# gRPC ネットワーク操作インターフェイス

Google リモートプロシージャコール（gRPC） ネットワーク操作インターフェイス（gNOI）は一連のマイクロサービスであり、それぞれが一連の操作に対応しています。このモジュールでは、サポートされている gNOI サービスについて説明します。


- gRPC ネットワーク操作インターフェイスに関する情報 （1 ページ）
- gRPC ネットワーク操作インターフェイスに関する追加情報 （19 ページ）
- gRPC ネットワーク操作インターフェイスの機能情報 （20 ページ）


# gRPC ネットワーク操作インターフェイスに関する情報

## gNOI プロトコル

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。gNMIサービスは、設定管理、動作状態の取得、およびストリーミングテレメトリによるバルクデータ収集の動作を定義します。 gNOI では、デバイスがサポートするサービスのみを採用できます。gNOIは、OSインストールサービスをサポートします。

gNOI は、ユーザ認証の有無にかかわらず使用できます。ユーザ認証はデフォルトでディセーブルになっています。**gnxi secure-password-auth** コマンドを使用してユーザ認証を有効にします。OpenConfig モデルによるユーザー認証の有効化については、https://github.com/YangModels/yang/blob/master/vendor/cisco/xe/1751/openconfig-system-management.yangを参照してください。

gNOI プロトコルは、次の操作をサポートします。

- 証明書の管理
- ブートストラップ
- OS インストールサービス
- factory-reset サービス

# 証明書管理サービス

証明書管理サービスでは、初めに 2 つの主要な RPC、Install と Rotate がエクスポートされます。これらの RPC はそれぞれ新しい証明書のインストールとデバイス上の既存の証明書のローテーションに使用されます。

証明書管理サービスでは、次の RPC がサポートされています。

- Install：証明書をインストールします。すべての証明書は、証明書 ID によって一意に識別されます。証明書 ID は文字列です。
- Rotate：既存の証明書をローテーションします。
- RevokeCertificates：1 つ以上の証明書を取り消します。
- GetCertificates：すべての証明書を照会します。
- CanGenerateCSR：デバイスが証明書署名要求（CSR）を生成できるかどうかを照会します。

前述の RPC によって作成されたトラストポイントと証明書は、スイッチオーバーおよびデバイスのリブート後も保持されます。

次に、証明書管理サービスの定義の例を示します。

```
service CertificateManagement {
 rpc Install(stream InstallCertificateRequest)
  returns (stream InstallCertificateResponse);

 rpc Rotate(stream RotateCertificateRequest)
  returns (stream RotateCertificateResponse);

 rpc RevokeCertificates(RevokeCertificateRequest)
  returns (RevokeCertificateResponse);

 rpc GetCertificates(GetCertificateRequest)
  returns (GetCertificateResponse);

 rpc CanGenerateCSR(CanGenerateCSRRequest)
  returns (CanGenerateCSRResponse);
}
```

## Install RPC

Install RPC は、新しい CSR 要求を作成して、新しい証明書をデバイスに追加します。新しい証明書は、デバイスの新しい証明書 ID に関連付けられます。デバイスに指定された証明書 ID を持つ既存の証明書がある場合、操作は失敗します。

Install RPC は、双方向ストリーミング RPC です。入力（InstallCertificateRequest）と出力（IntsallCertificateResponse）があり、どちらもストリーミングです。ストリームが中断されるか、プロセスのいずれかのステップが失敗すると、デバイスは変更をロールバックします。

次に、Install RPC の定義とメッセージの例を示します。

```
rpc Install(stream InstallCertificateRequest)
returns (stream InstallCertificateResponse);

// Request messages to install new certificates on the target.
message InstallCertificateRequest {
  // Request Messages.
  oneof install_request {
    GenerateCSRRequest generate_csr = 1;
    LoadCertificateRequest load_certificate = 2;
  }
}
// Request to generate the CSR.
message GenerateCSRRequest {
  // Parameters for creating a CSR.
  CSRParams csr_params = 1;
  // The certificate id with which this CSR will be associated. The target
  // configuration should bind an entity which wants to use a certificate to
  // the certificate_id it should use.
  string certificate_id = 2;
}
// Parameters to be used when generating a Certificate Signing Request.
message CSRParams {
  // The type of certificate which will be associated for this CSR.
  CertificateType type = 1;

  // Minimum size of the key to be used by the target when generating a
  // public/private key pair.
  uint32 min_key_size = 2;

  // If provided, the target must use the provided key type. If the target
  // cannot use the algorithm specified in the key_type, it should cancel the
  // stream with an Unimplemented error.
  KeyType key_type = 3;

  // --- common set of parameters applicable for any type of certificate --- //
  string common_name = 4;           // e.g "device.corp.google.com"
  string country = 5;               // e.g "US"
  string state = 6;                 // e.g "CA"
  string city = 7;                  // e.g "Mountain View"
  string organization = 8;          // e.g "Google"
  string organizational_unit = 9;   // e.g "Security"
  string ip_address = 10;
  string email_id = 11;
}
// A certificate.
message Certificate {
  // Type of certificate.
  CertificateType type = 1;

  // Actual certificate.
  // The exact encoding depends upon the type of certificate.
  // for X509, this should be a PEM encoded Certificate.
  bytes certificate = 2;
}

message LoadCertificateRequest {
  // The certificate to be Loaded on the target.
  Certificate certificate = 1;

  // The key pair to be used with the certificate. This is provided in the event
  // that the target cannot generate a CSR (and the corresponding public/private
  // keys).
  KeyPair key_pair = 2;
```

```
  // Certificate Id of the above certificate. This is to be provided only when
  // there is an externally generated key pair.
  string certificate_id = 3;

  // Optional pool of CA certificates to be used for authenticating the client.
  repeated Certificate ca_certificate = 4;
}

// A message representing a pair of public/private keys.
message KeyPair {
  bytes private_key = 1;
  bytes public_key = 2;
}

// Response Messages from the target for the InstallCertificateRequest.
message InstallCertificateResponse {
  // Response messages.
  oneof install_response {
    GenerateCSRResponse generated_csr = 1;
    LoadCertificateResponse load_certificate = 2;
  }
}

// GenerateCSRResponse contains the CSR associated with the Certificate ID
// supplied in the GenerateCSRRequest. When a Certificate is subsequently
// installed on the target in the same streaming RPC session, it must be
// associated to that Certificate ID.
//
// An Unimplemented error will be returned if the target cannot generate a CSR
// as per the request. In this case, the caller must generate its own key pair.
message GenerateCSRResponse {
  CSR csr = 1;
}

// A Certificate Signing Request.
message CSR {
  // Type of certificate.
  CertificateType type = 1;

  // Bytes representing the CSR.
  // The exact encoding depends upon the type of certificate requested.
  // for X509: This should be the PEM encoded CSR.
  bytes csr = 2;
}
```

