

UCS and ACI PHP Better Together (B2G) Tool Instructions

Computing System Platform Group
Cisco Systems, Inc.
Initial Document: Dan Hanson and David Nguyen
Document Version 1.0(1f)

Change Log

Revision	Date	Originator	Comments
1	11/24/2014	David Nguyen	Initial Draft.
2	12/15/2014	David Nguyen	Added Dan's inputs
3	1/15/2014	David Nguyen	Added OVA file information
4	3/16/2015	Dan Hanson	Multiple Modifications
5	7/7/2015	Dan Hanson	Final Edits for Inclusion in 1.0.1d ova
6	8/28/2015	Dan Hanson	Many Changes to offer deeper explanation of operations and new capabilities tracking – for 1.0.1f version

Table of Contents

Change Log.....	1
Overview.....	4
Audience.....	4
System Requirements.....	5
What is new in the version.....	5
1.1.1f.....	5
Program Architecture.....	6
Starting Point: Your first logon to the new OVA.....	7
Description of the setup fields and prompts.....	7
Next Area: B2G.init file.....	8
Using the APIC and UCSM simulators.....	9
Using the Connectivity testing program.....	9
How we can debug the running program.....	12
Logging infrastructure details.....	13
Pertinent Filesystem Organization Details.....	13
Directory Structure:.....	14
User Stories.....	15
Common Objects Independent of User Story:.....	15
Object Persistence.....	16
The concept of “BLACKLISTING” objects.....	16
The re-starting of B2G and effects on controllers.....	16
User Story 0 (and included in 10): vPC to UCS Domain.....	16
Starting Condition.....	16
Starting Stimulus.....	17
Progress on the CLI.....	17
Object Persistence.....	18
User Story 1 (and included in 10): VMM hosts on UCS Domain – VLAN.....	18
Starting Condition.....	18
Starting Stimulus.....	20
Progress on the CLI and End State.....	20
Object Persistence.....	20
User Story 2 (and included in 10): VMM hosts on UCS Domain – VXLAN.....	20

Starting Condition	20
Starting Stimulus.....	21
Progress on the CLI and End State	21
Object Persistence.....	21
User Story 3 (and included in 10): Bare Metal (UCS Blade) Installation.....	21
Starting Condition.....	21
Starting Stimulus.....	22
Progress on the CLI and End State	23
Object Persistence.....	23
User Story 4 (and included in 10): SPAN.....	23
Starting Condition.....	23
Starting Stimulus.....	24
Progress on the CLI and End State	24
Object Persistence.....	24
User Story 5 (NOT included in 10): Rack Installation	24
Starting Condition.....	24
Starting Stimulus.....	25
Progress on the CLI and End State	25
Object Persistence.....	26
Usage Tidbits:	26
CIMC Server Limits:	26
UCSM Setup:.....	26
APIC Setup:	26
Program Usage:.....	26
APIC	26
UCSM	26
CIMC	26
Additional Notes	27
Fore more information	27

Overview

This is a multithreaded php program that provides an over the top tool which enhances the overall user experience between UCS computing platforms and ACI fabrics. The program bridges the 2 data management engines (DME) between the UCS and ACI to help streamline some of the common operational use cases. When a user completes a configuration task on one system, through event notifications the tool will execute a set of API calls to the peered system to complete the configuration task at hand.

The tool is currently limited to a single UCS managed system and it also works with multiple C-series servers in standalone (CIMC) mode. The tool is a work in process, and can be further enhanced to meet ones needs.

The current use cases that this tool covers are:

- 1) Auto formation of a vPC between the UCS domain and the peer ACI leaf nodes
- 2) VMM (Vmware) domain encompassing ESX hosts within a UCSM domain – VLAN or VXLAN backed with vShield
- 3) UCSM blade server bare metal installation
- 4) Auto matching of the UCS SPAN destination, to the ACI leaf SPAN source instance
- 5) Auto formation of linkages from standalone CIMC managed UCS rack servers – connected to ACI leafs

NOTE: This is in its early stages and is being actively worked. However it does provide a good tool to show case the power of the programmability for UCS and ACI.

Audience

This documentation is intended for a technical audience who has a solid technical understanding and working knowledge with Unified Computing Systems B and C series platform, and Application Centric Infrastructure. It is not a must but if the customer is familiar with DME and MIT, then it will further help with overall understanding of the program and interaction between the systems (UCS, ACI, and CIMC).

The level of PHP coding will vary depending on the need of what the user wants to do – if it just wants to execute, then no coding is needed. However, if the user wants to modify or add on to it, then PHP coding knowledge will be needed.

There will be further documentation put out talking to the methods and techniques within the tools, to help those wishing to contribute code back upstream to further enhance this gain a deeper understanding.

System Requirements

B2G is released as an OVA file running CentOS 7, with the latest patches of OpenSSL and PHP v5.4 has been provided. However, the PHP code can be installed on any linux distribution with PHP version 5.4 and later, and requires openSSL. The linux system can either be a physical or virtual device. Ensure PHP, pThreads, openSSL, samba, ntp client, and libmcrypt library packages are installed.

New packages can be added for your preference using the yum, wget, or other standard tools, which are on this OVA.

What is new in the version

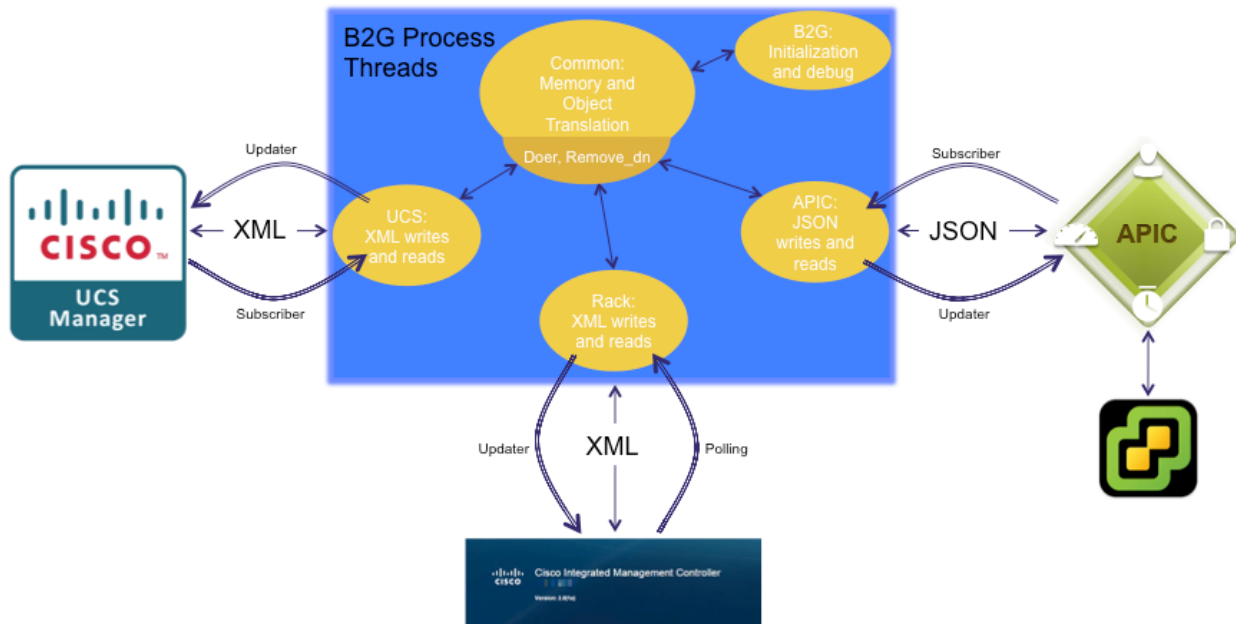
1.1.1f

This version is the first version that incorporates early field tester's feedback. We have strengthened many things directly based on the efforts of the distributed Cisco team and put some thank you's in the appendix. Some details on new items in this release include:

- We have added an out of the box (new .ova deployment) setup script to capture networking information on unconfigured systems
- Added a communications test utility to validate all communication workflows that the actual program will need to operate
- Improved feedback on ongoing activity while the program is running
- Now supports many ports in a block of VPC from the UCS to the ACI leaf nodes
- Now works in all cases identically between a simulated environment and a real hardware lab
- Now talks to TLS encryption, as earlier methods of SSL and TLS1.0 were dropped as defaults in APIC v1.1 and later
- Improved the documentation around the usage, debugability, etc.
- Added some sample screen outputs to the troubleshooting section
- Many bug fixes throughout

Program Architecture

As mentioned in the overview section, it is a multithreaded process with different modules. The PHP code is made up with the following modules – UCS, Rack, ACI, and common process and memory.



