USER GUIDE

Cisco UCS C-Series PowerTool, Release 0.9.3

1 Overview

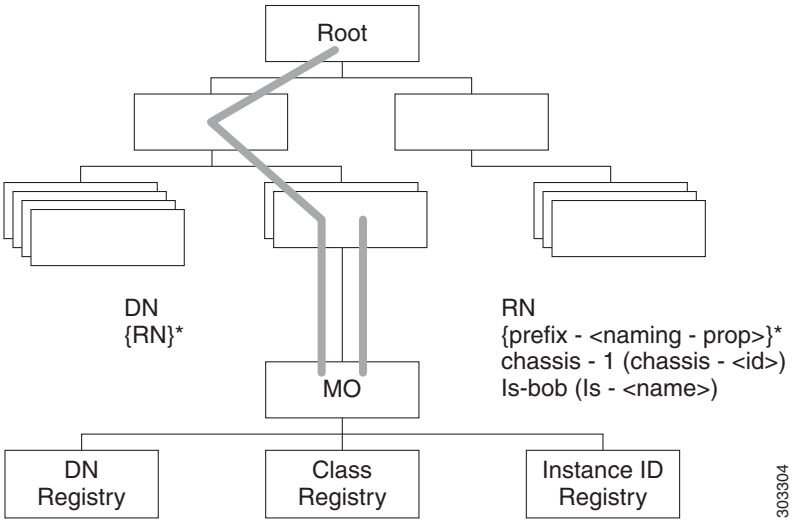
Cisco UCS C-Series PowerTool is a PowerShell module which helps automate all aspects of Cisco UCS C-Series Rack Servers. Cisco UCS C-Series PowerTool enables easy integration with existing IT management processes and tools.

The Cisco UCS C-Series PowerTool cmdlets work on the Cisco UCS CIMC Management Information Tree (MIT). The cmdlets allow you to create, modify, or delete actions on the Managed Objects (MOs) in the tree.

2 Management Information Model

All the physical and logical components that comprise a Cisco UCS C-Series Server are represented in a hierarchical Management Information Model (MIM), which is referred to as the Management Information Tree (MIT). Each node in the tree represents a Managed Object (MO), which is uniquely identified by its distinguished name (DN). See Figure 1.

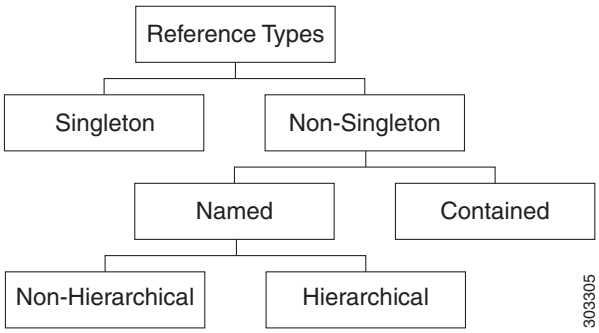
Figure 1 Management Information Model



Managed Objects

Managed Objects (MOs) (see Figure 2) are abstractions of Cisco UCS C-Series CIMC MIT resources, such as CPUs, DIMMs, adapter cards, fans, and power supply units. Managed Objects represent any physical or logical entity that is configured or managed in the Cisco UCS C-Series CIMC MIT. For example, physical entities such as CPUs, DIMMs, adapter cards and fans and logical entities such as users, communication services like HTTP, SSH are represented as MOs.

Figure 2 Managed Objects



Each MO is uniquely identified in the tree with its distinguished name (DN) and can be uniquely identified within the context of its parent with its relative name (RN). The DN identifies the place of the MO in the MIT. A DN is a concatenation of all the relative names that start from the root to the MO itself. Essentially, DN = [RN]/[RN]/[RN]/…/[RN].

In the following example, DN provides a fully qualified name for adapter-1 in the model.

```
< dn = "sys/rack-unit-1/adaptor-1" />
```

The above written DN is composed of the following RN:

```
topSystem MO: rn="sys" computeRackUnit MO: rn="rack-unit-1" adaptorUnit MO: rn ="adaptor-<id>"
```

A relative name (RN) might have a value of one or more of the MO properties embedded in it. This allows you to differentiate multiple MOs of the same type within the context of the parent. Any properties that form part of the RN as described earlier are referred to as naming properties.