ターゲットデバイスが起動し、gNOI がデフォルト状態になると、コントローラ（サードパーティの実装）は Install RPC を使用して、認証局（CA）によって署名された証明書をインストールします。証明書は、証明書 ID によって一意に識別されます。この ID は、公開キーインフラストラクチャ（PKI）設定でトラストポイント名として使用されます。既存の証明書 ID を持つ証明書をインストールしようとすると、インストールは失敗します。

次のセクションでは、デバイスによって CSR が生成される方法について説明します。

1. デバイスは、Install RPC を使用して自己署名証明書を生成します。暗号化モード（または gNMI のデフォルト状態）では、コントローラはターゲットデバイスによって提示された証明書を検証しないため、コントローラはこの証明書のコピーを必要としません。これは、デフォルトの状態です。
2. コントローラはデバイスに CSR の生成を要求し、CSR を CA に送信し、CA から署名証明書を取得します。

3. 署名証明書は、証明書の署名に使用される CA 証明書とともにデバイスにインストールされます。CA 証明書は ca_certificates バンドルに存在し、デバイス証明書をインストールするために PKI が要求します。
4. gNMI または gNOI サービスは、プロビジョニングされた状態になった、新しくインストールされた証明書を使用して再起動します。

## Rotate RPC

Rotate RPC により既存の証明書が更新されます。これはすでにインストールされている証明書です。証明書がまだインストールされていない場合、Rotate RPC は失敗します。使用されていない証明書はローテーションできますが、クライアントはそれをテストできません。

次に、Rotate RPC の定義の例を示します。

```
rpc Rotate(stream RotateCertificateRequest)
returns (stream RotateCertificateResponse);

// Request messages to rotate existing certificates on the target.
message RotateCertificateRequest {
  // Request Messages.
  oneof rotate_request {
    GenerateCSRRequest generate_csr = 1;
    LoadCertificateRequest load_certificate = 2;
    FinalizeRequest finalize_rotation = 3;
  }
}

// A Finalize message is sent to the target to confirm the Rotation of
// the certificate and that the certificate should not be rolled back when
// the RPC concludes. The certificate must be rolled back if the target returns
// an error after receiving a Finalize message.
message FinalizeRequest {
}

message RotateCertificateResponse {
  // Response messages.
  oneof rotate_response {
    GenerateCSRResponse generated_csr = 1;
    LoadCertificateResponse load_certificate = 2;
  }
}
```

Rotate RPC は、次の点で Install RPC と異なります。

- PKI は（ロールバックの目的で）新しい証明書をインストールするときに、古い証明書と CA 証明書を保存またはキャッシュする必要があります。
- コントローラは新しい接続を作成し、更新された証明書が機能するかどうかをテストし、成功した場合は証明書のローテーションを完了します。

## Revoke RPC

この RPC は、証明書 ID によって一意に識別される 1 つ以上の証明書を失効させるために使用されます。証明書を失効させると、対応するトラストポイントが Cisco IOS XE の設定から削除されます。対応するトラストポイントが現在使用されている場合、またはトラストポイントが存在しない場合は、証明書の失効が失敗する可能性があります。

RevokeCertificate RPC では、証明書の失効が成功する場合も失敗する場合もあります。ターゲットデバイスでは、失効は単純な削除操作です。 CA による実際の失効はクライアントによって行われます。クライアントが使用中の証明書を失効させた場合、新しい接続は失敗しますが、既存の接続は影響を受けません。

