Cisco ACI PowerTool Quick Start Guide

Nov 15, 2014

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.
THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version
of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.
IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network
topology diagrams, and other figuresincluded in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbersin illustrative content is unintentional
and coincidental.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: http://www.cisco.com/go/trademarks. Third-party trademarks mentioned 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. (1110R)

Table of Contents

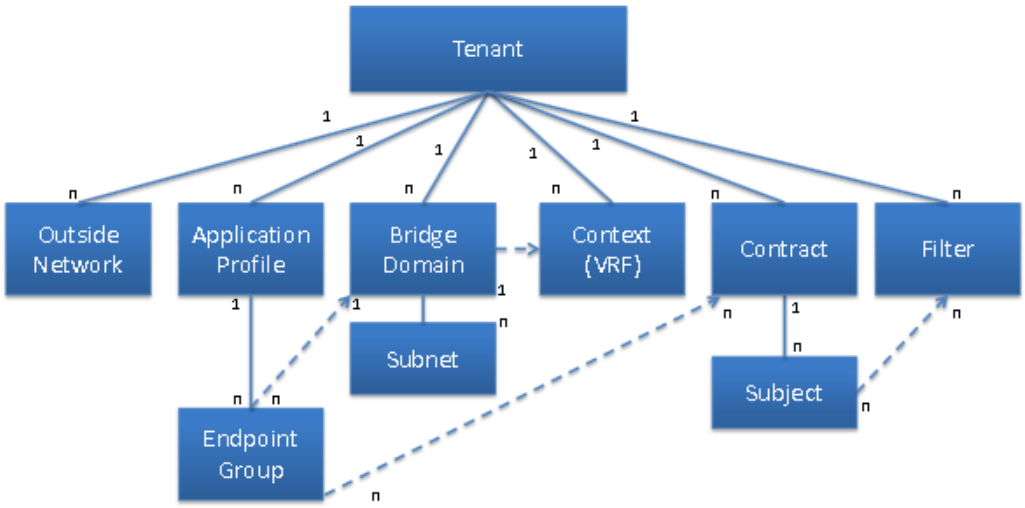
1 Overview

Cisco ACI PowerTool is a PowerShell module which helps automate all aspects of Cisco ACI management including applications, network and storage. PowerTool enables easy integration with existing IT management processes and tools.

Bulk of the PowerTool cmdlets work on the ACI's Management Information Tree (MIT), performing create, modify or delete actions on the Managed Objects (MO) in the tree. The next chapter provides an overview of the Cisco ACI's Management Information Model (MIM) and relation of PowerTool cmdlets with it.

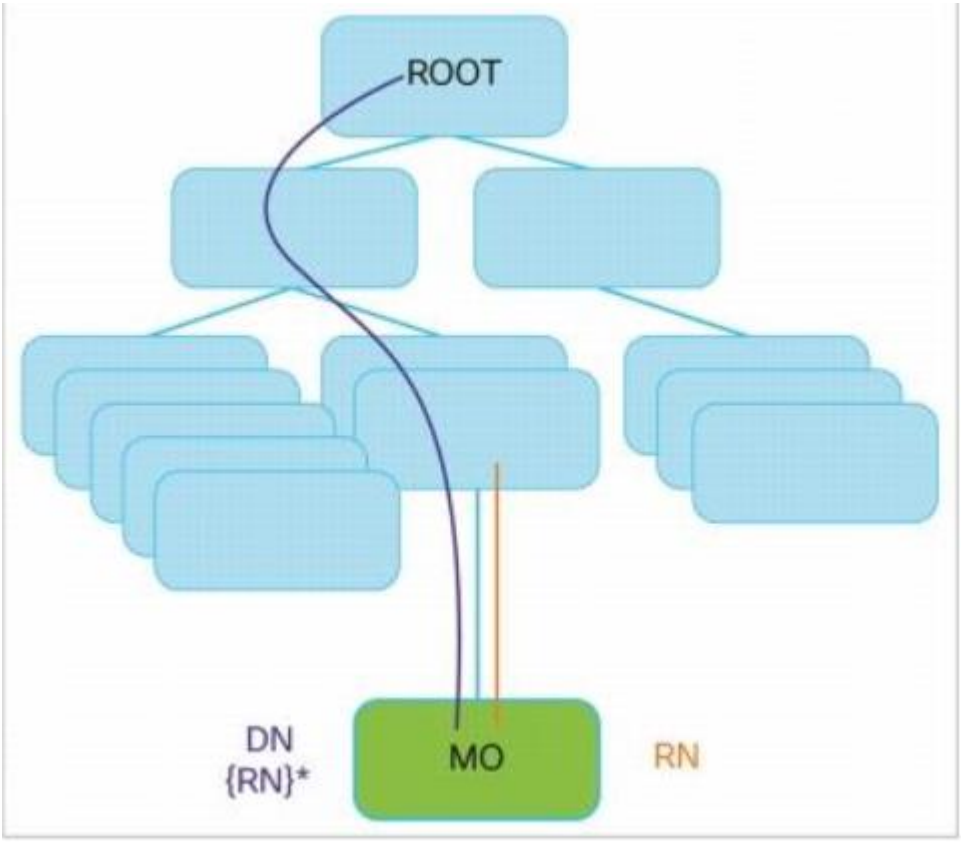
2 Management Information Model

The fabric comprises the physical and logical components as recorded in the Management Information Model (MIM), which can be represented in a hierarchical management information tree(MIT).



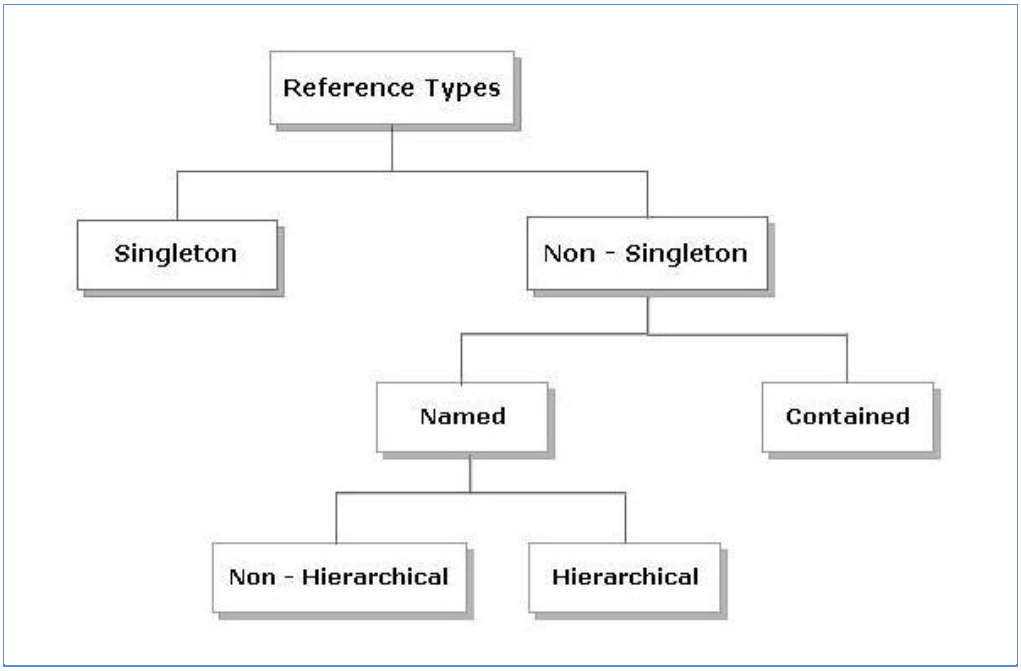
2.1 Locating Objects in the MIT

The Cisco ACI uses an information-model-based architecture (management information tree [MIT]) in which the model describes all the information that can be controlled by a management process. Object instances are referred to as managed objects (MOs). The following figure shows the distinguished name, which uniquely represents any given MO instance, and the relative name, which represents the MO locally underneath its parent MO. All objects in the MIT exist under the root object.



