gNOI：操作インターフェイス

gNOI について

gRPC ネットワーク オペレーション インターフェイス（gNOI）は、ネットワーク デバイス上で操作コマンドを実行するための gRPC ベースのマイクロサービス セットを定義します。

gNOI は Google リモートプロシジャコール（gRPC）をトランスポートプロトコルとして使用します。構成は gNMI と同じです。gNMI 構成の詳細については、gRPC エージェントを参照してください。gNOI RPC 要求を送信するには、各 RPC に gNOI クライアント インターフェイスを実装するクライアントが必要です。Cisco NX-OS リリース 10.1(1) では、gNOI は限られた数のコンポーネントに対してリモートプロシージャコール（RPC）を定義しており、その一部はハードウェア（光インターフェイスなど）に関連しています。

Proto ファイルは gRPC マイクロサービス用に定義されており、GitHub で入手できます。
https://github.com/openconfig/gnoi

表 *1*：サポートされる *gNOI RPC*

プロトコル	gNOI RPC	サポートあり
System	ping	○
	トレースルート	はい
	時間	はい
	SetPackage	はい
	SwitchControl プロセッサ	はい
	リブート	はい
	RebootStatus	はい
	CancelReboot	はい
OS	アクティブ化	はい
	インターフェイス	はい
Cert	LoadCertificate	はい
File	get	はい
	Put	はい
	Stat	はい
	削除	はい
FactoryReset	開始	はい
Containerz	[展開（Deploy）]	対応
	ListImage	はい
	RemoveImage	はい
	RemoveContainer	はい
	ListContainer	はい
	StartContainer	はい
	StopContainer	はい
	UpdateContainer	はい
	Log	はい
	CreateVolume	はい
	RemoveVolume	はい
	ListVolume	はい

gNOI のガイドラインと制限事項

gNOI 機能には、次の注意事項と制約事項があります。

- 最大 16 のアクティブな gNOI RPC がサポートされます。
- Cisco Nexus 9000 シリーズ スイッチは、1 つの gNMI サービスと 2 つの gNOI マイクロサービスを持つ 1 つのエンドポイントを実行します。

（注） このドキュメントは、gNOIクライアントの例を提供します。参照されるクライアントは、gNOI要求と応答の raw 交換を示す Python スクリプトです。ユーザーは、関心がある自分のクライアントを使用するものとします。

gNOI の構成

gNMI は、gRPC エージェントの子機能です。gRPC エージェントを有効にするには、 gRPC エージェント を参照してください。現在、gNOI の個別の構成はありません。

現在、gNOI の個別の構成はありません。

System .Proto

システム proto サービスは、設定およびテレメトリ パイプラインの外部でターゲットを管理できるようにする操作可能な RPC のコレクションです。

次に、システム proto の RPC サポートの詳細を示します。

RPC	サポート	説明（Description）	制限事項
ping	ping/ping6 cli コマンド	ターゲットで ping コマンドを実行し、結果をストリームバックします。一部のターゲットでは、すべての結果が使用可能になるまで結果がストリーミングされない場合があります。パケット数が明示的に指定されていない場合は、ping5 が使用されます。	do_not_resolve オプションはサポートされていません。

RPC	サポート	説明（Description）	制限事項
トレースルート	traceroute/traceroute6 cli コマンド	ターゲットで traceroute コマンドを実行し、結果をストリームバックします。一部のターゲットでは、すべての結果が使用可能になるまで結果がストリーミングされない場合があります。最大ホップカウント 30 が使用されます。	itial_ttl、marx_ttl、wait、do_not_fragment、do_not_resolve、および l4protocol オプションはサポートされていません。
時間	ローカル時刻	ターゲットの現在の時刻を返します。通常、ターゲットが応答しているかどうかをテストするために使用されます。	-
SetPackage	install{ add | activate } install all nxos	ターゲットのインストールコマンドを実行します。これは、OS 画像または RPM のいずれかのインストールをサポートします。	bootflash: ファイルシステム上の RPM または OS イメージのみをサポートします。
SwitchControl プロセッサ	system switchover cli コマンド	現在のルートプロセッサから指定されたルートプロセッサに切り替えます。スイッチオーバーは即座に発生します。応答がクライアントに返されることが保証されない場合があります。	スイッチオーバーは即座に発生します。その結果、応答がクライアントに返されることが保証されない場合があります。
リブート	reload module	ターゲットをリブートします。	message オプションはサポートされません。delay オプションはスイッチのリロードでサポートされます。path オプションは 1 つのモジュール番号を受け入れます。

