gNOI - Operation Interface

- About gNOI, on page 1
- Guidelines and Limitations for gNOI, on page 3
- Configuring gNOI, on page 3
- System .Proto, on page 3
- OS .Proto, on page 10
- Cert .Proto, on page 11
- File .Proto, on page 11
- Put, on page 12
- Factory Reset .Proto, on page 13
- Containerz .Proto, on page 14
- Troubleshooting gNOI, on page 30

About gNOI

gRPC Network Operations Interface (gNOI) defines a set of gRPC-based micro-services for executing operational commands on network devices.

gNOI uses Google Remote Procedure Call (gRPC) as the transport protocol and the configuration is same as that of gNMI. For details on gNMI configuration, see gRPC Agent. To send gNOI RPC requests, user needs a client that implements the gNOI client interface for each RPC. In Cisco NX-OS Release 10.1(1) the gNOI defines Remote Procedure Calls (RPCs) for a limited number of components and some of them are related to hardware (like optical interfaces).

Proto files are defined for the gRPC micro-services and are available at GitHub.

Table 1: Supported gNOI RPCs

Proto	gNOI RPC	Supported
System	Ping	Yes
	Traceroute	Yes
	Time	Yes
	SetPackage	Yes
	SwitchControl Processor	Yes
	Reboot	Yes
	RebootStatus	Yes
	CancelReboot	Yes
OS	Activate	Yes
	Verify	Yes
Cert	LoadCertificate	Yes
File	Get	Yes
	Put	Yes
	Stat	Yes
	Remove	Yes
FactoryReset	Start	Yes
Containerz	Deploy	Yes
	ListImage	Yes
	RemoveImage	Yes
	RemoveContainer	Yes
	ListContainer	Yes
	StartContainer	Yes
	StopContainer	Yes
	UpdateContainer	Yes
	Log	Yes
	CreateVolume	Yes
	RemoveVolume	Yes
	ListVolume	Yes

Guidelines and Limitations for gNOI

The gNOI feature has the following guidelines and limitations:

- A maximum of 16 active gNOI RPCs are supported.
- The Cisco Nexus 9000 series switches would run one endpoint with one gNMI service and two gNOI microservices.

Note This document can provide gNOI client examples. The referred client is a python script to illustrate the raw exchange of gNOI request and response. Users shall use their own clients of interest.

Configuring gNOI

gNMI is a child functionality of the gRPC agent. See gRPC Agent, to enable the gRPC agent. Currently there is no separate configuration for gNOI.

Currently there is no separate config for gNOI.

System .Proto

The System proto service is a collection of operational RPCs that allows the management of a target outside the configuration and telemetry pipeline.

The following are the RPC support details for System proto:

RPC	Support	Description	Limitation
Ping	ping/ping6 cli command	Executes the ping command on the target and streams back the results. Some targets may not stream any results until all results are available. If a packet count is not explicitly provided, ping5 is used.	do_not_resolve option is not supported.
Traceroute	traceroute/traceroute6 cli command	Executes the traceroute command on the target and streams back the results. Some targets may not stream any results until all results are available. Max hop count of 30 is used.	itial_ttl, marx_ttl, wait, do_not_fragment, do_not_resolve and l4protocol options are not supported.

RPC	Support	Description	Limitation
Time	local time	Returns the current time on the target. Typically used to test if the target is responding.	-
SetPackage	install{ add | activate } install all nxos	Execute the install command on the target. This supports to install either the OS image or RPM	Only support RPMs or OS images on the bootflash: filesystem
SwitchControl Processor	system switchover cli command	Switches from the current route processor to the provided route processor. Switchover happens instantly and the response may not be guaranteed to return to the client.	Switchover occurs instantly. As a result, the response may not be guaranteed to return to the client.
Reboot	cli: reload [module]	Causes the target to reboot.	message option is not supported. Delay option is supported for switch reload, and the path option accepts one module number.
RebootStatus	show version [module] cli command	Returns the status of the reboot for the target.	-
CancelReboot	reload cancel	Cancels any pending reboot request.	-

SetPackage

The gNOI setpackage RPC provides a mechanism by which a software package may be copied to the switch and optionally activated. The software package may be copied directly from the RPC caller, or a remote download may be specified. In either case, the package transfer is verified using a hash before the package is installed.

The setpackage RPC and all related messages are fully defined here:

https://github.com/openconfig/gnoi/blob/main/system/system.proto#L61

The intended use of this RPC is to install software packages to the switch. A software package is either a RPM or a bootable NXOS system image. A RPM package is either a Cisco signed RPM or a third-party RPM. For third-party RPMs the following config is required before installing:

- `system software allow third-party`

Guidelines and Limitations for SetPackage

Following are the guidelines and limitations for SetPackage:

- Only standard NXOS system images and RPM packages are supported. Any other file type will be rejected and fail the RPC.
- Packages may be copied only to the “bootflash:” filesystem.
- No support for bundled images where NXOS system images are combined with third-party application packages
- No support for the RemoteDownload.source_address option
- The Package.filename must conform to standard NXOS file naming conventions and must only reside on “bootflash:”
- The Package.version may be empty or set to a version string. If empty, no check is done. If set, this version string must match the string of the package being installed.
- A remote copy operation that uses SSH/SFTP must use passwordless ssh, which must be configured by the administrator. This is outside the scope of the RPC operation. A password given in RemoteDownload.credentials will be rejected and fail the RPC.
- A remote copy operation that uses HTTP(S) may require the use of a proxy. This requires manual configuration in /etc/.curlrc.
- A remote copy assumes the vrf is “default” unless otherwise specified in RemoteDownload.source_vrf

RPC Options

The below table shows the Package message fields and values. Note that this information is client implementation independent, this is just the format of the protobuf messages and the data they contains. It is up to the client to properly construct each protobuf message.

