モデル駆動型テレメトリ

- テレメトリについて （1 ページ）
- テレメトリのライセンス要件 （3 ページ）
- テレメトリの注意事項と制約事項 （4 ページ）
- CLI を使用したテレメトリの構成 （8 ページ）
- NX-API を使用したテレメトリの構成 （27 ページ）
- その他の参考資料 （41 ページ）

テレメトリについて

分析やトラブルシューティングのためのデータ収集は、ネットワークの健全性をモニタリングする上で常に重要な要素であり続けています。

Cisco NX-OS は、ネットワークからデータを収集するための、SNMP、CLI や Syslog といった複数のメカニズムを提供します。これらのメカニズムには、自動化や拡張に対する制約があります。ネットワーク要素からのデータの最初の要求がクライアントから出された場合、プルモデルの使用が制限されることもその制約の 1 つです。プル モデルは、ネットワーク内に複数のネットワーク管理ステーション（NMS）がある場合は拡張しません。このモデルを使用すると、クライアントが要求した場合に限り、サーバーがデータを送信します。このような要求を開始するには、手動による介入を続けて行う必要があります。このような手動による介入を続けると、プル モデルの効率が失われます。

プッシュ モデルは、ネットワークからデータを継続的にストリーミングし、クライアントに通知します。テレメトリはプッシュ モデルをイネーブルにし、モニタリング データにほぼリアルタイムでアクセスできるようにします。

テレメトリ コンポーネントとプロセス

テレメトリは、次の 4 つの主要な要素で構成されます。

- **データ収集**：テレメトリ データは、識別名（DN）パスを使用して指定されたオブジェクト モデルのブランチにあるデータ管理エンジン（DME）データベースから収集されます。データは定期的に取得されるか（頻度ベース）、指定したパスのオブジェクトで変更が

あった場合にのみ取得できます（イベント ベース）。NX-API を使用して、頻度ベースのデータを収集できます。

- **データ エンコーディング**：テレメトリ エンコーダが、収集されたデータを目的の形式で転送できるようにカプセル化します。

  NX-OS は、テレメトリ データを Google Protocol Buffers（GPB）および JSON 形式でエンコードします。

- **データ トランスポート**：NX-OS は、JSON エンコードに HTTP を使用してテレメトリ データを転送し、GPB エンコードに Google リモート プロシージャ コール (gRPC) プロトコルを使用します。gRPC レシーバーは、4 MB を超えるメッセージ サイズをサポートします。（証明書が構成されている場合は、HTTPS を使用したテレメトリ データもサポートされます。）

  Cisco NX-OS リリース 7.0(3)I7(1) 以降、UDP およびセキュア UDP（DTLS）がテレメトリ トランスポート プロトコルとしてサポートされています。UDP を受信する接続先を追加できます。UDP およびセキュア UDP のエンコーディングは、GPB または JSON にすることができます。

  次のコマンドを使用して、JSON または GPB のデータグラム ソケットを使用してデータをストリーミングするように UDP トランスポートを構成します。

```
destination-group num
  ip address xxx.xxx.xxx.xxx port xxxx protocol UDP encoding {JSON | GPB }
```

  IPv4 接続先の例:

```
destination-group 100
  ip address 171.70.55.69 port 50001 protocol UDP encoding GPB
```

  UDP テレメトリには次のヘッダーがあります。

```
typedef enum tm_encode_ {
  TM_ENCODE_DUMMY,
  TM_ENCODE_GPB,
  TM_ENCODE_JSON,
  TM_ENCODE_XML,
  TM_ENCODE_MAX,
} tm_encode_type_t;

typedef struct tm_pak_hdr_ {
  uint8_t version; /* 1 */
  uint8_t encoding;
  uint16_t msg_size;
  uint8_t secure;
  uint8_t padding;
}__attribute__ ((packed, aligned (1))) tm_pak_hdr_t;
```

  次のいずれかの方法で、ペイロードの最初の 6 バイトを使用して、UDP を使用してテレメトリ データを処理します。

  - 受信側が複数のエンドポイントから異なるタイプのデータを受信することになっている場合は、ヘッダーの情報を読んで、データのデコードに使用するデコーダー（JSON または GPB）を決定します。

- 1 つのデコーダー（JSON または GPB）が必要で、もう 1 つのデコーダーは必要ない場合は、ヘッダーを削除します。

（注） UDPプロトコルを使用した場合、受信側のOSやネットワークの負荷によってはパケットドロップが発生する場合があります。

- **テレメトリ レシーバー**：テレメトリ レシーバーは、テレメトリ データを保存するリモート管理システムです。

GPB エンコーダーは、汎用キーと値の形式でデータを格納します。また、データを GPB 形式に変換するには、コンパイルされた `.proto` ファイル形式のメタデータが GPB エンコーダに必要です。

データ ストリームを正しく受信してデコードするには、受信側でエンコードとトランスポート サービスを記述した `.proto` ファイルが必要です。エンコードは、バイナリ ストリームをキー値の文字列のペアにデコードします。

GPB エンコーディングと gRPC トランスポートを記述する telemetry `.proto` ファイルは、Cisco の GitLab で入手できます。 https://github.com/CiscoDevNet/nx-telemetry-proto

テレメトリ プロセスの高可用性

テレメトリ プロセスの高可用性は、次の動作でサポートされています。

- **[システムのリロード（System Reload）]** — システムのリロード中に、テレメトリ構成とストリーミング サービスが復元されます。
- **[スーパーバイザ フェールオーバー（Supervisor Failover）]** — テレメトリはホット スタンバイではありませんが、テレメトリ構成とストリーミング サービスは、新しい現用系スーパーバイザが実行されているときに復元されます。
- **[プロセスの再起動（Process Restart）]** — なんらかの理由でテレメトリ プロセスがフリーズまたは再起動した場合、テレメトリが再開されると、構成およびストリーミング サービスが復元されます。

テレメトリのライセンス要件

製品	ライセンス要件
Cisco NX-OS	テレメトリにはライセンスは必要ありません。ライセンス パッケージに含まれていない機能は Cisco NX-OS イメージにバンドルされており、無料で提供されます。NX-OS ライセンス方式の詳細については、『*Cisco NX-OS Licensing Guide*』を参照してください。

テレメトリの注意事項と制約事項

テレメトリ 構成時の注意事項および制約事項は、次のとおりです。

- テレメトリは、7.0（3）I5（1）を始めとするデータ管理エンジン（DME）ネイティブ モデルをサポートするリリースをCisco NX-OS リリースでサポートします。
- リリース 7.0（3）I6（1）が DME データ収集、NX-API データ 送信元、Google リモート プロシージャ コール（gRPC）トランスポート経由のGoogleプロトコルバッファ（GPB）エンコーディングと HTTP 経由の JSON エンコーディングをサポートします。
- サポートされている最小の送信間隔（ケイデンス）は、深さが 0 の場合の 5 秒です。0 より大きい深度値の最小ケイデンス値は、ストリーミングされるデータのサイズによって異なります。最小値未満のケイデンスでもを構成すると、望ましくないシステム動作が発生する可能性があります。
- 最大 5 つの遠隔管理受信者（接続先）がサポートされます。5 つ以上の遠隔受信者を構成すると、システムが望ましくない動作をする可能性があります。
- テレメトリ 受信者がダウンした場合、その他の受信者にはデータ フローが中断したことが表示されます。障害が発生したレシーバーは再起動する必要があります。次に、接続先グループの下にある障害受信者のIPアドレスを構成解除してから再構成することにより、スイッチとの新しい接続を開始します。
- テレメトリは、CPU 技術情報の最大 20% を消費する可能性があります。
- SSL 証明書ベースの認証とストリーミング データの暗号化を構成するには、**certificate** *ssl cert path* **hostname "CN"** コマンドで自己署名 SSL 証明書を提供します。（NX-OS 7.0（3）I7（1）以降）。
- QoS 明示的輻輳通知（ECN）統計は、Cisco Nexus 9364C、9336C-FX、および 93240YC-FX スイッチでのみサポートされます。

古いリリースにダウングレードした後の構成コマンド

古いリリースにダウングレードした後、古いリリースではサポートされていない可能性があるため、一部の構成コマンドまたはコマンドオプションが機能不全になる可能性があります。古いリリースにダウングレードするときのベストプラクティスとして、サポートされていないコマンドまたはコマンドオプションの失敗を回避するために、新しいイメージが起動した後にテレメトリ機能を構成解除して再構成します。

次の例は、この手順を表示しています。

- テレメトリ構成をファイルにコピーします。

```
switch# show running-config | section telemetry
feature telemetry
telemetry
  destination-group 100
    ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB
```

```
  sensor-group 100
    path sys/bgp/inst/dom-default depth 0
  subscription 600
    dst-grp 100
    snsr-grp 100 sample-interval 7000
switch# show running-config | section telemetry > telemetry_running_config
switch# show file bootflash:telemetry_running_config
feature telemetry
telemetry
  destination-group 100
    ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB
  sensor-group 100
    path sys/bgp/inst/dom-default depth 0
  subscription 600
    dst-grp 100
    snsr-grp 100 sample-interval 7000
switch#
```