RPC	サポート	説明（Description）	制限事項
RebootStatus	show version [module] cli コマンド	ターゲットのリブートのステータスを返します。	-
CancelReboot	reload cancel	保留中の再起動要求をキャンセルします。	-

SetPackage

gNOI setpackage RPC は、ソフトウェア パッケージをスイッチにコピーし、必要に応じてアクティブ化できるメカニズムを提供します。ソフトウェアパッケージは、RPC発信者から直接コピーすることも、リモート ダウンロードを指定することもできます。いずれの場合も、パッケージをインストールする前に、ハッシュを使用してパッケージの転送が確認されます。

setpackage RPC と関連するすべてのメッセージについての完全な定義は、次を参照してください：

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

この RPC の目的は、ソフトウェア パッケージをスイッチにインストールすることです。ソフトウェア パッケージは、RPM またはブート可能 NXOS システム イメージのいずれかです。RPMパッケージは、シスコの署名付き RPM またはサードパーティ RPM のいずれかです。サードパーティ RPM の場合、インストール前に次の構成が必要です。

- `system software allow third-party`

SetPackage のガイドラインと制約事項

次に、SetPackage に関するガイドラインと制限事項を示します：

- 標準規格の NXOS システム イメージと RPM パッケージのみがサポートされています。その他のファイル タイプは拒否され、RPC に失敗します。
- パッケージは「bootflash:」ファイルシステムにのみコピーできます。
- NXOS システム イメージがサード パーティのアプリケーション パッケージと組み合わされているバンドル イメージはサポートされません。
- RemoteDownload.source_address オプションのサポートなし
- Package.filename は標準規格のNXOS ファイル命名規則に準拠する必要があり、「bootflash:」にのみ存在する必要があります。
- Package.version は空であるか、バージョン文字列に設定されます。空の場合、チェックは行われません。設定されている場合、このバージョンの文字列はインストールされているパッケージの文字列に一致する必要があります。

- SSH/SFTP を使用するリモートコピー操作では、パスワードなしの ssh を使用する必要があります。これは管理者が構成する必要があります。これは、RPC 操作の範囲外です。RemoteDownload で指定されたパスワード。ログイン情報は拒否され、RPC に失敗します。
- HTTP(S) を使用するリモートコピー操作では、プロキシの使用が必要になる場合があります。これには、/etc/.curlrc での手動構成が必要です。
- リモートコピーは、RemoteDownload.source_vrf に特に指定されていない限り、VRF が「デフォルト」であると想定します。

RPC オプション

次の表に、パッケージメッセージのフィールドと値を示します。この情報はクライアントの実装に依存しないことに注意してください。これは protobuf メッセージとそれらに含まれるデータの形式にすぎません。各 protobuf メッセージを適切に構築するのはクライアントの責任です。

オプション	説明	値
string filename	スイッチ上のパッケージの宛先ファイル名	bootflash: 上の有効な NXOS ファイル名。
string version	パッケージ バージョンの文字列。	空にするか、パッケージバージョンに設定する可能性があります
bool activate	パッケージをインストールのみ、またはインストールパッケージを追加でアクティブ化するかどうかを指定します。デフォルトは ‘‘false’ です。	**activate = false および OS image** スイッチをリロードせずにイメージをインストール **activate = false および RPM** RPM を有効にせずにリポジトリにインストールする **activate = true および OS image** イメージをインストールし、スイッチをリロードする **activate = true および RPM** RPM をインストールして有効にする NX OS SMU RPM の場合、スイッチをリロードする場合とリロードしない場合があります。これは、SMU の影響範囲によって決定されます。SMU RPM のリリース情報を参照してください。

remote_download	パッケージを別のサーバーの場所から取得するように指定します。	非インタラクティブなリモートコピーをのみサポートします：パスワードなしの SSH が設定されている必要があります。RemoteDownload メッセージで提供されるリモート情報。

次の表は、RemoteDownload メッセージを示しています。

オプション	説明	値
path	取得元のホストとパスの情報を指定します。ホスト名または IP を使用できます。	http(s) : example.com/path/to/package.rpm sftp/scp: a.b.c.d:/path/to/package.rpm
protocol	リモート コピー プロトコル	SFTP、SCP、HTTP、HTTPS。パスワードレスである必要があります（scp/sftp）。
source_address	サポート対象外	
source_vrf	リモート コピーを実行する vrf。	指定されていない場合、「デフォルト」が使用されます
credentials	コピーのためにリモート リソースにアクセスするためのクレデンシャル	必要な場合、ユーザー名とパスワード
• ユーザー名	リモート サーバーのコピー操作に使用するユーザー名	リモート コピー操作には必須です。
• パスワード	必要な場合のパスワード。クリアテキストまたはハッシュ	パスワードが必要な場合（http（s））。scp/sftp では、パスワードレスの ssh キーが必要なため、必要ありません。