The UCS and ACI processes are each made up of 2 key sub-components – a subscriber where the module will subscribe to event channels on the controller, and updater where the data to be written to the respective controller will be written. The rack process will just have an updater and a polling function as the CIMC has no event subscription capacity. The modules normalize the different API languages (XML or JSON) between UCS, CIMC, and APIC. If an event is signaled on one of the subscriptions, the module will trigger the doer sub-component within the common logic component to handle the interactions needed. The doer performs the following functions -- create, delete, and updates the respective systems.

The memory module (common class) includes an inventory of information such as physical linkages, VLANs, EPGs, etc. This information is consistently updated and relies on the individual sub-components to update it. Based on the information contained in the b2g.init file, it creates a table, which maps the relationships between UCS and ACI. The memory only stores relevant information and blacklists any other information that it is not relevant to keep the memory footprint small. There is another sub-component called the soaker, which is a simple wait time for certain functions – sort of like a delay timer.

Starting Point: Your first logon to the new OVA

The appliance should be started, and please use the “open console” option in VMware. There is no reason this should not run in HyperV or KVM or even VirtualBox. Please login as “root” with the default password of “ucsandaci”. If this is the first time running, you will be sent into the setup script to configure the IP, NTP, hostname, etc. information. The script will restart the networking processes after you save the information. If there is an error or some mistyping, you can go into the /opt/aciucs/B2G_VM_Config directory and enter “./setup” to restart it. When completed, a file called .configured-ip will be placed in this directory. The subsequent logon’s will not run the setup script again if this file exists. The IP you will be using is stored in this file, and is used by the main process (for future iterations where we setup a clustered pair, etc.).

Description of the setup fields and prompts

You will be presented with the image above, and the default entry is contained in the brackets. This will be invoked on the first time login as root of the B2G appliance, and can be re-run as shown below or any

```
[root@localhost B2G_VM_Config]# ./setup

*****B2G Virtual Appliance Re-write Basic Setup*****

Enter IP address method [dhcp|static] (static): static
Enter static IP (x.x.x.x): 172.25.177.129
Enter the subnet mask (255.255.255.0): 255.255.255.0
Enter the default gateway (x.x.x.x): 172.25.177.1
Enter the IP of the primary DNS server (none): 171.70.168.183
Enter the IP of the secondary DNS server (none):
Secondary DNS server empty
Enter the IP of the NTP server (none): 171.68.38.65
Enter the B2G virtual appliance hostname (aciucs-1-0-1f): aciucs-1-0-1f
Setting the hostname to: aciucs-1-0-1f
-----
You entered the data below:

IP configuration method: static
IP Address: 172.25.177.129
Netmask Bits: 255.255.255.0
Default Gateway: 172.25.177.1
Primary DNS Server: 171.70.168.183
Secondary DNS Server: none
NTP Server: 171.68.38.65
B2G Virtual Appliance Hostname: aciucs-1-0-1f

Note: The system is set to UTC timezone (and B2G itself just uses UTC). Instead of presenting a list of ~700 possibilities here,
just copy the file from /usr/share/zoneinfo (under one of the region directories) to overwrite the /etc/localtime file.

-----
Enter 1 to commit this, 2 to start over, or just enter to exit
```

time after.

Next Area: B2G.init file

This file is the common starting point to edit the connectivity and user access information, along with program usage options. The table below outlines the variables and the usage.

Variable	Meaning
userstory	Which story to run. Unless this is a group of C servers, we anticipate this to almost always be case 10
environmentType	Real physical system, or alternatively on a simulator (to simulate the cabling between the systems)
simFIA-LeafPortSeed	The base ports in a simulated environment on the leaf that our UCS VPC will attach to
simFIB-LeafPortSeed	Same as above, but for the B fabric interconnect
simLeafSpanPort	The single leaf SPAN source port that when configured in a simulated environment when the UCSM is configured with a SPAN port
simVPCkey	The VPC id created on the UCS that the tool will filter to do the simulated cabling. We have this defined as the tools might have other VPC's
simSPANkey	The SPAN port created on the UCS that the tool will filter to do the simulated cabling
UCSgroupAEPname	The Attachable Access Entity Profile that will be created on the APIC for the links to the UCS domain that the tool manages
UCSvSwitchPolicyname	The vSwitching policy that is needed for support on a UCSM environment
manageDNL2	A true or false flag to signal if we want the tool to manage our UCS uplinks into different Layer 2 domains
manageAPICVTEP	A true or false flag to signal if we want the tool to create the needed policy on the APIC to allow VTEP connectivity to the VTEPs on a vShield VXLAN environment
manageVMUUFlooding	A true or false flag (not implemented yet) that will set the flooding within the given bridge domain for the tenants when those EPG's are mapped to the UCS domain
physDomainStartupDelay	An amount of seconds to wait for doing any assigning of our bindings from the domain information – to allow for a complete reception of APIC existing binding information before the program attempts to assign bindings
apicip	IP of the APIC
apicuser	User ID of the APIC (this should have administrative privileges)
apicpwd	Password of the APIC. This is in clear text today – but we ask for feedback on preferred methods to hide this.
ucsmip	IP of the UCSM
ucsmuser	User ID of the UCSM (this should have administrative privileges)
ucsmpwd	Password of the UCSM
ucsmVXLANtransportVLAN	Since the UCS 6100 and 6200 FI's do not natively support VXLAN, we wrap those in a VLAN encapsulation within the UCS domain and “push and pop” the VLAN ID on the links to the ACI leaf, and on the vNIC facing the hypervisor utilizing VXLAN

ucsmVLANpoolmin	This is the minimum VLAN number to create a pool for UCS Bare Metal server bindings (more detail on how used in the user story 3 below
ucsmVLANpoolmax	This is the maximum VLAN number to use in UCS bare metal binding
cimcVLANpoolmin	This is the minimum VLAN number to create a pool for the UCS rack servers running a non-virtualized workload – where we automatically select a VLAN and created needed bindings on the right ports
cimcVLANpoolmax	This is the maximum VLAN number to use in the UCS rack server bare medal binding
cserverip	IP of one rack servers. We support multiple C servers with this tool
cserveruser	User ID of the rack server instance
cserverpwd	Password for the rack server instance
logGeneraloperations	Log the general internal operations in logMessages.txt
logBareMetaloperations	Log the bare metal UCS B case information in bmMessages.txt
logVMMoperations	Log the VM hypervisor operations to this UCS in vmmMessages.txt
logVPCoperations	Log the VPC formation and event operations in vpcMessages.txt
logRackoperations	Log the CIMC information in rackMessage.txt
logUCSMevents	Log the incoming UCS structured elements in UCSMeventdata.txt (can get large)
logDoerCalls	Log the calls to the doer process in doerMessages.txt
logSPANoperations	Log the SPAN operations in spanMessages.txt

Using the APIC and UCSM simulators

While the API methods are identical for the simulators to the physical environment for both controllers, there is no real connectivity, and this tool can be signaled to emulate the cables per the .init file variable settings in the table above.

Using the Connectivity testing program

A very good first step to validate all the communications needs exist, is the running of this connectivity test program. This program also reads the B2G.init file and tests connections to all of the items defined in those files. This runs a series of actions that test every method that the B2G tool makes use of. This section will outline the tests, and what you should see. These tests can take up to 2 minutes to complete (as things time out when the tests block), so please let this run to completion.

