Examples

This chapter contains the following sections:

Activate Cisco IMC Firmware

Activate the Cisco IMC firmware, using the following cmdlet:

```
Get-ImcFirmwareBootDefinition -Type "blade-controller" |
Get-ImcFirmwareBootUnit | Set-ImcFirmwareBootUnit-AdminState trigger -Image backup
-ResetOnActivate yes -Force
```

Add User

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

Note Clear-ImcLocalUser changes the status to inactive and does not remove the user or data.

Cisco IMC Desired State Configuration (DSC)

Desired State Configuration (DSC) is a new approach for configuring local and remote machines. You can use IMC DSC resources to configure multiple IMC in a datacenter from a centralized root server. PowerTool module Cisco.UCS.DesiredStateConfiguration contains all the custom IMC DSC resources.

```
Get-Module Cisco.UCS.DesiredStateConfiguration -ListAvailable
Get-DscResource | where{$_.Module -ilike 'Cisco*'
-and $_.Name -ilike 'imc*'} | Select Name
```

A DSC resource can execute in parallel, and maximum number of XML API connections on any Cisco IMC is limited to 4. So, specify add DependsOn property to each IMC DSC resource in such cases.

ImcManagedObject Resource

The ImcManagedObject resource is part of the Cisco.UCS.DesiredStateConfiguration module. It provides a mechanism to configure a Cisco IMC Managed Object (MO) by specifying the details of the MO on multiple Cisco IMC servers using a DSC framework.

Syntax

```
ImcManagedObject [string] #ResourceName
{
Dn = [string]
Identifier = [string]
ImcConnectionString = [string]
ImcCredentials = [PSCredential]
[ Action = [string] { Add | Set } ]
[ ClassId = [string] ]
[ DependsOn = [string[]] ]
[ Ensure = [string] { Absent | Present } ]
[ PropertyMap = [string] ]
[ WebProxyCredentials = [PSCredential] ]
}
```

Property	Description
Dn	Specifies the Dn of a managed object.
Identifier	Specifies the unique id for the DSC resource.

ImcConnectionString	Specifies the connection string for an IMC server.
	Syntax:
	`Name=<ipAddress>`
	`` [`nNoSsl=<bool>][`nPort=<ushort>] ``
	`` [`nProxyAddress=<proxyAddress>] ``
	`` [`nUseProxyDefaultCredentials=<bool>] ``

ImcCredentials	Indicates the credentials required to access IMC
Action	Specifies the action you want to perform on a managed object. Set this property to **Add** for adding a managed object. Set it to **Set** for modifying an existing managed object.
ClassId	Specifies the class id of a managed object.
DependsOn	Indicates that the configuration of another resource must run before this resource is configured. For example, the first ID of the resource configuration script block that you want to run is ResourceName and its type is ResourceType. The syntax for using this property is: `DependsOn = "[ResourceType]ResourceName"`

Property	Description
Ensure	Indicates if a managed object exists. Set this property to **Absent** to ensure that the managed object does not exist. Set to **Present** to ensure that the managed object does exist. The default is Present.
`PropertyMap`	Specifies the properties of a managed object as keyValue pairs. Syntax: `` `<key1>=<value1> `<key2>=<value2> ``
WebProxyCredentials	Indicates the credentials for a web proxy.

Example

The following example shows how to use the ImcManagebObject resource to add a Managed Object with Dn "sys/rack-unit-1/boot-policy/efi-read-only".

Use, Action="Set" to edit an existing MO.

Configuration ImcManagedObjectResourceDemo

```
{
param(
[Parameter(Mandatory=$true)]
[PsCredential] $imcCredential,

[Parameter(Mandatory=$true)]
[string] $connectionString
)
Import-DSCResource -ModuleName Cisco.Ucs.DesiredStateConfiguration
Node "localhost"
{
ImcManagedObject ResourceInstance
{
Ensure = "Present"
ClassId= "lsbootEfi"
Dn = "sys/rack-unit-1/boot-policy/efi-read-only"
PropertyMap = "Access = read-only `nType = efi `nOrder = 4"
ImcCredentials = $imcCredential
ImcConnectionString = $connectionString
Identifier = "2"
}

}

}
```

ImcScript Resource

ImcScript resource in a Cisco.Ucs.DesiredStateConfiguration module provides a mechanism to execute IMC PowerTool cmdlets.