For instance, adaptor MOs reside under a rack unit MO. The adaptor MO contains the adaptor identifier as part of its Rn(adaptor-[Id]), which uniquely identifies each adaptor MO in the context of a rack unit.

Methods

Methods are Cisco UCS XML APIs used to manage and monitor the system. The following methods are supported:

- Authentication
- aaaLogin—Initial method for logging in.
- aaaRefresh—Refreshes the current authentication cookie.
- aaaLogout—Exits the current session and deactivates the corresponding authentication cookie.
- configResolveDn—Retrieves objects by DN.
- configResolveClass—Retrieves objects of a given class.
- configResolveChildren—Retrieves the child objects of an object.
- configResolveParent—Retrieves the parent object of an object.
- configConfMo—Affects a single managed object (for example, a DN).
- eventSubscribe—To register for events

Cisco UCS C-Series PowerTool Mapping

All but about 10 of the Cisco UCS C-Series PowerTool cmdlets are generated from the MO specification. A noun is used in place of the type (Fan instead of EquipmentFan and so on). 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.

Add Cmdlet—Uses the ConfigConfMo method with the MO status “created” with the specified property values. If the Force parameter is specified, there is no prompt for confirmation.

Get Cmdlet—Uses the ConfigResolveClass method to retrieve MOs. XML API of Cisco UCS C-Series Rack Mount servers do not support any filters. So if any property parameters are specified, PowerTool gets all the instances of the specified class and filters them on the client side using the property values.

Set Cmdlet—Uses the ConfigConfMo method with MO status “modified” with the specified property values. If the Force parameter is specified, there is no prompt for confirmation.

Remove Cmdlet—Uses the ConfigConfMo method with the MO status “deleted”. If the Force parameter is specified, there is no prompt for confirmation.

This table 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

This table 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

This table lists the methods invoked to generate the required XML requests:

Cmdlet	Method
Add-UcsCSeries[1] Set-UcsCSeries[1]	ConfigConfMo
Get-UcsCSeries[1]	ConfigResolveClass with client side filters
Get-UcsCSeriesManagedObject -ClassId	ConfigResolveClass
Get-UcsCSeriesManagedObject –ClassId -Dnlist	ConfigResolveClass (The output is then filtered for the matching Dns)
Get-UcsCSeriesManagedObject –Dn	ConfigResolveDn
Connect-UcsCSeries	AaaLogin
Disconnect-UcsCSeries	AaaLogout
Background[1]	AaaRefresh
Get-UcsChild	ConfigResolveChildren

1. This is not a cmdlet. It is a background service.

Get-UcsCSeriesCmdletMeta is a 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 and so on), and the version of Cisco UCS CIMC that the property was introduced in.

3 Installation

Supports CIMC Version 1.5 or later.

Before You Begin

- Ensure that you have PowerShell v2.0 or above installed in your system.
- Close any instances of PowerShell running with the Cisco UCS C-Series PowerTool module loaded.

Installation

Step 1 Download and launch the installer.

Step 2 (Optional) Choose **Create Shortcut** to add a shortcut on the desktop.

Getting Started

Step 1 From the desktop shortcut, launch **Cisco UCS C-Series PowerTool**.

Step 2 View all cmdlets, functions, and aliases supported by Cisco UCS C-Series PowerTool.

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

Step 3 Connect to a Cisco UCS C-Series Server.

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

Note After logging on, by default, the Cisco UCS handle is added to the default Cisco UCS C-Series Server list, unless the `-NotDefault` option is specified. Every cmdlet that operates on a Cisco UCS C-Series Server takes the `–UcsCSeries` parameter, where the handle can be specified.

Step 4 Connect to a Cisco UCS C-Series Server using a 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-UcsCSeries <ip or hostname> -Proxy
$proxy
```

Step 5 Use the following cmdlets:

a. Get the consolidated status information from the Cisco UCS C-Series Servers.

```
Get-UcsCSeriesStatus -UcsCSeries $handle
```

b. Get the inventory summary of the C-Series Unit.

```
Get-UcsCSeriesRackUnit -UcsCSeries $handle
```

c. Disconnect.