First, the tool attempts an HTTPS connection to the UCS. We do this with TLSv1.1 and web sockets. If this is successful, then the program subscribes to events again using the TLSv1.1 security mechanism, and I will include 1 sample event received. This event monitoring closes after 20 seconds – the numbers in the [], and we then logout.

```

2015-08-27 00:14:34 -> ****TESTING UCSM LOGIN VIA HTTPS TO: (172.25.177.226)****
2015-08-27 00:14:34 -> UCSM HTTPS (172.25.177.226) LOGIN REPLY SUCCESS START -->>>
string(282) " <aaaLogin cookie="" response="yes" outCookie="1440634474/6469b947-8acc-4e86-b117-c85f6b201b67" out
="2.2(4b)" outName="admin"> </aaaLogin>"
2015-08-27 00:14:34 -> UCSM HTTPS (172.25.177.226) LOGIN REPLY SUCCESS END -->>>
2015-08-27 00:14:34 -> ****TESTING UCSM SSL EVENT SUBSCRIPTION FUNCTIONALITY FOR 20 SECONDS ON: (172.25.177.226)
2015-08-27 00:14:34 -> UCSM EVENT RECEIVE STARTING -->>>
[1]HTTP/1.1 200 OK
Date: Thu, 27 Aug 2015 00:14:34 GMT
Server: Apache/2.4.12 (Unix) OpenSSL/FIPS
Content-Type: text/html

[7][8]935
<methodVessel cookie=""> <inStimuli> <configMoChangeEvent cookie="" inEid="506743704"> <inConfig> <aaaSession
refreshPeriod="600" sessionTimeout="7200" status="created" switchId="A" term="web_43830" ui="web" user="admin"/
gin-admin-web_43830_A" id="web_43830_A" intId="137493865" localhost="172.25.177.224" name="admin" policyLevel="0
hangeEvent cookie="" inEid="506743706"> <inConfig> <aaaLog dn="aaa-log" size="9992" status="modified"/> </inConf
<methodVessel cookie=""> <inStimuli> <configMoChangeEvent cookie="" inEid="506743707"> <inConfig> <aaaWebLogin
<methodVessel cookie=""> <inStimuli> <configMoChangeEvent cookie="" inEid="506743708"> <inConfig> <biosBOT dn=
thodVessel>267
<methodVessel cookie=""> <inStimuli> <configMoChangeEvent cookie="" inEid="506743709"> <inConfig> <biosBOT dn=
thodVessel>669
<methodVessel cookie=""> <inStimuli> <configMoChangeEvent cookie="" inEid="506743710"> <inConfig> <biosBOT dn=
vent cookie="" inEid="506743711"> <inConfig> <biosBOT dn="sys/chassis-2/blade-1/bios/bdgp" lastUpdate="2015-08-
n="sys/chassis-1/blade-6/bios/bdgp" lastUpdate="2015-08-26T17:14:37.390" status="modified"/> </inConfig> </conf
<methodVessel cookie=""> <inStimuli> <configMoChangeEvent cookie="" inEid="506743713"> <inConfig> <biosBOT dn=
thodVessel>870
<methodVessel cookie=""> <inStimuli> <configMoChangeEvent cookie="" inEid="506743714"> <inConfig> <biosBOT dn=
vent cookie="" inEid="506743715"> <inConfig> <biosBOT dn="sys/chassis-1/blade-8/bios/bdgp" lastUpdate="2015-08-
n="sys/chassis-2/blade-3/bios/bdgp" lastUpdate="2015-08-26T17:14:37.383" status="modified"/> </inConfig> </conf
8-26T17:14:37.398" status="modified"/> </inConfig> </configMoChangeEvent> </inStimuli> </methodVessel>267
<methodVessel cookie=""> <inStimuli> <configMoChangeEvent cookie="" inEid="506743718"> <inConfig> <biosBOT dn=
thodVessel>[18][19][20]267
2015-08-27 00:14:40 -> UCSM EVENT RECEIVE ENDING -->>>
2015-08-27 00:14:40 -> UCSM HTTPS (172.25.177.226) LOGOUT REPLY SUCCESS START -->>>
string(70) " <aaaLogout cookie="" response="yes" outStatus="success"> </aaaLogout>"
2015-08-27 00:14:40 -> UCSM HTTPS (172.25.177.226) LOGOUT REPLY SUCCESS END -->>>

```

Next, we do this same test (just the HTTPS parts now) to the CIMC rack servers if we have them in the init file, and we have a userstory of 5. There is no even subscription today on these, so we poll in 15 second intervals. After that, we do this same test (just the HTTPS parts now) to the APIC. This is an event where we log in, do the event subscription, but here the events are more granular so we create a test VLAN pool, then immediately delete it to see the event on the channel. Then we logout.

```

2015-08-27 00:14:40 -> ****TESTING APIC HTTPS PUT FUNCTIONALITY ON: (172.25.180.32)****
2015-08-27 00:14:40 -> APIC LOGIN REPLY SUCCESS START -->>>
string(1558) [{"imdata":{"aaaLogin":{"attributes":{"token":"t9waQ1yuR8gb85mLgOWZWIH60ZWkMPuByNHPK0M0rwdSbY6Qnh1N/Es018x/tpGKI03F55qVtmt2LRp","refreshTimeoutSeconds":"600","maximumLifetimeSeconds":"86400","guiIdleTimeoutSeconds":"1200","restTimeoutSeconds":1000,"Ppr2Gwz053i1wLaCH6aRA=","lastName":"","firstName":"","version":"1.0(4h)","buildTime":"Tue May 05 22:06:47 PDT 2015","children":[{"aaaReadRoles":{"attributes":{}},"aaaWriteRoles":{"attributes":{}},"children":[{"role":{"attributes":{"name":"adminMapEntry":{"attributes":{"dn":"uni/tn-common","readPrivileges":"admin","writePrivileges":{"admin"}}},"DnDomainMapEntry":{"dn":"uni/tn-infra","readPrivileges":"admin","writePrivileges":{"admin"}}},"DnDomainMapEntry":{"attributes":{"dn":"uni/tn-danhanston","writePrivileges":"admin"}}}}}}}}]}]}]}
2015-08-27 00:14:40 -> APIC cookieHeader is: [Cookie: APIC-cookie=t9waQ1yuR8gb85mLgOWZWIH60ZWkMPuByNHPK0M0rwdSbY6Qnh1N/Es018x/tpGKI03F55qVtmt2LRp]
2015-08-27 00:14:40 -> APIC LOGIN REPLY SUCCESS END -->>>

2015-08-27 00:14:40 -> ****TESTING APIC HTTPS GET FUNCTIONALITY ON: (172.25.180.32)****
2015-08-27 00:14:40 -> APIC INTERFACE GET REPLY START -->>>
string(1030) [{"totalCount":"2","imdata":{"l3EncRtdIf":{"attributes":{"adminSt":"up","bw":"0","childAction":"","delay":"1","status":"00:00:00:00:000","ethpMcfgState":"0","id":"po1.4093","ifConnDn":"","lcoWn":"local","linkLogEn":"default","modTs":"2014-11-00:06:F6:30:2E:E5","status":"","},"l3EncRtdIf":{"attributes":{"adminSt":"up","bw":"0","childAction":"","delay":"1","descr":"00:00:000","ethpMcfgState":"0","id":"po1.4093","ifConnDn":"","lcoWn":"local","linkLogEn":"default","modTs":"2014-11-02T01:03:E0","status":""}}}}]}]}
2015-08-27 00:14:40 -> APIC INTERFACE GET REPLY END -->>>

****TESTING APIC SSL EVENT SUBSCRIPTION FUNCTIONALITY FOR 20 SECONDS ON: (172.25.180.32)****
sending on socket: GET /sockett9waQ1yuR8gb85mLgOWZWIH60ZWkMPuByNHPK0M0rwdSbY6Qnh1N/Es018x/tpGKOIj9MyFAZWgRpMM3drdNZ9J5Ro274iY
Upgrade: websocket
Connection: upgrade
Host: 172.25.180.32
Origin: https://172.25.180.32
Pragma: no-cache
Cache-Control: no-cache
Connection: keep-alive
Sec-WebSocket-Key: 6666777799998888
Sec-WebSocket-Version: 13
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits, x-webkit-deflate-frame
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36
Cookie: APIC-cookie=t9waQ1yuR8gb85mLgOWZWIH60ZWkMPuByNHPK0M0rwdSbY6Qnh1N/Es018x/tpGKOIj9MyFAZWgRpMM3drdNZ9J5Ro274iWSx+48dKY4

APIC EVENT RECEPTION START -->>>
[1]HTTP/1.1 101 Switching Protocols
Server: nginx/1.4.0
Date: Wed, 26 Aug 2015 16:27:02 GMT
Connection: upgrade
APIC INTERFACES SUBSCRIBE REPLY START -->>>
string(2082) [{"totalCount":"6","subscriptionId":"72058590487248897","imdata":{"fvnsVlanInstP":{"attributes":{"allocMode":"dynamic","monPolDn":"uni/fabric/monfab-default","name":"Dan-Vcenter55-Pool1","ownerKey":"","ownerTag":"","dn":"uni/infra/vlanns-[Dan-VXLAN-OuterPool]-dynamic","lcoWn":"local","modTs":"2015-05-15T09:09:31.272-07:00","monPolDn":"","attributes":{"allocMode":"dynamic","childAction":"","configIssues":"","descr":"","dn":"uni/infra/vlanns-[dn-vlan-pool]-dynamic","ownerTag":"","status":"","uid":"13284"}}},"fvnsVlanInstP":{"attributes":{"allocMode":"dynamic","childAction":"","configIssues":"","ownerTag":"","name":"vmm-pool","ownerKey":"","ownerTag":"","status":"","uid":"15374"}}},"fvnsVlanInstP":{"attributes":{"allocMode":"dynamic","childAction":"","configIssues":"","descr":"Auto Created by the B2G process","dn":"uni/infra/vlanns-[UCS_RACK_DOMAINS]-dynamic","ownerTag":"","status":"","uid":"15374"}}}}]}]}
APIC INTERFACES SUBSCRIBE REPLY END -->>>

Sec-WebSocket-Accept: gRUH5G1/hR+gXJaMPgW05e2Aas=
Upgrade: websocket

[9][10]A-d{"subscriptionId":["72058590487248897"],"imdata":{"fvnsVlanInstP":{"attributes":{"allocMode":"dynamic","childAction":"","monPolDn":"uni/fabric/monfab-default","name":"CONNECTTESTPOOL","ownerKey":"","ownerTag":"","rn":"","status":"created"},"fvnsVlanInstP":{"attributes":{"allocMode":"dynamic","childAction":"","configIssues":"","descr":"","dn":"uni/infra/vlanns-[CONNECTTESTPOOL]-dynamic","modTs":"2015-08-26T09:27:06.734-07:00","rn":"","status":"deleted"}}}}}[12][13][14]
2015-08-27 00:14:55 -> Testing Complete.