（注） **プロキシ**：http/https を介したリモートダウンロードの場合、プロキシを介してのみダウンロードが可能なシナリオが存在します。このような場合、ユーザーは curl プロキシ情報を更新する必要があります。

次の例を参照してください。

これにより、リモートダウンロードが指定されたプロキシを通過できるようになり、そのような構成はスイッチのリロード後も保持されます。

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

（注） **応答とスイッチのリロードのタイミング** ：SetPackage がスイッチをリロードすると、場合によってはスイッチがすぐにリロードされ、ネットワーク接続が中断されることがあります。クライアントが gNOI 応答を受信できない可能性があることが予想されます。

次に、setpackage RPC を使用してスイッチにソフトウェアをインストールするクライアントのインタラクションの例を示します。これらは、さまざまなクライアントを使用して RPC の使用方法を説明する例です。他のクライアントは同様のオプションを使用しますが、詳細は異なります。

例：RPM のインストール、アクティブ化なし

```
./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:/bootflash/nxos64-cs.10.6.1.IQD9.0.29.F.bin,package_file:bootflash:nxos64-cs.10.6.1.IQD9.0.29.F.bin,chunk_size:6000,hash_method:MD5,hash:0ad2bf5c3b0be5b7363ac1e18fcc5f8f,package_activate_file


[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
```

例： OS イメージのインストール、アクティブ化

```
./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"
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
```

例： OS イメージのインストール、リモート ダウンロード

```
./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
```

例：RPM のインストール、アクティブ化なし

```
> 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
```

例：RPM のインストール、アクティブ化

```
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:/tmp/valgrind-3.14.0-r0.corei7_64.rpm,remote_file:bootflash:valgrind-3.14.0-r0.corei7_64.rpm,chunk_size:4000,hash_method:3,hash_sum:02cb0e6a9b41229db1d560dbe83ecce6,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

OS サービスは、ターゲット上の OS インストールに対するインターフェイスを提供します。OS パッケージのファイル形式は、プラットフォームによって異なります。プラットフォームは、提供された OS パッケージが有効でブート可能であることを検証する必要があります。これには、既知の良好なハッシュに対するハッシュチェックを含める必要があります。ハッシュは OS パッケージに埋め込むことをお勧めします。

ターゲットは、独自の永続ストレージと OS インストールプロセスを管理します。一連の個別の OS パッケージを保存し、着信する新しい OS パッケージ用に常にプロアクティブにスペースを解放します。ターゲットには、有効な着信 OS パッケージ用の十分なスペースが常にあることが保証されます。現在実行中の OS パッケージは削除しないでください。クライアントは、最後に正常にインストールされたパッケージが使用可能であることを期待する必要があります。

次に、OS プロトコルの RPC サポートの詳細を示します。

RPC	サポート	説明（Description）	制限事項
アクティブ化	install all nxos bootflash:///img_name	要求された OS バージョンを、次回のリブート時に使用されるバージョンとして設定します。この RPC は、ターゲットを再起動します。	再起動に失敗した場合は、ロールバックまたは回復できません。
検証	show version	[検証（Verify）] は、実行中の OS バージョンを確認します。この RPC は、ターゲットの起動中に成功するまで複数回呼び出される場合があります。	-

（注） インストール RPC はサポートされていません。

Cert .Proto

証明書管理サービスは、ターゲットによってエクスポートされます。ローテーション、インストール、およびその他の証明書プロトコル RPC はサポートされていません。

RPC	サポート	説明（Description）	制限事項
LoadCertificate	crypto ca import <trustpoint> pkcs12 <file> <passphrase>	CA 証明書のバンドルをロードします。	-

ファイル proto

ファイル proto は、file.proto RPC の機能に基づいてメッセージをストリーミングします。

Get、Stat、および Remove RPC は、bootflash、bootflash://sup-remote、logflash、logflash://sup-remote、usb、volatile、volatile://sup-remote、および debug のファイル システムをサポートします。Put RPC は bootflash のみをサポートしています。

次に、ファイル proto の RPC サポートの詳細を示します。

RPC	サポート	説明（Description）	制限事項
結果		Get はターゲットからファイルの内容を読み取り、ストリーミングします。ファイルは連続したメッセージによってストリーミングされます。各メッセージには最大 64 KB のデータが含まれます。最後のメッセージが送信された後、送信されたデータのハッシュが送信され、ストリームが閉じられます。ファイルが存在しない場合、またはファイルの読み取り中にエラーが発生した場合は、エラーが返されます。	ファイルサイズの上限は 32 MB です。

RPC	サポート	説明（Description）	制限事項
Put		ターゲットスイッチにファイルをアップロードします。ファイルの接続先は、NXOS 命名構文に従います。ファイルタイプに制限はありません。	最大ファイル サイズ（Maximum file size (MB)） bootflash のみ：サポートされています
Stat		Stat は、ターゲット上のファイルに関するメタデータを返します。ファイルが存在しない場合、またはファイルのメタデータへのアクセス中にエラーが発生した場合は、エラーが返されます。	-
削除		Remove は、ターゲットから指定されたファイルを削除します。ファイルが存在しない場合、ディレクトリである場合、または削除操作でエラーが発生した場合は、エラーが返されます。	-

Put

Put操作により、ユーザはファイルをスイッチにアップロードできます。「remote_file」フィールドは、NX-OS の「copy」コマンドで必要な命名規則に従う必要があります。たとえば、「bootflash:example.txt」などです。スイッチにコピーされるファイルには、次の制限が適用されます。

- 3.5GB の最大ファイル サイズ
- ターゲット ファイルシステムはローカル bootflash: である必要があります。
- RPC ユーザーには、ターゲットファイルへの書き込み権限が必要です。

PutRequest メッセージの形式は、次の URL で詳しく説明されています：
https://github.com/openconfig/gnoi/blob/main/file/file.proto#L78

例