次に、RevokeCertificate RPC の例を示します。

```
// An RPC to revoke specific certificates.
// If a certificate is not present on the target, the request should silently
// succeed. Revoking a certificate should render the existing certificate
// unusable by any endpoints.
rpc RevokeCertificates(RevokeCertificatesRequest)
returns (RevokeCertificatesResponse);

message RevokeCertificatesRequest {
  // Certificates to revoke.
  repeated string certificate_id = 1;
}

message RevokeCertificatesResponse {
  // List of certificates successfully revoked.
  repeated string revoked_certificate_id = 1;

  // List of errors why certain certificates could not be revoked.
  repeated CertificateRevocationError certificate_revocation_error = 2;
}

// An error message indicating why a certificate id could not be revoked.
message CertificateRevocationError {
  string certificate_id = 1;
  string error_message = 2;
}
```

## GetCertificate RPC

この RPC はすべての証明書 ID を照会します。

クエリに対する応答には、次の情報が含まれます。

- 証明書 ID で識別されるすべての証明書の証明書情報。
- この証明書を使用するエンドポイント（トンネル、デーモンなど）のリスト。

（注） サポートされないエンドポイント。

（注） 応答には ca_certificate バンドルは含まれません。

次に、GetCertificate RPC の例を示します。

```
// An RPC to get the certificates on the target.
rpc GetCertificates(GetCertificatesRequest) returns (GetCertificatesResponse);

// The request to query all the certificates on the target.
message GetCertificatesRequest {
}

// Response from the target about the certificates that exist on the target what
// what is using them.
message GetCertificatesResponse {
  repeated CertificateInfo certificate_info = 1;
}

message CertificateInfo {
  string certificate_id = 1;
  Certificate certificate = 2;

  // List of endpoints using this certificate.
  repeated Endpoint endpoints = 3;

  // System modification time when the certificate was installed/rotated in
  // nanoseconds since epoch.
  int64 modification_time = 4;
}

// An endpoint represents an entity on the target which can use a certificate.
message Endpoint {
  // Type of endpoint that can use a cert. This list is to be extended based on
  // conversation with vendors.
  enum Type {
    EP_UNSPECIFIED = 0;
    EP_IPSEC_TUNNEL = 1;
    EP_DAEMON = 2;
  }
  Type type = 1;

  // Human readable identifier for an endpoint.
  string endpoint = 2;
}
```

## CanGenerateCSR RPC

この RPC は、デバイスが特定のキータイプ、証明書タイプ、およびキーサイズの CSR を生成できるかどうかを照会します。サポートされるキータイプは、Rivest、Shamir、およびAdelman（RSA）で、サポートされる証明書タイプは X.509 です。

この RPC 要求が Install RPC の一部として完全に新しい証明書をインストールするために作成されている場合、証明書 ID が新しいものであり、デバイス上のエンティティがこの証明書 ID にバインドされていないことをデバイスで確認する必要があります。既存の証明書が証明書 ID と一致する場合、この要求は失敗します。

この RPC 要求が、Rotate RPC の一部として既存の証明書をローテーションするように作成された場合、証明書 ID がすでに使用可能であることをデバイスで確認する必要があります。証明書のローテーションで証明書のロードを続行する場合は、新しい証明書を以前に作成した証明書 ID に関連付ける必要があります。

次に、CanGenerateCSR RPC の例を示します。

```
// An RPC to ask a target if it can generate a Certificate.
rpc CanGenerateCSR(CanGenerateCSRRequest) returns (CanGenerateCSRResponse);

// A request to ask the target if it can generate key pairs.
message CanGenerateCSRRequest {
  KeyType key_type = 1;
  CertificateType certificate_type = 2;
  uint32 key_size = 3;
}

// Algorithm to be used for generation the key pair.
enum KeyType {
  // 1 - 500, for known types.
  // 501 and onwards for private use.
  KT_UNKNOWN = 0;
  KT_RSA = 1;
}

// Types of certificates.
enum CertificateType {
  // 1 - 500 for public use.
  // 501 onwards for private use.
  CT_UNKNOWN = 0;
  CT_X509 = 1;
}

// Response from the target about whether it can generate a CSR with the given
// parameters.
message CanGenerateCSRResponse {
  bool can_generate = 4;
}
```

# 相互認証

相互認証は双方向認証です。2 つのパーティが同時に相互に認証します。相互認証を有効にするには、**gnmi-yang secure-peer-verify-trustpoint** コマンドを使用します。このコマンドが有効になっていない場合、認証サービスが gNMI クライアントをすべての既存のトラスポイントおよびトラストプールの内容に対して検証します。

相互認証のために CA 証明書をローテーションするには、クライアントがターゲットデバイスに新しいバンドルを提示し、古いバンドルを削除する必要があります。ただし、CA 証明書はトラストプールに存在しており、トラストプールから選択的に削除することはできません。

# 証明書サービスによるブートストラップ

gNOI 証明書をインストールした後、ブートストラップを使用してターゲットデバイスを設定または操作します。ターゲットデバイスに既存の証明書がない場合、gNOI 証明書管理サービスを使用してブートストラップにより証明書をインストールできます。証明書のインストール後、デバイスはセキュアな gNOI 接続または gNMI 接続を確立できます。このプロセスは、既存のセキュアな環境を前提としています。

gNMI ブートストラップを有効にするには、**gnxi secure-int** コマンドを使用します。

（注） gNOI 証明書管理サービスは、ブートストラップの前にインストールする必要があります。

gNOI 証明書管理サービスには 2 種類の状態があります。これらの状態は、gNOI サービスと gNMI サービスの両方でサポートされます。

- Default/Encrypted：デバイス上の gNOI と gNMI は、クライアントが検証しない自己署名（デフォルト）証明書を使用します。証明書は認証を必要としません。この状態では、gNOI 証明書サービスのみがターゲットデバイスで有効になります。
- Provisioned：デバイス上の gNOI および gNMI は、クライアントによって検証されたインストール済み証明書を使用します。クライアントはその証明書を提示し、デバイスは証明書ストアと照合して証明書を検証します。デバイスは、相互認証が有効になっている場合にのみクライアント証明書を検証します。