```

This concludes the test. Errors need to be corrected before the B2G can properly execute.

How we can debug the running program

A very useful feature to see what is in memory, monitor update events, events on the subscription channels, etc. is provided in the program. This only runs when we are running interactively with the runB2G process. To get the menu, we open a second terminal session to the B2G VM, and go to the same directory where we started the runB2G. You will see a file called "debug.out", and if you issue a "tail -f debug.out" this will give live output from the program. In the session you are running the B2G program in, just hitting a return will display the debug menu in the other window with the tail -f session running. An example is below.

```

===== <<<<<<<<<<<< MENU 2015-08-27 00:29:18 >>>>>>>>>>>>>>>>>>=====
? This help menu
0. storageindex_class
1. storage
2. APICeventids - APIC subscriptions
3. ucsstack
4. apicstack
5. apiextensions
6. nodelist
7. flowmap
8. tmpflowmap
9. apicallqueue
a. ucscallqueue
b. bastards
c. chill-pill
d. Stats
e. UCSMeventids - UCSM subscriptions
j. junkfilter - what types of UCSM objects to catch
k. rackservers/rackstack
m. Start/Stop Monitor all subscriptions and EVENTS
q. Start/Stop Monitor subscriptions UCS EVENTS
w. Start/Stop Monitor UCS UPDATES
n. Start/Stop Monitor subscriptions ACI EVENTS
r. Start/Stop Monitor ACI UPDATES
t. Start/Stop Monitor polling CIMC EVENTS
y. Start/Stop Monitor CIMC UPDATES
v. Show the Monitoring States
x. Logout of UCS and APIC and exit
  
```

Here is a brief table of the items in the debug menu.

storageindex_class	An index of internal memory classes learned
storage	Full internal memory of the synthetic objects and data about them
APICeventids	List of APIC objects we are currently subscribed to
ucsstack	Memory object for the UCS domain
apicstack	Memory object for the APIC controller
apiextensions	Temporal list of extensions we add – normally will be empty
nodelist	List of ACI leaf nodes we have discovered
flowmap	List if the flowmaps that define the objects, and parent child relationship rules
tmpflowmap	Temporal list of the flowmaps being processed – normally empty

apicallqueue	Queue where common and APIC threads share data and should be empty
ucscallqueue	Queue where common and UCS threads share data and should be empty
bastards	Discovered entities on the APIC we care about – but have no parent objects
chill-pill	Our soak timer object events in a hold state
Stats	Statistics on the objects we are looking at in UCS (U) and APIC (A)
UCSMeventids	List of the UCS objects we are tuned into
junkfilter	Filter of things we want. UCSM event subscription is very chatty, and we want to have easy filter method to optimize resources. This filter is by class type and evaluated early on in reception process
rackstack	Memory object for rackservers, and the domains formed by a grouping of rack servers.
Various start/stop monitor	Flags to set or clear to throw simple indications of APIC events or update ([A] or [a]), UCS events or updates ([U] or [u]), and CIMC poll events or updates ([R] or [r]) on the console where B2G is running interactively – useful to just see that we have data in and out to respective devices
Show monitoring state	Shows a list of the monitors above that are on or off
Exit logout	End the program

Logging infrastructure details

Depending on the flags set in the init file, we generate these logging text files for review. Some of these are below, and are related to individual threads. Some like logMessages.txt are more general.

```
[root@localhost APIC-UCS-B2G-v1.0.1f]# ls -l *.txt
-rw-r--r--. 1 root root 341198 Aug 26 17:14 bmMessages.txt
-rw-r--r--. 1 root root 17651 Aug 26 17:14 doerMessages.txt
-rw-r--r--. 1 root root 130400 Aug 26 15:01 logMessages.txt
-rw-r--r--. 1 root root 8800 Aug 26 15:01 spanMessages.txt
-rw-r--r--. 1 root root 381717 Aug 26 15:01 vmmMessages.txt
-rw-r--r--. 1 root root 41973 Aug 26 15:01 vpcMessages.txt
```

Pertinent Filesystem Organization Details

Download the OVA file and install on ESXi v5.5 or greater. The OVA file is a CentOS v7 machine; the login credentials are root/ucsandaci. This document, and a README-FIRST are in the /opt/aciucs directory. The actual PHP code is located in /opt/aciucs/CIMC/APIC-UCS-B2G-v1.0.1f/ for the C Series standalone code, and also the /opt/aciucs/UCSM/APIC-UCS-B2G-v1.0.1f/ for the UCSM code:

- b2g.init (userstory, pools, control elements are all set here)
- b2g.php (Where we read init, and set the flowmaps defining interactions)
- class.common.php (Common memory and translation functions)
- class.ucs.php (UCSM domain updating and event subscriptions)

- class.rack.php (CIMC updating and polling of data)
- class.apic.php (APIC updating and event subscriptions)
- runb2g (shell script) (A script to kick this all off in a contained command line)
- runb2g-bg (A script to kick off the program as a daemon)
- runConnectionTest (A script to kick off the program to test and validate proper connectivity)
- runningProcessID (An output of the process ID in case you wish to kill or restart)
- validateConnectivity.php (the implementation of checks that mirror program operation)
- debug.out (output messages when run in foreground)
- nohup.out (output messages when run as background process)
- cookie (a storage for the last UCSM cookie we are using)
- various “.txt” files (logs from the various processes)

The systems information (UCS/CIMC/ACI) is located in the b2g.init file. Edit the systems IP and login credentials. Also the script works over a TLS 1.1 encrypted session.

NOTE: If no connection is established, disable or check the firewall rules on the OS. This is a useful outcome of running the connection tests.

Directory Structure:

/root

.bashrc – this login file will run the setup script if this has not been done yet

opt/aciucs

B2G_VM_Config/

Bashrc – a sample file to copy to your /root home directory if needed

Setup – a setup script

B2G-version.txt – the version information of the program

Common-ifcfg-ens32 – the common portion of your network config files

Common-ntp.conf – the common portion of you ntp config files

.configured-ip – present with your IP if you used the setup script

ifcfg-ens32 – the network config script that was constructed and copied

ntp-conf - the ntp config file that was constructed and copied to /etc/

README-FIRST.1.0.1f – Includes the quick review, and how to set IP/mask/etc.

