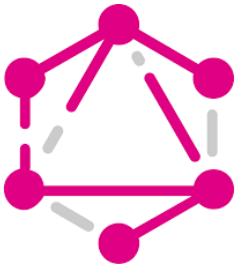
XMC 8.5 Workshop
Northbound Interface (NBI)

Markus Nikulski
Sr. Corporate System Engineer

October 2020

Extreme

XMC North Bound Interface (NBI)

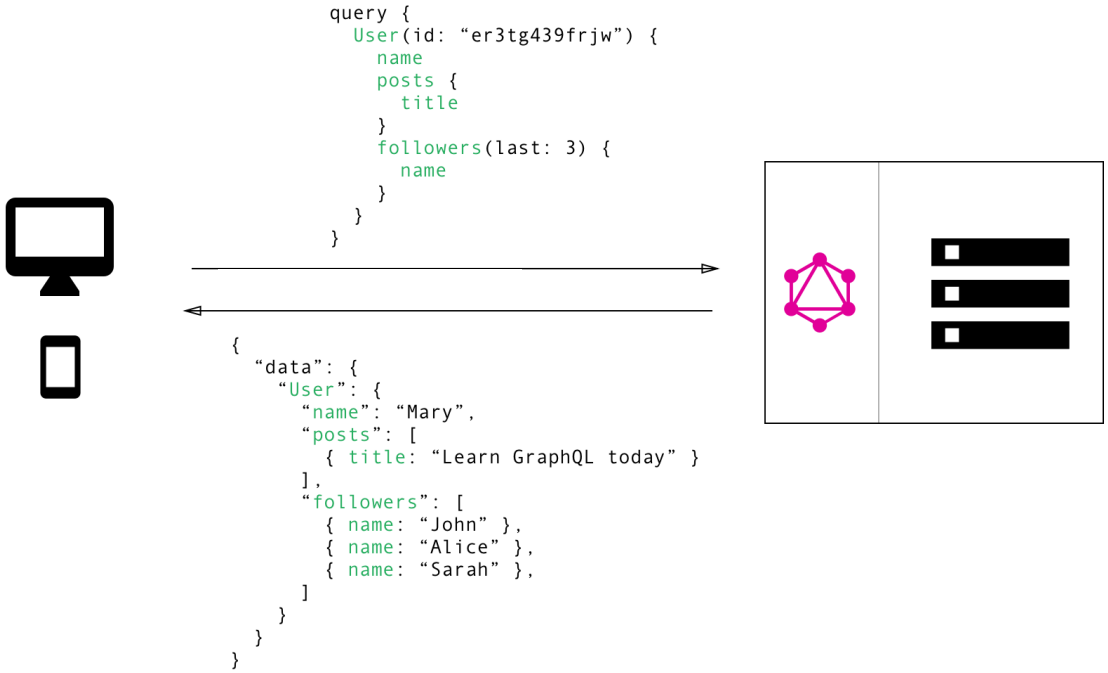


GraphQL is a data **query language** developed internally by Facebook before being publicly released in 2015. It provides an alternative to **REST** and **ad-hoc webservice** architectures.

It allows clients to define the structure of the data required,
and exactly the same structure of the data is returned from the server.

http://graphql.org/

XMC North Bound Interface (NBI)



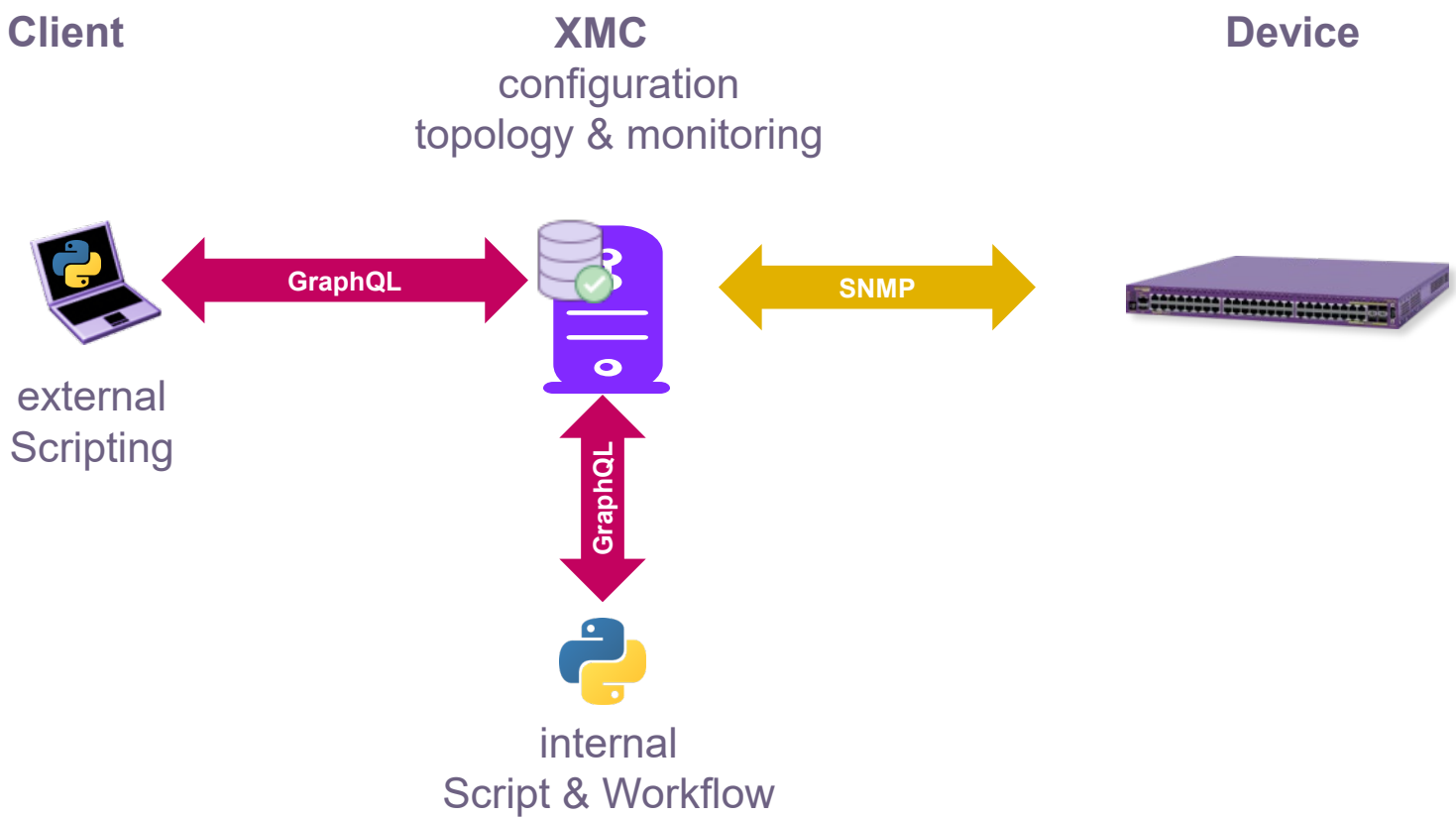
You’d simply send a single query to the GraphQL server that includes the data requirements. The server then responds with a JSON object where these requirements are fulfilled.

XMC North Bound Interface (NBI)

API	https://<xmc-ip-address>:8443/**nbi**/**graphql**
Online documentation	https://<xmc-ip-address>:8443/**nbi**/**graphiql**/**index.html**
Schema	https://<xmc-ip-address>:8443/**nbi**/**graphql**/**schema.idl**

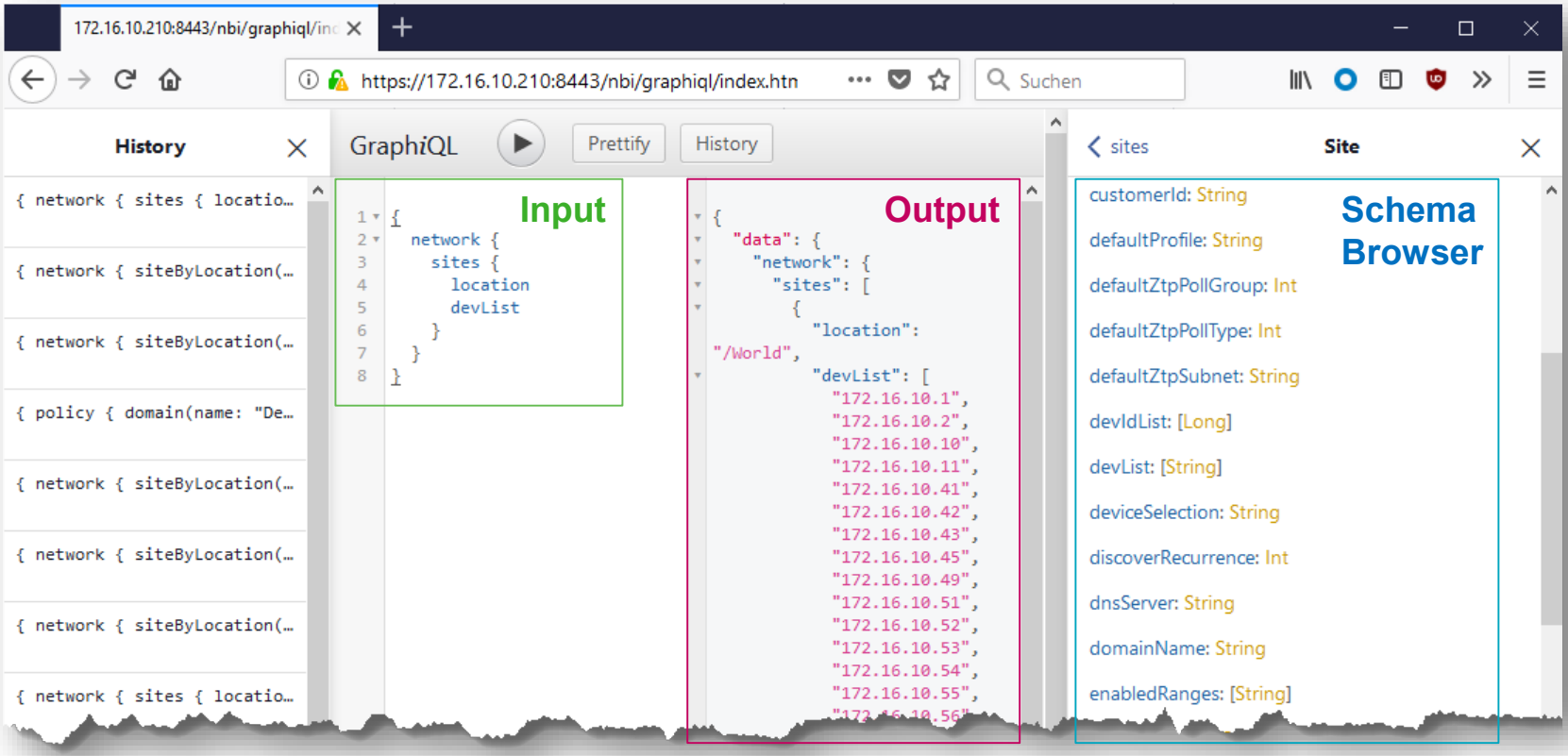
Extreme Networks API documentations	http://**developer**.extremenetworks.com