# OS インストールサービス

OS インストールサービスは、インストールに使用される gNOI API を定義します。OS インストールサービスは、gNOI プロトコルでサポートされています。

このサービスは、OS をデバイスにインストールするためのインターフェイスを提供します。次の 3 つの RPC をサポートしています。

- Install：この RPC はイメージをデバイスに転送します。これらのイメージは、バージョン文字列によって一意に識別されます。この RPC は **install add** コマンドに似ています。主な違いは、イメージが RPC の一部として転送されることです。
- Activate：この RPC は、RPC への入力の一部である要求された OS バージョンを、次回の再起動時に使用されるバージョンとして設定し、デバイスを再起動します。この RPC は、**install activate** および **install commit** コマンドと同じです。
- Verify：この RPC は現在の OS バージョンを確認します。

Cisco IOS XE デバイスは、ソフトウェアイメージの起動で、インストールモードとバンドルモードの両方をサポートします。

インストールモードでは、flash: ファイルシステム内に存在するソフトウェアパッケージのプロビジョニングファイルを起動して、デバイスを起動できます。インストールされている各

パッケージの ISO ファイルシステムは、フラッシュからルートファイルシステム（rootfs）に直接マウントされます。

バンドルモードでは、バンドル（.bin）ファイルを使用してデバイスを起動できます。パッケージはバンドルから取得され、RAM にコピーされます。各パッケージの ISO ファイルシステムは、rootfs にマウントされます。インストールモードでの起動とは異なり、バンドルモードでの起動では、バンドルのサイズに対応するサイズの追加メモリが使用されます。

次のシナリオでは、デバイスがバンドルモードで起動するとエラーメッセージが生成されます。

- デバイスが、バンドルモードで実行している現在のイメージで起動する。
- 新しいイメージをインストールするために、デバイスで Install RPC が開始される。

エラー メッセージの例を次に示します。

```
May 11 09:24:15.385 PST: %INSTALL-3-OPERATION_ERROR_MESSAGE:
Switch 1 R0/0: install_engine: Failed to install_add package
flash:gNOI_iosxe_17.05.01.0.144.1617180620.bin, Error: [2|install_add(ERR, )]:
Booted in bundle mode. For Bundle-to-Install mode conversion,
please use one-shot CLI - install add file <> activate commit
```

エラーメッセージが生成されても、Install RPC はクライアントに成功を返します。エラーメッセージは無視しても問題ありません。後続の Activate RPC は影響を受けません。新しいイメージで再起動すると、デバイスはインストールモードになります。

（注） このエラーメッセージは、デバイスが最初にインストールモードで実行していた場合は表示されません。これは、デバイスがバンドルモードで起動する場合にのみ該当します。

すべてのエラーメッセージを表示するには、https://github.com/openconfig/gnoi/blob/master/os/os.proto#L218 を参照してください。

インストールモードの詳細については、システム管理コンフィギュレーション ガイドの「デバイスのセットアップ設定の実行」の章を参照してください。

### デュアルルートプロセッサのサポート

シスコのデバイスは、インサービス ソフトウェア アップデート（ISSU）（インストールモードのみサポート）と非 ISSU モードの両方をサポートします。ISSU がサポートされていない場合、または Install RPC を介して使用できない場合、gNOI OS インストールサービスは非 ISSU インストールを要求します。

デュアルルートプロセッサ（RP）の場合にデバイスが ISSU アップグレードをサポートする場合、gNOI OS インストール サービス インターフェイスは install activate ISSU ワークフローを呼び出します。ISSU がサポートされていない、またはデバイスが単一の RP をサポートしている、他のすべてのシナリオでは、gNOI OS インストールサービスは通常の非 ISSU イメージ インストール ワークフローを使用して gRPC アクティベート要求を処理します。

バンドルモードでは、**install add file** *filename* **activate commit** コマンドを使用してアップグレードが実行されます。このアップグレードは、単一の RP を持つデバイスの場合も同じです。

ISSU がサポートされていないということは、両方の RP が同時にリロードされ、1 つの RP が起動するまでデバイスがダウンすることを意味します。

ISSU を使用しないインストールモードでは、両方の RP が同時にリロードされ、1 つの RP が起動するまでデバイスがダウンします。ISSU を使用したインストールモードでは、RP のリロードが同時に行われ、デバイスのダウンタイムが短くなります。

## OS Install RPC

Install RPC は、イメージをデバイスに転送します。この RPC は、入力の InstallRequest RPC と出力の InstallResponse RPC で構成されます。どちらも双方向ストリーミング RPC です。

