モデル駆動型テレメトリの設定

モデル駆動型テレメトリ（MDT）は、MDT 対応デバイスから宛先へとデータをストリームするメカニズムを提供します。ストリーミングされるデータは、サブスクリプションによって定義されます。

ストリーミングされるデータは、YANG モデルのデータ セットから登録されます。登録済みのデータ セットからのデータは、設定された定期的な間隔で、またはイベントが発生した場合にのみ、宛先にストリーミングされます。この動作は、MDT がパターンベースのテレメトリまたはイベントベースのテレメトリー（EDT）用に構成されているかどうかに基づきます。

イベントベースのテレメトリの設定は、サンプル間隔が異なる点を除き、パターンベースのテレメトリに似ています。サンプル間隔の値をゼロに設定すると、イベントベースのテレメトリのサブスクリプションが設定され、間隔をゼロ以外の値に設定すると、パターンベースのテレメトリのサブスクリプションが設定されます。

MDT の設定およびモニタには、次の YANG モデルが使用されます。

- **Cisco-IOS-XR-telemetry-model-driven-cfg.yang** および **openconfig-telemetry.yang**：NETCONF または merge-config over grpc を使用して MDT を設定します。
- **Cisco-IOS-XR-telemetry-model-driven-oper.yang**：MDT に関する運用情報を取得します。

イベント駆動型テレメトリ（EDT）をサポートするノードの場合、YANG モデルには `xr:event-telemetry` というステートメントが付いています。たとえば、EDT をサポートするインターフェイスには、次の例のような注釈があります。

```
leaf interface-name {
      xr:event-telemetry "Subscribe Telemetry Event";
      type xr:Interface-name;
      description "Member's interface name";
    }
```

MDT データをストリーミングするプロセスでは、次のコンポーネントが使用されます。

- **宛先**：ストリーミングされたデータを収集する 1 つ以上の宛先を指定します。
- **センサー パス**：データをストリーミングする必要がある YANG パスを指定します。
- **サブスクリプション**：1 つ以上のセンサー パスを宛先にバインドし、データをストリーミングする基準を指定します。パターンベースのテレメトリでは、データは設定された頻度

で継続的にストリーミングされます。イベントベースのテレメトリでは、設定されたモデルの状態またはデータの変更が発生した場合にのみデータがストリーミングされます。

- **トランスポートおよびエンコーディング**：テレメトリ データの配信メカニズムを表します。

主要コンポーネントの詳細については、モデル駆動型テレメトリ ストリーミングの主要コンポーネント を参照してください。

ルータと宛先の間のテレメトリ セッションを初期化するオプションは、次の2つのモードに基づいています。

- ダイヤルアウトモード：ルータはサブスクリプションに基づいて宛先へのセッションを開始します。
- ダイヤルインモード：宛先はルータへのセッションを開始し、ストリーミングされるデータに登録します。

（注） ダイヤルイン モードは、gRPC 上でのみサポートされています。

64 ビット Linux ベースの IOS XR オペレーティング システムがサポートされています。NCS5500、NCS5000、ASR9000、NCS 560 などの 64 ビット プラットフォームは、gRPC、UDP、および TCP プロトコルをサポートしています。

モデル駆動型テレメトリ データを目的の受信者にストリーミングするには、次の作業が必要です。

- ダイヤルアウト モードの設定 （2 ページ）
- ダイヤルイン モードの設定 （8 ページ）

ダイヤルアウト モードの設定

ダイヤルアウトモードでは、ルータはサブスクリプションに基づいて宛先へのセッションを開始します。

すべての 64 ビット IOS XR プラットフォーム（NCS 6000 シリーズ ルータを除く）は、gRPC、UDP、および TCP プロトコルをサポートしています。すべての 32 ビット IOS XR プラットフォームは、TCP のみをサポートしています。

ダイヤルアウト モードの詳細については、ダイヤルアウト モード を参照してください。

ダイヤルアウト モードを設定するプロセスには、次の手順が必要です。

宛先グループの作成

宛先グループは、テレメトリ データを送信するためにルータが使用する宛先アドレス、ポート、エンコーディングおよびトランスポートを指定します。

1. 宛先アドレス、ポート、トランスポート、およびエンコード形式を識別します。
2. 宛先グループを作成します。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#destination-group <group-name>

Router(config-model-driven-dest)#address family ipv4 <IP-address> port <port-number>

