ACI Deep Dive
Object Model

Roland Ducomble rducombl@cisco.com

Cisco TS Technical Leader – ACI Solution Support Team

CCIE 3745

Last update : 26th Sept 2017

Object Model

Managed Object (MO) in ACI

- Everything in ACI is represented by a Managed Object (MO)
- MOs are organized in a Managed Information Tree (MIT)
- You can query or view the MIT in many different ways:
  - Visore : https://apicIP/visore.html
  - Browsing MIT in shell : cd /mit/… or cd /aci
  - Moquery : cli query to the DB
  - REST : postman, curl GET and POST
  - icurl (local REST client on apic/leaf)
  - Python SDK

Filter

Class or DN: fvEpP

Property: Op: == Val1: Val2:

Run Query

Display URI of last query

Display last response

fvEpP ?

bdDefDn	uni/bd-[uni/tn-Coke/BD-CokeBD]-isSvc-no
bdDefStQual	none
childAction	
ctxDefDn	
ctxDefStQual	none
ctxSeg	0
deplSt	deployable
descr	
dn	uni/epp/fv-[uni/tn-Coke/ap-InsiemePortal/epg-WEB]
enfPref	hw
epgDn	
epgName	WEB
epgPKey	uni/tn-Coke/ap-InsiemePortal/epg-WEB
l2FDSeg	0
l3CtxEncap	unknown
lcOwn	resolveOnBehalf
modTs	2014-05-30T22:20:24.934+00:00
monPolDn	uni/tn-common/monepg-default
name	
npName	InsiemePortal
operSt	allocated
ownerKey	
ownerTag	
pcTag	44034
prio	unspecified
scopeId	2
status	
tName	Coke

DN of the BDDef used by this EpP

DN of the CtxDef used by this EpP

Here Resolved model object

For an Epg (fvEpP)

Use arrow to browse the parent/child hierarchy

DN of the Epg corresponding to this EpP

Other EPg parameters

Click on link to jump to different Part of the MIT:

- To the Resolved BD
- To the Logical Epg

Layout of a MO

```
admin@pod2-apic1:~> moquery -d uni/tn-DC/BD-BD1
Total Objects shown: 1

# fv.BD
name                    : BD1
arpFlood                : no
bcastP                  : 225.0.24.80
childAction             :
descr                   :
dn                      : uni/tn-DC/BD-BD1
epMoveDetectMode        :
lcOwn                   : local
limitIpLearnToSubnets   : no
llAddr                  : ::
mac                     : 00:22:BD:F8:19:FF
modTs                   : 2015-12-22T11:29:24.534+01:00
monPolDn                : uni/tn-common/monepg-default
mtu                     : inherit
multiDstPktAct          : bd-flood
ownerKey                :
ownerTag                :
pcTag                   : 16387
rn                      : BD-BD1
scope                   : 2097153
seg                     : 15335345
status                  :
uid                     : 15374
unicastRoute            : no
unkMacUcastAct          : flood
```

HereQuery for a dn (-d) returns the exact dn used as arg
Other option query for a class –c returna all MO of that class

The class of the MO which is displayed

Dn : The “address” of the Mo in the MIT.
Here a BD in tenant DC with name BD1

All the rest of various properties of the object (arpFlood, BcastP, scope,….)
Those properties are different for every classes

moquery – some examples

- Find all EPGs with access encapsulation VLAN 3399.
  - Class query , requesting output in json and filtering on encaps property of the class :

```
moquery -c fvRsPathAtt -o json -f ‘fv.RsPathAtt.encap=="vlan-3399"‘
```

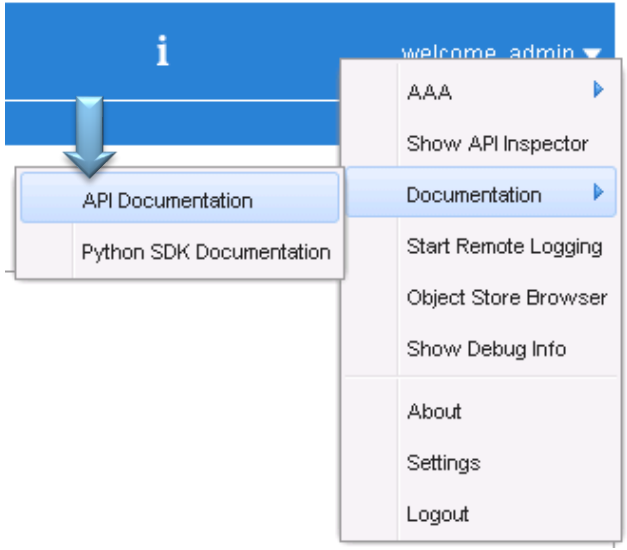
- Finding any interface where the ingress 5 min average pps is above 1000:

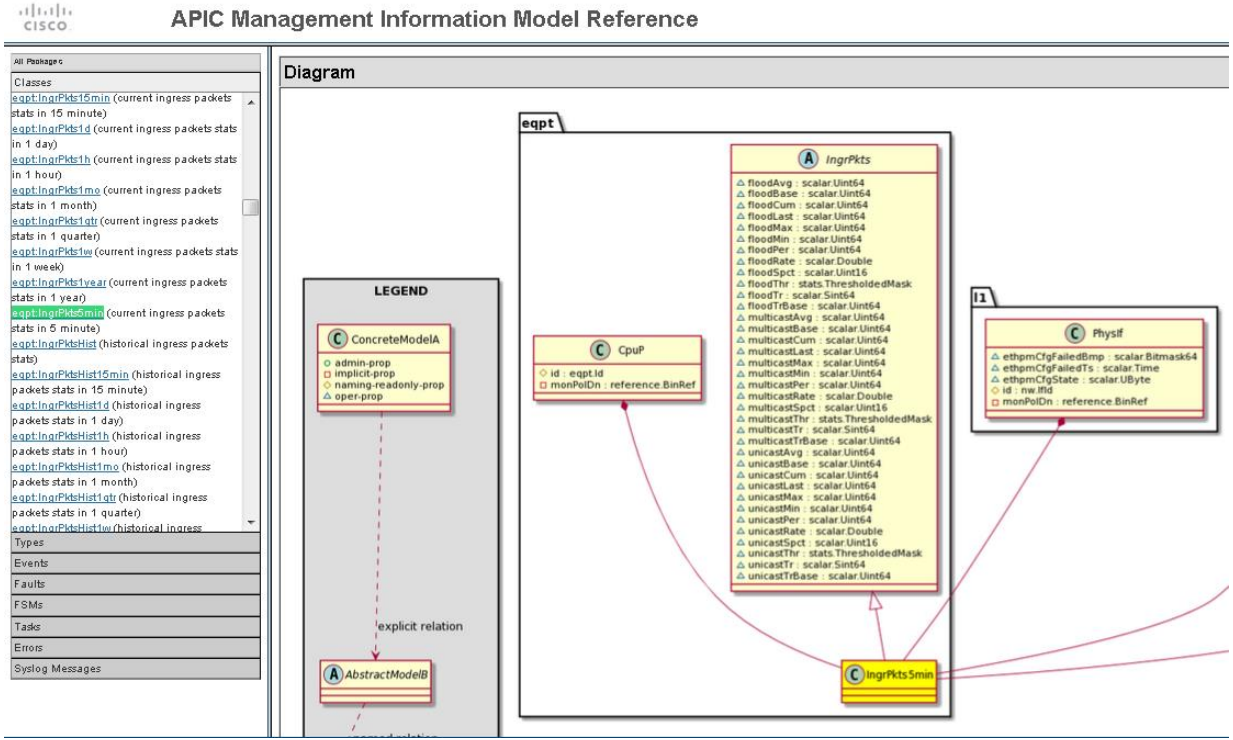
```
moquery -c eqptIngrPkts5min -f 'eqpt.IngrPkts5min.unicastRate>"1000"' | egrep -e
"^dn|^unicastRate"
dn                  : topology/pod-1/node-101/sys/phys-[eth1/34]/CDeqptIngrPkts5min
unicastRate         : 1742.12
```

- Q: that’s cool, but how do I know which object/class to query …?
- Q: it looks cryptic to me ... how do I find meaning of each field?

