gRPC エージェント

gRPC エージェントについて

Cisco NX-OS gRPC プロトコルは、ネットワーク デバイスを管理し、その構成データを取得してインストールするためのメカニズムを定義します。プロトコルは、クライアントがデバイス構成を管理するために使用できる完全で正式なアプリケーションプログラミングインターフェイス（API）を公開します。

Cisco NX-OS gRPC プロトコルは、リモート プロシージャ コール（RPC）パラダイムを使用します。このパラダイムでは、外部クライアントが、Google プロトコル バッファ（GPB）で定義された API コールとそのサービス固有の引数を使用してデバイス設定を操作します。これらの GPB 定義 API は、同じ GPB 定義 API コンテキストで応答を返すデバイスへの RPC コールを透過的に引き起こします。

gRPC エージェントは、TLS によるセキュアな転送と、AAA によるユーザー認証と認可を提供します。

Cisco NX-OS gRPC プロトコルの機能上の目的は、特にステートレスとステートフルの両方の構成操作に関して、NETCONF によって提供されるプロトコルをミラーリングして、運用の柔軟性を最大限に高めることです。

Cisco NX-OS gRPC エージェントは、次のプロトコル操作をサポートしています。

- 結果
- GetConfig
- GetOper
- edit-config

- start-session
- CloseSession
- kill-session

gRPC エージェントは、次の 2 種類の操作をサポートしています：

- **ステートレス操作**は、セッションを作成せずに単一のメッセージ内で完全に実行されます。
- **ステートフル操作**は、複数のメッセージを使用して実行されます。実行される動作のシーケンスは、次のとおりです：
  1. セッションを開始します。このアクションは、一意のセッション ID を取得します。
  2. セッション ID を使用してセッションタスクを実行します。
  3. セッションを終了します。このアクションにより、セッション ID が無効になります。

次にあるのは、サポートされている操作です。gRPC エージェントによってエクスポートされる **.proto** ファイルの RPC 定義については、付録を参照してください。

動作	説明
start-session	クライアントとサーバー間の新しいセッションを開始し、一意のセッション ID を取得します。
edit-config	指定された YANG データ サブセットをターゲット データストアに書き込みます。
GetConfig	送信元データストアから指定された YANG 構成データのサブセットを取得します。
GetOper	送信元データストアから指定された YANG 動作データを取得します。
結果	指定された YANG 構成および動作データを送信元データストアから取得します。
KillSession	セッションを強制終了します。
CloseSession	セッションの適切な終了を要求します。

GetConfig、GetOper、および Get はステートレス操作であるため、セッション ID は必要ありません。

EditConfig は、ステートレスまたはステートフルのいずれかです。ステートレス操作の場合は、SessionID を 0 に指定します。ステートフル操作の場合、有効な（ゼロ以外の）SessionID が必要です。

gRPC エージェントは、セッションのタイムアウトをサポートします。セッションのアイドル タイムアウトはデバイスで構成できます。アイドル セッションは終了して削除されます。

注意事項と制約事項

gRPC エージェントには、次のガイドラインと制限があります。

- gRPC は、RFC 6536 で指定されている拡張ロールベース アクセス コントロール（RBAC）をサポートしていません。「network-admin」ロールを持つユーザーのみに、gRPC エージェントへのアクセス権が付与されます。

Cisco NX-OS リリース 9.3（2）以前の gRPC エージェントの構成

gRPC エージェントは、構成ファイル（ **/etc/mtx.conf** ）の **[grpc]** セクションで次の構成パラメータをサポートします。

パラメータ	説明
idle_timeout	（オプション）アイドル状態のクライアント セッションが切断されるまでのタイムアウトを分単位で指定します。 デフォルトのタイムアウトは 5 分です。 値を 0 に設定するとタイムアウトが無効になります。
limit	（オプション）同時クライアント セッションの最大数を指定します。 デフォルト制限値は 5 セッションです。 範囲は 1 ～ 50 です。
lport	（任意） gRPC エージェントがリッスンするポート番号を指定します。 デフォルトのポートは 50051 です。
key	TLS 認証のキーファイルの場所を指定します。 デフォルトの場所は **/opt/mtx/etc/grpc.key** です。
cert	TLS 認証の認定ファイルの場所を指定します。 デフォルトの場所は **/opt/mtx/etc/grpc.pem** です。

パラメータ	説明
security	セキュアな接続のタイプを指定します。 有効な選択肢は、次のとおりです。 • **TLS** の TLS • セキュアでない接続の場合は **NONE**

gRPC エージェントの使用

一般的なコマンド

gRPC エージェントを有効または無効にするには、 [**no**] **feature grpc** コマンドを発行します。

例：JSON フォーマットの基本的な YANG パス

```
client-host % cat payload.json
{
  "namespace": "http://cisco.com/ns/yang/cisco-nx-os-device",
  "System": {
    "bgp-items": {
      "inst-items": {
        "dom-items": {
          "Dom-list": {
            "name": "default",
            "rtrId": "7.7.7.7",
            "holdIntvl": "100"
          }
        }
      }
    }
  }
}
```

