テレメトリ

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。

テレメトリのコンポーネントと用語

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、またはパスのブランチにある YANG インフラから収集されます。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 プロトコルバッファ（GPB）と JSON 形式でエンコードします。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エンコーダに必要です。
- **データ トランスポート**：NX-OS は、JSON エンコードには HTTP を使用し、GPB エンコードには Google リモートプロシージャコール（gRPC）プロトコルを使用して、テレメトリデータを転送します。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。）
- **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。データストリームを正しく受信してデコードするには、受信側に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テレメトリの `.proto` ファイルは、Cisco の GitLab で入手できます：https://github.com/CiscoDevNet/nx-telemetry-proto

（注） テレメトリのコンテキストでは、gRPC は特定の独自仕様 .proto を指します。標準規格の gNMI/gNOI サービスをホストする「gRPC エージェント」と混同しないでください。

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**：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を復元します。
- **スーパーバイザフェールオーバー**：テレメトリがホットスタンバイではなくても、新しいアクティブなスーパーバイザが実行されているときに、テレメトリ構成とストリーミングサービスを復元します。

- **プロセスの再起動**：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、再起動時に、構成およびストリーミング サービスを復元します。

テレメトリのライセンス要件

表 *1*: 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限事項

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。

- サポートされるプラットフォームの詳細については、Nexus Switch Platform Matrix を参照してください。
- 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 リリースは、テレメトリをサポートします。
- 以下のサポートが実施されています。
  - DME データ収集
  - NX-API データ ソース
  - Google リモート プロシージャ コール（gRPC）トランスポートを介した 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
- 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。最小値未満のど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。
- テレメトリは、最大 5 つの遠隔管理受信者（接続先）をサポートします。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。
- テレメトリは、CPU 技術情報の最大 20% を消費する可能性があります。

- NX-OS リリース 10.4(1)F より前のリリースでは、DME 構成 MO の時間プロパティは現地時間で印刷されますが、タイム ゾーンのオフセットは常に 00.00 です。たとえば、時刻は 2024-05-21T14:01:03.012+00:00 のように表示されます。

  NX-OS リリース 10.5(1)F 以降、currentTime、modTs などの DME 構成 MO のタイムスタンプは、localtime-timezone_offset 形式で表示されます。たとえば、時刻は 2024-05-21T14:01:03.012-04:00 のように表示されます。

- Cisco NX-OS リリース 10.4(2)F 以降、テレメトリは Cisco Nexus 93400LD-H1 プラットフォームスイッチでサポートされます。

- Cisco NX-OS リリース 10.4(3)F 以降、テレメトリは 92348GC-X でサポートされます。

- Cisco NX-OS リリース 10.4(2)F 以降、テレメトリは Cisco Nexus N9KC9364C-H1 プラットフォームスイッチでサポートされます。

- Cisco NX-OS リリース 10.6(1)F 以降、テレメトリは Cisco Nexus N9336C-SE1 プラットフォーム スイッチでサポートされます。

- Cisco NX-OSリリース 10.6(1)F 以降、Cisco Live Protect 機能による NX- OSのセキュアリングでは、nxsecure の新しいセンサーパスを使用してテレメトリ機能を使用してエクスポートできるイベント ログがサポートされています。

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場合は、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。このシーケンスにより、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できます。

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 受信者が 20 のエラーを送信した場合、スイッチ クライアントは gRPC 受信者への接続を無効化します。gRPC 受信者を有効にするには、接続先グループの下の接続先 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する。
- 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 チャンキングのサポート

Cisco NX-OSは、gRPC チャンクをサポートします。ストリーミングを正常に行うには、gRPC が 12 MB を超えるデータ量を受信者に送信する必要がある場合、チャンクを有効にする必要があります。

gRPC ユーザーは、gRPC チャンクを行う必要があります。gRPC クライアント側は断片化を行い、gRPC サーバ側はリアセンブルを行います。テレメトリは引き続きメモリにバインドされており、メモリ サイズがテレメトリに許可されている制限である 12 MB を超えると、データが削除される可能性があります。チャンクをサポートするには、次で説明されているように、gRPC チャンク用に更新された、Cisco の GibLab で入手可能なテレメトリ `.proto` ファイルを使用します。

チャンク サイズは 64 ～ 4096 バイトです。

次に、NX-API CLI による構成例を表示します。

```
feature telemetry !
telemetry
destination-group 1
ip address 171.68.197.40 port 50051 protocol gRPC encoding GPB use-chunking size 4096
destination-group 2
ip address 10.155.0.15 port 50001 protocol gRPC encoding GPB use-chunking size 64
```

```
sensor-group 1 path sys/intf depth unbounded
sensor-group 2 path sys/intf depth unbounded
subscription 1
dst-grp 1
snsr-grp 1 sample-interval 10000
subscription 2
dst-grp 2 snsr-grp 2 sample-interval 15000
```

次に、NX-API REST による構成例を表示します。

```
{
"telemetryDestGrpOptChunking": {
"attributes": {
"chunkSize": "2048",
"dn": "sys/tm/dest-1/chunking"
}
```

Cisco MDS シリーズ スイッチなど、gRPC チャンクをサポートしていないシステムでは、次のエラー メッセージが表示されます。

```
switch# use-chunking size 200
ERROR: Operation failed: [chunking support not available]
```

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します。

   ```
   show
           <command> | json
   ```

   つまり `show <command> | json pretty` のようにします。

（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. show コマンドを調整して、フィルタまたはオプションを含めます。

   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。show vlan id 100、show vlan id 101 などです。パフォーマンスを改善するため、可能であれば、代わりに CLI 範囲オプションを使用してください。show vlan id 100-110,204 などです。

   サマリーまたはカウンタのみが必要な場合は、show コマンド出力全体をダンプすることは避け、データ収集で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

テレメトリVRFのサポートにより、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 ポートを介して出力され、SSHまたは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"
}
}
}
]
}
}
```

与えられた構成で、特定のIPアドレスとポートの組み合わせでデータのルーティングを行うために使用できるVRFは1つだけです。使用されるVRFは、最後に行われたvrf構成によって決定されます。

次に例を示します。

```
telemetry
  destination-group 1
    ip address 91.1.1.1 port 50007
    use-vrf default
  destination-group 2
    ip address 91.1.1.1 port 50007
    use-vrf test
  sensor-group 1
    data-source DME
    path sys/fm
  subscription 1
    dst-grp 1
    dst-grp 2
    snsr-grp 1 sample-interval 1000
```

上記の構成では、destination-group 1 と destination-group 2 の両方に同じ宛先 IP アドレスとポート（91.1.1.1 50007）が設定されていますが、異なる VRF が構成されています（default と test）。このシナリオでは、destination-group 2 が後から構成されていた場合、91.1.1.1 50007 に送信されるデータは、**test** VRF を使用してルーティングされます。destination-group に VRF が構成されていない場合は、暗黙的に**管理** VRF が使用されます。

証明書トラストポイント サポート

テレメトリは trustpoint キーワードをサポートします。

次にあるのは、コマンド シンタックスです。

```
switch(config-telemetry)# certificate ?
trustpoint    specify trustpoint label
WORD          .pem certificate filename (Max Size 256)
switch(config-telemetry)# certificate trustpoint
WORD          trustpoint label name (Max Size 256)
switch(config-telemetry)# certificate trustpoint trustpoint1 ?
WORD  Hostname associated with certificate (Max Size 256)
switch(config-telemetry)#certificate trustpoint trustpoint1 foo.test.google.fr
```

接続先ホスト名サポート

テレメトリは、destination-group コマンドで **host** キーワードをサポートしています。

次に、接続先ホスト名のサポートの例を示します。

```
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# ?
certificate Specify certificate
host Specify destination host
ip Set destination IPv4 address
ipv6 Set destination IPv6 address
...
switch(conf-tm-dest)# host ?
A.B.C.D|A:B::C:D|WORD  IPv4 or IPv6 address or DNS name of destination
switch(conf-tm-dest)#
switch(conf-tm-dest)# host abc port 11111 ?
protocol  Set transport protocol
switch(conf-tm-dest)# host abc port 11111 protocol ?
HTTP
UDP
gRPC
switch(conf-tm-dest)# host abc port 11111 protocol gRPC ?
encoding  Set encoding format
switch(conf-tm-dest)# host abc port 11111 protocol gRPC encoding ?
Form-data    Set encoding to Form-data only
GPB          Set encoding to GPB only
GPB-compact  Set encoding to Compact-GPB only
JSON         Set encoding to JSON
XML          Set encoding to XML
switch(conf-tm-dest)# host ip address 1.1.1.1 port 2222 protocol HTTP encoding JSON
<CR>
```

ノード識別子のサポート

テレメトリは、**use-nodeid** コマンドを使用して、テレメトリ受信者のカスタム Node ID 文字列をサポートします。デフォルトではホスト名が使用されますが、ノード ID のサポートにより、テレメトリ受信者データの `node_id_str` 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(config)# telemetry
switch(config-telemetry)# destination-profile
switch(conf-tm-dest-profile)# use-nodeid test-srvr-10
switch(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
```

host コマンドの下の **use-nodeid** サブコマンドを使用します。接続先レベルの **use-nodeid** 構成は、グローバルレベルの構成よりも優先されます。次の例はコマンドシンタックスを表示します。

```
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# host 172.19.216.78 port 18112 protocol http enc json
switch(conf-tm-dest-host)# use-nodeid ?
WORD  Node ID (Max Size 128)
switch(conf-tm-dest-host)# use-nodeid session_1:18112
```

テレメトリ 受信者の出力の例を表示します：

```
>> Message size 923
Telemetry msg received @ 23:41:38 UTC Msg Size: 11
node_id_str : session_1:18112 collection_id : 3118
data_source : DME
encoding_path : sys/ch/psuslot-1/psu collection_
start_time : 1598485314721
collection_end_time : 1598485314721
data :
```

YANG モデルのストリーミングのサポート

テレメトリは YANG（「Yet Another Next Generation」）データ モデリング言語をサポートします。テレメトリは、デバイス YANG と OpenConfig YANG の両方のデータ ストリーミングをサポートします。

プロキシのサポート

テレメトリは、ホストコマンドで **proxy** キーワードをサポートします。次にあるのは、コマンド シンタックスです。

```
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# host 172.19.216.78 port 18112 protocol http enc json
switch(conf-tm-dest-host)# proxy ?
A.B.C.D|A:B::C:D|WORD IPv4 or IPv6 address or DNS name of proxy server
<1-65535> Proxy port number, Default value is 8080 username Set proxy authentication
username
password Set proxy authentication password
```

gRPC 非同期モード

gRPC 非同期モードは、**host** コマンドでのみ使用できます。通常のストリーム状態では、このモードにおいて、受信者は **WriteDone()** 呼び出しを終了したり、受信したりしなくても、**mdtDialout** 呼び出しでデータをストリーミングできます。

次にあるのは、コマンド シンタックスです。

```
nxosv-1(config-telemetry)# destination-group 1
nxosv-1(conf-tm-dest)# host 172.22.244.130 port 50007 ?
nxosv-1(conf-tm-dest-host)# grpc-async ?
```

センサ グループの単一テレメトリ収集

Cisco NX-OS 10.5(2)F 以降、ユーザーは **telemetry** CLI で新しい CLI オプション **merge-subscriptions** を使用して、センサー グループが複数のサブスクリプションに含まれており、接続先グループで解析ファイルが構成されていない場合に、センサー グループの単一のテレメトリ コレクションを作成できます。

- この設定はイベント サブスクリプションには使用できません。
- 接続先グループとマージサブスクリプションのフィルタファイル設定は、相互に互換性がありません。NX-OS ではブロックされません。その場合、収集はすべてのセンサーグループに対して個別に行われます。これは以前のリリースと同じです。
- サンプル間隔が大きいサブスクリプションのデータ送信のサンプル間隔は、min_sample_interval*floor(cur_sample_interval /min_sample_interval) の式を使用して決定されます。
  - min_sample_interval は、すべてのサブスクリプションのセンサー グループの最小サンプル間隔です。
  - cur_sample 間隔は、そのセンサー グループの特定のサブスクリプションのサンプル間隔です。
- このオプションは以前のリリースでは使用できません。互換性チェックでの失敗を回避するには、ダウングレードする前にこの設定を削除してください。

CLI を使用したテレメトリの構成

CLI による構成

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。

手順の概要

1. **configure terminal**
2. **feature telemetry**

3. **feature nxapi**
4. **nxapi use-vrf management**
5. **telemetry**
6. **[no] merge-subscriptions**
7. （任意） **certificate** *certificate_path host_URL*
8. **sensor-group** *sgrp_id*
9. **path** *sensor_path* **depth unbounded [filter-condition** *filter]* **[alias** *path_alias]*
10. （任意） **data-source native**
11. （任意） **path event-nxsecure**
12. **destination-group** *dgrp_id*
13. （任意） **ip address** *ip_address* port *port* **protocol** *procedural-protocol* **encoding** *encoding-protocol*
14. （任意） **ipv6 address** *ipv6_address port* **port protocol** *procedural-protocol* **encoding** *encoding-protocol*
15. （任意） **source-interface** *interface*
16. *ip_version* **address ip_address port** *portnum*
17. （任意） **use-chunking size** *chunking_size*
18. **subscription sub_id**
19. **snsr-grp** *sgrp_id* **sample-interval** *interval*
20. **dst-grp** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	**configure terminal** 例： `switch# configure terminal`	グローバル構成モードを開始します。
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**	NX-API を有効にします。

ステップ 4	**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リングできるため、10.2(3)F より前のリリースでは次の警告が表示されます。
		警告
		警告: 構成された管理 ACL は、HTTP サービスに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。

	コマンドまたはアクション	目的
		（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 警告：非管理 VRF で構成された ACL は、その VRF の HTTP サービスには有効ではありません。

ステップ 5	**telemetry**
	例：
	`switch# 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	**[no] merge-subscriptions** 例： `switch# merge-subscriptions`	複数のサブスクリプションに含まれるすべてのセンサー グループに対して 1 つのコレクションを作成します。
ステップ 7	（任意） **certificate** *certificate_path host_URL* 例： `switch# 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	**sensor-group** *sgrp_id*
	例：
	`switch# sensor-group 100`
	`switch(config-telemetry)#`
		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。
		英数字の識別子が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 9	**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 または YANG 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")`
	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`
		ここでの無制限とは、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 ソース DME パスにのみ適用されます。
		