APIC Management Information Model Reference

From the WebUI

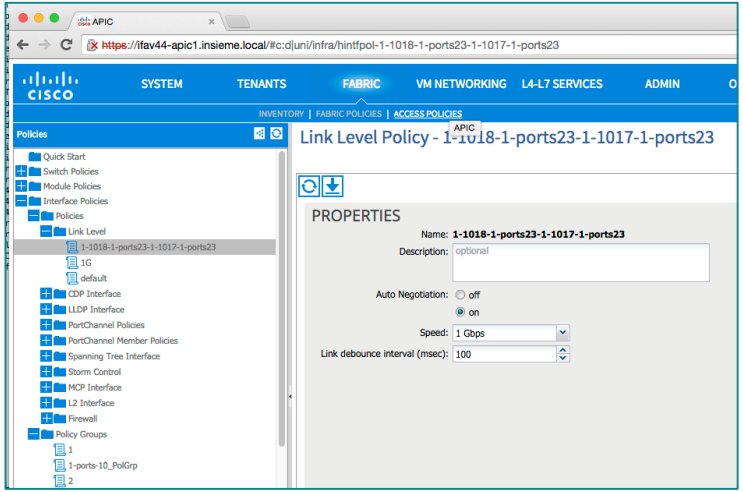




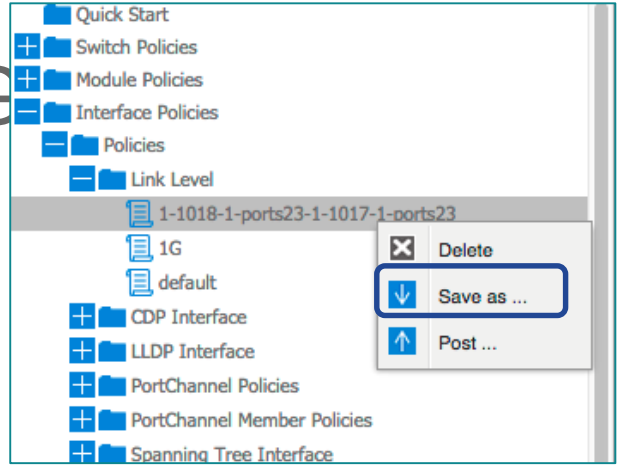
direct URL

https://apic/doc/html/

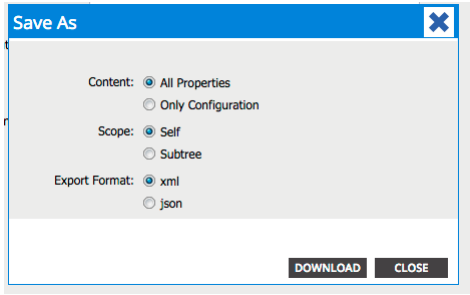
How to get object DN from G



1



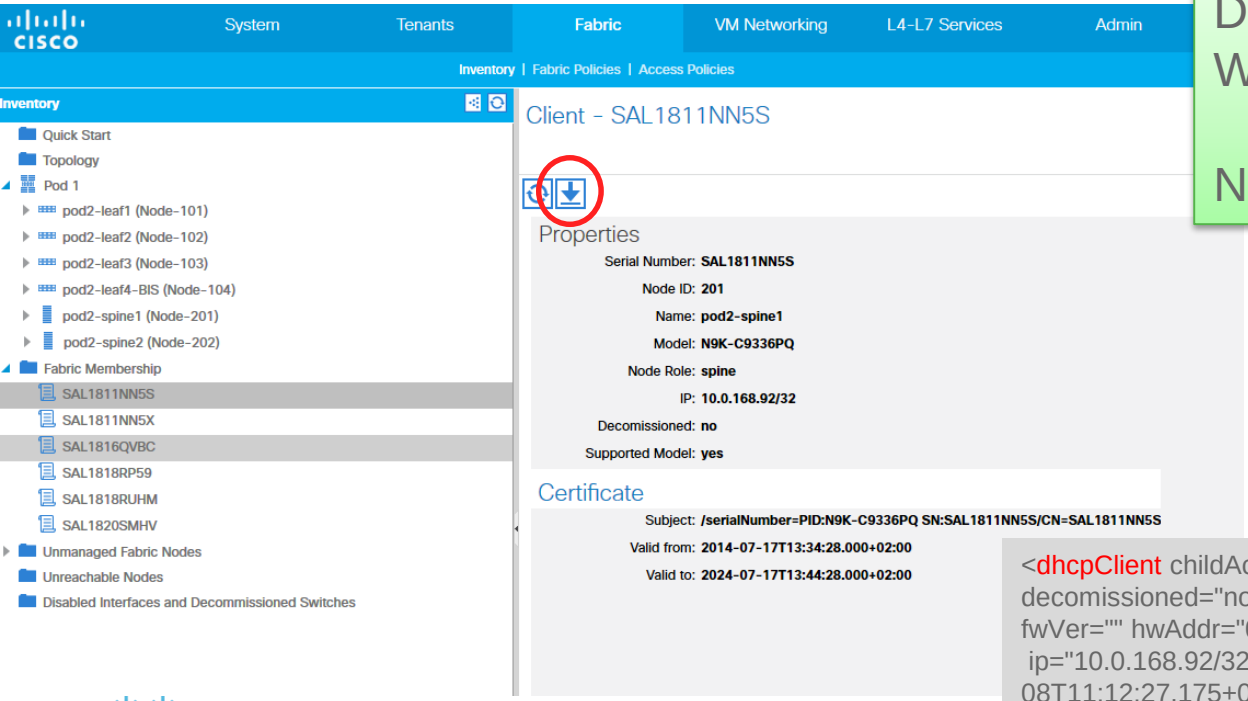
2



3


```xml
<?xml version="1.0" encoding="UTF-8"?><imdata totalCount="1"><fabricHIfPol autoNeg="on" childAction="" descr="" dn="uni/infra/hintfpol-1-1018-1-ports23-1-1017-1-ports23" lcOwn="local" linkDebounce="100" modTs="2015-03-09T15:47:24.023+00:00" monPolDn="" name="1-1018-1-ports23-1-1017-1-ports23" ownerKey="" ownerTag="" speed="1G" status="" uid="15374"/></imdata>
```

GUI download XML do not show all you need ?