Smb.conf – the basic samba setup files

UCSM/

APIC-UCS-B2G-v1.0.1f/

B2G.init – edit this to add the IP and credentials for APIC and UCSM

Files above – and the ./runB2G is the main item to start it up

CIMC/

APIC-UCS-B2G-v1.0.1f/

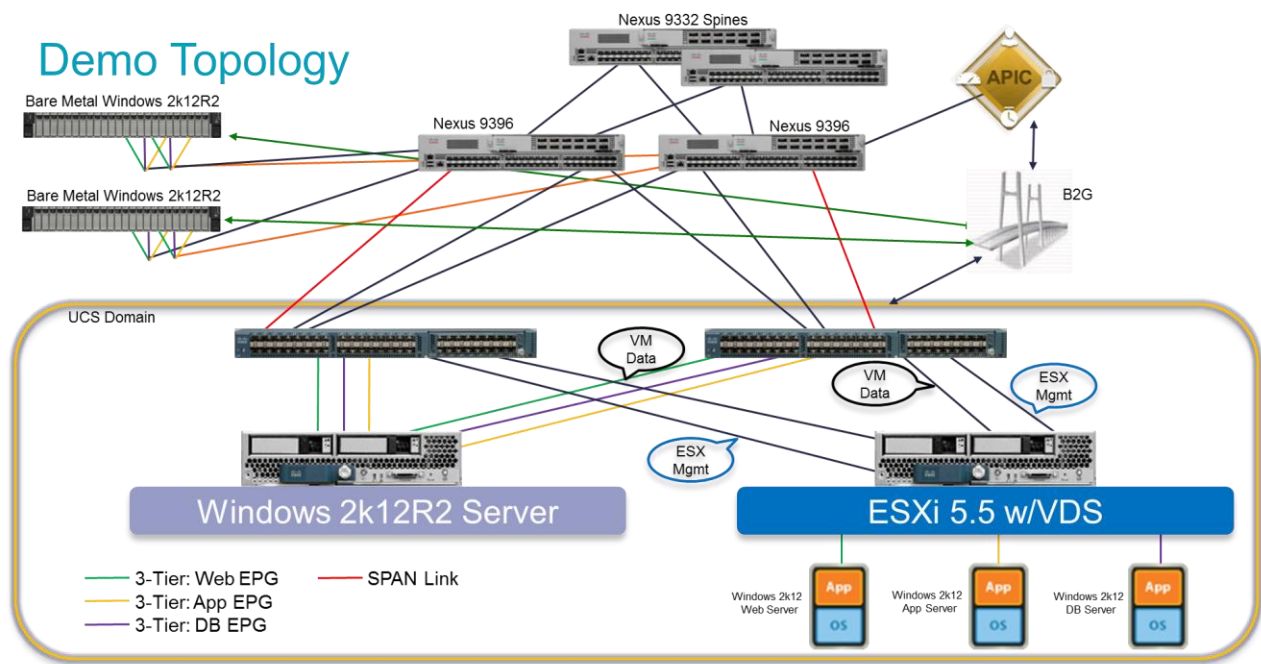
B2G.init – edit this to add the IP and credentials for APIC and each CIMC server

Files above – and the ./runB2G is the main item to start it up

This tool also has samba installed, so that you can directly get into it via OS-X with the finder connect to: option, or from within windows with a [\\IP-address](#) method also. Please note there is an error in OS-X 10.9 where directly editing files on a samba share will often corrupt them. Using a windows text editor will not be an issue.

User Stories

The script currently has 5 use cases but can easily be expanded to encompass additional tasks. This section will provide a high-level overview of each use case. The drawing below, represents the overall ideal demo topology for this tool, and will be used in more detail in later user stories as we focus on the appropriate area.



Common Objects Independent of User Story:

The tool sets up a few baseline items regardless of user story. One common item that is needed from an internal plumbing perspective is the event subscription channels, and keeping those sockets open. We don't just want to trust that our channels are up and functioning always, we actually run a timer on received events and show the receptions in the CLI window of the B2G program. If we go around 5 minutes with no reception data on either channel, we form some test data, and remove it to test the channel. On the APIC we create a KEEPALIVEPOOL which is a VLAN pool, then remove it half a second later. On the UCS we create a B2Gtest VLAN instance, and also remove that. If these fail, the tool will keep trying – but the user would have to restart the program (assuming the devices or simulators themselves are not down).

Object Persistence

The table of items we maintain synchronization are detailed below. If the admin removes these items, the tool will re-create them.

Parent Object	Object Kept in Sync
B2G Startup (No specific Object)	ACIleafVMQ (UCS VMQ Policy)
B2G Startup (No specific Object)	ACI-LLDP (UCS LLDP Policy on VIFs)
B2G Startup (No specific Object)	for-VXLAN-multicast (UCS Multicast Policy)
B2G Startup (No specific Object)	ACIleafHV (UCS Virtualization Server QoS Policy)
B2G Startup (No specific Object)	Physical Domain with UCS system name in APIC
B2G Startup (No specific Object)	VLAN Pool to assign to this physical domain when doing bare metal servers
B2G Startup (No specific Object)	vNIC template with name of all known VMM domains created at startup

The reason for the vNIC template in a VMM domain is one that needs to be specifically called out. We do this to allow the establishment of UCS service profiles in one of 2 methods below for the establishment of virtualization hosts. If we did not have a template to start from, then we would have a chicken and the egg problem where we could not build an ESX server, then join it to a vCenter, which then allows the tool to build the templates needed. Instead the tool builds a vNIC template in UCSM that we can then pull into a service profile and image our hosts. This adapter template on the server is then fully managed with the required needs for a virtualization host, based on information learned from APIC and the vCenter(s) it is linked into.

The concept of “BLACKLISTING” objects

The program does not assume there is no configuration on either the APIC or UCSM when it gets connected. When we see objects on either controller that exist in our subscriptions or filters that match the classes we are interested in, but that we have not created for our use and cannot re-use, then we blacklist those objects. This allows us to quickly move on, and not process incoming event data relating to objects we don't think we manage.

The re-starting of B2G and effects on controllers

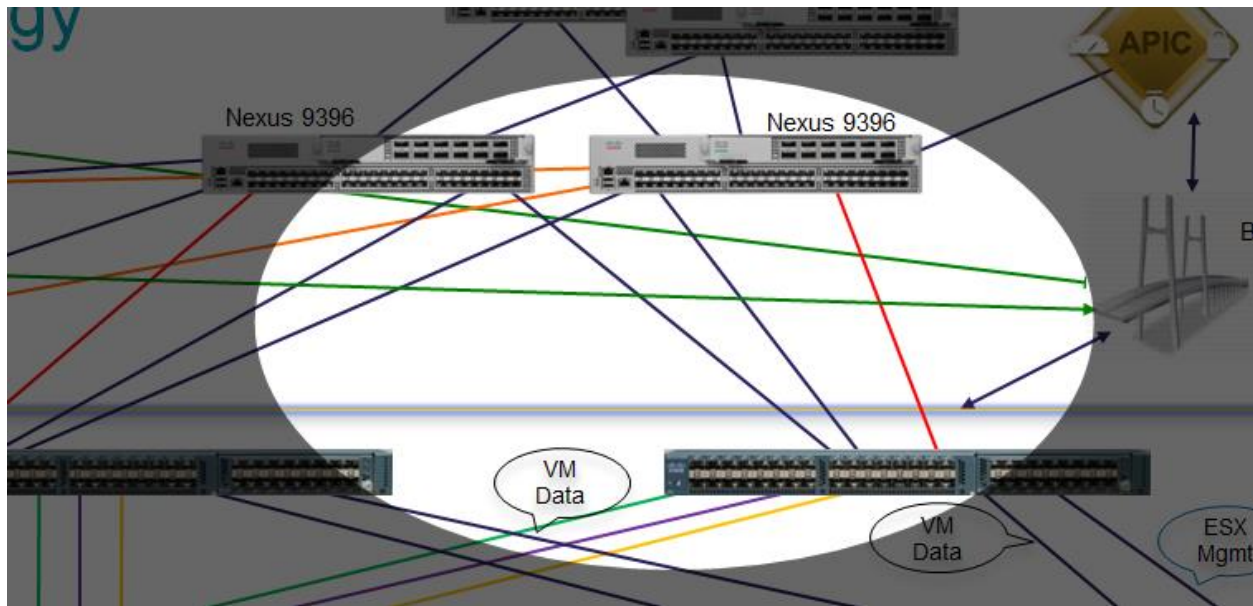
The program is built with the goal of being able to start and stop, without any negative effect on the systems. We also only maintain the in-memory data we learn from the systems, and the startup has been implemented to re-capture all the needed data and continue running where it left off.

User Story 0 (and included in 10): vPC to UCS Domain

Starting Condition

Once the user has physically connected the UCS domain to the ACI leafs, then when the UCSM user creates network uplink ports and forms a VPC on the ports to the ACI leaf the script will create the vPC configuration on the ACI system. All required information is learned via LLDP adjacency information contained within the APIC and UCSM. Currently the configuration task will only be done on the ACI. The vPC configuration stimulus must be initiated from the UCS; the future goal of the script is it is capable of doing it from either endpoint.

