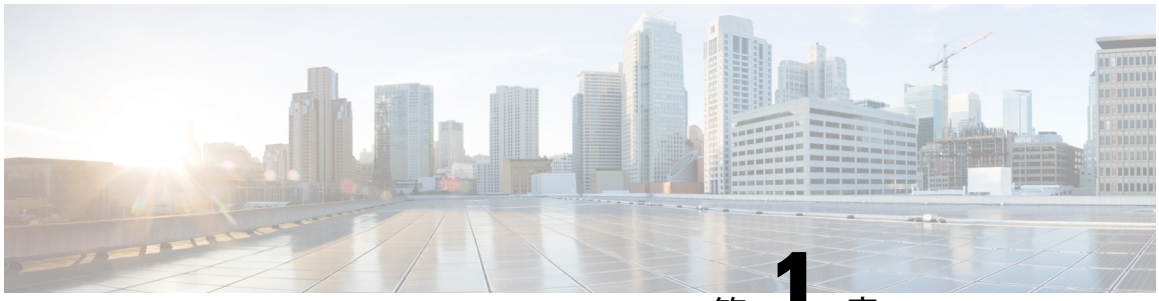
Cisco NCS 560 シリーズ ルータ（IOS XR リリース 7.1.x）テレメトリ コンフィギュレーション ガイド

初版：2020 年 1 月 29 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

目 次



第 1 章

新機能および変更機能に関する情報

この項では、『*Cisco NCS 560* シリーズ ルータ テレメトリ コンフィギュレーション ガイド』に関連する新機能と変更された機能をすべて示します。

• テレメトリの新機能および変更された機能 （1 ページ）

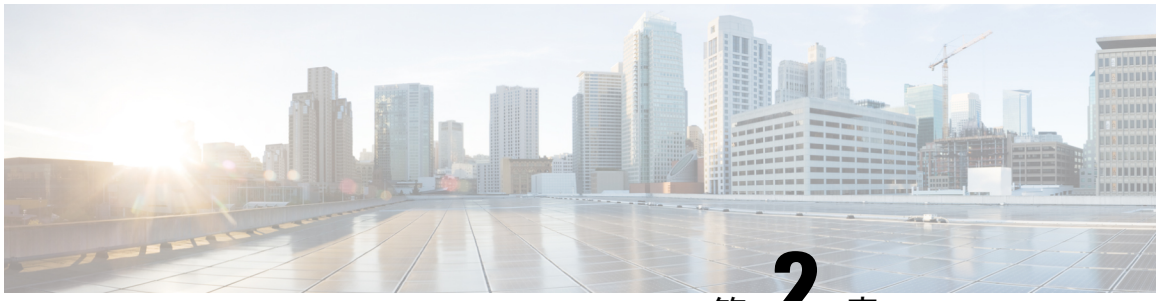
テレメトリの新機能および変更された機能

機能	説明	変更が行われたリリース	参照先
openconfig-platform データモデルを使用したテレメトリデータのストリーミング	openconfig-platform データモデルを使用して、デバイスの動作状態や設定といった、デバイスの基盤となる特性に関するデータをストリーミングできます。	リリース 7.1.1	このデータ モデルは、Github リポジトリから入手してください。

機能	説明	変更が行われたリリース	参照先

テレメトリの輻輳制御	テレメトリの輻輳管理を行うためのサポート。
	輻輳制御を使用すると、宛先ごとに最大4000の未処理メッセージが許可されます。未処理のメッセージが 3000 を超えると、イベントがスロットリングされます。未処理のメッセージが 250 を超えると、連続メッセージのスロットリングが発生します。イベントは、連続メッセージよりも優先順位が高くなります。
	次に、出力例を示します。
	`Router#show telemetry model-driven destination`
	`DialIn_1002 1 192.x.x.x`
	`19687 self-describing-gpb dialin Active`
	`TLS: False`
	`Collection statistics:`
	`Maximum tokens : 4000`
	`Event tokens : 750`
	`Cadence tokens : 723`
	`Token processed at : <time-stamp>`
	`Cadence token advertised at : <time-stamp>`
	`Event token advertised at : >time-stamp>`
	`GNMI initial synchronization time:`
	`Pending queue size : 0`
	`Processed events : 0`
	`Collection tokens : 723`
		リリース 7.1.1	NA