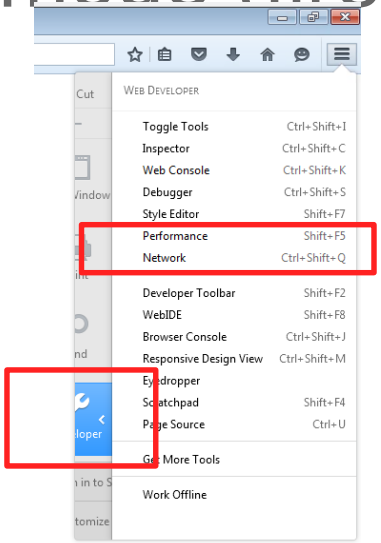


Download Xml from that page
We only see dhcp info

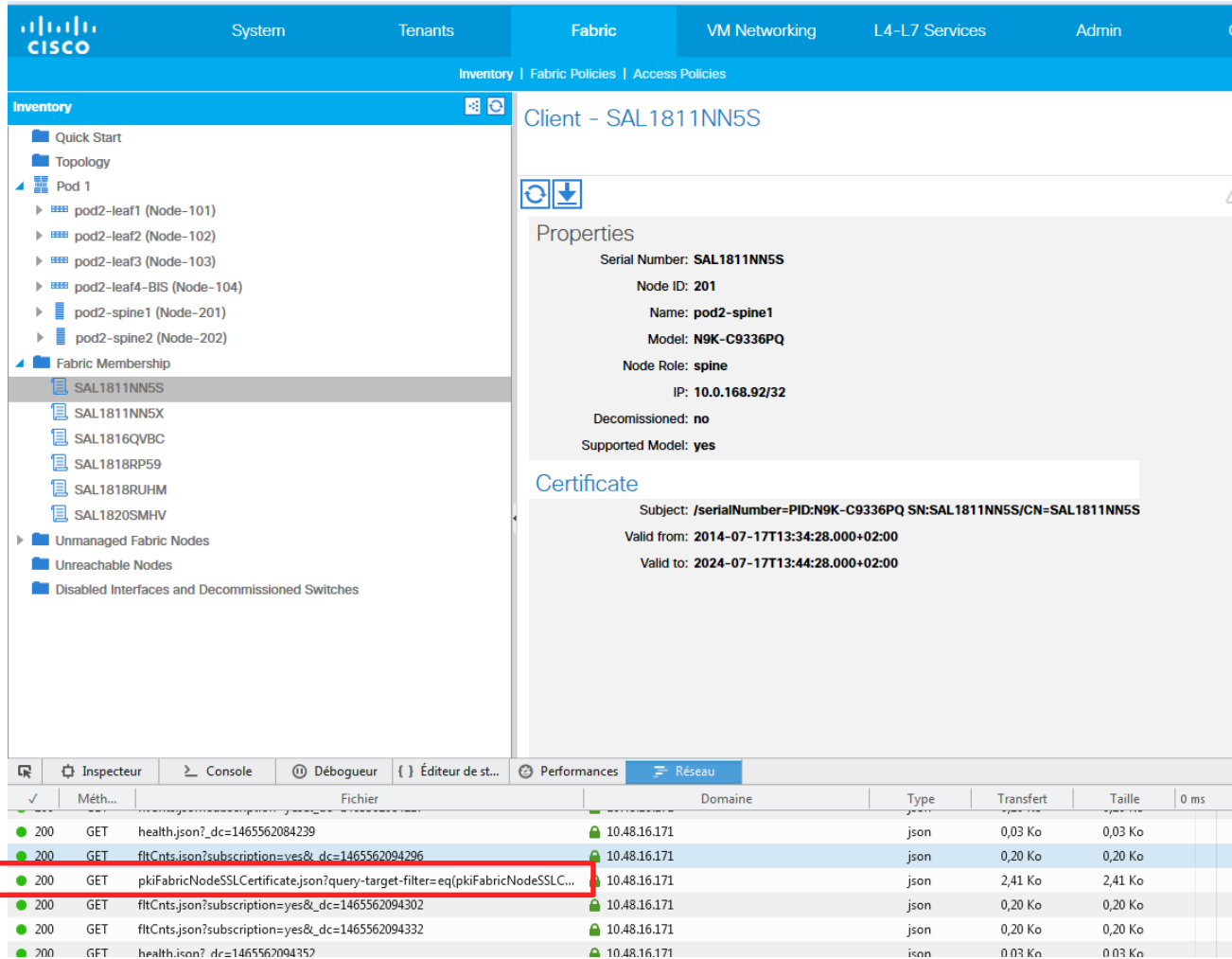
Nothing about cert ???


```
<dhcpClient childAction="" clientEvent="assigned"
decomissioned="no" dn="client-[SAL1811NN5S]" fabricId="1"
fwVer="" hwAddr="00:00:00:00:00:00" id="SAL1811NN5S«
 ip="10.0.168.92/32" lcOwn="local" modTs="2015-10-
08T11:12:27.175+02:00" model="N9K-C9336PQ" name="pod2-
spine1" nodeId="201" nodeRole="spine" podId="1" spineLevel="1"
status="" supported="yes"/>
```

Enable Dev mode (firefox)



We see every query
Needed to display the
Page including REST
GET for class
So we see class name..



Or use API Inspector

https://10.48.16.171/#c:a.1|client-[SAL1811NN5S]

Filters: ☑ trace ☑ debug ☑ info ☑ warn ☑ error ☑ fatal ☑ all

Search: Reset ☐ Regex ☐ Match case ☐ Disable

Options: ☑ Log ☐ Wrap ☐ Newest at the top ☑ Scroll to latest Clear Close


```
timestamp: 14:40:10 DEBUG
timestamp: 14:40:22 DEBUG
method: GET
url: https://10.48.16.171/api/node/mo/client-[SAL1811NN5X]/fltCnts.json?subscription=yes
response: {"totalCount":"1","subscriptionId":"72057598350000161","imdata":[{"faultCounts":{"attributes":{"childAction":"","crit":"0","dn":"client-[SAL1811NN5X]/fltCnts","maj":"0","minor":"0","status":"","war
timestamp: 14:40:22 DEBUG
method: GET
url: https://10.48.16.171/api/node/class/pkiFabricNodeSSLCertificate.json?query-target-filter=eq(pkiFabricNodeSSLCertificate.nodeId,"202")&subscription=yes
response: {"totalCount":"1","subscriptionId":"72057598350000162","imdata":[{"pkiFabricNodeSSLCertificate":{"attributes":{"authorityKeyIdentifier":"keyid:D0:C5:22:26:AB:4F:46:60:EC:AE:05:91:C7:DC:5A:D1:B0:47:
timestamp: 14:40:22 DEBUG
method: GET
url: https://10.48.16.171/api/node/mo/client-[SAL1811NN5X]/fltCnts.json?subscription=yes
response: {"totalCount":"1","subscriptionId":"72057598350000163","imdata":[{"faultCounts":{"attributes":{"childAction":"","crit":"0","dn":"client-[SAL1811NN5X]/fltCnts","maj":"0","minor":"0","status":"","war
timestamp: 14:40:22 DEBUG
method: GET
url: https://10.48.16.171//api/node/mo/client-[SAL1811NN5X]/health.json
response: {"totalCount":"0","imdata":[]}
timestamp: 14:40:22 DEBUG
method: GET
url: https://10.48.16.171/api/node/mo/client-[SAL1811NN5X]/fltCnts.json?subscription=yes
response: {"totalCount":"1","subscriptionId":"72057598350000164","imdata":[{"faultCounts":{"attributes":{"childAction":"","crit":"0","dn":"client-[SAL1811NN5X]/fltCnts","maj":"0","minor":"0","status":"","war
timestamp: 14:40:24 DEBUG
method: GET
url: https://10.48.16.171/api/node/mo/client-[SAL1811NN5S]/fltCnts.json?subscription=yes
response: {"totalCount":"1","subscriptionId":"72057598350000165","imdata":[{"faultCounts":{"attributes":{"childAction":"","crit":"0","dn":"client-[SAL1811NN5S]/fltCnts","maj":"0","minor":"0","status":"","war
timestamp: 14:40:24 DEBUG
method: GET
url: https://10.48.16.171/api/node/class/pkiFabricNodeSSLCertificate.json?query-target-filter=eq(pkiFabricNodeSSLCertificate.nodeId,"201")&subscription=yes
response: {"totalCount":"1","subscriptionId":"72057598350000166","imdata":[{"pkiFabricNodeSSLCertificate":{"attributes":{"authorityKeyIdentifier":"keyid:D0:C5:22:26:AB:4F:46:60:EC:AE:05:91:C7:DC:5A:D1:B0:47:
timestamp: 14:40:24 DEBUG
method: GET
url: https://10.48.16.171/api/node/mo/client-[SAL1811NN5S]/fltCnts.json?subscription=yes
response: {"totalCount":"1","subscriptionId":"72057598350000167","imdata":[{"faultCounts":{"attributes":{"childAction":"","crit":"0","dn":"client-[SAL1811NN5S]/fltCnts","maj":"0","minor":"0","status":"","war
timestamp: 14:40:24 DEBUG
method: GET
url: https://10.48.16.171/api/node/mo/client-[SAL1811NN5S]/fltCnts.json?subscription=yes
response: {"totalCount":"1","subscriptionId":"72057598350000168","imdata":[{"faultCounts":{"attributes":{"childAction":"","crit":"0","dn":"client-[SAL1811NN5S]/fltCnts","maj":"0","minor":"0","status":"","war
timestamp: 14:40:24 DEBUG
method: GET
url: https://10.48.16.171//api/node/mo/client-[SAL1811NN5S]/health.json
response: {"totalCount":"0","imdata":[]}
timestamp: 14:40:26 DEBUG
method: GET
url: https://10.48.16.171/api/node/mo/info.json
response: {"totalCount":"1","imdata":[{"topInfo":{"attributes":{"childAction":"","currentTime":"2016-06-10T14:40:34.558+02:00","dn":"info","id":"1","role":"controller","status":""}}}]}
```

MIT

- MIT is a distributed tree between APIC and each switch nodes.
- Some MO's are on APIC , some are on switches

- DME – Owner of MO (process)
- Shard – Location and storing of MO

Object model on APIC

- Sharding – 32 shard (0-31)
- Every Object have is part of a shard and is replicated to the 3 APIC
  - The Shard leader and 2 shard followers
  - The APIC who's shard leader for the shard for which an MO belong is responsible to write to it If request comes from another APIC
- An MO is managed by one DME, each DME are present on each APIC.

Object model on Switches

- MO on switches can be splitted in multiple chunk.
- Primary chunk is the MO owner
- Some MO have more than one chunk (2nd, 3rd chunk). Each chunk contains some of the property of an MO.
- Each chunk may have different chunk owner.
- Owner is a process (DME or NXOS process)
- Everyone have Read access to all chunk of all MO
- Only the owner have Write access to a Mo/Property

Modifying Object Model on switch

- Whenever a process modify an property of an MO it owns, it will send an MTS notification to a specific MTS group (there are thousands of those).
- Any process can suscribe to notification for the MTS group they are interested in.
- Suscriber will get the notification and will check in the DB what was modified for the MO and eventually will takes actions.