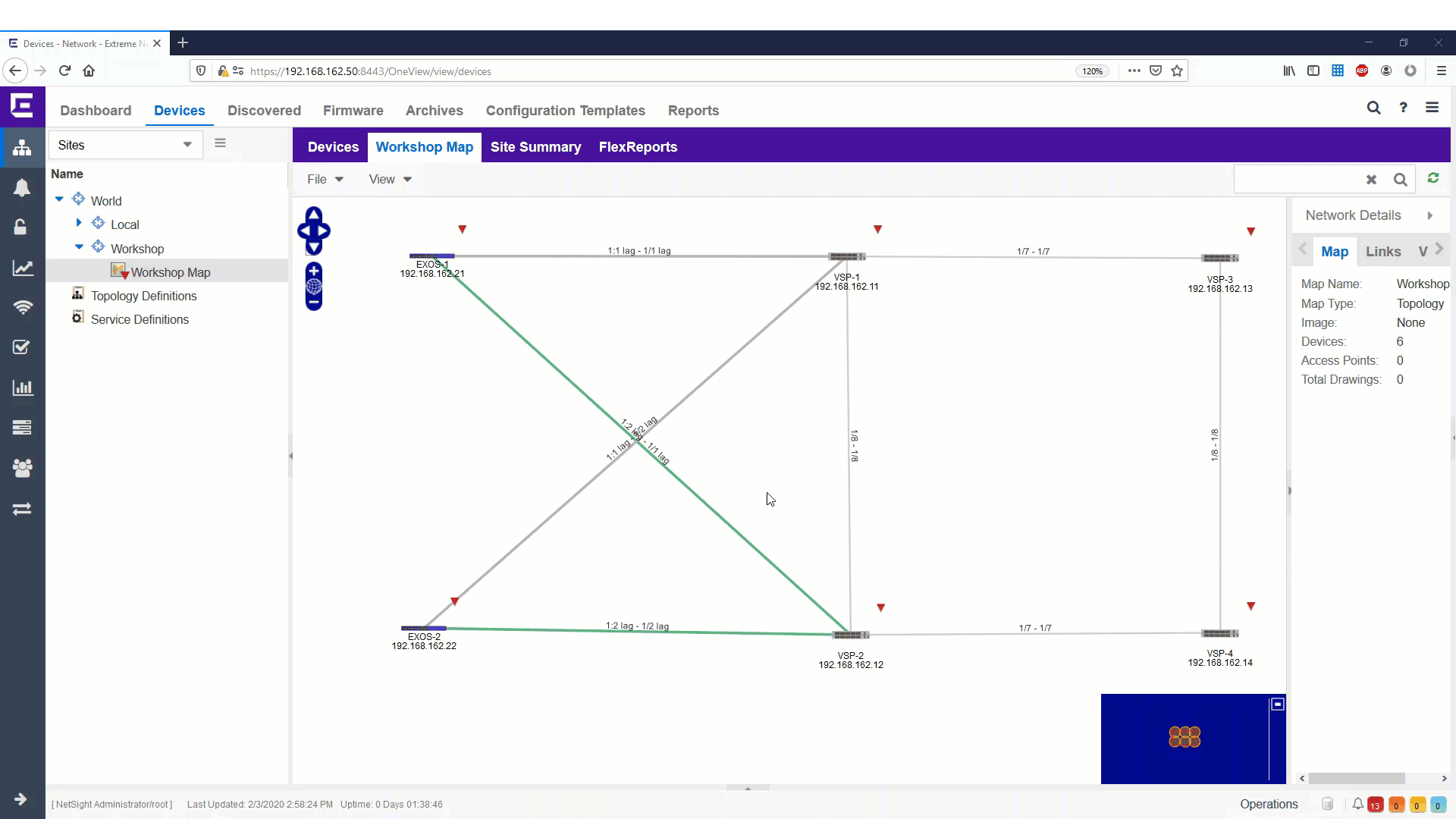
XMC NBI GraphQL usage



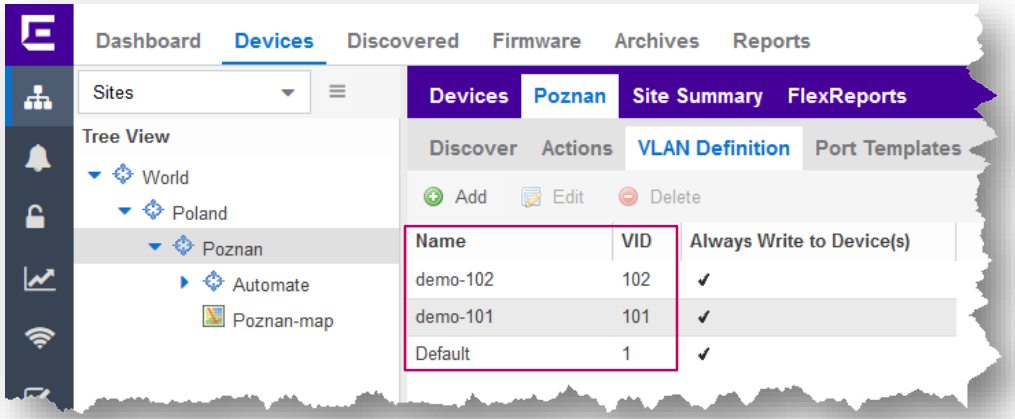
XMC NBI GraphiQL

https://<xml-ip-address>:8443/**nbi/graphiql/index.html**




Devices - Network - Extreme N
https://192.168.162.50:8443/OneView/view/devices
120%
Dashboard
Devices
Discovered
Firmware
Archives
Configuration Templates
Reports
Sites
Name
World
Local
Workshop
Workshop Map
Topology Definitions
Service Definitions
Devices
Workshop Map
Site Summary
FlexReports
File
View
EXOS-1
192.168.162.21
VSP-1
192.168.162.11
VSP-3
192.168.162.13
EXOS-2
192.168.162.22
VSP-2
192.168.162.12
VSP-4
192.168.162.14
1:1 lag - 1/1 lag
1/7 - 1/7
1:2 lag - 1/2 lag
1:1 lag - 1/1 lag
1/8 - 1/8
1/8 - 1/8
1:2 lag - 1/2 lag
1/7 - 1/7
Network Details
Map
Links
Map Name: Workshop
Map Type: Topology
Image: None
Devices: 6
Access Points: 0
Total Drawings: 0
[NetSight Administrator/root]
Last Updated: 2/3/2020 2:58:24 PM
Uptime: 0 Days 01:38:46
Operations

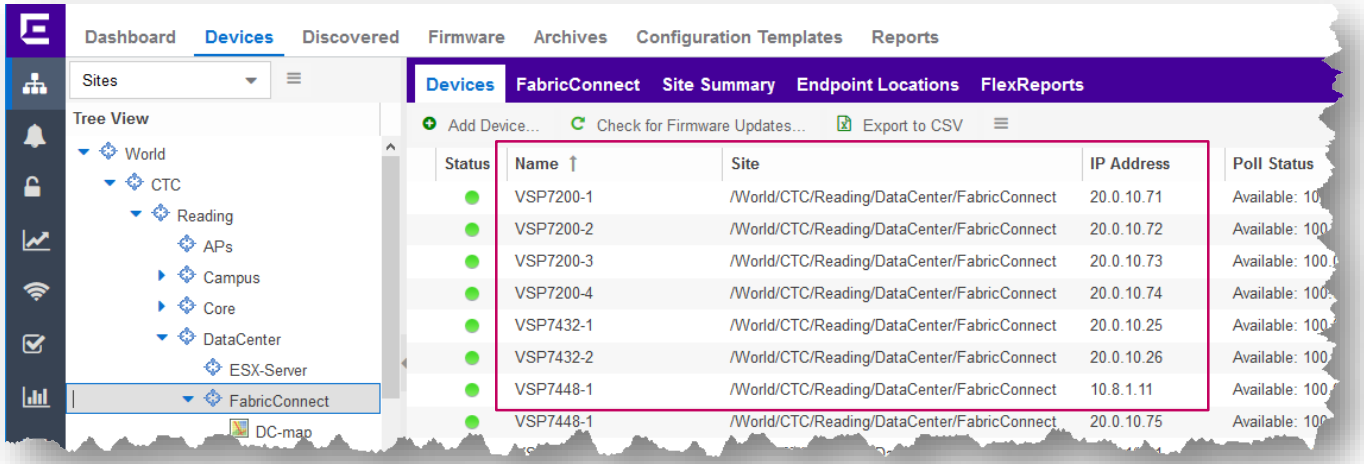
XMC NBI GraphiQL query examples




```
{
  network {
    siteByLocation(location: "/World/Poland/Poznan") {
      vlans {
        name
        vid
      }
    }
  }
}
```

```
{
  "data": {
    "network": {
      "siteByLocation": {
        "vlans": [
          {
            "name": "demo-102",
            "vid": 102
          },
          {
            "name": "demo-101",
            "vid": 101
```

XMC NBI GraphiQL query examples




```
{
  network {
    devices {
      baseMac
      ip
      deviceName
      deviceData {
        defaultSitePath
        sysDescr
      }
    }
  }
}
```

```
{ "data": {
    "network": {
      "devices": [
        { "baseMac": "64:6A:52:C5:5C:00",
          "ip": "20.0.10.23",
          "deviceName": "VSP8400-3",
          "deviceData": {
            "defaultSitePath": "/World/CTC/Reading/DataCenter/FabricConnect",
            "sysDescr": "VSP-8404C (8.0.6.0)"
          }
        },
        { "baseMac": "00:04:96:A4:FB:08",
```

XMC NBI GraphiQL combined query examples

```
{
  network {
    siteByLocation(location: "/World/CTC/Reading/Campus") {
      vlans {
        name
        vid
      }
    }
    devices {
      baseMac
      deviceName
      deviceData {
        defaultSitePath
        sysDescr
      }
    }
  }
}
```

```
{
  "data": {
    "network": {
      "siteByLocation": {
        "vlans": [
          {
            "name": "DATA",
            "vid": 2000
          },
          {
            "name": "Default",
            "vid": 1
          }
        ]
      },
      "devices": [
        {
          "baseMac": "00:04:96:A5:12:72",
          "deviceName": "X690-2",
          "deviceData": {
            "defaultSitePath": "/World/CTC/Reading/DataCenter/IP-Fabric",
            "sysDescr": "ExtremeXOS (X690-48x-2q-4c) …"
          }
        },
```

Using documentation

variable declaration

timeout: Int = 30

Request timeout in seconds.Timeout must be >= 5 && <= 3600

```
timeout: 6
```

siteId: Long

Unique identifier for an existing site.

```
siteId: 123e4567-e89b-12d3-a456-426655440000
```

autoAddDevices: Boolean

```
autoAddDevice: false
autoAddDevice: true
```

siteLocation: String

Required when creating a Site, otherwise its optional when a siteId is used. Example: siteLocation: "/World/Site1"

```
siteLocation: "/World/demo"
```

mutationType: ListMutationTypeInput

SiteCustomVariablesConfigInput **ListMutationTypeInput**

No Description

VALUES

ADD
Adds a new entry to the list. Fails if a matching entry already exists.

REMOVE
Removes an existing entry in the list. Fails if a matching entry is not found.

REMOVE_ALL
Removes all the entries in the list.

REPLACE
Replaces an existing entry in the list. Adds a new entry if a matching entry is not found.

REPLACE_ALL
Replaces all existing entries in the list.

UPDATE
Update an existing entry in the list. Modifies only the parameters given

```
mutationType: ADD
```

variable declaration

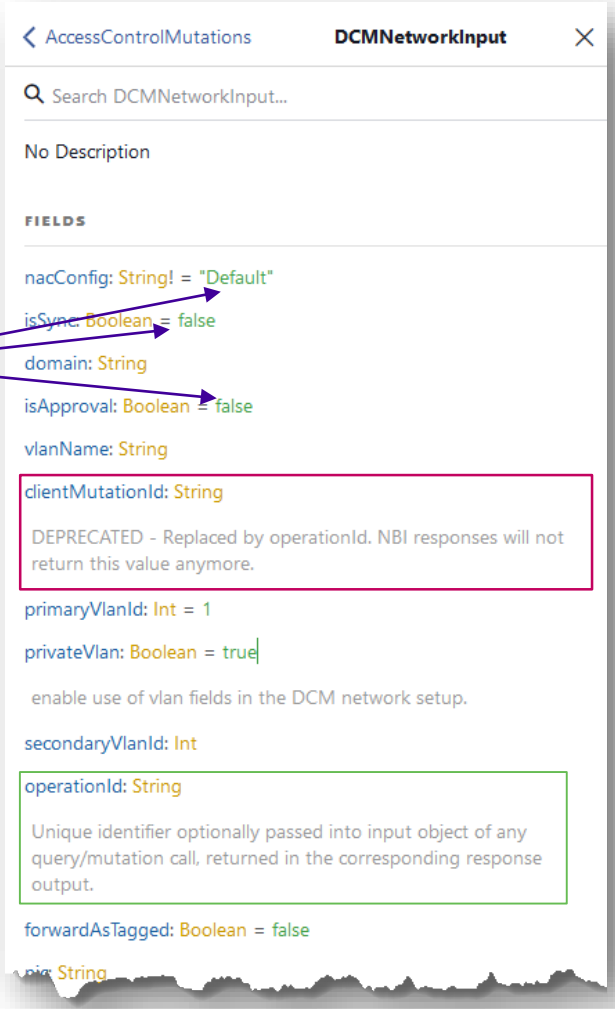
default value

should not be used

is mandatory

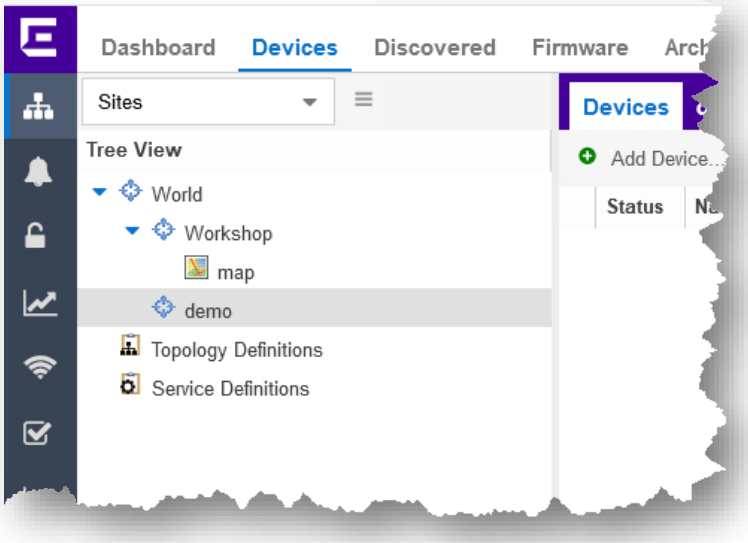
```
name: String!
```

useful for
advanced users



Site examples

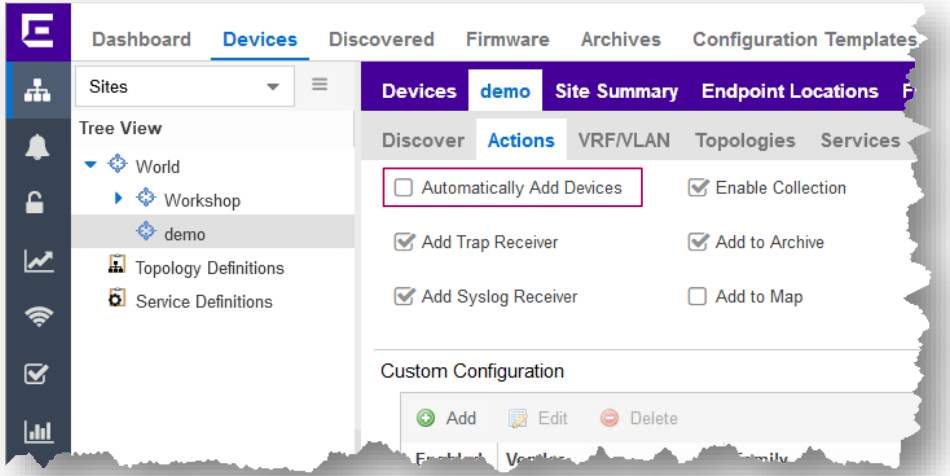
Create a site




```
mutation {
  network {
    createSite(input: {siteLocation: "/World/demo" }) {
      message
      status
    }
  }
}
```

```
{
  "data": {
    "network": {
      "createSite": {
        "message": "",
        "status": "SUCCESS"
      }
    }
  }
}
```