- ダウングレード操作を実行します。イメージが表示され、スイッチの準備ができたら、テレメトリ構成をスイッチにコピーして戻します：

```
switch# copy telemetry_running_config running-config echo-commands
`switch# config terminal`
`switch(config)# feature telemetry`
`switch(config)# telemetry`
`switch(config-telemetry)# destination-group 100`
`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB `
`switch(conf-tm-dest)# sensor-group 100`
`switch(conf-tm-sensor)# path sys/bgp/inst/dom-default depth 0`
`switch(conf-tm-sensor)# subscription 600`
`switch(conf-tm-sub)# dst-grp 100`
`switch(conf-tm-sub)# snsr-grp 100 sample-interval 7000`
`switch(conf-tm-sub)# end`
Copy complete, now saving to disk (please wait)...
Copy complete.
switch#
```

gRPC エラーの動作

gRPC レシーバーが 20 のエラーを送信した場合、スイッチ クライアントは gRPC レシーバーへの接続を無効化します。gRPC レシーバーを有効にするには、接続先グループの下のレシーバーの IP アドレスの構成を解除して再構成する必要があります。一部のエラーの内容は、次のとおりです。

- gRPC クライアントがセキュアな接続に対して誤った証明書を送信している。
- gRPC レシーバでのクライアント メッセージの処理に時間がかかりすぎて、タイムアウトが発生する。別のメッセージ処理スレッドを使用してメッセージを処理することで、タイムアウトを回避している。

gRPC トランスポートのテレメトリ圧縮

Cisco NX-OS 7.0（3）I7（1）以降、gRPC トランスポートでは、圧縮のサポートが利用できます。**use-compression gzip** コマンドを使用して、圧縮を有効にすることができます。（**no use-compression gzip**コマンドで圧縮を無効にします。）

次の例では、圧縮を有効にします。

```
switch(config)# telemetry
switch(config-telemetry)# destination-profile
switch(config-tm-dest-profile)# use-compression gzip
```

次の例は、圧縮が有効になっていることを表示しています。

```
switch(conf-tm-dest)# show telemetry transport 0 stats

Session Id:                    0
Connection Stats
   Connection Count            0
   Last Connected:             Never
   Disconnect Count            0
   Last Disconnected:          Never
Transmission Stats
   Compression:                gzip

   Transmit Count:             0
   Last TX time:               None
   Min Tx Time:                0                   ms
   Max Tx Time:                0                   ms
   Avg Tx Time:                0                   ms
   Cur Tx Time:                0                   ms
```

```
switch2(config-if)# show telemetry transport 0 stats

Session Id: 0
Connection Stats
Connection Count 0
Last Connected: Never
Disconnect Count 0
Last Disconnected: Never
Transmission Stats
Compression: disabled
Source Interface: loopback1(1.1.3.4)
Transmit Count: 0
Last TX time: None
Min Tx Time: 0 ms
Max Tx Time: 0 ms
Avg Tx Time: 0 ms
Cur Tx Time: 0 ms
switch2(config-if)#
```

以下は、POST ペイロードとしての use-compression の例です。

```
{
              "telemetryDestProfile": {
                "attributes": {
                  "adminSt": "enabled"
                },
                "children": [
                  {
                    "telemetryDestOptCompression": {
                      "attributes": {
                        "name": "gzip"
                      }
                    }
                  }
                ]
```

```
    }
  }
```

NX-API センサー パスの制限

NX-API は、**show** コマンドを使用して、DME にまだ存在しないスイッチ情報を収集してストリーミングできます。ただし、DME からデータをストリーミングする代わりに NX-API を使用すると、次に示すように、固有の拡張制限があります。

- スイッチ バックエンドは、**show** コマンドなどの NX-API 呼び出しを動的に処理します。
- NX-API は、CPU の最大 20% を消費する可能性のあるいくつかのプロセスを生成します。
- NX-API データは、CLI から XML、JSON に変換されます。

以下は、過度の NX-API センサー パス帯域幅消費を制限するのに役立つ推奨ユーザー フローです。

1. **show** コマンドが NX-API をサポートしているかどうかを確認します。パイプ オプションを使用して、NX-API が VSH からのコマンドをサポートしているかどうかを確認できます：`<command> | json` または`<command> | json pretty`。

（注） スイッチが JSON 出力を返すまでに 30 秒以上かかるコマンドは避けてください。

2. フィルタまたはオプションを含めるように **show** コマンドを調整します。
   - 個々の出力に対して同じコマンドを列挙することは避けてください。つまり、show vlan 識別子 100、show vlan 識別子 101 などです。代わりに、CLI 範囲オプションを使用してください。つまり、パフォーマンスを向上させるために可能な限り、VLAN 識別子 100-110、204 を表示します。

     サマリー / カウンターのみが必要な場合は、show コマンド出力全体をダンプして、データ収集に必要な帯域幅とデータ ストレージを制限しないようにします。

3. NX-API をデータ送信元として使用するセンサー グループでテレメトリを構成します。**show** コマンドをセンサー パスとして追加する
4. CPI の使用を制限するために、それぞれの **show** コマンドの処理時間の 5 倍の周期でテレメトリを構成します。
5. ストリーミングされた NX-API 出力を既存の DME コレクションの一部として受信して処理します。

テレメトリの VRF サポート

Cisco NX-OS 7.0（3）I7（1）以降、テレメトリ VRF サポートにより、トランスポート VRF を指定できます。これは、テレメトリ データ ストリームがフロント パネルのポートを介して送信され、SSH / NGINX 制御セッション間の競合の可能性を回避できることを意味します。

use-vrf *vrf-name* コマンドを使用して、トランスポート VRF を指定できます。

次の例では、トランスポート VRF を指定しています。

```
switch(config)# telemetry
switch(config-telemetry)# destination-profile
switch(config-tm-dest-profile)# use-vrf test_vrf
```

以下は、POST ペイロードとしての use-vrf の例です。

```
{
              "telemetryDestProfile": {
                "attributes": {
                  "adminSt": "enabled"
                },
                "children": [
                  {
                    "telemetryDestOptVrf": {
                      "attributes": {
                        "name": "default"
                      }
                    }
                  }
                ]
              }
            }
```

CLI を使用したテレメトリの構成

NX-OS CLI を使用したテレメトリの構成

次の手順では、ストリーミング テレメトリを有効にし、データ ストリームの送信元と接続先を構成します。これらの手順には、SSL/TLS 証明書と GPB エンコーディングを有効にして構成するオプションの手順も含まれています。

始める前に

スイッチは、Cisco NX-OS リリース 7.3(0)I5(1) 以降のリリースを実行している必要があります。

手順

	コマンドまたはアクション	目的
ステップ 1	（任意） **openssl** *argument* 例： 次のような特定の引数を使用して、SSL/TLS 証明書を生成します。	データを受信するサーバー上に SSL または TLS 証明書を作成します。ここで、*private*.key ファイルは秘密キーであり、*public*.crt は公開キーです。

	コマンドまたはアクション	目的

	• RSA 秘密キーを生成するには：**openssl genrsa** *-cipher* **-out** *filename.key cipher-bit-length*
	例：
	`switch# openssl genrsa -des3 -out server.key 2048`
	• RSA キーを作成するには：**openssl rsa -in** *filename.key* **-out** *filename.key*
	例：
	`switch# openssl rsa -in server.key -out server.key`
	• 公開キーまたは秘密キーを含む証明書を作成するには、次の手順を実行します：**openssl req**
	-encoding-standard filename.key filename.csr **-new -new -out -subj '/CN=localhost'**
	例：
	`switch# openssl req -sha256 -new -key server.key -out server.csr`
	`-subj '/CN=localhost'`
	• 公開キーを作成するには：**openssl x509 -req** *-encoding-standard* **-days** *timeframe* **-in** *filename.csr* **-signkey** *filename.key* **-out** *filename.csr*
	例：
	`switch# openssl x509 -req -sha256 -days 365 -in server.csr -signkey server.key -out server.crt`	

ステップ 2	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		グローバル構成モードを開始します。

ステップ 3	**feature telemetry**	ストリーミングテレメトリ機能を有効にします。
ステップ 4	**feature nxapi**	NX-API を有効にします。

	コマンドまたはアクション	目的

ステップ 5	**nxapi use-vrf management**
	例：
	`switch(config)#`
	`switch(config)# nxapi use-vrf management`
	`switch(config)#`
		NX-API 通信に使用する VRF 管理を有効にします。
		（注） ACL はネットスタックパケットのみをフィルタ処理できるため、10.2（3）F より前のリリースでは次の意味の警告が表示されます：「警告：構成された管理 ACL は、HTTP サービスでは有効になりません。iptables を使用してアクセスを制限してください」。
		（注） 10.2(3)F 以降、ACL は、管理 vrf に着信する netstack パケットと kstack パケットの両方をフィルタリングできます。次の意味の警告が表示されます：「警告：非管理 VRF で構成された ACL は、その VRF の HTTP サービスでは有効になりません」。

ステップ 6	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		ストリーミングテレメトリの構成モードに入ります。