Tasks framework

- DMEs communicate using message entities called “stimulus” which usually have request and response
- Format : **System Type: System ID: Service ID, Slot Number, Shard ID, Replica ID.**
  - Eg:
  - `5047||16-03-02 20:57:52.024+01:00||polUpdate__||DBG4||fr=ifc_policymgr:2:1:6:0:31:1,to=ifc_policyelem:1:101:5:0:255:127,co=doer:0:0:0x1328d6:1,si=0x1061f138f97a98:0x23||(envelope 0x3000000159dfd: RECEIVE-BULK:REQUEST[polUpdate/]) CONTENT :`
  - `From policymgr (6) on Apic (1) node-1 (1) to policyelem (5) on Switch (2)node 101 (101)`
  - https://techzone.cisco.com/t5/Application-Centric/Identities-for-Intra-Fabric-Messaging-between-DME/ta-p/781679

Tasks framework

- Tasks framework is used to deliver stimulus reliably to handle
  - Packet drop in network
  - Packet drop due to flow control
  - Application errors
- Tasks are stored as internal Mos which store minimal context required to recreate the stimulus payload (seen typically only in DME logs)

  eg:
  ```
  <polUpdate dn="pcons/refcont-[registry/class-2150/instdn-[uni/vmmp-Microsoft/dom-SCVMM-Pod2/ctrlr-SCVMM-Pod2]/ra-[uni/vmmp-Microsoft/domD-SCVMM-Pod2/ctrlrD-SCVMM-Pod2]-5-101-1-0-Subtree-mo]/trdn-[uni/vmmp-Microsoft/dom-SCVMM-Pod2/ctrlr-SCVMM-Pod2]"
  action="13" run="47435" stage="12" timestamp="1456948351945">
  <pconsRefTask childAction="deleteNonPresent" data="" descr="" dn="action/policymgrsubj-[pcons/refcont-[registry/class-2150/instdn-[uni/vmmp-Microsoft/dom-SCVMM-Pod2/ctrlr-SCVMM-Pod2]/ra-[uni/vmmp-Microsoft/domD-SCVMM-Pod2/ctrlrD-SCVMM-Pod2]-5-101-1-0-Subtree-mo]/trdn-[uni/vmmp-Microsoft/dom-SCVMM-Pod2/ctrlr-SCVMM-Pod2]]/pconsRefTask-PolUpdate" endTs="never" fail="0" id="PolUpdate"
  invErrCode="none" invErrDescr="" invRslt="" lcOwn="local" modTs="2016-03-02T20:52:31.945+01:00" oDn="pcons/refcont-[registry/class-2150/instdn-[uni/vmmp-Microsoft/dom-SCVMM-Pod2/ctrlr-SCVMM-Pod2]/ra-[uni/vmmp-Microsoft/domD-SCVMM-Pod2/ctrlrD-SCVMM-Pod2]-5-101-1-0-Subtree-mo]/trdn-[uni/vmmp-Microsoft/dom-SCVMM-Pod2/ctrlr-SCVMM-Pod2]" operSt="processing" originMinority="no" runId="47435"
  startTs="2016-03-02T20:52:31.945+01:00" status="created" try="0" ts="2016-03-02T20:52:31.945+01:00"/>
  <inIgnoreNonResolvedPols value="yes"/>
  <inConfigs>
  ```

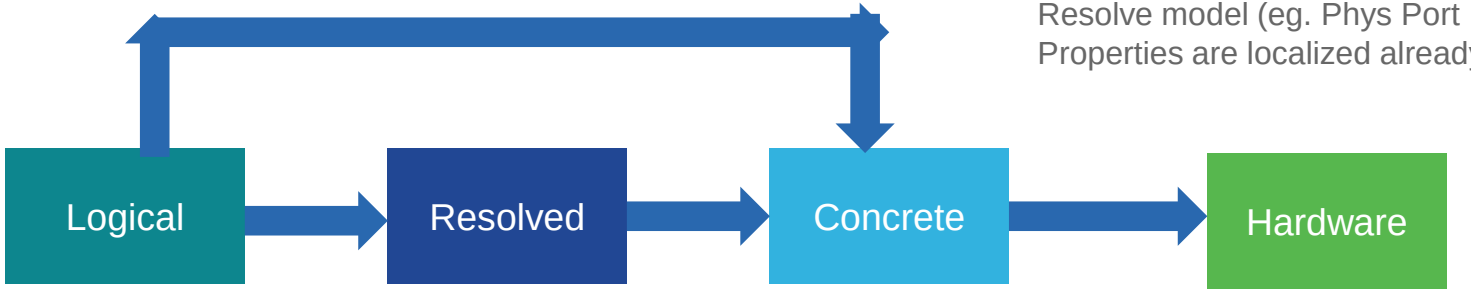
- Periodic timers reissue the task till the success response is received
- Task Mos are persisted and replicated like other MO – so tasks are reliably delivered even across DME / APIC crashes

Object Model – logical – resolved - concrete

Types of Objects

- Logical, resolved, and concrete
  - Logical = configured in the GUI by the user
  - Resolved = created by the APIC as a unit/object to communicate and pass information to the switches
  - Concrete = objects used by the switches to program hardware

Some Object do not need Resolve model (eg. Phys Port Properties are localized already to a leaf/port)



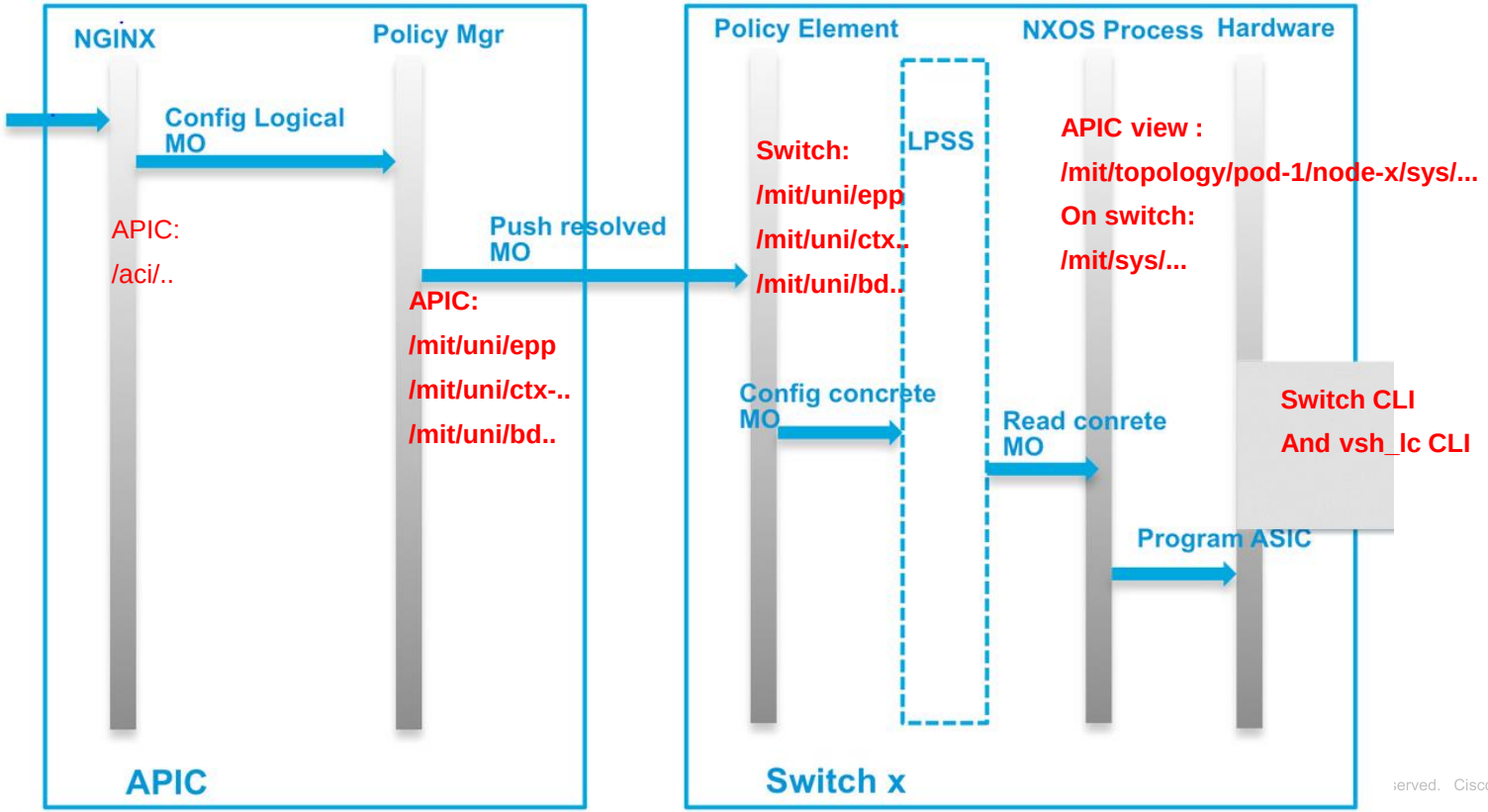
User creates a BD

Later a VM is added on an EPG part of this BD on leaf 101
→ Leaf creates a part of the Mit in resolved model to instantiate BD and EPG on node 101