		センサー グループにセンサー パスを追加します。
		• Cisco NX-OS 9.3(5) リリース以降では、alias キーワードが導入されています。

	コマンドまたはアクション	目的

	`and(updated(l2BD.operSt),eq(l2BD.operSt,"down"))`
	UTR 側のパスを区別するには、次の構文を使用します。
	`switch(conf-tm-sensor)# path sys/ch/ftslot-1/ft alias ft_1`
	• 次のコマンドは、DME ではなく、NX-API または YANG に適用されます：
	`switch(conf-tm-sensor)# path "show interface" depth 0`
	• 次のコマンドは、デバイス YANG に適用されます。
	`switch(conf-tm-sensor)# path Cisco-NX-OS-device:System/bgp-items/inst-items`
	• 次のコマンドは、OpenConfig YANG に適用されます。
	`switch(conf-tm-sensor)# path openconfig-bgp:bgp`
	`switch(conf-tm-sensor)# path Cisco-NX-OS-device:System/bgp-items/inst-items alias bgp_alias`
	• 次のコマンドは、NX-API に適用されます：
	`switch(conf-tm-sensor)# path "show interface" depth 0 alias sh_int_alias`
	• 次のコマンドは、OpenConfig に適用されます。
	`switch(conf-tm-sensor)# path openconfig-bgp:bgp alias oc_bgp_alias`
		• [深さ（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、および無制限の深さ設定がサポートされています。
		（注）
		深さ 0 がデフォルトの深さです。NX-API ベースのセンサーパスは、深さ 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に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] フィルタ条件は、operSt が変更されたとき、および operSt が down のままである間に 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。
		• YANG モデルの場合、センサー パスの形式は module_name : YANG_path です。module_name は YANG モデル ファイルの名前です。次に例を示します。
		• デバイスYANG の場合：Cisco-NX-OS-device:System/bgp-items/inst-items
		• OpenConfig YANG の場合：openconfig-bgp:bgp
		（注）
		depth、filter-condition、および and query-condition パラメータは、現在 YANG ではサポートされていません。
		openconfig YANG モデルの場合は、https://github.com/YangModels/yang/tree/master/vendor/cisco/nx に移動して、最新リリースの適切なフォルダに移動します。

	コマンドまたはアクション	目的
		特定のモデルをインストールする代わりに、すべての OpenConfig モデルを含む openconfig-all RPM をインストールできます。 次に例を示します。 install add mtx-openconfig-bgp-1.0.0.0.0-7.0.3.IHD8.1.lib32_n9000.rpm activate
ステップ 10	（任意） **data-source native** 例： `switch(conf-tm-sensor)# data-source native`	特定のモデルやデータベースを必要とせずに、ネイティブ アプリケーションがストリーム データを使用できるように、データ送信元をネイティブに設定します。

ステップ 11	（任意） **path event-nxsecure**
	例：
	`switch(conf-tm-sensor)# path event-nxsecure`
	`switch(conf-tm-sensor)# path event-history`
	`switch(conf-tm-sensor)# path event-monitor`
		テレメトリ パス センサー タイプにより、ログ ファイルを外部の受信者にエクスポートできます。このコマンドは、テレメトリ パス センサー タイプを、次の 3 つのイベント ベースのパスのいずれかとして設定します。
		• `event-nxsecure` ：nxsecure ログ ファイルをリモート https サーバに転送できるようにします。ファイルは https を介して転送されます。
		• `event-history`：履歴イベント ログを維持および設定します
		• `event-monitor`：リアルタイムのイベント データ ストリームを統合します。

ステップ 12	**destination-group** *dgrp_id* 例： `switch(conf-tm-sensor)# destination-group 100`	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。 英数字の識別子がサポートされています。

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`
		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	（任意） **ipv6 address** *ipv6_address port* **port protocol** *procedural-protocol* **encoding** *encoding-protocol* 例：	エンコードされたテレメトリ データを受信する IPv6 IP アドレスとポートを指定します。 （注）

	コマンドまたはアクション	目的

	`switch(conf-tm-sensor)# ipv6 address 10:10::1 port 8000 protocol gRPC encoding GPB`
	`switch(conf-tm-sensor)# ipv6 address 10:10::1 port 8001 protocol HTTP encoding JSON`
	`switch(conf-tm-sensor)# ipv6 address 10:10::1 port 8002 protocol UDP encoding JSON`
		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 15	（任意） **source-interface** *interface* **例**： `switch(conf-tm-sensor)# source-interface vlan1500`	接続先IPアドレスのテレメトリ送信元インターフェイスを指定します。 • 接続先 IP：ポートごとに、1 つの送信元インターフェイスのみがサポートされます。このような接続先 IP：ポートに対しては、最後に構成された送信元インターフェイスが使用されます。 • 送信元インターフェイス構成には、接続先グループでuse-vrfCLIを使用して、そのインターフェイスごとの vrf を指定する必要があります。use-vrf が指定されていない場合は、管理 vrf が使用されます。

ステップ 16	*ip_version* **address ip_address port** *portnum*
	例：
	IPv4 の場合：
	`switch(conf-tm-dest)# ip address 1.2.3.4 port 50003`
	IPv6 の場合：
	`switch(conf-tm-dest)# ipv6 address 10:10::1 port 8000`
		発信データの宛先プロファイルを作成します。ip_version は、ip（IPv4 の場合）または ipv6（IPv6 の場合）です。
		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 17	（任意） **use-chunking size** *chunking_size* **例**： `switch(conf-tm-dest)# use-chunking size 64`	gRPC チャンクを有効にして、チャンク サイズを 64～4096 バイトに設定します。詳細については、「gRPCチャンクのサポート」セクションを参照してください。

ステップ 18	**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。
		（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。

	コマンドまたはアクション	目的
ステップ 19	**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 20	**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。

YANG パスの頻度の設定

YANG パスの頻度は、合計ストリーミング時間よりも長くする必要があります。合計ストリーミング時間と頻度が正しく構成されていない場合、テレメトリ データの収集にストリーミング間隔よりも長くかかることがあります。この状況では、次のことがわかります。

- テレメトリ データが受信側へのストリーミングよりも速く蓄積されるため、徐々に満たされるキュー。
- 現在の間隔からではない古いテレメトリ データ。

合計ストリーミング時間よりも大きい値に頻度を構成します。

手順の概要

1. **show telemetry control database sensor-groups**
2. **sensor group** *number*
3. **subscription** *number*
4. **snsr-grp** *number* **sample-interval** *milliseconds*
5. **show system resources**

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**show telemetry control database sensor-groups**
	例：
	`switch# show telemetry control database sensor-groups`
	`Sensor Group Database size = 2`
	`----------`
	`Row ID Sensor Group ID Sensor Group type`
	`Sampling interval(ms) Linked subscriptions SubID`
	`----------`
	`1 2 Timer /YANG`
	`5000 /Running 1 1`
	
	`Collection Time in ms (Cur/Min/Max):`
	`2444/2294/2460`
	`Encoding Time in ms (Cur/Min/Max): 56/55/57`
	`Transport Time in ms (Cur/Min/Max): 0/0/1`
	`Streaming Time in ms (Cur/Min/Max):`
	`2515/2356/28403`
	
	`Collection Statistics:`
	`collection_id_dropped = 0`
	`last_collection_id_dropped = 0`
	`drop_count = 0`
	
	`2 1 Timer /YANG`
	`5000 /Running 1 1`
	
	`Collection Time in ms (Cur/Min/Max): 144/142/1471`
	`Encoding Time in ms (Cur/Min/Max): 0/0/1`
	`Transport Time in ms (Cur/Min/Max): 0/0/0`
	`Streaming Time in ms (Cur/Min/Max): 149/147/23548`
	`Collection Statistics:`
	`collection_id_dropped = 0`
	`last_collection_id_dropped = 0`
	`drop_count = 0`
	
	`switch#`
	`telemetry`
	`destination-group 1`
	`ip address 192.0.2.1 port 9000 protocol HTTP`
	`encoding JSON`
	`sensor-group 1`
	`data-source YANG`
	`path /Cisco-NX-OS-device:System/procsys-items`
	`depth unbounded`
	`sensor-group 2`
	`data-source YANG`
	`path`
	`/Cisco-NX-OS-device:System/intf-items/phys-items`
	`depth unbounded`
	`subscription 1`
	`dst-grp 1`
		合計ストリーミング時間を計算します。
		合計ストリーミング時間は、各センサーグループの個々の現在のストリーミング時間の合計です。個々のストリーミング時間は、ミリ秒単位のストリーミング時間（Cur）に表示されます。この例では、合計ストリーミング時間は2.664秒（2515ミリ秒+149ミリ秒）です。
		構成された頻度をセンサーグループの合計ストリーミング時間と比較します。
		頻度はサンプル間隔で表示されます。この例では、合計ストリーミング時間（2.664秒）がケイデンス（デフォルトの5.000秒）よりも短いため、頻度は正しく構成されています。

	コマンドまたはアクション	目的
	snsr-grp 1 sample-interval 5000 snsr-grp 2 sample-interval 5000	
ステップ 2	**sensor group** *number* 例： `switch(config-telemetry)# sensor group1`	合計ストリーミング時間がその頻度以上の場合、間隔を設定したいセンサーグループを入力します。
ステップ 3	**subscription** *number* 例： `switch(conf-tm-sensor)# subscription 100`	センサーグループのサブスクリプションを編集します。
ステップ 4	**snsr-grp** *number* **sample-interval** *milliseconds* 例： `switch(conf-tm-sub)# snsr-grp number sample-interval 5000`	適切なセンサーグループについて、サンプル間隔を合計ストリーミング時間よりも大きい値に設定します。 この例では、サンプル間隔は 5.000 秒に設定されています。これは、2.664 秒の合計ストリーミング時間よりも長いため、有効です。

ステップ 5	**show system resources**
	例：
	`switch# show system resources`
	`Load average: 1 minute: 0.38 5 minutes: 0.43 15 minutes: 0.43`
	`Processes: 555 total, 3 running`
	`CPU states : 24.17% user, 4.32% kernel, 71.50% idle`
	`CPU0 states: 0.00% user, 2.12% kernel, 97.87% idle`
	`CPU1 states: 86.00% user, 11.00% kernel, 3.00% idle`
	`CPU2 states: 8.08% user, 3.03% kernel, 88.88% idle`
	`CPU3 states: 0.00% user, 1.02% kernel, 98.97% idle`
	`Memory usage: 16400084K total, 5861652K used, 10538432K free`
	`Current memory status: OK`
		CPUの使用状況を確認してください。この例に示すように、CPUユーザー状態が高い使用率を示している場合、頻度とストリーミング値が正しく構成されていません。この手順を繰り返して、頻度を正しく設定します。

CLI を使用したテレメトリの構成例

単一のテレメトリ DME ストリームの構成

ケイデンス 10 秒で GPB エンコーディングを使用します。

```
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)# source-interface vlan1500
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
```

実行構成の表示

```
switch# show running-config telemetry
telemetry
destination-group 1
ip address 171.70.59.62 port 50051 protocol gRPC encoding GPB
source-interface Vlan1500
sensor-group sg1
data-source DME
subscription 1
dst-grp 1
snsr-grp sg1 sample-interval 10000
```

sys/bgp ルート MO の登録

5 秒ごとに接続先 IP 1.2.3.4、ポート50003 に送信します。

```
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
```

sys/intf MO の登録

- 5 秒ごとに接続先 IP 1.2.3.4、ポート50003 に送信します。
- 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
```

sys/cdp MOの登録

15 秒ごとに接続先 IP 1.2.3.4、ポート50004 へ

```
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
```

show コマンドのケイデンスベースの収集の登録

750 秒ごと。

```
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
```

sys/fm へのイベントベースサブスクリプションの登録

sys/fm MO に変更がある場合にのみストリーミング

```
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
```

複数のセンサーおよび接続先グループへの登録

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
```

UDP トランスポートへの登録

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 }
Example for an IPv6 destination:
destination-group 100 ipv6 address 10:10::1 port 8000 protocol gRPC 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。

テレメトリ構成の確認

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
```

Telemetry Merge サブスクリプションの構成例

次に、merge-subscription を構成する例を示します。

```
Telemetry
  merge-subscriptions
  destination-group 1
    ip address 192.186.1.2 port 1 protocol HTTP encoding JSON
  Destination-group 2
    ip address 192.168.1.3 port 2 protocol gRPC encoding GPB
  sensor-group 1
    path sys/fm
  subscription 1
    dst-grp 1
    snsr-grp 1 sample-interval 10000
```

```
  subscription 2
    dst-grp 2
    snsr-grp 1 sample-interval 25000
```

センサー グループ 1 のデータは 10 秒ごとに収集されます。収集されたデータは、10 秒ごとにサブスクリプション 1 の接続先に送信され、20 秒ごとにサブスクリプション 2 の接続先に送信されます。

サブスクリプション 2 の snsr-grp 1 の例では、min_sample 間隔は 10 秒、cur_sample_interval は 25 秒です。したがって、そのデータは 20 秒ごとに送信されます。

```
Sample_interval = 10*floor(25/10)
            = 10*2
                        = 20s
```

構成の確認

次のコマンドを使用して、IOA 構成を確認します。

```
show telemetry control database sensor-groups
Sensor Group Database size = 1
Row ID     Sensor Group ID  Sensor Group type  Sampling interval(ms)  Linked subscriptions
  SubID
1          1                Timer   /DME            10000/Running      2
      1
Collection Time in ms (Cur/Min/Max): 1/1/2
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 10003/10003/10003
Streaming Time in ms (Cur/Min/Max): 10004/10004/10006
Collection Statistics:
collection_id_dropped      = 0
last_collection_id_dropped = 0
drop_count                 = 0
Configuration method: CONFIG_DME-ADMIN
2          1                Timer   /DME            20000*/Running      2
       2
Collection Time in ms (Cur/Min/Max): 1/1/1
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 4004/4004/4004
Streaming Time in ms (Cur/Min/Max): 4005/4005/4005
Collection Statistics:
collection_id_dropped      = 0
last_collection_id_dropped = 0
drop_count                 = 0
Configuration method: CONFIG_DME-ADMIN
*Calculated sample interval for merge-subscriptions
```

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 yang direct-path cisco-nxos-device

このコマンドは、他のパスよりもパフォーマンスが向上するように直接エンコードされた YANG パスを表示します。

```
switch# show telemetry yang direct-path cisco-nxos-device
) Cisco-NX-OS-device:System/lldp-items
```