```
> 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

この .proto は現在、1 つの RPC のみを定義しています。「https://github.com/openconfig/gnoi/blob/master/factory_reset/factory_reset.proto」を参照してください。

RPC	サポート	説明（Description）	制限事項
FactoryReset	factory-reset module all [bypass-secure-erase] preserve-image force	ターゲットで factory-rest コマンドを実行します。	詳細については、後述の内容を参照してください。

FactoryReset

gNOI の初期設定へのリセット操作を行うと、指定されたモジュールのすべての永続ストレージが消去されます。これには、構成、すべてのログデータ、およびフラッシュと SSD（ソリッドステートドライブ）のすべての内容が含まれます。リセットは直前のブートイメージでブートし、ライセンスを含むすべてのストレージを消去します。 gNOI の初期設定へのリセットは、次の 2 つのモードをサポートしています。

- 再フォーマットと再パーティションのみが可能な高速消去。
- データをセキュアに消去してワイプし、回復不可能にする、セキュア消去。

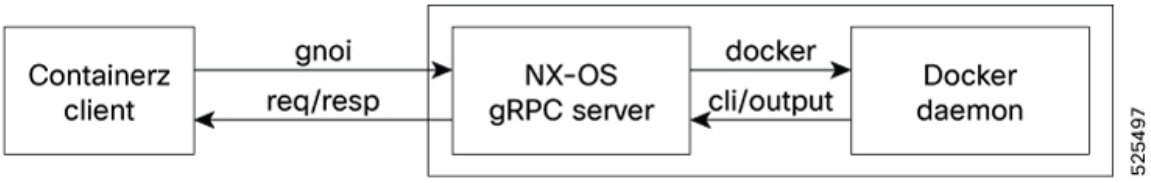
オプション	Description	値
factory_os	工場出荷時の OS バージョンにロールバックするかどうかを指定します。	NX-OS では **true** に設定することはサポートされていません。現在のブートイメージを保持する必要があります。
zero_fill	時間のかかる、全面的なセキュア消去を実行するかどうかを指定します。	**zero_fill = true**：factory-reset module all preserve-image force を指定します。 **zero_fill = false**：factory-reset module all bypass-secure-erase preserve-image force を指定します。

Containerz .Proto

NX-OS は、特権 bash シェルを介したネイティブの Docker アクセスをサポートしています。詳細については、『Cisco Nexus 9000 シリーズ NX-OS プログラマビリティ ガイド』の「Cisco NX-OS での Docker を使用」の章 を参照してください。