Option	Description	Values
string filename	Destination filename of package on switch	A valid NXOS filename on bootflash:.
string version	The package version string	Maybe be empty, or set to the package version
bool activate	Specify whether to install the package only, or additionally activate the install package. The default is ‘'false’	**activate = false and OS image** Install the image without reloading the switch **activate = false and RPM** Install the RPM into the repository without enabling it **activate = true and OS image** Install the image and reload the switch **activate = true and RPM** Install and enable the RPM In caseofNX-OS SMU RPM, then it may or may not reload the switch. This is decided by the impact scope of the SMU. Please refer to the release information of the SMU RPM.

remote_download	Specifies the package is to be retrieved from another server location	Only non-interactive remote copy supported. Must have passwordless SSH configured. Remote information given in RemoteDownload message.

The following table showing RemoteDownload message

Option	Description	Values
path	Gives the host and path information to retrieve from. Hostname or IP may be used.	http(s) : example.com/path/to/package.rpm sftp/scp: a.b.c.d:/path/to/package.rpm
protocol	Remote copy protocol	SFTP, SCP, HTTP, HTTPS. Must be passwordless (scp/sftp).
source_address	Not supported	
source_vrf	The vrf over which to perform the remote copy.	If not specified “default” will be used
credentials	Credentials to access the remote resource for copying	The username and password, if required
• username	The username to use for the copy operation on remote server	Mandatory for remote copy operations
• password	The password if required, may be cleartext or hashed	If password is required (http(s)). Not required for scp/sftp as this requires passwordless ssh keys.

Note **Proxy** - For remote downloads through http/https, there exist scenarios where the download is only possible through a proxy. In such case, the user needs to update the curl proxy information.

Please refer to the below example.

This would allow the remote download to go through the specified proxy, and such config would be persisted across switch reload.

```
# feature bash
# run bash sudo su -
# <edit> /etc/.curlrc
--proxy http://<proxy>:<proxy-port>
# ln -s /etc/.curlrc /etc/curlrc
```

Note **Timing of response and switch reload** - When SetPackage reloads the switch, in some cases the switch may reload immediately which disrupts the network connectivity. It is expected that the client might not be able to receive the gnoi response.

Below are example interactions of clients using the setpackage RPC to install software on the switch. These are just examples using different clients to illustrate how the RPC is used. Other clients would use similar options but the details would differ.

Example – Install RPM, no activate

```
./gnxi-console --host <ip> --port 50051 --cafile /tmp/grpc.pem --hostnameoverride
ems.cisco.com -u admin -p <passwd> --operation gnoi.system.setpackage --arg
local_file=/data/xyz/bootflash/nxos64-cs.10.6.1.IQD9.0.29.F.bin,package.filename=bootflash:nxos64-cs.10.6.1.IQD9.0.29.F.bin,chunk_size=4000,hash_method=3,checksum=0ad2bf5c3b0be5b7363ac1e18fcc5f8f,package.activate=false


[gnoi.system.setpackage]-------------------------------

SetPackageRequest package {

    filename: "bootflash:nxos64-cs.10.6.1.IQD9.0.29.F.bin"

}

Sent 50777 content RPC messages

End>>

hash {

  method: MD5

  hash: "\n\322\277\\;\013\345\2676:\301\341\217\314_\217"

}



Hex coded checksum: 0ad2bf5c3b0be5b7363ac1e18fcc5f8f

[RESP] : 0






n9k_pi2(config)# show boot

Current Boot Variables:

sup-1

NXOS variable = bootflash:/nxos64-cs.10.6.1.IQD9.0.29.F.bin
```

Example – Install OS image, activate

```
./gnxi-console --host <ip> --port 50051 --cafile /tmp/grpc.pem --hostnameoverride
ems.cisco.com -u admin -p <passwd> --operation gnoi.system.setpackage --arg
local_file=/data/xyz/bootflash/nxos64-cs.10.6.1.IQD9.0.29.F.bin,package.filename=bootflash:nxos64-cs.10.6.1.IQD9.0.29.F.bin,chunk_size=4000,hash_method=3,checksum=0ad2bf5c3b0be5b7363ac1e18fcc5f8f,package.activate=true



[gnoi.system.setpackage]-------------------------------

SetPackageRequest package { filename: "bootflash:nxos64-cs.10.6.1.IQD9.0.29.F.bin" activate:
 true }
```

```
Sent 50777 content RPC messages

End>>

hash {

  method: MD5

  hash: "\n\322\277\\;\013\345\2676:\301\341\217\314_\217"

}


Hex coded checksum: 0ad2bf5c3b0be5b7363ac1e18fcc5f8f

[RESP] : 0



// switch reloaded into new image
```

Example – Install OS image, Remote download

```
./gnxi-console --host <ip> --port 50051 --cafile /tmp/grpc.pem --hostnameoverride
ems.cisco.com -u admin -p <passwd> --operation gnoi.system.setpackage --arg
[illegible]

[gnoi.system.setpackage]-------------------------------

SetPackageRequest package {

    filename: "bootflash:nxos64-cs.10.6.1.IQD9.0.29.F.bin"

    remote_download {

        path: "10.30.216.231:/nobackup/xyz/nxos64-cs.10.6.1.IQD9.0.29.F.bin"

        protocol: SCP

        credentials {

            username: "<user_id>"

        }

        source_vrf: "management" } }

End>>

hash {

    method: MD5

    hash: "\n\322\277\;\013\345\2676:\301\341\217\314_\217"

}

Hex coded checksum: 0ad2bf5c3b0be5b7363ac1e18fcc5f8f

[RESP] : 0
```

```
n9k_pi2(config)# show boot

Current Boot Variables:

sup-1

NXOS variable = bootflash:/nxos64-cs.10.6.1.IQD9.0.29.F.bin

Boot Variables on next reload:

sup-1

NXOS variable = bootflash:/nxos64-cs.10.6.1.IQD9.0.29.F.bin
```

