Book of Prism - APIs and Interfaces

[PDF generated December 08 2021. For all recent updates please see the Nutanix Bible releases notes located at https://nutanixbible.com/release_notes.html. Disclaimer: Downloaded PDFs may not always contain the latest information.]

The HTML5 UI is a key part to Prism to provide a simple, easy to use management interface. However, another core ability are the APIs which are available for automation. All functionality exposed through the Prism UI is also exposed through a full set of REST APIs to allow for the ability to programmatically interface with the Nutanix platform. This allows customers and partners to enable automation, 3rd-party tools, or even create their own UI.

Core to any dynamic or "software-defined" environment, Nutanix provides a vast array of interfaces allowing for simple programmability and interfacing. Here are the main interfaces:

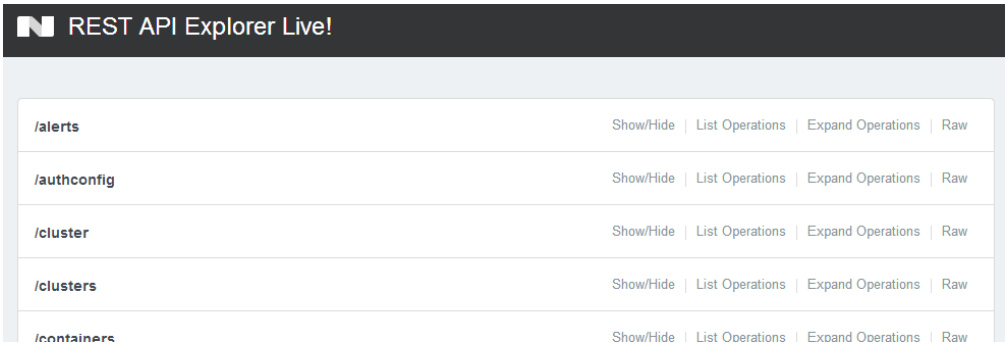
- REST API
- CLI - ACLI & NCLI
- Scripting interfaces

Nutanix.dev - Nutanix Developer Portal

To learn more about the available Nutanix Prism and product APIs, review sample code and go through self-paced labs, be sure to check out https://www.nutanix.dev!

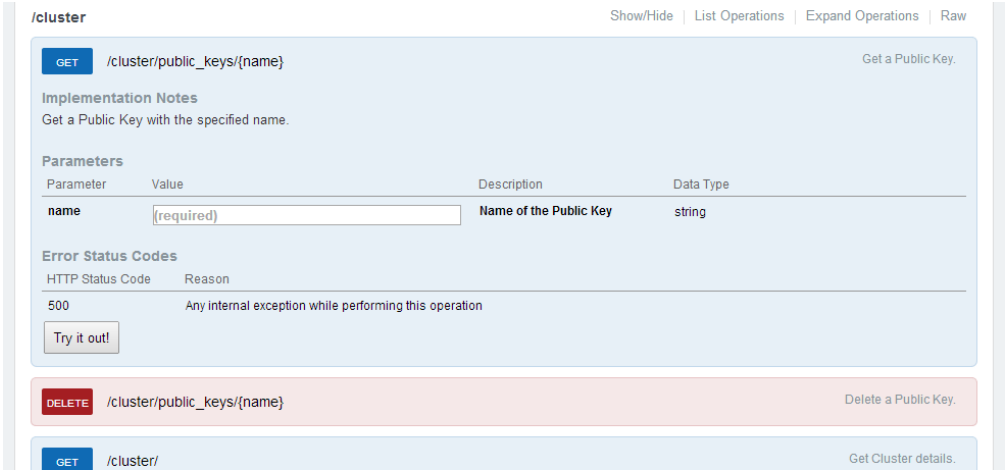
Core to this is the REST API which exposes every capability and data point of the Prism UI and allows for orchestration or automation tools to easily drive Nutanix action. This enables tools like Saltstack, Puppet, vRealize Operations, System Center Orchestrator, Ansible, etc. to easily create custom workflows for Nutanix. Also, this means that any third-party developer could create their own custom UI and pull in Nutanix data via REST.

The following figure shows a small snippet of the Nutanix REST API explorer which allows developers to interact with the API and see expected data formats:



Prism REST API Explorer

Operations can be expanded to display details and examples of the REST call:



Prism REST API Sample Call

API Authentication Scheme(s)

As of 4.5.x basic authentication over HTTPS is leveraged for client and HTTP call authentication.

ACLI

The AOS CLI (ACLI) is the CLI for managing the AOS portion of the Nutanix product. These capabilities were enabled in releases after 4.1.2.

