



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)
- [Telemetry のインストールとアップグレード](#) (3 ページ)
- [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](#) (20 ページ)
- [その他の参考資料](#) (33 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は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。

データは定期的を取得されるか（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、4MB 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。）

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 }
```

num は 1 ~ 4095 の数値です。

UDP テレメトリは、次のヘッダーと共に送信されます。

```
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 データを正常に処理します。

- 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、データの復号化に使用するデコーダー (JSON または GPB) を決定します。または、
- 1 つのデコーダー (JSON または GPB) が必要で、もう 1 つは必要ない場合は、ヘッダーを削除します。
- **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復号化するには、受信側で暗号化とトランスポート 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、『 <i>Cisco NX-OS Licensing Guide</i> 』を参照してください。

Telemetry のインストールとアップグレード

アプリケーションのインストール

テレメトリ アプリケーションは機能 RPM としてパッケージ化されており、NX-OS リリースに含まれています。RPM は、イメージブートアップの一部としてデフォルトでインストールされます。 `feature telemetry` コマンドを使用して、アプリケーションを起動します。RPM ファイルは `/rpms` ディレクトリにあり、次のような名前が付けられています。

次の例のように：

増分更新と修正のインストール

RPM をデバイスのブートフラッシュにコピーし、bash プロンプトから次のコマンドを使用します：

```
feature bash
run bash sudo su
```

そして、デバイスブートフラッシュに RPM のコピーをします。bash プロンプトから次のコマンドを使用します：

```
yum upgrade telemetry_new_version.rpm
```

アプリケーションがアップグレードされ、アプリケーションを再起動すると変更が表示されます。

以前のバージョンにダウングレードします

テレメトリ アプリケーションを以前のバージョンにダウングレードするには、bash プロンプトから次のコマンドを使用します。

```
yum downgrade telemetry
```

アクティブなバージョンの確認

現用系なバージョンを確認するには、スイッチの exec プロンプトから次のコマンドを実行します。

```
show install active
```



(注) [現用系のインストールを表示します (show install active)] コマンドは、アップグレードが実行された後に、インストールされている現用系な RPM のみを表示します。NX-OS にバンドルされているデフォルトの RPM は表示されません。

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、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 Cisco NX-OS リリースでサポートされています。
- 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、HTTP 経由の JSON エンコーディングがサポートされています。
- 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。

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
    use-chunking size 4096
  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
    use-chunking size 4096
  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なら、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。

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。

ノード識別子のサポート

NX-OS リリース 9.3.1 以降、**use-nodeid** コマンドを使用してテレメトリ受信者のカスタム ノード識別子文字列を構成できます。デフォルトではホスト名が使用されますが、ノード受信者のサポートにより、テレメトリ受信者データの `node_id_str` の識別子を設定または変更できます。

usenode-id コマンドを使用して、テレメトリ接続先プロファイルを介してノード ID を割り当てることができます。このコマンドはオプションです。

次の例は、ノード識別子の構成を表示しています。

```
switch-1(config)# telemetry
switch-1(config-telemetry)# destination-profile
switch-1(conf-tm-dest-profile)# use-nodeid test-srvr-10
switch-1(conf-tm-dest-profile)#
```

次の例は、ノード識別子が構成された後の受信側でのテレメトリ通知を表示しています。

```
Telemetry receiver:
=====
node_id_str: "test-srvr-10"
subscription_id_str: "1"
encoding_path: "sys/ch/psuslot-1/psu"
collection_id: 3896
msg_timestamp: 1559669946501
```

テレメトリの VRF サポート

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。

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。

始める前に

スイッチは、Cisco NX-OS リリース 9.2(1) 以降のリリースを実行している必要があります。

手順の概要

1. **configure terminal**
2. **feature telemetry**
3. **feature nxapi**
4. **nxapi use-vrf management**
5. **telemetry**
6. (任意) **certificate certificate_path host_URL**
7. **sensor-group sgrp_id**
8. **path sensor_path depth 0 [filter-condition filter]**
9. **destination-group dgrp_id**
10. (任意) **ip address ip_address port port protocol procedural-protocol encoding encoding-protocol**
11. **ip_version address ip_address port portnum**
12. **subscription sub_id**
13. **snsr-grp sgrp_id sample-interval interval**
14. **dst-grp dgrp_id**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例 :	グローバル構成モードを開始します。