Example – Install RPM, no activate

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> system.setpackage
  --arg local_file=valgrind-3.14.0-r0.corei7_64.rpm,

        package.filename=bootflash:valgrind-3.14.0-r0.corei7_64.rpm,
        package.version=3.14.0,
        hash.method=3,
        package.activate=false


[REQ]

SetPackageRequest

package {

  filename: "bootflash:valgrind-3.14.0-r0.corei7_64.rpm"

}



Sent 16010 content RPC messages

End>>

hash {

  method: MD5

  hash: "\002\313\016j\233A\"\235\261\325`\333\350>\314\346"

}



Hex coded checksum: 02cb0e6a9b41229db1d560dbe83ecce6



[RESP] : 0
```

Example – Install RPM, activate

```
n9k_pi2# show install active



Active Packages:
```

```
gnxi-console --host <ip> --port 50051 --cafile /tmp/grpc.pem --hostnameoverride ems.cisco.com
 -u admin -p <passwd> --operation gnoi.system.setpackage      --arg
local_file=/auto/n3k-dev/sanity/images/valgrind-3.14.0-r0.corei7_64.rpm,package.filename=bootflash:valgrind-3.14.0-r0.corei7_64.rpm,chunk_size=4000,hash.method=3,checksum=02cb0e6a9b41229db1d560dbe83ecce6,package.activate=true


### [gnoi.system.setpackage]-------------------------------

SetPackageRequest

package {

  filename: "bootflash:valgrind-3.14.0-r0.corei7_64.rpm"

  activate: true

}


Sent 65 content RPC messages

End>>

hash {

  method: MD5

  hash: "\002\313\016j\233A\"\235\261\325`\333\350>\314\346"

}


Hex coded checksum: 02cb0e6a9b41229db1d560dbe83ecce6

[RESP] : 0




n9k_pi2# show install active




Active Packages:

        valgrind-3.14.0-r0.corei7_64
```

OS .Proto

The OS service provides an interface for OS installation on a Target. The OS package file format is platform dependent. The platform must validate that the OS package that is supplied is valid and bootable. This must include a hash check against a known good hash. It is recommended that the hash is embedded in the OS package.

The Target manages its own persistent storage, and OS installation process. It stores a set of distinct OS packages, and always proactively frees up space for incoming new OS packages. It is guaranteed that the Target always has enough space for a valid incoming OS package. The currently running OS packages must never be removed. The Client must expect that the last successfully installed package is available.

The following are the RPC support details for OS proto:

RPC	Support	Description	Limitation
Activate	install all nxos bootflash:///img_name	Sets the requested OS version as the version that is used at the next reboot. This RPC reboots the Target.	Cannot rollback or recover if the reboot fails.
Verify	show version	Verify checks the running OS version. This RPC may be called multiple times while the Target boots until it is successful.	-

Note The Install RPC is not supported.

Cert .Proto

The certificate management service is exported by targets. Rotate, Install and other Certificate Proto RPCs are not supported.

RPC	Support	Description	Limitation
LoadCertificate	crypto ca import <trustpoint> pkcs12 <file> <passphrase>	Loads a bundle of CA certificates.	-

File .Proto

The file proto streams messages based on the features of the file.proto RPCs.

Get, Stat, and Remove RPCs support file systems such as - bootflash, bootflash://sup-remote, logflash, logflash://sup-remote, usb, volatile, volatile://sup-remote and debug. Put RPC only supports bootflash.

The following are the RPC support details for File proto:

RPC	Support	Description	Limitation
Get		Get reads and streams the contents of a file from the target. The file is streamed by sequential messages, each containing up to 64 KB of data. A final message is sent prior to closing the stream that contains the hash of the data sent. An error is returned if the file does not exist or there was an error reading the file.	Maximum file size limit is 32 MB.
Put		Upload a file to the target switch. Destination of the file follows the NXOS naming syntax . There is no restriction on the file type.	Maximum file size 3.5GB Only bootflash: supported
Stat		Stat returns metadata about a file on the target. An error is returned if the file does not exist or if there is an error in accessing the metadata.	-
Remove		Remove removes the specified file from the target. An error is returned if the file does not exist, if it is a directory, or the remove operation encounters an error.	-

Put

The Put operation allows the user to upload a file to the switch. The “remote_file” field shall follow the naming convention required by the NX-OS “copy” command, for example “bootflash:example.txt”. The following limitations are applied to files copied to the switch:

- Maximum file size of 3.5GB
- Target filesystem must be local bootflash:
- RPC user must have write permissions for the target file

The format of the PutRequest message is fully documented here: https://github.com/openconfig/gnoi/blob/main/file/file.proto#L78

Example

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> file.put
  --arg local_file=valgrind-3.14.0-r0.corei7_64.rpm,

        local_file=bootflash:valgrind-3.14.0-r0.corei7_64.rpm,
        remote_file=bootflash:test.rpm,
        hash.method=3


[REQ]

Open>>

open {

  remote_file: "bootflash:test.rpm"

  permissions: 493

}



...


End>>

hash {

  method: MD5

  hash: "\002\313\016j\233A\"\235\261\325`\333\350>\314\346"

}


Hex coded checksum: 02cb0e6a9b41229db1d560dbe83ecce6




[RESP] : 0
```

Factory Reset .Proto

This .proto currently defines only one RPC. Refer to https://github.com/openconfig/gnoi/blob/master/factory_reset/factory_reset.proto.

RPC	Support	Description	Limitation
FactoryReset	factory-reset module all [bypass-secure-erase] preserve-image force	Executes the factory-rest command on the target.	See below for detail.