ステップ 7	（任意） **certificate** *certificate_path host_URL* 例： `switch(config-telemetry)# certificate /bootflash/server.key localhost`	既存の SSL/TLS 証明書を使用します。 EOR デバイスの場合、証明書もスタンバイ SUP にコピーする必要があります。
ステップ 8	（任意） トランスポート VRF を指定するか、gRPC トランスポートのテレメトリ圧縮を有効にします。 例： `switch(config-telemetry)#`	• **destination-profile** コマンドを入力して、デフォルトの接続先プロファイルを指定します。 • 次のコマンドを任意で入力します。

	コマンドまたはアクション	目的
	destination-profile switch(conf-tm-dest-profile)# **use-vrf default** switch(conf-tm-dest-profile)# **use-compression gzip**	• **use-vrf** *vrf* で接続先 VRF を指定します。 • **use-compression gzip** を使用して、接続先の圧縮方法を指定します。 （注） **use-vrf** コマンドを構成した後、新しい VRF 内に新しい接続先 IP アドレスを構成する必要があります。ただし、接続先を構成解除して再構成することにより、同じ接続先 IP アドレスを再利用できます。このアクションにより、テレメトリ データは新しい VRF でも同じ接続先 IP アドレスにストリーミングされます。
ステップ 9	**sensor-group** *sgrp_id* 例： switch(config-telemetry)# **sensor-group 100** switch(conf-tm-sensor)#	ID *srgp_id* を持つセンサー グループを作成し、センサー グループ構成モードを開始します。 現在は、数字の ID 値のみサポートされています。センサー グループでは、テレメトリ レポートのモニタリング対象ノードを定義します。
ステップ 10	（任意） **data-source** *data-source-type* 例： switch(config-telemetry)# **data-source NX-API**	データ ソースを選択します。データ ソースとして DME または NX-API のいずれかを選択します。 （注） DME はデフォルトのデータ ソースです。
ステップ 11	**path** *sensor_path* **depth unbounded** [**filter-condition** *filter*] [**alias** *path_alias*] 例： • 次のコマンドは、NX-API ではなく、DME に適用されます： switch(conf-tm-sensor)# **path sys/bd/bd-[vlan-100] depth 0 filter-condition eq(l2BD.operSt, "down")**	ここでの制限なしとは、出力に子管理対象オブジェクト（MO）を含めることを意味します。したがって、POLL テレメトリ ストリームの場合、そのパスと EVENT のすべての子 MO が子 MO で行われた変更を取得します。 （注） これは、データ送信元 DME パスにのみ適用されます。

	コマンドまたはアクション	目的

	以下の構文を使用し、状態ベースのフィルタリングを使用して、**operSt** が **up** から **down** に変化したときにのみトリガーするようにします。MO が変化しても通知しません。
	`switch(conf-tm-sensor)# path sys/bd/bd-[vlan-100] depth 0 filter-condition and(updated(l2BD.operSt),eq(l2BD.operSt,"down"))`
	• 次のコマンドは、DME ではなく、NX-API に適用されます：
	`switch(conf-tm-sensor)# path "show interface" depth 0`
		センサー グループにセンサー パスを追加します。
		•
		• **depth** 設定では、センサー パスの取得レベルを指定します。**0 - 32**、**unbounded** の深さ設定がサポートされています。
		（注） **depth 0** デフォルトの深さです。
		NX-API ベースのセンサー パスは、**depth 0** のみを使用できます。
		イベント収集のパスがサブスクライブされている場合、深さは 0 とバウンドなしのみをサポートします。その他の値は 0 として扱われます。
		• オプションの **filter-condition** パラメータを指定して、イベントベースのサブスクリプション用の特定のフィルタを作成できます。
		状態ベースのフィルタ処理の場合、フィルタ処理は、状態が変化したときと、指定された状態でイベントが発生したときの両方を返します。つまり、**eq(l2Bd.operSt, "down")** の DN **sys/bd/bd-[vlan]** のフィルタ条件は、operSt が変更されたとき、および operSt が **down** である間に DN のプロパティが変更されたとき（VLAN が動作上 **down** である間に **no shutdown** コマンドが発行された場合など）にトリガーされます。

	コマンドまたはアクション	目的
		（注） query-condition パラメータ — DM の場合、DN に基づいて、次の構文で MOTL および一時データをフェッチするために query-conditi パラメータを指定できます。クリ条件「rsp-foreign-subtree=ephemeral」

ステップ 12	**destination-group** *dgrp_id*
	例：
	`switch(conf-tm-sensor)# destination-group 100`
	`switch(conf-tm-dest)#`
		接続先グループを作成して、接続先グループ構成モードを開始します。
		現在、*dgrp_id* は、数字の ID 値のみをサポートしています。

ステップ 13	（任意） **ip address** *ip_address* **port** *port* **protocol** *procedural-protocol* **encoding** *encoding-protocol*
	例：
	`switch(conf-tm-sensor)# ip address 171.70.55.69 port 50001 protocol gRPC encoding GPB`
	`switch(conf-tm-sensor)# ip address 171.70.55.69 port 50007 protocol HTTP encoding JSON`
	`switch(conf-tm-sensor)# ip address 171.70.55.69 port 50009 protocol UDP encoding JSON`
		エンコードされたテレメトリデータを受信する IPv4 IP アドレスとポートを指定します。
		（注） gRPC はデフォルトのトランスポート プロトコルです。
		GPB がデフォルトのエンコーディングです。

ステップ 14	*ip_version* **address** *ip_address* **port** *portnum* 例： • IPv4 の場合： `switch(conf-tm-dest)# ip address 1.2.3.4 port 50003`	発信データの接続先プロファイルを作成します。 接続先グループがサブスクリプションにリンクされている場合、テレメトリデータは、このプロファイルで指定されている IP アドレスとポートに送信されます。

ステップ 15	**subscription** *sub_id*
	例：
	`switch(conf-tm-dest)# subscription 100`
	`switch(conf-tm-sub)#`
		ID を持つサブスクリプション ノードを作成し、サブスクリプション構成モードを開始します。
		現在、*sub_id* は、数字の ID 値のみをサポートしています。

	コマンドまたはアクション	目的
		（注） DN にサブスクライブする場合は、イベントが確実にストリーミングされるように、その DN が REST を使用して DME でサポートされているかどうかを確認します。
ステップ 16	**snsr-grp** *sgrp_id* **sample-interval** *interval* 例： `switch(conf-tm-sub)# snsr-grp 100 sample-interval 15000`	ID *sgrp_id* のセンサー グループを現在のサブスクリプションにリンクして、データのサンプリング間隔（ミリ秒単位）を設定します。 間隔の値が 0 の場合、イベント ベースのサブスクリプションが作成され、テレメトリ データは、指定された MO での変更時にのみ送信されます。0 より大きい間隔値の場合、テレメトリ データが指定された間隔で定期的に送信される頻度に基いたサブスクリプションが作成されます。たとえば、間隔値が 15000 の場合、テレメトリ データは 15 秒ごとに送信されます。
ステップ 17	**dst-grp** *dgrp_id* 例： `switch(conf-tm-sub)# dst-grp 100`	ID *dgrp_id* を持つ接続先グループをこのサブスクリプションにリンクします。

CLI を使用したテレメトリの構成例

次の手順では、GPB エンコーディングを使用して 10 秒のリズムで単一のテレメトリ DME ストリームを構成する方法について説明します。

```
switch# configure terminal
switch(config)# feature telemetry
switch(config)# telemetry
switch(config-telemetry)# destination-group 1
switch(config-tm-dest)# ip address 171.70.59.62 port 50051 protocol gRPC encoding GPB
switch(config-tm-dest)# exit
switch(config-telemetry)# sensor group sg1
switch(config-tm-sensor)# data-source DME
switch(config-tm-dest)# path interface depth unbounded query-condition keep-data-type
switch(config-tm-dest)# subscription 1
switch(config-tm-dest)# dst-grp 1
switch(config-tm-dest)# snsr grp 1 sample interval 10000
```

この例では、sys/bgp ルート MO のデータを宛先 IP 1.2.3.4 ポート 50003 に 5 秒ごとにストリーミングするサブスクリプションを作成します。

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/bgp depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50003
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 5000
switch(conf-tm-sub)# dst-grp 100
```

次に、sys/intf のデータを 5 秒ごとに、宛先 IP 1.2.3.4 ポート 50003 にストリーミングし、test.pem を使用して検証された GPB エンコーディングを使用してストリームを暗号化するサブスクリプションの作成例を示します。

```
switch(config)# telemetry
switch(config-telemetry)# certificate /bootflash/test.pem foo.test.google.fr
switch(conf-tm-telemetry)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50003 protocol gRPC encoding GPB
switch(config-dest)# sensor-group 100
switch(conf-tm-sensor)# path sys/bgp depth 0
switch(conf-tm-sensor)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 5000
switch(conf-tm-sub)# dst-grp 100
```

この例では、sys/cdp のデータを接続先 IP 1.2.3.4 ポート 50004 に 15 秒ごとにストリーミングするサブスクリプションを作成します。

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/cdp depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 15000
switch(conf-tm-sub)# dst-grp 100
```