2.2 References to Managed Objects

The different types of references can be classified as shown below:



A singleton MO type is found at most once in the entire MIT and is typically referred to implicitly.

Non-Singleton MO type may be instantiated one or more times in the MIT. In many cases, when an MO refers to another, the reference is made by name. Depending on the type of the referenced MO, the resolution may be hierarchical.

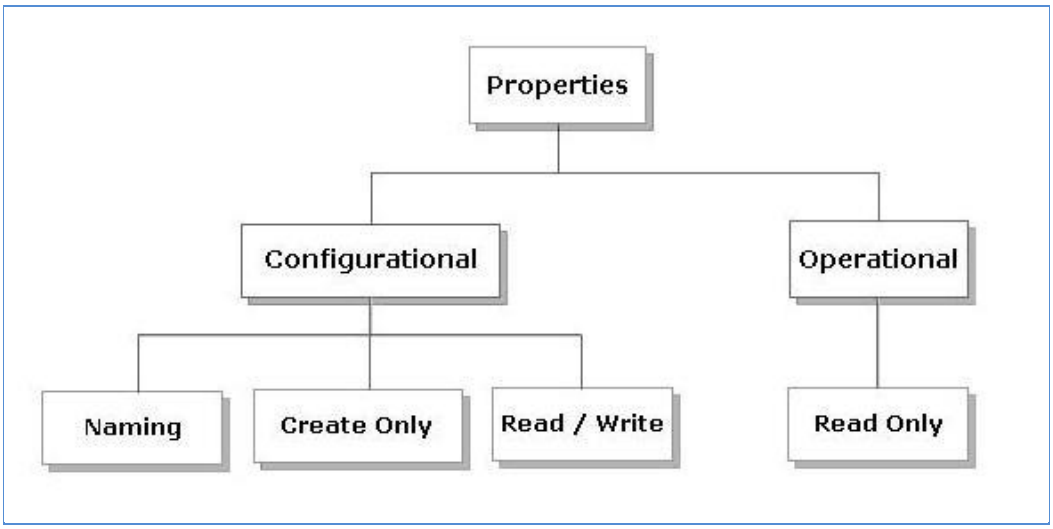
2.3 Properties of Managed Objects

Properties of Managed Objects may be classified as Configuration or Operational.

Configuration properties may be classified as:

- **Naming properties: Form part of the Rn. *Needs* to be specified only during MO creation and cannot be modified later.**
- **Create-Only properties: *May* be specified only during MO creation and cannot be modified later. If the property is not specified, a default value is assumed.**
- **Read / Write properties: *May* be specified during MO creation and can also be modified subsequently.**

Operational properties indicate the current status of the MO / system and are hence read-only.



The table below lists the examples of the various property types.

Property Type	Example
Naming	Name in fvTenant
Create-Only	Name in aaaDefinition
Read / Write	Description in fvAp
Read-Only	Dn in all class is readonly

2.4 Filters

The supported filters are:

1. **Property Filters:**
    a. **-callbits – Match if all specified values present in a multi-valued property**
    b. **-canybit – Match if any of the specified values present in a multi-valued property**
    c. **-bw – Match if the property's value lies between the two values specified**
    d. **-ceq – Match if property's value is the same as the specified value (Case sensitive)**
    e. **-ge – Match if property's value is greater than or equal to the specified value**
    f. **-gt - Match if property's value is greater than the specified value**
    g. **-le – Match if property's value is lesser than or equal to the specified value**
    h. **-lt – Match if property's value is lesser than the specified value**
    i. **-ne – Match if property's value is not equal to the specified value**
    j. **-cmatch – Match if property's value matches the keyword specified (Case sensitive)**
    k. **–cnotmatch – Match if property's value does not match the keyword specified(Case sensitive)**
    l. **–clike – Match if property's value matches the pattern specified (Case sensitive)**
    m. **–cnotlike - Match if property's value does not match the pattern specified(Case sensitive)**
    n. **–ccontains – Match if multi-values property specified**
    o. **-cnotcontains**
2. **Composite Filters (Acts on sub-filters)**
    a. **not – Negates result of sub-filter**
    b. **and – True, if all the sub-filters return true**
    c. **or – True, if any of the sub-filters return true**

2.5 PowerTool Mapping

Except few custom cmdlets, most of the PowerTool cmdlets are generated from the MO specification. Get, Add, Set, Remove cmdlets or a subset is generated for the various MO types. All cmdlets support the Xml parameter, which dumps the Xml request and response on the screen. ACI support REST protocol hence appropriate URL is built to for each cmdlet and REST calls (GET/POST/DELETE) are used to push/get configuration through cmdlets

Add Cmdlet: Appropriate payload is built based on the parameters passed to cmdlet and REST POST call is made to ACI on an appropriate URL to push the configuration to ACI.

Get Cmdlet: REST GET call is made to ACI on an appropriate URL to fetch the data from ACI. If any property parameters are specified, they are used to generate "eq" filters. If multiple property parameters are specified, the multiple "eq" filters are combined with a "and" filter.

Set Cmdlet: Appropriate payload is built based on the parameters (which needs to be modified) passed to cmdlet and REST POST call is made to ACI on an appropriate URL to push the configuration to ACI. If the Force parameter is specified, there will be no prompt for confirmation.

Remove Cmdlet: REST DELETE call is made to ACI on an appropriate URL to remove particular MOs from ACI. If the Force parameter is specified, there will be no prompt for confirmation.

The table below lists the properties that can be specified for a given Verb:

Property	Get	Add	Set
Naming	Yes (Positional)	Yes (Positional)	No
Create-Only	Yes	Yes	No
Read-Write	Yes	Yes	Yes
Operational/ Read-Only	Yes	No	No

The table below lists the types that can come down the pipeline for corresponding cmdlets:

Verb / Type	Pipeline Input
Get	Singleton – none Non-singleton – Parent Type
Add	Singleton – none Non-singleton – Parent Type
Set	MO has naming property – Same Type MO has no naming property – Same or Parent Type
Remove	Same Type

Get-AciCmdletMeta is useful cmdlet to explore the MO types, the corresponding nouns, supported Verbs, properties of the MOs, the details of properties including the type (Naming, Read/Write etc.) and the version of ACI the property was introduced in etc.

