# XenServer 6.2.0 Powershell SDK Overview


- CTX138083
- Created On Jun 25, 2013
- Updated On Jun 25, 2013
- 21 found this helpful

- Product Documentation


**See Applicable Products**

## Overview of the new PowerShell SDK

The XenServer 6.2.0 SDK includes two versions of the PowerShell language bindings:

1. The "old" PowerShell SDK. This is the previously released XenServer PowerShell v1.0 Snap-In and as of XenServer 6.2.0 is considered deprecated.
2. The PowerShell SDK. This is the new, redesigned XenServer PowerShell v2.0 Snap-In.

This document provides a high level walk-through of the new PowerShell SDK to aid developers with their transition from the deprecated one.

Unlike the deprecated SDK, where each XenAPI call was exposed as a different cmdlet, in the new SDK the XenAPI calls have been grouped by API class. The naming scheme has also been revisited to provide cmdlet names complying with the known PowerShell naming conventions.

Users of the deprecated SDK will recognise the cmdlets `Connect-XenServer`, `Disconnect-XenServer`, `Get-XenSession`, `Wait-XenTask`. The `Connect-XenServer` cmdlet has been improved so now it is possible to specify and connect to multiple servers using the same credentials. Also, unlike before, more than one sessions can be opened to the same server. If more than one sessions are to be used, the parameter `-SetDefaultConnection` can be used to set the most frequently used session as the default one.

As before, all XenAPI calls are executed in the context of a login session. Previously, this session was specified by `-Url` or `-Server/-Port`. This has now been substituted by the `-SessionOpaqueRef` parameter.

**Note**: if there is only one open session or if a default session has been set, `-SessionOpaqueRef` does not need to be specified.

The cmdlets fall into the following categories:

- **Class Getters**.
  These retrieve a XenAPI object. Previously named as **Get-Xerver:T**, they now have names such as **Get-XenT**, where **T** is an exposed XenAPI class. The object to get can be specified by `-Ref` or, for those that have a uuid or name, `-Name` or `-Uuid`. If no parameters are specified all objects of this type are returned. For example:

  ```
  PS> Get-XenHost -Name "Demo Host"
  ```

- **Constructors.**

  These create a new XenAPI object. Previously named as **Create-Xenserver:T**, they now have names such as **New-XenT**, where **T** is an exposed XenAPI class. For example:

  ```
  PS> $vm = Get-XenVM -Name "Demo VM"
  PS> New-XenVBD -VM $vm -VDI $null -Userdevice 3 -Bootable $false -Mode RO -Type
  CD -Unpluggable $true -Empty $true -OtherConfig @{} -QosAlgorithmType "" -
  QosAlgorithmParams @{}
  ```

- **Class Removers**.
  Previously named as **Destroy-XenServer:T**, these now have names such as **Remove-XenT**, where **T** is an exposed XenAPI class. They destroy a XenAPI object. To specify the object to remove use the parameter `-T`, where `T` is an exposed XenAPI class, or `-Ref` or, for those objects that have a uuid or name, `-UUID` or `-Name`. For example:

  ```
  PS> Get-XenSR -Name "Demo SR" | Remove-XenSR
  ```

- **Property Setters.**
  Previously named as **Set-XenServer:T.Property**, these now have names such as **Set-XenT**, where **T** is an exposed XenAPI clas. They set a field of a XenAPI object. To specify the object use the parameter `-T`, where `T` is an exposed XenAPI class, or `-Ref` or, for those objects that have a uuid or name, `-UUID` or `-Name`. The field to set can be specified as an accordingly named parameter. Note that more than one fields at a time can be set in a synchronous call. For example:

  ```
  PS> Get-XenVM -Name "Demo VM" | Set-XenVM -NameLabel "New Name" -NameDescription
  "VM used for testing"
  ```

- **Property Adders**.
  Previously named as **Add-XenServer:T.Property**, these now have names such as **Add-XenT**, where **T** is an exposed XenAPI clas. They add an element to a field of a XenAPI object. To specify the object use the parameter `-T`, where `T` is an exposed XenAPI class, or `-Ref` or, for those objects that have a uuid or name, `-UUID` or `-Name`. The field to which the element will be added can be specified as an accordingly named parameter. Note that elements can be added to more than one fields at a time in a synchronous call. For example:

  ```
  PS> Add-XenHost -Name "Demo Host" -Tags "Tag1"
  ```

- **Property Removers**.
  Previously named as **Remove-XenServer:T.Property**, these now have names such as **Remove-XenTProperty**, where **T** is an exposed XenAPI class. They remove an element from a field of a XenAPI object. To specify the object use the parameter `-T`, where `T` is an exposed XenAPI class, or `-Ref` or, for those objects that have a uuid or name, `-UUID` or `-Name`. The field from which the element will be removed can be specified as an accordingly named parameter. Note that elements can be removed from more than one fields at a time in a synchronous call. For example:

  ```
  PS> Remove-XenHostProperty -Name "Demo Host" -Tags "Tag1" -OtherConfig "myKey"
  ```

- **Property Getters**.
  Previously named as **Get-XenServer:T.Property,** these now have names such as **Get-XenTProperty**, where **T** is an exposed XenAPI class. They retrieve the value of a field of a XenAPI object. To specify the object use the parameters `-T`, where `T` is an exposed XenAPI class, or `-Ref`. To specify the field, use the `enum` parameter `-XenProperty`. For example:

  ```
  PS> Get-XenPIFProperty -Ref OpaqueRef:f433bf7b-2b0c-5f53-7018-7d195addb3ca -
  XenProperty Network
  ```

- **Invokers.**
  Previously named as **Invoke-XenServer:T.Action**, these now have names such as **Invoke-XenT**, where **T** is an exposed XenAPI class. To specify the object use the parameter `-T`, where `T` is an exposed XenAPI class, or `-Ref` or, for those objects that have a uuid or name, `-UUID` or `-Name`. To specify the call to invoke, use the `enum` parameter `-XenAction`. For example:

  ```
  PS> Get-XenPBD -Uuid 1871ac51-ce6b-efc3-7fd0-28bc65aa39ff | Invoke-XenPBD -
  XenAction Unplug
  ```

Most of the XenAPI calls can be run synchronously or asynchronously. To run a cmdlet asynchronously use the parameter `-Async` where available.

**Note**: in the case of the setters only one field can be set asynchronously at a time, and in the case of the adders and removers, elements can be added or removed asynchronously to only one field at a time.

An important difference from the deprecated SDK is that the cmdlets that are not explicit "getters" but return objects, now do so only when the standard parameter `-PassThru` is specified. These cmdlets are `Connect-XenServer`, `Wait-XenTask`, the adders, setters, certain invokers, the constructors, the property removers, as well as all the asynchronous calls from any category (in the latter case the Task is returned).

Finally, as many of the cmdlets handle objects of type `XenRef<T>`, where `T` is an exposed API class, a new cmldet, `ConvertTo-XenRef` has been provided to aid conversion between the two. For example:

```
PS> Get-XenVM -Name "Demo VM" | ConvertTo-XenRef
```

## Further Information

For XenServer 6.2.0 SDK Release Notes, refer to CTX138082 – XenServer 6.2.0 SDK Release Notes

For XenServer 6.2.0 Release Notes, refer to CTX137826 – XenServer 6.2.0 Release Notes

For frequently asked questions about XenServer 6.2.0, refer to CTX137836 – XenServer 6.2.0 Technical FAQ

If you experience any difficulties, contact Citrix Technical Support.

## Applicable Products

- XenServer 6.2.0

### Share your comments or find out more about this topic

Citrix Forums



Exit Print View