この例では、750 秒ごとに **show** コマンド データのケイデンス ベースのコレクションを作成します。

```
switch(config)# telemetry
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# ip address 172.27.247.72 port 60001 protocol gRPC encoding GPB
switch(conf-tm-dest)# sensor-group 1
switch(conf-tm-sensor# data-source NX-API
switch(conf-tm-sensor)# path "show system resources" depth 0
switch(conf-tm-sensor)# path "show version" depth 0
switch(conf-tm-sensor)# path "show environment power" depth 0
switch(conf-tm-sensor)# path "show environment fan" depth 0
switch(conf-tm-sensor)# path "show environment temperature" depth 0
switch(conf-tm-sensor)# path "show process cpu" depth 0
switch(conf-tm-sensor)# path "show nve peers" depth 0
switch(conf-tm-sensor)# path "show nve vni" depth 0
switch(conf-tm-sensor)# path "show nve vni 4002 counters" depth 0
switch(conf-tm-sensor)# path "show int nve 1 counters" depth 0
switch(conf-tm-sensor)# path "show policy-map vlan" depth 0
```

```
switch(conf-tm-sensor)# path "show ip access-list test" depth 0
switch(conf-tm-sensor)# path "show system internal access-list resource utilization"
depth 0
switch(conf-tm-sensor)# subscription 1
switch(conf-tm-sub)# dst-grp 1
switch(conf-tm-dest)# snsr-grp 1 sample-interval 750000
```

この例では、sys/fm のイベント ベースのサブスクリプションを作成します。sys/fm MO に変更がある場合にのみ、データは接続先にストリーミングされます。

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/fm depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50005
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 0
switch(conf-tm-sub)# dst-grp 100
```

動作中に、サンプル間隔を変更することで、センサー グループを周波数ベースからイベントベースに変更したり、イベントベースから周波数ベースに変更したりできます。この例では、センサー グループを前の例から頻度ベースに変更します。次のコマンドの後、テレメトリ アプリケーションは 7 秒ごとに sys/fm データの接続先へのストリーミングを開始します。

```
switch(config)# telemetry
switch(config-telemetry)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 7000
```

複数のセンサー グループと接続先を 1 つのサブスクリプションにリンクできます。この例のサブスクリプションは、イーサネット ポート 1/1 のデータを 4 つの異なる接続先に 10 秒ごとにストリーミングします。

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/intf/phys-[eth1/1] depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# ip address 1.2.3.4 port 50005
switch(conf-tm-sensor)# destination-group 200
switch(conf-tm-dest)# ip address 5.6.7.8 port 50001 protocol HTTP encoding JSON
switch(conf-tm-dest)# ip address 1.4.8.2 port 60003
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 10000
switch(conf-tm-sub)# dst-grp 100
switch(conf-tm-sub)# dst-grp 200
```

次に、センサー グループに複数のパスを含め、接続先グループに複数の接続先プロファイルを含め、サブスクリプションを複数のセンサー グループと宛先グループにリンクできる例を表示します。

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/intf/phys-[eth1/1] depth 0
```

```
switch(conf-tm-sensor)# path sys/epId-1 depth 0
switch(conf-tm-sensor)# path sys/bgp/inst/dom-default depth 0

switch(config-telemetry)# sensor-group 200
switch(conf-tm-sensor)# path sys/cdp depth 0
switch(conf-tm-sensor)# path sys/ipv4 depth 0

switch(config-telemetry)# sensor-group 300
switch(conf-tm-sensor)# path sys/fm depth 0
switch(conf-tm-sensor)# path sys/bgp depth 0

switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# ip address 4.3.2.5 port 50005

switch(conf-tm-dest)# destination-group 200
switch(conf-tm-dest)# ip address 5.6.7.8 port 50001

switch(conf-tm-dest)# destination-group 300
switch(conf-tm-dest)# ip address 1.2.3.4 port 60003

switch(conf-tm-dest)# subscription 600
switch(conf-tm-sub)# snsr-grp 100 sample-interval 7000
switch(conf-tm-sub)# snsr-grp 200 sample-interval 20000
switch(conf-tm-sub)# dst-grp 100
switch(conf-tm-sub)# dst-grp 200

switch(conf-tm-dest)# subscription 900
switch(conf-tm-sub)# snsr-grp 200 sample-interval 7000
switch(conf-tm-sub)# snsr-grp 300 sample-interval 0
switch(conf-tm-sub)# dst-grp 100
switch(conf-tm-sub)# dst-grp 300
```

この例に示すように、**show running-config telemetry** コマンドを使用してテレメトリ構成を確認できます。

```
switch(config)# telemetry
switch(config-telemetry)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50003
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# end
switch# show run telemetry

!Command: show running-config telemetry
!Time: Thu Oct 13 21:10:12 2016

version 7.0(3)I5(1)
feature telemetry

telemetry
destination-group 100
ip address 1.2.3.4 port 50003 protocol gRPC encoding GPB
ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB
```

この例に示すように、**use-vrf** コマンドと **use-compression gzip** コマンドを使用して、gRPC のトランスポート VRF とテレメトリ データ圧縮を指定できます。

```
switch(config)# telemetry
switch(config-telemetry)# destination-profile
```

```
switch(conf-tm-dest-profile)# use-vrf default
switch(conf-tm-dest-profile)# use-compression gzip
switch(conf-tm-dest-profile)# sensor-group 1
switch(conf-tm-sensor)# path sys/bgp depth unbounded
switch(conf-tm-sensor)# destination-group 1
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# subscription 1
switch(conf-tm-sub)# dst-grp 1
switch(conf-tm-sub)# snsr-grp 1 sample-interval 10000
```

テレメトリの構成と統計情報の表示

次の NX-OS CLI **show** コマンドを使用して、テレメトリの構成、統計情報、エラー、およびセッション情報を表示します。

show telemetry control database

次に、テレメトリの構成を反映している内部データベースのコマンドを表示します。

```
switch# show telemetry control database ?
  <CR>
  >                   Redirect it to a file
  >>                  Redirect it to a file in append mode
  destination-groups  Show destination-groups
  destinations        Show destinations
  sensor-groups       Show sensor-groups
  sensor-paths        Show sensor-paths
  subscriptions       Show subscriptions
  |                   Pipe command output to filter

switch# show telemetry control database

Subscription Database size = 1

------------------------------------------------------------------------------
Subscription ID      Data Collector Type
------------------------------------------------------------------------------
100                  DME NX-API

Sensor Group Database size = 1

------------------------------------------------------------------------------
Sensor Group ID  Sensor Group type  Sampling interval(ms)  Linked subscriptions
------------------------------------------------------------------------------
100              Timer              10000(Running)         1

Sensor Path Database size = 1

------------------------------------------------------------------------------
Subscribed Query Filter  Linked Groups  Sec Groups  Retrieve level  Sensor Path
------------------------------------------------------------------------------
No                       1              0           Full            sys/fm

Destination group Database size = 2

------------------------------------------------------------------------------
Destination Group ID  Refcount
------------------------------------------------------------------------------
100                   1
```

```
Destination Database size = 2

------------------------------------------------------------------------------
Dst IP Addr     Dst Port   Encoding   Transport  Count
------------------------------------------------------------------------------
192.168.20.111       12345      JSON       HTTP       1
192.168.20.123 50001      GPB        gRPC       1
```

show telemetry control stats

このコマンドは、テレメトリの構成についての内部データベースの統計を表示します。

```
switch# show telemetry control stats
show telemetry control stats entered

------------------------------------------------------------------------------
Error Description                                              Error Count
------------------------------------------------------------------------------
Chunk allocation failures                                      0
Sensor path Database chunk creation failures                   0
Sensor Group Database chunk creation failures                  0
Destination Database chunk creation failures                   0
Destination Group Database chunk creation failures             0
Subscription Database chunk creation failures                  0
Sensor path Database creation failures                         0
Sensor Group Database creation failures                        0
Destination Database creation failures                         0
Destination Group Database creation failures                   0
Subscription Database creation failures                        0
Sensor path Database insert failures                           0
Sensor Group Database insert failures                          0
Destination Database insert failures                           0
Destination Group Database insert failures                     0
Subscription insert to Subscription Database failures          0
Sensor path Database delete failures                           0
Sensor Group Database delete failures                          0
Destination Database delete failures                           0
Destination Group Database delete failures                     0
Delete Subscription from Subscription Database failures        0
Sensor path delete in use                                      0
Sensor Group delete in use                                     0
Destination delete in use                                      0
Destination Group delete in use                                0
Delete destination(in use) failure count                       0
Failed to get encode callback                                  0
Sensor path Sensor Group list creation failures                0
Sensor path prop list creation failures                        0
Sensor path sec Sensor path list creation failures             0
Sensor path sec Sensor Group list creation failures            0
Sensor Group Sensor path list creation failures                0
Sensor Group Sensor subs list creation failures                0
Destination Group subs list creation failures                  0
Destination Group Destinations list creation failures          0
Destination Destination Groups list creation failures          0
Subscription Sensor Group list creation failures               0
Subscription Destination Groups list creation failures         0
Sensor Group Sensor path list delete failures                  0
Sensor Group Subscriptions list delete failures                0
Destination Group Subscriptions list delete failures           0
Destination Group Destinations list delete failures            0
Subscription Sensor Groups list delete failures                0
Subscription Destination Groups list delete failures           0
```