FactoryReset

The gNOI factory reset operation erases all persistent storage on the specified module. This includes configuration, all log data, and the full contents of flash and (Solid State Drives) SSDs. The reset boots to the last boot image, erases all storage including license. gNOI factory reset supports two modes:

- A fast erase which can reformat and repartition only.
- A secure erase which can erase securely and wipe the data which is impossible to recover.

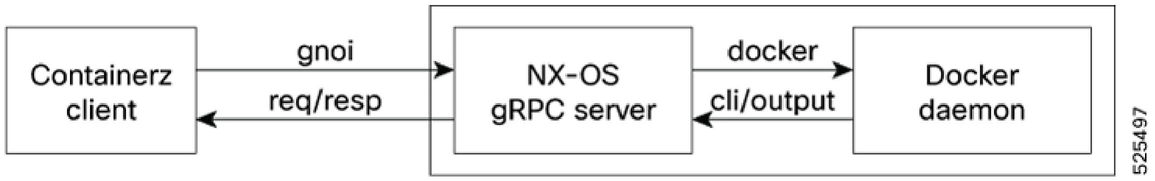
Option	Description	Values
factory_os	Specifies to rollback to the OS version as shipped from factory.	Setting to **true** on NX-OS is not supported, and it is mandatory to preserve the current boot image.
zero_fill	Specifies whether to perform more time consuming and comprehensive secure erase.	**zero_fill = true**: Specifies factory-reset module all preserve-image force. **zero_fill = false**: Specifies factory-reset module all bypass-secure-erase preserve-image force.

Containerz .Proto

NX-OS supports native docker access through the privileged bash shell. For more information, see Cisco Nexus 9000 Series NX-OS Programmability Guide - Chapter: Using Docker with Cisco NX-OS.

This gNOI containerz further provides the GRPC-ish wrapper interface around the existing NX-OS docker functionality. To leverage the containerz API interface, the user first needs to first start and provision the docker service per the above guideline. Once the native docker service is running and accessible via the NX-OS bash shell, then the user can use containerz client to manage the docker service. When NX-OS receives a containerz request, it then translates the given gnoi request to the corresponding docker command and invoke it. The docker command output would be translated back to the respective containerz response format, and then deliver back to the client.

Figure 1:



The following are the RPC support details for Containerz proto:

RPC	Support	Description
Deploy	docker image load	Upload a docker image tar and install into the repository
ListImage	docker image ls	List the deployed images

RemoveImage	docker image rm	Remove an image
RemoveContainer	docker container rm	Remove a container
ListContainer	docker container ls	List containers
StartContainer	docker container run docker container start	Start a new or stopped container
StopContainer	docker container stop docker container restart	Stop or restart a container
UpdateContainer	docker container stop docker container start	Update container
Log	docker container logs	Fetch the container logs
CreateVolume	docker volume create	Create docker volume
RemoveVolume	docker volume rm	Remove docker volume
ListVolume	Docker volume ls	List docker volume

Note The StartPlugin, StopPlugin, ListPlugin and RemovePlugin RPC are not supported.

<Deploy>

The Deploy operation allows the user to upload a docker image (either "tar" or "tar.gz" format) to the switch and register to the switch's repository.

The deploy RPC is documented here:

https://github.com/openconfig/gnoi/blob/main/containerz/containerz.proto#L54

The purpose of the deploy RPC is to load an image archive into the docker image registry. The deploy RPC provides equivalent functionality to the ""docker load" command line. The image archive file is expected to be a GZIP or TAR file and contain a valid docker image.

Guidelines and Limitations for <Deploy>

Following are guidelines and limitations for deploy operation.

- The file being deployed must be a valid Docker image in a Gzip or TAR format
- Plugins are not supported
- Only non-interactive remote copy can be done, meaning for SCP/SFTP the administrator must have setup passwordless SSH before the RPC is executed.
- For remote download case, the switch will not send an ImageTransferReady and it is assumed the client will not send a ImageTransferEnd.

RPC Options

The following protobuf messages are exchanged during the RPC execution (refer to RPC proto definition for complete details). These messages are sent by the client to the switch.

The ImageTransfer message is the first message sent from the client to the switch to initiate the deploy operation.

Message option	**Description**	**Value**
string name	The name to give the installed image	A valid docker i
string tag	The tag applied to the image once installed	A valid docker i
Unit64 image_size	Size (bytes) of the installed image. This will be check to ensure there is enough free space to load the image.	The user must p can be found wi Size" . The imag free docker part
Bool is_plugin	Not supported	False
RemoteDownload remote_download		Optional if swit Only non-intera have passwordl information giv

Table showing RemoteDownload message

Option	Description	Values
path	Gives the host and path information to retrieve from. Hostname or IP may be used.	http(s) : example.com/pa sftp/scp: a.b.c.d:/path/to/
protocol	Remote copy protocol	SFTP, SCP, HT (scp/sftp).
source_address	Not supported	
source_vrf	The vrf over which to perform the remote copy.	If not specified
credentials	Credentials to access the remote resource for copying	The username a
• username	The username to use for the copy operation on remote server	Mandatory for r
• password	The password if required, may be cleartext or hashed	If password is r scp/sftp as this r

Bytes Message

Option	Description	Values
content	This message carries the data if doing a direct copy to the switch	Raw bytes of th

ImageTransferEnd Message

Option	Description	Values
<none>	This is an empty message signaling the end of the bytes transfer to the switch.	No data in th

The following protobuf messages are sent from the switch to the client.in the form of a DeployResponse message.