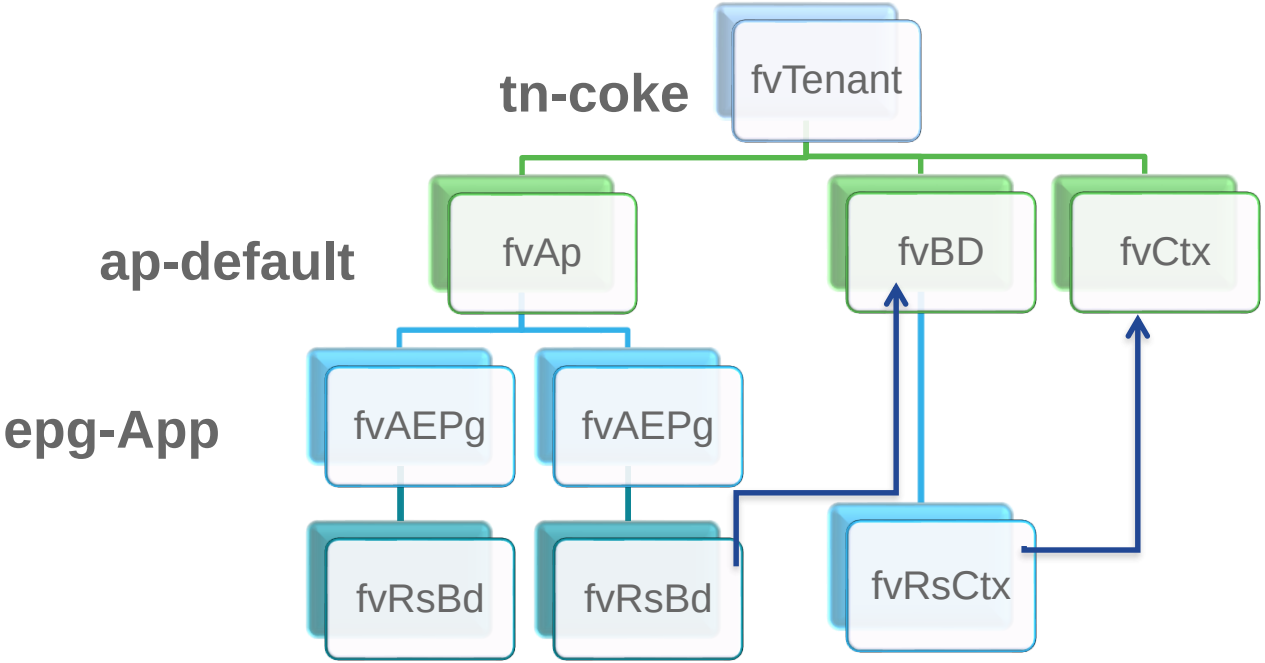
This subset of the MIT is pushed to Leaf 101 by APIC DME (policymgr).
Leaf creates another part of the MIT tree to actually reserve real Hw ressource (internal vlan…)

The concrete oject are consume by switch software to programm ASICs

Logical – Resolved – Concrete model MO localisation in file structure



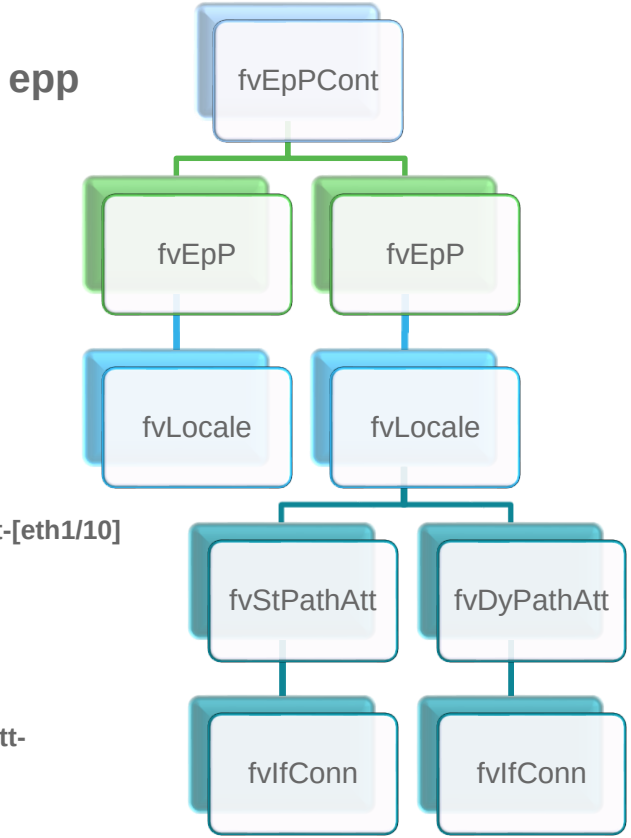
Logical Model Configured By User
EPg Containment Hierarchy



Logical Model Converted to Resolved Model

Endpoint Profile (EpP)

- **Internal deployment profile for Epg**
- **Deployed to node / policyelement**



Every Epg has an fvEpP class

fv-[uni/tn-coke/ap-default/epg-App]

There is an fvLocale for each leaf that needs the Epg

fv-[uni/tn-coke/ap-default/epg-App]/node-102

One fvStpathAtt for each Static path

One fvDyPathAtt for each host of the DVS(AVS) attached to the node in that Locale

fv-[uni/tn-coke/ap-default/epg-App]/node-102/stpathatt-[eth1/10]
Or
fv-[uni/tn-coke/ap-default/epg-App]/node-102/dyatt-[topology/pod-1/paths-102/pathep-[eth1/47]]

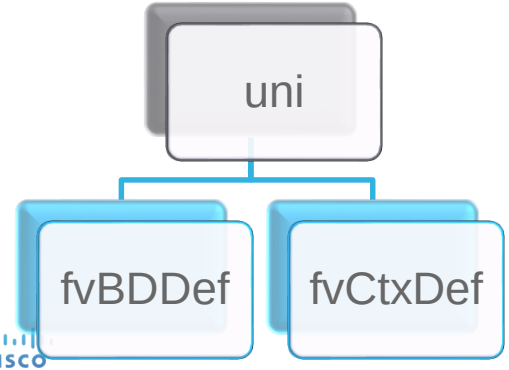
fvIfConn represents access encaps (vlan or vxlan)

fv-[uni/tn-coke/ap-default/epg-App]/node-102/stpathatt-[eth1/10]/conndef/conn-[vlan-19]-[0.0.0.0]
or
fv-[uni/tn-coke/ap-default/epg-App]/node-102/dyatt-[topology/pod-1/paths-102/pathep-[eth1/47]]/conndef/conn-[vxlan-8912897]-[0.0.0.0]

Logical Model Converted to Resolved Model

BD and Context (fvBDDef & fvCtxDef)

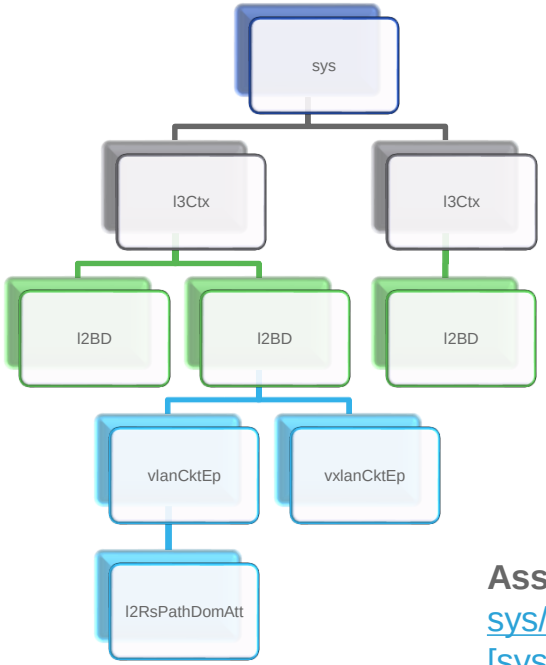
1. BDDef: Summarized internal representation of fvBD. Its naming property is the DN of the corresponding fvBD. It has BD identifiers like Segment ID, PcTag (policy control tag used for Contracts) etc. It also contains summarized internal representations of other BD policies like IGMP, DHCP, End Point Retention Policy etc.
2. CtxDef: Summarized internal representation of fvCtx. Its naming property is the DN of the corresponding fvCtx. Like BDDef it has Ctx identifiers like Segment ID and PcTag. It also contains summarized internal representations of other Ctx policies like End Point Retention Policy etc.



