
PUT /ers/config/pxGridNode/name/{nodeName}/approve

Authentication With Certificate and Auto Approval

```
$ curl -s https://pxGrid-client:none@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccountActivate \
--cacert ../demo-ca.cer --cert pxGrid-client.crt --key pxGrid-client.key -d '{}' \
-H 'Content-Type: application/json' | jq –M

{
  "accountState": "ENABLED",
  "version": "2.0"
}
```

username

Password is **not** checked



Client goes into ENABLED immediately

Python Example - Certificate

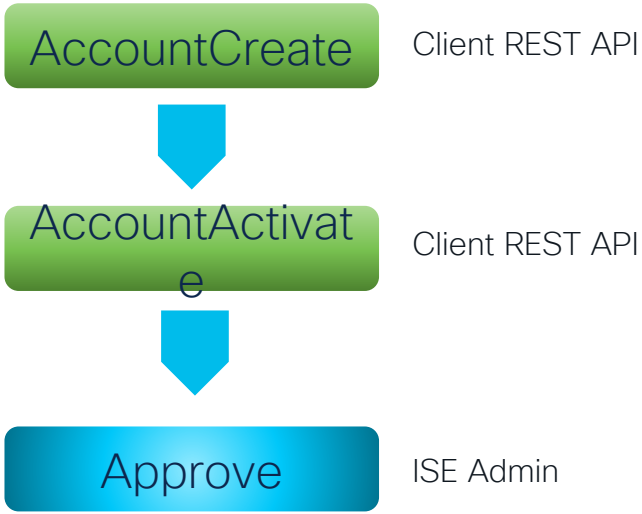
```
import requests
import json
from time import sleep

while True:
    r=requests.post(f"https://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccountActivate",
        cert=("pxGrid-client.crt","pxGrid-client.key"),
        verify="demo-ca.cer",
        auth=("pxGrid-client","none"),
        json={}
    )
    r.raise_for_status()
    json_response=r.json()
    print(json.dumps(json_response,indent=2))
    if json_response["accountState"]=="ENABLED":
        print("Account Approved")
        break
    sleep(60)
```

Warning:

The private key to your local certificate *must* be unencrypted. Currently, Requests does not support using encrypted keys.

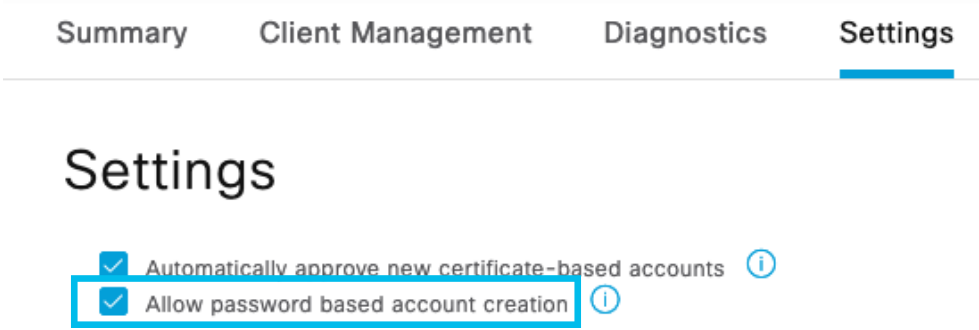
* From https://requests.readthedocs.io/en/latest/user/advanced/

Authentication Flow – Password



* Once account is approved, steps can be skipped. They verify that account is still ENABLED

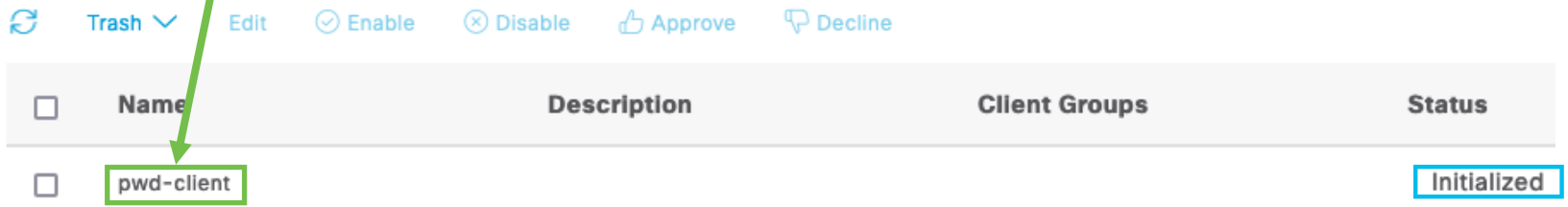
Authentication with Password 1/3



Authentication with Password 2/3

```
$ curl -s https://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccountCreate \
--cacert ../demo-ca.cer -d '{"nodeName": "pwd-client"}' \
-H 'Content-Type: application/json' | jq -M
{
  "nodeName": "pwd-client"
  "password": "F7d22FCc46vxwRyr",
  "userName": "pwd-client"
}
```

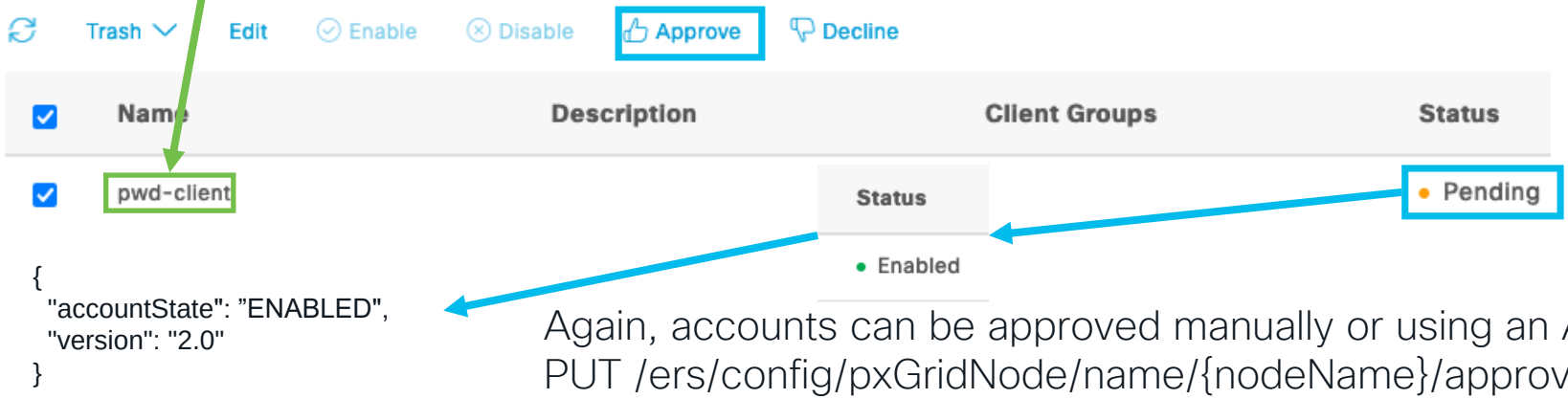
ISE generates random password and Status is initialized. New password is generated on ever



Authentication with Password 3/3

```
$ curl -s https://pwd-client:F7d22FCc46vxwRyr@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccountActivate \
--cacert ../demo-ca.cer -d '{}' \
-H 'Content-Type: application/json' | jq -M
{
  "accountState": "PENDING",
  "version": "2.0"
}
```

Once account is used with AccountActivate, AccountCreate cannot be used.