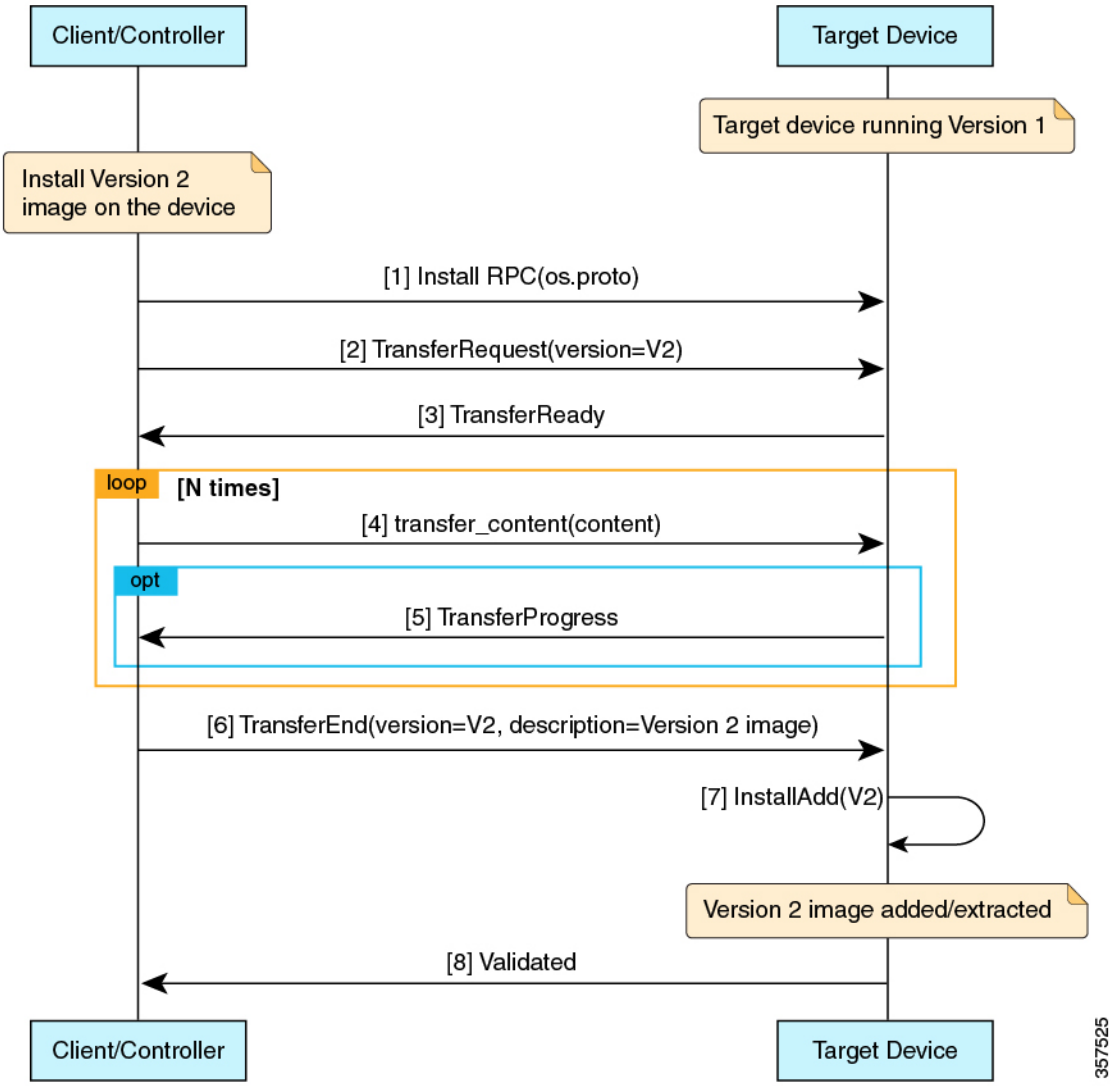
この RPC はソフトウェア メンテナンス アップデート（SMU）をサポートしていません。

次に、単一の RP がオペレーティング システム バージョン 1 を実行しているデバイスでの Install RPC のメッセージシーケンスの概要を示します。

1. クライアントがデバイスへの Install RPC を開始します。
2. クライアントは、バージョンをバージョン 2 に設定した TransferRequest メッセージをデバイスに送信します。
3. デバイスは、TransferReady メッセージでクライアントに応答します。これは、クライアントがイメージの転送を開始するために必要です。
4. クライアントは、複数の transfer_content メッセージをデバイスに送信して、イメージを転送します。
5. オプションで、デバイスはクライアントに TransferProgress メッセージを送信します。
6. クライアントは、イメージ転送が完了したことを示すTransferEnd メッセージをデバイスに送信します。
7. インストールモードでは、デバイスは **install add** コマンドと同等の操作をプログラムで実行します。パッケージの内容が抽出されます。
8. デバイスは、イメージから抽出したバージョンを含む Validated メッセージをクライアントに送信し、イメージ転送が有効であることを示します。

（注） クライアントによって Install RPC が途中で停止した場合、または操作の一部が失敗した場合は、ローカルイメージファイルが削除され、**install remove inactive** コマンドが自動的に呼び出されます。適切なステータスコードがクライアントに返されます。