uni/bd-[uni/tn-mgmt/BD-inb]

uni/ctx-[uni/tn-mgmt/ctx-inb]

Resolved Model Converted to Concrete Model



top::System, it's the root of concrete model

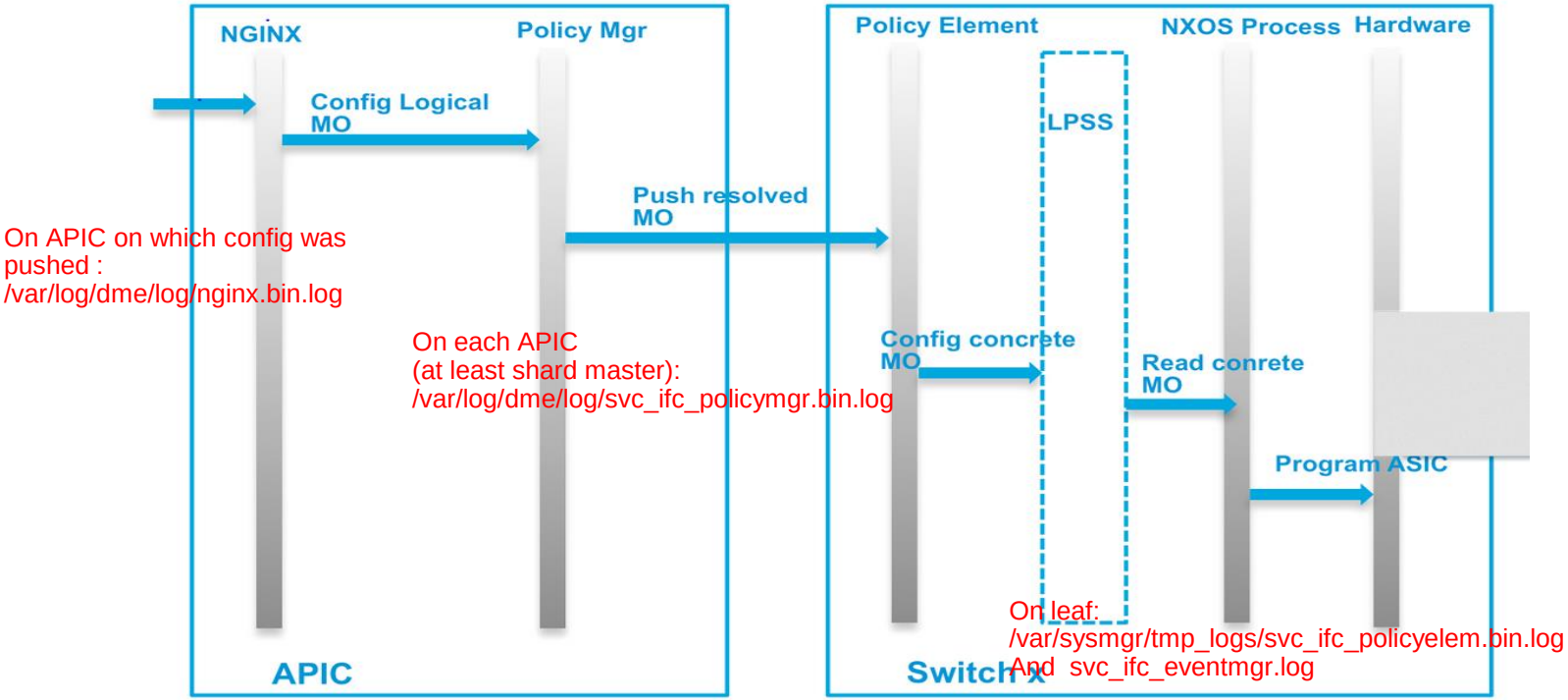
Private network (aka VRF).
sys/ctx-[vxlan-2555904]

Bridge Domain
sys/ctx-[vxlan-2555904]/bd-[vxlan-15826914]

Encap i.e. VLAN OR VXLAN
sys/ctx-[vxlan-2555904]/bd-[vxlan-15826914]/vlan-[vlan-17]

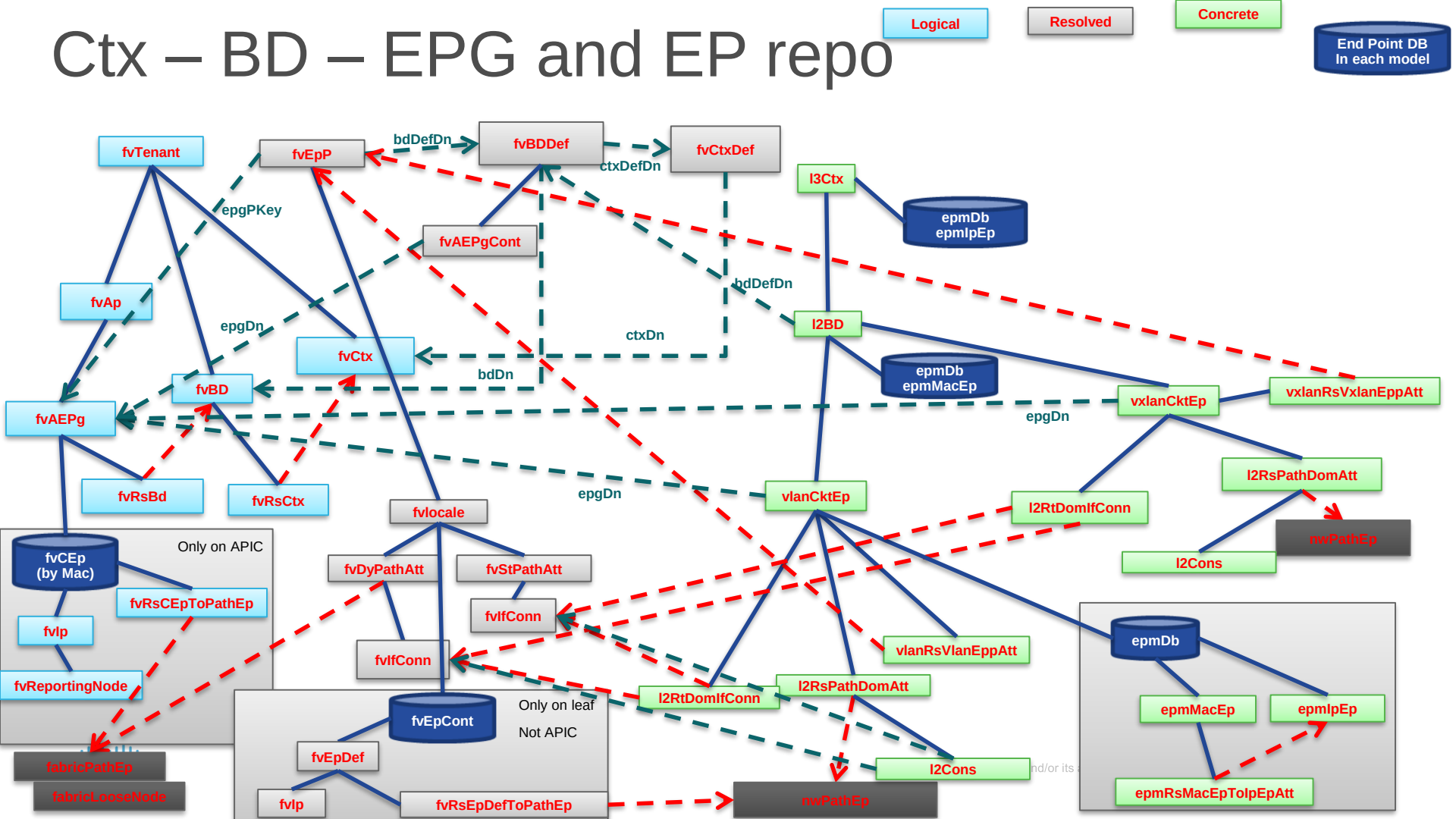
Association from Encap to Port/Port-Channel
sys/ctx-[vxlan-2555904]/bd-[vxlan-15826914]/vlan-[vlan-17]/rspathDomAtt-[sys/conng/path-[eth1/16]]

Which Logs ?