- ImageTransferReady -- switch is ready to receive data
- ImageTransferProgress – indicates to client the number of bytes received so far
- ImageTransferSuccess -- indicates to client the transfer was successful
- Google.rpc.Status -- RPC status code

ImageTransferReady Message

Option	Description	Values
chunk_size	Indicates to client who much data to carry in each content message	Will always

ImageTransferProgress Message

Option	Description	Values
Bytes_received	The number of bytes successfully transferred to switch so far in data copy	The total nu of received doing a rem

ImageTransferSuccess Message

Option	Description	Values
name	The name of the image as loaded in the registry	Docker nam
tag	The tag that was used for this image	Docker tag a
image_size	The loaded image size	unit64 value

Example Client RPC Interaction

The below examples illustrate the usage of the deploy RPC using one particular client. Other clients will differ in the details but this illustrates the basic interaction.

Example – Direct upload

```
./gnxi-console --host 172.22.244.142 --port 50051 --cafile /tmp/grpc.pem --hostnameoverride
 ems.cisco.com -u admin -p <passwd>
--operation gnoi.containerz.deploy --arg
image_transfer.name=alpine_dev,image_transfer.tag="DEV-1.0.1",image_transfer.image_size=3309581,
image_transfer.is_plugin=false,local_file=/auto/mtx-dev/sanity/docker/docker_alpine_latest.tar.gz


### [gnoi.containerz.deploy]-------------------------------
```

```
Transfer>>
image_transfer {
  name: "alpine_dev"
  tag: "DEV-1.0.1"
  image_size: 3309581
}

[RESP] : 0
 image_transfer_ready {
  chunk_size: 64000
}

Sent 52 chunks
End>>
image_transfer_end {
}

[RESP] : 0
 image_transfer_progress {
  bytes_received: 2112000
}

[RESP] : 0
 image_transfer_success {
  name: "alpine_dev"
  tag: "DEV-1.0.1"
  image_size: 3309581
}

The state of docker on n9k:

bash-5.1# docker image ls
REPOSITORY                                  TAG          IMAGE ID       CREATED          SIZE
busybox                                    latest      af4709625109   5 months ago    4.27MB
alpine_dev                                 DEV-1.0.1   2c64cf60f6f0   6 months ago    7.35MB
```

Example – Remote Deploy

```
(pyats) [sjc-ads-7111]$ ./gnxi-console --host <ip> --port 50051 --cafile /tmp/grpc.pem
--hostnameoverride ems.cisco.com -u admin -p <passwd> --operation gnoi.containerz.deploy
--arg
[illegible]

[gnoi.containerz.deploy]--------------------------------
Transfer>> image_transfer {
     name: "alpine"
     tag: "new_image"
     image_size: 7349436
     remote_download {
         path: "10.0.0.1:/auto/mtx-dev/sanity/docker/docker_alpine_latest.tar.gz"
        protocol: SCP
        credentials {
                   username: "<user>"
           }
     source_vrf: "management" } }
[RESP] : 0 image_transfer_success {
       name: "alpine"
       tag: "new_image"
       image_size: 7349436
  }
bash-5.1# docker image ls
REPOSITORY    TAG         IMAGE ID       CREATED        SIZE
busybox       latest      af4709625109   8 months ago   4.27MB
alpine        new_image   2c64cf60f6f0   9 months ago   7.35MB
```

<ListImage>

The ListImage operation would return the list of images in the switch's local repository. Each image would include the information such as image id, name, and tag.

The ListImage operation supports the following options.

Option	Description	Values
Limit	The "limit" would restrict the number of image entries returned in the response, following the same display order of the "docker image list" command.	Valid numbe
Filter {key:value}	ListImage operation would return all images that match with the filter	Valid string

The following example illustrates the usage of ListImage operation.

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.listimage
  --arg limit=2


[REQ]
 limit: 2

[RESP] : 1
 id: "f7a401783329"
 image_name: "ghcr.io/openconfig/gnmic"
 tag: "latest"

[RESP] : 2
 id: "2c64cf60f6f0"
 image_name: "alpine"
 tag: "latest"
```

<RemoveImage>

The RemoveImage operation would remove a docker container identified by the name.

By default, the operation would fail if the image is used by some containers. The "force" option may be used to force to stop and remove the image. However, in this case, docker only untags the image.

The RemoveImage operation supports the following options.

Option	Description	Values
Name	Image name to be removed	Valid image
Tag	Tag to be removed	Valid tag as
Force	If "force=True", the image would be removed/untagged forcibly	True/False. I

following example illustrates the usage of RemoveImage operation

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.removeimage
  --arg name=alpine

[REQ]
 name: "alpine"
```

```
[RESP]
 code: UNKNOWN
detail: "Error response from daemon: conflict: unable to remove repository reference
\"alpine\" (must force) - container 21e718a3bf52 is using its referenced image 2c64cf60f6f0\n"
```

Following example illustrates force removal (untag only).

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.removeimage
  --arg name=alpine,force=True

[REQ]
 name: "alpine"
force: true

[RESP]
 code: SUCCESS
 detail: "Untagged: alpine:latest\n"
```

Following example illustrates Force removal again after the container is stopped.

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.removeimage
  --arg name= 2c64cf60f6f0,force=True

[REQ]
 name: "foo"
force: true

[RESP]
 code: UNKNOWN
detail: "Deleted: sha256:2c64cf60f6f05256c049f58403d0c3b33f2145a70830cb8e24efa826854c1a46\n"
```

<StartContainer>

The StartContainer operation could be used to either start a new container or restart a previously stopped container. The decision is based on the following conditions.

- If the the container does not exist, then start a new container
- Otherwise
    - If the container is already running, then return an error.
    - If the container is stopped but the request includes more fields than just the instance name, then return an error.
    - If the container is stopped and the request contains only the instance name, then start the container.