Syntax

ImcScript [string] #ResourceName

```
{
Dn = [string]
Identifier = [string]
ImcConnectionString = [string]
ImcCredentials = [PSCredential]
Script = [string]
[ Action = [string] { Add | Set } ]
[ DependsOn = [string[]] ]
[ Ensure = [string] { Absent | Present } ]
[ WebProxyCredentials = [PSCredential] ]
}
```

Property	Description
Dn	Specifies Dn of a managed object.
Identifier	Specifies the unique id for the DSC resource.
Script	Specifies set of PowerTool cmdlets. Use *`n* as new cmdlet prefix.

ImcConnectionString	Specifies the connection string for an IMC server.
	Syntax:
	`Name=<ipAddress>`
	``[`nNoSsl=<bool>][`nPort=<ushort>]``
	``[`nProxyAddress=<proxyAddress>]``
	``[`nUseProxyDefaultCredentials=<bool>]``

ImcCredentials	Indicates the credentials required to access an IMC server.
Action	Specifies the action you want to perform on a managed object. Set this property Add for adding a managed object. Set it to Set to modify an existing managed object.
DependsOn	Indicates that the configuration of another resource must run before this resource is configured. For example, if the ID of the resource configuration script block that you want to run first is ResourceName and its type is ResourceType. The syntax for using this property is: `DependsOn = "[ResourceType]ResourceName"`
Ensure	Indicates if Script executes or not. The default is Present.
WebProxyCredentials	Indicates the credentials for a web proxy.
WebProxyCredentials	Indicates the credentials for a web proxy.

Syntax

```
Configuration ImcScriptResourceDemo
{
param(
[Parameter(Mandatory=$true)]
[PsCredential] $imcCredential,

[Parameter(Mandatory=$true)]
[string] $connectionString
)
Import-DSCResource -ModuleName Cisco.Ucs.DesiredStateConfiguration


Node "localhost"
{
ImcScript ResourceInstance
{
Ensure = "Present"
Dn = "sys/svc-ext/snmp-svc/snmpv3-user-9"
Script= "Clear-ImcSnmpUser -id 2 -force
`n Add-ImcSnmpUser -Id 9 -Name 'testuser'
-Auth MD5 -AuthPwd password1 -Privacy AES
-PrivacyPwd password2 -SecurityLevel authpriv
`n Clear-ImcSnmpUser -id 2 -force "
ImcCredentials = $imcCredential
ImcConnectionString = $connectionString
Identifier = "2"

} }
}
```

Cisco IMC Firmware Update

Create a user credential, using the following cmdlet:

```
$user = "<username>"
$password = "<password>"
$cred = New-Object System.Management.Automation.PSCredential($user,$password)
```

Update Cisco IMC Firmware, using the following cmdlet:

```
Get-ImcFirmwareUpdatable -Type blade-controller | Set-ImcFirmwareUpdatable -AdminState
trigger -Type blade-controller -Protocol ftp -RemoteServer "10.65.183.111" -RemotePath
"/UcseBin/UCSE_CIMC_2.3.1.bin"-RemoteCredential $cred-Force
```

Clear a Boot Drive

To clear the boot drive, use the following cmdlet:

```
Get-ImcStorageController | Set-ImcStorageController -AdminAction "clear-boot-drive" -Force
```

Configure NTP Settings

Configure the NTP settings, using the following cmdlet:

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

Confirm Flag

When Confirm - Switch parameter in a PowerTool cmdlet is specified, you are prompted to confirm the changes. Cmdlet sends a request to confirm the changes applied to the system which is outside of the Windows PowerShell environment. For example, if a cmdlet is executed to clear an SNMP user, the cmdlet requires confirmation from the user to complete the action.

Syntax

```
Get-ImcSnmpUser -Name snmpuser | Clear-ImcSnmpUser -Confirm
Confirm
Are you sure you want to perform this action?
Performing the operation "Clear-ImcSnmpUser" on target "Clear".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

Configure SoL

Configure the SoL, using the following cmdlet:

```
Get-ImcSolif -Dn “sys/rack-unit-1/sol-if” | Set-ImcSolIf -AdminState “enable” -Speed “57600”
 -Force
```

Create a Virtual Drive

Create a virtual drive using unused physical drive.