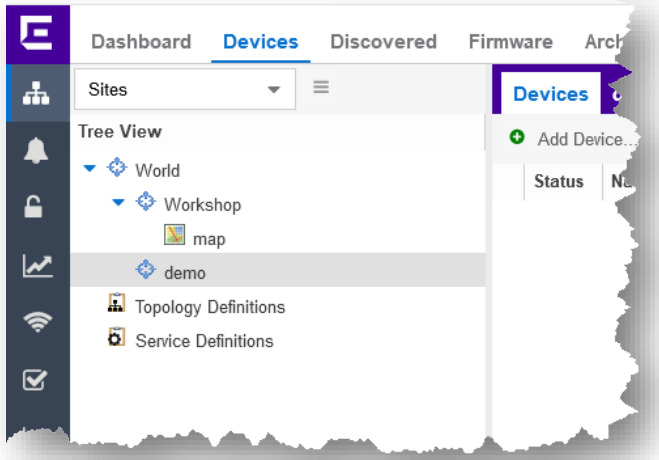
Configure site parameter




```
mutation {
  network {
    modifySite(input: {
      siteId: 6
      siteConfig: {
        actionsConfig: {
          autoAddDevices: false
        }
      }
    }) {
      message
      status
    }
  }
}
```

```
{
  "data": {
    "network": {
      "modifySite": {
        "message": "",
        "status": "SUCCESS"
      }
    }
  }
}
```

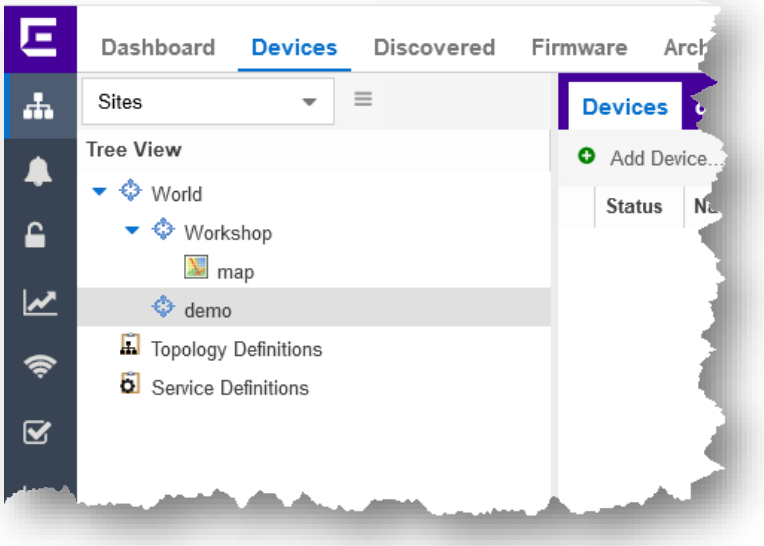
get all sites




```
{
  network {
    sites {
      location
    }
  }
}
```

```
{
  "data": {
    "network": {
      "sites": [
        {
          "location": "/World"
        },
        {
          "location": "/World/Workshop"
        },
        {
          "location": "/World/demo"
        }
      ]
    }
  }
}
```

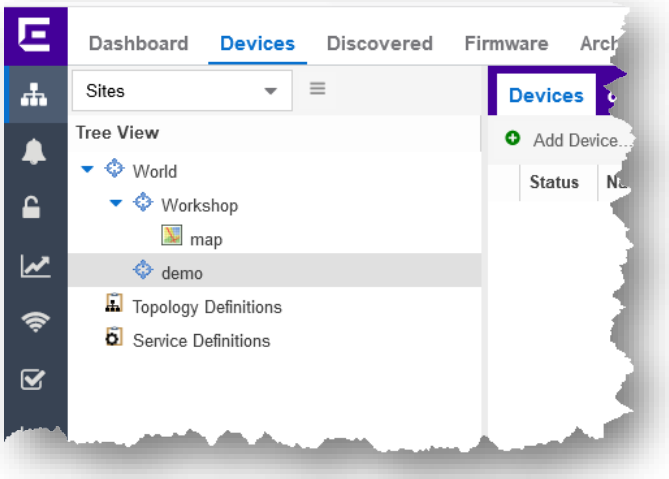
get specific site using filter




```
{
  network {
    siteByLocation(location: "/World/demo") {
      defaultProfile
    }
  }
}
```

```
{
  "data": {
    "network": {
      "siteByLocation": {
        "defaultProfile": "public_v2_Profile"
      }
    }
  }
}
```

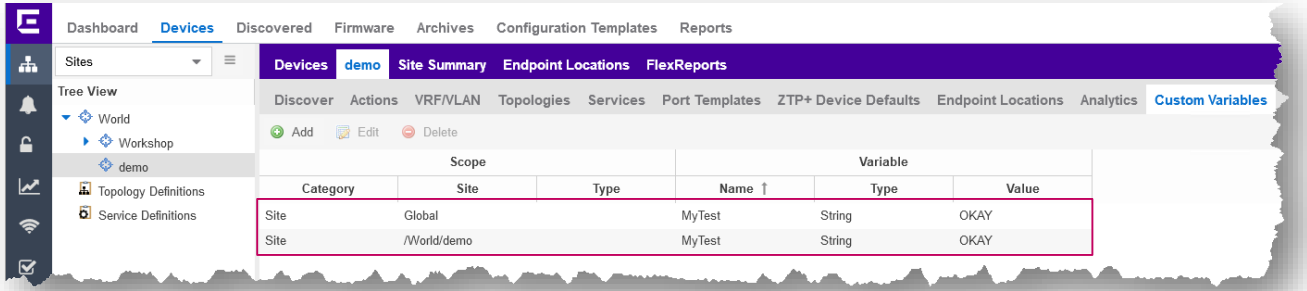
delete site




```
mutation {
  network {
    deleteSite(input: { siteLocation: "/World/demo" }) {
      message
      status
    }
  }
}
```

```
{
  "data": {
    "network": {
      "deleteSite": {
        "message": "",
        "status": "SUCCESS"
      }
    }
  }
}
```

Using site variables (set)

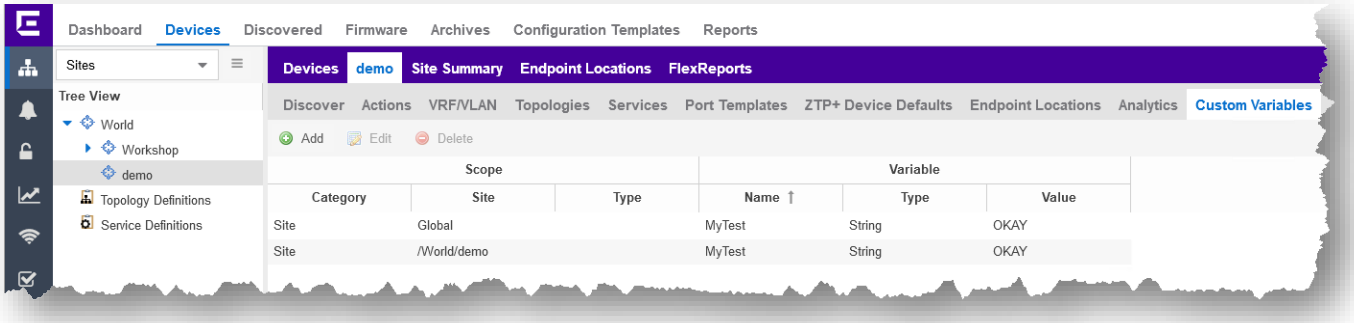



```
mutation {
  network {
    modifySite(input: {
      siteLocation: "/World/demo"
      siteConfig: {
        customVariablesConfig: {
          mutationType: ADD
          customVariables: {
            name: "MyTest"
            value: "OKAY"
            valueType: STRING
            scopeCategory: SITE
          }
        }
      }
    }) {
      message
      status
    }
  }
}
```

```
{
  "data": {
    "network": {
      "modifySite": {
        "message": "",
        "status": "SUCCESS"
      }
    }
  }
}
```

even if **sopeCategory** is **SITE**, an additional global variable with this name is created

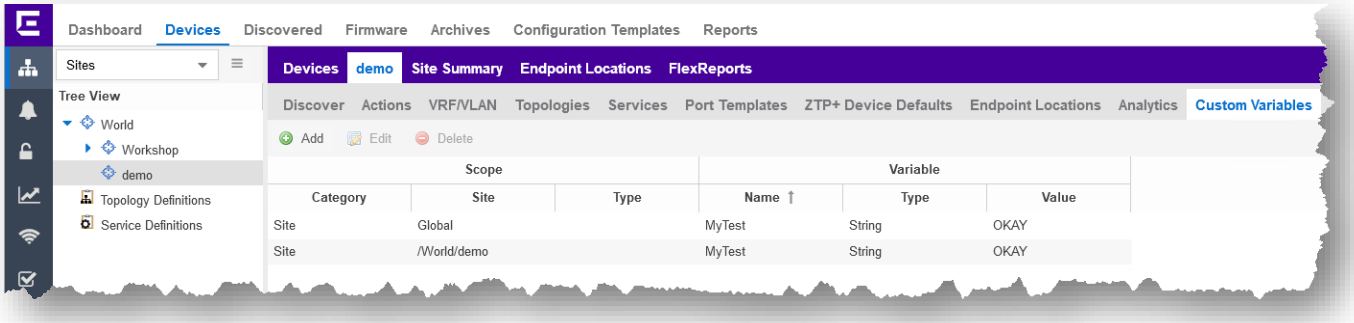
Using site variables (read **site** context)




```
{
  network {
    siteByLocation(location: "/World/demo") {
      customVariables {
        name
        value
        valueType
        scopeType
      }
    }
  }
}
```

```
{
  "data": {
    "network": {
      "siteByLocation": {
        "customVariables": [
          {
            "name": "MyTest",
            "value": "OKAY",
            "valueType": "STRING",
            "scopeType": "/World/demo"
          },
          {
            "name": "MyTest",
            "value": "OKAY",
            "valueType": "STRING",
            "scopeType": "/World/demo"
          }
        ]
      }
    }
  }
}
```

Using site variables (read **device** context)




```
{
  network {
    device(ip: "192.168.0.11") {
      customVariables {
        name
        value
        valueType
        scopeType
      }
    }
  }
}
```

```
{
  "data": {
    "network": {
      "device": {
        "customVariables": [
          {
            "name": "MyTest",
            "value": "OKAY",
            "valueType": "STRING",
            "scopeType": "/World/demo"
          }
        ]
      }
    }
  }
}
```

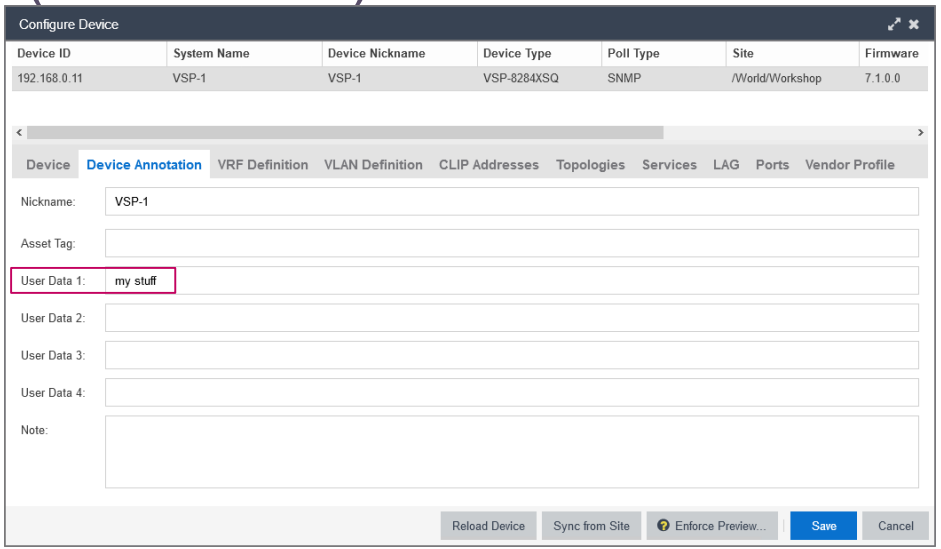
Using site variables (delete)

```
mutation {
  network {
    modifySite(input: {
      siteLocation: "/World/demo"
      siteConfig: {
        customVariablesConfig: {
          mutationType: REMOVE_ALL
          customVariables: {
            name: "MyTest"
            valueType: STRING
            scopeCategory: SITE
          }
        }
      }
    }) {
      message
      status
    }
  }
}
```

```
{
  "data": {
    "network": {
      "modifySite": {
        "message": "",
        "status": "SUCCESS"
      }
    }
  }
}
```