	コマンドまたはアクション	目的
	switch# configure terminal switch(config)#	
ステップ 2	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 3	feature nxapi	nxapi を有効にします。
ステップ 4	nxapi use-vrf management	nxapi 通信に使用する VRF 管理を有効にします。
ステップ 5	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 6	(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。
ステップ 7	sensor-group sgrp_id 例： switch(config-telemetry)# sensor-group 100 switch(conf-tm-sensor)#	ID <i>sgrp_id</i> 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 8	path sensor_path depth 0 [filter-condition filter] 例： <ul style="list-style-type: none">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(12BD.operSt, "down") 以下の構文を使用して、状態ベースのフィルタリングを使用して、 operSt が上から下に变化したときにのみトリガーし、MO がいつ变化したかを通知しません。 switch(conf-tm-sensor)# path sys/bd/bd-[vlan-100] depth 0 filter-condition and(updated(12BD.operSt),eq(12BD.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リングの場合、フィルタは、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

	コマンドまたはアクション	目的
		<p>まり、<code>eq(l2Bd.operSt, "down")</code> の DN <code>sys/bd/bd-[vlan]</code> のフィルタ条件は、<code>operSt</code> が変更されたとき、および <code>operSt</code> が残っている間に DN のプロパティが変更されたとき (<code>vlan down</code> が <code>down</code> で動作している間に <code>no shutdown</code> コマンドが発行されるなど) にトリガーされま</p>
ステップ 9	<p>destination-group <i>dgrp_id</i></p> <p>例 :</p> <pre>switch(conf-tm-sensor)# destination-group 100 switch(conf-tm-dest)#</pre>	<p>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。</p> <p>現在、<i>dgrp_id</i> は、数字の ID 値のみをサポートしています。</p>
ステップ 10	<p>(任意) ip address <i>ip_address</i> port <i>port</i> protocol <i>procedural-protocol</i> encoding <i>encoding-protocol</i></p> <p>例 :</p> <pre>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</pre>	<p>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。</p> <p>(注) gRPC はデフォルトのトランスポートプロトコルです。</p> <p>GPB がデフォルトのエンコーディングです。</p>
ステップ 11	<p>ip_version <i>address</i> <i>ip_address</i> port <i>portnum</i></p> <p>例 :</p> <pre>switch(conf-tm-dest)# ip address 1.2.3.4 port 50003</pre>	<p>発信データの接続先プロファイルを作成します。</p> <p>接続先グループがサブスクリプションノード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。</p>
ステップ 12	<p>subscription <i>sub_id</i></p> <p>例 :</p> <pre>switch(conf-tm-dest)# subscription 100 switch(conf-tm-sub)#</pre>	<p>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。</p> <p>現在、<i>sub_id</i> は、数字の ID 値のみをサポートしています。</p> <p>(注) DN にサブスクライブするときは、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。</p>
ステップ 13	<p>snsr-grp <i>sgrp_id</i> sample-interval <i>interval</i></p> <p>例 :</p> <pre>switch(conf-tm-sub)# snsr-grp 100 sample-interval 15000</pre>	<p>ID <i>sgrp_id</i> 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。</p> <p>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 での変更時のみ送信されます。0 よ</p>

	コマンドまたはアクション	目的
		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づ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 14	dst-grp <i>dgrp_id</i> 例： switch(conf-tm-sub) # dst-grp 100	ID <i>dgrp_id</i> 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(config-tm-sub)# dst-grp 100
switch(config-tm-sub)# dst-grp 200

switch(config-tm-dest)# subscription 900
switch(config-tm-sub)# snsr-grp 200 sample-interval 7000
switch(config-tm-sub)# snsr-grp 300 sample-interval 0
switch(config-tm-sub)# dst-grp 100
switch(config-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(config-tm-dest)# ip address 1.2.3.4 port 50003
switch(config-tm-dest)# ip address 1.2.3.4 port 50004
switch(config-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

```

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

次に、テレメトリの構成に関する内部データベース関連の統計情報のコマンドを表示します。

```

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

次に、データ収集に関する簡単な統計情報のコマンドを表示します。

```
switch# show telemetry data collector brief
```

```

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

```

show telemetry data collector details

次に、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のコマンドを表示します。

```
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 コレクションに関するエラー統計を表示します。

```
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 コレクションに関する統計を表示します。

```
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

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

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
Encoding:           GPB
Transport:          gRPC
Status:             Disconnected

```

```
Last Connected:      Fri Sep 02 11:45:57.505 UTC
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
Encoding:             JSON
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。

```
switch# show telemetry transport 0 stats
```

```
Session Id:          0
IP Address:Port      192.168.20.123:50001
Encoding:             GPB
Transport:            GRPC
Status:               Connected
Last Connected:      Mon May 01 11:29:46.912 PST
Last Disconnected:   Never
Tx Error Count:       0
Last Tx Error:       None
```

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 Stats
  Connection Count    1
  Last Connected:     Mon May 01 11:29:46.912 PST
  Disconnect Count    0
  Last Disconnected:  Never
Transmission Stats
  Transmit Count:     1225
  Last TX time:       Tue May 02 11:40:03.531 PST
  Min Tx Time:        7 ms
  Max Tx Time:        1760 ms
  Avg Tx Time:        500 ms
```