3 Installation

3.1 Pre-Install Checklist

- **Ensure you have .NET version 4.5 installed on your system.**
- **Ensure you have PowerShell v4 installed in your system.**
- **Close any instances of PowerShell running with the PowerTool module loaded.**

3.2 Installation Steps

- **Download & Launch Installer.**
- **After successful installation, shortcut will be created on desktop.**

3.3 Getting Started

 # Launch "Cisco ACI PowerTool" Shortcut on Desktop.
 # View all Cmdlets, Functions, Alias supported by Cisco ACI PowerTool.

```
Get-Command -Module CiscoACIPS
Get-Command -Module CiscoACIPS | group CommandType
Get-Command -Module CiscoACIPS | measure
```

 # Connect to a ACI system using https. Print $handle to see the response

```
$handle = Connect-Aci <ip or hostname> -NotDefault
```

 # If login is successful Connect-Aci, by default, adds the ACI handle to the Default ACI
 # list, unless the -NotDefault option is specified. Every cmdlet that operates on ACI
 # takes the –Aci parameter, where the handle can be specified.
 # If username is with the domain, we need to put "domain\username" in the username field.

 # Connect to a ACI system using http. Print $handle to see the response

```
$handle = Connect-Aci <ip or hostname> -NoSsl -Notdefault
```

 # Connect to a ACI system using proxy.

```
$proxy = New-Object System.Net.WebProxy
$proxy.Address = "http:\\<url>:<port>"
$proxy.UseDefaultCredentials = $false
$proxy.Credentials = New-Object System.Net.NetworkCredential("<user name>",
"<password>")
$handle = Connect-Aci <ip or hostname> -Proxy $proxy
```

 # Get ACI Tenants information.

```
Get-AcifvTenant -Aci $handle
```

 # Get ACI application profile information.

```
Get-AcifvAp -Aci $handle
```

 # Disconnect.

```
Disconnect-ACI –Aci $handle
```

3.4 Default ACI

 # If no handle or name is specified, the ACI handle is added to a Default ACI list
 # Unless the –Aci parameter is specified, the first cmdlet in the pipeline operates
 # on the Default Aci List.

```
Connect-Aci <ip or hostname>
```

 # Get the Default ACI list.

```
Get-AciPSSession
```

 # Get ACI's Tenant information.

```
Get-AciFvTenant
```

 # Enable HTTP on the FI.

```
Get-AciCommHttp | Set-AciCommHttp -AdminSt enabled
```

 # Disable HTTP on the FI.

```
Get-AciCommHttp | Set-AciCommHttp -AdminSt disabled
```

 # Disconnect.

```
Disconnect-Aci
```

3.5 Default ACI list with Multiple ACI

 # PowerTool cmdlets can work with multiple ACI systems.
 # This can be done by specifying multiple handles.

 # Connect to a ACI system.

```
$handle1 = Connect-Aci <ip1> -NotDefault
$handle2 = Connect-Aci <ip2> -NotDefault
Get-AcifvTenant -Aci $handle1,$handle2
Disconnect-Aci -Aci $handle1,$handle2
```

 # By default, Multiple ACI handles are not allowed in DefaultAci.

 # This can be overridden using the Set-AciPowerToolConfiguration cmdlet.

```
Get-AciPowerToolConfiguration
Set-AciPowerToolConfiguration -SupportMultipleDefaultAci $true

Connect-Aci <ip1>
Connect-Aci <ip2>
```

 # Tenant information will come from the default ACI Handle

```
Get-AcifvTenant
Disconnect-Aci
```

 # Connect to multiple Aci using same credentials.

```
$user = "<username>"
$password = "<password>" | ConvertTo-SecureString -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential($user,
$password)
$servers = @("<aci1>", "<aci2>", "<aci3>")
Connect-Aci $servers -Credential $cred
```

3.6 Credentials to/from File

```
Connect-Aci <ip1>
```

 # Credentials can be stored to a file.
 # The stored credentials are encrypted with a specified Key.

```
Export-AciPSSession -LiteralPath C:\work\labs.xml
```

 # Disconnect ACI

```
Disconnect-Aci
```

 # Login can be initiated from credentials stored in a file.

```
Connect-Aci -LiteralPath C:\work\labs.xml
```

 # Specify proxy while logging in with credentials stored in a file.

```
$proxy = New-Object System.Net.WebProxy
$proxy.Address = "http:\\<url>"
$proxy.UseDefaultCredentials = $false
$proxy.Credentials = New-Object System.Net.NetworkCredential("<user name>",
"<password>")
Connect-Aci -LiteralPath C:\work\lab.xml –Proxy $proxy
```

3.7 SSL Handling

#when a user connects to a Cisco ACI and the server cannot recognize any valid certificates; connection establishment depends on InvalidCertificateAction.
 # InvalidCertificateAction is set to Ignore by default.
 # By default PowerTool is configured to establish the connection without taking into account # if the certificate is invalid.

 # This can be overridden using the Set-AciPowerToolConfiguration cmdlet.

```
Get-AciPowerToolConfiguration
Set-AciPowerToolConfiguration -InvalidCertificateAction Fail
```

 # Fail - cmdlet will not establish connection if the certificate is not valid.

 # Ignore - cmdlet will establish the connection without taking into account that the certificate is invalid.

 # Default – Default Windows behavior, cmdlet will establish connection if the certificate is valid.

3.8 Aliases

 # Some aliases have been defined for convenience.

```
gal | ? { $_.Name -like "*-Aci*" } | select Name
```

```
Name
----
Acknowledge-AciFault
```

4 Examples

4.1 PowerTool Cmdlet Generation

`ConvertTo-AciCmdlet` cmdlet will be given to users so that it generates cmdlets syntax on the shell for specific actions on ACI GUI.

This cmdlet requires additional setup of log server. The log server is packaged with powertool bundle.