Device examples

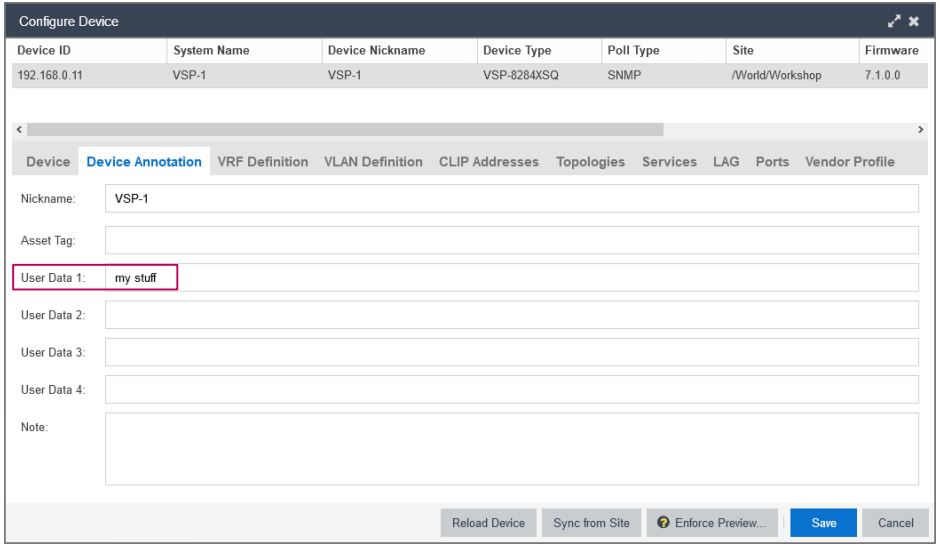
write device variables (Annotation)




```
mutation {
  network {
    configureDevice(input: {
      deviceConfig: {
        ipAddress: "192.168.0.11"
        deviceAnnotationConfig: {
          userData1: "my stuff"
        }
      }
    }) {
      message
      status
    }
  }
}
```

```
{
  "data": {
    "network": {
      "configureDevice": {
        "message": "",
        "status": "SUCCESS"
      }
    }
  }
}
```

read device variables (Annotation)

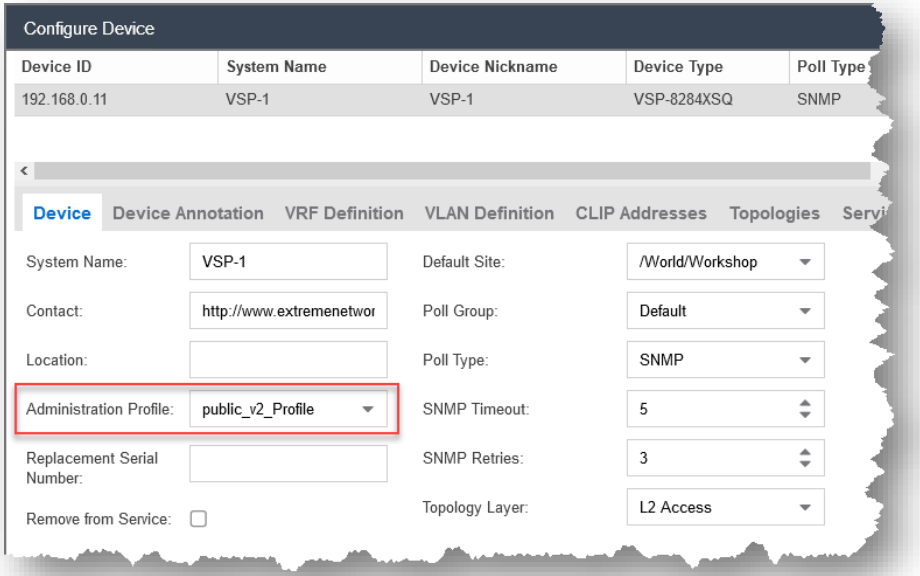



```
{
  network {
    device(ip: "192.168.0.11") {
      deviceData {
        userData1
      }
    }
  }
}
```

```
{
  "data": {
    "network": {
      "device": {
        "deviceData": {
          "userData1": "my stuff"
        }
      }
    }
  }
}
```

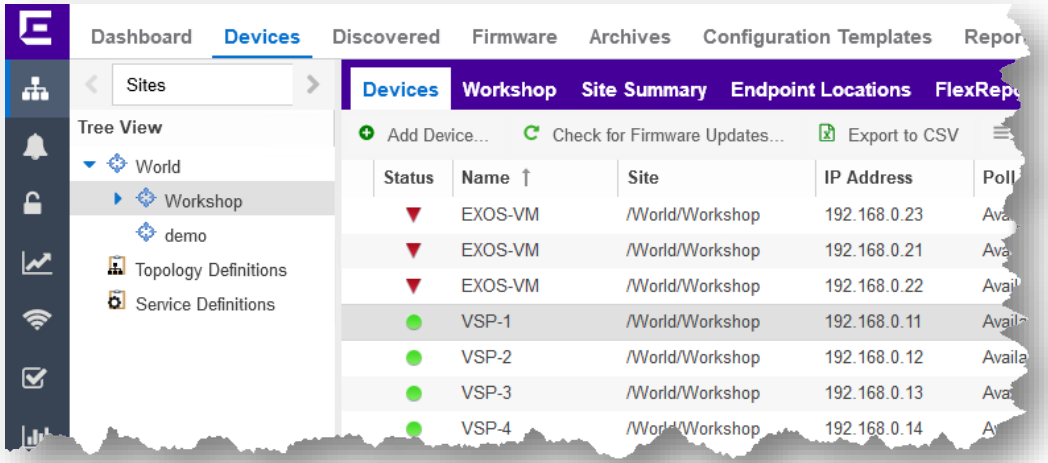
change device profile

```
mutation {
  network {
    configureDevice(input: {
      deviceConfig: {
        ipAddress: "192.168.0.11"
        generalConfig: {
          adminProfile: "public_v2_Profile"
          pollType: SNMP
          pollGroup: DEFAULT
        }
        }
      }
    }) {
      status
      message
    }
  }
}
```




```
{
  "data": {
    "network": {
      "configureDevice": {
        "status": "SUCCESS",
        "message": ""
      }
    }
  }
}
```

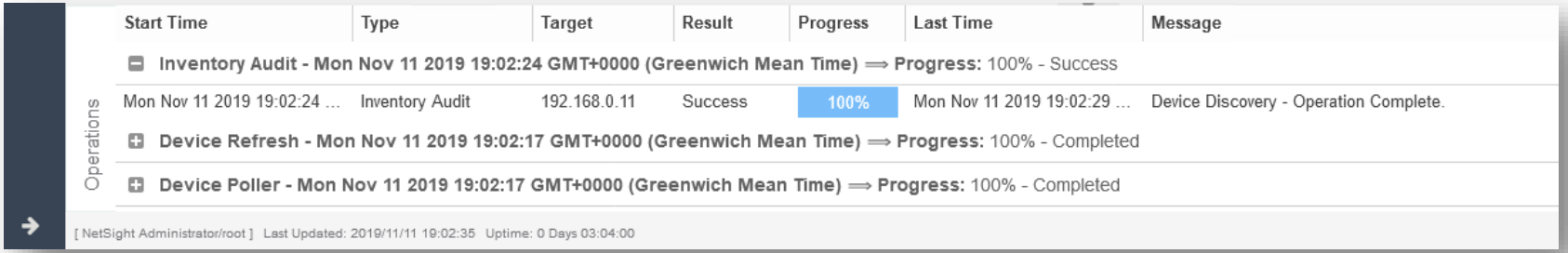
get all devices from a specific site




```
{
  network {
    devicesBySiteLocation(location: "/World/Workshop") {
      ip
      deviceDisplayFamily
    }
  }
}
```


```
{
  "data": {
    "network": {
      "devicesBySiteLocation": [
        {
          "ip": "192.168.0.11",
          "deviceDisplayFamily": "VSP Series"
        },
        {
          "ip": "192.168.0.12",
          "deviceDisplayFamily": "VSP Series"
        },
        {
          "ip": "192.168.0.13",
          "deviceDisplayFamily": "VSP Series"
        },
        {
          "ip": "192.168.0.14",
          "deviceDisplayFamily": "VSP Series"
        },
        {
          "ip": "192.168.0.21",
          "deviceDisplayFamily": "Summit Series"
        },
        . . .
```

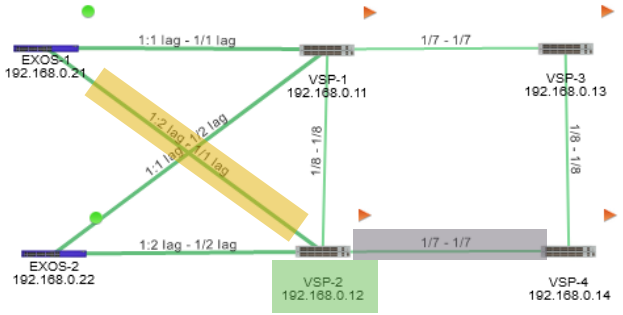
Rediscover device




```
mutation {
  network {
    rediscoverDevices(input: {
      devices: {
        ipAddress: "192.168.0.11"
      }
    }) {
      status
      message
    }
  }
}
```

```
{
  "data": {
    "network": {
      "rediscoverDevices": {
        "status": "SUCCESS",
        "message": ""
      }
    }
  }
}
```

get device neighbors

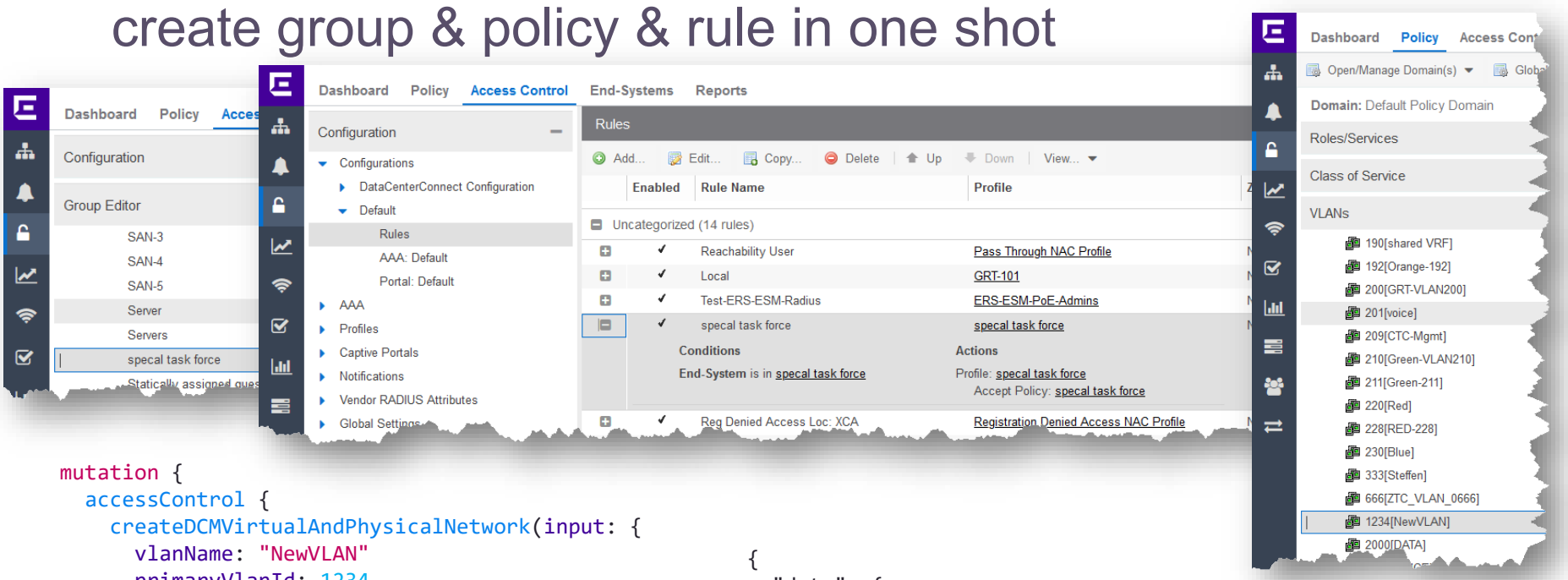



```
{ network {
    device(ip: "192.168.0.12") {
      deviceName
      sitePath
      links {
        deviceIp1
        deviceIp2
        ifName1
        ifName2
        remoteIp(localIp: "192.168.0.12")
        remoteIfName(localIp: "192.168.0.12")
      }
    }
} }
```


```
{
  "data": {
    "network": {
      "device": {
        "deviceName": "VSP-2",
        "sitePath": "/World/Workshop",
        "links": [
          {
            "deviceIp1": "192.168.0.14",
            "deviceIp2": "192.168.0.12",
            "ifName1": "1/7",
            "ifName2": "1/7",
            "remoteIp": "192.168.0.14",
            "remoteIfName": "1/7"
          },
          {
            "deviceIp1": "192.168.0.12",
            "deviceIp2": "192.168.0.21",
            "ifName1": "1/1",
            "ifName2": "1:2",
            "remoteIp": "192.168.0.21",
            "remoteIfName": "1:2"
          },    . . .
```

NAC examples

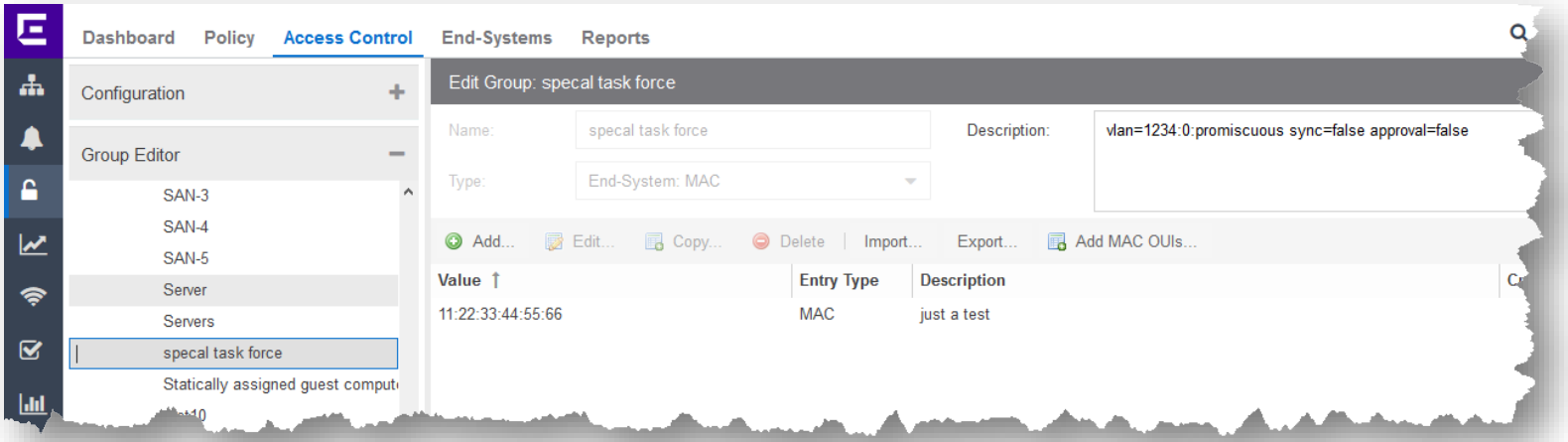
create group & policy & rule in one shot




```
mutation {
  accessControl {
    createDCMVirtualAndPhysicalNetwork(input: {
      vlanName: "NewVLAN"
      primaryVlanId: 1234
      name: "special task force"
      nacConfig: "Default"
    }) {
      status
      message
    }
  }
}
```

```
{
  "data": {
    "accessControl": {
      "createDCMVirtualAndPhysicalNetwork": {
        "status": "SUCCESS",
        "message": null
      }
    }
  }
}
```