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
```

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 [DME のテレメトリ モデル, on page 31](#). 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.

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 configured to run the NX-API from the CLI:

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

SUMMARY STEPS

1. Enable the telemetry feature.
2. Create the root level of the JSON payload to describe the telemetry configuration.
3. Create a sensor group to contain the defined sensor paths.
4. Define a telemetry destination group.
5. Define a telemetry destination profile.
6. Define one or more telemetry destinations, consisting of an IP address and port number to which telemetry data will be sent.
7. Create a telemetry subscription to configure the telemetry behavior.
8. Add the sensor group object as a child object to the **telemetrySubscription** element under the root element (**telemetryEntity**).
9. 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.
10. Define one or more sensor paths or nodes to be monitored for telemetry.
11. Add sensor paths as child objects to the sensor group object (**telemetrySensorGroup**).
12. Add destinations as child objects to the destination group object (**telemetryDestGroup**).
13. Add the destination group object as a child object to the root element (**telemetryEntity**).
14. Create a relation object as a child object of the telemetry sensor group to associate the sensor group to the subscription.
15. Create a relation object as a child object of the telemetry destination group to associate the destination group to the subscription.
16. Create a relation object as a child object of the subscription to associate the subscription to the telemetry destination group.
17. Send the resulting JSON structure as an HTTP/HTTPS POST payload to the NX-API endpoint for telemetry configuration.

DETAILED STEPS

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

	Command or Action	Purpose
	<pre> "attributes" : { "adminSt" : "enabled" } }] } </pre>	
ステップ 2	<p>Create the root level of the JSON payload to describe the telemetry configuration.</p> <p>Example:</p> <pre> { "telemetryEntity": { "attributes": { "dn": "sys/tm" }, } } </pre>	<p>The root element is telemetryEntity and the base path for this element is <code>sys/tm</code>. Configure the dn attribute as <code>sys/tm</code>.</p>
ステップ 3	<p>Create a sensor group to contain the defined sensor paths.</p> <p>Example:</p> <pre> "telemetrySensorGroup": { "attributes": { "id": "10", "rn": "sensor-10" }, "children": [{ }] } </pre>	<p>A telemetry sensor group is defined in an object of class telemetrySensorGroup. Configure the following attributes of the object:</p> <ul style="list-style-type: none"> • 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. <p>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.</p>
ステップ 4	<p>Define a telemetry destination group.</p> <p>Example:</p> <pre> { "telemetryDestGroup": { "attributes": { "id": "20" } } } </pre>	<p>A telemetry destination group is defined in telemetryEntity. Configure the id attribute.</p>
ステップ 5	<p>Define a telemetry destination profile.</p> <p>Example:</p> <pre> { "telemetryDestProfile": { "attributes": { "adminSt": "enabled" } } } </pre>	<p>A telemetry destination profile is defined in telemetryDestProfile.</p> <ul style="list-style-type: none"> • Configure the adminSt attribute as <code>enabled</code>. • Under telemetryDestOptSourceInterface, configure the name attribute with an interface name to stream

	Command or Action	Purpose
	<pre> }, "children": [{ "telemetryDestOptSourceInterface": { "attributes": { "name": "lo0" } }] } </pre>	<p>data from the configured interface to a destination with the source IP address.</p>
<p>ステップ 6</p>	<p>Define one or more telemetry destinations, consisting of an IP address and port number to which telemetry data will be sent.</p> <p>Example:</p> <pre> { "telemetryDest": { "attributes": { "addr": "1.2.3.4", "enc": "GPB", "port": "50001", "proto": "gRPC", "rn": "addr-[1.2.3.4]-port-50001" } } } </pre>	<p>A telemetry destination is defined in an object of class telemetryDest. Configure the following attributes of the object:</p> <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • Google protocol buffers (GPB) for gRPC. • JSON for C. • proto — The transport protocol type of the telemetry data to be sent. NX-OS supports: <ul style="list-style-type: none"> • gRPC • HTTP
<p>ステップ 7</p>	<p>Create a telemetry subscription to configure the telemetry behavior.</p> <p>Example:</p> <pre> "telemetrySubscription": { "attributes": { "id": "30", "rn": "subs-30" }, "children": [{ }] } </pre>	<p>A telemetry subscription is defined in an object of class telemetrySubscription. Configure the following attributes of the object:</p> <ul style="list-style-type: none"> • 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. <p>Children of the subscription object will include relation objects for sensor groups (telemetryRsSensorGroupRel) and destination groups (telemetryRsDestGroupRel).</p>