1. **Set JAVA_HOME variable to your jre path (C:\Program Files (x86)\Java\jre7) from my computer->properties->Advanced-system-settings->environment variables**
2. **Go to command line and type "java –version" to make sure your variable is set properly**
3. **Go to installation directory of ACI Power Tool and go to subdirectory - \apache-tomcat-7.0.54\webapps\CiscoLogServer\WEB-INF\classes\config.properties and assign your ACI url against aci_url variable**
4. **Go to installation directory of ACI powertool and go to subdirectory - /cisco/ciscoACIPS/apache-tomcat-7.0.54/bin and run "startup.bat" as administrator.**
5. **Open log server URL(e.g. http://localhost:8987/ciscologserver or https://localhost:8443/ciscologserver if ACI is running on https) in web browser and you shell see "Welcome To Cisco Log Server" which shows that your server is started and running.**
6. **Login into ACI UI and start remote logging. Set URL to http://{your_server}:8987/ciscologserver/acilog or https://{your_server}:8443/ciscologserver/acilog (If ACI is running on https)**

#Start ConvertTo-AciCmdlet

```
ConvertTo-AciCmdlet –user cisco –password cisco -LogServerUrl
http://localhost:8987/ciscologserver/acilog
```

 # All the GUI actions on ACI will get converted to cmdlets like below:

```
Add-AciFvTenant -Name "abc" -XtraProperty @{OwnerTag=""; OwnerKey="";
Descr="";}

Get-AciFvTenant -Name "abc" | Set-AciFvTenant -Descr "This is testing" -
XtraProperty @{OwnerTag=""; OwnerKey=""; }

Get-AciFvTenant -Name "abc" | Add-AciFvAp -Name "ap" -XtraProperty
@{OwnerTag=""; OwnerKey=""; Prio=""; Descr=""; }
```

4.2 Faults

 # Retrieve faults, group them by severity.

```
Get-AciFaultInst | group Severity
```

 # Retrieve critical faults.

```
Get-AciFaultInst -Severity major | select Dn, Cause
```

 # Acknowledge all unacknowledged faults.
```
Get-AciFaultInst | ? { $_.Ack -ceq "no" } | Acknowledge-AciFault
```

 # List all acknowledged major severity faults.
```
Get-AciFaultInst | ? { $_.Ack -ceq "no" } | ? { $_.severity -ceq "major" }
```

4.3 Get Cmdlet -Subtree Flag

 # Get Managed Object including its entire subtree.
```
Get-AcifvTenant –Name Acme –subtree | select dn
```

4.4 Get Cmdlet –Children Flag

 # Display tenant information including its immediate children
```
Get-AcifvTenant -Name Acme –children | select dn
```

 # Display application profile and its immediate children under tenant Acme
```
Get-AcifvTenant –Name Acme | Get-AcifvAp -children
```

4.5 Tenant Configuration

 # Add new Tenant called Acme
```
Add-AcifvTenant –Name Acme –Descr “This is Testing project”
```

 # Remove newly added Tenant called Acme without asking for acknowledgement
```
Get-AcifvTenant –Name Acme | Remove-AcifvTenant -force
```

4.6 Application Profile Configuration

 # Create a new Tenant
```
$tenant = Add-AcifvTenant –Name Acme –Descr “Testing”
```

 # Add Application profile Under Tenant Acme
```
$tenant | Add-AcifvAp –Name www.acme.com –Descr “This is testing”
```

 # Remove newly added Application profile
```
Get-AcifvAp –Name www.acme.com | remove-AcifvAp -force
```

4.7 End Point Group Configuration

 # Create a new Tenant
```
$tenant = Add-AcifvTenant –Name Acme –Descr “Testing”
```

 # Add Application profile Under Tenant Acme
```
$appProfile = $tenant | Add-AcifvAp –Name www.acme.com –Descr “This is
testing”
```

 # Add WEB, APP, DB EPGs Under Tenant Acme and Application profile www.acme.com
```
$epgWeb= $appProfile| Add-AcifvAEpg –Name WEB
$epgApp = $appProfile| Add-AcifvAEpg –Name APP
$epgDb = $appProfile| Add-AcifvAEpg –Name DB
```

4.8 Physical Domain

 # Create a new Tenant
```
$tenant = Add-AcifvTenant –Name Acme –Descr “Testing”
```

 # Create a new bridge domain
```
$bd = $tenant | Add-AcifvBd –Name BD1
```

 # Add Application profile Under Tenant Acme
```
$appProfile = $tenant | Add-AcifvAp –Name www.acme.com –Descr “This is
testing”
```

 # Add WEB EPG Under Tenant Acme and Application profile www.acme.com
```
$epgWeb= $appProfile| Add-AcifvAEpg –Name WEB
```

 # Bind bridge domain with EPG
```
$epgWeb | set-AcifvRsBd –TnFvBDName $bd.name
```

 # Remove the Tenant and all its children
```
Get-AcifvTenant –Name Acme | Remove-AcifvTenant
```

4.9 Contracts and Filters

 # Create a new Tenant
```
$tenant = Add-AcifvTenant –Name Acme –Descr “Testing”
```

 # Add Application profile Under Tenant Acme
```
$appProfile = $tenant | Add-AcifvAp –Name www.acme.com –Descr “This is
testing”
```

 # Add WEB EPG Under Tenant Acme and Application profile www.acme.com
```
$epgWeb= $appProfile| Add-AcifvAEpg –Name WEB
```

 # Add a filter under tenant Acme
```
$filter = $tenant | Add-AcivzFilter –Name Webfilter1
```

 # Create an entry under filter
```
$filter | Add-AcivzEntry –Name HttpPort –EtherT ip –Prot 6 –SfromPort 443 –
StoPort 443 –DfromPort 25 –DtoPort 25
```

#create a binary contract (vz.BrCP object) container within the tenant
```
$contract = $tenant | add-Acivzbrcp –name WebContract
```

#create a subject container for associating the filter with the contract

```
$subject = $contract | add-Acivzsubj –Name WebSubject
$subject | add-AciVzRsSubjFiltAtt –TnVzFilterName $filter.name
```

#Add newly created contract as consumed contract with the EPG WEB
```
$epgWeb | Add-AcifvRsCons –tnVzBrCPName $contract.name
```

#Add newly created contract as provider contract with the EPG WEB
```
$epgWeb | Add-AcifvRsProv –tnVzBrCPName $contract.name
```

 # Remove the Tenant and all its children
```
Get-AcifvTenant –Name Acme | Remove-AcifvTenant
```

4.10 Namespace

 # Create Vlan Instance
```
$vlanInst = Add-AcifvnsVlanInstP –Name namespace1 –AllocMod dynamic
```

 # Create encapsulation on vlan instance.
```
$vlanInst | Add-AcifvnsEncapBlk –Name encap –From vlan-5 –To vlan-10
```

 # Remove newly created vlan Instance
```
Get-AcifvnsVlanInstP –Name namespace1 | Remove-AcifvnsVlanInstP -Force
```

4.11 VM Networking

 # Create Vlan Instance
```
$vlanInst = Add-AcifvnsVlanInstP –Name namespace4 –AllocMod dynamic
```

 # Create encapsulation on vlan instance.
```
$vlanInst | Add-AcifvnsEncapBlk –Name encap –From vlan-5 –To vlan-10
```

 # Get VMware Vendor
```
$vmProvider = Get-AciVmmProvP –Vendor VMware
```

 # Create VMware domain
```
$vmDomain = $vmProvider | Add-AciVmmDomP -Name CrestDataSys
```

 # Create user account
```
$useraccount = $vmDomain | Add-AciVmmUsrAccP –Name Default –Pwd ‘in$1eme’ –
Usr administrator
```

 # Bind Vlan instance with vmDomain
```
$vmDomain | Set-AciInfraRsVlanNs –TDn $vlanInst.Dn
```

 # Create Controller
```
$controller = $vmDomain | Add-AciVmmCtrlrP –Name vcenter-01 –HostOrIp
172.21.128.166 –rootContName CrestDataSys
```