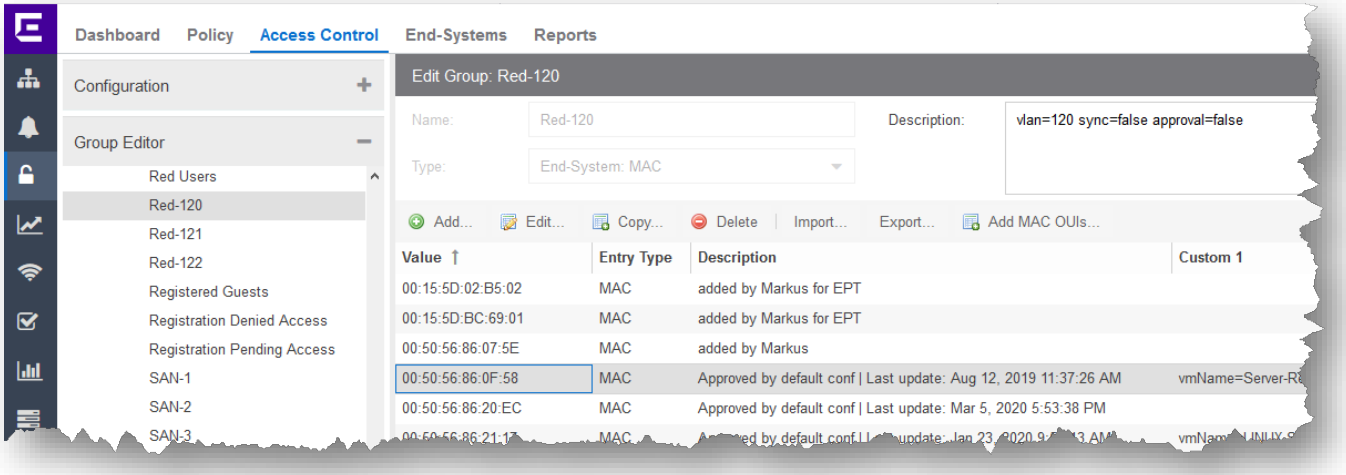
add MAC




```
mutation {
  accessControl {
    addMACToEndSystemGroup(input: {
      group: "specal task force"
      value: "11:22:33:44:55:66"
      description: "just a test"
    }) {
      status
      message
    }
  }
}
```

```
{
  "data": {
    "accessControl": {
      "addMACToEndSystemGroup": {
        "status": "SUCCESS",
        "message": null
      }
    }
  }
}
```

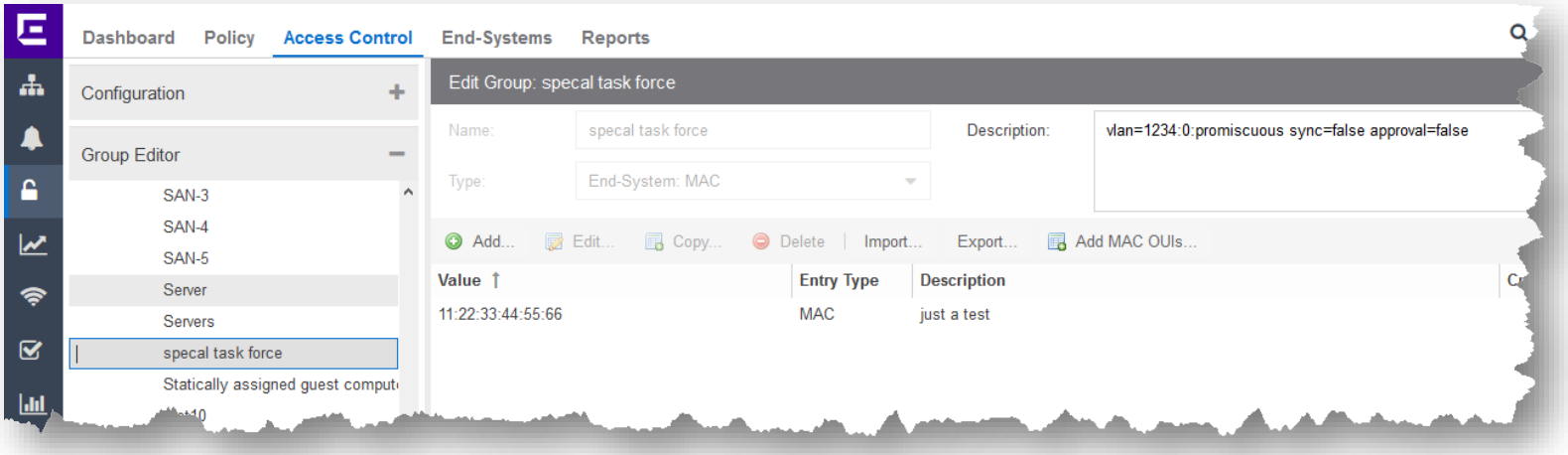
get MAC




```
{
  accessControl {
    endSystemInfoByMac(macAddress: "00:50:56:86:0F:58") {
      endSystemInfo {
        custom1
        custom2
        custom3
        custom4
        groupDescr1
      }
    }
  }
}
```

```
{
  "data": {
    "accessControl": {
      "endSystemInfoByMac": {
        "endSystemInfo": {
          "custom1": "vmName=Server-Red",
          "custom2": "",
          "custom3": "",
          "custom4": "OneView|virtual;vmware;vnic|",
          "groupDescr1": "Red-120=Approved by default conf"
        }
      }
    }
  }
}
```

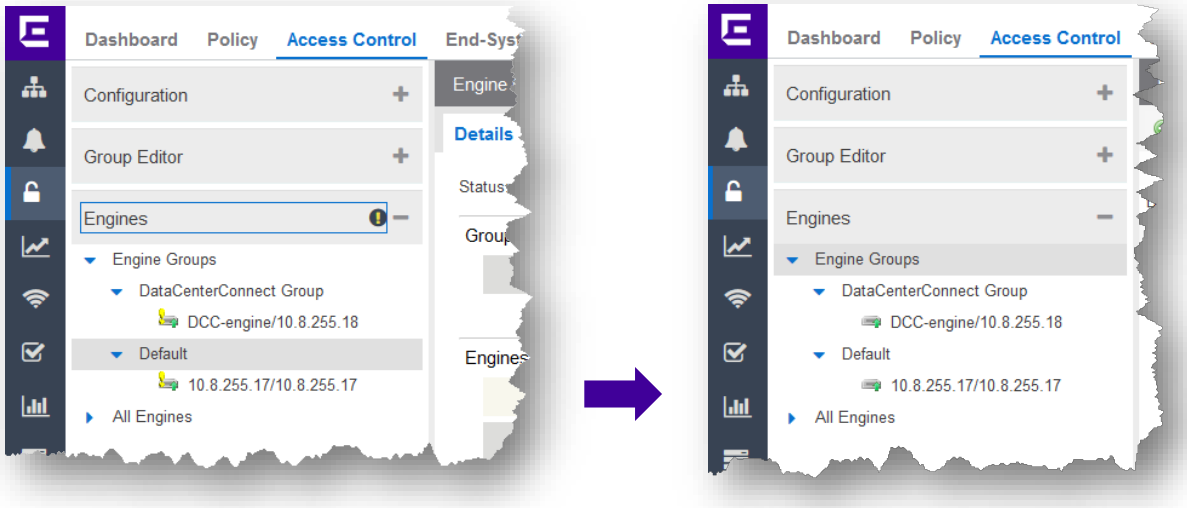
delete MAC




```
mutation {
  accessControl {
    deleteEndSystemByMac(input: {
      macAddress: "11:22:33:44:55:66"
    }) {
      status
      results
    }
  }
}
```

```
{
  "data": {
    "accessControl": {
      "deleteEndSystemByMac": {
        "status": "SUCCESS",
        "message": null
      }
    }
  }
}
```

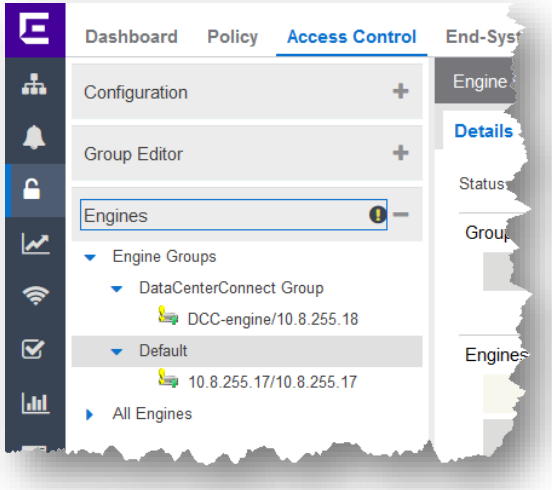
enforce on all engines

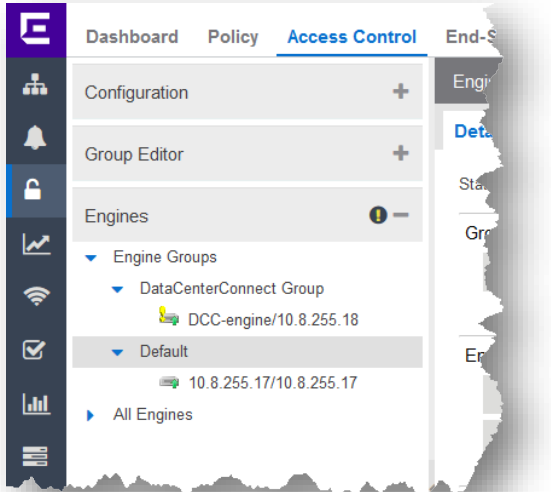



```
mutation {
  accessControl {
    enforceAllAccessControlEnginesForceSwitchesAndPortal {
      status
      message
    }
  }
}
```

```
{
  "data": {
    "accessControl": {
      " enforceAllAccessControlEnginesForceSwitchesAndPortal ": {
        "status": "SUCCESS",
        "message": null
      }
    }
  }
}
```

enforce a specific Domain or Engine change






```
mutation {
  accessControl {
    enforceAccessControlEngines(input: {
      engineGroup: "Default"
      engineIps: "10.8.255.17"
    }) {
      status
      message
    }
  }
}
```

optional

```
{
  "data": {
    "accessControl": {
      "enforceAccessControlEngines": {
        "status": "SUCCESS",
        "message": null
      }
    }
  }
}
```

GET all MAC Addresses

```
{
  accessControl {
    allEndSystemMacs
  }
}
```

```
{
  "data": {
    "accessControl": {
      "allEndSystemMacs": [
        "00:00:5E:00:01:65",
        "00:00:5E:00:02:65",
        "00:50:56:5E:C9:F9",
        ...
```

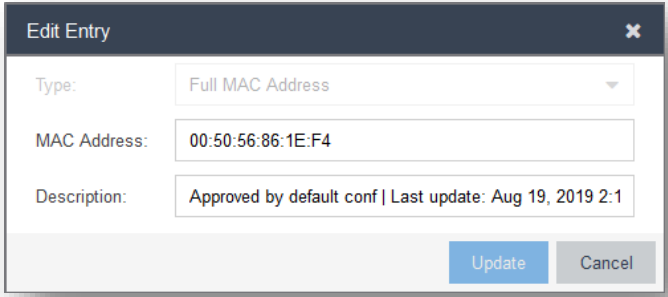
GET all groups

```
{
  accessControl {
    endSystemCategoryGroupNames
  }
}
```

```
{
  "data": {
    "accessControl": {
      "endSystemCategoryGroupNames": [
        "HA",
        "ThisGroupDoesNotExist",
        "Server",
        "Group-9",
        ...
```