The StartContainer supports the following options

Option	Description	Values
image_name	Image name to start container	Valid image nar
tag	Image tag to start container	Valid tag as strii
cmd	Command to run in a new container	Valid command
instance_name	Identifier for a new container	Valid instance_

port -internal -external	List of internal ports to be exposed outside the container	Valid interna Ex. Internal External por
environment =<VAR1>:<value>	Set environment variables in the container	Environmen or key value Ex. VAR1 o
network	Network to attach this container to	Valid netwo This could b available in
Cap.ADD	Capabilities to be added	The list of v https://man7.o This is defau AUDIT_WF FOWNER, I NET_BIND SETGID, SE
Cap.REMOVE	Capabilities to be removed	
Restart.policy	Set the restart policy	Valid restart NONE, ALV ON_FAILU
Restart.attempts	Set the restart attempts	Valid numbe
RunAs.user RunAs.group	Set the user and group this container should run as	Valid UID a
Label <key>:<val>	A key-value pair to set the metadata to container	String labels Ex. my-labe
Limits.max_cpu	Set number of max CPUs this container can use	A valid num Ex. a value c maximum o half a cpu.
Limits.soft_mem_bytes	Set memory soft limit	A valid men smaller than Ex limits.so
Limits.hard_mem_bytes	Set memory hard limit	A valid men Ex. limits.ha

Guidelines and Limitations

- The StartContainer operation is achieved via “docker run” CLI. The maximum supported length for this CLI is 2500 bytes.
- In the case of RunAs, current docker only supports UID and GID. Container starts with any value of UID and GID and fails when username and group names are provided.
- In case of any warning while starting the container, NXOS would start the container without sending the warning back.

The following example illustrates the usage of StartContainer operation

Example – Start a new container

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.startcontainer
  --arg instance_name=foo,
        image_name=busybox,
        cmd="sh -c \"sleep 300\"",
        volume=my_volume:/docker

[REQ]
 image_name: "busybox"
 cmd: "sh -c \"sleep 300\""
 instance_name: "foo"
 volumes {
   name: "my_volume"
   mount_point: "/docker"
 }

[RESP]
 start_ok {
  instance_name: "foo"
}
```

Example – Start a new container

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.startcontainer --arg
 image_name=alpine,
                        cmd="sh -c \"while true; do $(echo date); sleep 1; done\"",
                        instance_name=foo,
                        volume=my_volume:/docker:True,
                        env=VAR1:value1,env=VAR2:value2,
                        cap.add=FOWNER,
                        cap.add=AUDIT_WRITE,
                        cap.remove=SETFCAP,
                        cap.remove=NET_RAW,
                        network=host,
                        restart.policy=3,restart.attempts=2,
                        run_as.user=1001,run_as.group=2003,
                        label=env:prod,label=team:neteng,
                        limits.max_cpu=1.5,
                        limits.soft_mem_bytes=7000000,
                        limits.hard_mem_bytes=90000000,
                        port=1001:2001
[REQ]
image_name: "alpine2"
cmd: "sh -c \"while true; do date; sleep 1; done\""
instance_name: "foo"
ports {
  internal: 2001
  external: 1001
}
environment {
  key: "VAR1"
  value: "value1"
}
environment {
  key: "VAR2"
  value: "value2"
}
volumes {
  name: "my_volume"
  mount_point: "/docker"
  read_only: true
}
network: "host"
cap {
  add: "FOWNER"
  add: "AUDIT_WRITE"
  remove: "SETFCAP"
  remove: "NET_RAW"
}
restart {
  policy: ON_FAILURE
  attempts: 2
}
run_as {
  user: "1001"
  group: "2003"
}
labels {
  key: "env"
  value: "prod"
}
labels {
  key: "team"
  value: "neteng"
}
limits {
  max_cpu: 1.5
  soft_mem_bytes: 7000000
  hard_mem_bytes: 90000000
}

[RESP]
start_ok {
  instance_name: "foo"
}
```

Example – Start an existing and running container

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.startcontainer
  --arg instance_name=foo,
        image_name=busybox

[REQ]
 image_name: "busybox"
 instance_name: "foo "

[RESP]
 start_error {
  error_code: UNKNOWN
  details: "container is already running"
}
```

Example – Start an existing but stopped container

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.startcontainer
  --arg instance_name=foo
```

```
[REQ]
 instance_name: "foo "

[RESP]
 start_ok {
  instance_name: "foo "
}
```

<RemoveContainer>

The RemoveContainer operation would remove a docker container identified by the name.

By default, the operation would fail if the container is still running. The **force** option can be used to force to stop and remove the container.

The RemoveContainer operation supports the following options.

Option	Description	Values
Name	Container name to be removed	Valid container
Force	If "force=True", the container would be stopped and removed forcibly	True/False. Def

It is important to know that containers may continue to exist even if the underlying image has been removed, as there is a certain degree of independence before the docker images and containers. The user should remove the images and containers with caution.

Example

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.removecontainer

[REQ]
 name: "foo"

[RESP]
 code: RUNNING
detail: "Error response from daemon: You cannot remove a running container
706723d4ae83e3fa7b5dcfe4edbb8edd570ea8af690c3eef2526f9c603bfba97. Stop the container before
 attempting removal or force remove\n"
```

Example – Force removal

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.removecontainer
  --arg force=True

[REQ]
 name: "foo"
force: true

[RESP]
 code: SUCCESS
detail: "Container removed successfully"
```

<ListContainer>

The ListContainer operation would return the list of containers in the switch's local repository.

Each container would include the information such as the image name, container id, name, running status, metadata which includes the labels and image hash computed by the container runtime.