Again, accounts can be approved manually or using an API:
PUT /ers/config/pxGridNode/name/{nodeName}/approve

Python Example - Password

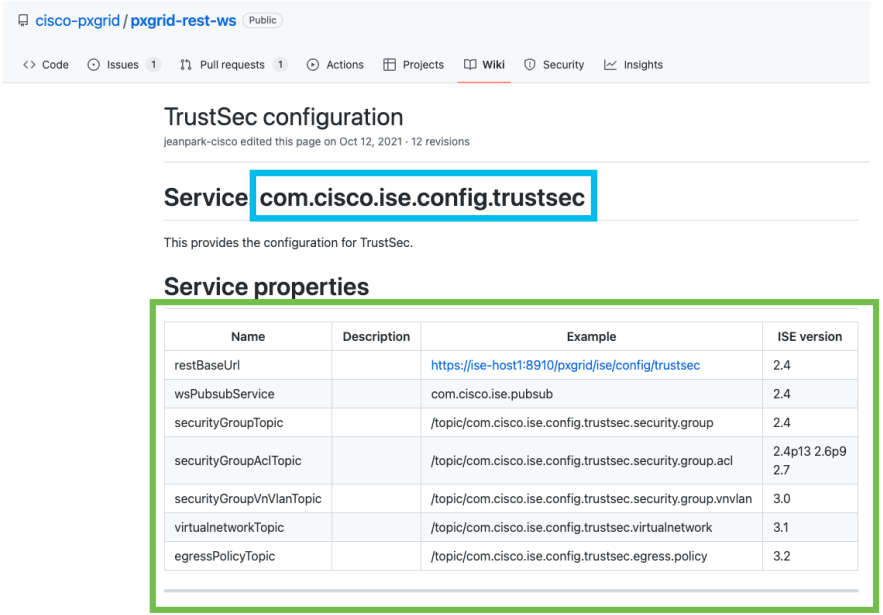
```
import requests
import json
from time import sleep
r=requests.post(f"https://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccountCreate",
    verify="demo-ca.cer",
    json={
        "nodeName": "pwd-client"
    }
)
r.raise_for_status()
password=r.json()["password"]
while True:
    r=requests.post(f"https://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccountActivate",
        verify=".demo-ca.cer",
        auth=("pwd-client",password),
        json={}
    )
    r.raise_for_status()
    json_response=r.json()
    print(json.dumps(json_response,indent=2))
    if json_response["accountState"]=="ENABLED":
        print(f"Account Approved. Password is {password}")
        break
    sleep(60)
```

Password must be saved. Cannot be retrieved from ISE later!

REST API

API Reference Example

https://github.com/cisco-pxGrid/pxGrid-rest-ws/wiki/TrustSec-configuration



cisco-pxgrid / pxgrid-rest-ws Public

Code | Issues 1 | Pull requests 1 | Actions | Projects | Wiki | Security | Insights

TrustSec configuration

jeanpark-cisco edited this page on Oct 12, 2021 · 12 revisions

Service com.cisco.ise.config.trustsec

This provides the configuration for TrustSec.

Service properties

Name	Description	Example	ISE version
restBaseUrl		https://ise-host1:8910/pxgrid/ise/config/trustsec	2.4
wsPubsubService		com.cisco.ise.pubsub	2.4
securityGroupTopic		/topic/com.cisco.ise.config.trustsec.security.group	2.4
securityGroupAclTopic		/topic/com.cisco.ise.config.trustsec.security.group.acl	2.4p13 2.6p9 2.7
securityGroupVnVlanTopic		/topic/com.cisco.ise.config.trustsec.security.group.vnvlan	3.0
virtualnetworkTopic		/topic/com.cisco.ise.config.trustsec.virtualnetwork	3.1
egressPolicyTopic		/topic/com.cisco.ise.config.trustsec.egress.policy	3.2

HTTP APIs

POST [restBaseUrl]/getSecurityGroups

This is used to get security groups. The security group `id` can be specified for a particular security group. If not specified, all existing security groups are returned. These can be filtered by remaining optional parameters.

POST [restBaseUrl]/getSecurityGroupAcls

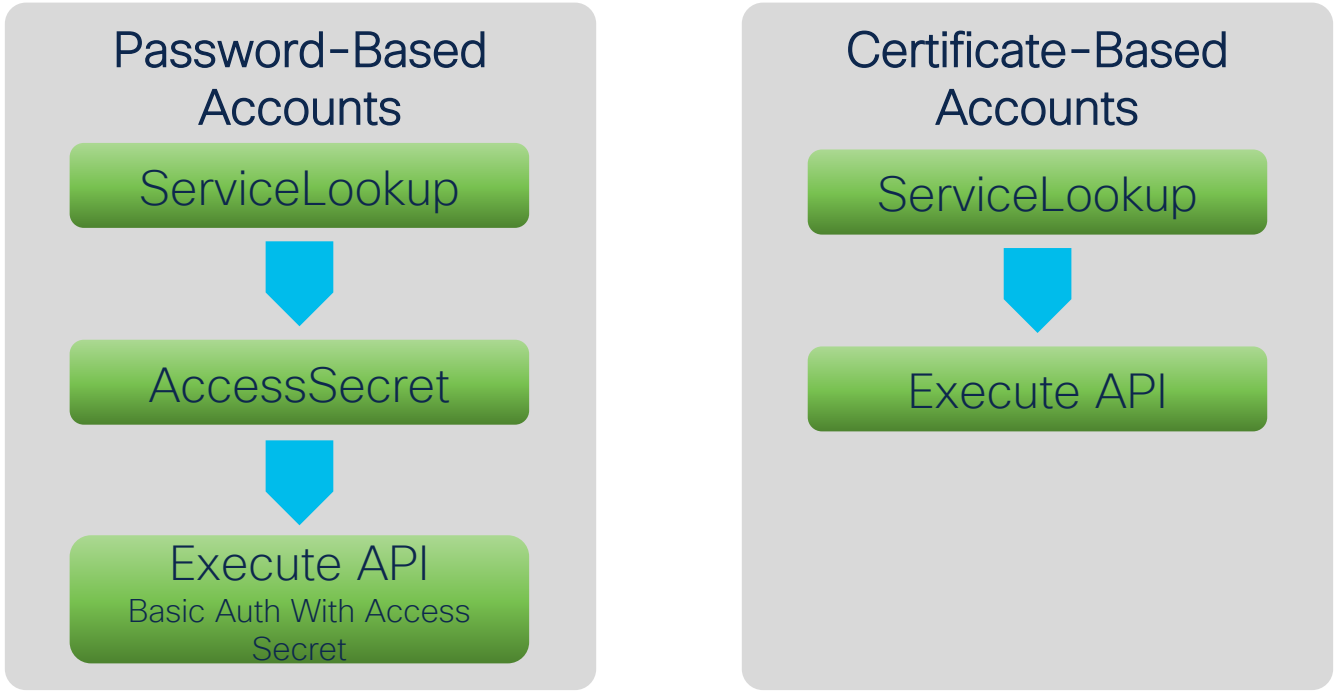
This is used to get security group ACLs. The `id` for security group ACLs can be specified. If not specified, all security group ACLs are returned.