NOTE: All of these actions can be performed via the HTML5 GUI and REST API. I just use these commands as part of my scripting to automate tasks.

Enter ACLI shell

Description: Enter ACLI shell (run from any CVM)

```
Acli
```

OR

Description: Execute ACLI command via Linux shell

```
ACLI Command
```

Output ACLI response in json format

Description: Lists AOS nodes in the cluster.

```
Acli -o json
```

List hosts

Description: Lists AOS nodes in the cluster.

```
host.list
```

Create network

Description: Create network based on VLAN

```
net.create TYPE.ID[.VSWITCH] ip_config=A.B.C.D/NN
```

```
Example: net.create vlan.133 ip_config=10.1.1.1/24
```

List network(s)

Description: List networks

```
net.list
```

Create DHCP scope

Description: Create dhcp scope

```
net.add_dhcp_pool NET NAME start=START IP A.B.C.D end=END IP W.X.Y.Z
```

Note: .254 is reserved and used by the AOS DHCP server if an address for the AOS DHCP server wasn't set during network creation

```
Example: net.add_dhcp_pool vlan.100 start=10.1.1.100 end=10.1.1.200
```

Get an existing network's details

Description: Get a network's VMs and details including VM name / UUID, MAC address and IP

```
net.list_vms NETNAME
```

```
Example: net.list_vms vlan.133
```

Configure DHCP DNS servers for network

Description: Set DHCP DNS

```
net.update_dhcp_dns NETNAME servers=COMMA SEPARATED DNS IPs domains=COMMA SEPARATED DOMAINS
```

```
Example: net.set_dhcp_dns vlan.100 servers=10.1.1.1,10.1.1.2 domains=splab.com
```

Create Virtual Machine

Description: Create VM

```
vm.create COMMA SEPARATED VM NAMES memory=NUM MEM MB num_vcpus=NUM VCPU num_cores_per_vcpu=NUM CORES ha_priority=PRIORITY INT
```

```
Example: vm.create testVM memory=2G num_vcpus=2
```

Bulk Create Virtual Machine

Description: Create bulk VM

```
vm.create  CLONEPREFIX[STARTING INT..END INT] memory=NUM MEM MB num_vcpus=NUM VCPU num_cores_per_vcpu=NUM CORES ha_priority=PRIORITY INT
```

```
Example: vm.create testVM[000..999] memory=2G num_vcpus=2
```

Clone VM from existing

Description: Create clone of existing VM

```
vm.clone CLONE NAME(S) clone_from_vm=SOURCE VM NAME
```

```
Example: vm.clone testClone clone_from_vm=MYBASEVM
```

Bulk Clone VM from existing

Description: Create bulk clones of existing VM

```
vm.clone CLONEPREFIX[STARTING INT..END INT] clone_from_vm=SOURCE VM NAME
```

Example: vm.clone testClone[001..999] clonefromvm=MYBASEVM

Create disk and add to VM

```
# Description: Create disk for OS
```

```
vm.disk_create VM NAME create_size=Size and qualifier, e.g. 500G container=CONTAINER NAME
```

```
Example: vm.disk_create testVM create_size=500G container=default
```

Add NIC to VM

Description: Create and add NIC

```
vm.nic_create VM NAME network=NETWORK NAME model=MODEL
```

```
Example: vm.nic_create testVM network=vlan.100
```

Set VM’s boot device to disk

Description: Set a VM boot device

Set to boot from specific disk id

```
vm.update_boot_device VM NAME disk_addr=DISK BUS
```

```
Example: vm.update_boot_device testVM disk_addr=scsi.0
```

Set VM’s boot device to CD-ROM

Set to boot from CD-ROM

```
vm.update_boot_device VM NAME disk_addr=CD-ROM BUS
```

```
Example: vm.update_boot_device testVM disk_addr=ide.0
```

Mount ISO to CD-ROM

Description: Mount ISO to VM CD-ROM

Steps:

1. Upload ISOs to container
2. Enable whitelist for client IPs
3. Upload ISOs to share

Create CD-ROM with ISO

```
vm.disk_create VM NAME clone_nfs_file=PATH TO ISO CD-ROM=true
```

```
Example: vm.disk_create testVM clone_nfs_file=/default/ISOs/myfile.iso CD-ROM=true
```

If a CD-ROM is already created just mount it

```
vm.disk_update VM NAME CD-ROM BUS clone_nfs_file=PATH TO ISO
```

```
Example: vm.disk_update atestVM1 ide.0 clone_nfs_file=/default/ISOs/myfile.iso
```

Detach ISO from CD-ROM

Description: Remove ISO from CD-ROM

```
vm.disk_update VM NAME CD-ROM BUS empty=true
```

Power on VM(s)

Description: Power on VM(s)

```
vm.on VM NAME(S)
```

```
Example: vm.on testVM
```

Power on all VMs

```
Example: vm.on *
```

Power on all VMs matching a prefix

```
Example: vm.on testVM*
```

Power on range of VMs

```
Example: vm.on testVM[0-9][0-9]
```

NCLI

NOTE: All of these actions can be performed via the HTML5 GUI and REST API. I just use these commands as part of my scripting to automate tasks.