図 *1*: 単一 *RP* のイメージ インストール ワークフロー




## OS Activate RPC

Activate RPC は、要求されたオペレーティングシステムのバージョンを次回の再起動時に使用するバージョンとして設定し、ターゲットデバイスを再起動します。この RPC は、インストールされたオペレーティングシステムのバージョンをアクティブ化します。指定されたバージョンがまだインストールされていない場合、Activate RPC は失敗します。

クライアントは、Install RPC の Validated メッセージで受信したバージョンを提供する必要があります。

次に、単一の RP がオペレーティング システム バージョン 1 を実行しているデバイスでの Activate RPC のメッセージシーケンスを示します。

1. クライアントは、デバイスに対して Activate RPC を開始します。
2. クライアントは、デバイスにバージョン 2 の ActivateRequest メッセージを送信します。

   このメッセージシーケンスでは、バージョン 2 が Install RPC によってすでにインストールされているものとします。
3. デバイスは、インストールモードの場合は **install activate commit** コマンドに相当するプログラム操作を、バンドルモードの場合は **install add file activate commit** コマンドに相当するプログラム操作を実行します。
4. アクティブ化プロセスでエラーが検出されないため、デバイスはクライアントに ActivateResponse(OK) メッセージで応答します。
5. デバイスにバージョン 2 がリロードされます。
6. リロード後にデバイスが起動すると、**install commit** コマンドと同等のプログラム操作が実行されます。

（注） 1 つの非アクティブイメージバージョンのみがサポートされます。このため、クライアントがバージョン 2 をインストールしてからバージョン 3 をインストールすると、バージョン 2 のファイルが削除されます。

次の図は、イメージのアクティブ化ワークフローを示しています。

図 *2:* シングル *RP* イメージのアクティブ化ワークフロー

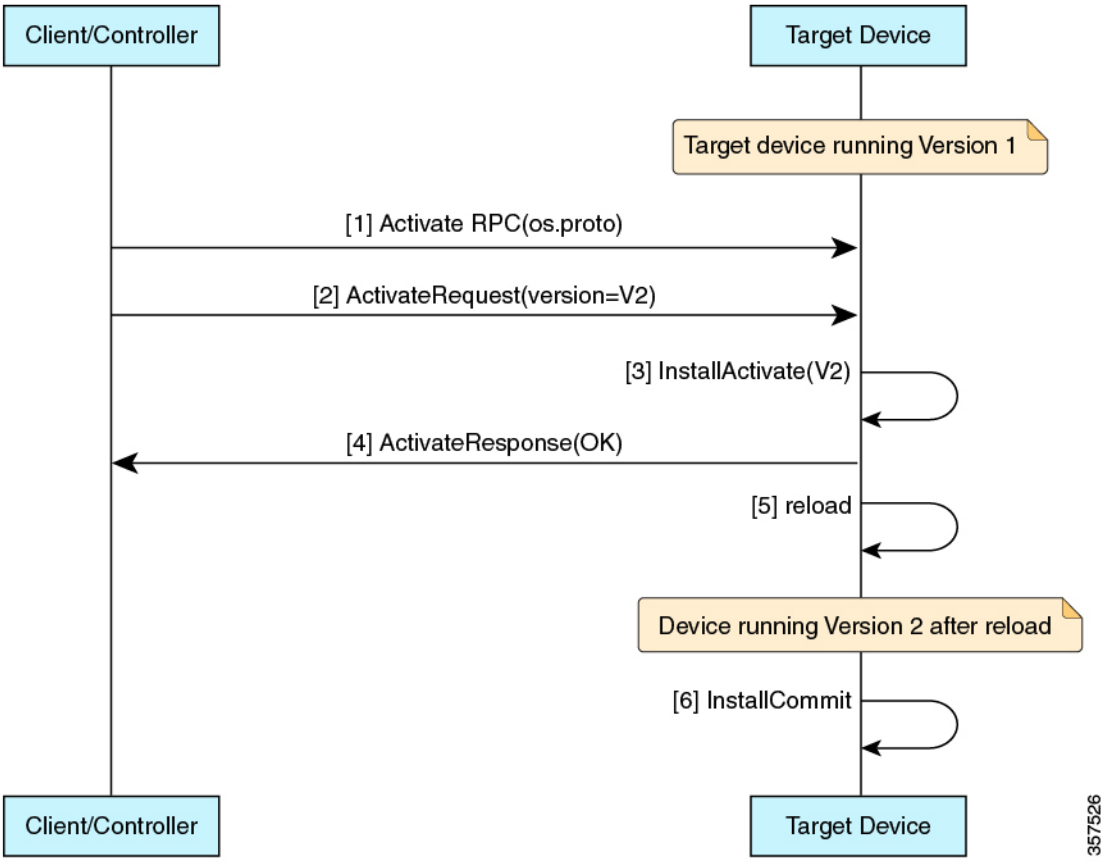


図 *3* : バンドルモードでのデュアル *RP* イメージインストール + 非 *ISSU* アクティブ化のワークフロー

図 *4 :* デュアル *RP* イメージインストール *+* 非 *ISSU* アクティブ化のンワークフロー

## OS Verify RPC

Verify RPC は、実行中の OS バージョンを検証します。RPC への応答には、スタンバイ RP のサポートとプレゼンスに関する情報が含まれています。

最後の Activate RPCでエラーが発生した場合は、そのエラーが文字列として応答で返されます。gNOIOSインストールサービスは、インストール運用モデルとプラットフォームモデルを使用してこの情報を入力します。現在、インストール運用モデルは、2 つの RP で実行される異なるバージョンをサポートしていません。