Request

```
{
  // Request to filter results
  "id": string (optional),
  "startIndex": int (optional),
  "recordCount": int (optional),
  "startTimestamp": ISO8601 Datetime (optional),
  "endTimestamp": ISO8601 Datetime (optional),
}
```

- `id` -- id of desired record
- `startIndex` -- first index to begin returning records
- `recordCount` -- number of records to return
- `startTimestamp` (inclusive) -- filters existing and deleted records starting from given time
- `endTimestamp` (inclusive) -- filters existing records up to given time. Also filters deleted records if `startTimestamp` is provided

REST API Flow



ServiceLookup – TrustSec

Password-Based Account Example

```
$ curl -s https://pxGrid-client:F7d22FCc46vxwRyr@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/ServiceLookup \
-d '{"name":"com.cisco.ise.config.trustsec"}' \
--cacert ../demo-ca.cer -H 'Content-Type: application/json' | jq -M
{
  "services": [
    {
      "name": "com.cisco.ise.config.trustsec",
      "nodeName": "~ise-admin-vb-cl-ise-adm1",
      "properties": {
        "virtualnetworkTopic": "/topic/com.cisco.ise.config.trustsec.virtualnetwork",
        "wsPubsubService": "com.cisco.ise.pubsub",
        "restBaseUrl": "https://vb-cl-ise-adm1.ciscodemo.net:8910/pxgrid/ise/config/trustsec",
        "securityGroupVnVlanTopic": "/topic/com.cisco.ise.config.trustsec.security.group.vnvlan",
        "securityGroupTopic": "/topic/com.cisco.ise.config.trustsec.security.group",
        "egressPolicyTopic": "/topic/com.cisco.ise.config.trustsec.egress.policy",
        "securityGroupAclTopic": "/topic/com.cisco.ise.config.trustsec.security.group.acl"
      }
    }
  ]
}
```

Password from AccountCreate

We need these for the next step

AccessSecret

Only Required For Password-Based Accounts

```
$ curl -s https://pxGrid-client:F7d22FCc46vxwRyr@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccessSecret \
-d '{"peerNodeName":"~ise-admin-vb-cl-ise-adm1"}' \
--cacert ../demo-ca.cer -H 'Content-Type: application/json' |jq -M
{
  "secret": "iGYcSUPZgum4xJnz"
}
```

Password from AccountCreate

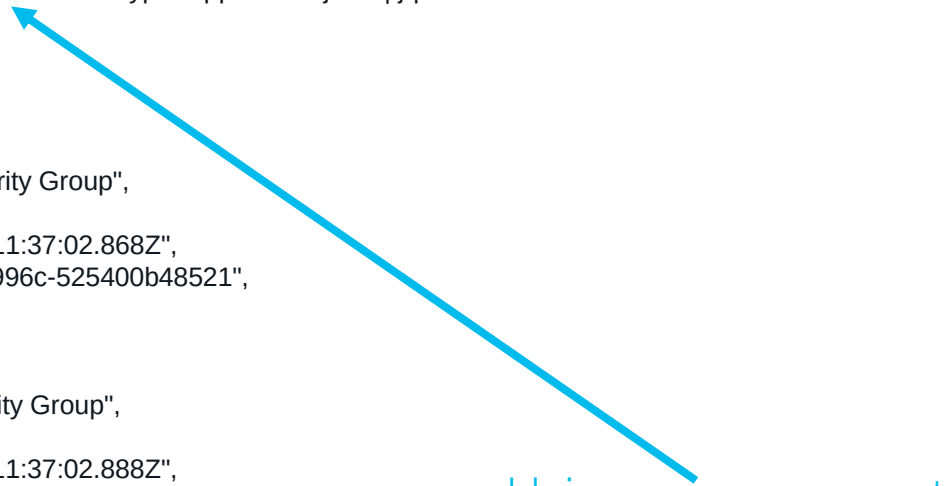
Node name from last step

We need this for the next step if using password-based accounts

Execute API – getSecurityGroups

Password-Based Account Example

```
$ curl -s https://pxGrid-client:iGYcSUPZgum4xJnz@vb-cl-ise-adm1.ciscodemo.net:8910/pxgrid/ise/config/trustsec/getSecurityGroups \
-d '{}' --cacert ../demo-ca.cer -H 'Content-Type: Application/json' | jq -M
{
  "totalCount": "17",
  "version": "1.0.0",
  "securityGroups": [
    {
      "description": "Auditor Security Group",
      "tag": 9,
      "timestamp": "2022-08-30T11:37:02.868Z",
      "id": "934557f0-8c01-11e6-996c-525400b48521",
      "name": "Auditors"
    },
    {
      "description": "BYOD Security Group",
      "tag": 15,
      "timestamp": "2022-08-30T11:37:02.888Z",
      "id": "935d4cc0-8c01-11e6-996c-525400b48521",
      "name": "BYOD"
    },
  ],
  "deletedSecurityGroups": []
}
```



* Output trimmed

Python Example – getSecurityGroups

Password-Based Account Example

```
import requests
import json

r=requests.post(f"https://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/ServiceLookup",
    verify="demo-ca.cer",
    auth=("pxGrid-client","F7d22FCc46vxwRyr"),
    json={
        "name": "com.cisco.ise.config.trustsec"
    }
)
r.raise_for_status()
service_info=r.json()["services"][0]
node_name=service_info["nodeName"]
rest_url=service_info["properties"]["restBaseUrl"]
r=requests.post(f"https://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccessSecret",
    verify="demo-ca.cer",
    auth=("pxGrid-client", "F7d22FCc46vxwRyr"),
    json={
        "peerNodeName": node_name
    }
)
r.raise_for_status()
secret=r.json()["secret"]
```

```
r=requests.post(f"{rest_url}/getSecurityGroups",
    verify="demo-ca.cer",
    auth=("pxGrid-client",secret),
    json={}
)
r.raise_for_status()
print(json.dumps(r.json(),indent=2))
```

ServiceLookup – ANC

Certificate-Based Account Example

```
$ curl -s https://pxGrid-client:None@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/ServiceLookup \
--cert pxGrid-client.crt --key pxGrid-client.key -d '{"name": "com.cisco.ise.config.anc"}' \
--cacert ../demo-ca.cer -H 'Content-Type: application/json' | jq -M
{
  "services": [
    {
      "name": "com.cisco.ise.config.anc",
      "nodeName": "~ise-admin-vb-cl-ise-mnt2",
      "properties": {
        "wsPubsubService": "com.cisco.ise.pubsub",
        "restBaseUrl": "https://vb-cl-ise-mnt2.ciscodemo.net:8910/pxgrid/ise/config/anc",
        "statusTopic": "/topic/com.cisco.ise.config.anc.status"
      }
    },
    {
      "name": "com.cisco.ise.config.anc",
      "nodeName": "~ise-admin-vb-cl-ise-mnt1",
      "properties": {
        "wsPubsubService": "com.cisco.ise.pubsub",
        "restBaseUrl": "https://vb-cl-ise-mnt1.ciscodemo.net:8910/pxgrid/ise/config/anc",
        "statusTopic": "/topic/com.cisco.ise.config.anc.status"
      }
    }
  ]
}
```