Add subnet to NFS whitelist

Description: Adds a particular subnet to the NFS whitelist

```
ncli cluster add-to-nfs-whitelist ip-subnet-masks=10.2.0.0/255.255.0.0
```

Display Nutanix Version

Description: Displays the current version of the Nutanix software

```
ncli cluster version
```

Display hidden NCLI options

Description: Displays the hidden ncli commands/options

```
ncli helpsys listall hidden=true [detailed=false|true]
```

List Storage Pools

Description: Displays the existing storage pools

```
ncli sp ls
```

List containers

Description: Displays the existing containers

```
ncli ctr ls
```

Create container

Description: Creates a new container

```
ncli ctr create name=NAME sp-name=SP NAME
```

List VMs

Description: Displays the existing VMs

```
ncli vm ls
```

List public keys

Description: Displays the existing public keys

```
ncli cluster list-public-keys
```

Add public key

Description: Adds a public key for cluster access

SCP public key to CVM

Add public key to cluster

```
ncli cluster add-public-key name=myPK file-path=~/mykey.pub
```

Remove public key

Description: Removes a public key for cluster access

```
ncli cluster remove-public-keys name=myPK
```

Create protection domain

Description: Creates a protection domain

```
ncli pd create name=NAME
```

Create remote site

Description: Create a remote site for replication

```
ncli remote-site create name=NAME address-list=Remote Cluster IP
```

Create protection domain for all VMs in container

Description: Protect all VMs in the specified container

```
ncli pd protect name=PD NAME ctr-id=Container ID cg-name=NAME
```

Create protection domain with specified VMs

Description: Protect the VMs specified

```
ncli pd protect name=PD NAME vm-names=VM Name(s) cg-name=NAME
```

Create protection domain for DSF files (aka vDisk)

Description: Protect the DSF Files specified

```
ncli pd protect name=PD NAME files=File Name(s) cg-name=NAME
```

Create snapshot of protection domain

Description: Create a one-time snapshot of the protection domain

```
ncli pd add-one-time-snapshot name=PD NAME retention-time=seconds
```

Create snapshot and replication schedule to remote site

Description: Create a recurring snapshot schedule and replication to n remote sites

```
ncli pd set-schedule name=PD NAME interval=seconds retention-policy=POLICY remote-sites=REMOTE SITE NAME
```

List replication status

Description: Monitor replication status

```
ncli pd list-replication-status
```

Migrate protection domain to remote site

Description: Fail-over a protection domain to a remote site

```
ncli pd migrate name=PD NAME remote-site=REMOTE SITE NAME
```

Activate protection domain

Description: Activate a protection domain at a remote site

```
ncli pd activate name=PD NAME
```

Enable DSF shadow clones

Description: Enables the DSF Shadow Clone feature

```
ncli cluster edit-params enable-shadow-clones=true
```

Enable dedup for vDisk

Description: Enables fingerprinting and/or on disk dedup for a specific vDisk

```
ncli vdisk edit name=VDISK NAME fingerprint-on-write=true/false on-disk-dedup=true/false
```

Check cluster resiliency status

```
# Node status
ncli cluster get-domain-fault-tolerance-status type=node
```

```
# Block status
ncli cluster get-domain-fault-tolerance-status type=rackable_unit
```

PowerShell CMDlets

The below will cover the Nutanix PowerShell CMDlets, how to use them and some general background on Windows PowerShell.

Basics

Windows PowerShell is a powerful shell (hence the name ;P) and scripting language built on the .NET framework. It is a very simple to use language and is built to be intuitive and interactive. Within PowerShell there are a few key constructs/Items:

CMDlets

CMDlets are commands or .NET classes which perform a particular operation. They are usually conformed to the Getter/Setter methodology and typically use a Verb-Noun based structure. For example: Get-Process, Set-Partition, etc.