 # Associate Controller with user account
```
$controller | Set-AciVmmRsAcc –Tdn $useraccount.Dn
```

```
#remove newly created vmware domain
Get-AciVmmDomP -Name CrestDataSys | Remove-AciVmmDomP
```

4.12 Configuration Import/Export

 # Create and download configuration backup of ACI. This creates a binary file that includes a snapshot of the entire configuration on ACI. You can use the file generated from this backup to restore the system during disaster recovery. This file can restore or rebuild the configuration on the original ACI or recreate the configuration on a different ACI. You can use this file for an import.

```
Export-AciConfiguration –Protocol scp –RemoteHost 172.21.128.123 –RemotePort
22 –RemotePath /home/ -UserName root –Password insieme
```

#Above command will create a file on remote server with following format. This file will have entire configuration in xml format

```
ce_configBackup-2014-05-16T00-23-31.tar.gz
```

4.13 Import Configuration

 # Import configuration which was previously backed up to ACI
 # You can perform an import while the system is up and running

 # ImportType=replace option will replace the existing configuration on ACI. This is the default option

```
Import-AciConfiguration –Protocol scp –RemoteHost 172.21.128.123 –RemotePort
22 –RemotePath /home/ -UserName root –Password insieme –FileName
ce_configBackup-2014-05-17T03-13-11.tar.gz –importType replace –importMode
<atomic/best-effort>
```

 # ImportType=merge option will merge the configuration with existing configuration

```
Import-AciConfiguration –Protocol scp –RemoteHost 172.21.128.123 –RemotePort
22 –RemotePath /home/ -UserName root –Password insieme –FileName
ce_configBackup-2014-05-16T00-23-31.tar.gz –importType merge –importMode
<atomic/best-effort>
```

4.14 Tech Support

Get-AciTechsupport cmdlet will be given to users so that technical support data for the entire ACI instance will be created and downloaded to the specified file.

```
Export-AciTechSupport -Protocol scp -RemoteHost 10.0.1.211 -RemotePort 22 -
RemotePath /root/ -UserName root -Password redhat
```

This will create a file called "tsexp-default_apic1_sysid-1_2014-09-04T00-25.tgz" on the host given in the cmdlet in a specified location.

4.15 Filters

#Get Application Profile with name = test

```
Get-AcifvAp -Filter 'name -ceq Test' | select Dn, Name
```

 # Get all application Profiles where name is not equal to Test.

```
Get-AcifvAp -Filter 'name -cne Test' | select Dn, Name
```

 # Get all fabric node where Id is less than 20

```
Get-AciFabricNode -Filter 'id –lt 20' | select Name, Dn
```

 # Get all fabric node where Id is greater than 20

```
Get-AciFabricNode -Filter 'id –gt 20' | select Name, Dn
```

 # Get all objects with health score less than or equal to 80

```
Get-AciHealthInst –Filter ‘cur –le 100’ | select Dn, MaxSev
```

 # Get all objects with health score greater or equal to 80

```
Get-AciHealthInst –Filter ‘cur –ge 95’ | select Dn, MaxSev
```

 # Get all faults created between 4/18 and 4/19.

```
Get-AciFaultInst -Filter 'Created -bw "6/15/2014 9:00","6/15/2014 12:00"' |
select Cause, Dn, Created
```

 # Get roles with one of the fabric l1, l2, l3 privilege

```
Get-AciAaaRole -Filter 'priv -canybit fabric-connectivity-l1,fabric-connectivity-
l2,fabric-connectivity-l3’ | select Name, Dn
```

 # Get roles with all of the fabric l1, l2, l3 privilege

```
Get-AciAaaRole -Filter 'priv -callbits fabric-connectivity-l1,fabric-
connectivity-l2,fabric-connectivity-l3’ | select Name, Dn
```

 # Get all tenants where name has "Test" word in it

```
Get-AcifvTenant -Filter 'name -cmatch Test' | select Name, Dn
```

 # Get all tenants where name does not have "Test" word in it

```
Get-AcifvTenant -Filter 'name -cnotmatch Test' | select Name, Dn
```

 # Get all tenants where name starts with "Acme"
```
Get-AcifvTenant -Filter 'name –clike Acme*' | select Name, Dn
```

 # Get all tenants where name does not start with "Acme"
```
Get-AcifvTenant -Filter 'name -cnotlike Acme*' | select Name, Dn

#Get all tenants where name is either A or B
Get-AciFvTenant –Filter 'name –ceq A –or name –ceq B'

#Get all tenants where name is A and description is "Testing"
Get-AciFvTenant –Filter 'name –ceq A –and descr –ceq Testing'

#Get role which contain a specific privilege
Get-AciAaaRole -Filter 'priv -ccontains fabric-connectivity-l1'

#Get role which does not contain a specific privilege
Get-AciAaaRole -Filter 'priv -cnotcontains fabric-connectivity-l1'
```

4.16 Privileges

 # List out all privileges on the APIC.
```
Get-AciPrivilege
```

4.17 User Roles

 # Add a user role "test_role" with admin privileges.
```
Add-AciAaaRole -Name user_role -Priv admin
```

 # Change privileges for a user role to allow read-and-write access to fabric-connectivity-l2
```
Get-AciAaaRole -Name user_role | Set-AciAaaRole -Priv fabric-connectivity-l2
```

 # Set multiple privileges using Set-AciAaaRole.
```
Add-AciAaaRole -Name multi_priv_role –Priv admin| Set-aciaaaRole -Priv
fabric-connectivity-l1,fabric-connectivity-l2
```

 # Get all user roles in ACI
```
Get-AciAaaRole
```

 # Get a user role by name.
```
Get-AciAaaRole -Name multi_priv_role
```

 # Remove a user role.
```
Get-AciAaaRole -Name multi_priv_role | Remove-AciAaaRole
```

4.18 Local User

 # Add a Locale.
```
Add-AciaaaUser -Name asia_pacific -Descr "Locale for Asia Pacific users"
```

 # Get all locales.
```
Get-AciaaaUser
```

 # Remove a locale.
```
Get-AciaaaUser -Name asia_pacific | Remove-AciaaaUser
```

4.19 Security Domain

 # Add security domain
```
$domain = Add-AciaaaDomain -Name ciscoSecurity
```

#Add a local user and assign newly created security domain to this user with a particular role "admin"

```
$user = Add-AciaaaUser -Name asia_pacific -Descr "Locale for Asia Pacific
users"

$userDomain = $user | add-AciaaaUserDomain –name ciscoSecurity | Add-
AciaaaUserRole –name admin
```

4.20 Remote Authentication - RADIUS

 # Set global configuration for RADIUS authentication.
```
Set-AciaaaRadiusEp  -Timeout 20 -Retries 3 –Force
```

 # Create a RADIUS server instance with server key "test1234" and maximum 2 retries.
```
Add-AciaaaRadiusProvider  -Name "192.168.23.84" –Descr "Radius Server
configuration" -Key test1234 -Retries 2
```