Router(config-model-driven-dest-addr)#encoding <encoding-format>
Router(config-model-driven-dest-addr)#protocol <transport>
Router(config-model-driven-dest-addr)#commit
```

例：TCP ダイヤルアウトの宛先グループ

次の例は、キー値 Google プロトコル バッファ（自己記述 gpb とも呼ばれる）エンコーディングを使用してTCPダイヤルアウト設定用に作成された宛先グループ `DGroup1` を示しています。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#destination-group DGroup1
Router(config-model-driven-dest)#address family ipv4 172.0.0.0 port 5432
Router(config-model-driven-dest-addr)#encoding self-describing-gpb
Router(config-model-driven-dest-addr)#protocol tcp
Router(config-model-driven-dest-addr)#commit
```

例：UDP ダイヤルアウトの宛先グループ

次の例は、キー値 Google プロトコル バッファ（自己記述 gpb とも呼ばれる）エンコーディングを使用してUDPダイヤルアウト設定用に作成された宛先グループ `DGroup1` を示しています。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#destination-group DGroup1
Router(config-model-driven-dest)#address family ipv4 172.0.0.0 port 5432
Router(config-model-driven-dest-addr)#encoding self-describing-gpb
Router(config-model-driven-dest-addr)#protocol udp
Router(config-model-driven-dest-addr)#commit
```

UDPがコネクションレス型であるため、UDP宛先はコレクタの状態に関係なく `Active` として表示されます。

UDPを使用したモデル駆動型テレメトリは、ビジーなネットワークには適していません。メッセージがコレクタに到達する前にネットワークによって廃棄された場合、再試行は行われません。

センサー グループの作成

センサーグループは、ストリーミングされる YANG モデルのリストを指定します。

1. XR YANG モデルのセンサー パスを識別します。
2. センサー グループを作成します。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group <group-name>
```

```
Router(config-model-driven-snsr-grp)# sensor-path <XR YANG model>
Router(config-model-driven-snsr-grp)# commit
```

例：ダイヤルアウト用センサー グループ

（注） gRPC は、64 ビット プラットフォームでのみサポートされています。

次の例は、インターフェイス統計情報に関して YANG モデルによるダイヤルアウト設定用に作成されたセンサー グループ `SGroup1` を示しています。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group SGroup1
Router(config-model-driven-snsr-grp)# sensor-path
Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters
Router(config-model-driven-snsr-grp)# commit
```

次の作業：

サブスクリプションを作成します。

サブスクリプションの作成

サブスクリプションは、宛先グループをセンサーグループに関連付け、ストリーミング方式（パターンベースまたはイベントベースのテレメトリ）を設定します。

サブスクリプション グループ内の送信元インターフェイスは、セッションを確立して宛先にデータをストリーミングするために使用されるインターフェイスを指定します。VRF と送信元インターフェイスの両方が設定されている場合、送信元インターフェイスは、確立するセッションの宛先グループで指定されたものと同じ VRF になければなりません。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription <subscription-name>
Router(config-model-driven-subs)#sensor-group-id <sensor-group> sample-interval <interval>

Router(config-model-driven-subs)#destination-id <destination-group>
Router(config-model-driven-subs)#source-interface <source-interface>
Router(config-mdt-subscription)#commit
```

例：パターンベースのダイヤルアウト設定のサブスクリプション

次の例は、センサーグループと宛先グループを関連付け、データをストリーミングする間隔を 30 秒に設定するよう作成されたサブスクリプション `Sub1` を示しています。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription Sub1
Router(config-model-driven-subs)#sensor-group-id SGroup1 sample-interval 30000
Router(config-model-driven-subs)#destination-id DGroup1
Router(config-mdt-subscription)# commit
```

例：イベントベースのダイヤルアウト設定のサブスクリプション

次の例は、センサーグループと宛先グループを関連付け、データをストリーミングするためのイベントベースの方式を設定するよう作成されたサブスクリプション `Sub1` を示しています。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription Sub1
Router(config-model-driven-subs)#sensor-group-id SGroup1 sample-interval 0
Router(config-model-driven-subs)#destination-id DGroup1
Router(config-mdt-subscription)# commit
```