```
2) Cisco-NX-OS-device:System/acl-items
3) Cisco-NX-OS-device:System/mac-items
4) Cisco-NX-OS-device:System/intf-items
5) Cisco-NX-OS-device:System/procsys-items/sysload-items
6) Cisco-NX-OS-device:System/ospf-items
7) Cisco-NX-OS-device:System/procsys-items
8) Cisco-NX-OS-device:System/ipqos-items/queuing-items/policy-items/out-items
9) Cisco-NX-OS-device:System/mac-items/static-items
10) Cisco-NX-OS-device:System/ch-items
11) Cisco-NX-OS-device:System/cdp-items
12) Cisco-NX-OS-device:System/bd-items
13) Cisco-NX-OS-device:System/eps-items
14) Cisco-NX-OS-device:System/ipv6-items
```

show telemetry control database

このコマンドは、他のパスよりもパフォーマンスが向上するように直接エンコードされたYANGパスを表示します。

```
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 database sensor-paths

このコマンドは、テレメトリ設定のセンサーパスの詳細を表示します。これには、エンコーディング、収集、トランスポート、およびストリーミングのカウンタが含まれます。

```
switch(conf-tm-sub)# show telemetry control database sensor-paths
Sensor Path Database size = 4
------------------------------------------------------------------------------------------------------------

 Row ID Subscribed Linked Groups Sec Groups Retrieve level Path(GroupId) : Query :
Filter
------------------------------------------------------------------------------------------------------------
1 No 1 0 Full sys/cdp(1) : NA : NA
GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 65785/65785/65785
Collection Time in ms (Cur/Min/Max): 10/10/55
Encoding Time in ms (Cur/Min/Max): 8/8/9
Transport Time in ms (Cur/Min/Max): 0/0/0
Streaming Time in ms (Cur/Min/Max): 18/18/65
2 No 1 0 Self show module(2) : NA : NA
GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 1107/1106/1107
Collection Time in ms (Cur/Min/Max): 603/603/802
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 0/0/1
Streaming Time in ms (Cur/Min/Max): 605/605/803
3 No 1 0 Full
GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 0/0/44
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 0/0/0
Streaming Time in ms (Cur/Min/Max): 1/1/44 sys/bgp(1) : NA : NA
4 No 1 0 Self
GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 2442/2441/2442
Collection Time in ms (Cur/Min/Max): 1703/1703/1903
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 0/0/0
Streaming Time in ms (Cur/Min/Max): 1703/1703/1904
switch(conf-tm-sub)#
 show version(2) : NA : NA
```

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単な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。

- **[成功（Successful）]**：完了した収集の数を反映します。成功数には、成功した収集の数のみが含まれます。試行回数は含まれません。収集が成功しても、必ずしもデータが返されるわけではありません。無効なパス、OC モデルがインストールされていない、データ収集のエラーなどの場合、失敗とされます。データがないことは、失敗とは見なされません。
- **[ペイロード（Payload）]**：収集ごとのペイロードの数を反映します。
- **[失敗（Failed）]**：失敗した収集またはペイロードの数を反映します。
- **[スキップ（Skipped）]**：スキップされた収集の数を反映します。収集は、アグレッシブなサンプル間隔やシステムリソースの不足が原因でスキップされる場合があります。
- **[ドロップ済み（Dropped）]**：ドロップされたペイロードの数を反映します。ペイロードは、システムリソースまたはトランスポートの状態が原因でドロップされる可能性があります。
- **[センサーパス（Sensor Path）]**：センサーパスを表示します。

```
switch# show telemetry data collector details
------------------------------------------------------------------------------------------------------
Row ID         Successful     Payloads   Failed      Skipped        Dropped        Sensor
 Path(GroupId)
------------------------------------------------------------------------------------------------------
1              3              6120       0           0              0
Cisco-NX-OS-device:System/intf-items(1)
```

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 コレクション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



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 *2 : show telemetry transport* の構文の説明

構文	説明（Description）
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
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
errors	（オプション）すべてのテレメトリ エラー
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
port	（オプション）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
transport_last_connected	（オプション）最後に接続されたトランス
transport_last_disconnected	（オプション）この接続先構成が最後に削
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 tx
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

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）この接続先構成が最後に削除さ
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
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）最後のトランスポート戻りコー
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
ts_timer_retry_bytes	（オプション）タイマー再試行バッファ サイ
ts_event_retry_size	（オプション）メッセージのイベント再試行回
ts_timer_retry_size	（オプション）タイマー再試行メッセージ回数
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

timer_retry_bytes	（オプション）タイマー再試行バッファ サ
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
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

Tx Error Count:      3
Last Tx Error:       Wed Apr 19 15:56:51.617 PDT
The following example shows output from an IPv6 entry.
switch# show telemetry transport 0
Session Id: 0
IP Address:Port [10:10::1]:8000
Transport: GRPC
Status: Idle
Last Connected: Never
Last Disconnected: Never
Tx Error Count: 0
Last Tx Error: None
Event Retry Queue Bytes: 0
Event Retry Queue Size: 0
Timer Retry Queue Bytes: 0
Timer Retry Queue Size: 0
Sent Retry Messages: 0
Dropped Retry Messages: 0
```

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
```

show telemetry control databases sensor-paths

これらの構成手順が、**show telemetry control databases sensor-paths** コマンドの出力に表示されます。

```
below. feature telemetry
telemetry
destination-group 1
ip address 172.25.238.13 port 50600 protocol gRPC encoding GPB
sensor-group 1
path sys/cdp depth unbounded path sys/intf depth unbounded path sys/mac depth 0
subscription 1
dst-grp 1 snsr-grp 1 sample-interval 1000 Command output. switch# show telemetry control
 databases sensor-paths
Sensor Path Database size = 3
------------------------------------------------------------------------------
--------------------
Row ID Subscribed Linked Groups Sec Groups Retrieve level Path(GroupId) : Query : Filter
------------------------------------------------------------------------------
--------------------
1 No 1 0 Full sys/cdp(1) : NA
: NA
GPB Encoded Data size in bytes (Cur/Min/Max): 30489/30489/30489
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 6/5/54
Encoding Time in ms (Cur/Min/Max): 5/5/6
Transport Time in ms (Cur/Min/Max): 1027/55/1045
Streaming Time in ms (Cur/Min/Max): 48402/5/48402
2 No 1 0 Full sys/intf(1) : N
A : NA
GPB Encoded Data size in bytes (Cur/Min/Max): 539466/539466/539466
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 66/64/114
Encoding Time in ms (Cur/Min/Max): 91/90/92
Transport Time in ms (Cur/Min/Max): 4065/4014/5334
Streaming Time in ms (Cur/Min/Max): 48365/64/48365
```

```
 3 No 1 0 Self sys/mac(1) : NA
: NA
GPB Encoded Data size in bytes (Cur/Min/Max): 247/247/247
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 1/1/47
Encoding Time in ms (Cur/Min/Max): 1/1/1
Transport Time in ms (Cur/Min/Max): 4/1/6
Streaming Time in ms (Cur/Min/Max): 47369/1/47369
```

show telemetry transport sessions

次のコマンドは、すべてのトランスポートセッションをループし、1つのコマンドで情報を出力します。

```
switch# show telemetry transport sessions
switch# show telemetry transport stats
switch# show telemetry transport errors
switch# show telemetry transport all
The following is an example for telemetry transport session:
switch# show telemetry transport sessions
Session Id:           0
IP Address:Port       172.27.254.13:50004
Transport:            GRPC
Status:               Transmit Error
SSL Certificate:      trustpoint1
Last Connected:       Never
Last Disconnected:    Never
Tx Error Count:       2
Last Tx Error:        Wed Aug 19 23:32:21.749 UTC
…
Session Id:           4
IP Address:Port       172.27.2
```

テレメトリ エフェメラル イベント

エフェメラル イベントをサポートするために、新しいセンサー パス クエリ条件が追加されました。アカウンティング ログの外部イベント ストリーミングを有効にするには、次のクエリ条件を使用します。

```
sensor-group 1 path sys/accounting/log query-condition
query-target=subtree&complete-mo=yes&notify-interval=1

The following are the other sensor paths that support ephemeral event:
sys/pim/inst/routedb-route, sys/pim/pimifdb-adj, sys/pim/pimifdb-prop
sys/igmp/igmpifdb-prop, sys/igmp/inst/routedb, sys/igmpsnoop/inst/dom/db-exptrack,
sys/igmpsnoop/inst/dom/db-group, sys/igmpsnoop/inst/dom/db-mrouter
sys/igmpsnoop/inst/dom/db-querier, sys/igmpsnoop/inst/dom/db-snoop
```

NX-API を使用したテレメトリの構成

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
          |    |
          |    |----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
```

NX- API を使用した構成

スイッチ DME のオブジェクト モデルでは、「DME のテレメトリ モデル」のセクションで説明されているように、テレメトリ機能の構成がオブジェクトの階層構造で定義されています。構成する主なオブジェクトは次のとおりです。

- **fmEntity**：NX-API およびテレメトリ機能の状態が含まれています。
- **fmNxapi**：NX-API の状態が含まれています。
- **fmTelemetry**：テレメトリ機能の状態が含まれています。
- **telemetryEntity**：テレメトリ機能の構成が含まれています。
- **telemetrySensorGroup**：テレメトリのために監視される 1 つ以上のセンサーパスまたはノードの定義が含まれています。テレメトリ エンティティには、1 つ以上のセンサー グループを含めることができます。
- **telemetryRtSensorGroupRel**：センサーグループをテレメトリサブスクリプションに関連付けます。
- **telemetrySensorPath**：モニター対象のパス。センサー グループには、このタイプのオブジェクトを複数含めることができます。
- **telemetryDestGroup**：テレメトリデータを受信する 1 つ以上の接続先の定義が含まれています。テレメトリ エンティティには、1 つ以上の接続先グループを含めることができます。
- **telemetryRtDestGroupRel**：接続先グループをテレメトリサブスクリプションに関連付けます。
- **telemetryDest** ：接続先アドレス。接続先グループには、このタイプのオブジェクトを複数含めることができます。
- **telemetrySubscription**：1 つ以上のセンサーグループからのテレメトリデータを 1 つ以上の接続先グループに送信する方法とタイミングを指定します。
- **telemetryRsDestGroupRel**：テレメトリサブスクリプションを接続先グループに関連付けます。
- **telemetryRsSensorGroupRel** ：テレメトリサブスクリプションをセンサーグループに関連付けます。
- **telemetryCertificate**： テレメトリサブスクリプションを証明書とホスト名に関連付けます。

NX-API を使用してテレメトリ機能を設定するには、テレメトリ オブジェクト構造の JSON 表現を構築し、HTTP または HTTPS POST 操作で DME にプッシュする必要があります。

（注） NX-API の使用に関する詳細な手順は、『*Cisco Nexus 3000 and 9000 Series NX-API REST SDK User Guide and API Reference*』を参照してください。

始める前に

CLI から NX-API を実行するようにスイッチを構成する必要があります。