 # Set RADIUS as default authentication.
```
Set-AciaaaDefaultAuth -Realm radius
```

 # Set TACACS as console authentication.
```
Set-AciaaaConsoleAuth -Realm radius
```

4.21 Remote Authentication - TACACS

 # Set global configuration for TACACS authentication.

```
Set-AciaaaTacacsPlusEp -Descr "TACACS authentication configuration" -Timeout
20 -Retries 3
```

 # Add a TACACS Provider.

```
Add-AciaaaTacacsPlusProvider -Name "192.168.23.84" –Descr “ TACACS provider”
-Key test1234
```

 # Set TACACS as default authentication.

```
Set-AciaaaDefaultAuth -Realm tacacs
```

 # Set TACACS as console authentication.

```
Set-AciaaaConsoleAuth -Realm tacacs
```

4.22 Remote Authentication - LDAP

 # Set global configuration for LDAP authentication.

```
Set-AciaaaLdapEp -Descr 'LDAP authentication configuration' -Timeout 20 -
Retries 3 –Force
```

 # Add a LDAP Provider.

```
add-AciaaaLdapProvider  -Attribute 'CiscoAVPair' -Basedn
'CN=Users,DC=qasamlab,DC=com' -Key 'Bbv03515' -Name '10.193.23.84' -Rootdn
'CN=Administrator,CN=Users,DC=qasamlab,DC=com'
```

 # Set LDAP as default authentication.

```
Set-AciaaaDefaultAuth -Realm ldap
```

 # Set TACACS as console authentication.

```
Set-AciaaaConsoleAuth -Realm ldap
```

4.23 RADIUS Provider

 # Create a RADIUS server instance with server key “test1234” and maximum 2 retries.

```
Add-AciaaaRadiusProvider -Name "192.168.23.84" -Key test1234 -Retries 2
```

 # Add a RADIUS provider group and set it as the default remote authentication.

```
Add-AciaaaRadiusProviderGroup -Name radiusprovidergroup1
```

```
#Add RADIUS server in the provider group
Get-AciaaaRadiusProviderGroup -Name radiusprovidergroup1 | Add-
AciaaaProviderRef -Name "192.168.23.84"
```

 # Set Radius as default authentication.
```
Set-AciaaaDefaultAuth -Realm radius
```

#Set providergroup1 as default authentication
```
Get-AciaaaDefaultAuth | Set-AciaaaDefaultAuth -ProviderGroup
radiusprovidergroup1
```

 # Set radius as console authentication.
```
Set-AciaaaConsoleAuth -Realm radius
```

#Set providergroup1 as console authentication
```
Get-AciaaaConsoleAuth | Set-AciaaaConsoleAuth -ProviderGroup
radiusprovidergroup1
```

4.24 TACACS Provider

 # Add a TACACS provider.
```
Add-AciaaaTacacsPlusProvider -Name "192.168.23.84" -Key test1234
```

 # Add a TACACS provider group and set it as the default remote authentication.
```
Add-AciAaaTacacsPlusProviderGroup -Name tacacsprovidergroup1
```

```
#Add TACACS server in the TACACS group
Get-AciAaaTacacsPlusProviderGroup -Name tacacsprovidergroup1| Add-
AciAaaProviderRef -Name "192.168.23.84"
```

 # Set tacacs as default authentication.
```
Set-AciaaaDefaultAuth -Realm tacacs
```

#Set tacacsprovidergroup1 as default authentication
```
Get-AciaaaDefaultAuth | Set-AciaaaDefaultAuth -ProviderGroup
tacacsprovidergroup1
```

 # Set tacacs as console authentication.
```
Set-AciaaaConsoleAuth -Realm tacacs
```

#Set providergroup1 as console authentication
```
Get-AciaaaConsoleAuth | Set-AciaaaConsoleAuth -ProviderGroup
tacacsprovidergroup1
```

4.25 LDAP Provider

 # Add a LDAP Provider.
```
add-AciaaaLdapProvider  -Attribute 'CiscoAVPair' -Basedn
```

```
'CN=Users,DC=qasamlab,DC=com' -Key 'Bbv03515' -Name '10.193.23.84' -Rootdn
'CN=Administrator,CN=Users,DC=qasamlab,DC=com'
```

 # Add an LDAP provider group and set it as the default remote authentication.

```
Add-AciaaaLdapProviderGroup -Name ldapprovidergroup1
```

```
#Add LDAP server in the group
Get-AciaaaLdapProviderGroup -Name ldapprovidergroup1| Add-AciAaaProviderRef -
Name "10.193.23.84"
```

 # Set LDAP as default authentication.

```
Set-AciaaaDefaultAuth -Realm ldap
```

#Set ldapprovidergroup1 as default authentication

```
Get-AciaaaDefaultAuth | Set-AciaaaDefaultAuth -ProviderGroup
ldapprovidergroup1
```

 # Set ldap as console authentication.

```
Set-AciaaaConsoleAuth -Realm ldap
```

#Set providergroup1 as console authentication

```
Get-AciaaaConsoleAuth | Set-AciaaaConsoleAuth -ProviderGroup
ldapprovidergroup1
```

4.26 Authentication Domains

 # Authentication Domains configure simultaneous support for
 # different authentication methods (local, # TACACS+, RADIUS, and LDAP/Active Directory)
 # and provider groups.

 # Configure a TACAS Provider Group with a TACACS Provider.

```
$tp = Add-AciAaaTacacsPlusProvider -Name "192.168.23.84" -Key test1234
$tpg = Add-AciAaaTacacsPlusProviderGroup -Name tacacsprovidergroup1
$tpg | Add-AciAaaProviderRef -Name $tp.Name
```

 # Create an Login Domain and add a reference to the TACACS Provider group.

```
$ad = Add-AciaaaLoginDomain -Name adtacacs
$ad | Set-AciaaaDomainAuth -Realm tacacs -ProviderGroup tacacsprovidergroup1
```

 # Now if a user logs in from the console, GUI or XML API with the
 # user name being "adtacacs\user" the TACACS configuration created above
 # will be used for authentication.