この gNOI コンテナは、既存の NX-OS Docker 機能に関する GRPC-ish ラッパー インターフェイスをさらに提供します。containerz API インターフェイスを利用するには、ユーザーは最初に上記のガイドラインに従って Docker サービスを起動してプロビジョニングする必要があります。ネイティブの Docker サービスが実行され、NX-OS bash シェル経由でアクセスできるようになると、ユーザーはコンテナを使用して Docker サービスを管理できます。NX-OS は、containerz 要求を受信すると、与えられた gnoi 要求を対応する docker コマンドに変換して呼び出します。docker コマンド出力は、それぞれの containerz 応答形式に変換されてクライアントに配信されます。

図 *1 :*



次に、Containerz proto の RPC サポートの詳細を示します：

RPC	サポート	説明（Description）
展開	docker image load	tar 形式の docker イメージをアップロードし ポジトリにインストールする
ListImage	docker image ls	デプロイされたイメージの一覧表示
RemoveImage	docker image rm	イメージの削除
RemoveContainer	docker container rm	コンテナの削除
ListContainer	docker container ls	コンテナの一覧表示（List containers）
StartContainer	docker container run docker container start	新しいコンテナまたは停止したコンテナを開 る
StopContainer	docker container stop docker container restart	コンテナの停止または再起動
UpdateContainer	docker container stop docker container start	コンテナの更新
Log	docker container logs	コンテナ ログを取得する
CreateVolume	docker volume create	docker ボリュームの作成
RemoveVolume	docker volume rm	docker ボリュームの削除
ListVolume	docker volume ls	Docker ボリュームのリスト

（注） StartPlugin、StopPlugin、ListPlugin、および RemovePlugin RPC はサポートされていません。

<Deploy>

展開操作により、ユーザーはDockerイメージ(「tar」または「tar.gz」形式)をスイッチにアップロードし、スイッチのリポジトリに登録できます。

展開 RPC については、次の場所に記載されています：

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

展開 RPC の目的は、イメージアーカイブを Docker イメージレジストリにロードすることです。展開 RPC は、「docker load」コマンドラインと同等の機能を提供します。イメージアーカイブファイルは、GZIP または TAR ファイルで、有効な Docker イメージが含まれている必要があります。

遅延展開に関するガイドラインおよび制限事項

展開操作に関するガイドラインと制限事項は次のとおりです。

- 展開するファイルは、Gzip または TAR 形式の有効な Docker イメージである必要があります。
- プラグインは、サポートされません。
- 非インタラクティブなリモートコピーのみを実行できます。つまり、 SCP/SFTP の場合、管理者は RPC を実行する前にパスワードレス SSH をセットアップ必要があります。
- リモートダウンロードの場合、スイッチはImageTransferReadyを送信せず、クライアントは ImageTransferEnd を送信しないと見なされます。

RPC オプション

RPC の実行中に次の protobuf メッセージが交換されます（詳細については、RPC プロトコルの定義を参照してください）。これらのメッセージは、クライアントによってスイッチに送信されます。

ImageTransfer メッセージは、展開操作を開始するためにクライアントからスイッチに送信される最初のメッセージです。

メッセージ オプション	説明	値
文字列の名前	インストールしたイメージに付ける名前	有効な Doc
文字列のタグ	インストール後に画像に適用されるタグ	有効な dock
Unit64 image_size	インストールされるイメージのサイズ（バイト）。これは、イメージをロードするのに十分な空き領域があることを確認するためにチェックされます。	ユーザーは ます。この Size」で確 Docker パー なりません
Bool is_plugin	サポート対象外	いいえ（Fa

メッセージ オプション	説明	値
RemoteDownload remote_download		スイッチ る場合に ブなリモ パスワー があります 供される

RemoteDownload メッセージを示す表

オプション	説明	値
path	取得元のホストとパスの情報を指定します。ホスト名または IP を使用できます。	http(s) : example sftp/scp: a.b.c.d:/p
protocol	リモート コピー プロトコル	SFTP、S である必
source_address	サポート対象外	
source_vrf	リモート コピーを実行する vrf。	指定され されます
credentials	コピーのためにリモート リソースにアクセスするためのクレデンシャル	必要な場
• ユーザ名	リモート サーバーのコピー操作に使用するユーザー名	リモート
• パスワード	必要な場合のパスワード。クリアテキストまたはハッシュ	パスワー では、パ 必要あり