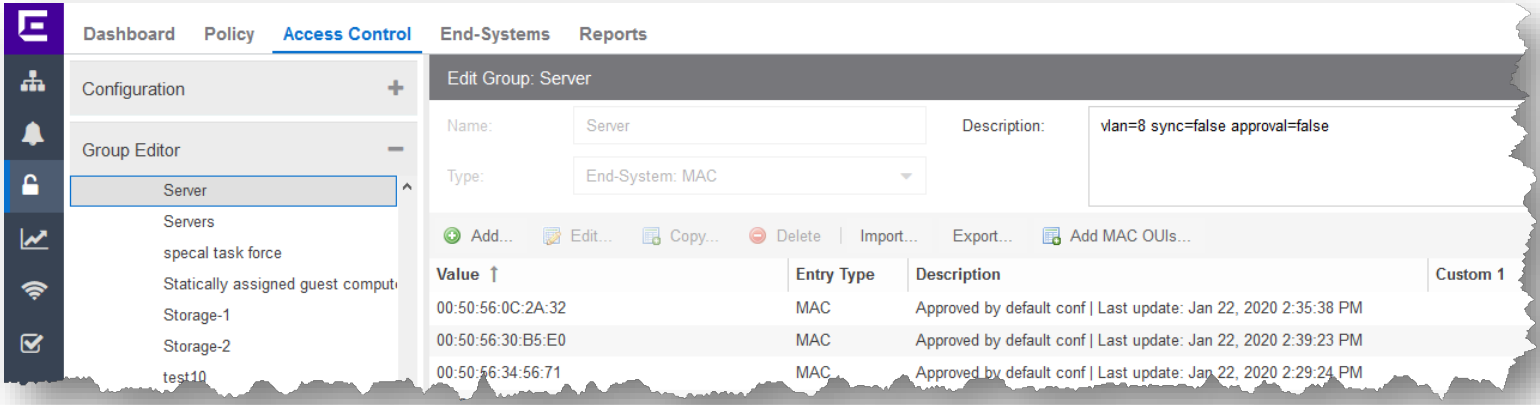

GET MAC Addresses event details

```
{
  accessControl {
    extendedEndSystemArrByMac(macAddress:"00:50:56:86:1E:F4")
  }
}
```




```
{
  "data": {
    "accessControl": {
      "extendedEndSystemArrByMac": [
        "username=",
        "enumSource=NAC_APPLIANCE",
        "switchIP=20.0.10.72",
        "macAddress=00:50:56:86:1E:F4",
        "nacApplianceGroupName=DataCenterConnect Group",
        "reason=Rule: \"Blue-130\"",
        "stateDescr=Unable to resolve IP address using SNMP, NetBIOS, or DHCP",
        "enumAuthType=AUTH_MAC_PAP",
        "lastAuthEventTime=2020-03-19 15:14:41.0",
        "switchPort=6144",
        "groupDescr1=Blue-130=Approved by default conf | Last update: Aug 19, 2019 2:16:47 PM",
        "allAuthTypes=AUTH_MAC",
        "policy=FA-VLAN-ISID='130:0', Session-Timeout='1200'",
        ...
```

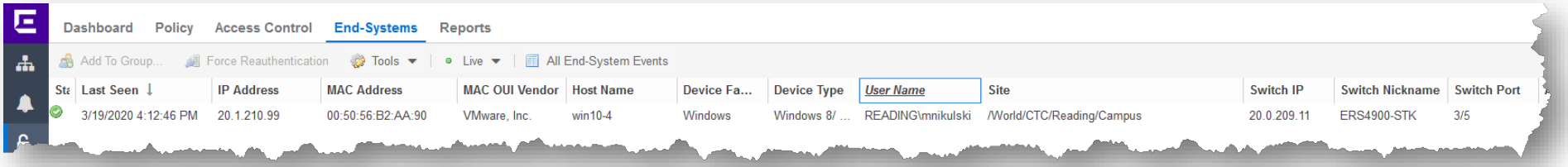
GET group details




```
{
  accessControl {
    group(name: "Server") {
      description
      name
      typeStr
      values
      valueDescriptions
    }
  }
}
```

```
{
  "data": {
    "accessControl": {
      "group": {
        "description": "vlan=8 sync=false approval=false ",
        "name": "Server",
        "typeStr": "MAC“
        "values": [
           "00:50:56:86:4A:05",
           "00:50:56:86:3D:4C",
           . . .
```

End-System status information

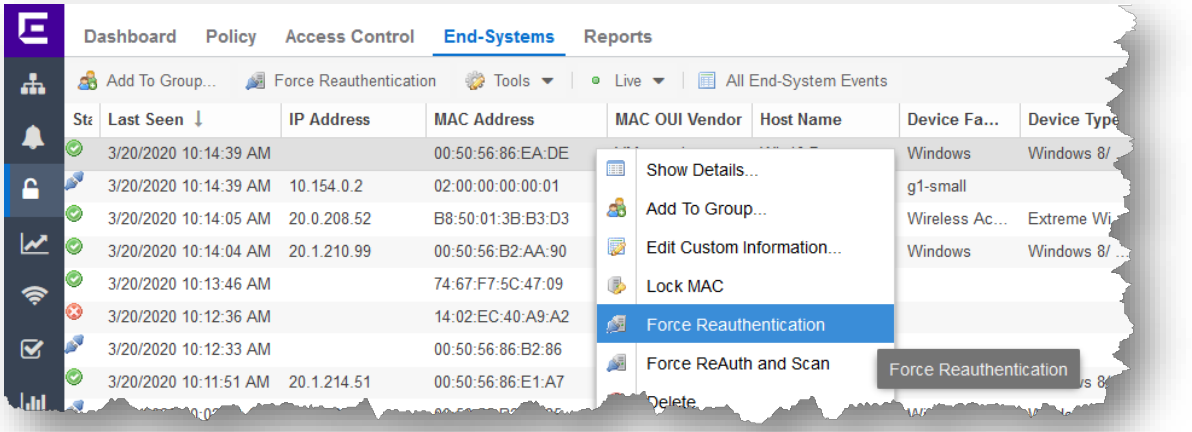



```
{
  accessControl {
    endSystems(maxResults: 100, firstResult: 0) {
      endSystems {
        macAddress
        state
        stateDescr
        reason
        firstSeenTime
        lastSeenTime
        username
        hostName
        ipAddress
        switchIP
        switchPortId
      }
    }
  }
}
```


```
{
  "data": {
    "accessControl": {
      "endSystems": {
        "endSystems": [
          {
            "macAddress": "00:50:56:B2:AA:90",
            "state": "ACCEPT",
            "stateDescr": "",
            "reason": "Rule: \"CSE-Team\"",
            "firstSeenTime": "2019-08-14T15:34:24",
            "lastSeenTime": "2020-03-19T15:12:46",
            "username": "READING\\mnikulski",
            "hostName": "DHCPFP:win10-4",
            "ipAddress": "20.1.210.99",
            "switchIP": "20.0.209.11",
            "switchPortId": "3/5 "
          },
```


The **firstSeenTime** timestamp is updated every time the End System details becomes updated like hostname change, IP changed/DHCP, Kerberos snooped hostname/username...

MAC re-authentication

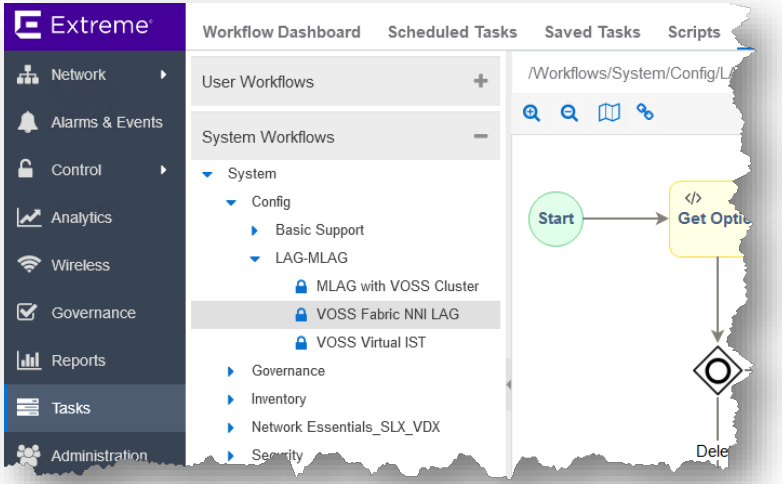



```
mutation {
  accessControl {
    reauthenticate(input: { macAddress: "00:50:56:86:EA:DE"  } )
    {
      status
      message
    }
  }
}
```

```
{
  "data": {
    "accessControl": {
      "reauthenticate": {
        "status": "SUCCESS",
        "message": null
      }
    }
  }
}
```

Workflow execution examples

get all Workflows

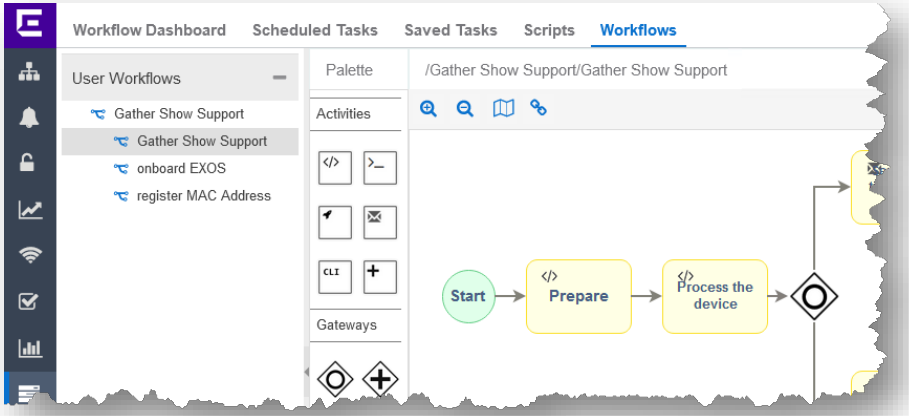



```
{
  workflows {
    allWorkflows {
      id
      name
      category
      groupPath
      workflowScopes
    }
  }
}
```

```
{
  "data": {
    "workflows": {
      "allWorkflows": [
        {
          "id": 1,
          "name": "VOSS Fabric NNI LAG",
          "category": "Config",
          "groupPath": "/Workflows/System/Config/LAG-MLAG",
          "workflowScopes": [
            "DEVICE",
            "MULTIDEVICE"
          ]
        },
        . . . ,
        . . . ,
        {
          "id": 65,
          "name": "Gather Show Support",
          "category": "System",
          "groupPath": "/Workflows",
          "workflowScopes": [
            "DEVICE",
            "ALARM"
          ]
        }
      ]
    }
  }
}
```

long list of records

get Workflow details

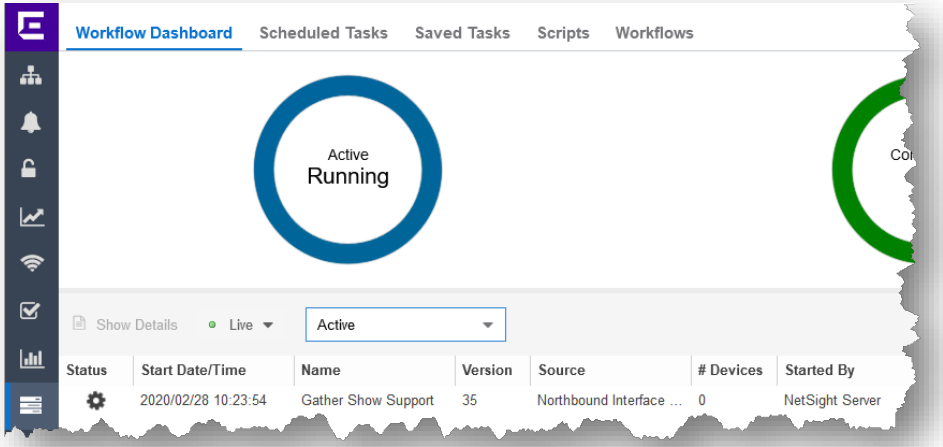



```
{
 workflows {
   workflow(id: 65) {
     name
       inputs {
         inputs {
           name
           displayName
           type
           value
         }
       }
     }
   }
}
```

```
{
  "data": {
    "workflows": {
      "workflow": {
        "name": "Gather Show Support",
        "inputs": [
          {
            "inputs": [
              {
                "name": "workflowTimeout",
                "displayName": "Timeout",
                "type": "NUMBER",
                "value": "600"
              }
            ]
          },
          {
            "inputs": [
              {
                "name": "EmailTo",
                "displayName": "Send email to",
                "type": "STRING",
                "value": ""
              }
            ]
          }
        ]
      }
    }
  }
}
```

string

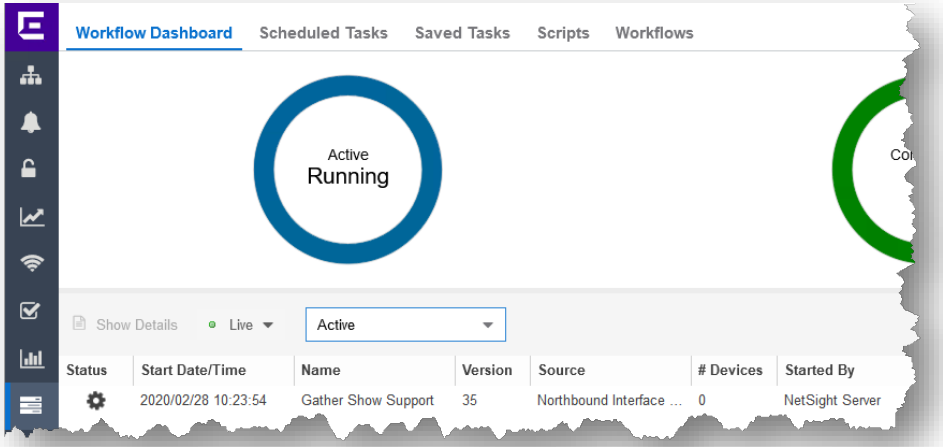
execute a Workflow




```
mutation {
  workflows {
    startWorkflow (input: {
                     id: 65
                     variables: {
                       device: "10.8.255.17"
                       EmailTo: "user0@extreme.lab"
                     }
                   }) {
      status
      executionId
      errorCode
      message
    }
  }
}
```

```
{
  "data": {
    "workflows": {
      "startWorkflow": {
        "executionId": 12,
        "errorCode": 0,
        "message": "",
        "status": "SUCCESS",
        "workflowId": null
      }
    }
  }
}
```