4.27 Communication Services - Telnet

 # Get Aci telnet configuration.
```
$tn = Get-AcicommTelnet
```

 # Allow telnet connections.
```
$tn | Set-AcicommTelnet -AdminSt enabled
```

4.28 Communication Services - SNMP

 # Add Aci SNMP policy.
```
Add-AciSnmpPol –name abc
```

 # Enable SNMP
```
$ts = Get-AciSnmpPol –name abc
```

 # Enable SNMP with community string being "public"
```
$ts |Set-AciSnmpPol -Descr "SNMP config for APIC" -AdminSt enabled -Contact
CiscoSystems -loc Bangalore
```

 # Create SNMP community string "public"
```
$ts| Add-AcisnmpCommunityP –Name "public" –Descr "This is public string"
```

 # Add a Aci SNMP user.
```
$ts | Add-AciSnmpUserP -Name joe -AuthType hmac-md5-96 –AuthKey "22222222"
```

 # Set a Aci SNMP user.
```
$ts| Get-AciSnmpUserP -Name joe | Set-AciSnmpUserP –Descr "This is testing"
```

 # Remove a Aci SNMP user.
```
Get-AciSnmpPol –name abc| Get-AciSnmpUserP -Name joe | Remove-AciSnmpUserP
```

#create SNMP group which contains the information needed to send traps

$group1 = Add-AciSnmpGroup –name group1

 # Set SNMP Trap information
```
$group1 | Add-AciSnmpTrapDest –Name trap1 -host 168.65.120.32 -Port 162 –
secName ciscoSecurity
```

4.29 Communication Services – HTTP

 # Get ACI http configuration.
```
$hp= Get-AciCommHttp
```

 # Set Aci http configuration to enable http and enable http to https redirection.
```
$hp | Set-AciCommHttp -AdminSt enabled -RedirectSt enabled
```

4.30 Communication Services - HTTPS

```
# Get Aci https configuration.
$hps = Get-AciCommHttps


# Create a keyring with a key size of 1024 bits.
Add-AcipkiKeyring -Name keyring1024 -Modulus mod1024


Add-AcipkiTp -Name TPkeyring1024 -CertChain "

-----BEGIN CERTIFICATE-----
MIICmjCCAgOgAwIBAgIJAN/5bfh1yhWBMA0GCSqGSIb3DQEBBQUAMGYxCzAJBgNV
BAYTAlVTMQswCQYDVQQIDAJDQTEQMA4GA1UEBwwHQW55dG93bjEcMBoGA1UECgwT
RGVmYXVsdCBDb21wYW55IEx0ZDELMAkGA1UECwwCSVQxDTALBgNVBAMMBEFQSUMw
HhcNMTQwNzAzMDA1OTQwWhcNMTcwMzI5MDA1OTQwWjBmMQswCQYDVQQGEwJVUzEL
MAkGA1UECAwCQ0ExEDAOBgNVBAcMB0FueXRvd24xHDAaBgNVBAoME0RlZmF1bHQg
Q29tcGFueSBMdGQxCzAJBgNVBAsMAklUMQ0wCwYDVQQDDARBUElDMIGfMA0GCSqG
SIb3DQEBAQUAA4GNADCBiQKBgQC44IpMIgKAXB6a/0pABjo9UINp7NaLPaxjDVvv
PxkpVC1adXuHvvCSDUHRXGEGMOMlna8yPshC/e/xbgNtiUXaWsyOdxaGbUctJREu
wz3PWJ/DSdW8mVYRJp74Jq+VoMoMibAIbp8rQklgQB81jLvP/12gXbnforQpVh2V
RojkZwIDAQABo1AwTjAdBgNVHQ4EFgQUoWpkXrNNsRTAdSSbstklsnIeJf4wHwYD
VR0jBBgwFoAUoWpkXrNNsRTAdSSbstklsnIeJf4wDAYDVR0TBAUwAwEB/zANBgkq
hkiG9w0BAQUFAAOBgQC0vqsVxm6MpVmM63A0whUoruEsNzNUCJrzY9P9K18fphng
3FxWH7TBNahIWwcn5BJBwuZ4kG4hOFAP9b+v4wVRjfhyEUcxVehpY30yX9dhk5nt
1XjHRH65Lo0Fu7jL6U698l3kWv240mDmD4A/KxpvA1EjSe50AEJm87fEd0icDA==
-----END CERTIFICATE-----"

Get-AciPkiKeyRing keyring1024  | Set-AciPkiKeyRing -Tp TPkeyring1024

Get-AcipkiKeyRing -Name keyring1024 | Set-AcipkiCertReq -SubjName insieme

Get-Acipkikeyring –Name keyring1024 | Set-Acipkikeyring -Cert "

-----BEGIN CERTIFICATE-----
MIIB5zCCAVACAQEwDQYJKoZIhvcNAQEFBQAwZjELMAkGA1UEBhMCVVMxCzAJBgNV
BAgMAkNBMRAwDgYDVQQHDAdBbnl0b3duMRwwGgYDVQQKDBNEZWZhdWx0IENvbXBh
bnkgTHRkMQswCQYDVQQLDAJJVDENMAsGA1UEAwwEQVBJQzAeFw0xNDA3MDMwMTEy
NDBaFw0xNzAzMjkwMTEyNDBaMBIxEDAOBgNVBAMMB2luc2llbWUwgZ8wDQYJKoZI
hvcNAQEBBQADgY0AMIGJAoGBAKqb73dNC8HlkSBotIM2mkTG9WFf+5IR9STV832z
qngSu83NTkCNwcNEcT7okTlTC0CX3JmqNRwR5wsFAUzNK8bBKC4UGlAcKOKpBuOi
sjJZk/S2T4YxZCdVgOya+ByAnXoTg384y9KTgudq4wxjrS9e2zKwnH6l6h7UTXAE
iEK7AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAeD3g1QpzlFr8O71Ke2aa6HeKjYhc
nNe7Pb9TZrK8FK+XX0v2Anv5LXlYIQelFakc0oDSLbRoSpxZN7FDtXwiakhCbLT8
XM6+3lQpZDprxiCPh2CPYDpnvu1m8gbH2F0RONYYdJSdm+yiOTNoRLP+vCmvh7NK
K5bBZVhLE+a26IY=
-----END CERTIFICATE-----"

$hps |  Set-AciCommRsKeyRing -TnPkiKeyRingName keyring1024
```

 # Access APIC through https should now give no "untrusted connection" message.

4.31 Generic Managed Object Queries

 # Get Managed Object of a specific DN.

```
Get-AciManagedObject -Dn "uni/tn-common"
```

 # Get all Managed Objects of a particular class.

```
Get-AciManagedObject -ClassId fvTenant
```

 # Get DNs of Managed Objects of a particular class.

```
Get-AciManagedObject -ClassId fvTenant –DnList
```

 # Get names of all Service Profiles from org-root.

```
Get-AcifvTenant –name common| Get-AciManagedObject -ClassId fvAp | Select
Name
```

 # Get immediate children of fvTenant common

```
Get-AcifvTenant –name common | Get-AciChild
```

 # Get parent of a Managed Object.

```
Get-AcifvTenant –name common | Get-AciParent
```

4.32 Generic Managed Object Cmdlets

 # Create an Application profile using parent object.

```
$propMap = @{Name = "www.Acme.com"}


Get-AciFvTenant –name A| Add-AciManagedObject -ClassId fvAp -PropertyMap
$propMap
```

 # Create an application profile using parent object

```
$propMap = @{Name = "www.Acme.com”}
Add-AciManagedObject -ClassId fvAp -PropertyMap $propMap -Parent (Get-
AciFvTenant –name A)
```

 # Create an application profile using DN.

```
$propMap = @{Dn = "uni/tn-A/ap-www.Acme.com"; Name = "www.Acme.com”}
Add-AciManagedObject -ClassId fvAp -PropertyMap $propMap
```