# gNOI factory-reset サービス

Cisco IOS XE Cupertino 17.7.1 は、reset.proto で指定されている gNOI factory-reset サービスをサポートしています。

gNOI factory-reset サービスは、単一の RPC である *Start* をサポートしています。この RPC は、既存の状態を消去し、工場出荷時と同じ状態で起動するようにデバイスに指示します。既存の状態には、ストレージ、設定、ログ、証明書、ライセンス、crashinfo、および Rommon 変数が含まれます。すべての Rommon 変数が削除されるわけではなく、プラットフォームごとに十分な量が保持され、保持されたイメージでイメージを自動的に再起動できます。次に、デバイスは現在のオペレーティング システム イメージで再起動し、簡素化されたブートストラップ ワークフローに基づいてデフォルトの状態に戻ります。この RPC は、ターゲットデバイスがプロビジョニング済み状態の場合にのみ受け入れられます。

ただし、開始 RPC は **factory-reset all** コマンドに似ています。 イメージを削除するコマンドとは異なり、RPC は現在のオペレーティング システム イメージを保持します。現在のイメージが存在するフラッシュまたはハードディスクは、初期状態へのリセットスクリプトの一部としてどちらもクリーンアップされます。ただし、初期状態へのリセットスクリプトを実行すると、ブートイメージまたはパッケージが */tmp* フォルダにバックアップされ、復元されます。

通常の factory-reset では、デバイスに保存されているお客様固有のデータがすべて消去され、デバイスの設定は出荷時の元の設定に復元されます消去されるデータには、設定、ログファイル、ブート変数、コアファイル、および連邦情報処理標準関連（FIPS 関連）のキーなどのクレデンシャルが含まれます。NIST SP 800-88 Rev. 1 で説明されているように、消去は clear メソッドと一致します。詳細については、ご使用のプラットフォームの『*System Management Configuration Guide*』の「Performing Factory Reset Services」を参照してください。

## gNOI factory-reset エラー メッセージ

gNOI factory-reset サービスでは、デバイスで factory-reset が正常にトリガーされると、空の ResetSuccess メッセージが返されます。

このセクションでは、gRPC および factory-reset サービスにおけるコンテキストで返されるエラーメッセージの一部について説明します。

表 *1 : gNOI factory-reset* エラー メッセージ

| エラーメッセージ | エラーの説明 |
|---|---|
| *factory_os* フィールドが要求されると、GNMIB は「Factory OS rollback is not supported.（Factory OS ロールバックはサポートされていません。）」というメッセージとともに、INVALID_ARGUMENT の gRPC エラーコードを返します。 | ResetError メッセージでは、クライアントは TRUE に設定された factory_os_unsupported フィールドも受け取ります。メッセージの他のフィールドにはデフォルト値があります。 |
| このデバイスは要求されたゼロ入力オプションをサポートしていません。 | *StartRequest* メッセージには、永続的なストレージ状態データをゼロ入力するようターゲットデバイスに指示する、オプションの *zero_fill* フィールドがあります。<br>クライアントがゼロ入力を要求し、デバイスがゼロ入力を実行できない場合、「This device does not support the requested zero-fill option.（このデバイスは要求されたゼロ入力オプションをサポートしていません。）」というメッセージとともに、INVALID_ARGUMENT の gRPC エラーコードがクライアントに返されます。<br>*ResetError* メッセージでは、zero_fill_unsupported フィールドは TRUE に設定されています。 |
| このデバイスは要求されたゼロ入力オプションをサポートしていません。 | デバイスがゼロ入力を実行できるが、クライアントがゼロ入力を要求していない場合、「This device does not support the requested zero-fill option.（このデバイスは要求されたゼロ入力オプションをサポートしていません。）」というメッセージとともに、INVALID_ARGUMENT の gRPC エラーコードがクライアントに返されます。<br>*ResetError* メッセージでは、zero_fill_unsupported フィールドは FALSE に設定されています。 |
| factory-reset 機能がありません。 | サポートされていないプラットフォームで gNOI factory-reset へのリセットスクリプトが実行されると、factory-reset サービスは「Factory reset capability is not present.（factory-reset 機能がありません。）」というメッセージとともに、UNIMPLEMENTED の gRPC エラーコードを返します。 |

| エラーメッセージ | エラーの説明 |
|---|---|
| factory-reset インターフェイスの準備ができていません。 | gNOI factory-reset 管理インターフェイスがダウンしているかビジー状態である場合、factory-reset サービスは「Factory reset interface is not ready.（factory-reset インターフェイスの準備ができていません。）」というメッセージとともに、UNAVAILABLE の gRPC エラーコードを返します。 |

cert.proto プロビジョニング操作を使用しないか、署名付き証明書（自己署名ではない）で gNOI を設定しないと、gNOI factory-reset サービスは常に FAILED_PRECONDITION エラーコードを返します。