The ListContainer operation supports the following options

Option	Description	Values
All	The “all” option can be used to return all running and non-running containers. By default, ListContainer operation only returns the running containers.	**True/False.**
Limit	The “limit” would restrict the number of container entries returned in the response, following the same display order of the “docker container list” command.	Valid numbe
Filter {key:value}	ListImage operation would return all images that match with the filter	Valid string

Example

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.listcontainer
  --arg name=my_volume

[REQ]

[RESP] : 1
 id: "706723d4ae83e3fa7b5dcfe4edbb8edd570ea8af690c3eef2526f9c603bfba97"
 name: "my_volume"
 image_name: "alpine"
 status: RUNNING
 hash {
   method: SHA256
   hash: "sha256:2c64cf60f6f05256c049f58403d0c3b33f2145a70830cb8e24efa826854c1a46"
 }
```

Example – All containers

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.listcontainer
  --arg all=True

[REQ]
 all: true

[RESP] : 1
 id: "706723d4ae83e3fa7b5dcfe4edbb8edd570ea8af690c3eef2526f9c603bfba97"
 name: "foo"
 image_name: "alpine"
 status: RUNNING
 hash {
   method: SHA256
   hash: "sha256:2c64cf60f6f05256c049f58403d0c3b33f2145a70830cb8e24efa826854c1a46"
 }

[RESP] : 2
 id: "21e718a3bf525635b606b6a121bf8a79c323da4edc044f2e6c3811cd3219de94"
 name: "test"
 image_name: "alpine"
 status: STOPPED
 hash {
   method: SHA256
   hash: "sha256:2c64cf60f6f05256c049f58403d0c3b33f2145a70830cb8e24efa826854c1a46"
 }
```

<StopContainer>

The StopContainer operation would stop a docker container identified by the name, and is achieved via the "docker container stop" command. When the "restart" option is specified, It would also restart the docker container using "docker container restart" command.

StopContainer operation supports the following options

Option	Description	Values
Instance_name	Container instance to be stopped	Valid instance_
force	If this option is true, container would be stopped forcibly	True/False. The
restart	If this option is true, container would be restarted using "docker container restart"	True/False. The

Limitations and Caveats

- When the running container is stopped using StopContainer operation, it would take 10s to stop the container without returning error.
- When the running container is stopped with the "force" option, the container will stop after 5s.

The following example illustrates the usage of StopContainer operation.

Example

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.stopcontainer
  --arg instance_name=foo


[REQ]
 Instance_name: "foo"

[RESP]
 code: SUCCESS
details: "Container stopped/re-started successfully"
```

<UpdateContainer>

The UpdateContainer operation would update a container instance with a new image and arguments. If the update fails, it shall restore the container back to its previous status.

Note that currently the native docker does not support this operation, and thus NX-OS would achieve this functionality via a sequence docker operation

1. Stop the given container if it is currently running
2. Rename the container to a temporary name as the backup
3. Start a new container with the same container name, using the given new image and arguments
4. If the new container succeeds to run, then remove the backup container
5. If the new container fails to run, then try to restore
    - Stop and remove the new container
    - Rember the backup back to its original container name

• Start the container if it was running

The UpdateContainer operation supports the following option

Option	Description	Values
instance_name	The name of the running container to update.	A valid insta
image_name	The image to update the container to.	A valid imag
image_tag	The tag to update the container to.	A valid tag a
async	Run this operation asynchronously or not	True/False. T
All options supported by startContainer operation to run the container with.		

Example

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.updatecontainer
  --arg instance_name=foo,image_name=alpine,params=<same as StartContainer>

[REQ]
 instance_name: "foo"
 image_name: ”alpine”
 params {
  instance_name: "foo"
  cap {
  }
  ...
 }

[RESP]
 updateOk: {
  instanceName: “foo”
 }
```

<Log>

The Log operation allows to client to fetch the container logs, similar to the “docker container logs” command. The operation can choose whether to “follow” the log.

- If follow=False, the switch would fetch and return the current logs
- If follow=True, the switch would return the current logs, and stream the new logs until the RPC is canceled by the client, or when the container stops.

The Log operation supports the following option

Option	Description	Values
instance_name	Container name	Valid contai
follow	If set, the stream remains open until the client cancels it.	True/False. T

Example

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.log
  --arg instance_name=test

[REQ]
```

```
 instance_name: "test"

[RESP] : 1
 msg: "Wed Jun 11 20:21:05 UTC 2025\nWed Jun 11 20:21:06 UTC 2025\nWed Jun 11 20:21:07 UTC
 2025\nWed Jun 11 20:21:08 UTC 2025\nWed Jun 11 20:21:09 UTC 2025\nWed Jun 11 20:21:10 UTC
 2025\nWed Jun 11 20:21:11 UTC 2025\nWed Jun 11 20:21:12 UTC 2025\n
Wed Jun 11 20:21:13 UTC 2025\nWed Jun 11 20:21:14 UTC 2025\nWed Jun 11 20:21:15 UTC 2025\nWed
 Jun 11 20:21:16 UTC 2025\nWed Jun 11 20:21:17 UTC 2025\nWed Jun 11 20:21:18 UTC 2025\nWed
 Jun 11 20:21:19 UTC 2025\nWed Jun 11 20:21:20 UTC 2025\nWed Jun 11 20:21:21 UTC 2025\nWed
 Jun 11 20:21:22 UTC 2025\nWed Jun 11 20:21:23 UTC 2025\nWed Jun 11 20:21:24 UTC 2025\n"

[RESP] : 2
 msg: "Wed Jun 11 20:21:25 UTC 2025\n
```

Example – Follow the logs

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.log
  --arg instance_name=test,follow=True

[REQ]
 instance_name: "test"

…

[RESP] : 16
 msg: "Wed Jun 11 20:26:06 UTC 2025\nWed Jun 11 20:26:07 UTC 2025\nWed Jun 11 20:26:08 UTC
 2025\nWed Jun 11 20:26:09 UTC 2025\nWed Jun 11 20:26:10 UTC 2025\nWed Jun 11 20:26:11 UTC
 2025\nWed Jun 11 20:26:12 UTC 2025\nWed Jun 11 20:26:13 UTC 2025\n"

[RESP] : 17
 msg: "Wed Jun 11 20:26:14 UTC 2025\n"

[RESP] : 18
 msg: "Wed Jun 11 20:26:15 UTC 2025\n"

…
```

<CreateVolume>

The CreateVolume operation would create a docker volume identified by the name.