 # Modify an application profile using Managed Object.

```
$apname = Get-AciFvAp -Name 'www.Acme.com'
$propMap = @{Descr=”This is testing”}
Set-AciManagedObject -PropertyMap $propMap -ManagedObject $apname
```

 # Modify an application profile using Dn

```
$propMap = @{Dn = "uni/tn-A/ap-www.Acme.com"; Descr=”This is testing”}
Set-AciManagedObject -PropertyMap $propMap -ClassId fvAp
```

 # Remove a Managed Object.

```
Get-AciFvAp –Name www.Acme.com | Remove-AciManagedObject
```

4.33 Generic Cmdlet -XmlTag

 # The XmlTag parameter enables us to work with unknown Managed Objects.

 # Create Tenant.

```
Add-AciManagedObject -XmlTag fvTenant -PropertyMap @{Dn="uni/tn-A"; Name="A";
Descr="This is testing";}
```

 # Change the description

```
Set-AciManagedObject -XmlTag fvTenant -PropertyMap @{Dn = "uni/tn-A";
Descr="Testing -xmlTag";}
```

4.34 XtraProperty in Get/Add/Set cmdlets

 # The XtraProperty parameter ensures that unknown Managed Object properties can also be
 # used in Get/Add/Set cmdlets.

 # Create an application profile with extra property ExtIPPoolName.

```
Get-AcifvTenant –name A | Add-AcifvAp -Name C -XtraProperty @{Descr = "This
is testing";}
```

 # Get all service profiles that have ext-mgmt as ExtIPPoolName.

```
Get-AcifvAp -XtraProperty @{Descr = "This is testing";}
```

4.35 CCO Integration - TBD

 # There are 2 Cmdlets related to CCO Image handling:
Get-ACICcoImageList – Get the list of images from CCO
Get-ACICcoImage – This will let user download a particular image from CCO

This is still under development.

4.36 Upload Firmware - TBD

Send-AciFormware cmdlet will be given to user to be able to upgrade ACI Fabric through cmdlet.

This is still under development.

4.37 Export to XML

Following cmdlets will be given to users to be able to import/export MOs as XML. This is under development.

 # Export the configuration of a Managed Object.

```
Export-AciXml -Dn uni/tn-A -children -LiteralPath C:\cmd.xml
```

 # Export the xml of a Managed Object into a file.

```
Get-AcifvAp -Name A | Export-AciMoXml –writeXml AllConfig | Out-File
c:\mo.xml
```

4.38 Import from XML

Following cmdlets will be given to users to be able to import/export MOs as XML. This is under development.

 # Import the configuration from the XML file.

```
Import-AciXml -LiteralPath C:\cmd.xml
```

 # Import xml of a Managed Object and convert it into objects

```
Import-AciMoXml -LiteralPath c:\mo.xml
```

4.39 Cmdlet Meta Information

 # Get Meta information about all Managed Object mapped cmdlets.

```
Get-AciCmdletMeta
```

 # Get Meta information about fvTenant mapped cmdlets.

```
Get-AciCmdletMeta -ClassId fvTenant
```

 # View the hierarchy of the fvTenant class.

```
Get-AciCmdletMeta -Noun fvTenant -Tree
```

 # Get Meta information for the fvTenant noun.

```
Get-AciCmdletMeta -Noun fvTenant
```

 # See the Managed Object information for fvTenant.

```
Get-AciCmdletMeta -ClassId fvTenant | Select -ExpandProperty MoMeta
```

 # See the Managed Object property information for fvTenant.
```
Get-AciCmdletMeta -ClassId fvTenant | Select -ExpandProperty MoMeta | Select
-ExpandProperty PropertyMeta
```

4.40 Compare-AciManagedObject - Dn Translation

 # Create a Application profile under Tenant A. Assume that Tenants A & B are in place already.
```
$srcTenant = Get-AcifvTenant -Name A
$destTenant = Get-AcifvTenant -Name B
$srcAp = Get-AcifvAp –fvTenant $srcTenant –Name apA
$destAp = Get-AcifvAp –fvTenant $destTenant –Name apB
```

#Make Changes into apA which are different then apB

 # Create a translation map with DNs of entities that needs to be translated.
```
$xlateDn = @{ }
$xlateDn['uni/tn-A/ap-apA'] = 'uni/tn-B/ap-apB'
```

 # Combine the translation map with Compare-AciMo to see the changes required.
```
Compare-AciManagedObject (Get-AcifvAp  -fvTenant $destTenant -Name apB) (Get-
AcifvAp -fvTenant $srcTenant -Name apA ) -XlateMap $xlateDn
```

 # Sync a service profile from org A to org B while renaming it.
```
Sync-AciManagedObject (Compare-AciManagedObject (Get-AcifvAp  -fvTenant
$destTenant -Name apB ) (Get-AcifvAp -fvTenant $srcTenant -Name apA) -
XlateMap $xlateDn) -Force | select Dn
```

4.41 Compare-AciManagedObject - GetPropertyDiff()

 # Use GetPropertyDiff() function on output of Compare- AciManagedObject to see the difference in properties.
```
$ap1 = Get-AcifvAp -Dn uni/tn-A/ap-apA
$ap2 = $ap1 | Set-AcifvAp -Descr 'GetPropertyDiff Example' -Force
$diff = Compare-AciManagedObject $ap1 $ap2
```

 # Display all the properties having difference
 # If $diff is an array of objects, then GetPropertyDiff works on $diff[<index>]
```
$diff.GetPropertyDiff()
```

 # For a specific property
```
$diff.GetPropertyDiff('descr')
```

 # Include all operational properties of MOs in comparison
```
Compare-AciManagedObject $ap1 $ap2 –IncludeOperational
```

4.42 Transaction Support

Transaction cmdlets let user group multiple cmdlets in one transaction and send it to ACI via one REST call.

#connect ACI

```
connect-aci 10.0.1.200
```

#Start Transaction

```
Start-AciTransaction
$tn = Add-AciFvTenant –name testAci –Descr “This is testing”
$ap = $tn | Add-AciFvAp –name ap
```

 # Undo above configuration

```
Undo-AciTransaction
```

#Status Transaction again

```
Start-AciTransaction
$tn = Add-AciFvTenant –name newtestAci –Descr “This is testing”
$ap = $tn | Add-AciFvAp –name ap
Get-AciFvTenant –name acitest | Remove-AciFvTenant
Add-AciaaaUser –name nehatest –firstName John –lastName Smith
```

 # complete transaction

```
Complete-AciTransaction
```

Please note following while using transaction commands.

- **Cmdlets within a single transaction can be in any order as long as parent and child hierarchy is maintained.**
- **GET cmdlets are only valid on existing configuration. Configuration which is newly being added in the same transaction cannot be queried using Get cmdlet in the same transaction.**
- **SET/REMOVE cmdlets are valid on existing as well as new configuration which is being added in the same transaction.**
- **Undo-transaction will undo all previously typed cmdlets**
- **If any cmdlet in transaction fails, nothing will get pushed to ACI.**