例：インターフェイス パスのイベント駆動型テレメトリの設定

```
telemetry model-driven
 destination-group 1
  address family ipv4 <ip-address> port <port-number>
   encoding self-describing-gpb
   protocol grpc no-tls
  !
 !
 sensor-group 1
  sensor-path
Cisco-IOS-XR-ipv6-ma-oper:ipv6-network/nodes/node/interface-data/vrfs/vrf/global-briefs/global-brief

 !
 sensor-group 2
  sensor-path Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface
 !
 subscription 1
  sensor-group-id 1 sample-interval 0
  sensor-group-id 2 sample-interval 0
  destination-id 1
!
```

次の作業：

設定を検証します。

ダイヤルアウト設定の検証

次のコマンドを使用して、ダイヤルアウト用にルータを正しく設定していることを確認します。

```
Router#show telemetry model-driven subscription <subscription-group-name>
```

例：TCP ダイヤルアウトの検証

```
Router#show telemetry model-driven subscription Sub1
Thu Jul 21 15:42:27.751 UTC
Subscription:  Sub1                     State: ACTIVE
-------------
  Sensor groups:
  Id                Interval(ms)        State
  SGroup1           30000               Resolved

  Destination Groups:
```

```
Id                  Encoding            Transport   State   Port    IP
DGroup1             self-describing-gpb tcp         Active  5432    172.0.0.0
```

例：LLDP 用イベント駆動型テレメトリの設定

テレメトリは、NETCONF クライアントがサブスクリプションを通じて NETCONF サーバからイベント通知を受信するように設定されている NETCONF イベント通知をサポートします。NETCONF クライアントは、`create-subscription` 要求を使用して登録する必要があります。現在は、Link Layer Discovery Protocol（LLDP）からのイベントのみがサポートされています。これらのイベント通知は、NETCONF セッションまたはサブスクリプションが終了するまで送信されます。

（注） NETCONF 通知を受け取るためにセンサー グループとサブスクリプションを設定する必要はありません。テレメトリ イベントを受信するにはセンサー パスとサブスクリプションの設定が必要ですが、NETCONF 通知を受信するには NETCONF `create-subscription` が必要です。

NETCONF 通知を作成するには、次の手順を実行します。

1. NETCONF エージェントと SSH サブシステムを有効にします。

```
ssh server netconf
netconf-yang agent ssh
```

2. モデル駆動型テレメトリを有効にします。

```
telemetry model-driven
```

3. LLDP をイネーブルにします。

```
lldp
```

次の例に、LLDP 設定データ用のイベント駆動型テレメトリを示します。

1. 宛先グループを作成します。

```
grpc
port 56782
address-family ipv4
!
telemetry model-driven
destination-group <destination-udp>
  address-family ipv4 <client-ip>1 port <udp port num>
   encoding self-describing-gpb
   protocol udp
  !
!
destination-group <destination-tcp>
  address-family ipv4 <client-ip> port <tcp port num>
   encoding gpb
   protocol tcp
  !
destination-group <destination-grpc>
  address-family ipv4 <grpc client ip>port <grpc port num>
```

```
   encoding self-describing-gpb
   protocol grpc no-tls
```

2. センサー グループを作成します。

```
sensor-group <sensor-group-name>
  sensor-path Cisco-IOS-XR-ethernet-lldp-oper:lldp/global-lldp/lldp-info
  sensor-path Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/interfaces/interface

  sensor-path Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/details/detail

 !
```

3. サブスクリプションを作成します。

```
subscription udp-out
  sensor-group-id <sensor-group-name> sample-interval 0
  destination-id <destination-udp>
!

subscription <subscription-name>
  sensor-group-id <sensor-group-name> sample-interval 0
  destination-id <destination-tcp>

subscription <subscription-name>
  sensor-group-id <sensor-group-name> sample-interval 0
!
netconf-yang agent
ssh
!
```

4. イベントが発生したときにデータをストリーミングする通知を設定します。

```
Router(config-lldp)#timer 12
Router(config-lldp)#commit

Router(config-lldp)#holdtime 150
Router (config-lldp)#commit

Router (config-lldp)#exit
#506
<?xml version="1.0"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <eventTime>Date-and-Time</eventTime>
 <lldp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ethernet-lldp-oper">
  <global-lldp>
   <lldp-info>
    <chassis-id>000b.1bc9.e700</chassis-id>
    <chassis-id-sub-type>4</chassis-id-sub-type>
    <system-name>ios</system-name>
    <timer>12</timer>
    <hold-time>120</hold-time>
    <re-init>2</re-init>
   </lldp-info>
  </global-lldp>
 </lldp>
</notification>
Ready to send a request.
Paste your request or enter 'get', 'get-config', 'create-sub', or 'bye' to quit):
```