機能	説明	変更が行われたリリース	参照先
スレッドレベルでのCPU使用率に関する情報を取得するためのサポート	Cisco-IOS-XR-wdsysmon-fd-oper.yang データモデルが拡張され、実行中のプロセスごとにスレッドレベルの CPU 使用率が含められました。 次にプロセスの出力例を示します。 <process-cpu> <process-name>tam_sync</process-name> <process-id>5062</process-id> <process-cpu-one-minute>0</process-cpu-one-minute> <process-cpu-five-minute>0</process-cpu-five-minute> <process-cpu-fifteen-minute>0</process-cpu-fifteen-minute> <thread-cpu> <thread-name>lwm_service_thr</thread-name> <thread-id>5063</thread-id> <process-cpu-one-minute>0</process-cpu-one-minute> <process-cpu-five-minute>0</process-cpu-five-minute> <process-cpu-fifteen-minute>0</process-cpu-fifteen-minute> </thread-cpu> </process-cpu>	リリース 7.1.1	NA
プロセススレッドに関する情報を取得するためのサポート	Cisco-IOS-XR-procthreadname-oper.yang データモデルは、スレッド名、優先度、状態、実行中のプロセスのスタックサイズなどのスレッドレベルの詳細を照会するのに役立ちます。 次に、出力例を示します。 <thread> <name>qsm_service_thr</name> <state>Sleeping</state> <stack>0K</stack> <pri>20</pri> <rtpri>0</rtpri> <jid>69381</jid> <tid>3866</tid> </thread>	リリース 7.1.1	NA



第 2 章

テレメトリを使用したネットワーク モニタリング戦略の拡張

SNMP、Syslog、CLI などの従来のポーリング方式を使用してネットワークをモニタしていますか。その場合、ネットワークから抽出したデータは、次の質問に答えるために役立ちますか。

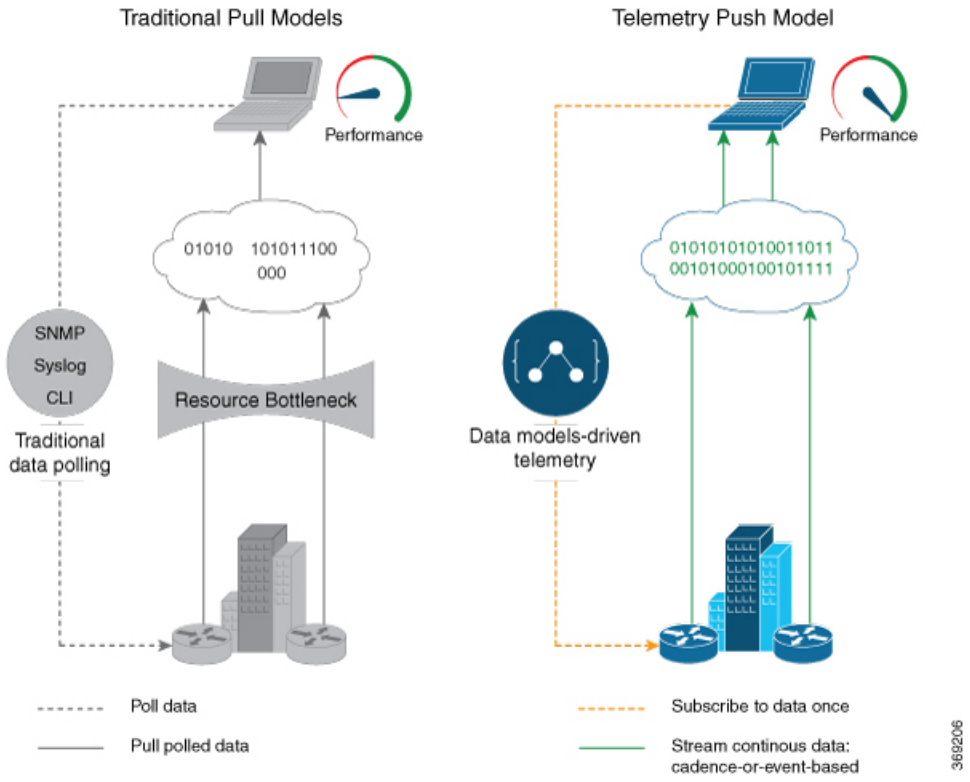
- ネットワーク トラフィックは現在、ネットワーク帯域幅の何パーセントを消費していますか。
- ネットワーク内のすべてのリンクは 100% の使用率で実行されますか。
- 無人ルータに障害が発生した場合、問題と関連する影響はネットワーク オペレータにはリアルタイムで通知されますか。
- CPU 使用率が過大または過小ではありませんか。
- トラフィックおよびデータ損失に基づいてネットワークの効率性を計算できますか。
- トラフィック損失またはネットワーク遅延の原因となるパフォーマンスの問題として、どのようなものが考えられますか。
- 発生する可能性のある問題をどのようにしてプロアクティブに防止しますか。データはネットワーク パターンの調査をリアルタイムでサポートしていますか。