Different Service

Execute API – createPolicy

Certificate-Based Account Example

```
$ curl -s https://pxGrid-client:none@vb-cl-ise-mnt2.ciscodemo.net:8910/pxgrid/ise/config/anc/createPolicy \
-d '{"name":"Block","actions":["QUARANTINE"]}' \
--cert pxGrid-client.crt --key pxGrid-client.key \
--cacert ../demo-ca.cer -H 'Content-Type: Application/json' | jq -M
{
  "name": "Block",
  "actions": [
    "QUARANTINE"
  ]
}
```

POST [restBaseUrl]/createPolicy

There is no need to set the "id" field for the request policy object. After successful creati
in the returned policy object

If the policy name is already used in an existing policy, HTTP status "409 Conflict" will be

Request

```
{
  policy object
}
```

"policy" object

Name	Type	Description	ISE version
name	string		2.4
actions	array of action type		2.4

"action" type

"action" type can be on of the following strings:

- QUARANTINE (Disconnect the target client(after which it may reconnect)
- SHUT_DOWN (For wired devices, shutdown the port of the device, preventing reconnection.)
- PORT_BOUNCE
- RE_AUTHENTICATE (Force a target client to do Re-Authentication. Since ISE 2.6p7, 2.7p2, 3.0)

JSON List []

Python Example – createPolicy

Certificate-Based Account Example

```
import requests
import json

r=requests.post(f"https://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/ServiceLookup",
    cert=("pxGrid-client.crt", "pxGrid-client.key"),
    verify="demo-ca.cer",
    auth=("pxGrid-client", "none"),
    json={
        "name": "com.cisco.ise.config.anc"
    }
)
r.raise_for_status()
service_info=r.json()["services"][0]
node_name=service_info["nodeName"]
rest_url=service_info["properties"]["restBaseUrl"]
r=requests.post(f"{rest_url}/createPolicy",
    cert=("pxGrid-client.crt","pxGrid-client.key"),
    verify="demo-ca.cer",
    auth=("pxGrid-client", "none"),
    json={
        "name": "Block", "actions": ["QUARANTINE"]
    }
)
r.raise_for_status()
print(json.dumps(r.json(),indent=2))
```

Execute API – applyEndpointByMacAddress

Certificate-Based Account Example

```
$ curl -s https://pxGrid-client:none@vb-cl-ise-mnt2.ciscodemo.net:8910/pxgrid/ise/config/anc/applyEndpointByMacAddress \
-d {"policyName":"Block","macAddress":"11:22:33:44:55:66"} \
--cert pxGrid-client.crt --key pxGrid-client.key \
--cacert ../demo-ca.cer -H 'Content-Type: Application/json' | jq -M
{
  "operationId": "vb-cl-ise-mnt2.ciscodemo.net:0",
  "macAddress": "11:22:33:44:55:66",
  "status": "SUCCESS",
  "policyName": "Block"
}
```

POST [restBaseUrl]/applyEndpointByMacAddress

Apply a policy to the endpoint using MAC Address. If endpoint already has existing policy applied, the return status will be FAILURE with reason "mac address is already associated with this policy".

Request

```
{
  "policyName": string (required),
  "macAddress": string (required)
}
```

Python Example – applyEndpointByMacAddress

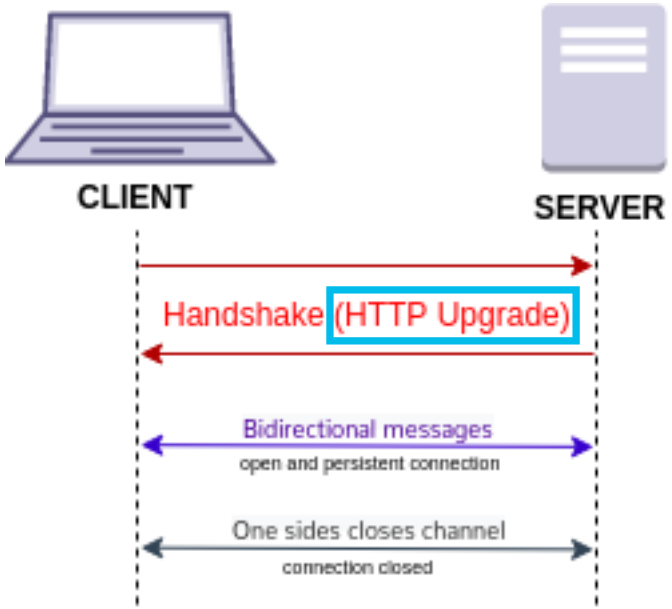
Certificate-Based Account Example

```
r=requests.post(f"{rest_url}/applyEndpointByMacAddress",
    cert=("pxGrid-client.crt", "pxGrid-client.key"),
    verify="demo-ca.cer",
    auth=("pxGrid-client", "none"),
    json={
        "policyName": "Block",
        "macAddress":"11:22:33:44:55:66"
    }
)
r.raise_for_status()
print(json.dumps(r.json(),indent=2))
```

Websockets

What are Websockets

A WebSocket is a persistent bi-directional communication channel between a client (e.g. a browser) and a backend service. In contrast with HTTP request/response connections, websockets can transport any number of protocols and provide server-to-client message delivery without polling.



STOMP

https://stomp.github.io/stomp-specification-

STOMP Frames

STOMP is a frame based protocol which assumes a reliable 2-way streaming network protocol (such as TCP) underneath. The client and server will communicate using STOMP frames sent over the stream. A frame's structure looks like:

```
COMMAND
header1:value1
header2:value2

Body^@
```

Frames are null (0x00) terminated

```
00000000  53 55 42 53 43 52 49 42  45 0a 64 65 73 74 69 6e  |SUBSCRIBE.destin|
00000010  61 74 69 6f 6e 3a 2f 74  6f 70 69 63 2f 63 6f 6d  |ation:/topic/com|
00000020  2e 63 69 73 63 6f 2e 69  73 65 2e 73 65 73 73 69  |.cisco.ise.sessi|
00000030  6f 6e 0a 69 64 3a 63 6c  69 0a 0a 00              |on.id:cli...|
0000003c
```

STOMP and pxGrid

https://developer.cisco.com/docs/pxgrid/#!technical-overview/stomp-and-pxGrid

pxGrid will use port 8910 on ISE for pxGrid-related REST and Websocket communication. The server will play Websocket's ping-pong game to detect offline, slow, or faulty pxGrid clients and network failures. Messages over Websocket will be sent and received in binary format. These messages should conform to the STOMP messaging protocol. You can read more about STOMP here. The following STOMP commands are supported on pxGrid:

- CONNECT
- DISCONNECT
- SUBSCRIBE
- UNSUBSCRIBE
- SEND
- MESSAGE
- ERROR

pxGrid consumers will typically implement SUBSCRIBE while providers will implement SEND.

websocat

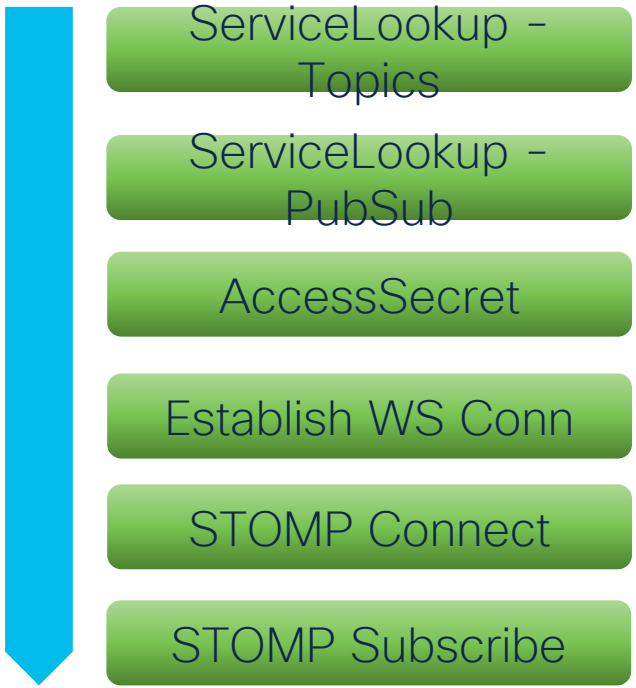
Base64 only relevant for websocat.
ISE does not base64-encode messages

- CLI websocket client
- STOMP messages are binary due to null termination
- Binary messages are base64-encoded

Example base64 of a STOMP connect message

```
$ echo -n -e 'CONNECT\naccept-version:1.2\nhost:~ise-pubsub-vb-cl-ise-px1\n\n\x00'
CONNECT
accept-version:1.2
host:~ise-pubsub-vb-cl-ise-px1

$ echo -n -e 'CONNECT\naccept-version:1.2\nhost:~ise-pubsub-vb-cl-ise-px1\n\n\x00'|base64
Q09OTkVDVAphY2NlcHQtdmVyc2lvbjoxLjIKaG9zdDp+aXNlLXB1YnN1Yi12Yi1jbC1pc2UtcHgxCgoA
```

Websocket Flow - Subscribe



Certificate Required if Using Certificates

Optional Access Secret lookup if not using certificate

Basic Auth with Access Secret OR certificates

* Assuming client account is approved

ServiceLookup – com.cisco.ise.session

Certificate-Based Account Example

For Reference

```
$ curl -s https://pxGrid-client:None@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/ServiceLookup \
--cert pxGrid-client.crt --key pxGrid-client.key -d '{"name":"com.cisco.ise.session"}' \
--cacert ../demo-ca.cer -H 'Content-Type: application/json' | jq -M
{
  "services": [
    {
      "name": "com.cisco.ise.session",
      "nodeName": "~ise-mnt-vb-cl-ise-mnt1",
      "properties": {
        "sessionTopic": "/topic/com.cisco.ise.session",
        "groupTopic": "/topic/com.cisco.ise.session.group",

        "wsPubsubService": "com.cisco.ise.pubsub",
        "restBaseURL": "https://vb-cl-ise-mnt1.ciscodemo.net:8910/pxgrid/mnt/sd",
        "restBaseUrl": "https://vb-cl-ise-mnt1.ciscodemo.net:8910/pxgrid/mnt/sd"
      }
    },
---- SNIP ----
  ]
}
```

For Reference

ServiceLookup - com.cisco.ise.pubsub

Certificate-Based Account Example

```
$ curl -s https://pxGrid-client:None@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/ServiceLookup \
--cert pxGrid-client.crt --key pxGrid-client.key -d '{"name":"com.cisco.ise.pubsub"}' \
--cacert ../demo-ca.cer -H 'Content-Type: application/json' | jq -M
{
  "services": [
    {
      "name": "com.cisco.ise.pubsub",
      "nodeName": "~ise-pubsub-vb-cl-ise-px1",
      "properties": {
        "wsUrl": "wss://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/ise/pubsub"
      }
    },
    {
      "name": "com.cisco.ise.pubsub",
      "nodeName": "~ise-pubsub-vb-cl-ise-px2",
      "properties": {
        "wsUrl": "wss://vb-cl-ise-px2.ciscodemo.net:8910/pxgrid/ise/pubsub"
      }
    }
  ]
}
```

For Reference

AccessSecret – PubSub

```
$ curl -s https://pxGrid-client:None@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccessSecret \
--cert pxGrid-client.crt --key pxGrid-client.key -d '{"peerNodeName": "~ise-pubsub-vb-cl-ise-px1"}' \
--cacert ../demo-ca.cer -H 'Content-Type: application/json' | jq -M
{
  "secret": "nP30k3Ir14Bdte7Y"
}
```

Subscribe to session topic

Connect Websocket

```
websocat -k --base64 --binary-prefix hex --basic-auth pxGrid-client:nP30k3Ir14Bdte7Y \
wss://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/ise/pubsub
```

Send STOMP Connect

```
hexQ09OTkVDVAphY2NlcHQtdmVyc2lvbioxLjIKaG9zdDp+aXNlLXB1YnN1Yi12Yi1jbC1pc2UtcHgxCgoA
hexQ09OTkVDVEVECmhlYXJ0LWJlYXQ6MCwwCnZlcnNpb246MS4yCgoA
```

CONNECT
accept-version:1.2
host:~ise-pubsub-vb-cl-ise-px1

CONNECTED
heart-beat:0,0
version:1.2

Response from pxGrid

Send STOMP Subscribe

```
hexU1VCU0NSSUJFCmRlc3RpbmF0aW9uOi90b3BpYy9jb20uY2lzY28uaXNlLnNlc3Npb24KaWQ6Y2xpCgoA
```

SUBSCRIBE
destination:/topic/com.cisco.ise.session
id:cli

STOMP Message

hexTUVTU0FHRQpjb250ZW50LWxlbmd0aDo2ODAKZGV
dlLWlkOjE5NTQ0CnN1YnNjcmlwdGlvbjpjbGkKCnsic2Vzc
TA1OjAwIiwic3RhdGUiOiJBVVR
---------- SNIP ----------
c2tFbmNyeXB0ZWQiOmZhbHNlLCJtZG1KYWlsQnJva2V
oelByb2ZpbGVzIjpbIlBlcm1pdEFjY2VzcyJdLCJhdXRoTWV
LCJzZXF1ZW5jZSI6MTR9AA==

Decoded from base64

MESSAGE
content-length:680
destination:/topic/com.cisco.ise.session
message-id:19544
subscription:cli