```
Get-ImcStorageVirtualDriveCreatorUsingUnusedPhysicalDrive |
Set-ImcStorageVirtualDriveCreatorUsingUnusedPhysicalDrive
-AdminState trigger -size "400 MB" -DriveGroup "[2]" -RaidLevel 0 -VirtualDriveName "vd_111"
 -Force
```

Create a virtual drive using a virtual drive group

```
Get-ImcStorageVirtualDriveCreatorUsingVirtualDriveGroup |
Set-ImcStorageVirtualDriveCreatorUsingVirtualDriveGroup
-AdminState trigger -VirtualDriveName "vd_New"-SharedVirtualDriveId "3" -Size "100 MB"
-Force
```

Create a virtual drive from multiple drives

```
Get-ImcStorageController |
Set-ImcStorageVirtualDriveCreatorUsingUnusedPhysicalDrive
-AdminState trigger -DriveGroup "[1,2]" -RaidLevel 1 -Size "285148 MB" -VirtualDriveName
"RAID1_12" -WritePolicy "Always Write Back" -Force
```

Disable Drive Security

Disables the controller lock key depending on its current state on the disk.

Note On disabling the drive security, the data on all secure drives becomes unusable.

```
Get-ImcStorageController | Disable-ImcDriveSecurity -Force
Get-ImcSelfEncryptStorageController | Disable-ImcDriveSecurity -Force
```

Enable Drive Security

Enables the controller lock key depending on its current state on the disk.

```
Get-ImcStorageController | Enable-ImcDriveSecurity -KeyId "myKey123" -SecurityKey "myPass123"
 -Force
```

Enable-ImcPidCatalog

Enables the uploaded PID catalogue on the IMC server.

Syntax

```
Get-ImcPidCatalog | Enable-ImcPidCatalog -Force
```

Enable IP Blocking

Enable IP blocking, using the following cmdlet:

```
Get-ImcIpBlocking | Set-ImcIpBlocking -Enable "yes"
```

Export-ImcHardwareInventory

The **Export-ImcHardwareInventory** cmdlet exports the hardware inventory of the system to a remote location. You can also specify the remote server details, such as IP/HostName, protocol, path and filename, username and password, if any.

Syntax

```
Export-ImcHardwareInventory -Chassis <EquipmentChassis> -Hostname <string> [-Proto <string>]
 [-Pwd <string>]
```

```
-RemoteFile <string> [-User <string>] [-XtraProperty <Hashtable>] [-Force]
[<CommonParameters>]

Export-ImcHardwareInventory -TopSystem <TopSystem> -Hostname <string> [-Proto <string>]
[-Pwd <string>] -RemoteFile <string> [-User <string>] [-XtraProperty <Hashtable>] [-Force]
 [<CommonParameters>]
```

Example

```
Get-ImcTopSystem | Export-ImcHardwareInventory -Hostname "10.10.10.10" -Proto scp -User
root
-Pwd <password> -RemoteFile "/root/test/InventoryExportReport.txt" -Force
```

Filters

# Get SysdebugMEpLog managed object, where Type equals to "SEL" or "Syslog".

```
Get-ImcRackUnit | Get-ImcMgmtController | Get-ImcSysdebugMEpLog -Filter '(type -ilike SEL)
 -or (Type -clike Syslog)'
```

# Get SysdebugMEpLog managed object, where Type equals to "SEL" or #"Syslog", and Id equals to "0".

```
Get-ImcRackUnit | Get-ImcMgmtController | Get-ImcSysdebugMEpLog -Filter '(type -ilike SEL)
 -or (Type -clike Syslog)' -Id 0 -Type SEL
```

# Get a local user, where a name can be "admin" (case sensitive).

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

# Get User, where a name can be"test*" (support * regular expression or case sensitive).

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

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

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

# Get a local user, where AccountStatus matches 'inacti'.

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

# Get a local user, where AccountStatus matches with 'active' (starts with active or case sensitive).

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

# Get a local user, where AccountStatus does not match 'active' (starts with active or case sensitive).

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

# Get a local user, where Accountstatus is not 'active' (starts with active or case sensitive).

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

Force Flag

All the set and remove cmdlets in PowerTool, prompt for a confirmation, you can skip this confirmation by using -Force flag.

Syntax

```
Get-ImcSnmpUser -Name snmpuser | Clear-ImcSnmpUser -Force
```

Get Adapter and Controller Information

# PCI Adapter Properties