The user can mount this volume into multiple containers.

The CreateVolume operation supports the following options

Option	Description	Values
name	Name of the volume	A valid volume
driver	The volume driver	NXOS only sup
options	Options for the driver. The actual option keys and values are driver specific.	NXOS does not
labels	Labels to apply to the volume. Labels are metadata for the the volume.	String labels. It Ex. my-label or

Guideines and Limitations

- NXOS only supports “LOCAL” driver ro create volume.
- NXOS does not support “Local mount options”.

Example

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.createvolume
  --arg name=my_volume

[REQ]
 name: "my_volume"

[RESP] : 1
 name: “my_volume”
```

\

<RemoveVolume>

The RemoveVolume operation would remove a docker volume identified by the name.

The RemoveVolume operation supports the following options

Option	Description	Values
name	Name of volume to be removed	Valid volum
force	Force the volume removal	Not support

Guidelines and Limitations

- The “force” option is not supported by docker, and thus please make sure to detach the volume from all containers to allow the volume to remove successfully

Example

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.removevolume
  --arg name=my_volume

[REQ]
 name: "my_volume"

[RESP] : 1
 name: “my_volume”
```

<ListVolume>

The ListVolume operation would return the list of docker volumes.

It includes the volume name, created timestamp, driver, and volume options and labels if applicable.

The ListVolume operation supports the following options

Option	Description	Values
Filter {key:value}	ListVolume operation would return all volumes that match with the filter	Valid string

Example

```
> gnoi-client --host <ip> --port <port> -u admin -p <pass> containerz.listvolume

[REQ]

[RESP] : 1
 name: "92945dc4e9ffdf571c85994738562b1d1f54158f784cac3eadc080c558e034ee"
```

```
created {
  seconds: 1748490463
}
driver: "local"
```

Troubleshooting gNOI

Debug gNOI

To verify the gNOI status, enter the following commands.

Show Commands

Command	Description
clear grpc gnoi rpc	Serves to clean up the counters or calls.
debug grpc events {**events**|**errors**} **show grpc nxsdk event-history** {**events**|**errors**}	Debugs the events and errors from the event history.
clear grpc gnoi rpc	Serves to clean up the counters or calls.

Example Output

show grpc gnmi service statistics

```
=============
gRPC Endpoint
=============

Vrf            : management
Server address : [::]:50051

Status         : Running - certificate expired
Cert notBefore : Jun 20 16:43:49 2023 GMT
Cert notAfter  : Jun 21 16:43:49 2023 GMT
Client Root Cert notBefore : n/a
Client Root Cert notAfter  : n/a

Max concurrent calls            :  16
Active calls                    :  0
```

Bash Commands

In particular for containerz, see the debug steps in Cisco Nexus 9000 Series NX-OS Programmability Guide - Chapter: Using Docker with Cisco NX-OS. Basically, the user shall be able to directly access the docker utilities to inspect the detail of each image and container for debug purpose.

Command	**Description**
docker image list	List the docker images

docker container list	List the docker containers
docker volume list	List the docker volumes

Example Output

Docker image list

```
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE

busybox      latest    af4709625109   8 months ago    4.27MB

alpine       latest    2c64cf60f6f0   9 months ago    7.35MB
```

docker container list

```
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

root@nxosv-104#docker container list -a

CONTAINER ID   IMAGE            COMMAND                    CREATED       STATUS
       PORTS     NAMES

9bd2156dd341   busybox:latest   "sh"                       5 days ago    Exited (255) 3 days
 ago              stupefied_euclid

c8d945cfc5f6   busybox:latest   "/bin/bash"                5 days ago    Created
                  determined_lehmann

7463c28fd039   busybox          "sh -c 'while true; …"     5 days ago    Exited (255) 3 days
 ago              busy_test

26dc68f0e4d2   alpine:latest    "/bin/bash"                5 days ago    Created
                  admiring_raman

0f5de1376925   alpine:latest    "/bin/bash"                5 days ago    Created
                  vigilant_keller

4ac6c84be9f1   alpine:latest    "/bin/bash"                5 days ago    Created
                  optimistic_dirac

e5b43676fc98   alpine           "sh -c 'sleep 100'"        3 weeks ago   Exited (0) 3 weeks
ago               gnoi_docker_container_20250519_183038_7

d912c55525ae   alpine           "sh -c 'sleep 100'"        3 weeks ago   Exited (0) 3 weeks
ago               gnoi_docker_container_20250519_183028_6

0360fcde5af4   alpine           "sh -c 'sleep 100'"        3 weeks ago   Exited (0) 3 weeks
ago               gnoi_docker_container_20250519_183019_5

53f2e94399ca   alpine           "sh -c 'sleep 100'"        3 weeks ago   Exited (0) 3 weeks
ago               gnoi_docker_container_20250519_182049_4

35130a5e3f86   alpine           "sh -c 'sleep 100'"        3 weeks ago   Exited (0) 3 weeks
ago               gnoi_docker_container_20250519_182039_3

5d6dbe0dc904   alpine           "sh -c 'sleep 100'"        3 weeks ago   Exited (0) 3 weeks
ago               gnoi_docker_container_20250519_181102_2
```

Gathering Debug Logs

gNOI is a child service of the gRPC agent. For more information, see gRPC Agent chapter.

Docker log

In particular for containerz, please investigate the log at the below location. This file maintains the debug messages for the docker daemon. Please refer to https://docs.docker.com/engine/daemon/logs/ for the details.

```
/var/log/docker
```