```
switch(config)# feature nxapi

nxapi use-vrf vrf_name nxapi http port port_number
```

手順の概要

1. テレメトリ機能を有効にします。
2. テレメトリ構成を記述するために、JSON ペイロードのルート レベルを作成します。
3. 定義されたセンサー パスを含むセンサーグループを作成します。
4. （任意） SSL/TLS 証明書とホストを追加します。
5. テレメトリの接続先グループを定義します。
6. テレメトリの接続先プロファイルを定義します。
7. テレメトリ データの送信先となる IP アドレスとポート番号で構成される、1 つ以上のテレメトリの接続先を定義します。
8. gRPC チャンクを有効にして、チャンク サイズを 64 ～ 4096 バイトに設定します。
9. テレメトリサブスクリプションを作成して、テレメトリの動作を構成します。
10. ルート要素の下の telemetrySubscription 要素に子オブジェクトとしてセンサー グループ オブジェクトを追加します（telemetryEntity）。
11. サブスクリプションの子オブジェクトとして関係オブジェクトを作成して、サブスクリプションをテレメトリ センサー グループに関連付け、データ サンプリング動作を指定します。
12. テレメトリをモニタリングする 1 つ以上のセンサー パスまたはノードを定義します。
13. センサーパスを子オブジェクトとしてセンサーグループオブジェクト（telemetrySensorGroup）に追加します。
14. 接続先を子オブジェクトとして接続先グループオブジェクト（telemetryDestGroup）に追加します。
15. 接続先グループオブジェクトを子オブジェクトとしてルート要素に追加します（telemetryEntity）。
16. センサー グループをサブスクリプションに関連付けるために、テレメトリ センサー グループの子オブジェクトとして関係オブジェクトを作成します。
17. テレメトリ接続先グループの子オブジェクトとして関係オブジェクトを作成して、接続先グループをサブスクリプションに関連付けます。
18. サブスクリプションの子オブジェクトとして関係オブジェクトを作成して、サブスクリプションをテレメトリ接続先グループに関連付けます。

19. テレメトリ構成のために、結果の JSON 構造を HTTP/HTTPS POST ペイロードとして NX-API エンドポイントに送信します。

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	テレメトリ機能を有効にします。 例： `{ "fmEntity" : { "children" : [{ "fmTelemetry" : { "attributes" : { "adminSt" : "enabled" } } } ] } }`	ルート要素は fmTelemetry であり、この要素のベース パスは `sys/fm` です。adminSt属性を `enabled` に構成します。
ステップ 2	テレメトリ構成を記述するために、JSONペイロードのルート レベルを作成します。 例： `{ "telemetryEntity": { "attributes": { "dn": "sys/tm" }, } }`	ルート要素は telemetryEntity で、この要素のベースパスは sys/tm です。dn 属性を sys/tm として構成します。
ステップ 3	定義されたセンサー パスを含むセンサーグループを作成します。 例： `"telemetrySensorGroup": { "attributes": { "id": "10", "rn": "sensor-10" }, "children": [{ }] }`	テレメトリセンサーグループは、クラス telemetrySensorGroup のオブジェクトで定義されます。以下のオブジェクト属性を構成します。 • id：センサーグループの識別子。現在は、数字の ID 値のみサポートされています。 • rm：センサーグループオブジェクトの相対名（形式：sensor-id）。 • dataSrc：DEFAULT、DME、YANG、または NX- API の中からデータ送信元を選択します。 センサーグループオブジェクトの子には、センサーパスと 1 つ以上の関係オブジェクト（telemetryRtSensorGroupRel）が含まれ、センサー

	コマンドまたはアクション	目的
		グループをテレメトリサブスクリプションに関連付けます。
ステップ 4	（任意） SSL/TLS 証明書とホストを追加します。 **例**： `{ "telemetryCertificate": { "attributes": { "filename": "root.pem" "hostname": "c.com" } } }`	telemetryCertificate は、テレメトリのサブスクリプション/接続先で SSL/TLS 証明書の場所を定義します。
ステップ 5	テレメトリの接続先グループを定義します。 **例**： `{ "telemetryDestGroup": { "attributes": { "id": "20" } } }`	テレメトリ接続先グループは telemetryEntity で定義されます。id 属性を構成します。
ステップ 6	テレメトリの接続先プロファイルを定義します。 **例**： `{ "telemetryDestProfile": { "attributes": { "adminSt": "enabled" }, "children": [ { "telemetryDestOptSourceInterface": { "attributes": { "name": "lo0" } } } ] } }`	テレメトリの接続先プロファイルは、telemetryDestProfile で定義されています。 • adminSt属性を `enabled` に構成します。 telemetryDestOptSourceInterface の下で、構成されたインターフェイスから接続先に、ソース IP アドレスとともにデータをストリーミングするためのインターフェイス名を使用して name 属性を構成します。
ステップ 7	テレメトリ データの送信先となる IP アドレスとポート番号で構成される、1 つ以上のテレメトリの接続先を定義します。 **例**： `{ "telemetryDest": { "attributes": { "addr": "1.2.3.4",`	テレメトリの接続先は、クラス telemetryDest のオブジェクトで定義されます。以下のオブジェクト属性を構成します。 • addr：接続先の IPアドレス。 • port：接続先のポート番号。 • rn：path-[path] 形式の接続先オブジェクトの相対名。

	コマンドまたはアクション	目的

	<pre>"enc": "GPB",
	"port": "50001",
	"proto": "gRPC",
	"rn": "addr-[1.2.3.4]-port-50001"
	}
	}
	}</pre>
		• enc：送信されるテレメトリデータのエンコーディングタイプ。NX-OS は以下をサポートします。
		• gRPC の Google Protocol Buffer（GBP）。
		• C の JSON。
		• proto：送信されるテレメトリデータのトランスポートプロトコルタイプ。NX-OS は以下をサポートします。
		• gRPC
		• HTTP
		• サポートされているエンコード タイプは次のとおりです。
		• HTTP/JSON はい
		• HTTP/Form-data はい Bin Logging でのみサポートされます。
		• GRPC/GPB-Compact はい ネイティブデータ ソースのみ。
		• GRPC/GPB はい
		• UDP/GPB はい
		UDP/JSON はい

ステップ 8	gRPC チャンクを有効にして、チャンク サイズを 64 ～ 4096 バイトに設定します。
	例：
	<pre>{
	"telemetryDestGrpOptChunking": {
	"attributes": {
	"chunkSize": "2048",
	"dn": "sys/tm/dest-1/chunking"
	}
	}
	}</pre>
		詳細については、「gRPC チャンク」セクションを参照してください。https://www.cisco.com/c/en/us/td/docs/dcn/nx-os/nexus9000/103x/programmability/cisco-nexus-9000-series-nx-os-programmability-guide-release-103x/m-n9k-model-driven-telemetry-101x.html#Cisco_Reference.dita_a97d6406-43b4-4c8d-bb24-f33bb187f126

ステップ 9	テレメトリサブスクリプションを作成して、テレメトリの動作を構成します。 **例**： <pre>"telemetrySubscription": {</pre>	テレメトリサブスクリプションは、クラス telemetrySubscription のオブジェクトで定義されます。以下のオブジェクト属性を構成します。 • id：サブスクリプションの識別子。現在は、数字の ID 値のみサポートされています。

	コマンドまたはアクション	目的

	<pre> "attributes": {
	        "id": "30",
	        "rn": "subs-30"
	    }, "children": [{
	    }]
	}</pre>
		• rn：subs-id という形式のサブスクリプションオブジェクトの相対名。
		
		サブスクリプションオブジェクトの子には、センサー グループ（telemetryRsSensorGroupRel）および接続先グループ（telemetryRsDestGroupRel）の関係オブジェクトが含まれます。

ステップ 10	ルート要素の下の telemetrySubscription 要素に子オブジェクトとしてセンサー グループ オブジェクトを追加します（telemetryEntity）。
	例：
	<pre>{
	      "telemetrySubscription": {
	        "attributes": {
	          "id": "30"
	        }
	        "children": [{
	          "telemetryRsSensorGroupRel": {
	            "attributes": {
	              "sampleIntvl": "5000",
	              "tDn": "sys/tm/sensor-10"
	            }
	          }
	        }
	        ]
	      }
	    }</pre>	

ステップ 11	サブスクリプションの子オブジェクトとして関係オブジェクトを作成して、サブスクリプションをテレメトリ センサー グループに関連付け、データ サンプリング動作を指定します。
	例：
	<pre>"telemetryRsSensorGroupRel": {
	    "attributes": {
	        "rType": "mo",
	        "rn":
	"rssensorGroupRel-[sys/tm/sensor-10]",
	        "sampleIntvl": "5000",
	        "tCl": "telemetrySensorGroup",
	        "tDn": "sys/tm/sensor-10",
	        "tType": "mo"
	    }
	 }</pre>
		関係オブジェクトはクラス telemetryRsSensorGroupRel に属しており、telemetrySubscription の子オブジェクトです。関係オブジェクトの以下のオブジェクト属性を構成します。
		
		• rn：rssensorGroupRel-[sys/tm/sensor-group-id] 形式の関係オブジェクトの相対名。
		
		• sampleIntvl — ミリ秒単位のデータサンプリング期間。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。

	コマンドまたはアクション	目的
		• tCl：ターゲット（センサーグループ）オブジェクトのクラス。telemetrySensorGroup です。 • tDn：ターゲット（センサーグループ）オブジェクトの識別名。sensor-group-id です。 • rType：関係タイプ。管理対象オブジェクトの mo です。 tType: ターゲットタイプ。管理対象オブジェクトの mo です。

ステップ 12	テレメトリをモニタリングする 1 つ以上のセンサーパスまたはノードを定義します。
	例：
	単一センサー パス
	
	`{`
	`  "telemetrySensorPath": {`
	`  "attributes": {`
	`  "path": "sys/cdp",`
	`  "rn": "path-[sys/cdp]",`
	`  "excludeFilter": "",`
	`  "filterCondition": "",`
	`  "path": "sys/fm/bgp",`
	`  "secondaryGroup": "0",`
	`  "secondaryPath": "",`
	`  "depth": "0"`
	`  }`
	`  }`
	`}`
	例：
	複数のセンサー パス
	
	`{`
	`  "telemetrySensorPath": {`
	`  "attributes": {`
	`  "path": "sys/cdp",`
	`  "rn": "path-[sys/cdp]",`
	`  "excludeFilter": "",`
	`  "filterCondition": "",`
	`  "path": "sys/fm/bgp",`
	`  "secondaryGroup": "0",`
	`  "secondaryPath": "",`
	`  "depth": "0"`
	`  }`
	`  }`
	`},`
	`{`
	`  "telemetrySensorPath": {`
	`  "attributes": {`
		センサーパスは、クラス telemetrySensorPath のオブジェクトで定義されます。以下のオブジェクト属性を構成します。
		• path：モニタリングされるパス。
		• rn：path-[path] 形式のパスオブジェクトの相対名：
		• depth：センサーパスの取得レベル。depth 設定が 0 の場合、ルート MO プロパティのみを取得します。
		
		filterCondition：（オプション）イベントベースのサブスクリプション用に特定のフィルタを作成します。DME はフィルター式を提供します。フィルタリングの詳細については、クエリの作成に関する Cisco APIC REST API の使用ガイドラインを参照してください。次の Cisco APIC ドキュメントのランディングページで確認できます。
		https://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html

	コマンドまたはアクション	目的

	<pre>"excludeFilter": "",
	"filterCondition": "",
	"path": "sys/fm/dhcp",
	"secondaryGroup": "0",
	"secondaryPath": "",
	"depth": "0"
	}
	}
	}</pre> **例**：
	BGP 無効化イベントの単一センサー パス フィルタリング：
	<pre>{
	"telemetrySensorPath": {
	"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 13	センサーパスを子オブジェクトとしてセンサーグループオブジェクト（telemetrySensorGroup）に追加します。	
ステップ 14	接続先を子オブジェクトとして接続先グループオブジェクト（telemetryDestGroup）に追加します。	
ステップ 15	接続先グループオブジェクトを子オブジェクトとしてルート要素に追加します（telemetryEntity）。	

ステップ 16	センサー グループをサブスクリプションに関連付けるために、テレメトリ センサー グループの子オブジェクトとして関係オブジェクトを作成します。
	例：
	<pre>"telemetryRtSensorGroupRel": {
	"attributes": {
	"rn":
	"rtsensorGroupRel-[sys/tm/subs-30]",
	"tCl": "telemetrySubscription",
	"tDn": "sys/tm/subs-30"
	}
	}</pre>
		関係オブジェクトはクラス telemetryRtSensorGroupRel に属しており、 telemetrySensorGroup の子オブジェクトです。関係オブジェクトの以下のオブジェクト属性を構成します。
		• rn：rtsensorGroupRel-[sys/tm/subscription-id] 形式の関係オブジェクトの相対名。
		• tCl：サブスクリプションオブジェクトのターゲットクラス。telemetrySubscription です。
		tDn：サブスクリプションオブジェクトのターゲット識別名。sys/tm/subscription-id です。

	コマンドまたはアクション	目的
ステップ 17	テレメトリ接続先グループの子オブジェクトとして関係オブジェクトを作成して、接続先グループをサブスクリプションに関連付けます。 **例**： `"telemetryRtDestGroupRel": { "attributes": { "rn": "rtdestGroupRel-[sys/tm/subs-30]", "tCl": "telemetrySubscription", "tDn": "sys/tm/subs-30" } }`	関係オブジェクトはクラス telemetryRtDestGroupRel に属しており、telemetryDestGroup の子オブジェクトです。関係オブジェクトの以下のオブジェクト属性を構成します。 • rn：rtdestGroupRel-[sys/tm/subscription-id] 形式の関係オブジェクトの相対名。 • tCl：サブスクリプションオブジェクトのターゲットクラス。telemetrySubscription です。 tDn：サブスクリプションオブジェクトのターゲット識別名。sys/tm/subscription-id です。
ステップ 18	サブスクリプションの子オブジェクトとして関係オブジェクトを作成して、サブスクリプションをテレメトリ接続先グループに関連付けます。 **例**： `"telemetryRsDestGroupRel": { "attributes": { "rType": "mo", "rn": "rsdestGroupRel-[sys/tm/dest-20]", "tCl": "telemetryDestGroup", "tDn": "sys/tm/dest-20", "tType": "mo" } }`	関係オブジェクトはクラス telemetryRsDestGroupRel に属しており、telemetrySubscription の子オブジェクトです。関係オブジェクトの以下のオブジェクト属性を構成します。 • rn：rsdestGroupRel-[sys/tm/destination-group-id] 形式の関係オブジェクトの相対名。 • tCl：ターゲット（接続先グループ）オブジェクトのクラス。telemetryDestGroup です。 • tDn：ターゲット（接続先グループ）オブジェクトの識別名。sys/tm/destination-group-id です。 • rType：関係タイプ。管理対象オブジェクトの mo です。 tType: ターゲットタイプ。管理対象オブジェクトの mo です。
ステップ 19	テレメトリ構成のために、結果の JSON 構造を HTTP/HTTPS POST ペイロードとして NX-API エンドポイントに送信します。 **例**：	テレメトリ エンティティのベース パスは `sys/tm` で、NX-API エンドポイントは次のとおりです。 `{{URL}}/api/node/mo/sys/tm.json`

例

以下は、1 つの POST ペイロードに収集された、その前のすべてのステップの例です（一部の属性が一致しない場合があることに注意してください）。

```
{
    "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

テレメトリ機能の構成を開始し、単一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する例については、https://github.com/CiscoDevNet/nx-telemetry-proto/tree/master/postman_collections を参照してください。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。

マルチキャスト フロー パスの可視性

この機能は、Nexus 3548-XL スイッチで使用可能な必要なすべてのマルチキャスト ステートをエクスポートする手段を提供します。エクスポートにより、各フローが送信元から各受信者までたどるパスの完全で信頼性の高いトレーサビリティが確保されます。

この機能は、DME ですべての適切な情報を公開することを目的としており、プッシュモデル（ソフトウェアテレメトリ）またはプルモデル（DME REST クエリ）のいずれかを介してコンシューマ/コントローラにアクセスできるようにします。

この機能の利点は次のとおりです。

- フロー パスの可視化
- 障害検出のためにフローの統計と状態のエクスポート
- ユーザがフローパス上のスイッチで適切なデバッグコマンドを実行できるようにすることによる根本原因の分析

MFDM は、上位レベルのコンポーネントからの情報を消費し、各マルチキャスト機能のインテリジェンスを構築してから、情報をコンシューマに伝達するマルチキャスト FIB 分散管理です。これは、機能が DME とともに実装されるコアコンポーネントです。MRIB によって提供される情報と MFIB によって収集された統計情報に基づいて、すべてのマルチキャスト ステートを DME にパブリッシュします。

DME は、コンシューマ/コントローラが使用できるようにする必要があるすべての情報を保存するために使用されます。また、イベントベースの通知をサポートするためにオブジェクトが作成、削除、または変更されるたびに、テレメトリへの適切な通知を生成します。

テレメトリプロセスは、DME に保存されているすべてのデータをコンシューマにストリーミングし、データを適切な形式でフォーマットします。

マルチキャスト フロー パスの可視性のための CLI

次に、マルチキャスト フロー パスの可視性の正確な機能を確認するために導入された CLI を示します。

- DME への情報のエクスポートを有効にするコンフィギュレーション コマンド。この CLI は、システムに存在するすべてのルートに対してこの機能を有効にします。

```
switch(config)# multicast flow-path export

        switch(config)# sh system internal dme run all dn sys/mca/config
```

- MFDM と DME に存在する状態間の整合性チェックを実行する整合性チェッカーの show コマンド。このコマンドを使用すると、特に大規模なセットアップで不整合をすばやく検出できます。

```
switch# show forwarding distribution internal multicast
        consistency-checker flow-path route

Starting flow-path DME consistency-check for VRF:
        default

(0.0.0.0/0, 230.0.0.1/32). Result:
      PASS

(10.0.0.10/32, 230.0.0.1/32). Result:
      PASS

(0.0.0.0/0, 232.0.0.0/8) 結果：パス
```