# gRPC ネットワーク操作インターフェイスに関する追加情報

### 関連資料

| 関連項目 | マニュアル タイトル |
|---|---|
| DevNet | https://developer.cisco.com/site/ios-xe/ |
| gNOI | https://github.com/openconfig/gnoi |
| OS サービス | https://github.com/openconfig/gnoi/blob/master/os/os.proto |
| gNOI factory-reset サービス | https://github.com/openconfig/gnoi/blob/master/factory_reset/factory_reset.proto |
| デバイスのセットアップ設定の実行 | • システム管理コンフィギュレーション ガイド（Catalyst 9200 スイッチ）<br>• システム管理コンフィギュレーション ガイド（Catalyst 9300 スイッチ）<br>• システム管理コンフィギュレーション ガイド（Catalyst 9400 スイッチ）<br>• システム管理コンフィギュレーション ガイド（Catalyst 9500 スイッチ）<br>• システム管理コンフィギュレーション ガイド（Catalyst 9600 スイッチ） |

| 関連項目 | マニュアル タイトル |
|---|---|
| 初期設定へのリセットの実行 | • システム管理コンフィギュレーション ガイド（Catalyst 9300 スイッチ）<br>• システム管理コンフィギュレーション ガイド（Catalyst 9300 スイッチ）<br>• システム管理コンフィギュレーション ガイド（Catalyst 9300 スイッチ）<br>• システム管理コンフィギュレーション ガイド（Catalyst 9300 スイッチ）<br>• システム管理コンフィギュレーション ガイド（Catalyst 9300 スイッチ） |

### シスコのテクニカル サポート

| 説明 | リンク |
|---|---|
| シスコのサポートWebサイトでは、シスコの製品やテクノロジーに関するトラブルシューティングにお役立ていただけるように、マニュアルやツールをはじめとする豊富なオンラインリソースを提供しています。<br>お使いの製品のセキュリティ情報や技術情報を入手するために、Cisco Notification Service（Field Notice からアクセス）、Cisco Technical Services Newsletter、Really Simple Syndication（RSS）フィードなどの各種サービスに加入できます。<br>シスコのサポート Web サイトのツールにアクセスする際は、Cisco.com のユーザ ID およびパスワードが必要です。 | http://www.cisco.com/support |

# gRPC ネットワーク操作インターフェイスの機能情報

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコ ソフトウェアイメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *2 : gRPC* ネットワーク操作インターフェイスの機能情報

| 機能名 | リリース | 機能情報 |
| --- | --- | --- |
| gNOI 証明書の管理 | Cisco IOS XE Amsterdam 17.3.1 | gNOI 証明書の管理サービスは、RPC を提供して、インストール、ローテーション、証明書の取得、証明書の失効、および証明書署名要求の生成を行います。<br>この機能は、Cisco IOS XE Amsterdam 17.3.1 で次のプラットフォームに実装されました。<br>• Cisco Catalyst 9200 シリーズ スイッチ<br>• Cisco Catalyst 9300 シリーズ スイッチ<br>• Cisco Catalyst 9400 シリーズ スイッチ<br>• Cisco Catalyst 9500 シリーズ スイッチ<br>• Cisco Catalyst 9600 シリーズ スイッチ |

| 機能名 | リリース | 機能情報 |
| --- | --- | --- |
| 証明書サービスによる gNOI ブートストラップ | Cisco IOS XE Amsterdam 17.3.1 | gNOI 証明書をインストールした後、ブートストラップを使用してターゲットデバイスを設定または操作します。gNMI ブートストラップは、**gnxi-secure-init** コマンドで有効、**secure-allow-self-signed-trustpoint** コマンドで無効になります。<br>この機能は、Cisco IOS XE Amsterdam 17.3.1 で次のプラットフォームに実装されました。<br>• Cisco Catalyst 9200 シリーズ スイッチ<br>• Cisco Catalyst 9300 シリーズ スイッチ<br>• Cisco Catalyst 9400 シリーズ スイッチ<br>• Cisco Catalyst 9500 シリーズ スイッチ<br>• Cisco Catalyst 9600 シリーズ スイッチ |
| gNOI OS インストール サービス | Cisco IOS XE Bengaluru 17.5.1 | gNOI OSインストールサービスは、インストールに使用される gNOI API を定義します。<br>この機能は、Cisco IOS XE Bengaluru 17.5.1で次のプラットフォームに実装されました。<br>• Cisco Catalyst 9300 シリーズ スイッチ<br>• Cisco Catalyst 9400 シリーズ スイッチ<br>• Cisco Catalyst 9500 および 9500 ハイパフォーマンス シリーズ スイッチ<br>• Cisco Catalyst 9600 シリーズ スイッチ |

| 機能名 | リリース | 機能情報 |
| --- | --- | --- |
| gNOI factory-reset サービス | Cisco IOS XE Cupertino 17.7.1 | gNOI factory-reset サービスは、現在の状態を消去し、工場出荷時と同じ状態でデバイスを起動するようにターゲットデバイスに指示するインターフェイスを提供します。<br>Cisco IOS XE Cupertino 17.7.1 では、この機能は次のプラットフォームに実装されました。<br>• Cisco Catalyst 9300 シリーズ スイッチ<br>• Cisco Catalyst 9400 シリーズ スイッチ<br>• Cisco Catalyst 9500 および 9500 ハイパフォーマンス シリーズ スイッチ<br>• Cisco Catalyst 9800-40 ワイヤレスコントローラ<br>• Cisco Catalyst 9800-80 ワイヤレスコントローラ |

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。