Log: Watch out some logs wraps quickly check oldlog as well and creation time to get the right one

Ctx – BD – EPG and EP repo



MO relationship

- References between MO instances that do not share a containment (parent-child) relationship are expressed through relationship definitions in the object model.
- Relationship definitions allow the framework to generically track object inter-dependencies.
- The object model definition consists of source (from) and to (target) classes, along with cardinality and exclusivity rules.
- The source relationship MO is contained by the FROM object. Its MO class is named:
  - {SOURCE MO PKG}::Rs{RELATION NAME}
  - Ex: compRsNicAdj -> child of compHpNic (Nic of an Hypervisor) with tDn = hvsAdj
- The target relationship MO is contained by the TO object. Its MO class is named:
  - {SOURCE MO PKG}::Rt{RELATION NAME}
  - Ex : hvsRtNicAdj -> child of hvsAdj with tDn = compHpNic
- Relationship can be either:
  - Explicit : tDn (target DN) is fixed in the model for this class
  - Named : tDn is determined by the policy resolution out of few possible target name (aka epg to bd policy resolution for example)

Example - Wants to find out whether and where an encap vlan is in use ?

- List all vlan use anywhere as encap in the fabric :

```
admin@pod2-apic3:~>  moquery -c fvIfConn | egrep "dn.*vlan-" | awk '{print $3}' | sed
's/.*vlan-//g' | sed 's/]-.*//g' | uniq | sort -n | tr '\n' ' '
101 102 102 103 112 120 211 240 850 851 852 853 860 870 1000 1004 1004 1070 1071 1072 1072
1073 1074 1074 1075 1134 1134 1137 1138 1139 1140 3891 3962
```

- **Where do we use vlan-120 ?**

```
admin@pod2-apic1:~> moquery -c fvIfConn | egrep "dn.*vlan-120"
dn           : uni/epp/fv-[uni/tn-DC/ap-App/epg-EPG1]/node-102/stpathatt-[n6k2-vpc]/conndef/conn-[vlan-120]-[0.0.0.0]
dn           : uni/epp/fv-[uni/tn-DC/ap-App/epg-EPG1]/node-101/stpathatt-[n6k2-vpc]/conndef/conn-[vlan-120]-[0.0.0.0]
dn           : uni/epp/fv-[uni/tn-DC/ap-App/epg-EPG1]/node-101/stpathatt-[eth1/33]/conndef/conn-[vlan-120]-[0.0.0.0]
admin@pod2-apic1:~>
```

Used by EPG : EPG1 in tn-DC on node 101 to 102,
This is stpathatt(Static Path) going for example on a VPC on 101-102 called n7k2-vpc and also on 1/33 on node 101

Where do we use vlan-850 ?

```
admin@pod2-apic3:~>  moquery -c fvIfConn | egrep "dn.*vlan-850"
dn           : uni/epp/rtd-[uni/tn-OSPF/out-OSPF-Ext/instP-ospf-net]/node-101/stpathatt-[eth1/33]/conndef/conn-[vlan-850]-[10.253.255.22/29]
admin@pod2-apic3:~>
```

Used by a L3 out called ospf-EXT in tn-OSPF with node101 port 1/33

Using API

Visore

Filter

Class or DN: fvEpP

Property: Op: == Val1: Val2:

Run Query

Display URI of last query

Display last response

fvEpP ?

bdDefDn	uni/bd-[uni/tn-Coke/BD-CokeBD]-isSvc-no
bdDefStQual	none
childAction	
ctxDefDn	
ctxDefStQual	none
ctxSeg	0
deplSt	deployable
descr	
dn	uni/epp/fv-[uni/tn-Coke/ap-InsiemePortal/epg-WEB]
enfPref	hw
epgDn	
epgName	WEB
epgPKey	uni/tn-Coke/ap-InsiemePortal/epg-WEB
l2FDSeg	0
l3CtxEncap	unknown
lcOwn	resolveOnBehalf
modTs	2014-05-30T22:20:24.934+00:00
monPolDn	uni/tn-common/monepg-default
name	
npName	InsiemePortal
operSt	allocated
ownerKey	
ownerTag	
pcTag	44034
prio	unspecified
scopeId	2
status	
tName	Coke

DN of the BDDef used by this EpP

DN of the CtxDef used by this EpP

Here Resolved model object

For an Epg (fvEpP)

Use arrow to browse the parent/child hierarchy

DN of the Epg corresponding to this EpP

Other EPg parameters

Click on link to jump to different Part of the MIT:

- To the Resolved BD
- To the Logical Epg

Moquery

Moquery Pros/cons

- Pros :
  - easy to use,
  - good to catch few object,
  - purely cli based, no need of anything else,
  - output very easy to read
  - One line per parameter, easy to grep
- Cons:
  - no browsing capability (parent-child) like visore
  - No as fast/efficient as direct api call

Moquery help

```
apic1# moquery -h
usage: Command line cousin to visore [-h] [-i HOST] [-p PORT] [-d DN]
                                     [-c KLASS] [-f FILTER] [-a ATTRS]
                                     [-o OUTPUT] [-u USER]
                                     [-x [OPTIONS [OPTIONS ...]]]

optional arguments:
  -h, --help            show this help message and exit
  -i HOST, --host HOST  Hostname or ip of apic
  -p PORT, --port PORT  REST server port
  -d DN, --dn DN        dn of the mo
  -c KLASS, --klass KLASS
                        comma seperated class names to query
  -f FILTER, --filter FILTER
                        property filter to accept/reject mos
  -a ATTRS, --attrs ATTRS
                        type of attributes to display (config, all)
  -o OUTPUT, --output OUTPUT
                        Display format (block, table, xml, json)
  -u USER, --user USER  User name
  -x [OPTIONS [OPTIONS ...]], --options [OPTIONS [OPTIONS ...]]
                        Extra options to the query
```

Moquery usage

Moquery -c <class-name>

Class query , usually bundle with a grep on dn and selected property

Moquery -d <dn>

Direct object query to see all its property

Moquery -f <filter>

With class query to get a filter of a class.

Moquery -o [json|xml]

The output is in json or xml instead of plain ASCII

Moquery with filter

You can use as filter code Any property of an object of that class .. For faultInst that could be: Severity, lc (lifecycle), code, …

```
• admin@pod2-apic1:~> moquery -c faultInst -f 'fault.Inst.code == "F0467"' | egrep dn
• dn                  : topology/pod-1/node-101/local/svc-policyelem-id-0/uni/epp/rtd-
  [uni/tn-L3/out-BGP-Out/instP-epg-l3-bgp]/node-101/stpathatt-[eth1/8]/nwissues/fault-
  F0467
• dn                  : topology/pod-1/node-102/local/svc-policyelem-id-0/uni/epp/fv-
  [uni/tn-DC/ap-App/epg-EPG3]/node-102/stpathatt-[eth1/33]/nwissues/fault-F0467
• dn                  : topology/pod-1/node-104/local/svc-policyelem-id-0/uni/epp/fv-
  [uni/tn-infra/ap-access/epg-default]/node-104/attEntitypathatt-[VMM]/rsstPathAtt-
  [sys/conng/path-[eth1/46]]/nwissues/fault-F0467
• admin@pod2-apic1:~>
```

Using moquery to dump/sort active faults (faultInst)

```
admin@apic1:~> moquery -c faultInst | egrep -e "^descr" | sort | uniq -c
```

quickly sorts all active faults

```
      2 descr     : Configuration failed for EPG default due to Not Associated With Management Zone
      3 descr     : Datetime Policy Configuration for F5clock  failed due to : access-epg-not-specified
      1 descr     : Failed to form relation to MO AbsGraph-VEStandAloneFuncProfile of class vnsAbsGraph
      1 descr     : Failed to form relation to MO fwP-default of class nwsFwPol in context uni/infra
      1 descr     : Ntp configuration on leaf leaf1 is Not Synchronized
      1 descr     : Ntp configuration on leaf leaf2 is Not Synchronized
      1 descr     : Ntp configuration on spine spine1 is Not Synchronized
      1 descr     : Power supply shutdown. (serial number DCB18CLUS15)
```

Now we could query all faults by criteria – such as description (fault.Inst.descr)

```
moquery –c faultInst –f fault.Inst.descr==“: Failed to form relation to MO AbsGraph-VEStandAloneFuncProfile …”
```