	Command or Action	Purpose
ステップ 8	<p>Add the sensor group object as a child object to the telemetrySubscription element under the root element (telemetryEntity).</p> <p>Example:</p> <pre> { "telemetrySubscription": { "attributes": { "id": "30" } "children": [{ "telemetryRsSensorGroupRel": { "attributes": { "sampleIntvl": "5000", "tDn": "sys/tm/sensor-10" } }] } } </pre>	
ステップ 9	<p>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.</p> <p>Example:</p> <pre> "telemetryRsSensorGroupRel": { "attributes": { "rType": "mo", "rn": "rssensorGroupRel-[sys/tm/sensor-10]", "sampleIntvl": "5000", "tCl": "telemetrySensorGroup", "tDn": "sys/tm/sensor-10", "tType": "mo" } } </pre>	<p>The relation object is of class telemetryRsSensorGroupRel and is a child object of telemetrySubscription. Configure the following attributes of the relation object:</p> <ul style="list-style-type: none"> • 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.

	Command or Action	Purpose
<p>ステップ 10</p>	<p>Define one or more sensor paths or nodes to be monitored for telemetry.</p> <p>Example: Single sensor path</p> <pre>{ "telemetrySensorPath": { "attributes": { "path": "sys/cdp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "", "path": "sys/fm/bgp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } }</pre> <p>Example: Multiple sensor paths</p> <pre>{ "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" } } }</pre> <p>Example: Single sensor path filtering for BGP disable events:</p> <pre>{ "telemetrySensorPath": {</pre>	<p>A sensor path is defined in an object of class telemetrySensorPath. Configure the following attributes of the object:</p> <ul style="list-style-type: none"> • 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 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
	<pre> "attributes": { "path": "sys/cdp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "eq(fmBgp.operSt.\"disabled\")", "path": "sys/fm/bgp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } } </pre>	
ステップ 11	Add sensor paths as child objects to the sensor group object (telemetrySensorGroup).	
ステップ 12	Add destinations as child objects to the destination group object (telemetryDestGroup).	
ステップ 13	Add the destination group object as a child object to the root element (telemetryEntity).	
ステップ 14	<p>Create a relation object as a child object of the telemetry sensor group to associate the sensor group to the subscription.</p> <p>Example:</p> <pre> "telemetryRtSensorGroupRel": { "attributes": { "rn": "rtsensorGroupRel-[sys/tm/subs-30]", "tCl": "telemetrySubscription", "tDn": "sys/tm/subs-30" } } </pre>	<p>The relation object is of class telemetryRtSensorGroupRel and is a child object of telemetrySensorGroup. Configure the following attributes of the relation object:</p> <ul style="list-style-type: none"> • 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 15	<p>Create a relation object as a child object of the telemetry destination group to associate the destination group to the subscription.</p> <p>Example:</p> <pre> "telemetryRtDestGroupRel": { "attributes": { "rn": "rtdestGroupRel-[sys/tm/subs-30]", "tCl": "telemetrySubscription", "tDn": "sys/tm/subs-30" } } </pre>	<p>The relation object is of class telemetryRtDestGroupRel and is a child object of telemetryDestGroup. Configure the following attributes of the relation object:</p> <ul style="list-style-type: none"> • 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 16	<p>Create a relation object as a child object of the subscription to associate the subscription to the telemetry destination group.</p> <p>Example:</p>	<p>The relation object is of class telemetryRsDestGroupRel and is a child object of telemetrySubscription. Configure the following attributes of the relation object:</p>

	Command or Action	Purpose
	<pre> "telemetryRsDestGroupRel": { "attributes": { "rType": "mo", "rn": "rsdestGroupRel-[sys/tm/dest-20]", "tCl": "telemetryDestGroup", "tDn": "sys/tm/dest-20", "tType": "mo" } } </pre>	<ul style="list-style-type: none"> • 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.
<p>ステップ 17</p>	<p>Send the resulting JSON structure as an HTTP/HTTPS POST payload to the NX-API endpoint for telemetry configuration.</p>	<p>The base path for the telemetry entity is <code>sys/tm</code> and the NX-API endpoint is:</p> <pre> {{URL}}/api/node/mo/sys/tm.json </pre>

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"
        }
      }
    ]
  },
  "telemetryDestGroup": {
    "attributes": {
      "id": "20"
    }
  }
}
        
```



```

        "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"
        }
    },

```



```

|--property
|   @name:adminSt
|   @type:AdminState
|
|----mo [name:SensorGroup]
|   |   @name:SensorGroup
|   |   @label:Sensor Group
|   |--property
|   |   @name:id [key]
|   |   @type:string:Basic
|   |
|   |----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

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。