```
Disconnect-UcsCSeries –UcsCSeries $handle
```

Default Cisco UCS

If no handle or name is specified, the Cisco UCS C-Series Server handle is added to a *DefaultUcsCSeries* server list unless the `–UcsCSeries` parameter is specified. The first cmdlet in the pipeline operates on the default UCS list.

Connect to UCS C-Series rack server

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

Get the default rack server.

```
Get-UcsCSeriesPSSession
```

Get the status information and CIMC version of the rack server.

```
Get-UcsCSeriesStatus
```

Get information about the physical unit of the rack server.

```
Get-UcsCSeriesRackUnit
```

Enable HTTP on the rack server.

```
Get-UcsCSeriesHttp | Set-UcsCSeriesHttp -AdminState enabled
```

Disable HTTP on the rack server.

```
Get-UcsCSeriesHttp | Set-UcsCSeriesHttp -AdminState disabled
```

Disconnect the rack server.

```
Disconnect-UcsCSeries
```

Default UCS List with Multiple UCS

Cisco UCS C-Series PowerTool cmdlets can work with multiple Cisco UCS C-Series Rack Servers if you specify multiple handles.

Connect to a Cisco UCS C-Series Server.

```
$handle1 = Connect-UcsCSeries <ip1> -NotDefault
$handle2 = Connect-UcsCSeries <ip2> -NotDefault
Get-UcsCSeriesStatus -UcsCSeries $handle1,$handle2
Disconnect-UcsCSeries -UcsCSeries $handle1,$handle2
```

By default, multiple Cisco UCS C-Series handles are not allowed in DefaultUcsCSeries. You can override this restriction by using the `Set-UcsPowerToolConfiguration` cmdlet.

```
Get-UcsCSeriesPowerToolConfiguration
Set-UcsCSeriesPowerToolConfiguration -SupportMultipleDefaultUcsCSeries $true
Connect-UcsCSeries <ip1>
Connect-UcsCSeries <ip2>
Get-UcsCSeriesStatus
Disconnect-UcsCSeries
```

Connect to multiple Cisco UCS C-Series Servers by using the same login credentials.

```
$user = "<username>"
$password = "<password>" |
 ConvertTo-SecureString -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential($user, $password)
$servers = @("<UcsCSeries1>", "<UcsCSeries2>", "<UcsCSeries3>")
Connect-UcsCSeries $servers -Credential $cred
```

Credentials to/from a File

```
Connect-UcsCSeries <ip1>
Connect-UcsCSeries <ip2>
```

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

```
Export-UcsCSeriesPSSession -LiteralPath C:\work\labs.xml
Disconnect-UcsCSeries
```

A login can be initiated from credentials stored in a file.

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

Log in to an additional system and add the credentials to the file.

```
Connect-UcsCSeries <ip3>
Export-UcsCSeriesPSSession -Path C:\work\lab?.xml -Merge
```

SSL Handling

When a user connects to a Cisco UCS C-Series Server and the server cannot recognize any valid certificates, the connection establishment depends on InvalidCertificateAction. InvalidCertificateAction is set to Ignore by default. By default Cisco UCS C-Series PowerTool is configured to establish the connection without taking into account if the certificate is invalid.

You can override this setting by using the `Set-UcsCSeriesPowerToolConfiguration` cmdlet.

```
Get-UcsCSeriesPowerToolConfiguration
Set-UcsCSeriesPowerToolConfiguration -InvalidCertificateAction Fail
```

This table describes the options to check the validity of the certificate.

	Description
`Fail`	The cmdlet does not establish connection if the certificate is not valid.
`Ignore`	The cmdlet establishes a connection without taking into account that the certificate is invalid.
`Default`	(Windows default) The cmdlet establishes a connection if the certificate is valid.

Aliases

Some aliases have been defined for convenience.