```
Destination Destination Groups list delete failures          0
Failed to delete Destination from Destination Group          0
Failed to delete Destination Group from Subscription         0
Failed to delete Sensor Group from Subscription              0
Failed to delete Sensor path from Sensor Group               0
Failed to get encode callback                                0
Failed to get transport callback                             0
switch#  Destination Database size = 1

-------------------------------------------------------------------------------
Dst IP Addr     Dst Port   Encoding   Transport  Count
-------------------------------------------------------------------------------
192.168.20.123 50001      GPB        gRPC       1
```

show telemetry data collector brief

このコマンドは、データ収集に関する簡略化した統計情報を表示します。

```
switch# show telemetry data collector brief

------------------------------------------------------------------------
Collector Type       Successful Collections     Failed Collections
------------------------------------------------------------------------
DME                   143                        0
```

show telemetry data collector details

このコマンドは、すべてのセンサーパスの詳細を含む、データ収集に関する詳細な統計情報を表示します。

```
switch# show telemetry data collector details

-------------------------------------------------------------------------------
Succ Collections      Failed Collections     Sensor Path
-------------------------------------------------------------------------------
150                   0                      sys/fm
```

show telemetry event collector errors

このコマンドは、イベント収集に関するエラー統計情報を表示します。

```
switch# show telemetry event collector errors

-------------------------------------------------------------------------------
Error Description                                 Error Count
-------------------------------------------------------------------------------
APIC-Cookie Generation Failures                    - 0
Authentication Failures                            - 0
Authentication Refresh Failures                    - 0
Authentication Refresh Timer Start Failures        - 0
Connection Timer Start Failures                    - 0
Connection Attempts                                - 3
Dme Event Subscription Init Failures               - 0
Event Data Enqueue Failures                        - 0
Event Subscription Failures                        - 0
Event Subscription Refresh Failures                - 0
```

```
Pending Subscription List Create Failures          - 0
Subscription Hash Table Create Failures            - 0
Subscription Hash Table Destroy Failures           - 0
Subscription Hash Table Insert Failures            - 0
Subscription Hash Table Remove Failures            - 0
Subscription Refresh Timer Start Failures          - 0
Websocket Connect Failures                         - 0
```

show telemetry event collector stats

このコマンドは、すべてのセンサーパスの内訳を含むイベント収集に関する統計情報を表示します。

```
switch# show telemetry event collector stats

------------------------------------------------------------------------------
Collection Count  Latest Collection Time    Sensor Path
------------------------------------------------------------------------------
```

show telemetry control pipeline stats

このコマンドは、テレメトリ パイプラインの統計情報を表示します。

```
switch# show telemetry pipeline stats
Main Statistics:
    Timers:
        Errors:
            Start Fail        =     0

    Data Collector:
        Errors:
            Node Create Fail  =     0

    Event Collector:
        Errors:
            Node Create Fail  =     0    Node Add Fail     =     0
            Invalid Data      =     0

    Memory:
            Allowed Memory Limit               = 1181116006 bytes
            Occupied Memory                    = 93265920 bytes

Queue Statistics:
    Request Queue:
        High Priority Queue:
            Info:
                Actual Size       =    50    Current Size      =     0
                Max Size          =     0    Full Count        =     0

            Errors:
                Enqueue Error     =     0    Dequeue Error     =     0

        Low Priority Queue:
            Info:
                Actual Size       =    50    Current Size      =     0
                Max Size          =     0    Full Count        =     0

            Errors:
                Enqueue Error     =     0    Dequeue Error     =     0
```

```
Data Queue:
    High Priority Queue:
        Info:
            Actual Size       =    50    Current Size      =     0
            Max Size          =     0    Full Count        =     0

        Errors:
            Enqueue Error     =     0    Dequeue Error     =     0

    Low Priority Queue:
        Info:
            Actual Size       =    50    Current Size      =     0
            Max Size          =     0    Full Count        =     0

        Errors:
            Enqueue Error     =     0    Dequeue Error     =     0
```

show telemetry transport

次に、構成されているすべての転送セッションの例を表示します。

```
switch# show telemetry transport

Session Id      IP Address      Port       Encoding   Transport  Status
--------------------------------------------------------------------------------
0               192.168.20.123   50001      GPB        gRPC       Connected
```

表 *1 : show telemetry transport* の構文の説明

構文	説明
show	実行中のシステム情報を表示します。
telemetry	テレメトリ情報を表示します
トランスポート	テレメトリのトランスポート情報を表示します。
session_id	（オプション）セッション ID
stats	（オプション）すべてのテレメトリ統計情報を表示します。
errors	（オプション）すべてのテレメトリ エラー情報を表示します。
読み取り専用	（オプション）
TABLE_transport_info	（オプション）トランスポート情報
session_idx	（オプション）セッション ID
ip_address	（オプション）トランスポート IP アドレス

構文	説明
port	（任意）トランスポート ポート
dest_info	（オプション）接続先情報
encoding_type	（オプション）エンコーディング タイプ
transport_type	（オプション）トランスポート タイプ
transport_status	（オプション）トランスポート ステータス
transport_security_cert_fname	（オプション）トランスポートセキュリティ ファイル名
transport_last_connected	（オプション）最後に接続されたトランスポート
transport_last_disconnected	（オプション）この宛先構成が最後に削除された時刻
transport_errors_count	（オプション）トランスポート エラー数
transport_last_tx_error	（オプション）トランスポートの最後の送信エラー
transport_statistics	（オプション）トランスポート統計情報
t_session_id	（オプション）トランスポートセッションID
connect_statistics	（オプション）接続統計情報
connect_count	（オプション）接続数
last_connected	（オプション）最終接続のタイムスタンプ
Disconnect_count	（オプション）切断数
last_disconnected	（オプション）この宛先構成が最後に削除された時刻
trans_statistics	（オプション）トランスポート統計情報
compression	（オプション）圧縮ステータス
source_interface_name	（オプション）送信元インターフェイス名
source_interface_ip	（オプション）送信元インターフェイス IP
transmit_count	（オプション）送信数
last_tx_time	（オプション）最終送信時刻

構文	説明
min_tx_time	（オプション）最小送信時間
max_tx_time	（オプション）最大送信時間
avg_tx_time	（オプション）平均送信時間
cur_tx_time	（オプション）現在の送信時間
transport_errors	（オプション）トランスポート エラー
connect_errors	（オプション）接続エラー
connect_errors_count	（オプション）接続エラー数
trans_errors	（オプション）トランスポート エラー
trans_errors_count	（オプション）トランスポート エラー数
last_tx_error	（オプション）最後のトランスポートエラー
last_tx_return_code	（オプション）最後の送信戻りコード
transport_retry_stats	（オプション）再試行統計情報
ts_event_retry_bytes	（オプション）イベント再試行バッファ サイズ
ts_timer_retry_bytes	（オプション）タイマー再試行バッファ サイズ
ts_event_retry_size	（オプション）イベント再試行メッセージ数
ts_timer_retry_size	（オプション）タイマー再試行メッセージ数
ts_retries_sent	（オプション）送信された再試行回数
ts_retries_dropped	（オプション）ドロップされた再試行回数
event_retry_bytes	（オプション）イベント再試行バッファ サイズ
timer_retry_bytes	（オプション）タイマー再試行バッファ サイズ
retries_sent	（オプション）送信された再試行回数
retries_dropped	（オプション）ドロップされた再試行回数
retry_buffer_size	（オプション）再試行バッファ サイズ

show telemetry transport <session-id>

次のコマンドでは、特定の転送セッションの詳細なセッション情報が表示されます。

```
switch# show telemetry transport 0

Session Id:          0
IP Address:Port      192.168.20.123:50001
Transport:           gRPC
Status:              Disconnected
Last Connected:      Fri Sep 02 11:45:57.505 UTC
Last Disconnected:   Never
Tx Error Count:      224
Last Tx Error:       Fri Sep 02 12:23:49.555 UTC

switch# show telemetry transport 1

Session Id:          1
IP Address:Port      10.30.218.56:51235
Transport:           HTTP
Status:              Disconnected
Last Connected:      Never
Last Disconnected:   Never
Tx Error Count:      3
Last Tx Error:       Wed Apr 19 15:56:51.617 PDT
```

show telemetry transport <session-id> stats

次に、特定の転送セッションの詳細のコマンドを示します。

```
Session Id:                    0
Transmission Stats
   Compression:                disabled
   Source Interface:           not set()
   Transmit Count:             319297
   Last TX time:               Fri Aug 02 03:51:15.287 UTC
   Min Tx Time:                1                    ms
   Max Tx Time:                3117                 ms
   Avg Tx Time:                3                    ms
   Cur Tx Time:                1                    ms
```

show telemetry transport <session-id> errors

次のコマンドでは、特定の転送セッションの詳細なエラーの統計情報が表示されます。

```
switch# show telemetry transport 0 errors
Session Id:                 0
Connection Errors
Connection Error Count:     0
Transmission Errors
Tx Error Count:             30
Last Tx Error:              Thu Aug 01 04:39:47.083 UTC
Last Tx Return Code:        No error
```

テレメトリ ログとトレース情報の表示

ログとトレース情報を表示するには、次の NX-OS CLI コマンドを使用します。

テクニカル サポート テレメトリを表示

この NX-OS CLI コマンドは、テクニカル サポート ログからテレメトリ ログの内容を収集します。この例では、コマンド出力がブートフラッシュのファイルにリダイレクトされます。