5. NETCONF エージェントから受信した応答を検証します。

```
#506
<?xml version="1.0"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <eventTime>Date-and-Time</eventTime>
 <lldp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ethernet-lldp-oper">
  <global-lldp>
   <lldp-info>
    <chassis-id>000b.1bc9.e700</chassis-id>
    <chassis-id-sub-type>4</chassis-id-sub-type>
    <system-name>ios</system-name>
    <timer>12</timer>
    <hold-time>150</hold-time>
    <re-init>2</re-init>
   </lldp-info>
  </global-lldp>
 </lldp>
</notification>
```

ダイヤルイン モードの設定

ダイヤルイン モードでは、宛先はルータへのセッションを開始し、ストリーミングされるデータに登録します。

（注） ダイヤルイン モードは、64 ビット プラットフォームでのみ gRPC 経由でサポートされています。

ダイヤルイン モードの詳細については、ダイヤルイン モード を参照してください。

ダイヤルイン モードを設定するプロセスには、次のタスクが含まれます。

- gRPC の有効化
- センサー グループの作成
- サブスクリプションの作成
- 設定の検証

gRPC の有効化

ルータの gRPC サーバをコレクタからの着信接続を受け入れるように設定します。

1. HTTP/2 接続で gRPC を有効にします。

```
Router# configure
Router (config)# grpc
```

2. 指定されたポート番号へのアクセスを有効にします。

```
Router (config-grpc)# port <port-number>
```

<port-number> の範囲は 57344 ～ 57999 です。ポート番号が使用できない場合は、エラーが表示されます。

（注） gRPC サーバが設定されている場合、ルータへのサブスクリプションを要求するための Google ネットワーク管理インターフェイス（gNMI）クライアントのサポートが有効になります。ユーザは、関心のある oper パスにサブスクライブしてデータをストリーミングできます。

3. コンフィギュレーション モードでセッションパラメータを設定します。

```
Router (config)# grpc{ address-family | dscp | max-request-per-user | max-request-total
 | max-streams | max-streams-per-user | no-tls | service-layer | tls-cipher |
tls-mutual | tls-trustpoint | vrf }
```

値は次のとおりです。

- **address-family**：アドレス ファミリ識別子タイプを設定します
- **dscp**：送信された gRPC で QoS マーキング DSCP を設定します
- **max-request-per-user**：ユーザあたりの同時要求の最大数を設定します
- **max-request-total**：合計同時要求の最大数を設定します
- **max-streams**：同時 gRPC 要求の最大数を設定します。サブスクリプションの上限は 128 要求です。デフォルトは 32 要求です
- **max-streams-per-user**：ユーザあたりの同時 gRPC 要求の最大数を設定します。サブスクリプションの上限は 128 要求です。デフォルトは 32 要求です
- **no-tls**：トランスポート レイヤ セキュリティ（TLS）を無効化します。TLS はデフォルトで有効になっています。
- **service-layer**：gRPC サービス レイヤの設定を有効にします
- **tls-cipher**：gRPC TLS 暗号スイートを有効にします
- **tls-mutual**：相互認証を設定します
- **tls-trustpoint**：トラストポイントを設定します
- **vrf**：サーバ VRF を有効にします

4. TLS 暗号を設定します。

```
Router(config-grpc)#tls-cipher default <enable | disable>
Router(config-grpc)#tls-cipher disable <cipher1, cipher2, ...>
Router(config-grpc)#tls-cipher enable <cipher1, cipher2, ...>
```

5. 設定をコミットします。

```
Router(config-grpc)#commit
```

（注）

- **tls-cipher default enable** および **tls-cipher disable <all the ciphers>** が設定されている場合、1 つではなくすべての暗号が有効になります。
- **tls-cipher default disable** および **tls-cipher enable <>** が設定されていない場合、1 つではなくすべての暗号が有効になります。
- tls-cipher の設定が変更されると、gRPC サーバの停止と再起動が発生する可能性があります。