（注） JSON 構造は、読みやすいように整形されています。

サーバーへの EditConfig 要求の送信

```
client-host % ./grpc_client -username=admin -password=cisco -operation=EditConfig
-e_oper=Merge -def_op=Merge -err_op=stop-on-error -infile=payload.json -reqid=1
-source=running -tls=true -serverAdd=192.0.20.123 -lport=50051

#####################################################
Starting the client service
#####################################################
TLS set true for client requests1ems.cisco.com
TLS FLAG:1
```

```
192.0.20.123:50051
All the client connections are secured
Sending EditConfig request to the server
sessionid is
0
reqid:1
{"rpc-reply":{"ok":""}}
```

サーバーへの GetConfig 要求の送信

```
client-host % ./grpc_client -username=admin -password=cisco -operation=GetConfig
-infile=payload.json -reqid=1 -source=running -tls=true -serverAdd=192.0.20.123
-lport=50051

#######################################################
Starting the client service
#######################################################
TLS set true for client requests1ems.cisco.com
TLS FLAG:1
192.0.20.123:50051
All the client connections are secured
Sending GetConfig request to the server
in get config
Got the response from the server
########################################
Yang Json is:
########################################
{"rpc-reply":{"data":{"System":{"bgp-items":{"inst-items":{"dom-items":{"Dom-list":{"name":"default","rtrId":"7.7.7.7","holdIntvl":"100"}}}}}}}}
########################################
client-host %
```

gRPC エージェントのトラブルシューティング

接続のトラブルシューティング

- クライアント システムから、エージェントがポートでリッスンしていることを確認します。次に例を示します。

  ```
  client-host % nc -z 192.0.20.222 50051
  Connection to 192.0.20.222 50051 port [tcp/*] succeeded!
  client-host % echo $?
  0
  client-host %
  ```