```
switch# show tech-support telemetry > bootflash:tmst.log
```

tmtrace.bin

この BASH シェル コマンドは、テレメトリ トレースを収集して出力します。

```
switch# configure terminal
switch(config)# feature bash
switch(config)# run bash
bash-4.2$ tmtrace.bin -d tm-errors
bash-4.2$ tmtrace.bin -d tm-logs
bash-4.2$ tmtrace.bin -d tm-events
```

例：

```
bash-4.2$ tmtrace.bin -d tm-logs
[01/25/17 22:52:24.563 UTC 1 29130] [3944724224][tm_ec_dme_auth.c:59] TM_EC: Authentication
 refresh url http://127.0.0.1/api/aaaRefresh.json
[01/25/17 22:52:24.565 UTC 2 29130] [3944724224][tm_ec_dme_rest_util.c:382] TM_EC:
Performed POST request on http://127.0.0.1/api/aaaRefresh.json
[01/25/17 22:52:24.566 UTC 3 29130] [3944724224][tm_mgd_timers.c:114] TM_MGD_TIMER:
Starting leaf timer for leaf:0x11e17ea4 time_in_ms:540000
[01/25/17 22:52:45.317 UTC 4 29130] [3944724224][tm_ec_dme_event_subsc.c:790] TM_EC:
Event subscription database size 0
[01/25/17 22:52:45.317 UTC 5 29130] [3944724224][tm_mgd_timers.c:114] TM_MGD_TIMER:
Starting leaf timer for leaf:0x11e17e3c time_in_ms:50000
bash-4.2#
```

（注） **tm-logs** オプションは冗長であるため、デフォルトでは有効になっていません。

`tmtrace.bin -LD tm-logs` コマンドで **tm-logs** を有効にします。

`tmtrace.bin -LW tm-logs` コマンドを使用して **tm-logs** を無効にします。

show system internal telemetry trace

show system internal telemetry trace [**tm-events** | **tm-errors** |**tm-logs** | **all**] コマンドは、システムの内部テレメトリ トレース情報を表示します。

```
switch# show system internal telemetry trace all
Telemetry All Traces:
Telemetry Error Traces:
[07/26/17 15:22:29.156 UTC 1 28577] [3960399872][tm_cfg_api.c:367] Not able to destroy
dest profile list for config node rc:-1610612714 reason:Invalid argument
[07/26/17 15:22:44.972 UTC 2 28577] [3960399872][tm_stream.c:248] No subscriptions for
destination group 1
[07/26/17 15:22:49.463 UTC 3 28577] [3960399872][tm_stream.c:576] TM_STREAM: Subscriptoin
 1 does not have any sensor groups
```

```
3 entries printed
Telemetry Event Traces:
[07/26/17 15:19:40.610 UTC 1 28577] [3960399872][tm_debug.c:41] Telemetry xostrace buffers
 initialized successfully!
[07/26/17 15:19:40.610 UTC 2 28577] [3960399872][tm.c:744] Telemetry statistics created
 successfully!
[07/26/17 15:19:40.610 UTC 3 28577] [3960399872][tm_init_n9k.c:97] Platform intf:
grpc_traces:compression,channel
switch#

switch# show system internal telemetry trace tm-logs
Telemetry Log Traces:
0 entries printed
switch#
switch# show system internal telemetry trace tm-events
Telemetry Event Traces:
[07/26/17 15:19:40.610 UTC 1 28577] [3960399872][tm_debug.c:41] Telemetry xostrace buffers
 initialized successfully!
[07/26/17 15:19:40.610 UTC 2 28577] [3960399872][tm.c:744] Telemetry statistics created
 successfully!
[07/26/17 15:19:40.610 UTC 3 28577] [3960399872][tm_init_n9k.c:97] Platform intf:
grpc_traces:compression,channel
[07/26/17 15:19:40.610 UTC 4 28577] [3960399872][tm_init_n9k.c:207] Adding telemetry to
 cgroup
[07/26/17 15:19:40.670 UTC 5 28577] [3960399872][tm_init_n9k.c:215] Added telemetry to
cgroup successfully!

switch# show system internal telemetry trace tm-errors
Telemetry Error Traces:
0 entries printed
switch#
```

NX-API を使用したテレメトリの構成

Configuring Telemetry Using the NX-API

In the object model of the switch DME, the configuration of the telemetry feature is defined in a hierarchical structure of objects as shown in the section "Telemetry Model in the DME." Following are the main objects to be configured:

- **fmEntity** — Contains the NX-API and Telemetry feature states.
  - **fmNxapi** — Contains the NX-API state.
  - **fmTelemetry** — Contains the Telemetry feature state.
- **telemetryEntity** — Contains the telemetry feature configuration.
  - **telemetrySensorGroup** — Contains the definitions of one or more sensor paths or nodes to be monitored for telemetry. The telemetry entity can contain one or more sensor groups.
    - **telemetryRtSensorGroupRel** — Associates the sensor group with a telemetry subscription.
    - **telemetrySensorPath** — A path to be monitored. The sensor group can contain multiple objects of this type.

- **telemetryDestGroup** — Contains the definitions of one or more destinations to receive telemetry data. The telemetry entity can contain one or more destination groups.
  - **telemetryRtDestGroupRel** — Associates the destination group with a telemetry subscription.
  - **telemetryDest** — A destination address. The destination group can contain multiple objects of this type.
- **telemetrySubscription** — Specifies how and when the telemetry data from one or more sensor groups is sent to one or more destination groups.
  - **telemetryRsDestGroupRel** — Associates the telemetry subscription with a destination group.
  - **telemetryRsSensorGroupRel** — Associates the telemetry subscription with a sensor group.
- **telemetryCertificate** — Associates the telemetry subscription with a certificate and hostname.

To configure the telemetry feature using the NX-API, you must construct a JSON representation of the telemetry object structure and push it to the DME with an HTTP or HTTPS POST operation.

Note For detailed instructions on using the NX-API, see the *Cisco Nexus 3000 and 9000 Series NX-API REST SDK User Guide and API Reference*.

Before you begin

Your switch must be running Cisco NX-OS Release 7.3(0)I5(1) or a later release.

Your switch must be configured to run the NX-API from the CLI:

```
switch(config)# feature nxapi
```

NX-API sends telemetry data over management VRF:

```
switch(config)# nxapi use-vrf management
```

Procedure

	Command or Action	Purpose
ステップ 1	Enable the telemetry feature. **Example:** `{ "fmEntity" : { "children" : [{ "fmTelemetry" : { "attributes" : { "adminSt" : "enabled" } } }`	The root element is **fmTelemetry** and the base path for this element is `sys/fm`. Configure the **adminSt** attribute as `enabled`.

	Command or Action	Purpose

	<pre> }
	    ]
	  }
	}</pre>	

ステップ 2	Create the root level of the JSON payload to describe the telemetry configuration.
	Example:
	<pre>{
	    "telemetryEntity": {
	        "attributes": {
	            "dn": "sys/tm"
	        },
	    }
	}</pre>
		The root element is **telemetryEntity** and the base path for this element is `sys/tm`. Configure the **dn** attribute as `sys/tm`.

ステップ 3	Create a sensor group to contain the defined sensor paths.
	Example:
	<pre>"telemetrySensorGroup": {
	    "attributes": {
	        "id": "10",
	        "rn": "sensor-10"
	        "dataSrc": "NX-API"
	    }, "children": [{
	    }]
	}</pre>
		A telemetry sensor group is defined in an object of class **telemetrySensorGroup**. Configure the following attributes of the object:
		• **id** — An identifier for the sensor group. Currently only numeric ID values are supported.
		• **rn** — The relative name of the sensor group object in the format: **sensor-***id*.
		• **dataSrc** — Selects the data source from **DEFAULT**, **DME**, or **NX-API**.
		Children of the sensor group object will include sensor paths and one or more relation objects (**telemetryRtSensorGroupRel**) to associate the sensor group with a telemetry subscription.

ステップ 4	(Optional) Add an SSL/TLS certificate and a host.
	Example:
	<pre>{
	    "telemetryCertificate": {
	        "attributes": {
	            "filename": "root.pem"
	             "hostname": "c.com"
	         }
	    }
	}</pre>
		The **telemetryCertificate** defines the location of the SSL/TLS certificate with the telemetry subscription/destination.

ステップ 5	Define a telemetry destination group. **Example:**	A telemetry destination group is defined in **telemetryEntity**. Configure the id attribute.

	Command or Action	Purpose

	<pre>{
	      "telemetryDestGroup": {
	        "attributes": {
	          "id": "20"
	        }
	      }
	}</pre>	

ステップ 6	Define a telemetry destination profile.
	Example:
	<pre>{
	   "telemetryDestProfile": {
	      "attributes": {
	         "adminSt": "enabled"
	      },
	      "children": [
	         {
	
	"telemetryDestOptSourceInterface": {
	               "attributes": {
	                  "name": "lo0"
	               }
	            }
	         }
	      ]
	   }
	}</pre>
		A telemetry destination profile is defined in **telemetryDestProfile**.
		• Configure the **adminSt** attribute as `enabled`.
		• Under **telemetryDestOptSourceInterface**, configure the **name** attribute with an interface name to stream data from the configured interface to a destination with the source IP address.

ステップ 7	Define one or more telemetry destinations, consisting of an IP address and port number to which telemetry data will be sent.
	Example:
	<pre>{
	    "telemetryDest": {
	        "attributes": {
	             "addr": "1.2.3.4",
	             "enc": "GPB",
	             "port": "50001",
	             "proto": "gRPC",
	             "rn":
	"addr-[1.2.3.4]-port-50001"
	        }
	    }
	}</pre>
		A telemetry destination is defined in an object of class **telemetryDest**. Configure the following attributes of the object:
		• **addr** — The IP address of the destination.
		• **port** — The port number of the destination.
		• **rn** — The relative name of the destination object in the format: **path-[***path***]**.
		• **enc** — The encoding type of the telemetry data to be sent. NX-OS supports:
		  • Google protocol buffers (GPB) for gRPC.
		  • JSON for C.
		  • GPB or JSON for UDP and secure UDP (DTLS).

	Command or Action	Purpose

		• **proto** — The transport protocol type of the telemetry data to be sent. NX-OS supports:
		  • gRPC
		  • HTTP
		  • VUDP and secure UDP (DTLS)

ステップ 8	Create a telemetry subscription to configure the telemetry behavior. **Example:** `"telemetrySubscription": { "attributes": { "id": "30", "rn": "subs-30" }, "children": [{ }] }`	A telemetry subscription is defined in an object of class **telemetrySubscription**. Configure the following attributes of the object: • **id** — An identifier for the subscription. Currently only numeric ID values are supported. • **rn** — The relative name of the subscription object in the format: **subs-***id*. Children of the subscription object will include relation objects for sensor groups (**telemetryRsSensorGroupRel**) and destination groups (**telemetryRsDestGroupRel**).
ステップ 9	Add the sensor group object as a child object to the **telemetrySubscription** element under the root element (**telemetryEntity**). **Example:** `{ "telemetrySubscription": { "attributes": { "id": "30" } "children": [{ "telemetryRsSensorGroupRel": { "attributes": { "sampleIntvl": "5000", "tDn": "sys/tm/sensor-10" } } } ] } }`	

	Command or Action	Purpose
ステップ 10	Create a relation object as a child object of the subscription to associate the subscription to the telemetry sensor group and to specify the data sampling behavior. **Example:** `"telemetryRsSensorGroupRel": { "attributes": { "rType": "mo", "rn": "rssensorGroupRel-[sys/tm/sensor-10]", "sampleIntvl": "5000", "tCl": "telemetrySensorGroup", "tDn": "sys/tm/sensor-10", "tType": "mo" } }`	The relation object is of class **telemetryRsSensorGroupRel** and is a child object of **telemetrySubscription**. Configure the following attributes of the relation object: • **rn** — The relative name of the relation object in the format: **rssensorGroupRel-[sys/tm/***sensor-group-id***]**. • **sampleIntvl** — The data sampling period in milliseconds. An interval value of 0 creates an event-based subscription, in which telemetry data is sent only upon changes under the specified MO. An interval value greater than 0 creates a frequency-based subscription, in which telemetry data is sent periodically at the specified interval. For example, an interval value of 15000 results in the sending of telemetry data every 15 seconds. • **tCl** — The class of the target (sensor group) object, which is **telemetrySensorGroup**. • **tDn** — The distinguished name of the target (sensor group) object, which is **sys/tm/***sensor-group-id*. • **rType** — The relation type, which is **mo** for managed object. • **tType** — The target type, which is **mo** for managed object.
ステップ 11	Define one or more sensor paths or nodes to be monitored for telemetry. **Example:** Single sensor path `{ "telemetrySensorPath": { "attributes": { "path": "sys/cdp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "", "path": "sys/fm/bgp", "secondaryGroup": "0",`	A sensor path is defined in an object of class **telemetrySensorPath**. Configure the following attributes of the object: • **path** — The path to be monitored. • **rn** — The relative name of the path object in the format: **path-[***path***]** • **depth** — The retrieval level for the sensor path. A depth setting of **0** retrieves only the root MO properties. • **filterCondition** — (Optional) Creates a specific filter for event-based

	Command or Action	Purpose

	"secondaryPath": "",
	"depth": "0"
	}
	}
	}
	
	Example:
	Single sensor path for NX-API
	
	{
	"telemetrySensorPath": {
	"attributes": {
	"path": "show interface",
	
	"path": "show bgp",
	"rn": "path-[sys/cdp]",
	"excludeFilter": "",
	"filterCondition": "",
	"path": "sys/fm/bgp",
	"secondaryGroup": "0",
	"secondaryPath": "",
	"depth": "0"
	}
	}
	}
	
	Example:
	Multiple sensor paths
	
	{
	"telemetrySensorPath": {
	"attributes": {
	"path": "sys/cdp",
	"rn": "path-[sys/cdp]",
	"excludeFilter": "",
	"filterCondition": "",
	"path": "sys/fm/bgp",
	"secondaryGroup": "0",
	"secondaryPath": "",
	"depth": "0"
	}
	}
	},
	{
	"telemetrySensorPath": {
	"attributes": {
	"excludeFilter": "",
	"filterCondition": "",
	"path": "sys/fm/dhcp",
	"secondaryGroup": "0",
	"secondaryPath": "",
	"depth": "0"
	}
	}
	}
	
	Example:
		subscriptions. The DME provides the filter expressions. For more information regarding filtering, see the Cisco APIC REST API Usage Guidelines on composing queries: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/2-x/rest_cfg/2_1_x/b_Cisco_APIC_REST_API_Configuration_Guide/b_Cisco_APIC_REST_API_Configuration_Guide_chapter_01.html#d25e1534a1635

	Command or Action	Purpose
	Single sensor path filtering for BGP disable events: `{ "telemetrySensorPath": { "attributes": { "path": "sys/cdp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "eq(fmBgp.operSt.\"disabled\")", "path": "sys/fm/bgp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } }`	
ステップ 12	Add sensor paths as child objects to the sensor group object (**telemetrySensorGroup**).	
ステップ 13	Add destinations as child objects to the destination group object (**telemetryDestGroup**).	
ステップ 14	Add the destination group object as a child object to the root element (**telemetryEntity**).	
ステップ 15	Create a relation object as a child object of the telemetry sensor group to associate the sensor group to the subscription. **Example:** `"telemetryRtSensorGroupRel": { "attributes": { "rn": "rtsensorGroupRel-[sys/tm/subs-30]", "tCl": "telemetrySubscription", "tDn": "sys/tm/subs-30" } }`	The relation object is of class **telemetryRtSensorGroupRel** and is a child object of **telemetrySensorGroup**. Configure the following attributes of the relation object: • **rn** — The relative name of the relation object in the format: **rtsensorGroupRel-[sys/tm/***subscription-id***]**. • **tCl** — The target class of the subscription object, which is **telemetrySubscription**. • **tDn** — The target distinguished name of the subscription object, which is **sys/tm/***subscription-id*.
ステップ 16	Create a relation object as a child object of the telemetry destination group to associate the destination group to the subscription. **Example:**	The relation object is of class **telemetryRtDestGroupRel** and is a child object of **telemetryDestGroup**. Configure the following attributes of the relation object:

	Command or Action	Purpose

	<pre>"telemetryRtDestGroupRel": {
	    "attributes": {
	        "rn":
	"rtdestGroupRel-[sys/tm/subs-30]",
	        "tCl":
	"telemetrySubscription",
	        "tDn": "sys/tm/subs-30"
	    }
	}</pre>
		• **rn** — The relative name of the relation object in the format: **rtdestGroupRel-[sys/tm/***subscription-id***]**.
		• **tCl** — The target class of the subscription object, which is **telemetrySubscription**.
		• **tDn** — The target distinguished name of the subscription object, which is **sys/tm/***subscription-id*.

ステップ **17**	Create a relation object as a child object of the subscription to associate the subscription to the telemetry destination group.
	Example:
	<pre>"telemetryRsDestGroupRel": {
	    "attributes": {
	        "rType": "mo",
	        "rn":
	"rsdestGroupRel-[sys/tm/dest-20]",
	        "tCl": "telemetryDestGroup",
	        "tDn": "sys/tm/dest-20",
	        "tType": "mo"
	    }
	}</pre>
		The relation object is of class **telemetryRsDestGroupRel** and is a child object of **telemetrySubscription**. Configure the following attributes of the relation object:
		• **rn** — The relative name of the relation object in the format: **rsdestGroupRel-[sys/tm/***destination-group-id***]**.
		• **tCl** — The class of the target (destination group) object, which is **telemetryDestGroup**.
		• **tDn** — The distinguished name of the target (destination group) object, which is **sys/tm/***destination-group-id*.
		• **rType** — The relation type, which is **mo** for managed object.
		• **tType** — The target type, which is **mo** for managed object.

ステップ **18**	Send the resulting JSON structure as an HTTP/HTTPS POST payload to the NX-API endpoint for telemetry configuration.	The base path for the telemetry entity is `sys/tm` and the NX-API endpoint is: `{{URL}}/api/node/mo/sys/tm.json`

Example

The following is an example of all the previous steps collected into one POST payload (note that some attributes may not match):