```
Get-ImcPciEquipSlot -Id "1"
```

# Network Adapter Information

```
Get-ImcNetworkAdapterEthIf -Dn "sys/rack-unit-1/network-adapter-L/eth-1"
```

# Storage Controller Information

```
Get-ImcStorageController -Dn "sys/rack-unit-1/board/storage-SAS-SLOT-4"
```

Get-ImcKmipDownloadStatus

The **Get-ImcKmipDownloadStatus** cmdlet provides an option to get the download status of a KMIP entity like Root CA Certificate, Client Certificate, and Client Private Key.

Syntax

```
Get-ImcKmipDownloadStatus [-Type <string>] [-XtraProperty <Hashtable>] [<CommonParameters>]
```

Example

```
Get-ImcKmipDownloadStatus
Get-ImcKmipDownloadStatus -Type RootCACertificate
Get-ImcKmipDownloadStatus -Type ClientCertificate
Get-ImcKmipDownloadStatus -Type ClientPrivateKey
```

Get-ImcKmipUploadStatus

The **Get-ImcKmipUploadStatus** cmdlet provides an option to get the upload status of a KMIP entity like Root CA Certificate, Client Certificate, and Client Private Key.

Syntax

```
Get-ImcKmipUploadStatus [-Type <string>] [-XtraProperty <Hashtable>] [<CommonParameters>]
```

Example

```
Get-ImcKmipUploadStatus
Get-ImcKmipUploadStatus -Type RootCACertificate
Get-ImcKmipUploadStatus -Type ClientCertificate
Get-ImcKmipUploadStatus -Type ClientPrivateKey
```

HUU Firmware Update

Create a user credential, using the following cmdlet:

```
$user = "<username>"
$password = "<password>"
$cred = New-Object System.Management.Automation.PSCredential($user,$password)
```

Update HUU firmware, using the following cmdlet:

```
Set-ImcHuuFirmwareUpdater -AdminState trigger -MapType nfs -RemoteIp 10.105.219.83
-RemoteCredential $cred-RemoteShare "/huuIso/ucs-c2x-huu-2.0.3d-1.iso" -StopOnError yes
-TimeOut 60 -UpdateComponent All-VerifyUpdate no -force -Xml
```

HUU Firmware Update through SD Card

NFS Mapping:

```
Get-ImcStorageFlexUtilVirtualDriveImageMap -VirtualDrive "HUU" |
Set-ImcStorageFlexUtilVirtualDriveImageMap -AdminAction map -Map nfs -RemoteShare
"x.x.x.x:/nfsShareLocation"
-RemoteFile "ucs-c240m5-huu-3.1.3a.iso" -MountOptions "nolock" -Force
```

Update the mapped image to the HUU partition from specified mount location:

```
Get-ImcStorageFlexUtilVirtualDrive -PartitionName HUU |
Set-ImcStorageFlexUtilVirtualDrive -AdminAction update-vd -Force
```

Update status can be found using the below query:

```
Get-ImcStorageFlexUtilVirtualDrive -PartitionName HUU |
select OperationInProgress, LastOperationStatus,HostAccessible
```

Note `OperationInProgress:` value should be `Update-Success`

Request to enable the virtual drive which would make the partition visible to the host:

```
Get-ImcStorageFlexUtilVirtualDrive -PartitionName HUU |
Set-ImcStorageFlexUtilVirtualDrive -AdminAction enable-vd -Force
```

Note `HostAccessible`: Value should be `Connected`

Get the LUN ID to set the boot order:

```
Get-ImcStorageFlexUtilVirtualDrive -PartitionName HUU | select LunId
```

Set the boot order to boot from flex-util HUU partition based on LUN ID:

```
Get-ImcLsbootSd | set-ImcLsbootSd -Lun <lunId selected in above cmdlet>
-Order 1 -State enabled -Subtype flex-util -Force
Get-ImcLsbootDevPrecision | Set-ImcLsbootDevPrecision -RebootOnUpdate yes
```

Start HUU Firmware update process:

```
$user = "testUser"
$password = "testPassword" | ConvertTo-SecureString -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential($user,$password)
Set-ImcHuuFirmwareUpdater -AdminState trigger -MapType nfs -RemoteIp "NA" -RemoteCredential
 $cred -RemoteShare "NA"
-StopOnError yes -TimeOut 120 -UpdateComponent All -VerifyUpdate no -BootMedium "microsd"
-Force
```

Modify Drive Security Information

Update security key/keyId for a drive security MO, using the following cmdlet:

```
Get-ImcStorageController |Set-ImcDriveSecurity -KeyId “newkey” -KeyManagement local
-SecurityKey “password4321”
-ExistingSecurityKey “myPass123” -Force
```

Managed Object Synchronization

# Enable SupportMultipleDefaultUcs to connect to multiple Cisco IMC, using the following cmdlet:

```
Set-UcsPowerToolConfiguration -SupportMultipleDefaultUcs $true
```

# Get the credential and store it in a variable, using the following cmdlet:

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

# Connect to different Cisco IMC, using the following cmdlet:

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

# Get a local user from different Cisco IMC, using the following cmdlet:

```
$user1 = Get-ImcLocalUser -Imc $cimc1 -Id 1
$user2 = Get-ImcLocalUser -Imc $cimc2 -Id 1
```

# Synchronize a set of MOs from Cisco IMC2 to Cisco IMC1, using the following cmdlet:

```
Compare-ImcManagedObject $user1 $user2
Sync-ImcManagedObject (Compare-ImcManagedObject $user1 $user2) -Imc $cimc1
```

Modify Syslog Settings

Modify the syslog settings, using the following cmdlet:

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

New Signing Certificate Request

Generate a certificate signing request (CSR) to obtain a new certificate. You can upload the new certificate to the Cisco IMC to replace the current server certificate. A public Certificate Authority (CA), such as VeriSign, or by your own certificate authority certifies the server. The generated certificate key length is 2048 bits.

```
New-ImcCertificateSigningRequest -CommonName “CSR2” -CountryCode India -Locality “GG6”
-Organization “cisco” -OrganizationalUnit “Tpidev” -Protocol ftp -State “Haryana” -RemoteFile
 “ImcCertificate.txt” -RemoteServer 10.105.219.xx -User administrator -Pwd *****
```

PowerTool Cmdlet Generation

ConvertTo-ImcCmdlet:

Cisco IMC GUI does not support XML logging. To generate the ConvertTo-ImcCmdlet cmdlets, rely on the output of the Get cmdlet and generate cmdlets to replicate the same object hierarchy.

Generate cmdlets for the specified MOs.

```
Get-ImcBiosSettings -Hierarchy | ConvertTo-ImcCmdlet
```

Save the cmdlet output in a file.

```
Get-ImcBiosSettings -Hierarchy | ConvertTo-ImcCmdlet -OutputPath "C:/OutputFile.txt"
```

Receive Certificate for IMC

Gets the information of current certificate available on the Cisco IMC server.

```
Receive-ImcCertificate
```

Receive-ImcKmipEntity

The **Receive-ImcKmipEntity** cmdlet provides an option to download a KMIP entity like Root CA Certificate, Client Certificate, and Client Private Key.

Syntax

```
Receive-ImcKmipEntity -Type <string> [-Protocol <string>] [-Pwd <string>] [-RemoteFile
<string>]
[-RemoteServer <string>] [-User <string>] [-XtraProperty <Hashtable>] [-Force]
[<CommonParameters>]
```

Example

```
Receive-ImcKmipEntity -Type RootCACertificate -RemoteServer 10.10.10.10 -User root -Pwd
<password>
-Protocol scp -RemoteFile "/root/test/RootCACertificate.pem" -Force
Receive-ImcKmipEntity -Type ClientCertificate -RemoteServer 10.10.10.10 -User root -Pwd
<password>
-Protocol scp -RemoteFile "/root/test/ClientCertificate.pem" -Force
Receive-ImcKmipEntity -Type ClientPrivateKey -RemoteServer 10.10.10.10 -User root -Pwd
<password>
-Protocol scp -RemoteFile "/root/test/ClientPrivateKey.pem" -Force
```

Receive-ImcLdapCACertificate

Exports the LDAP CA certificate from the IMC server to a remote server.

Syntax

```
Get-ImcExportLdapCACertificate | Receive-ImcLdapCACertificate
-Protocol scp -RemoteServer "10.10.10.10" -RemoteFile
"/root/test/ExportFileLdapCACertificate.crt" -User
"user" -Pwd "Password123" -Force
```

Remove-ImcLdapCACertificate

Removes the LDAP CA Certificate from the IMC server.

Syntax