Now, we will focus in on the VPC area of the key drawing as shown below.



The fundamental rules that we assume for this current version of the B2G tool are as follows:

- We have a minimum of 1 link from a given FI to 2 individual ACI leafs cabled
- We can have up to 8 members (max tested) with 4 to each FI on a pair
- Each FI has the same port numbers on the ACI leaf connected to it
 - Example, if ports 20-24 on leaf-101 connect to UCS FI-A, then leaf-102 should also use ports 20-24
 - For the UCS FI-B in this example, then ports 25-28 on each leaf should be used
- These links are the black links in the drawing – the red one is for the SPAN case later

The interface connection of the ACI Leaf needs to be mirrored between the vPC Leaf pair. For example, if port 1/31 on the ACI Leaf pair A, then the connection to the other ACI Leaf needs to be the same – 1/31.

Starting Stimulus

When the user creates this port-channel on the UCS (from already configured and enabled uplink ports from the UCS into the ACI leaf), the process will configure the appropriate policies on the APIC. The policy name will be user friendly and is based on the UCS domain system name so the policies can be differentiated.

Progress on the CLI

This is an assumed first step, as the later integrations want to make use of this information to add functionality – over a vPC instead of just discrete ports. This version only works on VPC links to the ACI leafs. You will see the objects being created in memory (VPC_ROOT, and VPC_CHILD).

Object Persistence

The table of items we maintain synchronization are detailed below. When the administrator removes the parent object, the tool removes the individual child objects. If someone accidentally removes a child object, we re-create it. We do this cleanup automatically.

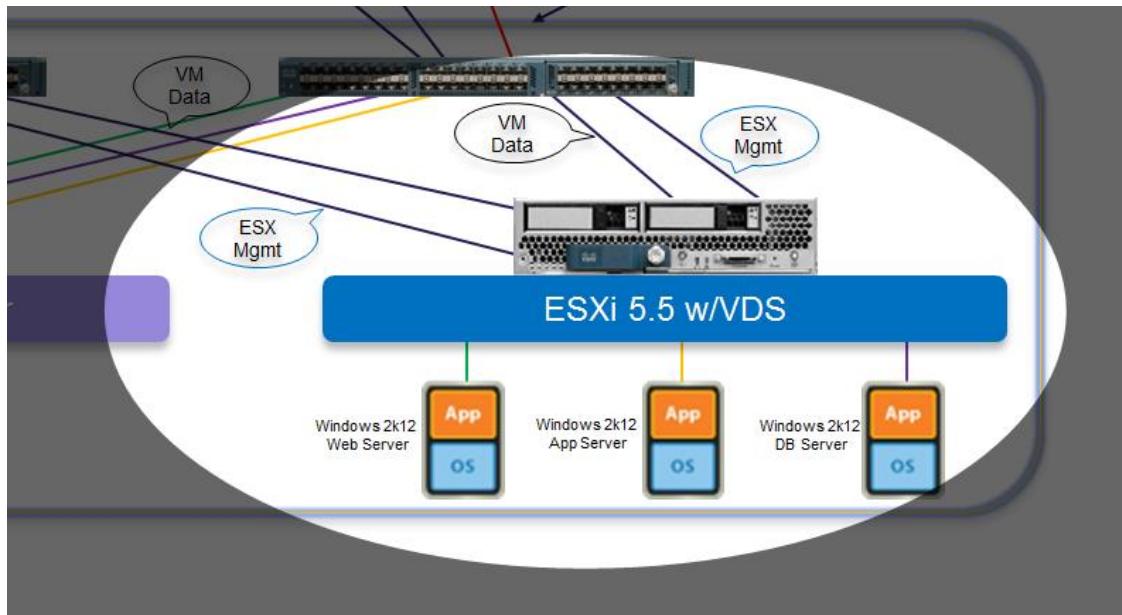
Parent Object	Object Kept in Sync
UCS Port Channel Configuration	UCS_Fabric_Interconnect_Links LACP Policy
UCS Port Channel Configuration	UCS-vSwitch-MAC-Pin Policy
UCS Port Channel Configuration	Interface Policy Group for each FI VPC
UCS Port Channel Configuration	Interface Policy Group AEP
UCS Port Channel Configuration	Interface Policy Group Connectivity Filter
UCS Port Channel Configuration	Interface Profile Port Blocks
UCS Port Channel Configuration	Switch Profile per UCS FI VPC
UCS Port Channel Configuration	Switch Explicit Port Group

User Story 1 (and included in 10): VMM hosts on UCS Domain – VLAN

Starting Condition

The current implementation only works with ESXi and assumes the basic vmm configuration to APIC has been done, such as the provider and vcenter credentials has been created. The tool initially will create and updating adapter template on the UCSM with no backing networks, such that ESXi hosts can be created and joined. This tool assumes there will be 2 adapters for management of the ESXi host that are not managed by this tool, and 2 additional interfaces that will be kept in sync with EPG to VLAN mappings as needed. The tool will listen for vmm domain to be associated to an EPG(s). If an event does occur, then the tool will determine what the VLAN in the backend of the ACI that is being used by ACI. The tool will configure the UCS and create that VLAN and named with Tenant-AP-EPG and VLAN mapping based on host membership (with an extension of –V1 or –V2... to indicate the VMM domains). Through the GUID that vCenter learns for each physical host in the VMM domain, the tool will be able to determine if the peer UCS domain needs this mapping and creates it.

Case 1 assumes the existence of the VPC configuration in the last story already for doing the mappings of needed networks to the right ports towards the ACI leafs. The area of the overall drawing is here.



The fundamental rules that we assume for this current version of the B2G tool are as follows:

- On program startup, it will create empty adapter templates with the vCenter domain name with 1 per UCS fabric where the VM data will pass
- As EPGs are mapped to the VMM domain, multiple things are done on the UCS:
 - The backing VLAN is created on the UCS domain with a truncation of the tenant-appprofile-epg-Vx (where x is the number of VMM domain)
 - The backing VLAN is dynamically added to this template if this UCS domain has hosts on that VMM domain (else they are not added)
 - If allowed in the B2G.init file, we update the UCS disjoint L2 uplinks to put this VLAN on the right uplinks
- B2G does not talk to vCenter on a separate API, rather we learn everything via the “normalized” API from the APIC
- We have 2 options for the management connectivity to the ESX hosts today
 - Create a 3rd and 4th vNIC in the UCS service profile for the management access (and optionally vMotion, etc.)
 - After the updating template is created, we can add in another network for the management and tag it as default – the tool will not manage that network on the adapter however
 - With the default tool, a 4 adapter minimal configuration for an ESX host is ideal (2 for management, 2 that are dynamically managed for VM networks)
- The administrator can then install the ESXi software on the host, and the networks will be intelligently added as needed (the UCS will not need to reboot the server as we can add these to an updating vNIC template)

- This provides advantages in the P,V utilization on UCS domains to help us scale. We move away from a model of sending all VLANs everywhere “in case they are needed” to sending the VLANs to the host as that VMM domain utilizes them
- These are sent when the EPG’s are mapped into VMM domains – but only if we have a host in that VMM domain – we find this out from the UUID key between the devices
- Remember to look at the BD utilized for this EPG to VMM mapping – you may need to enable L2 flooding in this situation

Starting Stimulus

The APIC administrator maps an EPG to a given VMM domain. The B2G tool takes action to update the system with the backing VLAN that APIC selects (or the user defines on APIC) if we have a host on that VMM domain.

Progress on the CLI and End State

You will see the network formations, the adapter updates, and finally, the networks on the UCS. You will see those networks under the vNIC templates if you have hosts on that domain. If using the simulator, we just assume that there is a host on this VMM domain and act in that fashion. You will see the objects being created in memory (VMM_ROOT, VMM_CHILD, and many VMM_* objects for the added policy objects).

Object Persistence

The table of items we maintain synchronization are detailed below. When the administrator removes the parent object, the tool removes the individual child objects. We do this cleanup automatically.