例：ルータでの gRPC および TLS の有効化

次に、`show grpc` コマンドの出力例を示します。出力例には、ルータで TLS が有効になっている場合の gRPC 設定が表示されています。

```
Router#show grpc

Address family              : ipv4
Port                        : 57300
VRF                         : global-vrf
TLS                         : enabled
TLS mutual                  : disabled
Trustpoint                  : none
Maximum requests            : 128
Maximum requests per user   : 10
Maximum streams             : 32
Maximum streams per user    : 32

TLS cipher suites
  Default                   : none
  Enable                    : none
  Disable                   : none

  Operational enable        : ecdhe-rsa-chacha20-poly1305
                            : ecdhe-ecdsa-chacha20-poly1305
                            : ecdhe-rsa-aes128-gcm-sha256
                            : ecdhe-ecdsa-aes128-gcm-sha256
                            : ecdhe-rsa-aes256-gcm-sha384
                            : ecdhe-ecdsa-aes256-gcm-sha384
                            : ecdhe-rsa-aes128-sha
                            : ecdhe-ecdsa-aes128-sha
                            : ecdhe-rsa-aes256-sha
                            : ecdhe-ecdsa-aes256-sha
                            : aes128-gcm-sha256
                            : aes256-gcm-sha384
                            : aes128-sha
                            : aes256-sha
  Operational disable       : none
```

次の作業：

センサー グループを作成します。

センサー グループの作成

センサー グループは、ストリーミングされる YANG モデルのリストを指定します。

1. XR YANG モデルのセンサー パスを識別します。
2. センサー グループを作成します。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group <group-name>
Router(config-model-driven-snsr-grp)# sensor-path <XR YANG model>
Router(config-model-driven-snsr-grp)# commit
```

例：gRPC ダイヤルインのセンサー グループ

次の例は、インターフェイスに対し YANG モデルを使用して gRPC ダイヤルイン設定用に作成されたセンサー グループ `SGroup3` を示しています。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group SGroup3
Router(config-model-driven-snsr-grp)# sensor-path
openconfig-interfaces:interfaces/interface
Router(config-model-driven-snsr-grp)# commit
```

次の作業：

サブスクリプションを作成します。

サブスクリプションの作成

サブスクリプションは、センサーグループをストリーミング間隔に関連付けます。コレクタは、ルータとの接続を確立するときにセンサー パスへのサブスクリプションを要求します。

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription <subscription-name>
Router(config-model-driven-subs)#sensor-group-id <sensor-group> sample-interval <interval>

Router(config-model-driven-subs)#destination-id <destination-group>
Router(config-mdt-subscription)#commit
```

例：gRPC ダイヤルインのサブスクリプション

次の例は、センサーグループを 30 秒間隔のデータ ストリーミングに関連付けるために作成されたサブスクリプション `Sub3` を示しています。

```
Router(config)telemetry model-driven
Router(config-model-driven)#subscription Sub3
Router(config-model-driven-subs)#sensor-group-id SGroup3 sample-interval 30000
Router(config-mdt-subscription)#commit
```

次の作業：

設定を検証します。

ダイヤルイン設定の検証

次のコマンドを使用して、gRPC ダイヤルイン用にルータを正しく設定していることを確認します。

```
Router#show telemetry model-driven subscription
```

例：gRPC ダイヤルインの検証

```
RP/0/RP0/CPU0:SunC#show telemetry model-driven subscription Sub3
Thu Jul 21 21:32:45.365 UTC
Subscription:  Sub3
-------------
  State:       ACTIVE
  Sensor groups:
  Id: SGroup3
    Sample Interval:      30000 ms
    Sensor Path:          openconfig-interfaces:interfaces/interface
    Sensor Path State:    Resolved

  Destination Groups:
  Group Id: DialIn_1002
    Destination IP:       172.30.8.4
    Destination Port:     44841
    Encoding:             self-describing-gpb
    Transport:            dialin
    State:                Active
    Total bytes sent:     13909
    Total packets sent:   14
    Last Sent time:       2016-07-21 21:32:25.231964501 +0000

  Collection Groups:
  ------------------
    Id: 2
    Sample Interval:      30000 ms
    Encoding:             self-describing-gpb
    Num of collection:    7
    Collection time:      Min:    32 ms Max:    39 ms
    Total time:           Min:    34 ms Avg:    37 ms Max:    40 ms
    Total Deferred:       0
    Total Send Errors:    0
    Total Send Drops:     0
    Total Other Errors:   0
    Last Collection Start:2016-07-21 21:32:25.231930501 +0000
    Last Collection End:  2016-07-21 21:32:25.231969501 +0000
    Sensor Path:          openconfig-interfaces:interfaces/interface
```