- NX-OS で、 を発行して gRPC エージェントのステータスを確認します **show feature | grep grpc**。

gRPC Protobuf ファイル

gRPC エージェントは、/opt/mtx/etc/nxos_grpc.proto にある proto 定義ファイルでサポートされている操作とデータ構造をエクスポートします。ファイルは、gRPC エージェント RPM に含まれています。次に、定義を示します。

```
// Copyright 2016, Cisco Systems Inc.
// All rights reserved.

syntax = "proto3";

package NXOSExtensibleManagabilityService;

// Service provided by Cisco NX-OS gRPC Agent
service gRPCConfigOper {

    // Retrieves the specified YANG configuration data subset from the
    // source datastore
    rpc GetConfig(GetConfigArgs) returns(stream GetConfigReply) {};

    // Retrieves the specified YANG operational data from the source datastore
    rpc GetOper(GetOperArgs) returns(stream GetOperReply) {};

    // Retrieves the specified YANG configuration and operational data
    // subset from the source datastore
    rpc Get(GetArgs) returns(stream GetReply){};

    // Writes the specified YANG data subset to the target datastore
    rpc EditConfig(EditConfigArgs) returns(EditConfigReply) {};

    // Starts a new session between the client and server and acquires a
    // unique session ID
    rpc StartSession(SessionArgs) returns(SessionReply) {};

    // Requests graceful termination of a session
    rpc CloseSession(CloseSessionArgs) returns (CloseSessionReply) {};

    // Forces the termination of a session
    rpc KillSession(KillArgs) returns(KillReply) {};

// Unsupported; reserved for future
    rpc DeleteConfig(DeleteConfigArgs) returns(DeleteConfigReply) {};

    // Unsupported; reserved for future
    rpc CopyConfig(CopyConfigArgs) returns(CopyConfigReply) {};

    // Unsupported; reserved for future
    rpc Lock(LockArgs) returns(LockReply) {};

    // Unsupported; reserved for future
    rpc UnLock(UnLockArgs) returns(UnLockReply) {};

    // Unsupported; reserved for future
    rpc Commit(CommitArgs) returns(CommitReply) {};

    // Unsupported; reserved for future
    rpc Validate(ValidateArgs) returns(ValidateReply) {};

    // Unsupported; reserved for future
    rpc Abort(AbortArgs) returns(AbortReply) {};
```

```
}

message GetConfigArgs
{
    // JSON-encoded YANG data to be retrieved
    string YangPath = 1;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 2;

    // (Optional) Specifies the source datastore; only "running" is supported.
    // Default is "running".
    string Source = 3;
}

message GetConfigReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // JSON-encoded YANG data that was retrieved
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message GetOperArgs
{
    // JSON-encoded YANG data to be retrieved
    string YangPath = 1;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 2;
}

message GetOperReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // JSON-encoded YANG data that was retrieved
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message GetArgs
{
    // JSON-encoded YANG data to be retrieved
    string YangPath=1;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 2;
}

message GetReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // JSON-encoded YANG data that was retrieved
```

```
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message EditConfigArgs
{
    // JSON-encoded YANG data to be edited
    string YangPath = 1;

    // Specifies the operation to perform on teh configuration datastore with
    // the YangPath data.  Possible values are:
    //   create
    //   merge
    //   replace
    //   delete
    //   remove
    // If not specified, default value is "merge".
    string Operation = 2;

    // A unique session ID acquired from a call to StartSession().
    // For stateless operation, this value should be set to 0.
    int64 SessionID = 3;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 4;

    // (Optional) Specifies the target datastore; only "running" is supported.
    // Default is "running".
    string Target = 5;

    // Specifies the default operation on the given object while traversing
    // the configuration tree.
    // The following operations are possible:
    //   merge:     merges the configuration data with the target datastore;
    //              this is the default.
    //   replace:   replaces the configuration data with the target datastore.
    //   none:      target datastore is unaffected during the traversal until
    //              the specified object is reached.
    string DefOp = 6;

    // Specifies the action to be performed in the event of an error during
    // configuration.  Possible values are:
    //   stop
    //   roll-back
    //   continue
    // Default is "roll-back".
    string ErrorOp = 7;
}

message EditConfigReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // If EditConfig is successful, YangData contains a JSON-encoded "ok" response.
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message DeleteConfigArgs
```

```
{
    // A unique session ID acquired from a call to StartSession().
    // For stateless operation, this value should be set to 0.
    int64 SessionID = 1;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 2;

    // (Optional) Specifies the target datastore; only "running" is supported.
    // Default is "running".
    string Target = 3;
}

message DeleteConfigReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // If DeleteConfig is successful, YangData contains a JSON-encoded "ok" response.
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message CopyConfigArgs
{
    // A unique session ID acquired from a call to StartSession().
    // For stateless operation, this value should be set to 0.
    int64 SessionID = 1;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 2;

    // (Optional) Specifies the source datastore; only "running" is supported.
    // Default is "running".
    string Source = 3;

    // (Optional) Specifies the target datastore; only "running" is supported.
    // Default is "running".
    string Target = 4;
}

message CopyConfigReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // If CopyConfig is successful, YangData contains a JSON-encoded "ok" response.
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message LockArgs
{
    // A unique session ID acquired from a call to StartSession().
    int64 SessionID = 1;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID=2;

    // (Optional) Specifies the target datastore; only "running" is supported.
```

```
    // Default is "running".
    string Target = 3;
}

message LockReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // If Lock is successful, YangData contains a JSON-encoded "ok" response.
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message UnLockArgs
{
    // A unique session ID acquired from a call to StartSession().
    int64 SessionID = 1;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 2;

    // (Optional) Specifies the target datastore; only "running" is supported.
    // Default is "running".
    string Target = 3;
}

message UnLockReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // If UnLock is successful, YangData contains a JSON-encoded "ok" response.
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message SessionArgs
{
    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 1;
}

message SessionReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;
    int64 SessionID = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message CloseSessionArgs
{
    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 1;

    // A unique session ID acquired from a call to StartSession().
    int64 SessionID = 2;
```

```
}

message CloseSessionReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // If CloseSession is successful, YangData contains a JSON-encoded "ok" response.
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message KillArgs
{
    // A unique session ID acquired from a call to StartSession().
    int64 SessionID = 1;

    int64 SessionIDToKill = 2;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 3;
}

message KillReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // If Kill is successful, YangData contains a JSON-encoded "ok" response.
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message ValidateArgs
{
    // A unique session ID acquired from a call to StartSession().
    int64 SessionID = 1;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 2;
}

message ValidateReply
{
    // The request ID specified in the request.
    int64 ReqID = 1;

    // If Validate is successful, YangData contains a JSON-encoded "ok" response.
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message CommitArgs
{
    // A unique session ID acquired from a call to StartSession().
    int64 SessionID = 1;

    // (Optional) Specifies the request ID.  Default value is 0.
```

```
    int64 ReqID = 2;
}

message CommitReply
{
    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 1;

    // If Commit is successful, YangData contains a JSON-encoded "ok" response.
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}

message AbortArgs
{
    // A unique session ID acquired from a call to StartSession().
    int64 SessionID = 1;

    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 2;
}

message AbortReply
{
    // (Optional) Specifies the request ID.  Default value is 0.
    int64 ReqID = 1;

    // If Abort is successful, YangData contains a JSON-encoded "ok" response.
    string YangData = 2;

    // JSON-encoded error information when request fails
    string Errors = 3;
}
```

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。