Curl/icurl

Curl / icurl

- icurl is just a wrapper for regular linux curl using the token authen of your ssh session.
- icurl directly on apic can use direct api access on port 7777
- You can use curl from any linux system (after authentication)
- Allows more easily some advance filtering…

Curl (from any linux/bash shell)

1. **Authenticate to apic to get token (may need to update curl if ssl can't authenticate)**

```
[root@Centos-RD1 ~]# curl -1  -X POST https://x.x.x.x/api/aaaLogin.xml  -d  '<aaaUser
name="admin" pwd="passwd" />' -k --dump-header cookie
<?xml version="1.0" encoding="UTF-8"?><imdata totalCount="1">
<aaaLogin
token="SFDtJKYN3KFnfzDnGkCivhNUesQysi+OkBZn/mrZENSQOwyu9srSIY0ZLGZneI/Y3EnsmYiO0vxkUURZD8QMs/bybMD4PM95so0nop4tNvBabxMDTL35iOZ7fGhykHO
8SmOejhfLiTDTCOtlcKK97dwKh2Of7UibDouGxEMGGAs=" siteFingerprint="/GdgoFMAQPRCiENu" refreshTimeoutSeconds="300"
maximumLifetimeSeconds="86400" guiIdleTimeoutSeconds="1200" restTimeoutSeconds="90" creationTime="1456524250"
firstLoginTime="1456524250" userName="admin" remoteUser="false" unixUserId="15374" sessionId="kIAhvTq4Tp6ryVUMAGcYKw==" lastName=""
firstName="" version="1.2(2g)" buildTime="Sun Feb 21 02:49:35 PST 2016" node="topology/pod-1/node-1">
<aaaUserDomain name="all" rolesR="admin" rolesW="admin">
<aaaReadRoles/>
<aaaWriteRoles>
<role name="admin"/>
</aaaWriteRoles>
</aaaUserDomain>
<DnDomainMapEntry dn="uni/tn-common" readPrivileges="admin" writePrivileges="admin"/>
<DnDomainMapEntry dn="uni/tn-infra" readPrivileges="admin" writePrivileges="admin"/>
<DnDomainMapEntry dn="uni/tn-mgmt" readPrivileges="admin" writePrivileges="admin"/>
```

2. GET or POST what is needed

- `curl -1 -b cookie  -k -X GET https://x.x.x.x/api/class/fvTenant.xml`
- `curl -1 -b cookie  -k -X GET https://x.x.x.x/api/mo/uni/tn-DC.xml`
- `curl -1 -b cookie  -k -X POST https://x.x.x.x/api/mo/uni.xml -d '<fvTenant name="DC2"/>'`
- `curl -1 -b cookie  -k -X POST https://x.x.x.x/api/mo/uni.xml -d '<fvTenant name="DC2" status ="deleted"/>'`

Icurl other example

Fetching fault record with fault F0467 after Sept 2016

- **icurl 'http://localhost:7777/api/class/faultRecord.xml?query-target-filter=and(and(gt(faultRecord.created,"2016-09-01"))and(eq(faultRecord.code,"F0467")))'**

All fault ordered by time between 8:30 and 8:40 on dec 4 :

- icurl 'http://localhost:7777/api/class/faultRecord.xml?query-target-filter=and(and(gt(faultRecord.created,"2015-12-04T08:30:00"))and(lt(faultRecord.created,"2015-12-04T08:40:00")))&order-by=faultRecord.created|desc' | xmllint --format -

Tech Support next gen ☺

Step 1 - From APIC – create a new directory in /tmp or /home/admin
And get the following in that directory

```
Apic# mkdir TAC
Apic# cd TAC


icurl 'http://localhost:7777/api/class/faultInfo.xml'  > faultInfo.xml
icurl 'http://localhost:7777/api/class/faultRecord.xml?query-target-filter=and(gt(faultRecord.created,"2016-02-01"))' >
faultRecord.xml
icurl 'http://localhost:7777/api/class/eventRecord.xml?query-target-filter=and(gt(eventRecord.created,"2016-02-01"))'  >
eventRecord.xml
icurl 'http://localhost:7777/api/class/firmwareARunning.xml'  > firmwareARunning.xml
icurl 'http://localhost:7777/api/class/aaaModLR.xmll?query-target-filter=and(gt(aaaModLR.created,"2016-02-15"'  > aaaModLR.xml
icurl 'http://localhost:7777/api/class/aaaSessionLR.xml'  > aaaSessionLR.xml
icurl 'http://localhost:7777/api/class/fabricNode.xml'  > fabricNode.xml
icurl 'http://localhost:7777/api/mo/.xml?query-target=subtree' > mo-subtree.xml
```

EventRecord, faultRecord and aaaModLR May be huge
It is crucial to specify time to start.
Change the date in the above to be the day you need

Make sure each file got created and contains real xml (not Just an error)

Copy the full directory and attach it to the case !!

The output of those will give you raw unsorted xml
Good to feed a script , bad for grep or manual check

grep friendly output

```
icurl 'http://localhost:7777/api/class/faultRecord.xml?query-target-filter=and(gt(faultRecord.created,"2016-02-15"))&order-by=faultRecord.created|desc' | xmllint --format - > faultRecord.xml

icurl 'http://localhost:7777/api/class/eventRecord.xml?query-target-filter=and(gt(eventRecord.created,"2016-02-15"))&order-by=eventRecord.created|desc' | xmllint --format - > faultRecord.xml

icurl 'http://localhost:7777/api/class/faultInfo.xml' | xmllint --format -  > faultInfo.xml

icurl 'http://localhost:7777/api/class/aaaModLR.xml?query-target-filter=and(gt(aaaModLR.created,"2016-02-15"))&order-by=aaaModLR.created|desc' | xmllint --format - > aaaModLR.xml

icurl 'http://localhost:7777/api/class/aaaSessionLR.xml?query-target-filter=and(gt(aaaSessionLR.created,"2016-02-25"))&order-by=aaaSessionLR.created|desc' | xmllint --format – > aaaSessionLR.xml


icurl 'http://localhost:7777/api/class/firmwareARunning.xml' | xmllint --format - > firmwareARunning.xml

icurl 'http://localhost:7777/api/class/fabricNode.xml' | xmllint --format - > fabricNode.xml

icurl 'http://localhost:7777/api/mo/.xml?query-target=subtree' | xmllint --format - > mo-subtree.xml

NODE :

icurl 'http://localhost:7777/api/mo/topology/pod-1/node-xxx/sys.xml?query-target=subtree' | xmllint --format - > node-xxx.xml
```

Xmllint –format - : makes a line break between each record
Order-by=… : gives you the event/fault time sorted with most recent first

Dataset is too big

- If you get from a query « dataset is toobig » you need to restrict the amount of MO fetched at a time, by either reducing time window..
- Or use some more tricks to still get all data …

Dataset is too big

If too much data, get only the last 100k record of each.

- `icurl 'http://localhost:7777/api/class/aaaModLR.xml?order-by=aaaModLR.created|desc&page-size=100000' > /data/techsupport/aaaModLR.xml`
- `icurl 'http://localhost:7777/api/class/faultRecord.xml?order-by=faultRecord.created|desc&page-size=100000’ > /data/techsupport/faultRecord.xml`
- `icurl 'http://localhost:7777/api/class/eventRecord.xml?order-by=eventRecord.created|desc&page-size=100000' > /data/techsupport/eventRecord.xml`

Getting all 500k record with pages

```
icurl 'http://localhost:7777/api/class/faultRecord.xml?order-by=faultRecord.created|desc&page-size=100000&page=0' >
/data/techsupport/faultRecord-0.xml

icurl 'http://localhost:7777/api/class/faultRecord.xml?order-by=faultRecord.created|desc&page-size=100000&page=1' >
/data/techsupport/faultRecord-1.xml

icurl 'http://localhost:7777/api/class/faultRecord.xml?order-by=faultRecord.created|desc&page-size=100000&page=2' >
/data/techsupport/faultRecord-2.xml

icurl 'http://localhost:7777/api/class/faultRecord.xml?order-by=faultRecord.created|desc&page-size=100000&page=3' >
/data/techsupport/faultRecord-3.xml

icurl 'http://localhost:7777/api/class/faultRecord.xml?order-by=faultRecord.created|desc&page-size=100000&page=4' >
/data/techsupport/faultRecord-4.xml
```

API other method

- Read only – visore browser
- Full REST client user friendly : POSTMAN extension in chrome

Q & A
Thank You