Parent Object	Object Kept in Sync
APIC EPG to VMM Domain Mapping	UCS VLAN with name from tenant-ap-epg-Vx
APIC EPG to VMM Domain Mapping	UCS VLAN in DJL2 set if allowed
APIC EPG to VMM Domain Mapping	UCS System QoS Best Effort MTU to 9000
APIC EPG to VMM Domain Mapping	UCS VMQ Policy of ACIleafVMQ
APIC EPG to VMM Domain Mapping	UCS Adapter QoS Policy of ACIleafHV
APIC EPG to VMM Domain Mapping	UCS Network Control Policy of ACI-LLDP
APIC EPG to VMM Domain Mapping	UCS vNIC Updating Templates based on VMM name (1 on fabric A, 1 on fabric B)

User Story 2 (and included in 10): VMM hosts on UCS Domain – VXLAN

Starting Condition

The connectivity in this case is the same as in user story 1 in the prior section. The real difference is the vShield integration to the vCenter will be reflected in our API dealings with the APIC controller. Again, we do not talk directly to vCenter or vShield, rather we use the “normalized” API methods via APIC.

The fundamental rules that we assume for this current version of the B2G tool are as follows:

- On program startup, it will still create empty adapter templates with the vCenter domain name with 1 per UCS fabric where the VM data will pass
- In this case, B2G will create the infra tenant configuration to allow VTEP communications between the VXLAN ESXi hosts, and the APIC nodes
 - We setup the DHCP forwarder needs for the infra tenant, and the BD that it uses
 - This points to the APIC controller IP on the infra VLAN – both of which B2G determines at startup
- You will see the tunnel interfaces to/from each ESXi host on the APIC leaf nodes
- The APIC setup still creates the backing VLANs out of the pool setup in the VMM domain, but will also add in the VXLAN virtual wires

Starting Stimulus

Same as the above case, but done on a VXLAN backed infrastructure.

Progress on the CLI and End State

Same as the above case.

Object Persistence

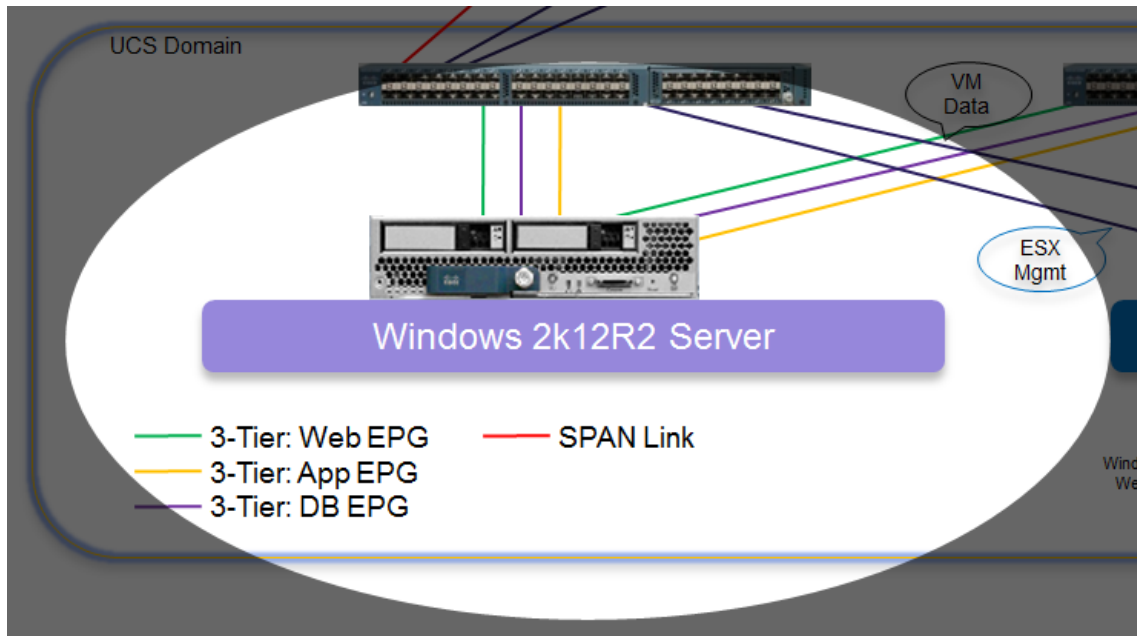
The table of items we maintain synchronization are detailed below. When the administrator removes the parent object, the tool removes the individual child objects. We do this cleanup automatically.

Parent Object	Object Kept in Sync
APIC EPG to VMM Domain Mapping	UCS multicast policy of for-VXLAN-mcast
B2G program with VXLAN case enabled (vShield seen)	UCS Infrastructure VLAN for VTEPs called B2G-VXLAN-Infrastructure with the infra VLAN
B2G program with VXLAN case enabled (vShield seen)	UCS transport VLAN since we wrap the VXLAN's into a VLAN through the UCS fabric. Value is from the B2G.init, and is only significant from the DJL2 to the adapter. VLAN ID's are removed at the DJL2 uplink, and also at the adapter to the ESX host using VXLAN
B2G program with VXLAN case enabled	APIC infra tenant DHCP relay policy B2G-vShield-DHCP-Relay
B2G program with VXLAN case enabled	APIC infra tenant DHCP relay label same as above

User Story 3 (and included in 10): Bare Metal (UCS Blade) Installation

Starting Condition

Now, we will focus in on the Bare Metal server area of the key drawing as shown below.



The fundamental rules that we assume for this current version of the B2G tool are as follows:

- The B2G tool will create a physical domain based on the UCS domain name
- The B2G tool will create a physical domain VLAN pool for all UCSM domains
 - The range of VLANs are in the B2G.init file and are used by the tool to create needed bindings for talking to the UCS domain
- When the APIC maps an EPG to this physical domain, multiple things happen:
 - We do the binding automatically on ACI leaf ports based on LLDP adjacency information
 - A backing VLAN with the truncated name of tenant-appprofile-epg is created on the UCS
 - The updating adapter template is created on the UCS domain with the needed backing VLAN
 - If allowed in the B2G.init file, we update the UCS disjoint L2 uplinks to put this VLAN on the right uplinks
- The ACI leaf bindings towards the UCS Fabric Interconnects are updated for the VPC facing the UCS domain with correct VLAN mapping (the tool pulls a VLAN ID from the pool)
- When the UCS Service Profiles are setup with vNICs drawn from those updating templates, we write the CDN (Consistent Device Name) entity which will allow the installed OS (only on Windows as of this document writing) to show the actual CDN name to the server interfaces. This makes it very easy to see which APIC EPGs our server adapters map into

Starting Stimulus

The APIC administrator maps EPG's to the UCS physical domain.

Progress on the CLI and End State

When the physical domain is being associated to the EPG, the tool will pick a VLAN from the VLAN pool in a top down fashion from the VLAN range that is defined within the b2g.init file. This event will trigger the script to auto-configure the APIC path bindings needed, and configure the corresponding VLAN on the UCS with the VLAN name of the tenant-AP-EPG-P (the P is added to signal a physical segment). The VLAN will be come available to be consumed by any bare metal servers. The tool will also create an updating adapter template with this backing EPG name and VLAN, and apply this name to the CDN (consistent device naming) field such that an installed OS will have visibility to the EPGs they connect to. You will see the memory object creation (BM_ROOT, and BM_CHILD).

Object Persistence

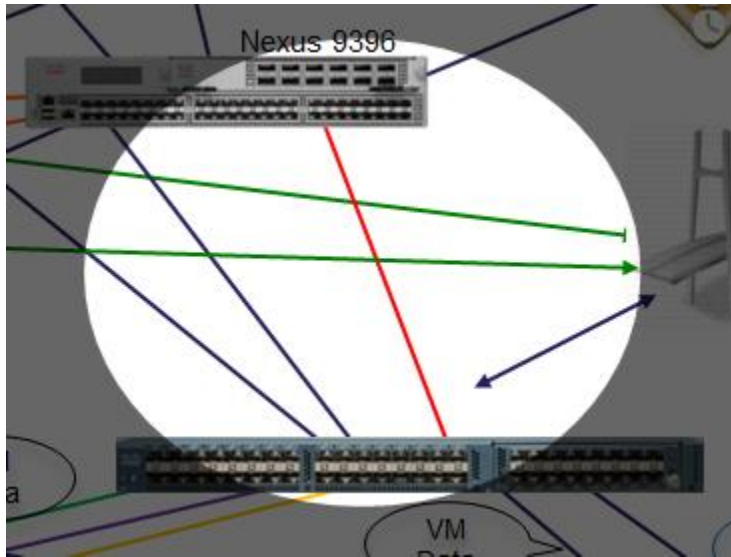
The table of items we maintain synchronization are detailed below. When the administrator removes the parent object, the tool removes the individual child objects. We do this cleanup automatically.