- グローバル show コマンドを使用して、この機能がシステムで有効になっているかどうかを確認します。

```
switch(config)# show forwarding distribution internal
        multicast global_state

**** MFDM Flow PATH VISIBILITY INFO
      ****

Multicast flow-path info export enabled:
      Y

BE DME ハンドラ：0x117c3e6c

PE DME ハンドラ：0x117b955c

switch(config)# show forwarding distribution internal
        multicast fpv CC

PASS/FAIL (In case of fail, it will highlight the
        inconsistencies)
```

クラウド スケール ソフトウェア テレメトリ

クラウド スケール ソフトウェア テレメトリについて

Cisco NX-OSは、Tahoe ASIC を使用するCisco Nexus クラウドスケールスイッチをサポートします。このプラットフォームで、サポートされているクラウドスケールスイッチは、ASIC と緊密に統合された TCP/IP サーバをホストします。これにより、スイッチからのテレメトリ データのレポートをすばやく処理できます。サーバーは TCP ポート 7891 を使用します。テレメトリ クライアントはこのポートでサーバーに接続して、最大 10 ミリ秒でハードウェア カウンタ データを取得できます。

クラウドスケールソフトウェアテレメトリには、独自のクライアントプログラムを作成したり、NX-OS リリース 9.3.1 以降にバンドルされているデフォルトのクライアントプログラムを

使用したりする柔軟性があります。クライアントプログラムは、Python 2.7 以降、C、PHP など、TCP/IP をサポートする任意のプログラミング言語で作成できます。クライアント プログラムは、正しいメッセージ フォーマットで作成する必要があります。

NX-OS リリース 9.3(1) 以降、クラウド スケール ソフトウェア テレメトリ機能は NX-OS で使用できます。この機能はデフォルトで有効になっているため、NX-OS 9.3(1) 以降を実行しているサポート対象のスイッチでは、この機能を使用できます。

Cloud Scale ソフトウェア テレメトリ メッセージの形式

Cloud Scale テレメトリは、クライアントとスイッチ上の TCP/IP サーバー間のハンドシェイクで始まります。その間にクライアントは TCP ソケットを介して接続を開始します。クライアント メッセージは、32 ビット整数での 0 です。スイッチは、特定の形式のカウンタ データを含むメッセージで応答します。

NX-OS リリース 9.3(1) では、次のメッセージ フォーマットがサポートされています。独自のクライアントプログラムを作成する場合は、クライアントが開始するメッセージがこの形式に準拠していることを確認してください

長さ	指定します。
4 バイト	ポート数、*N*
56 バイト	各ポートのデータ、合計 56 * *N* バイト。 データの各 56 バイト チャンクは、次のもので構成されます。 • 24 バイトのインターフェイス名 • 8 バイトの送信（TX）パケット • 8 バイトの送信（TX）バイト • 8 バイトの受信（RX）パケット 8 バイトの受信（RX）バイト

クラウド スケール ソフトウェア テレメトリの注意事項と制限事項

次に、クラウド スケール ソフトウェア テレメトリ機能の注意事項と制限事項を示します。

- Cisco NX-OS リリース 9.3(x) より前でサポートされるプラットフォームについては、そのガイドの https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/platform/platform.html のセクションを参照してください。Cisco NX-OS リリース 9.3(x) 以降でサポートされるプラットフォームについては、https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/platform/platform.html を参照してください。
- カスタムクライアントテレメトリ プログラムでは、サポートされているメッセージフォーマットは 1 つです。クライアント プログラムは、この形式に準拠している必要があります。

- Cisco NX-OS リリース 10.3(1)F 以降、ソフトウェアテレメトリは Cisco Nexus 9800 プラットフォームスイッチでサポートされています。

テレメトリ パス ラベル

テレメトリ パス ラベルについて

NX-OS リリース 9.3(1) 以降、モデル駆動型テレメトリはパス ラベルをサポートします。パス ラベルを使用すると、複数のソースからテレメトリ データを一度に簡単に収集できます。この機能では、収集するテレメトリ データのタイプを指定すると、テレメトリ機能によって複数のパスからそのデータが収集されます。次に、機能は情報を 1 つの統合された場所（パス ラベル）に返します。この機能により、次の作業が不要になるため、テレメトリの使用が簡素化されます。

- Cisco DME モデルに関する深く包括的な知識を持っています。
- 収集されるイベントの数と頻度のバランスを取りながら、複数のクエリを作成し、サブスクリプションに複数のパスを追加します。
- スイッチからテレメトリ情報の複数のチャンクを収集し、有用性を簡素化します。

パス ラベルは、モデル内の同じオブジェクト タイプの複数のインスタンスにわたり、カウンタまたはイベントを収集して返します。パス ラベルは、次のテレメトリ グループをサポートします。

- **環境**、ファン、温度、電力、ストレージ、スーパーバイザ、ラインカードなどのシャーシ情報をモニタリング。
- **インターフェイス**、すべてのインターフェイスカウンタとステータスの変更をモニタリング。

このラベルは、**query-condition** コマンドを使用して返されるデータを絞り込むための定義済みキーワードフィルタをサポートします。

- **リソース**、CPU 使用率やメモリ使用率などのシステムリソースをモニタリング。
- **VXLAN**、VXLAN ピア、VXLAN カウンタ、VLAN カウンター、および BGP ピアデータを含む VXLAN EVPN をモニタリング。

データの投票またはイベントの受信

センサー グループのサンプル間隔によって、テレメトリ データがパス ラベルに送信される方法とタイミングが決まります。サンプル間隔は、テレメトリ データを定期的に投票するか、イベントが発生したときにテレメトリ データを収集するように構成できます。

- テレメトリのサンプル間隔がゼロ以外の値に設定されている場合、テレメトリは各サンプル間隔中に環境、インターフェイス、情報技術、および vxlan ラベルのデータを定期的に送信します。
- サンプル間隔がゼロに設定されている場合、環境、インターフェイス、情報技術、vxlan ラベルで動作状態の更新、およびMOの作成と削除が発生するとテレメトリはイベント通知を送信します。

データの投票または受信イベントは相互に排他的です。パス ラベルごとに投票またはイベント駆動型テレメトリを構成できます。

パス ラベル注意事項と制約事項

テレメトリ パス ラベル機能には、次の注意事項と制約事項があります。

- この機能は、Cisco DME データ 送信元のみをサポートします。
- 同じセンサー グループ内の通常の DME パスとユーザビリティ パスを混在させて一致させることはできません。たとえば、`sys/intf` と `interface` を同じセンサー グループに構成することはできません。また、`sys/intf` と `interface` で同じセンサーグループを構成することもできません。この状況が発生した場合、NX-OS は構成を拒否します。
- `oper-speed` や `counters=[detailed]` などのユーザーフィルターキーワードは、`interface` パスに対してのみサポートされます。
- この機能は、`depth` や `filter-condition` などの他のセンサーパスオプションをサポートしていません。
- テレメトリ パス ラベルには、パス ラベルの使用に関する次の制限があります。
  - 小文字のプレフィックス **show** で開始する必要があります。大文字と小文字は区別されるからです。

たとえば、**show version** は許可されます。一方、**SHOW version** または version は使用できません。

- 次の文字を含めることはできません。
  - ;
  - |
- " "または' '
  - 次の単語を含めることはできません。
- telemetry
- conf t

設定

データまたはイベントをポーリングするためのインターフェイスパスの構成

インターフェイス パス ラベルは、すべてのインターフェイス カウンタとステータスの変更をモニタリングします。次のインターフェイス タイプをサポートします。

- 物理
- サブインターフェイス
- 管理
- ループバック
- VLAN
- ポート チャネル

インターフェイス パス ラベルを構成して、定期的にデータをポーリングするか、イベントを受信することができます。45ページの「データのポーリングまたはイベントの受信」を参照してください。

（注） このモデルは、サブインターフェイス、ループバック、または VLAN のカウンタをサポートしていないため、ストリームアウトされません。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path interface**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	**configure terminal** 例：	コンフィギュレーション モードを入力します。

	コマンドまたはアクション	目的
	`switch# configure terminal` `switch(config)#`	

ステップ 2	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		テレメトリ機能の構成モードに入ります。

ステップ 3	**sensor-group** *sgrp_id*
	例：
	`switch(config-telemetry)# sensor-group 6`
	`switch(conf-tm-sensor)#`
		テレメトリ データのセンサー グループを作成します。

ステップ 4	**path interface**
	例：
	`switch(conf-tm-sensor)# path interface`
	`switch(conf-tm-sensor)#`
		インターフェイス パス ラベルを構成して、複数の個々のインターフェイスに対して 1 つのテレメトリ データ クエリを送信できるようにします。ラベルは、複数のインターフェイスのクエリを 1 つに統合します。次に、テレメトリはデータを収集し、ラベルに返します。
		ポーリング間隔の設定方法に応じて、インターフェイスデータは定期的に、またはインターフェイスの状態が変化するたびに送信されます。

ステップ 5	**destination-group** *grp_id*
	例：
	`switch(conf-tm-sensor)# destination-group 33`
	`switch(conf-tm-dest)#`
		テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

ステップ 6	**ip address** *ip_addr* **port** *port*
	例：
	`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004`
	`switch(conf-tm-dest)#`
		サブスクリプションのテレメトリデータを構成して、指定された IP アドレスとポートにストリーミングします。

ステップ 7	**subscription** *sub_id*
	例：
	`switch(conf-tm-dest)# subscription 33`
	`switch(conf-tm-sub)#`
		テレメトリ サブスクリプション サブモードに入り、テレメトリ サブスクリプションを構成します。

ステップ 8	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリ データを定期的に送信するか、インターフェイスイベントが発生したときに送信するかを決定します。

	コマンドまたはアクション	目的

ステップ 9	**dst-group** *dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

非ゼロ カウンタのインターフェイス パスの構成

ゼロ以外の値を持つカウンターのみを返す事前定義されたキーワードフィルタを使用して、インターフェイス パス ラベルを構成できます。フィルタは `counters=[detailed]` です。

このフィルタを使用することにより、インターフェイス パスは使用可能なすべてのインターフェイス カウンターを収集し、収集したデータをフィルタ処理してから、結果を受信側に転送します。フィルタはオプションであり、使用しない場合、ゼロ値カウンターを含むすべてのカウンターがインターフェイス パスに表示されます。

（注） フィルタの使用は、show interface mgmt0 counters detailed の発行と概念的に似ています。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group***sgrp_id*
4. **path interface query-condition counters=[detailed]**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry** 例：	テレメトリ機能の構成モードに入ります。

	コマンドまたはアクション	目的
	`switch(config)# telemetry` `switch(config-telemetry)#`	

ステップ 3	**sensor-group***sgrp_id*
	例：
	`switch(config-telemetry)# sensor-group 6`
	`switch(conf-tm-sensor)#`
		テレメトリ データのセンサー グループを作成します。

ステップ 4	**path interface query-condition counters=[detailed]**
	例：
	`switch(conf-tm-sensor)# path interface query-condition counters=[detailed]`
	`switch(conf-tm-sensor)#`
		インターフェイス パス ラベルを構成し、すべてのインターフェイスからのゼロ以外のカウンターのみを照会します。

ステップ 5	**destination-group** *grp_id*
	例：
	`switch(conf-tm-sensor)# destination-group 33`
	`switch(conf-tm-dest)#`
		テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

ステップ 6	**ip address** *ip_addr* **port** *port*
	例：
	`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004`
	`switch(conf-tm-dest)#`
		サブスクリプションのテレメトリ データを構成して、指定されたIPアドレスとポートにストリーミングします。

ステップ 7	**subscription** *sub_id*
	例：
	`switch(conf-tm-dest)# subscription 33`
	`switch(conf-tm-sub)#`
		テレメトリ サブスクリプション サブモードに入り、テレメトリ サブスクリプションを構成します。

ステップ 8	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリ データを定期的に送信するか、インターフェイスイベントが発生したときに送信するかを決定します。

ステップ 9	**dst-group** *dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

動作速度のインターフェイス パスの構成

指定された動作速度のインターフェイスのカウンタを返す定義済みのキーワードフィルタを使用して、インターフェイスパス ラベルを構成できます。フィルタは `oper-speed=[]` です。次の動作速度がサポートされています: auto、10M、100M、1G、10G、40G、200G、および 400G。

このフィルタを使用することにより、インターフェースパスは指定された速度のインターフェースのテレメトリ データを収集し、その結果を受信側に転送します。フィルタはオプションです。使用しない場合、動作速度に関係なく、すべてのインターフェイスのカウンタが表示されます。

フィルタは、複数の速度をコンマ区切りのリストとして受け入れることができます。たとえば、`oper-speed=[1G,10G]` は、1 および 10 Gbps で動作するインターフェイスのカウンタを取得します。区切り文字として空白を使用しないでください。

（注） インターフェイス タイプ サブインターフェイス、ループバック、および VLAN には動作速度プロパティがないため、フィルタはこれらのインターフェイス タイプをサポートしません。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **snsr-group** *sgrp_id* **sample-interval** *interval*
4. **path interface query-condition oper-speed=[speed]**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		テレメトリ機能の構成モードに入ります。

	コマンドまたはアクション	目的

ステップ 3	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリ データを定期的に送信するか、インターフェイスイベントが発生したときに送信するかを決定します。

ステップ 4	**path interface query-condition oper-speed=[speed]**
	例：
	`switch(conf-tm-sensor)# path interface query-condition oper-speed=[1G,40G]`
	`switch(conf-tm-sensor)#`
		インターフェイスパスラベルを構成し、指定された速度（この例では 1 Gbps と 40 Gbps のみ）で動作しているインターフェイスからのカウンタを照会します。

ステップ 5	**destination-group** *grp_id*
	例：
	`switch(conf-tm-sensor)# destination-group 33`
	`switch(conf-tm-dest)#`
		テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

ステップ 6	**ip address** *ip_addr* **port** *port*
	例：
	`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004`
	`switch(conf-tm-dest)#`
		サブスクリプションのテレメトリ データを構成して、指定された IP アドレスとポートにストリーミングします。

ステップ 7	**subscription** *sub_id*
	例：
	`switch(conf-tm-dest)# subscription 33`
	`switch(conf-tm-sub)#`
		テレメトリ サブスクリプションサブモードに入り、テレメトリ サブスクリプションを構成します。

ステップ 8	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリ データを定期的に送信するか、インターフェイスイベントが発生したときに送信するかを決定します。