{"sessions":[{"timestamp":"2022-11-10T15:19:51.275-
05:00","state":"AUTHENTICATED","userName":"jsmith","
:33:22:11","ipAddresses":["4.3.2.3"],"macAddress":"E7:
],"endpointCheckResult":"none","identitySourcePortStart
DeviceProfileName":"Cisco","ssid":"66-55-44-33-22-
11","mdmRegistered":false,"mdmCompliant":false,"mdm
e,"selectedAuthzProfiles":["PermitAccess"],"authMethod


```
{
  "sessions": [
    {
      "timestamp": "2022-11-10T15:19:51.275-05:00",
      "state": "AUTHENTICATED",
      "userName": "jsmith",
      "callingStationId": "E7:B6:BB:A3:EA:9B",
      "calledStationId": "66:55:44:33:22:11",
      "ipAddresses": [
        "4.3.2.3"
      ],
      "macAddress": "E7:B6:BB:A3:EA:9B",
      "adNormalizedUser": "jsmith",
      "providers": [
        "None"
      ],
      "endpointCheckResult": "none",
      "identitySourcePortStart": 0,
      "identitySourcePortEnd": 0,
      "identitySourcePortFirst": 0,
      "networkDeviceProfileName": "Cisco",
      "ssid": "66-55-44-33-22-11",
      "mdmRegistered": false,
      "mdmCompliant": false,
      "mdmDiskEncrypted": false,
      "mdmJailBroken": false,
      "mdmPinLocked": false,
      "selectedAuthzProfiles": [
        "PermitAccess"
      ],
      "authMethod": "PAP_ASCII",
      "authProtocol": "PAP_ASCII"
    }
  ],
  "sequence": 14
}
```

Python Example 1/3 – REST API

Certificate-Based Account Example

```
import requests, websocket, ssl
from base64 import b64encode

r=requests.post(f"https://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/ServiceLookup",
    cert=("pxGrid-client.crt","pxGrid-client.key"),
    verify="demo-ca.cer",
    auth=("pxGrid-client","none"),
    json={
        "name": "com.cisco.ise.session"
    })
r.raise_for_status()
service_info=r.json()["services"][0]
session_topic=service_info["properties"]["sessionTopic"]
pubsub_service=service_info["properties"]["wsPubsubService"]
r=requests.post(f"https://snip:8910/pxgrid/control/ServiceLookup",
    cert=("pxGrid-client.crt","pxGrid-client.key"),
    verify="demo-ca.cer",
    auth=("pxGrid-client","none"),
    json={
        "name": pubsub_service
    })
r.raise_for_status()
service_info=r.json()["services"][0]
node_name=service_info["nodeName"]
ws_url=service_info["properties"]["wsUrl"]
```

Python Example 2/3 – Websocket

Certificate-Based Account Example

```
def on_open(wsapp):
    wsapp.send(f"CONNECT\naccept-version:1.2\nhost:{node_name}\n\n\x00",websocket.ABNF.OPCODE_BINARY)
    wsapp.send(f"SUBSCRIBE\ndestination:{session_topic}\nid:python\n\n\x00",websocket.ABNF.OPCODE_BINARY)

def on_message(wsapp,message):
    print(message.decode())

ssl_context=ssl.create_default_context()
ssl_context.load_verify_locations(cafile="demo-ca.cer")
ssl_context.load_cert_chain(certfile="pxGrid-client.crt", keyfile="pxGrid-client.key", password="KeyPasswd")
wsapp=websocket.WebSocketApp(ws_url,
  on_open=on_open,
  on_message=on_message,
   header={"Authorization": "Basic "+b64encode(("pxGrid-client:none").encode()).decode()}
)
wsapp.run_forever(sslopt={"context": ssl_context})
```

Python Example 3/3 – Output

Received Packet: CONNECTED
heart-beat:0,0
version:1.2

Received Packet: MESSAGE
content-length:679
destination:/topic/com.cisco.ise.session
message-id:32326
subscription:python

{"sessions":[{"timestamp":"2022-11-12T19:32:46.65-05:00","state":"AUTHENTICATED","userName":"jsmith","callingStationId":"29:F8:05:9C:36:E9","calledStationId":"66:55:44:33:22:11","ipAddresses":["4.3.2.3"],"macAddress":"29:F8:05:9C:36:E9","adNormalizedUser":"jsmith","providers":["None"],"endpointCheckResult":"none","identitySourcePortStart":0,"identitySourcePortEnd":0,"identitySourcePortFirst":0,"networkDeviceProfileName":"Cisco","ssid":"66-55-44-33-22-11","mdmRegistered":false,"mdmCompliant":false,"mdmDiskEncrypted":false,"mdmJailBroken":false,"mdmPinLocked":false,"selectedAuthzProfiles":["PermitAccess"],"authMethod":"PAP_ASCII","authProtocol":"PAP_ASCII"}],"sequence":21}

Websocket Flow – Context-In

ServiceRegister

ServiceLookup – PubSub

Certificate Required if Using Certificates

AccessSecret

Optional Access Secret lookup if not using certificate

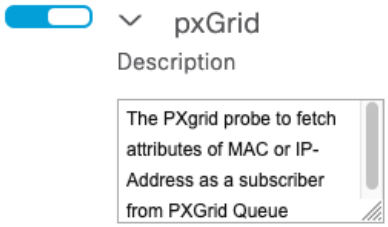
Establish WS Conn

Basic Auth with Access Secret OR certificates

STOMP Connect

STOMP Send

* Assuming client account is approved



Data picked up by pxGrid Profiler Probe

Asset JSON

Documented at https://github.com/cisco-pxGrid/pxGrid-rest-ws/wiki/Endpoint-Asset

```
{
  "opType": operation type,
  "asset": asset object
}
```

Objects

"opType" type

"opType" is one of the following strings:

- CREATE
- UPDATE
- DELETE

"asset" object

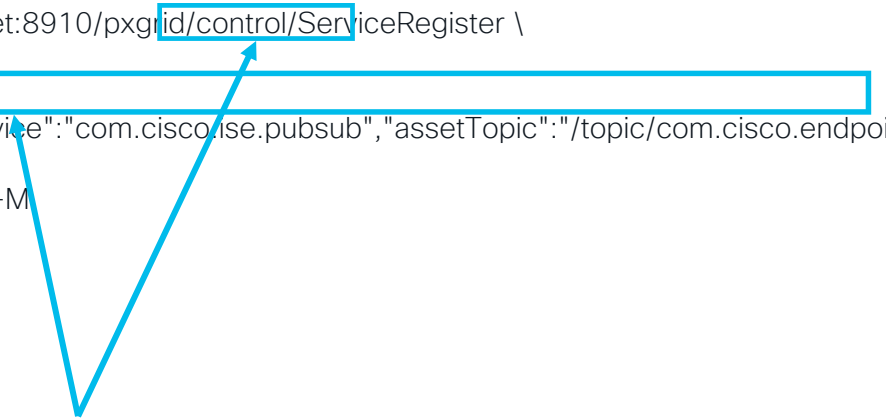
Name	Type	Description
assetId	string	
assetName	string	
assetIpAddress	string	
assetMacAddress	string	
assetVendor	string	
assetProductId	string	
assetSerialNumber	string	
assetDeviceType	string	
assetSwRevision	string	
assetHwRevision	string	
assetProtocol	string	
assetCustomAttributes	array of assetCustomAttributes	
assetConnectedLinks	array of assetConnectedLinks	

```
{
 "opType": "CREATE",
 "asset": {
  "assetId": 1,
  "assetName": "IOT1",
  "assetIpAddress": "1.2.3.4",
  "assetMacAddress": "22:33:44:55:66:77",
  "assetVendor": "CL",
  "assetHwRevision": "1.0",
  "assetSwRevision": "2.0",
  "assetProtocol": "Telnet",
  "assetProductId": "Wifi-IOT",
  "assetSerialNumber": "ABC12345",
  "assetDeviceType": "WiFi",
  "assetConnectedLinks": [
    {
     "key": "wifi1",
     "value": "ssid1"
    }
  ]
 }
}
```