```
PS C:\> gal | ? {$_.Name -like "*-Ucs*" } | select Name
Name
----
Add-UcsCSeriesMo
```

4 Examples

The following examples show how to execute the cmdlets:

- Add User
- Enable IP Blocking
- Configure NTP Settings
- Modify Syslog Settings
- Configure SoL
- Set Boot Order
- vMedia Configuration
- Power Settings
- Get Adapter and Controller Information
- Transaction Support
- PowerTool Cmdlet Generation
- Managed Object Synchronization
- Filters

Add User

```
Get-UcsCSeriesLocalUser - Id 9 | Set-UcsCSeriesLocalUser-Name “admin” -pwd “Password -AccountStatus “active” -Priv “admin”
```

Enable IP Blocking

```
Get-UcsCSeriesIpBlocking | Set-UcsCSeriesIpBlocking -Enable “yes”
```

Configure NTP Settings

```
Get-UcsCSeriesNtpServer | Set-UcsCSeriesNtpServer -NtpEnable “yes” -NtpServer1 1.1.1.1 -Force
```

Modify Syslog Settings

```
Get-UcsCSeriesSyslog | Set-UcsCSeriesSyslog -LocalSeverity warning -RemoteSeverity debug -Force
```

Configure SoL

```
Get-UcsCSeriesSolif -Dn "sys/CSeries-unit-1/sol-if | Set-UcsCSeriesSolIf -AdminState "enable" -Speed "57600"
-Force
```

Set Boot Order

```
Get-UcsCSeriesLsbootStorage | Set-UcsCSeriesLsbootStorage -Order 2 -Force
```

vMedia Configuration

```
Get-UcsCSeriesCommVMedia | Set-UcsCSeriesCommVMedia -AdminState "enabled" -EncryptionState "enabled" -Force
```

Power Settings

 # Server Power off

```
Get-UcsCSeriesRackUnit | Set-UcsCSeriesRackUnit -AdminPower "down" -Force
```

 # Server Power on

```
Get-UcsCSeriesRackUnit | Set-UcsCSeriesRackUnit -AdminPower "up" -Force
```

 # Server Soft Shutdown

```
Get-UcsCSeriesRackUnit | Set-UcsCSeriesRackUnit -AdminPower "soft-shut-down" -Force
```

 # Server Power Cycle

```
Get-UcsCSeriesRackUnit | Set-UcsCSeriesRackUnit -AdminPower "cycle-immediate" -Force
```

 # Server Hard Reset

```
Get-UcsCSeriesRackUnit | Set-UcsCSeriesRackUnit -AdminPower "hard-reset-immediate" -Force
```

Get Adapter and Controller Information

 # PCI Adapter Properties

```
Get-UcsCSeriesPciEquipSlot -Id "1"
```

 # Network Adapter Information

```
Get-UcsCSeriesNetworkAdapterEthIf -Dn "sys/CSeries-unit-1/network-adapter-L/eth-1"
```

 # Storage Controller Information

```
Get-UcsCSeriesStorageController "-Dn "sys/CSeries-unit-1/board/storage-SAS-SLOT-4"
```

Transaction Support

Transaction Support

 # Start a transaction.

```
Start-UcsTransaction
```

 # Perform an operation.

```
$adaptorHostEthIf = Get-UcsCSeriesAdaptorUnit | Add-UcsCSeriesAdaptorHostEthIf -Name AdaptorHostEth
$adaptorHostEthIfModify = $adaptorHostEthIf | Set-UcsCSeriesAdaptorHostEthIf -PxeBoot enabled -Force
$adaptorEthISCSIProfile = $adaptorHostEthIfModify | Add-UcsCSeriesAdaptorEthISCSIProfile -InitiatorName
AdaptorHostEth -InitiatorIPAddress  xx.xx.xx.xx  -InitiatorSubnetMask 255.255.255.0 -DhcpISCSI enabled
$adaptorEthISCSIProfile | Remove-UcsCSeriesAdaptorEthISCSIProfile -Force
$adaptorHostEthIfModify | Remove-UcsCSeriesAdaptorHostEthIf -Force
```

 # End a transaction.

```
Complete-UcsTransaction
```

 # Undo a transaction.

```
Undo-UcsTransaction
```

PowerTool Cmdlet Generation

 # Generate cmdlets for the specified Managed Object.

```
Get-UcsCSeriesLsbootVirtualMedia | ConvertTo-UcsCSeriesCmdlet
```

 # Generate cmdlets for the specified Managed Object with -Hierarchy.

```
Get-UcsCSeriesBootDefinition -Hierarchy | ConvertTo-UcsCSeriesCmdlet
```

Managed Object Synchronization

 # Enable SupportMultipleDefaultUcsCSeries to connect to multiple CIMC.

