gRPC ネットワーク操作インターフェイス

gRPC Network Operations Interface（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

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、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認証を有効にするには、https://github.com/openconfig/public/blob/master/release/models/system/openconfig-system-management.yang#L87 を参照してください。

Cisco IOS XE Amsterdam 17.3.1 では、次の操作がサポートされています。

- 証明書の管理
- ブートストラップ

証明書管理サービス

証明書管理サービスでは、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る 2 つの主要な RPC が優先的にエクスポートされます。

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（文字列）によって一意に識別されます。
- 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 Public Key Infrastructure（PKI）設定でトラストポイント名として使用され、既存の証明書 ID を持つ証明書の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、コントローラはその証明書のコピーを保持する必要がありません。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 が必要とします。
4. gNMI/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 のシーケンスには、次のように RPC インストールとの主な違いがあります。

- 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。

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;
  }
}
```

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、次のフィールドが含まれます。

- 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） CA 証明書バンドルが含まれていない応答。

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、Rivet、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をロードする場合は、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。GNMIB 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、GNMIB は gNMI クライアントをすべての既存のトラスポイントと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/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。

（注） 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。

- デフォルト/暗号化：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。
- プロビジョニング済み：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、クライアントは証明書ストアに対してデバイスが検証した証明書を提示します。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。

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

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。 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。 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 *1 : 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。 • Cisco Catalyst 9200 シリーズ スイッチ • Cisco Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9400 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ • Cisco Catalyst 9600 シリーズ スイッチ
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-int** コマンドで有効になります。 • Cisco Catalyst 9200 シリーズ スイッチ • Cisco Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9400 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ • Cisco Catalyst 9600 シリーズ スイッチ