ServiceRegister – com.cisco.endpoint.asset

Certificate-Based Account Example

```
$ curl -s https://pxGrid-client:None@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/ServiceRegister \
--cert pxGrid-client.crt --key pxGrid-client.key \
-d
'{"name":"com.cisco.endpoint.asset","properties":{"wsPubsubService":"com.cisco.ise.pubsub","assetTopic":"/topic/com.cisco.endpoint
.asset"}}' \
--cacert ../demo-ca.cer -H 'Content-Type: application/json' | jq -M
{
  "id": "28cd8b3c-20ca-4630-a230-92e8d8ce3e2a",
  "reregisterTimeMillis": 300000
}
```



Advanced Topic – see
https://github.com/cisco-pxgrid/pxgrid-rest-ws/wiki/pxGrid-Provider#serviceregister

ServiceLookup - com.cisco.ise.pubsub

Certificate-Based Account Example

```
$ curl -s https://pxGrid-client:None@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/ServiceLookup \
--cert pxGrid-client.crt --key pxGrid-client.key -d '{"name":"com.cisco.ise.pubsub"}' \
--cacert ../demo-ca.cer -H 'Content-Type: application/json' | jq -M
{
  "services": [
    {
      "name": "com.cisco.ise.pubsub",
      "nodeName": "~ise-pubsub-vb-cl-ise-px1",
      "properties": {
        "wsUrl": "wss://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/ise/pubsub"
      }
    },
    {
      "name": "com.cisco.ise.pubsub",
      "nodeName": "~ise-pubsub-vb-cl-ise-px2",
      "properties": {
        "wsUrl": "wss://vb-cl-ise-px2.ciscodemo.net:8910/pxgrid/ise/pubsub"
      }
    }
  ]
}
```

AccessSecret – PubSub

```
$ curl -s https://pxGrid-client:None@vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/control/AccessSecret \
--cert pxGrid-client.crt --key pxGrid-client.key -d '{"peerNodeName" "~ise-pubsub-vb-cl-ise-px1"}' \
--cacert ../demo-ca.cer -H 'Content-Type: application/json' | jq -M
{
  "secret": "nP30k3Ir14Bdte7Y"
}
```

STOMP SEND Command

```
$ ASSET=`sed -E 's/^ +//g' <endpoint.json |tr -d '\n'`;LEN=`echo -n $ASSET|wc -c|tr -d " "`; \
echo -n -e "SEND\\ndestination:/topic/com.cisco.endpoint.asset\\ncontent-length:$LEN\\n\\n$ASSET\x00"
SEND
destination:/topic/com.cisco.endpoint.asset
content-length:373

{"opType": "CREATE","asset": {"assetId": 1,"assetName": "IOT1","assetIpAddress": "1.2.3.4","assetMacAddress": "22:33:44:55:66:77",...
```

* Removes leading space and new lines

Send Asset Data

Connect Websocket

```
websocat -k --base64 --binary-prefix hex --basic-auth pxGrid-client:nP30k3Ir14Bdte7Y \
wss://vb-cl-ise-px1.ciscodemo.net:8910/pxgrid/ise/pubsub
```

Send STOMP Connect

```
hexQ09OTkVDVAphY2NlcHQtdmVyc2lvbjoxLjIKaG9zdDp+aXNlLXB1YnN1Yi12Yi1jbC1pc2UtcHgxCgoA
hexQ09OTkVDVEVECmhlYXJ0LWJlYXQ6MCwwCnZlcnNpb246MS4yCgoA
```

CONNECT
accept-version:1.2
host:~ise-pubsub-vb-cl-ise-px1

CONNECTED
heart-beat:0,0
version:1.2

Send STOMP Send

```
hexU0VORApkZXN0aW5hdGlvbjovdG9waWMvY29tLmNpc2NvLmVu..SNIP...
```

SEND
destination:/topic/com.cisco.endpoint.asset
content-length:373

{"opType": "CREATE","asset": {"assetId": 1,"assetName": "IOT1"..SNIP...

* No feedback from pxGrid.

Python Example – 1/2 REST API

Certificate-Based Account Example

```
import requests, websocket, ssl, json
from base64 import b64encode

r=requests.post(f"https://snip:8910/pxgrid/control/ServiceRegister",
    cert=("pxGrid-client.crt","pxGrid-client.key"),
    verify="demo-ca.cer",
    auth=("pxGrid-client","none"),
    json={
        "name": "com.cisco.endpoint.asset",
        "properties": {
            "wsPubsubService": "com.cisco.ise.pubsub",
            "assetTopic":"/topic/com.cisco.endpoint.asset"
        }
    })
r.raise_for_status()
r=requests.post(f"https://snip:8910/pxgrid/control/ServiceLookup",
    cert=("pxGrid-client.crt","pxGrid-client.key"),
    verify="demo-ca.cer",
    auth=("pxGrid-client","none"),
    json={
        "name": "com.cisco.ise.pubsub"
    })
```

```
r.raise_for_status()
service_info=r.json()["services"][0]
node_name=service_info["nodeName"]
ws_url=service_info["properties"]["wsUrl"]
```

Python Example – 2/2 Websocket

```
ssl_context=ssl.create_default_context()
ssl_context.load_verify_locations(cafile=".demo-ca.cer")
ssl_context.load_cert_chain(certfile="pxGrid-client.crt", keyfile="pxGrid-client.key", password="KeyPasswd")
ws=websocket.create_connection(ws_url,
    sslopt={"context": ssl_context},
    header={"Authorization": "Basic "+b64encode(("pxGrid-client:none").encode()).decode()}
)
with open("endpoint.json","r") as f:
    endpoint=json.dumps(json.loads(f.read()))
ws.send(f"CONNECT\naccept-version:1.2\nhost:{node_name}\n\n\x00",websocket.ABNF.OPCODE_BINARY)
ws.send(f"SEND\ndestination:/topic/com.cisco.endpoint.asset\ncontent-length:{len(endpoint)}\n\n{endpoint}\x00".encode("utf-8"),
    websocket.ABNF.OPCODE_BINARY)
ws.close()
```