Piping or Pipelining

Piping is an important construct in PowerShell (similar to its use in Linux) and can greatly simplify things when used correctly. With piping you're essentially taking the output of one section of the pipeline and using that as input to the next section of the pipeline. The pipeline can be as long as required (assuming there remains output which is being fed to the next section of the pipe). A very simple example could be getting the current processes, finding those that match a particular trait or filter and then sorting them:

```
Get-Service | where {$_.Status -eq "Running"} | Sort-Object Name
```

Piping can also be used in place of for-each, for example:

```
# For each item in my array
$myArray | %{
  # Do something
}
```

Key Object Types

Below are a few of the key object types in PowerShell. You can easily get the object type by using the .getType() method, for example: $someVariable.getType() will return the objects type.

Variable

```
$myVariable = "foo"
```

Note: You can also set a variable to the output of a series or pipeline of commands:

```
$myVar2 = (Get-Process | where {$_.Status -eq "Running})
```

In this example the commands inside the parentheses will be evaluated first then variable will be the outcome of that.

Array

```
$myArray = @("Value","Value")
```

Note: You can also have an array of arrays, hash tables or custom objects

Hash Table

```
$myHash = @{"Key" = "Value";"Key" = "Value"}
```

Useful commands

Get the help content for a particular CMDlet (similar to a man page in Linux)

```
Get-Help CMDlet Name
```

```
Example: Get-Help Get-Process
```

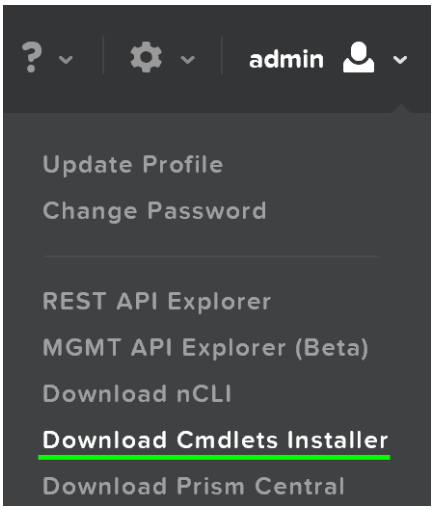
List properties and methods of a command or object

```
Some expression or object | Get-Member
```

```
Example: $someObject | Get-Member
```

Core Nutanix CMDlets and Usage

The Nutanix CMDlets can be downloaded directly from the Prism UI (post 4.0.1) and can be found on the drop down in the upper right hand corner:



Prism CMDlets Installer Link

Load Nutanix Snapin

Check if snapin is loaded and if not, load

```
if ( (Get-PSSnapin -Name NutanixCmdletsPSSnapin -ErrorAction SilentlyContinue) -eq $null )
{
    Add-PsSnapin NutanixCmdletsPSSnapin
}
```

List Nutanix CMDlets

```
Get-Command | Where-Object{$_.PSSnapin.Name -eq "NutanixCmdletsPSSnapin"}
```

Connect to a Nutanix cluster

```
Connect-NutanixCluster -Server $server -UserName "myuser" -Password (Read-Host "Password: " -AsSecureString) -AcceptInvalidSSLCerts
```

Get Nutanix VMs matching a certain search string

Set to variable

```
$searchString = "myVM"
$vms = Get-NTNXVM | where {$_.vmName -match $searchString}
```

Interactive

```
Get-NTNXVM | where {$_.vmName -match "myString"}
```

Interactive and formatted

```
Get-NTNXVM | where {$_.vmName -match "myString"} | ft
```

Get Nutanix vDisks

Set to variable

```
$vdisks = Get-NTNXVDisk
```

Interactive

```
Get-NTNXVDisk
```

Interactive and formatted

```
Get-NTNXVDisk | ft
```

Get Nutanix Containers

Set to variable

```
$containers = Get-NTNXContainer
```

Interactive

```
Get-NTNXContainer
```

Interactive and formatted

```
Get-NTNXContainer | ft
```

Get Nutanix Protection Domains

Set to variable

```
$pds = Get-NTNXProtectionDomain
```

Interactive

```
Get-NTNXProtectionDomain
```

Interactive and formatted

```
Get-NTNXProtectionDomain | ft
```

Get Nutanix Consistency Groups

Set to variable

```
$cgs = Get-NTNXProtectionDomainConsistencyGroup
```

Interactive

```
Get-NTNXProtectionDomainConsistencyGroup
```

Interactive and formatted

```
Get-NTNXProtectionDomainConsistencyGroup | ft
```

Resources and Scripts:

- Nutanix Github -https://github.com/nutanix/Automation
- Manually Fingerprint vDisks -http://bit.ly/1syOqch
- vDisk Report -http://bit.ly/1r34MIT
- Protection Domain Report -http://bit.ly/1r34MIT
- Ordered PD Restore -http://bit.ly/1pyolrb

NOTE: some scripts above are not maintained and should be used for reference only.

You can find more scripts on the Nutanix Github located at https://github.com/nutanix