```
Get-ImcLdapCACertificate | Remove-ImcLdapCACertificate -Force
```

Reset-ImcEventFilters

Resets event filters.

Syntax

```
Get-ImcEventManagement | Reset-ImcEventFilters -Force
Get-ImcRackUnit | Reset-ImcEventFilters -Force
```

Send-ImcBiosProfile

The **Send-ImcBiosProfile** cmdlet uploads the BIOS profile to the Cisco IMC. You can specify the profile details, such as IP/HostName, protocol, path and filename, username and password from a remote location.

Syntax

```
Send-ImcBiosProfile -BiosProfileManagement <BiosProfileManagement> [-Protocol <string>]
[-Pwd <string>] [-RemoteFile <string>] [-RemoteServer <string>] [-User <string>]
[-XtraProperty <Hashtable>]
[-Force] [<CommonParameters>]
```

Example

```
Get-ImcBiosProfileManagement | Send-ImcBiosProfile -Protocol scp -User root -Pwd <password>

-RemoteServer 10.10.10.10 -RemoteFile "/root/test/bios_profile_1" -Force
```

Send-ImcKmipEntity

The **Send-ImcKmipEntity** cmdlet provides an option to upload a KMIP entity, like Root CA Certificate, Client Certificate, and Client Private Key.

Syntax

```
Send-ImcKmipEntity -Type <string> [-Protocol <string>] [-Pwd <string>] [-RemoteFile <string>]

[-RemoteServer <string>] [-User <string>] [-XtraProperty <Hashtable>] [-Force]
[<CommonParameters>]
```

Example

```
Send-ImcKmipEntity -Type RootCACertificate -RemoteServer 10.10.10.10 -User root -Pwd
<password>
-Protocol scp -RemoteFile "/root/test/RootCACertificate.pem" -Force

Send-ImcKmipEntity -Type ClientCertificate -RemoteServer 10.10.10.10 -User root -Pwd
<password>
-Protocol scp -RemoteFile "/root/test/ClientCertificate.pem" -Force

Send-ImcKmipEntity -Type ClientPrivateKey -RemoteServer 10.10.10.10 -User root -Pwd <password>
```

```
-Protocol scp -RemoteFile "/root/test/ClientPrivateKey.pem" -Force
```

Send-ImcLdapCACertificate

Uploads the LDAP CA certificate located at the remote server on the IMC server.

Syntax

```
Get-ImcDownloadLdapCACertificate | Send-ImcLdapCACertificate
-Protocol scp -RemoteServer "10.10.10.10" -RemoteFile "
/root/test/LDAPCACErtificate.cer" -User "user" -Pwd
"Password123" -Force
```

Send-ImcPidCatalog

Uploads the PID catalogue file located at the remote server on the IMC server.

Syntax

```
Get-ImcPidCatalog | Send-ImcPidCatalog -Protocol scp
-RemoteServer "10.10.10.10" -RemoteFile
"/root/test/pid-ctlg-2_0_13a18.tar.gz" -User
"user" -Pwd "Password123" -Force

Get-ImcUploadPIDCatalog | Send-ImcPidCatalog -Protocol scp
-RemoteServer "10.10.10.10" -RemoteFile
"/root/test/pid-ctlg-2_0_13a18.tar.gz" -User
"user" -Pwd "Password123" -Force
```

Server Actions

The following table lists the new and changed cmdlets to perform server actions:

Action Description	Cmdlet in PowerTool Release 1.3.1 or earlier	Cmdlet in PowerTool 1.4.1 and Higher
Power On Server	Get-ImcRackUnit | Set-ImcRackUnit -AdminPower up	Get-ImcRackUnit | Start-ImcServer
Power Off Server	Get-ImcRackUnit | Set-ImcRackUnit -AdminPower soft-shut-down	Get-ImcRackUnit | Stop-ImcServer
Power Cycle Server	Get-ImcRackUnit | Set-ImcRackUnit -AdminPower cycle-immediate	Get-ImcRackUnit | Restart-ImcServer

Action Description	Cmdlet in PowerTool Release 1.3.1 or earlier	Cmdlet in PowerTool 1.4.1 and Higher
Hard Reset Server	Get-ImcRackUnit | Set-ImcRackUnit -AdminPower hard-reset-immediate	Get-ImcRackUnit | Reset-ImcServer
Turn On Locator LED	Get-ImcLocatorLed | Set-ImcLocatorLed -AdminState on	Get-ImcLocatorLed | Enable-ImcLocatorLed
Turn Off Locator LED	Get-ImcLocatorLed | Set-ImcLocatorLed -AdminState off	Get-ImcLocatorLed | Disable-ImcLocatorLed

Set a Boot Drive

Set a physical drive as a boot drive, using the following cmdlet:

```
Get-ImcStorageLocalDisk -Id 2 | Set-ImcStorageLocalDisk -AdminAction "set-boot-drive" -Force
```

Set a virtual drive as a boot drive, using the following cmdlet:

```
Get-ImcStorageVirtualDrive -Id 2 | Set-ImcStorageVirtualDrive -AdminAction "set-boot-drive"
 -Force
```

Change Disk Mode (JBOD to UG and vice-versa)

Change Disk Mode (JBOD to UG and vice versa)

```
Get-ImcStorageController | Set-ImcStorageController -AdminAction enable-jbod -Force -Xml
get-ImcStorageLocalDisk -Id 3 | Set-ImcStorageLocalDisk -AdminAction make-jbod -Force
get-ImcStorageLocalDisk -Id 3 | Set-ImcStorageLocalDisk -AdminAction make-unconfigured-good
 -Force
```

Set Boot Order

Set the boot order, using the following cmdlet:

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

```
Get-ImcLsbootDevPrecision | Add-ImcLsbootHdd -Name "RAID1_12" -Order 1 -State "Enabled"
-Type "LOCALHDD"
Get-ImcLsbootDevPrecision | Add-ImcLsbootVMedia -Name "CIMCDVD" -Order 2 -State "Enabled"
-Type "VMEDIA"
Get-ImcLsbootDevPrecision -Hierarchy | ConvertTo-ImcCmdlet
```

Setting BIOS Password

Note Setting BIOS password feature is applicable for E-Series servers only.

```
Get-ImcBiosPassword | Set-ImcBiosPassword -Password "<password>" -Force
```

Start-ImcOsInstallation

The **Start-ImcOsInstallation** cmdlet starts the NI-SCU operating system installation process.

Note For details on how to create the configuration files, answer files, and so on, see to Cisco UCS C-Series Server Configuration Utility documentation.

Syntax

```
Start-ImcOsInstallation -OsInstallation <OsiStart> [-AnswerFilePassword <string>]
[-AnswerFileShareFile <string>]
[-AnswerFileShareIp <string>] [-AnswerFileSharePath <string>] -AnswerFileShareType <string>
 [-AnswerFileUsername
<string>] [-ConfigShareFile <string>] [-ConfigShareIp <string>] [-ConfigSharePassword
<string>]
[-ConfigSharePath <string>] -ConfigShareType <string> [-ConfigShareUsername <string>]
-IsoShare <string> [-IsoShareIp <string>]
-IsoShareType <string> [-Password <string>] [-RemoteShareFile <string>] [-RemoteShareIp
<string>]
[-RemoteSharePassword <string>] [-RemoteSharePath <string>] -RemoteShareType <string>
[-RemoteShareUsername <string>] [-TimeOut
<uint>] [-Username <string>] [-XtraProperty <Hashtable>] [-Force] [<CommonParameters>]


Start-ImcOsInstallation -OsInstallationController <OsiController> [-AnswerFilePassword
<string>]
[-AnswerFileShareFile <string>] [-AnswerFileShareIp <string>] [-AnswerFileSharePath <string>]
 -AnswerFileShareType <string>
[-AnswerFileUsername <string>] [-ConfigShareFile <string>] [-ConfigShareIp <string>]
[-ConfigSharePassword <string>] [-ConfigSharePath <string>] -ConfigShareType <string>
[-ConfigShareUsername <string>] -IsoShare <string>
[-IsoShareIp <string>] -IsoShareType <string> [-Password <string>] [-RemoteShareFile <string>]

[-RemoteShareIp <string>] [-RemoteSharePassword <string>] [-RemoteSharePath <string>]
-RemoteShareType <string> [-RemoteShareUsername
<string>] [-TimeOut <uint>] [-Username <string>] [-XtraProperty <Hashtable>] [-Force]
[<CommonParameters>]
```

Example