バイトメッセージ

オプション	説明	値
content	このメッセージは、スイッチに直接コピーを行う場合にデータを伝送します。	コピーさ raw バイ

ImageTransferEnd メッセージ

オプション	説明	値
<none>	これは、スイッチに転送されたバイトの終了を通知する空のメッセージです。	このメッ

次の protobuf メッセージは、DeployResponse メッセージの形式でスイッチからクライアントに送信されます。

- ImageTransferReady：スイッチはデータを受信する準備が整っています。
- ImageTransferProgress：これまでに受信したバイト数をクライアントに示します。
- ImageTransferSuccess -- クライアントへの転送が成功したことを示します
- Google.rpc.Status：RPC ステータス コード

ImageTransferReady メッセージ

オプション	説明	値
chunk_size	各コンテンツ メッセージで伝送するデータ量をクライアントに示します。	常に 64KB

ImageTransferProgress メッセージ

オプション	説明	値
Bytes_received	データ コピーでこれまでにスイッチに正常に転送されたバイト数	2 MB の受信れた合計バト コピーの

ImageTransferSuccess メッセージ

オプション	説明	値
名前	レジストリにロードされたイメージの名前	文字列の D
tag	このイメージに使用されたタグ	文字列とし
image_size	読み込まれた画像サイズ	バイト単位

クライアント RPC インタラクションの例

次の例は、特定のクライアントを使用した展開 RPC の使用方法を示しています。詳細は他のクライアントによって異なりますが、これは基本的な相互作用を示しています。

例：直接アップロード

```
./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
```

例：リモート展開

```
(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>

ListImage操作は、スイッチのローカルリポジトリ内のイメージのリストを返します。各イメージには、イメージID、名前、タグなどの情報が含まれます。

ListImage 操作は、次のオプションのどれでもサポートします。

オプション	Description	値
Limit	「limit」は、「docker image list」コマンドと同じ表示順序に従って、応答で返されるイメージエントリの数を制限します。	int32 として
Filter {key:value}	ListImage操作は、フィルタに一致するすべての画像を返します	キーと値の

次の例は、ListImage 操作の使用方法を示しています。

```
> 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>

RemoveImage 操作は、名前で識別される Docker コンテナを削除します。

デフォルトでは、イメージが一部のコンテナによって使用されている場合、操作は失敗します。「force」オプションを使用すると、イメージを強制的に停止して削除できます。ただし、この場合、docker はイメージのタグを解除するだけです。

RemoveImage 操作は、次のオプションをサポートします。

オプション	説明	値
Name	削除するイメージ名	文字列とし
タグ	削除するタグ	有効な文字
強制	「force=true」の場合、イメージは強制的に削除/タグなしにされます	True/False。います。

次の例は、RemoveImage 操作の使用方法を示しています。

```
> 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"
```

次の例は、強制削除（タグ解除のみ）を示しています。

```
> 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"
```

次の例は、コンテナが停止した後に再度強制削除することを示しています。

```
> 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>

StartContainer操作は、新しいコンテナを開始するか、以前に停止したコンテナを再起動するために使用できます。判断は、次の条件に基づきます。

- コンテナが存在しない場合は、新しいコンテナを開始します
- otherwise
  - コンテナがすでに実行されている場合は、エラーを返します。
  - コンテナが停止しているが、リクエストにインスタンス名だけではない多くのフィールドが含まれている場合は、エラーを返します。
  - コンテナが停止し、リクエストにインスタンス名のみが含まれている場合は、コンテナを開始します。

StartContainer は次のオプションをサポートしています

オプション	Description	値
image_name	コンテナを起動するイメージ名	文字列と

tag	コンテナを開始するイメージ タグ	有効な文字
cmd	新しいコンテナで実行するコマンド	有効なコマ
instance_name	新しいコンテナの識別子	文字列とし
port -内部 -外部	コンテナの外部に公開される内部ポートのリスト	有効な内部 例：内部ポ 外部ポート
環境 =<VAR1> ：<value>	コンテナでの環境変数の設定	文字列とし ペアだけに 例：VAR1
ネットワーク （network）	このコンテナを接続するネットワーク	有効なネッ これは、「 ランタイム クです。
Cap.ADD	追加する機能	有効な機能 https://man7.o で確認でき トです。 AUDIT_WR FOWNER、 NET_BIND SETFCAP、 SYS_CHRO
Cap.REMOVE	削除される機能	
Restart.policy	再起動ポリシーの設定	有効な再起 NONE、AL ON_FAILU
Restart.attempts	再起動の試行を設定	unit32とし
RunAs.user RunAs.group	このコンテナを実行するユーザーとグループを設定	文字列とし
ラベル（Label） <key>:<val>	メタデータをコンテナに設定するためのキーと値のペア	文字列ラベ にすること 例: my-label
Limits.max_cpu	このコンテナで使用できる最大CPUの数を設定。	有効な CPU 例：値 0.5 CPU の半分

Limits.soft_mem_bytes	メモリのソフト制限を設定	有効な は、使 必要が 例：soft
Limits.hard_mem_bytes	メモリのハード制限を設定	有効な 例：limi

ガイドラインと制約事項

- StartContainer 操作は、「docker run」 CLI を介して実現されます。この CLIでサポートされる最大長は 2500 バイトです。
- RunAs の場合、現在の Docker は UID と GID のみをサポートします。コンテナは UID と GID の任意の値で開始し、ユーザー名とグループ名が指定されると失敗します。
- コンテナの起動中に警告が発生した場合、NXOS は警告を返さずにコンテナを起動します。

次の例は、StartContainer 操作を示しています。

例：新しいコンテナの開始