ステップ 9	**dst-group** *dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

複数のクエリによるインターフェイス パスの構成

インターフェイス パス ラベルの同じクエリ条件に対して複数のフィルタを構成できます。その場合、使用する個々のフィルタは AND で結合されます。

クエリ条件の各フィルタは、コンマを使用して区切ります。query-condition には、任意の数のフィルタを指定できますが、追加するフィルタが多いほど、結果の焦点が絞られることに注意してください。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path interface query-condition counters=[detailed],oper-speed=[1G,40G]**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		テレメトリ機能の構成モードに入ります。

ステップ 3	**sensor-group** *sgrp_id*
	例：
	`switch(config-telemetry)# sensor-group 6`
	`switch(conf-tm-sensor)#`
		テレメトリ データのセンサー グループを作成します。

ステップ 4	**path interface query-condition counters=[detailed],oper-speed=[1G,40G]**
	例：
	`switch(conf-tm-sensor)# path interface query-condition counters=[detailed],oper-speed=[1G,40G]`
	`switch(conf-tm-sensor)#`
		同じクエリで複数の条件を構成します。この例では、クエリは次の両方を実行します。
		• 1 Gbps で実行されているインターフェイスでゼロ以外のカウンターを収集して返します。
		40 Gbps で実行されているインターフェイスでゼロ以外のカウンターを収集して返します。

	コマンドまたはアクション	目的

ステップ 5	**destination-group** *grp_id*
	例：
	`switch(conf-tm-sensor)# destination-group 33`
	`switch(conf-tm-dest)#`
		テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

ステップ 6	**ip address** *ip_addr* **port** *port*
	例：
	`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004`
	`switch(conf-tm-dest)#`
		サブスクリプションのテレメトリ データを構成して、指定されたIPアドレスとポートにストリーミングします。

ステップ 7	**subscription** *sub_id*
	例：
	`switch(conf-tm-dest)# subscription 33`
	`switch(conf-tm-sub)#`
		テレメトリ サブスクリプション サブモードに入り、テレメトリ サブスクリプションを構成します。

ステップ 8	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリ データを定期的に送信するか、インターフェイスイベントが発生したときに送信するかを決定します。

ステップ 9	**dst-group** *dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

データまたはイベントをポーリングするための環境パスの構成

環境パス ラベルは、ファン、温度、電源、ストレージ、スーパーバイザ、ラインカードなどのシャーシ情報をモニタリングします。テレメトリ データを定期的にポーリングするか、イベントが発生したときにデータを取得するように環境パスを構成できます。詳細については、45 ページの「データのポーリングまたはイベントの受信」を参照してください。

定期的なポーリングまたはイベントに基づいてシステム リソース情報を返すようにリソース パスを設定できます。このパスはフィルタリングをサポートしていません。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path environment**
5. **destination-group** *grp_id*

6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		テレメトリ機能の構成モードに入ります。

ステップ 3	**sensor-group** *sgrp_id*
	例：
	`switch(config-telemetry)# sensor-group 6`
	`switch(conf-tm-sensor)#`
		テレメトリ データのセンサー グループを作成します。

ステップ 4	**path environment**
	例：
	`switch(conf-tm-sensor)# path environment`
	`switch(conf-tm-sensor)#`
		複数の個々の環境オブジェクトのテレメトリ データをラベルに送信できるようにする環境パス ラベルを構成します。ラベルは、複数のデータ入力を 1 つの出力に統合します。
		サンプル間隔に応じて、環境データはポーリング間隔に基づいてストリーミングされるか、イベントが発生したときに送信されます。

ステップ 5	**destination-group** *grp_id*
	例：
	`switch(conf-tm-sensor)# destination-group 33`
	`switch(conf-tm-dest)#`
		テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

ステップ 6	**ip address** *ip_addr* **port** *port*
	例：
	`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004`
	`switch(conf-tm-dest)#`
		サブスクリプションのテレメトリ データを構成して、指定されたIPアドレスとポートにストリーミングします。

ステップ 7	**subscription** *sub_id* 例：	テレメトリ サブスクリプション サブモードに入り、テレメトリ サブスクリプションを構成します。

	コマンドまたはアクション	目的
	`switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	

ステップ 8	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリデータを定期的に送信するか、環境イベントが発生したときに送信するかを決定します。

ステップ 9	**dst-group** *dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

電力使用量トラッキング機能の有効化

NX-OS リリース 10.4.1(F) 以降、Cisco Nexus 9336C-FX2 および 9332D-GX2B スイッチでは、電力消費を追跡するために **power usage-history** コマンドがサポートされています。デフォルトでは、この機能は無効になっています。

Cisco NX-OS リリース 10.4(2)F 以降、Cisco Nexus 9000 シリーズ プラットフォーム スイッチでは、 **power usage-history** コマンドがサポートされています。

この機能を有効にするには、次の手順を実行します。

手順の概要

1. **configure terminal**
2. **[no] power usage-history**

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		構成モードに入ります。

ステップ 2	**[no] power usage-history**
	例：
	`switch# power usage-history`
	`switch(config)#`
		電力使用量トラッキング機能を有効にします。この機能を無効化するには、このコマンドの **no** 形式を使用します。

電力消費履歴の表示

電力使用量追跡の表示コマンド

電力使用量追跡機能を有効にするには、電力使用量トラッキング機能の有効化 （62 ページ）を参照してください。有効にした後、 **show environment power history** を使用してさまざまなターゲットの電力使用量の統計情報を表示します。

コマンド	表示内容
show environment power history { peak | target 1min | target 1hr | target 14 days | target 14days day _day_no }	さまざまなターゲットの電力使用量情報。

コマンドの例

show environment power history peak コマンドの例を次に示します。

```
switch# show environment power history peak
 Power                                    Output Power       Peak  Time            Input
 Power      Peak  Time
Supply    Model             Status         (peak)         (Output Power)          (peak)
        (Input Power)
-------------------------------------------------------------------------------------------------

1      N9K-PAC-3000W-B       Ok             334.30 W      06/07/2023 10:54:29     362.45
 W  06/07/2023 10:54:59
2      N9K-PAC-3000W-B       Ok             362.45 W      06/07/2023 10:53:29     425.80
 W  06/07/2023 10:51:44
switch#



Last 1min usage data would contain average usage in last 15secs, 30secs and 60 secs.

module-4#  show environment power history target 1min

Power                                    Output/Input             Output/Input
    Output/Input
Supply    Model             Status          (15 sec)                (30 sec)
      (60 sec)
-------------------------------------------------------------------------------------------------

1      N9K-PAC-3000W-B       Ok          330.78 W / 362.45  W     330.00 W / 362.00  W
   330.00 W / 362.00  W
2      N9K-PAC-3000W-B       Ok          358.94 W / 415.24  W     358.00 W / 415.00  W
   358.00 W / 415.00  W
switch#
```

show environment power history target 1hr コマンドの出力を次に示します。

```
switch# show environment power history target 1hr
   1 min avg data for 1 Hr for
 slot: 1 Product Name: N9K-PAC-3000W-B status: Ok
   Output Power      Input Power             Time
-------------------------------------------------
    331.00 W          362.00 W          06/07/2023 11:34:44
    330.00 W          362.00 W          06/07/2023 11:33:44
    333.00 W          362.00 W          06/07/2023 11:32:44
```

```
    333.00 W          362.00 W          06/07/2023 11:31:44
    331.00 W          362.00 W          06/07/2023 11:30:44


 1 min avg data for 1 Hr for
 slot: 2 Product Name: N9K-PAC-3000W-B status: Ok
   Output Power      Input Power             Time
-------------------------------------------------------
    358.00 W          417.00 W          06/07/2023 11:34:44
    358.00 W          417.00 W          06/07/2023 11:33:44
    358.00 W          417.00 W          06/07/2023 11:32:44
    358.00 W          417.00 W          06/07/2023 11:31:44
    357.00 W          415.00 W          06/07/2023 11:30:44
```

show environment power history target 24hr コマンドの例を次に示します。

```
switch# show environment power history target 24hr
   1HR  avg data for 24 Hr for
 slot: 1 Product Name: N9K-PAC-3000W-B status: Ok
   Output Power      Input Power             Time
-------------------------------------------------------
    332.15 W          363.56 W          06/07/2023 12:50:44
    332.13 W          363.66 W          06/07/2023 11:50:44

 1HR  avg data for 24 Hr for
 slot: 2 Product Name: N9K-PAC-3000W-B status: Ok
   Output Power      Input Power             Time
-------------------------------------------------------
    358.23 W          416.68 W          06/07/2023 12:50:44
    358.35 W          417.05 W          06/07/2023 11:50:44
switch#
```

show environment power history target 14days コマンドの例を次に示します。

```
switch# show environment power history target 14days
   1 Day avg data  over a period of 14 days
 slot: 1 Product Name: N9K-PAC-3000W-B status: Ok
Day   Output Power        Input Power            Date
------------------------------------------------------------
  1      332.17 W          363.61 W             06/07/23

 1 Day avg data  over a period of 14 days
 slot: 2 Product Name: N9K-PAC-3000W-B status: Ok
Day   Output Power        Input Power            Date
------------------------------------------------------------
  1      358.23 W          416.81 W             06/07/23
switch#

This CLI displays the average usage throughout the day for each day in last 14days. For
 each PSU 14 days average usage is displayed. A detailed per hour usage for each day is
 displayed when day number is given. Output for that is given in next slide.
```

show environment power history target 14days day 1 コマンドの例を次に示します。

```
switch# show environment power history target 14days day 1
 1 HR  avg data for 1 Day
 slot: 1 Product Name: N9K-PAC-3000W-B status: Ok
Day 1
   Output Power      Input Power             Time
-------------------------------------------------------
    332.23 W          363.61 W          06/07/2023 13:50:44
    332.15 W          363.56 W          06/07/2023 12:50:44
    332.13 W          363.66 W          06/07/2023 11:50:44
```

```
 1 HR  avg data for 1 Day
 slot: 2 Product Name: N9K-PAC-3000W-B status: Ok
Day 1
   Output Power      Input Power             Time
--------------------------------------------------------
    358.11 W          416.71 W          06/07/2023 13:50:44
    358.23 W          416.68 W          06/07/2023 12:50:44
    358.35 W          417.05 W          06/07/2023 11:50:44
switch#
switch#
```

イベントまたはデータをポーリングするためのリソース パスの構成

リソースパスは、CPU使用率やメモリ使用率などのシステムリソースをモニタリングします。このパスを構成して、テレメトリデータを定期的に収集するか、イベントが発生したときに収集できます。45ページの「データのポーリングまたはイベントの受信」を参照してください。

このパスはフィルタリングをサポートしていません。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group***sgrp_id*
4. **path resources**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		テレメトリ機能の構成モードに入ります。

ステップ 3	**sensor-group***sgrp_id* 例：	テレメトリ データのセンサー グループを作成します。

	コマンドまたはアクション	目的
	`switch(config-telemetry)# sensor-group 6` `switch(conf-tm-sensor)#`	

ステップ 4	**path resources**
	例：
	`switch(conf-tm-sensor)# path resources`
	`switch(conf-tm-sensor)#`
		複数の個々のシステム リソースのテレメトリ データをラベルに送信できるようにするリソース パス ラベルを構成します。ラベルは、複数のデータ入力を 1 つの出力に統合します。
		サンプル間隔に応じて、リソースデータはポーリング間隔に基づいてストリーミングされるか、システム メモリが「Not OK」に変更されたときに送信されます。

ステップ 5	**destination-group** *grp_id*
	例：
	`switch(conf-tm-sensor)# destination-group 33`
	`switch(conf-tm-dest)#`
		テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

ステップ 6	**ip address** *ip_addr* **port** *port*
	例：
	`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004`
	`switch(conf-tm-dest)#`
		サブスクリプションのテレメトリ データを構成して、指定された IP アドレスとポートにストリーミングします。

ステップ 7	**subscription** *sub_id*
	例：
	`switch(conf-tm-dest)# subscription 33`
	`switch(conf-tm-sub)#`
		テレメトリ サブスクリプション サブモードに入り、テレメトリ サブスクリプションを構成します。

ステップ 8	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリ データを定期的に送信するか、リソース イベントが発生したときに送信するかを決定します。

ステップ 9	**dst-group** *dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

イベントまたはデータをポーリングするための VXLAN パスの構成

vxlan パス ラベルは、VXLAN ピア、VXLAN カウンター、VLAN カウンター、BGP ピア データなど、スイッチの仮想拡張 LAN EVPN に関する情報を提供します。このパス ラベルを構成して、定期的に、またはイベントが発生したときにテレメトリ情報を収集できます。「ネイ

ティブ データ送信元パス用にストリーミングされるテレメトリ データ （72 ページ）」を参照してください。

このパスはフィルタリングをサポートしていません。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **vxlan environment**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		テレメトリ機能の構成モードに入ります。

ステップ 3	**sensor-group** *sgrp_id*
	例：
	`switch(config-telemetry)# sensor-group 6`
	`switch(conf-tm-sensor)#`
		テレメトリ データのセンサー グループを作成します。

ステップ 4	**vxlan environment**
	例：
	`switch(conf-tm-sensor)# vxlan environment`
	`switch(conf-tm-sensor)#`
		複数の個々の VXLAN オブジェクトのテレメトリ データをラベルに送信できるようにする vxlan パス ラベルを構成します。ラベルは、複数のデータ入力を 1 つの出力に統合します。サンプル間隔に応じて、VXLAN データはポーリング間隔に基づいてストリーミングされるか、イベントが発生したときに送信されます。

	コマンドまたはアクション	目的

ステップ 5	**destination-group** *grp_id*
	例：
	`switch(conf-tm-sensor)# destination-group 33`
	`switch(conf-tm-dest)#`
		テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

ステップ 6	**ip address** *ip_addr* **port** *port*
	例：
	`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004`
	`switch(conf-tm-dest)#`
		サブスクリプションのテレメトリ データを構成して、指定されたIPアドレスとポートにストリーミングします。

ステップ 7	**subscription** *sub_id*
	例：
	`switch(conf-tm-dest)# subscription 33`
	`switch(conf-tm-sub)#`
		テレメトリ サブスクリプション サブモードに入り、テレメトリ サブスクリプションを構成します。

ステップ 8	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリデータを定期的に送信するか、VXLANイベントが発生したときに送信するかを決定します。

ステップ 9	**dst-group** *dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

パス ラベル 構成 を確認

いつでも、パスラベルが構成されていることを確認し、実行中のテレメトリ構成を表示してその値を確認できます。

手順の概要

1. **show running-config-telemetry**

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**show running-config-telemetry**
	例：
	`switch(conf-tm-sensor)# show running-config telemetry`
	