```
Get-ImcOsInstallation | Start-ImcOsInstallation -AnswerFileShareIp 10.10.10.10
```

```
-AnswerFileUsername root -AnswerFilePassword <password> -AnswerFileSharePath "/root/test/osi"

-AnswerFileShareFile "" -AnswerFileShareType scp -ConfigShareIp 10.10.10.10
-ConfigShareUsername root
-ConfigSharePassword <password> -ConfigSharePath "/root/test/osi" -ConfigShareFile
"conf_file1" -ConfigShareType scp
-IsoShareIp 11.11.11.11 -IsoShare "/nfsshare/ucs-cxxx-scu-5.0.1a.iso" -IsoShareType nfs
-Username administrator
-Password <password> -RemoteShareIp 10.10.10.10 -RemoteShareUsername root -RemoteSharePassword
 <password>
-RemoteSharePath "/root/test/osi" -RemoteShareFile "" -RemoteShareType scp -Force


Get-ImcOsInstallationController | Start-ImcOsInstallation -AnswerFileShareIp 10.10.10.10
-AnswerFileUsername root -AnswerFilePassword <password> -AnswerFileSharePath "/root/test/osi"

-AnswerFileShareFile "" -AnswerFileShareType scp -ConfigShareIp 10.10.10.10
-ConfigShareUsername root
-ConfigSharePassword <password> -ConfigSharePath "/root/test/osi" -ConfigShareFile
"conf_file1"
-ConfigShareType scp -IsoShareIp 11.11.11.11 -IsoShare "/nfsshare/ucs-cxxx-scu-5.0.1a.iso"

-IsoShareType nfs -Username administrator -Password <password> -RemoteShareIp 10.10.10.10
-RemoteShareUsername root -RemoteSharePassword <password> -RemoteSharePath "/root/test/osi"
 -RemoteShareFile "" -RemoteShareType scp -Force
```

Test-ImcLdapBinding

Tests the LDAP Binding on the IMC server

Syntax

```
Get-ImcLdapCACertificate | Test-ImcLdapBinding -User "user"
-Pwd "Password123" -Force
```

Transaction Support

# Start a transaction, using the following cmdlet:

```
Start-ImcTransaction
```

# Perform an operation, using the following cmdlets:

```
$adapterHostEthIf = Get-ImcadapterUnit | Add-ImcadapterHostEthIf -Name adapterHostEth
$adapterHostEthIfModify = $adapterHostEthIf | Set-ImcadapterHostEthIf -PxeBoot enabled
$adapterEthISCSIProfile = $adapterHostEthIfModify | Add-ImcadapterEthISCSIProfile
-InitiatorName adapterHostEth -InitiatorIPAddress xx.xx.xx.xx -InitiatorSubnetMask
255.255.255.0 -DhcpISCSI enabled
$adapterEthISCSIProfile | Remove-ImcadapterEthISCSIProfile
$adapterHostEthIfModify | Remove-ImcadapterHostEthIf
```

# End a transaction, using the following cmdlet:

```
Complete-ImcTransaction
```

# Undo a transaction, using the following cmdlet:

```
Undo-ImcTransaction
```

vMedia Configuration

Configure vMedia, using the following cmdlet:

```
Get-ImcCommVMedia | Set-ImcCommVMedia -AdminState “enabled” -EncryptionState “enabled”
-Force
```

Create vNIC/Adapter

Create vNIC/Adapter

```
Get-ImcAdaptorUnit -Id "1" | Add-ImcAdaptorHostEthIf  -Name "eth2"  -UplinkPort "0"
```

Cisco UCS Communities

Cisco UCS Communities is a platform to discuss, share, and learn about the Cisco Products and Technologies. For blogs, discussion forums and documents related to UCS integrations with Cisco UCS Communities partner ecosystem, visit https://communities.cisco.com/ucsintegrations .

Related Cisco IMC Documentation and Documentation Feedback

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

- Cisco UCS C-Series Documentation Roadmap
- Cisco IMC XML API Programmer’s Guide for Cisco UCS C-Series Servers
- Cisco UCS E-Series Documentation Roadmap
- Cisco IMC XML API Programmer’s Guide for Cisco UCS E-Series Servers

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/c/en/us/td/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.