Parent Object	Object Kept in Sync
APIC EPG mapping to UCS physical domain	APIC leaf port bindings
APIC EPG mapping to UCS physical domain	UCS VLAN with name from tenant-ap-epg-p
APIC EPG mapping to UCS physical domain	UCS VLAN in DJL2 set if allowed
APIC EPG mapping to UCS physical domain	UCS vNIC Updating Templates based on VMM name (1 on fabric A, 1 on fabric B)
APIC EPG mapping to UCS physical domain	UCS CDN on the actual vNIC's shown under the service profile network segments

User Story 4 (and included in 10): SPAN

Starting Condition

Now, we will focus in on the SPAN area of the key drawing as shown below. The SPAN occupies its own physical link from the UCS Fabric Interconnect to the ACI leaf. The B2G tool still uses CDP adjacency information automatically to find the connectivity. The tool helps facilitate of creating SPAN policy on the ACI. When SPAN is created on the UCS then and through CDP entries the tool can determine the physical attachment and create the corresponding monitoring policy on ACI.



The fundamental rules that we assume for this current version of the B2G tool are as follows:

- The B2G tool will create a SPAN source group on the APIC leaf port mapped to the UCS SPAN destination port
- Later, the ACI admin can map these to the same SPAN destination groups as they do the rest of the ACI infrastructure
- The UCS admin, would still configure its SPAN source in this release

Starting Stimulus

The UCS administrator configures a LAN monitoring session with the destination of the port connected to an ACI leaf. When we set this up, we will configure everything on the ACI side automatically.

Progress on the CLI and End State

You will see the memory object operations, around SPAN_ROOT and SPAN_CHILD.

Object Persistence

The table of items we maintain synchronization are detailed below. When the administrator removes the parent object, the tool removes the individual child objects. We do this cleanup automatically.

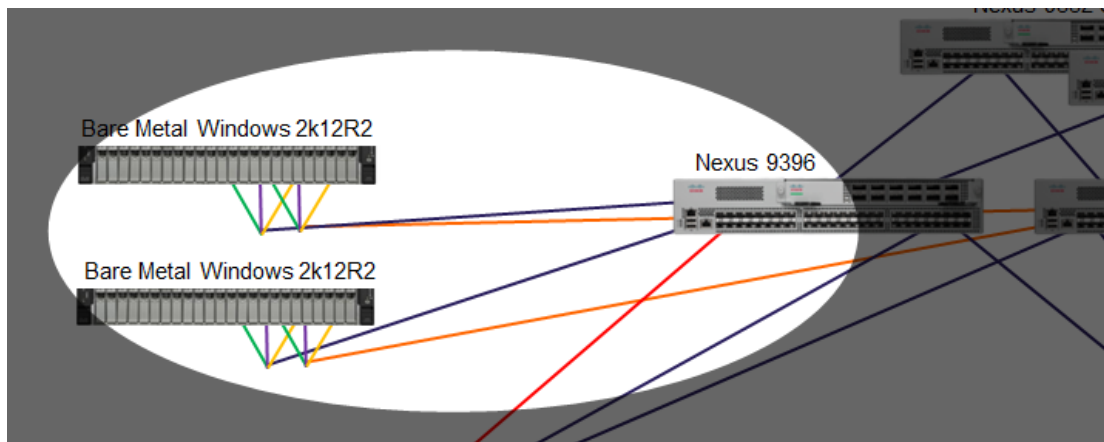
Parent Object	Object Kept in Sync
UCS LAN monitoring port setup	APIC SPAN source group

User Story 5 (NOT included in 10): Rack Installation

This is a user story that talks to the UCS C series servers, not to the UCSM domain. We have case 10 that runs all the others, but the C series is completely separate and runs with its own process.

Starting Condition

Now, we will focus in on the bare metal rackmount server area of the key drawing as shown below. We show the programmability of the UCS C series in this case.



The fundamental rules that we assume for this current version of the B2G tool are as follows:

- The B2G tool as of this writing, handles the rack servers running bare-metal workloads only. The APIC is optimized for rack servers running VMM domains, and we can add some capability later if needed
- The B2G tool will create a physical domain based on the UCS CIMC server label field
- If you have more than 1 UCS rack server with the same server label, they will be grouped and treated as a single entity with like configurations (a rack domain)
- The B2G tool will create a physical domain VLAN pool for all UCSM rack domains
 - The range of VLANs are in the B2G.init file and are used by the tool to create needed bindings for talking to the UCS rack domain
- When the APIC maps an EPG to this physical domain, multiple things happen:
 - A backing adapter with the right VLAN with the truncated name of tenant-appprofile-epg is created on the VIC equipped servers
 - The ACI leaf bindings towards the UCS Fabric Interconnects are updated for the VPC facing the UCS rack domain with correct VLAN mapping (the tool pulls a VLAN ID from the pool)
 - This binding is automatic, from the VLAN pool and also learned LLDP adjacency information
- When the UCS rack server is setup, we write the CDN (Consistent Device Name) entity which will allow the installed OS (only on Windows as of this document writing) to show the actual CDN name to the server interfaces. This makes it very easy to see which APIC EPGs our server adapters map into

Starting Stimulus

The APIC administrator maps an EPG to a rack physical domain, the actions will start.

Progress on the CLI and End State

We will see the objects in the “rackstack” being configured, including the VLAN bindings done.

Object Persistence

The table of items we maintain synchronization are detailed below. When the administrator removes the parent object, the tool removes the individual child objects. We do this cleanup automatically.

Parent Object	Object Kept in Sync
APIC EPG to rack domain mapping	APIC port bindings
APIC EPG to rack domain mapping	CIMC adapter existence

Usage Tidbits:

CIMC Server Limits:

- We do not do a vPC setup at this time (OS teaming but on per host basis) – please inform us if that is a priority
- Assumes 1 physical link from VIC port to an individual ACI leaf
- Need to refresh on the vNIC screen (not automatic)
- Speed of the API on CIMC - give it some number of seconds (~20s) per interface
- When running an ESXI or other hypervisor host on rack, this tool does nothing as the APIC is already optimized for this case

UCSM Setup:

- Assumes a vPC from each FI to a ACI leaf pair
- Single UCS domain for now, but will add to this quickly

APIC Setup:

Program Usage:

- Passwords in clear text on the .init file, until a more productized version is done

APIC

The minimum software version level that APIC must be on is 1.0(1k). Currently this tool has been tested up to the 1.1(1j) release.

UCSM

The minimum software version level that UCSM must be on is 2.2(4b). Currently this tool only supports a single UCS domain. Expanding support for multiple UCS domains will be added at a later time.

TIP: Use visore (<https://APIC-IP/visore.html>) to help understand the management object (MO) structure and classes for both UCS and APIC.

CIMC

The minimum code level the rack needs to be on is 2.0(4) or later. The description fields needs to be a unique identifiable name because the script keys off of the description filed and use that field to

configure a grouped set of C series servers as a physical domain on the ACI. The name should not contain any blank spaces.

TIP: The API calls to the CIMC takes about 20 seconds per interface. If this is not working, then try refreshing on the CIMC.

vPC is currently not supported with rack in standalone mode. High availability and increase bandwidth can be achieved through OS NIC teaming level – per host basis.

Additional Notes

Logging can be enabled and also create debug.out file within the same directory. This can be used to monitor the API calls and/or event updates from the systems. To use the logging tool, open a separate session to the linux system and monitor the debug.out file ==> tail -f debug.out. From the b2g.php, hit enter to list the debug menu when running interactive.

When running as a daemon, the system will keep the nohup.out file up to date, so you can do a tail -f nohup.out to view what is going on.

For more information

The following internal Cisco newsgroup alias has been created for this tool: aci-ucs-b2g@cisco.com

The following external newsgroup alias has been created for this tool: aci-ucs-b2g-external@cisco.com

This program is developed by Dan Hanson (danhanso@cisco.com) on a framework built by Roger Andersson



Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
168 Robinson Road
#28-01 Capital Tower
Singapore 068912
www.cisco.com
Tel: +65 6317 7777
Fax: +65 6317 7799

Europe Headquarters
Cisco Systems International BV
Haarlerbergpark
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands
www-europe.cisco.com
Tel: +31 0 800 020 0791
Fax: +31 0 20 357 1100

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

Cisco Systems, Inc. All rights reserved. CCVP, the Cisco logo, and the Cisco Square Bridge logo are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, GigaStack, HomeLink, Internet Quotient, IOS, IP/TV, IQ Expertise, the IQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networking Academy, Network Registrar, Packet, PIX, ProConnect, RateMUX, ScriptShare, SlideCast, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0609R)