

# gNOI-gRPC Network Operations Interface


- About gNOI, on page 1
- Supported gNOI RPCs, on page 1
- System Proto, on page 2
- OS Proto, on page 3
- Cert Proto, on page 4
- File Proto, on page 4
- gNOI Factory Reset, on page 5
- Guidelines and Limitations, on page 6
- Verifying gNOI, on page 6


## About gNOI

gRPC Network Operations Interface (gNOI) defines a set of gRPC-based micro-services for executing operational commands on network devices. The operational commands supported are Ping, Traceroute, Time, SwitchControlProcessor, Reboot, RebootStatus, CancelReboot, Activate and Verify.

gNOI uses gRPC as the transport protocol and the configuration is same as that of gNMI. For details on configuration, please refer to Configuring gNMI.

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 related to hardware (like optical interfaces).

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

## Supported gNOI RPCs

The following are the supported gNOI RPCs:

*Table 1:*

| Proto | gNOI RPC | Supported |
|---|---|---|
| System | Ping | Yes |
| | Traceroute | Yes |
| | Time | Yes |
| | SwitchControl Processor | Yes |
| | Reboot | Yes |
| | RebootStatus | Yes |
| | CancelReboot | Yes |
| OS | Activate | Yes |
| | Verify | Yes |
| Cert | LoadCertificate | Yes |
| File | Get | Yes |
| | Stat | Yes |
| | Remove | Yes |

# 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, 5 is used. | do_not_resolve option is not supported. |

| RPC | Support | Description | Limitation |
|---|---|---|---|
| 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. |
| Time | local time | Returns the current time on the target. Typically used to test if the target is responding. | - |
| 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. | - |

**Note** The SetPackage RPC is not supported.

# 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 Cert Proto RPCs are not supported.

The following are the RPC support details for Cert proto:

| RPC | Support | Description | Limitation |
|---|---|---|---|
| LoadCertificate | crypto ca import <trustpoint><br>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. Put and other RPCs that are not listed here are not supported in File Proto.

Get, Stat, and Remove RPCs support file systems - bootflash, bootflash://sup-remote, logflash, logflash://sup-remote, usb, volatile, volatile://sup-remote and debug.

The following are the RPC support details for File proto:

| RPC | 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. |
| 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, is a directory, or the remove operation encounters an error. | - |

# gNOI Factory Reset

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 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.

The gNOI factory reset operation as defined in factory_reset.proto erases all persistent storage on the device. Refer to factory_reset.proto link here https://github.com/openconfig/gnoi/blob/master/factory_reset/factory_reset.proto.

The following is the example of a gNOI FactoryReset service:

```
/ The FactoryReset service exported by Targets.
service FactoryReset {
  // The Start RPC allows the Client to instruct the Target to immediately
  // clean all existing state and boot the Target in the same condition as it is
  // shipped from factory. State includes storage, configuration, logs,
  // certificates and licenses.
  //
  // Optionally allows rolling back the OS to the same version shipped from
  // factory.
  //
  // Optionally allows for the Target to zero-fill permanent storage where state
  // data is stored.
  //
```

```
  // If any of the optional flags is set but not supported, a gRPC Status with
  // code INVALID_ARGUMENT must be returned with the details value set to a
  // properly populated ResetError message.
  rpc Start(StartRequest) returns (StartResponse);
}

message StartRequest {
  // Instructs the Target to rollback the OS to the same version as it shipped
  // from factory.
  bool factory_os = 1;
  // Instructs the Target to zero fill persistent storage state data.
  bool zero_fill = 2;
}
```

The following are the details of the arguments used in gNOI Factory Reset:

- **factory_os = false**: 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.

# Guidelines and Limitations

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.
- In 10.1(1) release, the gNOI RPCs are implemented with the equivalent CLI. The existing CLI restrictions or valid options remain as applicable.
- Beginning with 10.2(1)F release, the file.proto and cert.proto RPCs are supported.
- gRPC traffic destined for a Nexus device will hit the control-plane policer (CoPP) in the default class. To limit the possibility of gRPC drops, configure a custom CoPP policy using the gRPC configured port in the management class.

# Verifying gNOI

To verify the gNOI configuration, enter the following commands:

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

| Command | Description |
| --- | --- |
| show grpc internal gnoi rpc {summary\|detail} | An internal keyword command added for serviceability. |