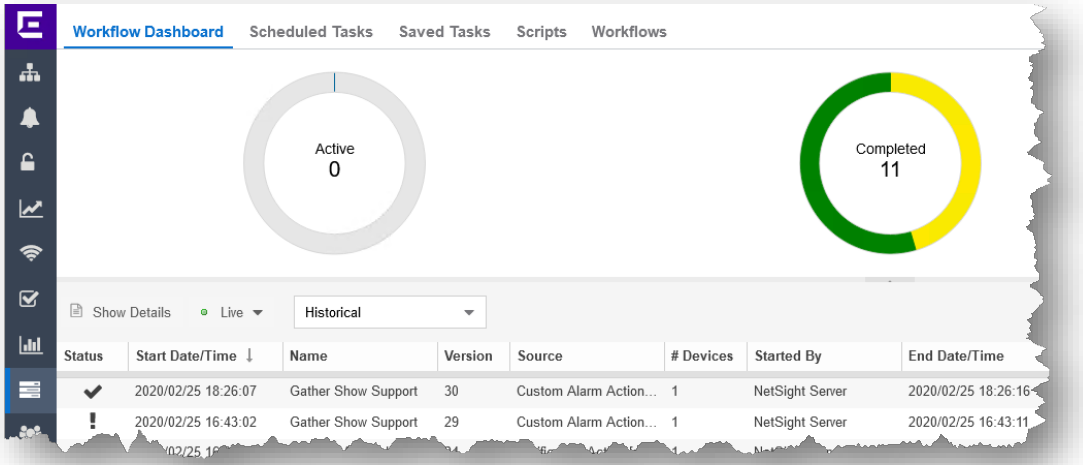
check a running Workflow




```
{
  workflows {
    activeExecutions {
      id
      status
      startTime
      endTime
      workflowName
      workflowPath
    }
  }
}
```

```
{
  "data": {
    "workflows": {
      "activeExecutions": [
        {
          "id": 0,
          "status": "RUNNING",
          "startTime": 1582883698000,
          "endTime": 0,
          "workflowName": "Gather Show Support",
          "workflowPath": "/Workflows/Gather Show Support"
        },
        . . . ,
```

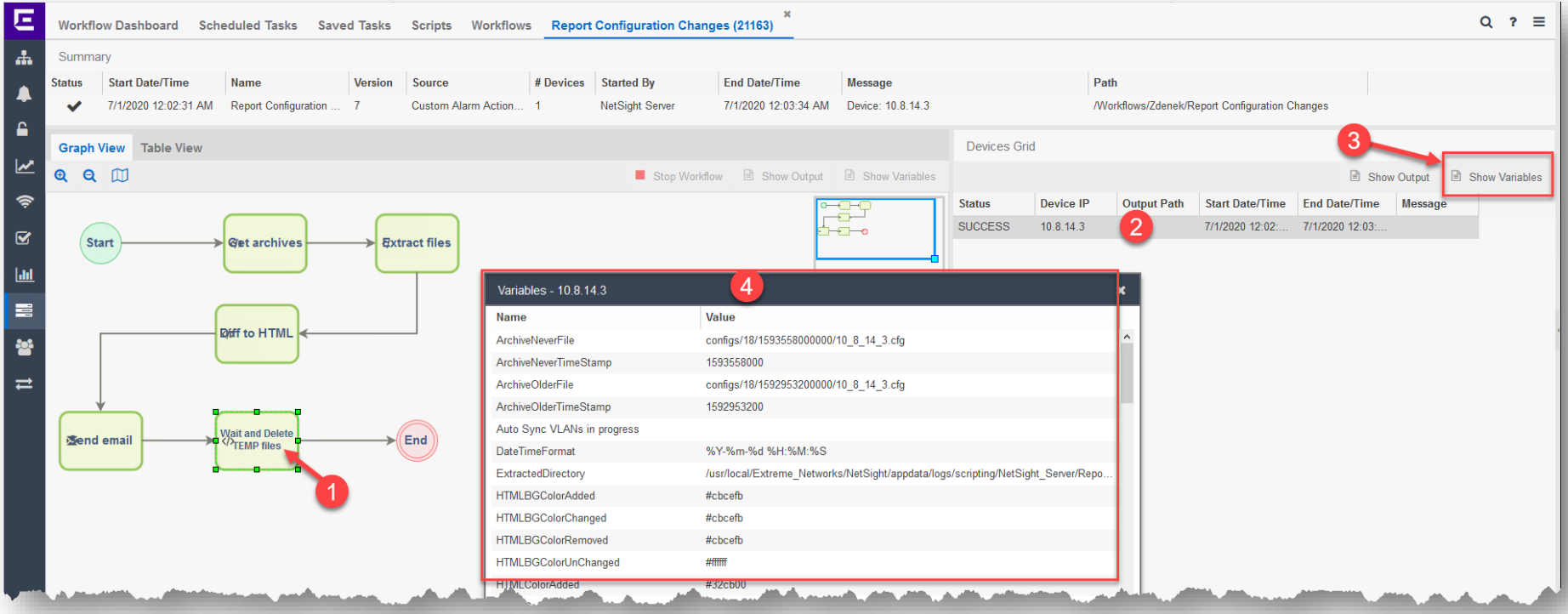
check a completed Workflow




```
{
  workflows {
    completedExecutions {
      id
      status
      user
      startTime
      endTime
      workflowName
      variables
    }
  }
}
```

```
{
  "data": {
    "workflows": {
      "completedExecutions": [
        {
          "id": 12,
          "status": "SUCCESS",
          "user": "NetSight Server",
          "startTime": 1582883698000,
          "endTime": 1582883709000,
          "workflowName": "Gather Show Support",
          "variables": "{ . . . }"
        },
        . . . ,
```

check a completed Workflow (variables)



check a completed Workflow (variables)

```
{
  workflows {
    completedExecutions(limit: 2, start: 0) {
      id
      workflowName
      status
      startTime
      endTime
      version
      user
      message
      variables
    }
  }
}
```


```
{
  "data": {
    "workflows": {
      "completedExecutions": [
        {
          "id": 21163,
          "workflowName": "Report Configuration Changes",
          "status": "SUCCESS",
          "startTime": 1593558151000,
          "endTime": 1593558214000,
          "version": 7,
          "user": "NetSight Server",
          "message": "Device: 10.8.14.3",
          "variables": {
            "ArchiveNeverFile": "configs/18/1593558000000/10_8_14_3.cfg",
            "ArchiveNeverTimeStamp": 1593558000,
            "ArchiveOlderFile": "configs/18/1592953200000/10_8_14_3.cfg",
            "ArchiveOlderTimeStamp": 1592953200,
            "DateTimeFormat": "%Y-%m-%d %H:%M:%S",
            "ExtractedDirectory": "/usr/local/Extreme_Networks/NetSight/appda
            "HTMLBGColorAdded": "#cbcefb",
            "HTMLBGColorChanged": "#cbcefb",
            "HTMLBGColorRemoved": "#cbcefb",
            "HTMLBGColorUnChanged": "#ffffff",
            "HTMLColorAdded": "#32cb00",
            "HTMLColorChanged": "#9a0000",
            "HTMLColorHeader": "#6200c9",
            "HTMLColorMiddle": "#000000",
            "HTMLColorRemoved": "#3531ff",
            "HTMLColorUnChanged": "#000000",
            "HTMLbodyAdd": "",
            "I-SID_prefix": "1234",
```

NBI internal use

NBI internal use

```
#####################################################################
# determinate device site relationship
def getSite(ip):
    query = '''
{ network {
    device(ip: "<ip>") {
        sitePath
    }
  }
}
'''

    result = emc_nbi.query( query.replace('<ip>', ip) )

    return result['network']['device']['sitePath']

#####################################################################

deviceSitePath = getSite( emc_vars["deviceIP"] )
```

NBI internal use

```
#####################################################################
# determinate all devices belong to the site
def getSiteMembers(site):
    switches = []
    query = '''
{ network {
    siteByLocation(location: "<site>") {
      devList
    }
  }
}
'''

    result = emc_nbi.query( query.replace('<site>', site) )

    # ignore own switch IP
    for ip in result['network']['siteByLocation']['devList']:
        if ip != emc_vars["deviceIP"]:
            switches.append( ip )

    return switches

#####################################################################
deviceSitePath = getSite( emc_vars["deviceIP"] )
neighbors = getSiteMembers( deviceSitePath )
```

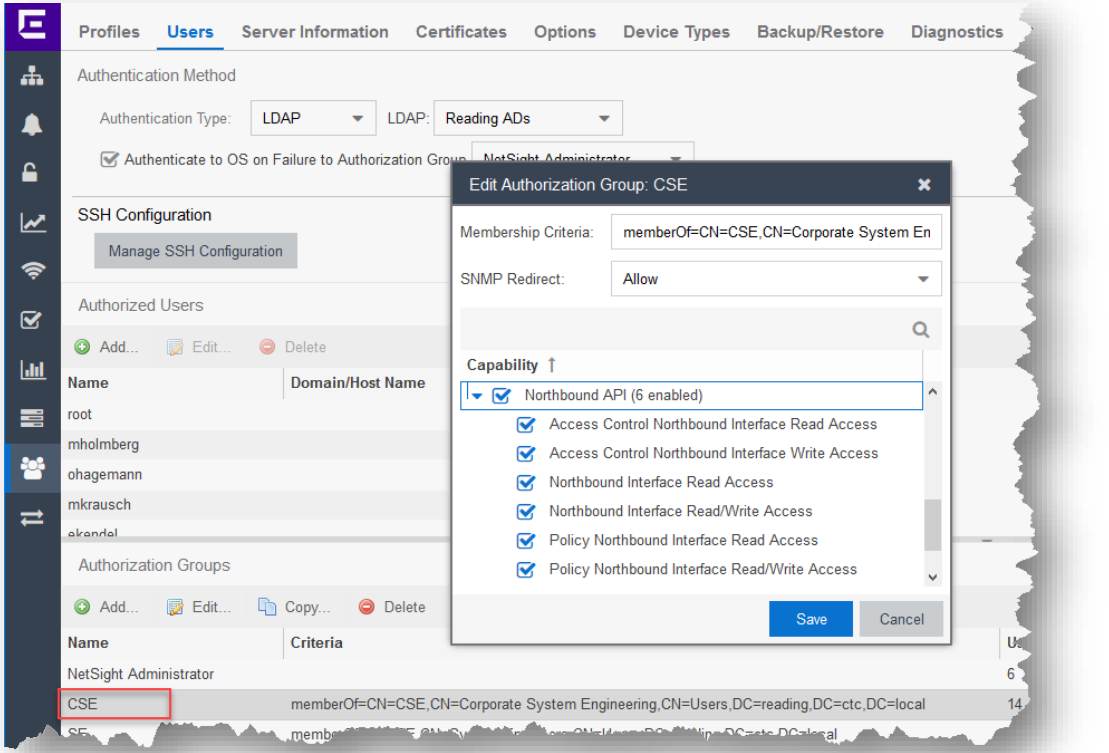
NBI external use

XMC Authentication Methods

Method is not recommended

Basic Authentication

Basic Authentication: user-name / password



is a regular user from XMC

access rights applied on group

XMC Authentication Methods

Method is not recommended

Basic Authentication

prepare

```
#!/usr/bin/env python

import json
import requests
from requests import Request, Session
from requests.auth import HTTPBasicAuth
from requests.packages.urllib3.exceptions import InsecureRequestWarning

# To disable SSL certificate verification
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

# define variables
xmcServerIp = '192.168.0.201'
xmcUser     = 'root'
xmcPassword = 'lassMichRein!'
nbiUrl      = 'https://' + xmcServerIp + ':8443/nbi/graphql'

# prepare HTTPs session
session         = Session()
session.verify  = False
session.timeout = 10
session.auth    = (xmcUser, xmcPassword)

session.headers.update({'Accept':           'application/json',
                        'Content-type':     'application/json',
                        'Cache-Control':    'no-cache',
                        })
```

query

```
# define NBI query
dataQery = '{network{devices{ip nickName }}}'

# execute NBI call
response = session.post(nbiUrl, json= {'query': dataQery} )

# validate result
if response.status_code != 200:
  print 'ERROR: HTTP ' + response.reason + ' (' + str(response.status_code) + ')'
else:
  callTine = float("{0:0.1f}".format( response.elapsed.total_seconds() * 1000 ))
  print 'INFO: query time [%s ms]' % (callTine)
```

output

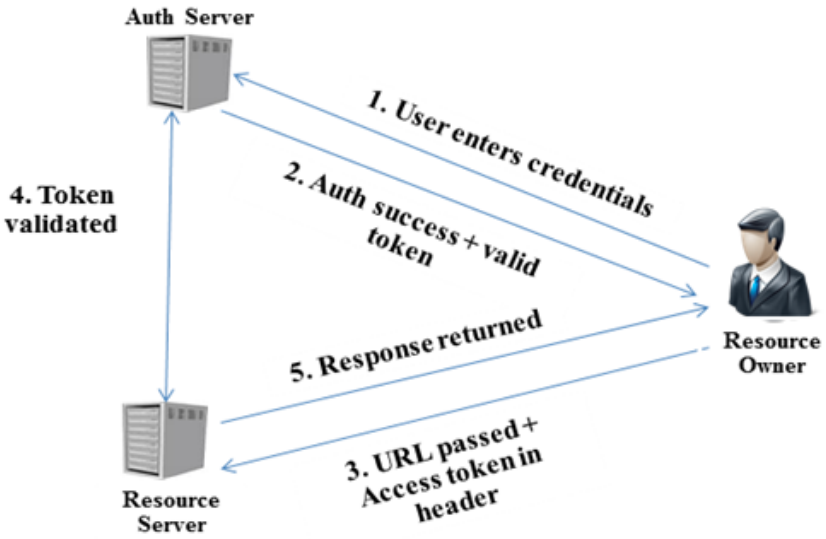
```
# convert JSON string to an data structure
inbound_data = json.loads(response.text)

# walk trough device list
for device in inbound_data['data']['network']['devices'] :
  print device['ip'] + ' \t' + device['nickName']
```