```
> 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"
}
```

例：新しいコンテナの開始

```
> 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"
}
```

例：既存の実行中のコンテナの開始

```
> 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"
}
```

例：既存の停止したコンテナの起動

```
> 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>

RemoveContainer 操作は、名前で識別される Docker コンテナを削除します。

デフォルトでは、コンテナがまだ実行中の場合、操作は失敗します。**force** オプションを使用すると、コンテナを強制的に停止して削除できます。

RemoveContainer 操作では、次のオプションがサポートされています。

オプション	Description	値
Name	削除するコンテナ名	文字列と
強制	「force=True」の場合、コンテナは停止し、強制的に削除されます	True/Fals います。

Dockerイメージとコンテナの前には一定の独立性があるため、基礎となるイメージが削除された場合でも、コンテナは存在し続ける可能性があることを認識することが重要です。ユーザーはイメージとコンテナを慎重に削除する必要があります。

例

```
> 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"
```

例：強制削除

```
> 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>

ListContainer 操作は、スイッチのローカル リポジトリ内のコンテナのリストを返します。

各コンテナには、イメージ名、コンテナ ID、名前、実行ステータス、コンテナランタイムによって計算されたラベルとイメージハッシュを含むメタデータなどの情報が含まれます。

ListContainer 操作では、次のオプションがサポートされます

オプション	Description	値
すべて	「all」オプションを使用すると、実行中のコンテナと実行されていないコンテナをすべて返すことができます。デフォルトでは、ListContainer 操作は実行中のコンテナのみを返します。	True/False。ています。
Limit	「limit」は、「docker container list」コマンドと同じ表示順序に従って、応答で返されるコンテナ エントリの数を制限します。	int32 として
Filter {key:value}	ListImage操作は、フィルタに一致するすべての画像を返します	キーと値の

例

```
> 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"
 }
```

例：すべてのコンテナ

```
> 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>

StopContainer 操作は、名前で識別される Docker コンテナを停止します。これは、「docker container stop」コマンドを介して実行されます。「restart」オプションを指定した場合、「docker chain restart」コマンドを使用して docker コンテナも再起動します。

StopContainer 操作は次のオプションをサポートしています

オプション	Description	値
Instance_name	停止するコンテナ インスタンス	文字列と
force	このオプションが true の場合、コンテナは強制的に停止されます	True/Fals います。
再起動	このオプションが true の場合、コンテナは「docker container restart」を使用して再起動されます。	True/Fals います。

制限と不具合

- 実行中のコンテナが StopContainer 操作を使用して停止されると、エラーを返さずにコンテナを停止するのに 10 秒かかります。
- 実行中のコンテナを「force」オプションで停止すると、コンテナは5秒後に停止します。

次の例は、StopContainer 操作を示しています。

例

```
> 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>

UpdateContainer 操作は、新しいイメージと引数でコンテナ インスタンスを更新します。更新が失敗すると、コンテナは以前の状態に復元されます。

現在、ネイティブのDockerはこの操作をサポートしていないため、NX-OSはシーケンスDocker操作を介してこの機能を実現することに注意してください。

1. 現在実行中の場合は、指定されたコンテナを停止する
2. コンテナの名前をバックアップとしての一時的な名前に変更します
3. 指定された新しいイメージと引数を使用して、同じコンテナ名で新しいコンテナを開始します。
4. 新しいコンテナの実行に成功した場合は、バックアップ コンテナを削除します。
5. 新しいコンテナの実行に失敗した場合は、復元を試みます
   - 新しいコンテナを停止して削除します。
   - バックアップを元のコンテナ名に戻します
   - 実行していた場合、コンテナを開始します。

UpdateContainer 操作では、次のオプションがサポートされています。

オプション	Description	値
instance_name	更新する実行コンテナの名前。	文字列とし
image_name	コンテナを更新する イメージ。	文字列とし
image_tag	コンテナを更新するためのタグ。	文字列とし
async	この操作を非同期で実行するかどうか	True/False。います。
コンテナを実行するための startContainer 操作でサポートされているすべてのオプション。		

例