```
Set-UcsCSeriesPowerToolConfiguration -SupportMultipleDefaultUcsCSeries $true
```

 # Get credential and store it in a variable.

```
$secpasswd = ConvertTo-SecureString password -AsPlainText -Force
$mycreds = New-Object System.Management.Automation.PSCredential ("admin",$secpasswd)
```

 # Connect to different CIMC.

```
$cimc1 = Connect-UcsCSeries  xx.xx.xx.xx -Credential $mycreds
$cimc2 = Connect-UcsCSeries xx.xx.xx.xx -Credential $mycreds
```

 # Get local user from different CIMC.

```
$user1 = Get-UcsCSeriesLocalUser -UcsCSeries $cimc1 -Id 1
$user2 = Get-UcsCSeriesLocalUser -UcsCSeries $cimc2 -Id 1
```

 # Synchronize a set of MOs from CIMC2 to CIMC1.

```
Compare-UcsCSeriesManagedObject $user1 $user2
Sync-UcsCSeriesManagedObject (Compare-UcsCSeriesManagedObject $user1 $user2) -UcsCSeries $cimc1
```

Filters

 # Get SysdebugMEpLog managed object where Type equals to “SEL” or Type equals to “Syslog”.

```
Get-UcsCSeriesRackUnit | Get-UcsCSeriesMgmtController | Get-UcsCSeriesSysdebugMEpLog -Filter '(type -ilike SEL)
-or (Type -clike Syslog)'
```

 # Get SysdebugMEpLog managed object where (Type equals to “SEL” or Type equals to #”Syslog”) and Id equals to “0” and Type equals to “SEL”.

```
Get-UcsCSeriesRackUnit  | Get-UcsCSeriesMgmtController | Get-UcsCSeriesSysdebugMEpLog -Filter '(type -ilike SEL)
-or (Type -clike Syslog)' -Id 0 -Type SEL
```

 # Get local user where name is like “admin” (case sensitive).

```
Get-UcsCSeriesManagedObject -ClassId aaaUser -Filter 'Name -clike admin'
```

 # Get User where name is like “test*” (support * regular expression/case sensitive).

```
Get-UcsCSeriesManagedObject -ClassId aaaUser -Filter 'Name -clike test*'
```

 # Get local user where AccountStatus is not equals to inactive.

```
Get-UcsCSeriesManagedObject -ClassId aaaUser -Filter 'AccountStatus -cne inactive'
```

 # Get local user where AccountStatus matches ‘inacti’.

```
Get-UcsCSeriesManagedObject -ClassId aaaUser -Filter 'AccountStatus -cmatch inacti'
```

 # Get local user where AccountStatus matches with ‘active’ (starts with active/case sensitive).

```
Get-UcsCSeriesManagedObject -ClassId aaaUser -Filter 'AccountStatus -cmatch ^active'
```

 # Get local user where AccountStatus does not matches with ‘active’ (starts with active/case sensitive).

```
Get-UcsCSeriesManagedObject -ClassId aaaUser -Filter 'AccountStatus -cnotmatch ^active'
```

 # Get local user where Accountstatus is not like ‘active’ (starts with active/case sensitive).

```
Get-UcsCSeriesManagedObject -ClassId aaaUser -Filter 'AccountStatus -cnotlike active'
```

5 Samples

Sample scripts developed using Cisco UCS C-Series PowerTool will be available in communities.cisco.com shortly.

6 Related Cisco UCS Documentation and Documentation Feedback

For more information, you can access related documents from the following links:

- Cisco UCS C-Series Documentation Roadmap:
  http://www.cisco.com/en/US/docs/unified_computing/ucs/overview/guide/UCS_rack_roadmap.html
- Cisco UCS C-Series CIMC XML API Guide:
  http://www.cisco.com/en/US/docs/unified_computing/ucs/c/sw/api/b_cimc_api_book.html

To provide technical feedback on this document, or to report an error or omission, please send your comments to ucs-docfeedback@external.cisco.com. We appreciate your feedback.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see *What’s New in Cisco Product Documentation* at: http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html.

Subscribe to *What’s New in Cisco Product Documentation*, which lists all new and revised Cisco technical documentation, as an RSS feed and deliver content directly to your desktop using a reader application. The RSS feeds are a free service.