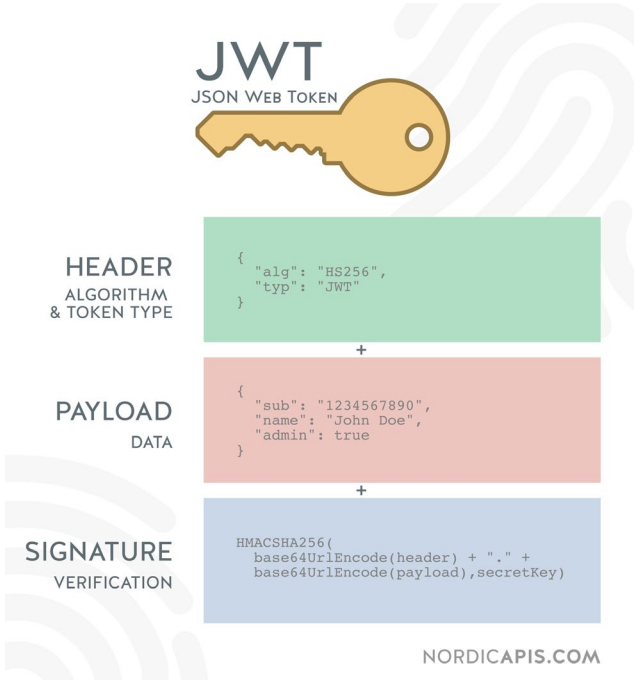
```
INFO: query time [293.6 ms]
192.168.162.11  VSP-1
192.168.162.12  VSP-2
192.168.162.22  EXOS-2
192.168.162.21  EXOS-1
```

XMC Authentication Methods

Method is recommended

OAuth2





https://jwt.io/

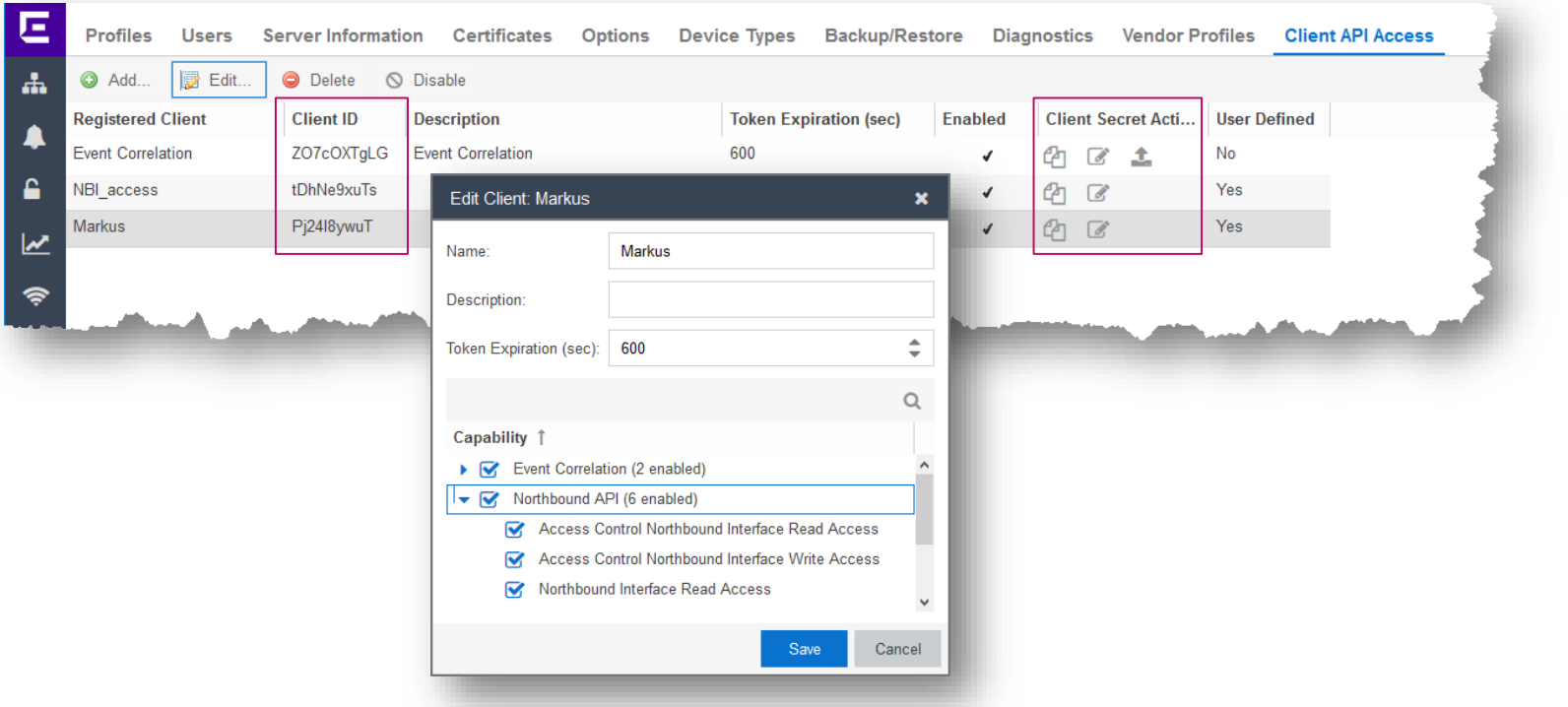
XMC Authentication Methods

Method is recommended

OAuth2

OAuth2 : Client-ID / secret

only for NBI calls
session token based
session time decoupled from WEB-UI



XMC Authentication Methods (login)

Method is recommended

OAuth2

prepare

```
#!/usr/bin/env python

# import all required classes
import json
import base64
from datetime import datetime
import requests
from requests import Request, Session
from requests.auth import HTTPBasicAuth
from requests.packages.urllib3.exceptions import InsecureRequestWarning

# To disable SSL certificate verification
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

# define variables
xmcServerIp = '192.168.162.50'
xmcClientID = 'vvlGKVjedc'
xmcSecret   = '02cb116f-9a64-465a-b430-8a76136ba08d'
xmcToken    = ''
nbiUrl      = 'https://' + xmcServerIp + ':8443/nbi/graphql'
```

prepare session

```
# prepare HTTPs session
session         = Session()
session.verify  = False
session.timeout = 10
session.headers.update({'Accept':          'application/json',
                        'Content-type':    'application/json',
                        'Authorization':   'Bearer ' + xmcToken,
                        'Cache-Control':   'no-cache',
                        })
```

login

```
# login and get session token
tokenurl = 'https://' + xmcServerIp + ':8443/oauth/token/access-token?grant_type=client_credentials'
headers = {"Content-Type" : "application/x-www-form-urlencoded"}

response = requests.post(tokenurl, auth=(xmcClientID, xmcSecret), headers=headers, verify=False)

if response.status_code == requests.codes.ok:
    callTine = float("{0:0.1f}".format( response.elapsed.total_seconds() * 1000 ))
    print 'INFO: successful login [%s ms]' % (callTine)
    result = response.json()
    xmcToken  = result[u'access_token']
```

```
INFO: successful login [236.9 ms]
```

XMC Authentication Methods (query)

Method is recommended

OAuth2

query

```
# define NBI query
dataQery = '{network{devices{ip nickName }}}'

# execute NBI call
response = session.post(nbiUrl, json= {'query': dataQery} )

# validate result
if response.status_code != 200:
  print 'ERROR: HTTP ' + response.reason + ' (' + str(response.status_code) + ')'
else:
  callTine = float("{0:0.1f}".format( response.elapsed.total_seconds() * 1000 ))
  print 'INFO: query time [%s ms]' % (callTine)
```

output

```
  # convert JSON string to an data structure
  inbound_data = json.loads(response.text)

  # walk trough device list
  for device in inbound_data['data']['network']['devices'] :
    print device['ip'] + ' \t' + device['nickName']
```

```
INFO: query time [45.3 ms]
192.168.162.11  VSP-1
192.168.162.12  VSP-2
192.168.162.22  EXOS-2
192.168.162.21  EXOS-1
```

XMC Authentication Methods

Method is recommended

OAuth2

Token decoding

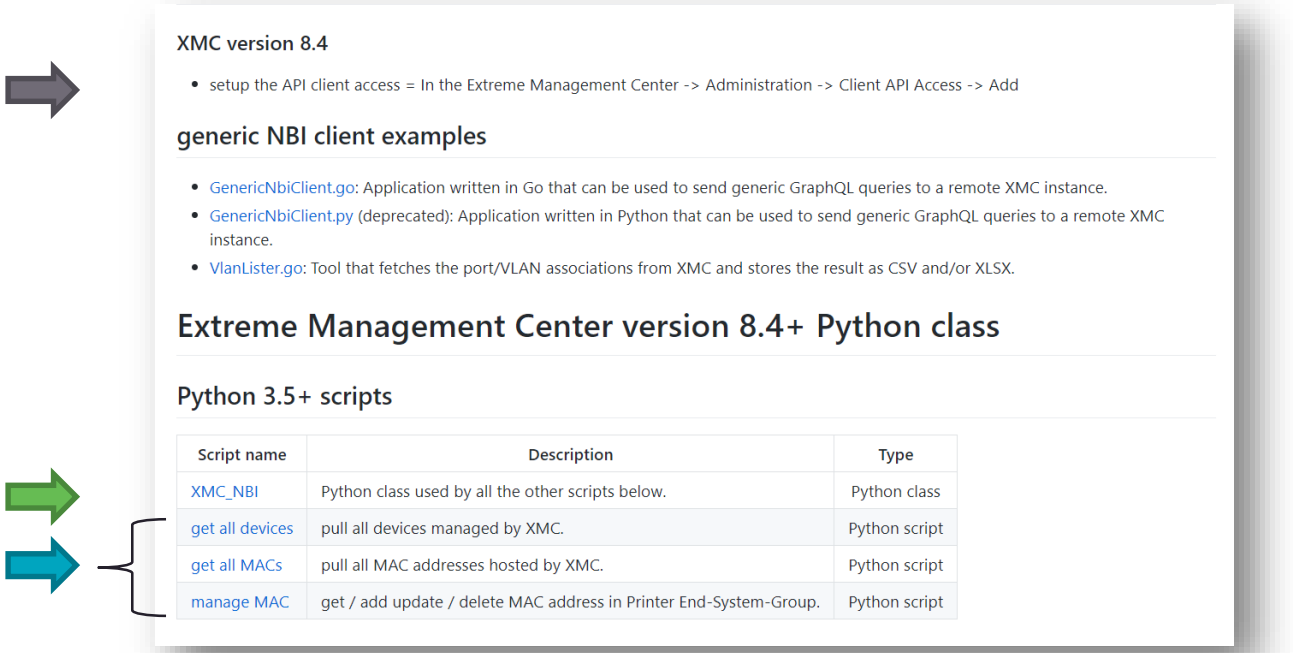
```
import base64
from datetime import datetime

result = response.json()
xmcToken  = result[u'access_token']
xmcTokenElements = xmcToken.split('.')
tokenData = json.loads( base64.b64decode(xmcTokenElements[1]+ "==") )
print 'SESSION TOKEN DATA:'
print '            Issuer: %s' % tokenData['iss']
print '           Subject: %s' % tokenData['sub']
print '            JWT ID: %s' % tokenData['jti']
print '         XMC-Roles: %s' % tokenData['roles']
print '         Issued at: %s' % datetime.fromtimestamp( tokenData['iat'] )
print '   Expiration Time: %s' % datetime.fromtimestamp( tokenData['exp'] )
print '        Not Before: %s' % datetime.fromtimestamp( tokenData['nbf’] )
```

```
SESSION TOKEN DATA:
            Issuer: xmc.extreme.lab
           Subject: vvlGKVjedc
            JWT ID: ce6ce5bc-ad8d-47ce-8977-7cbd3cb62f87
         XMC-Roles: [u'NetSightUser']
         Issued at: 2019-11-12 09:15:35
   Expiration Time: 2019-11-12 09:16:35
        Not Before: 2019-11-12 09:15:35
```

With many calls and long-term use,
it is wise to track the validity period of a token

XMC Python3 class

- Using a Python class
- https://github.com/extremenetworks/ExtremeScripting/tree/master/Netsight/nbi_clients



XMC version 8.4

- setup the API client access = In the Extreme Management Center -> Administration -> Client API Access -> Add

generic NBI client examples

- GenericNbiClient.go: Application written in Go that can be used to send generic GraphQL queries to a remote XMC instance.
- GenericNbiClient.py (deprecated): Application written in Python that can be used to send generic GraphQL queries to a remote XMC instance.
- VlanLister.go: Tool that fetches the port/VLAN associations from XMC and stores the result as CSV and/or XLSX.

Extreme Management Center version 8.4+ Python class

Python 3.5+ scripts

Script name	Description	Type
XMC_NBI	Python class used by all the other scripts below.	Python class
get all devices	pull all devices managed by XMC.	Python script
get all MACs	pull all MAC addresses hosted by XMC.	Python script
manage MAC	get / add update / delete MAC address in Printer End-System-Group.	Python script

XMC Python3 class

- How to use the XMC_NBI class

use class

login

pull devices

process data

```
#!/usr/bin/env python

import pprint
import XMC_NBI

######################################################################
xmcServerIp = '192.168.162.50'
xmcClientID = 'DVDnjOaqMQ'
xmcSecret   = '8c12a0e1-87f8-4b18-a8d1-2e8c74d27c65'

######################################################################

session = XMC_NBI.XMC_NBI(xmcServerIp, xmcClientID, xmcSecret)

data = session.getDevices()

pprint.pprint( data )
```

```
[{'ip': '192.168.162.2‘,  'nickName': 'Laptop'},
 {'ip': '192.168.162.21', 'nickName': 'EXOS-1'},
 {'ip': '192.168.162.22', 'nickName': 'EXOS-2'},
 {'ip': '192.168.162.31', 'nickName': 'cisco-
1'},
 {'ip': '192.168.162.51', 'nickName':
'Control'}]
```

Next Presentation

Use the following link to advance to the next PDF in the Workflow education presentation.

WWW.EXTREMENETWORKS.COM