```
> 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>

ログ操作により、「docker container logs」コマンドと同様に、クライアントはコンテナ ログを取得できます。操作で、ログを「フォローする」かどうかを選択できます。

- Follow=False の場合、スイッチは現在のログをフェッチして返します。
- Follow=True の場合、スイッチは現在のログを返し、RPC がクライアントによってキャンセルされるか、コンテナが停止するまで、新しいログをストリーミングします。

ログ操作は、次のオプションのどれでもサポートします

オプション	Description	値
instance_name	コンテナ名	文字列と
follow	設定されている場合、クライアントがキャンセルするまで、ストリームは開いたままになります。	True/Fals ていま

例

```
> 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
```

例：ログをたどる

```
> 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>

CreateVolume 操作では、名前で識別される Docker ボリュームが作成されます。

ユーザーは、このボリュームを複数のコンテナにマウントできます。

CreateVolume 操作は、次のオプションのどれでもサポートします

オプション	Description	値
名前	ボリュームの名前	文字列とし
ドライバ	ボリューム ドライバ	NXOS は「 します。
オプション	ドライバのオプション。実際のオプション キーと値はドライバ固有です。	NXOS はこ せん。
ラベル	ボリュームに適用するラベル。ラベルはボリュームのメタデータです。	文字列ラベ にすること 例: my-label

注意事項と制限事項

- NXOS は、「LOCAL」のドライバ ro create ボリュームのみをサポートします。
- NXOS は「ローカル マウント オプション」をサポートしていません。

例

```
> 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>

RemoveVolume 操作は、名前で識別される Docker ボリュームを削除します。

RemoveVolume 操作は次のオプションをサポートしています

オプション	Description	値
名前	削除するボリューム名	有効なボリ
force	ボリュームを強制的に削除する	NXOSでサ

ガイドラインと制約事項

- 「force」オプションはdockerではサポートされていないため、ボリュームを正常に削除できるように、すべてのコンテナからボリュームを切り離してください

例

```
> 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>

ListVolume 操作は Docker ボリュームのリストを返します。

これには、ボリューム名、作成されたタイムスタンプ、ドライバ、およびボリュームのオプションと該当する場合、ラベルが含まれます。

ListVolume 操作は次のオプションをサポートします

オプション	Description	値
Filter {key:value}	ListVolume 操作では、フィルタに一致するすべてのボリュームが返されます	キーと値

例

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

[REQ]

[RESP] : 1
 name: "92945dc4e9ffdf571c85994738562b1d1f54158f784cac3eadc080c558e034ee"
 created {
   seconds: 1748490463
 }
 driver: "local"
```

gNOI のトラブルシューティング

gNOI のデバッグ

gNOI のステータスを確認するには、次のコマンドを入力します。

show コマンド

コマンド	説明
clear grpc gnoi rpc	カウンタまたは呼び出しをクリーンアップするために使用されます。
debug grpc events {**events**|**errors**} **show grpc nxsdk event-history** {**events**|**errors**}	イベント履歴からイベントとエラーをデバッグします。
clear grpc gnoi rpc	カウンタまたは呼び出しをクリーンアップするために使用されます。

出力例

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 コマンド

特に、containerz については、「Cisco Nexus 9000 シリーズ NX-OS プログラマビリティ ガイド」の「Cisco NX-OS で Docker を使用」の章のデバッグ手順を参照してください。基本的に、ユーザーは Docker ユーティリティに直接アクセスして、デバッグ目的で各イメージとコンテナの詳細を調べることができるものとします。

コマンド	説明
docker image list	docker イメージをリスト
docker container list	Docker コンテナをリスト
docker volume list	Docker ボリュームをリスト

出力例

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
```

デバッグ ログの収集

gNOI は、gRPC エージェントの子サービスです。詳細については、「gRPC エージェント」の章を参照してください。

Docker ログ

特に、containerzについては、次の場所のログを調査してください。このファイルには、Docker デーモンのデバッグ メッセージが維持されます。詳細については、https://docs.docker.com/engine/daemon/logs/ を参照してください。

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

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。