	`!Command: show running-config telemetry`
	`!Running configuration last done at: Mon Jun 10 08:10:17 2019`
	`!Time: Mon Jun 10 08:10:17 2019`
	
	`version 9.3(1) Bios:version`
	`feature telemetry`
	
	`telemetry`
	`  destination-profile`
	`    use-nodeid tester`
	`    sensor-group 4`
	`      path interface query-condition and(counters=[detailed],oper-speed=[1G,10G])`
	`    sensor-group 6`
	`      path interface query-condition oper-speed=[1G,40G]`
	`    subscription 6`
	`      snsr-grp 6 sample-interval 6000`
	`nxosv2(conf-tm-sensor)#`
		現在のテレメトリの実行構成を表示します。この例では、センサーグループ 4 は、1 および 10 Gbps で動作しているインターフェイスから、ゼロ以外のカウンターを収集するように構成されています。センサー グループ 6 は、1 と 40 Gbps で実行されているインターフェイスからすべてのカウンターを収集するように構成されています。

パス ラベル情報の表示

パス ラベル表示コマンド

show telemetry usability コマンドを使用すると、クエリを発行したときにパスラベルがたどる個々のパスを表示できます。

コマンド	表示内容
show telemetry usability {all | environment | interface | resources | vxlan}	すべてのパス ラベルのすべての パス ラベルのすべてのテレメト 定期的なポーリングまたはイベ 報告するかどうかが示されます インターフェイス パス ラベルに たはクエリ条件も含まれます。
show running-config telemetry	テレメトリと選択されたパス情

コマンドの例

（注） show telemetry usability all コマンドは、このセクションに示されている個々のコマンドをすべて連結したものとして働きます。

次に、**show telemetry usability environment** コマンドの例を示します。

```
switch# show telemetry usability environment
1) label_name : environment
path_name : sys/ch query_type : poll query_condition :
rsp-subtree=full&query-target=subtree&target-subtree-class=eqptPsuSlot,eqptFtSlot,eqptSupCSlot,eqptPsu,eqptFt,eqptSensor,eqptLCSlot
2) label_name : environment
path_name : sys/ch query_type : event query_condition :
[illegible]
 switch#
```

次に、**show telemetry usability interface** コマンドの出力を示します。

```
switch# show telemetry usability interface
1) label_name : interface
path_name : sys/intf query_type : poll query_condition :
query-target=children&query-target-filter=eq(l1PhysIf.adminSt,"up")&rsp-subtree=children&rsp-subtree-class=rmonEtherStats,rmonIfIn,rmonIfOut,rmonIfHCIn,rmonIfHCOut
2) label_name : interface
path_name : sys/mgmt-[mgmt0] query_type : poll query_condition :
query-target=subtree&query-target-filter=eq(mgmtMgmtIf.adminSt,"up")&rsp-subtree=full&rsp-subtree-class=rmonEtherStats,rmonIfIn,rmonIfOut,rmonIfHCIn,rmonIfHCOut
3) label_name : interface
path_name : sys/intf query_type : event query_condition :
[illegible]
ethpmEncRtdIf.operSt,"down")),and(updated(ethpmEncRtdIf.operSt),eq(ethpmEncRtdIf.operSt,"up"))))
4) label_name : interface
path_name : sys/mgmt-[mgmt0] query_type : event query_condition :
query-target=subtree&query-target-filter=or(or(deleted(),created()),or(and(updated(mgmtIf.operSt),eq(mgmtIf.operSt,"down")),and(updated(mgmtIf.operSt),eq(mgmtIf.operSt,"up"))))
 switch#
```

次に、**show telemetry usability resources** コマンドの例を示します。

```
switch# show telemetry usability resources
  1) label_name          : resources

     path_name           : sys/proc
     query_type          : poll
     query_condition     : rsp-subtree=full&rsp-foreign-subtree=ephemeral

  2) label_name          : resources

     path_name           : sys/procsys
     query_type          : poll
     query_condition     :
query-target=subtree&target-subtree-class=procSys,procSCpu,procSMem,procSProc,procTable,procEntry,procSFork,procSTask,procSHistory,procSAvg,procSMemReq,procSMemResp,procSMisc,procSMiscEntry

  3) label_name          : resources

     path_name           : sys/procsys/sysmem
     query_type          : event
     query_condition     :
query-target-filter=and(updated(procSysMem.memstatus),ne(procSysMem.memstatus,"OK"))

switch#
```

次に、**show telemetry usability vxlan** コマンドの例を示します。

```
switch# show telemetry usability vxlan
  1) label_name          : vxlan

     path_name           : sys/bd
     query_type          : poll
     query_condition     : query-target=subtree&target-subtree-class=l2VlanStats

  2) label_name          : vxlan

     path_name           : sys/eps
     query_type          : poll
     query_condition     : rsp-subtree=full&rsp-foreign-subtree=ephemeral

  3) label_name          : vxlan

     path_name           : sys/eps
     query_type          : event
     query_condition     : query-target=subtree&target-subtree-class=nvoDyPeer

  4) label_name          : vxlan

     path_name           : sys/bgp
     query_type          : event
    query_condition    : query-target=subtree&query-target-filter=or(deleted(),created())


  5) label_name          : vxlan

     path_name           : sys/bgp
     query_type          : event
     query_condition     :
query-target=subtree&target-subtree-class=bgpDom,bgpPeer,bgpPeerAf,bgpDomAf,bgpPeerAfEntry,bgpOperRtctrlL3,bgpOperRtP,bgpOperRttEntry,bgpOperAfCtrl

switch#
```

ネイティブ データ送信元パス

ネイティブ データ送信元パスについて

NX-OSテレメトリは、特定のインフラストラクチャまたはデータベースに限定されないニュートラル データ 送信元であるネイティブ データ ソースをサポートします。代わりに、ネイティブ データ 送信元を使用すると、コンポーネントまたはアプリケーションをフックして、関連情報を発信テレメトリ ストリームに挿入できます。ネイティブ データ 送信元のパスはインフラストラクチャに属さないため、この機能は柔軟性を提供し、ネイティブアプリケーションはNX-OS テレメトリと対話できます。

ネイティブ データ 送信元 パスを使用すると、特定のセンサー パスに登録して、セレクトしたテレメトリ データを受信できます。この機能は NX-SDK と連携して、次のパスからのテレメトリ データのストリーミングをサポートします。

- IP ルートのテレメトリ データを送信する RIB パス。
- 静的および動的 MAC エントリのテレメトリ データを送信する MAC パス。
- IPv4 と IPv6 隣接のテレメトリ データを送信する隣接関係パス。

サブスクリプションを作成すると、選択したパスのすべてのテレメトリデータが基準値として受信者にストリーミングされます。基準値の後、イベント通知のみが受信者にストリーミングされます。

ネイティブ データ 送信元 パスのストリーミングは、次のエンコーディング タイプをサポートします：

- Google Protobuf（GPB）
- JavaScript Object Notation（JSON）
- コンパクト Google Protobuf (コンパクト GPB)

ネイティブ データ送信元パス用にストリーミングされるテレメトリ データ

次の表は、各ソース パスについて、サブスクリプションが最初に作成されたとき（ベースライン）とイベント通知が発生したときにストリーミングされる情報を示しています。

Path Type	サブスクリプション ベースライン	イベント通

RIB	全てのルートの送信	イベント イベント パスのテ ます： • ネク • レイ ネクスト

MAC	静的およびダイナミック MAC エントリに対して DME から GETALL を実行します。	イベントのベント通知スのテレメす： • MAC • MAC • VLAN • インタ • イベン イベント通ク エントリ
隣接	IPv4 および IPv6 隣接関係（アジャセンシー）を送信します。	イベントのベント通知係（アジャじてエクス • IP アド • MAC • インタ • 物理イ • VRF 名 • プリフ • 隣接の • 隣接関ファミ 隣接関係（プ

詳細については、Github の https://github.com/CiscoDevNet/nx-telemetry-proto を参照してください。

注意事項と制約事項

ネイティブ データ 送信元 パス機能には、次の注意事項と制約事項があります。

- RIB、MAC、および隣接関係（アジャセンシー）のネイティブデータ送信元パスからのストリーミングの場合、センサーパスプロパティの更新は、**depth**、**query-condition**、または **filter-condition** などのカスタム基準をサポートしません。
- Cisco NX-OSリリース 10.4(3)F 以降では、RIB ネイティブ パスのサンプル ベースのサブスクリプションまたは更新のみをサポートする、新しいクエリ条件が導入されています。

ルーティング情報のネイティブ データ送信元パスの構成

URIB に含まれるすべてのルートに関する情報を送信するルーティング情報のネイティブ データ送信元パスを構成できます。登録すると、基準値はすべてのルート情報を送信します。ベースラインの後、スイッチがサポートするルーティングプロトコルのルート更新と削除操作について通知が送信されます。RIB通知で送信されるデータについては、61ページの「ネイティブデータソースパス用にストリーミングされるテレメトリデータ」を参照してください。

始める前に

テレメトリ機能を有効にしていない場合は、ここで有効にします（**機能テレメトリ**）。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source native**
5. **path rib query-condition** [**data=ephemeral** | **updates_only**]
6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
8. subscription sub_id
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry** 例：	テレメトリ機能の構成モードに入ります。

	コマンドまたはアクション	目的
	`switch(config)# telemetry` `switch(config-telemetry)#`	

ステップ 3	**sensor-group** *sgrp_id*
	例：
	`switch(conf-tm-sub)# sensor-grp 6`
	`switch(conf-tm-sub)#`
		センサー グループを作成します。

ステップ 4	**data-source native**
	例：
	`switch(conf-tm-sensor)# data-source native`
	`switch(conf-tm-sensor)#`
		特定のモデルやデータベースを必要とせずに、ネイティブ アプリケーションがストリーム データを使用できるように、データ送信元をネイティブに設定します。

ステップ 5	**path rib query-condition** [**data=ephemeral** | **updates_only**]
	例：
	`nxosv2(conf-tm-sensor)# path rib`
	`nxosv2(conf-tm-sensor)#`
	例：
	`nxosv2(conf-tm-sensor)# path rib query condition data=ephemeral`
	`nxosv2(conf-tm-sensor)#`
	例：
	`nxosv2(conf-tm-sensor)# path rib query condition updates_only`
	`nxosv2(conf-tm-sensor)#`
		ルートとルート アップデート情報をストリーミングする RIB パスを構成します。
		query condition data=ephemeral（オプション）：サンプル間隔 0 または 0 以外を設定できます。このサンプル間隔によって、接続先にルート情報が定期的に送信される頻度が決まります（構成されたサンプル間隔で）。
		query condition updates-only（オプション）：サンプル間隔 0 でのみサポートされます。このクエリ条件では、最初のスナップショットデータは送信されず、ルート情報の更新のみが接続先に送信されます。

ステップ 6	**destination-group** *grp_id*
	例：
	`switch(conf-tm-sensor)# destination-group 33`
	`switch(conf-tm-dest)#`
		テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

ステップ 7	**ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
	例：
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json`
	`switch(conf-tm-dest)#`
	`Example:`
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb`
	`switch(conf-tm-dest)#`
	`Example:`
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact`
	`switch(conf-tm-dest)#`
		サブスクリプションのテレメトリ データを、指定された IP アドレスとポートにストリーミングするように構成し、データ ストリームのプロトコルとエンコードを設定します。

ステップ 8	subscription sub_id 例：	テレメトリ サブスクリプション サブモードに入り、テレメトリ サブスクリプションを構成します。

	コマンドまたはアクション	目的
	`switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	

ステップ 9	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリデータを定期的に送信するか、rib イベントが発生したときに送信するかを決定します。
		（注）
		サンプリング間隔に応じて、rib センサーパスはポーリング間隔に基づいてストリーミングします。

ステップ 10	**dst-group** *dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

MAC 情報のネイティブ データ送信元パスの構成

MAC テーブルのすべてのエントリに関する情報を送信する MAC 情報のネイティブ データ 送信元 パスを構成できます。登録すると、基準値はすべての MAC 情報を送信します。基準値の後、MAC アドレスの追加、更新、および削除操作の通知が送信されます。MAC 通知で送信されるデータについては、ネイティブ データ送信元パス用にストリーミングされるテレメトリデータ （72 ページ） を参照してください。

（注） 更新または削除イベントの場合、MAC 通知は、IP 隣接関係を持つ MAC アドレスに対してのみ送信されます。

始める前に

テレメトリ機能を有効にしていない場合は、ここで有効にします（**機能テレメトリ**）。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source native**
5. **path mac**
6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**

8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group***dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		テレメトリ機能の構成モードに入ります。

ステップ 3	**sensor-group** *sgrp_id*
	例：
	`switch(conf-tm-sub)# sensor-grp 6`
	`switch(conf-tm-sub)#`
		センサー グループを作成します。

ステップ 4	**data-source native**
	例：
	`switch(conf-tm-sensor)# data-source native`
	`switch(conf-tm-sensor)#`
		特定のモデルやデータベースを必要とせずに、ネイティブ アプリケーションがストリーム データを使用できるように、データ送信元をネイティブに設定します。

ステップ 5	**path mac**
	例：
	`nxosv2(conf-tm-sensor)# path mac`
	`nxosv2(conf-tm-sensor)#`
		MAC エントリおよび MAC 通知に関する情報をストリームする MAC パスを構成します。

ステップ 6	**destination-group** *grp_id*
	例：
	`switch(conf-tm-sensor)# destination-group 33`
	`switch(conf-tm-dest)#`
		テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

ステップ 7	**ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
	例：
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json`
	`switch(conf-tm-dest)#`
	例：
		サブスクリプションのテレメトリ データを、指定された IP アドレスとポートにストリーミングするように構成し、データ ストリームのプロトコルとエンコードを設定します。

	コマンドまたはアクション	目的

	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb`
	`switch(conf-tm-dest)#`
	例：
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact`
	`switch(conf-tm-dest)#`	

ステップ 8	**subscription** *sub_id*
	例：
	`switch(conf-tm-dest)# subscription 33`
	`switch(conf-tm-sub)#`
		テレメトリ サブスクリプションサブモードに入り、テレメトリ サブスクリプションを構成します。

ステップ 9	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリ データを定期的に送信するか、インターフェイス イベントが発生したときに送信するかを決定します。

ステップ 10	**dst-group***dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

すべての MAC 情報のネイティブデータ送信元パスの構成

レイヤ 3 およびレイヤ 2 から、MAC テーブルのすべてのエントリに関する情報を送信する MAC 情報のネイティブ データ 送信元 パスを構成できます。登録すると、基準値はすべての MAC 情報を送信します。