```
{
  "telemetryEntity": {
    "children": [{
      "telemetrySensorGroup": {
        "attributes": {
          "id": "10"
        }
```

```
        "children": [{
          "telemetrySensorPath": {
            "attributes": {
              "excludeFilter": "",
              "filterCondition": "",
              "path": "sys/fm/bgp",
              "secondaryGroup": "0",
              "secondaryPath": "",
              "depth": "0"
            }
          }
        }
        ]
      }
    },
    {
      "telemetryDestGroup": {
        "attributes": {
          "id": "20"
        }
        "children": [{
          "telemetryDest": {
            "attributes": {
              "addr": "10.30.217.80",
              "port": "50051",
              "enc": "GPB",
              "proto": "gRPC"
            }
          }
        }
        ]
      }
    },
    {
      "telemetrySubscription": {
        "attributes": {
          "id": "30"
        }
        "children": [{
          "telemetryRsSensorGroupRel": {
            "attributes": {
              "sampleIntvl": "5000",
              "tDn": "sys/tm/sensor-10"
            }
          }
        },
        {
          "telemetryRsDestGroupRel": {
            "attributes": {
              "tDn": "sys/tm/dest-20"
            }
          }
        }
        ]
      }
    }
    ]
  }
}
```

NX-API を使用したテレメトリの構成例

宛先へのストリーミング パス

この例では、パス sys/cdp および sys/ipv4 を接続先 1.2.3.4 ポート 50001 に 5 秒ごとにストリーミングするサブスクリプションを作成します。

```
POST https://192.168.20.123/api/node/mo/sys/tm.json

Payload:
{
    "telemetryEntity": {
        "attributes": {
            "dn": "sys/tm"
        },
        "children": [{
            "telemetrySensorGroup": {
                "attributes": {
                    "id": "10",
                    "rn": "sensor-10"
                },  "children": [{
                    "telemetryRtSensorGroupRel": {
                        "attributes": {
                            "rn": "rtsensorGroupRel-[sys/tm/subs-30]",
                            "tCl": "telemetrySubscription",
                            "tDn": "sys/tm/subs-30"
                        }
                    }
                }, {
                    "telemetrySensorPath": {
                        "attributes": {
                            "path": "sys/cdp",
                            "rn": "path-[sys/cdp]",
                            "excludeFilter": "",
                            "filterCondition": "",
                            "secondaryGroup": "0",
                            "secondaryPath": "",
                            "depth": "0"
                        }
                    }
                }, {
                    "telemetrySensorPath": {
                        "attributes": {
                            "path": "sys/ipv4",
                            "rn": "path-[sys/ipv4]",
                            "excludeFilter": "",
                            "filterCondition": "",
                            "secondaryGroup": "0",
                            "secondaryPath": "",
                            "depth": "0"
                        }
                    }
                }]
            }
        }, {
            "telemetryDestGroup": {
                "attributes": {
                    "id": "20",
                    "rn": "dest-20"
                },
                "children": [{
```

```
                    "telemetryRtDestGroupRel": {
                        "attributes": {
                            "rn": "rtdestGroupRel-[sys/tm/subs-30]",
                            "tCl": "telemetrySubscription",
                            "tDn": "sys/tm/subs-30"
                        }
                    }
                }, {
                    "telemetryDest": {
                        "attributes": {
                            "addr": "1.2.3.4",
                            "enc": "GPB",
                            "port": "50001",
                            "proto": "gRPC",
                            "rn": "addr-[1.2.3.4]-port-50001"
                        }
                    }
                }]
            }
        }, {
            "telemetrySubscription": {
                "attributes": {
                    "id": "30",
                    "rn": "subs-30"
                },
                "children": [{
                    "telemetryRsDestGroupRel": {
                        "attributes": {
                            "rType": "mo",
                            "rn": "rsdestGroupRel-[sys/tm/dest-20]",
                            "tCl": "telemetryDestGroup",
                            "tDn": "sys/tm/dest-20",
                            "tType": "mo"
                        }
                    }
                }, {
                    "telemetryRsSensorGroupRel": {
                        "attributes": {
                            "rType": "mo",
                            "rn": "rssensorGroupRel-[sys/tm/sensor-10]",
                            "sampleIntvl": "5000",
                            "tCl": "telemetrySensorGroup",
                            "tDn": "sys/tm/sensor-10",
                            "tType": "mo"
                        }
                    }
                }]
            }
        }]
    }
}
```

BGP 通知のフィルタ条件

次のペイロードの例では、`telemetrySensorPath` MO の `filterCondition` 属性に従って BFP 機能が無効になっているときにトリガーされる通知を有効にします。データは `10.30.217.80` ポート `50055` にストリーミングされます。

```
POST  https://192.168.20.123/api/node/mo/sys/tm.json

Payload:
{
```

```
"telemetryEntity": {
  "children": [{
    "telemetrySensorGroup": {
      "attributes": {
        "id": "10"
      }
      "children": [{
        "telemetrySensorPath": {
          "attributes": {
            "excludeFilter": "",
            "filterCondition": "eq(fmBgp.operSt,\"disabled\")",
            "path": "sys/fm/bgp",
            "secondaryGroup": "0",
            "secondaryPath": "",
            "depth": "0"
          }
        }
      }
      ]
    }
  },
  {
    "telemetryDestGroup": {
      "attributes": {
        "id": "20"
      }
      "children": [{
        "telemetryDest": {
          "attributes": {
            "addr": "10.30.217.80",
            "port": "50055",
            "enc": "GPB",
            "proto": "gRPC"
          }
        }
      }
      ]
    }
  },
  {
    "telemetrySubscription": {
      "attributes": {
        "id": "30"
      }
      "children": [{
        "telemetryRsSensorGroupRel": {
          "attributes": {
            "sampleIntvl": "0",
            "tDn": "sys/tm/sensor-10"
          }
        }
      },
      {
        "telemetryRsDestGroupRel": {
          "attributes": {
            "tDn": "sys/tm/dest-20"
          }
        }
      }
      ]
    }
  }
  ]
```

```
  }
}
```

テレメトリ構成のための Postman コレクションの使用

Postman コレクションの例は、テレメトリ機能の構成を開始する簡単な方法であり、1 つのペイロードですべてのテレメトリ CLI に相当するものを実行できます。好みのテキスト エディターを使用して前述のリンクのファイルを変更し、ペイロードをニーズに合わせて更新してから、Postman でコレクションを開いてコレクションを実行します。

DME のテレメトリ モデル

テレメトリ アプリケーションは、次の構造を持つ DME でモデル化されます。

```
model
|----package [name:telemetry]
    |  @name:telemetry
    |----objects
         |----mo [name:Entity]
              |    @name:Entity
              |      @label:Telemetry System
              |--property
              |    @name:adminSt
              |      @type:AdminState
              |
              |----mo [name:SensorGroup]
              |    |    @name:SensorGroup
              |    |      @label:Sensor Group
              |    |--property
              |    |    @name:id [key]
              |    |      @type:string:Basic
              |    |    @name:dataSrc
              |    |      @type:DataSource
              |    |
              |    |----mo [name:SensorPath]
              |    |         |   @name:SensorPath
              |    |         |     @label:Sensor Path
              |    |         |--property
              |    |         |    @name:path [key]
              |    |         |      @type:string:Basic
              |    |         |    @name:filterCondition
              |    |         |      @type:string:Basic
              |    |         |    @name:excludeFilter
              |    |         |      @type:string:Basic
              |    |         |    @name:depth
              |    |         |      @type:RetrieveDepth
              |
              |----mo [name:DestGroup]
              |    |    @name:DestGroup
              |    |      @label:Destination Group
              |    |--property
              |    |    @name:id
              |    |      @type:string:Basic
              |    |
              |    |----mo [name:Dest]
              |    |         |   @name:Dest
              |    |         |     @label:Destination
              |    |         |--property
              |    |         |    @name:addr [key]
```

```
|          |      @type:address:Ip
|          |    @name:port [key]
|          |      @type:scalar:Uint16
|          |    @name:proto
|          |      @type:Protocol
|          |    @name:enc
|          |      @type:Encoding
|
|----mo [name:Subscription]
     |    @name:Subscription
     |      @label:Subscription
     |--property
     |    @name:id
     |      @type:scalar:Uint64
     |----reldef
     |    |  @name:SensorGroupRel
     |    |    @to:SensorGroup
     |    |    @cardinality:ntom
     |    |    @label:Link to sensorGroup entry
     |    |--property
     |         @name:sampleIntvl
     |           @type:scalar:Uint64
     |
     |----reldef
          |  @name:DestGroupRel
          |    @to:DestGroup
          |    @cardinality:ntom
          |    @label:Link to destGroup entry
```

テレメトリで使用可能な DN

テレメトリ機能で使用できる DN のリストについては、モデル駆動型テレメトリ （1 ページ）を参照してください。

その他の参考資料

関連資料

関連項目	マニュアル タイトル
VXLAN EVPN のテレメトリ展開の構成例。	*[VXLAN EVPN ソリューションのテレメトリ展開（Telemetry Deployment for VXLAN EVPN Solution）]*

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。