Context Visibility - Result

PSN is the subscriber to asset topic


```
{
 "opType": "CREATE",
 "asset": {
  "assetId": 1,
  "assetName": "IOT1",
  "assetIpAddress": "1.2.3.4",
  "assetMacAddress": "22:33:44:55:66:77",
  "assetVendor": "CL",
  "assetHwRevision": "1.0",
  "assetSwRevision": "2.0",
  "assetProtocol": "Telnet",
  "assetProductId": "Wifi-IOT",
  "assetSerialNumber": "ABC12345",
  "assetDeviceType": "WiFi",
  "assetConnectedLinks": [
    {
     "key": "wifi1",
     "value": "ssid1"
    }
  ]
 }
}
```

CLI Client

Interactive pxGrid CLI client

https://github.com/vbobrov/pxAPI

- Support for both certificate and password authentication when connecting to pxGrid nodes
- Commands and methods to interact with most pxGrid services
- Websocket support for subscribing to topics.
- Debug capabilities to show all low level interactions with pxGrid

Installation

```
# Download
git clone https://github.com/vbobrov/pxAPI
cd pxAPI

# Optionally create virtual env
python3 -m venv env
. env/bin/activate

# Install requirements
pip3 install -r requirements.txt

# Start utility
$ ./pxShell.py
pxShell>
```

Built-in Help

```
pxShell> help

Documented commands (type help <topic>):
========================================
accountcreate  anc    debug  mdm     radius  sxp    trustsec
activate       config  help   profiler  session  system  trustseccfg

Undocumented commands:
======================
EOF
pxShell> help sxp
sxp options:
        bindings: List all SXP bindings
        topics: List topics available for subscription
        subscribe <topic>: Subscribe to a topic
```

For Reference

Config and Activate

```
pxShell> help config
Config options:
        save <file>: Save config to file
        load <file>: Load config from file
        apply [file]: Instantiate connection to pxGrid. Optionally load the file and apply in one step
        show: Show current settings
        pxnode <hostname>: Set pxGrid PSN FQDN
        name <clientname>: Set pxGrid client name
        cert <certfile>: Set client certificate file name
        key <keyfile>: Set client private key
        root [<rootfile>]: Set root CA file. Leave out <rootfile> to disable server certificate verification
        password <password>: Set password for password based authentication

pxShell> config pxnode vb-cl-ise-px1.ciscodemo.net
pxShell> config name pxGrid-client
pxShell> config cert pxGrid-client.crt
pxShell> config key pxGrid-client.key
pxShell> config root demo-ca.cer
pxShell> config apply
pxShell> activate
{
  "accountState": "ENABLED",
  "version": "2.0"
}
```

For Reference

REST API Calls

```
pxShell> session all
{
  "sessions": [
    {
      "timestamp": "2022-11-10T14:06:24.222-05:00",
      "state": "AUTHENTICATED",
      "userName": "jsmith",
      "auditSessionId": "ac1f1c1000005000636d4bb0",
      "ipAddresses": [
        "172.31.28.192"
      ],
----- SNIP ----
pxShell> anc create Stop QUARANTINE
{
  "name": "Stop",
  "actions": [
    "QUARANTINE"
  ]
}
```

```
pxShell> system perfs
{
  "performances": [
    {
      "timestamp": "2022-11-11T16:49:14.572391-05:00",
      "serverName": "vb-cl-ise-mnt2",
      "radiusRate": 0.0,
      "radiusCount": 0,
      "radiusLatency": 0.0
    },
    {
      "timestamp": "2022-11-11T16:49:14.572391-05:00",
      "serverName": "vb-cl-ise-mnt1",
      "radiusRate": 0.0,
      "radiusCount": 0,
      "radiusLatency": 0.0
    },

--- SNIP ---
```

For Reference

Subscribing to Topics

```
pxShell> session topics
"sessionTopic"
"groupTopic”
pxShell> session subscribe sessionTopic
Ctrl-C to disconnect...
Received Packet: command=CONNECTED content:
{}
Received Packet: command=MESSAGE content:
{
  "sessions": [
    {
      "timestamp": "2022-11-11T16:56:22.561-05:00",
      "state": "AUTHENTICATED",
      "userName": "jsmith",
      "callingStationId": "1C:F3:7D:B7:F5:A0",
--- SNIP ----
    }
  ],
  "sequence": 18
}
```

References

References

- pxGrid Reference: https://github.com/cisco-pxGrid/pxGrid-rest-ws/wiki
- Developer resources: https://developer.cisco.com/docs/pxGrid
- CLI Utility: https://github.com/vbobrov/pxAPI
- Code and commands: https://github.com/vbobrov/devnet-2032
  - The URL is correct; course number was reallocated after Viktor created it! ☺

Cisco Webex App

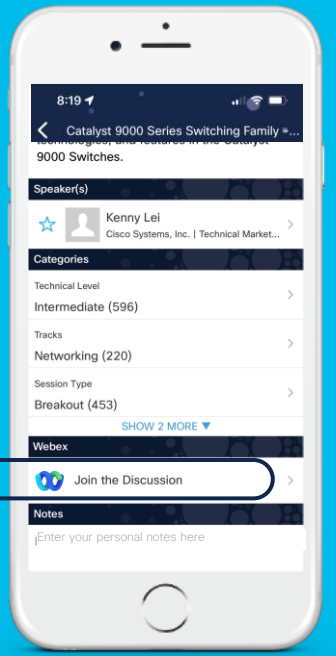
Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated until February 24, 2023.



Complete your Session Survey

- Please complete your session survey after each session. Your feedback is important.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Session Catalog and clicking the "Attendee Dashboard" at https://www.ciscolive.com/emea/learn/sessions/session-catalog.html

Continue Your Education

Visit the Cisco Showcase for related demos.

Book your one-on-one Meet the Engineer meeting.

Attend any of the related sessions at the DevNet, Capture the Flag, and Walk-in Labs zones.

Visit the On-Demand Library for more sessions at ciscolive.com/on-demand.

API Insights

API Insights is an open source solution, developed by Cisco, that enables developers to identify technical, documentation completeness and quality issues with APIs before production, helping to shift left.

Fork API Insights on GitHub | Get API Insights VS Code Extension ↗

Validates and scores APIs

API Insights validates and scores API definitions against an organization's guidelines. This allows you to track and improve API quality consistently and efficiently.

Built for your CI/CD pipeline

Developers can use API Insights through its own interface or as part of their CI/CD pipeline.

API Lifecycle Management

Provides a trend timeline of API quality, and generates both API changelogs and diff comparisons of API versions to identify breaking changes.

API Insights at a glance

cs.co/API

cisco Live!

WEBINAR SERIES

Secure the Future with a Zero Trust Security Approach

Explore How to Secure Your Applications and Data for Tomorrow's Threat Landscape in our Zero Trust Webinar Series for Developers

REGISTER NOW
http://cs.co/90093eC4T

cisco Live!

We want your feedback!

Answer a few questions in a short survey to be entered to win a DevNet Hoodie!

cs.co/DNZCLEUR2023


cisco
The bridge to possible
Thank you
cisco Live!


CISCO Live!
ALL IN