基準値の後、MACアドレスの追加、更新、および削除操作の通知が送信されます。MAC通知で送信されるデータについては、ネイティブデータ送信元パス用にストリーミングされるテレメトリ データ （72 ページ） を参照してください。

（注） 更新または削除イベントの場合、MAC 通知は、IP 隣接関係を持つ MAC アドレスに対してのみ送信されます。

始める前に

テレメトリ機能を有効にしていない場合は、ここで有効にします（**機能テレメトリ**）。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source** *native*
5. **path mac-all**
6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		テレメトリ機能の構成モードに入ります。

ステップ 3	**sensor-group** *sgrp_id*
	例：
	`switch(conf-tm-sub)# sensor-grp 6`
	`switch(conf-tm-sub)#`
		センサー グループを作成します。

ステップ 4	**data-source** *native*
	例：
	`switch(conf-tm-sensor)# data-source native`
	`switch(conf-tm-sensor)#`
		特定のモデルやデータベースを必要とせずに、ネイティブ アプリケーションがストリーム データを使用できるように、データ送信元をネイティブに設定します。

ステップ 5	**path mac-all**
	例：
	`nxosv2(conf-tm-sensor)# path mac-all`
	`nxosv2(conf-tm-sensor)#`
		すべての MAC エントリおよび MAC 通知に関する情報をストリームする MAC パスを構成します。

ステップ 6	**destination-group** *grp_id* 例：	テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

	コマンドまたはアクション	目的
	`switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	

ステップ 7	**ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
	例：
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json`
	`switch(conf-tm-dest)#`
	例：
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb`
	`switch(conf-tm-dest)#`
	例：
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact`
	`switch(conf-tm-dest)#`
		サブスクリプションのテレメトリ データを、指定された IP アドレスとポートにストリーミングするように構成し、データ ストリームのプロトコルとエンコードを設定します。

ステップ 8	**subscription** *sub_id*
	例：
	`switch(conf-tm-dest)# subscription 33`
	`switch(conf-tm-sub)#`
		テレメトリ サブスクリプション サブモードに入り、テレメトリ サブスクリプションを構成します。

ステップ 9	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリ データを定期的に送信するか、インターフェイス イベントが発生したときに送信するかを決定します。

ステップ 10	**dst-group** *dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

IP 隣接のネイティブ データ パスの構成

スイッチのすべての IPv4 と IPv6 隣接に関する情報を送信する IP 隣接情報のネイティブ データ送信元パスを構成できます。登録すると、基準値はすべての隣接情報を送信します。基準値の後、隣接操作の追加、更新、および削除に関する通知が送信されます。隣接関係通知で送信されるデータについては、61 ページの「ネイティブデータソースパス用にストリーミングされるテレメトリデータ」を参照してください。

始める前に

テレメトリ機能を有効にしていない場合は、ここで有効にします（**機能テレメトリ**）。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source** *native*
5. **path adjacency**
6. **destination-group** *grp_id*
7. **ip address ip_addr port port protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		テレメトリ機能の構成モードに入ります。

ステップ 3	**sensor-group** *sgrp_id*
	例：
	`switch(conf-tm-sub)# sensor-grp 6`
	`switch(conf-tm-sub)#`
		センサー グループを作成します。

ステップ 4	**data-source** *native*
	例：
	`switch(conf-tm-sensor)# data-source native`
	`switch(conf-tm-sensor)#`
		ネイティブ アプリケーションがストリーム データを使用できるように、データ送信元をネイティブに設定します。

ステップ 5	**path adjacency**
	例：
	`nxosv2(conf-tm-sensor)# path adjacency`
	`nxosv2(conf-tm-sensor)#`
		IPv4 と IPv6 隣接に関する情報をストリームする隣接パスを構成します。

ステップ 6	**destination-group** *grp_id* 例：	テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

	コマンドまたはアクション	目的
	`switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	

ステップ 7	**ip address ip_addr port port protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
	例：
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json`
	`switch(conf-tm-dest)#`
	例：
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb`
	`switch(conf-tm-dest)#`
	例：
	`switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact`
	`switch(conf-tm-dest)#`
		サブスクリプションのテレメトリ データを、指定された IP アドレスとポートにストリーミングするように構成し、データ ストリームのプロトコルとエンコードを設定します。

ステップ 8	**subscription** *sub_id*
	例：
	`switch(conf-tm-dest)# subscription 33`
	`switch(conf-tm-sub)#`
		テレメトリ サブスクリプション サブモードに入り、テレメトリ サブスクリプションを構成します。

ステップ 9	**snsr-group** *sgrp_id* **sample-interval** *interval*
	例：
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000`
	`switch(conf-tm-sub)#`
		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。サンプリング間隔は、スイッチがテレメトリ データを定期的に送信するか、インターフェイス イベントが発生したときに送信するかを決定します。

ステップ 10	**dst-group** *dgrp_id*
	例：
	`switch(conf-tm-sub)# dst-grp 33`
	`switch(conf-tm-sub)#`
		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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

ネイティブ データ ソース パス情報の表示

NX-OS の **show telemetry event collector** コマンドを使用して、ネイティブデータソースパスの統計情報とカウンタ、またはエラーを表示できます

統計の表示

show telemetry event collector stats コマンドを発行して、各ネイティブデータソースパスの統計情報とカウンタを表示できます。

RIB パスの統計情報の例：

```
switch# show telemetry event collector stats
------------------------------------------------------------------------------
 Row ID Collection Count Latest Collection Time Sensor Path(GroupId)
------------------------------------------------------------------------------
1 4 Mon Jul 01 13:53:42.384 PST rib(1) switch#
An example of the statistics for the MAC path:
switch# show telemetry event collector stats
------------------------------------------------------------------------------
 Row ID Collection Count Latest Collection Time Sensor Path(GroupId)
------------------------------------------------------------------------------
1 3 Mon Jul 01 14:01:32.161 PST mac(1) switch#
An example of the statistics for the Adjacency path:
 switch# show telemetry event collector stats
------------------------------------------------------------------------------
 Row ID Collection Count Latest Collection Time Sensor Path(GroupId)
------------------------------------------------------------------------------
 1 7 Mon Jul 01 14:47:32.260 PST adjacency(1)
switch#
```

エラー カウンタの表示

show telemetry event collector stats コマンドを使用して、すべてのネイティブデータソースパスのエラーの合計を表示できます。

```
switch# show telemetry event collector errors
------------------------------------------------------------------------------
 Error Description Error Count
------------------------------------------------------------------------------
 Dme Event Subscription Init Failures - 0
 Event Data Enqueue Failures - 0
 Event Subscription 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
switch#
```

ストリーミング Syslog

テレメトリ用のストリーミング Syslog について

Cisco NX-OS リリース 9.3(3) 以降、モデル駆動型テレメトリは、YANG をデータソースとして使用する syslog のストリーミングをサポートします。サブスクリプションを作成すると、すべての syslog が基準値として受信者にストリーミングされます。この機能は NX-SDK と連携して、次の syslog パスからのストリーミング syslog データをサポートします。

- Cisco-NX-OS-Syslog-oper:syslog
- Cisco-NX-OS-Syslog-oper:syslog/messages

基準値の後は、syslog イベント通知のみが受信者にストリーミングされます。syslog パスのストリーミングは、次のエンコーディング タイプをサポートします：

- Google Protobuf（GPB）

• JavaScript Object Notation（JSON）

syslog 情報のための YANG データ ソース パスの構成

スイッチで生成されたすべての syslog に関する情報を送信する syslog の syslog パスを構成できます。サブスクライブすると、ベースラインはすべての既存の syslog 情報を送信します。ベースラインの後、通知は、スイッチで生成された新しい syslog に対してのみ送信されます。

始める前に

テレメトリ機能を有効にしていない場合は、**feature telemetry** コマンドで有効にします。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data source** *data-source-type*
5. **path Cisco-NX-OS-Syslog-oper:syslog/messages**
6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
8. **subscription** *sub-id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

手順の詳細

手順

	コマンドまたはアクション	目的

ステップ 1	**configure terminal**
	例：
	`switch# configure terminal`
	`switch(config)#`
		コンフィギュレーション モードを入力します。

ステップ 2	**telemetry**
	例：
	`switch(config)# telemetry`
	`switch(config-telemetry)#`
		テレメトリ機能の構成モードに入ります。

ステップ 3	**sensor-group** *sgrp_id*
	例：
	`switch(conf-tm-sub)# sensor-grp 6`
	`switch(conf-tm-sub)#`
		センサー グループを作成します。

	コマンドまたはアクション	目的
ステップ 4	**data source** *data-source-type* 例： `switch(config-tm-sensor)# data source YANG`	データソースをYANGに設定し、ネイティブYANGストリーミングモデルを使用してsyslogをストリーミングできるようにします。
ステップ 5	**path Cisco-NX-OS-Syslog-oper:syslog/messages** 例： `switch(config-tm-sensor)# path Cisco-NX-OS-Syslog-oper:syslog/messages`	スイッチで生成された syslog をストリーミングする syslog パスを設定します。
ステップ 6	**destination-group** *grp_id* 例： `switch(config-tm-sensor)# destination-group 33`	テレメトリ接続先グループ サブモードに入り、接続先グループを構成します。

ステップ 7	**ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
	例：
	`switch(config-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json`
	例：
	`switch(config-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb`
		サブスクリプションのテレメトリ データを、指定された IP アドレスとポートにストリーミングするように構成し、データ ストリームのプロトコルとエンコードを設定します。

ステップ 8	**subscription** *sub-id* 例： `switch(config-tm-dest)# subscription 33`	テレメトリ サブスクリプション サブモードに入り、テレメトリ サブスクリプションを構成します。
ステップ 9	**snsr-group** *sgrp_id* **sample-interval** *interval* 例： `switch(config-tm-sub)# snsr-group 6 sample-interval 0`	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、データ サンプリングを 0 に設定して、syslogイベントが発生したときにスイッチがテレメトリ データを送信するようにします。interval については、0 のみが受け入れ可能な値です。
ステップ 10	**dst-group** *dgrp_id* 例： `switch(config-tm-sub)# dst-grp 33`	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。指定する接続先グループは、destination-group コマンドで構成した接続先グループと一致している必要があります。

Syslog パスのテレメトリ データ ストリーミング

送信元パスごとに、次のテーブルは、サブスクリプションが最初に作成されるときの「ベースライン」で、そしてイベントの通知が発生するときに、どんな情報がストリーミングされるかを示しています。

パス	サブスクリプション ベースライン	イベント通

Cisco-NX-OS-Syslog-oper:syslog/messages	スイッチから既存のすべての syslog をストリーミングします。	スイッ 送信し • mes • nod • time • time • time • cate • mes • seve • text

syslog パス情報の表示

Cisco NX-OS の **show telemetry event collector** コマンドを使用して、syslog パスの統計情報とカウンタ、またはエラーを表示できます

統計の表示

show telemetry event collector stats コマンドを入力して、各 syslog パスの統計情報とカウンタを表示できます。

次に、syslog パスの統計情報の例を示します。

```
------------------------------------------------------------------------------
Row ID         Collection Count  Latest Collection Time     Sensor Path(GroupId)
------------------------------------------------------------------------------
1              138               Tue Dec 03 11:20:08.200 PST
Cisco-NX-OS-Syslog-oper:syslog(1)
2              138               Tue Dec 03 11:20:08.200 PST
Cisco-NX-OS-Syslog-oper:syslog/messages(1)
```

エラー カウンタの表示

show telemetry event collector errors コマンドを使用して、すべての syslog パスのエラーの合計を表示できます。

```
switch(config-if)# show telemetry event collector errors

------------------------------------------------------------------------------
Error Description                                   Error Count
------------------------------------------------------------------------------
Dme Event Subscription Init Failures                        - 0
Event Data Enqueue Failures                                 - 0
Event Subscription 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
```

JSON 出力の例

次に、JSON 出力のサンプルを示します。

```
172.19.216.13 - - [03/Dec/2019 19:38:50] "POST
/network/Cisco-NX-OS-Syslog-oper%3Asyslog%2Fmessages HTTP/1.0" 200 -
172.19.216.13 - - [03/Dec/2019 19:38:50] "POST
/network/Cisco-NX-OS-Syslog-oper%3Asyslog%2Fmessages HTTP/1.0" 200 -
>>> URL            : /network/Cisco-NX-OS-Syslog-oper%3Asyslog%2Fmessages
>>> TM-HTTP-VER    : 1.0.0
>>> TM-HTTP-CNT    : 1
>>> Content-Type   : application/json
>>> Content-Length : 578
    Path => Cisco-NX-OS-Syslog-oper:syslog/messages
            node_id_str   : task-n9k-1
            collection_id : 40
            data_source   : YANG
            data          :
[
  [
    {
      "message-id": 420
    },
    {
      "category": "ETHPORT",
      "group": "ETHPORT",
      "message-name": "IF_UP",
      "node-name": "task-n9k-1",
      "severity": 5,
      "text": "Interface loopback10 is up ",
      "time-of-day": "Dec 3 2019 11:38:51",
      "time-stamp": "1575401931000",
      "time-zone": ""
    }
  ]
]
```

KVGPB の出力例

次に KVGPB の出力例を示します。

```
---Telemetry msg received @ 18:22:04 UTC


All the fragments:1 read successfully total size read:339

node_id_str: "task-n9k-1"
subscription_id_str: "1"
collection_id: 374
data_gpbkv {
  fields {
    name: "keys"
    fields {
      name: "message-id"
      uint32_value: 374
    }
  }

  fields {
    name: "content"
    fields {
      fields {
        name: "node-name"
        string_value: "task-n9k-1"
```

```
      }

      fields {
        name: "time-of-day"
        string_value: "Jun 26 2019 18:20:21"
      }

      fields {
        name: "time-stamp"
        uint64_value: 1574293838000
      }

      fields {
        name: "time-zone"
        string_value: "UTC"
      }

      fields {
        name: "process-name"
        string_value: ""
      }

      fields {
        name: "category"
        string_value: "VSHD"
      }

      fields {
        name: "group"
        string_value: "VSHD"
      }

      fields {
        name: "message-name"
        string_value: "VSHD_SYSLOG_CONFIG_I"
      }

      fields {
        name: "severity"
        uint32_value: 5
      }

      fields {
        name: "text"
        string_value: "Configured from vty by admin on console0"
      }
    }
  }
}
```


テレメトリのトラブルシューティング

-

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
```

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

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。