これらの従来の方式では、情報を一定の間隔で要求するプルモデルが使用されています。収集したデータは、管理可能なサイズのネットワークを効率的にモニタするのに役立ちます。ただし、ネットワークがより複雑かつ大規模になるにつれて、ポーリングするデータが効率的かつ効果的なモニタリングを行うためには不十分となる可能性があります。さらに、ポーリング方式ではリソースが大量に消費されるほか、ネットワーク オペレータが収集された情報におけるギャップに直面することになります。プルモデルでは、ネットワーク デバイス（サーバ）は、データ コレクタ（クライアント）が要求した場合にのみデータを送信します。このような要求を開始するには、手動による介入が継続的に必要となります。この手動による介入のため、このモデルは適切ではなく、自動化と拡張性も制限されます。ネットワークの可視性が制限されるため、ネットワークを効率的に制御することができません。ネットワークにさらなる復元力と安定性を与えるモニタリング戦略が必要です。

テレメトリでは、まさにそれが実現されます。テレメトリでは、ネットワーク デバイスからデータを自動的にストリーミングするプッシュ モデルが使用されます。コレクタが一定の間隔でデータを要求するのではなく、ネットワーク デバイスが運用データをリアルタイムでストリーミングします。

テレメトリは、規模、速度、および自動化によって実現される機能に重点を置いています。柔軟性によって実現される機能により、関心のあるデータをルータから選択し、そのデータをモニタリングのため、構造化された形式でリモート管理ステーションに送信することができます。テレメトリを通じて頻繁に得られるきめ細かいデータを利用することで、組織内のDevOps（開発および運用）エンジニアは、問題が発生したらすぐにそれを特定し、調査することができます。これにより、ネットワークのモニタリングおよびより適切な管理を協力して行うことができるようになります。

次の図は、テレメトリ プッシュ モデルを使用したテレメトリ データのストリーミングの利点を、従来のプル モデルと比較して示します。プル モデルでは、リソースのボトルネックが発生し、ルータから貴重な運用データが取得できなくなることがあります。他方、プッシュ モデルは、このようなボトルネックを排除し、データが効率的に提供されるように設計されています。



テレメトリ データによってどのようにネットワーク内のインテリジェンスが利用可能になり、問題をプロアクティブに予測してトラブルシューティングすることが可能になるかについては、こちらのビデオを視聴してください。

この記事では、テレメトリ データを使用する利点と、ネットワーク デバイスから意味のあるデータをストリーミングするためのさまざまな方法について説明します。

ネットワークモニタリングをプルモデルからテレメトリプッシュ モデルに移行することの利点

リアルタイムのテレメトリ データは、次の場合に役に立ちます。

- **ネットワークのリモート管理**：テレメトリの主な利点は、エンドユーザがネットワーク要素の状態をリモートでモニタできるようになることです。ネットワークの展開後は、ネットワーク サイトに駐在して、何が役立っているのか、何が煩雑となっているのか特定することはできません。テレメトリを使用すると、これらの分析情報をリモート ロケーションから分析および活用し、実行することができます。

- **トラフィックの最適化**：ネットワーク内でのリンク使用率やパケットのドロップ数を頻繁にモニタリングすると、リンクの追加や削除、トラフィックのリダイレクト、ポリシングの変更などを簡単に行えます。高速再ルーティングのようなテクノロジーにより、ネットワークは新しいパスに切り替わり、従来の SNMP ポーリング インターバル メカニズムよりも迅速に再ルーティングできます。テレメトリ データのストリーミングは、トラフィックの高速転送への応答時間を短縮するのに役立ちます。

- **予防的なトラブルシューティング**：ネットワーク状態インジケータ、ネットワーク統計情報、および重要なインフラストラクチャ情報がアプリケーション レイヤに公開され、操作性能の向上やトラブルシューティング時間の短縮に使用されます。テレメトリを通じて頻繁に得られるきめ細かいデータが、パフォーマンスのモニタリングを向上させ、それによって優れたトラブルシューティングが行えます。

- **データの可視化**：テレメトリ データは、ネットワーク展開に関する貴重な分析情報を可視化するために分析ツール チェーンおよびアプリケーションが使用するデータ レイクとして機能します。

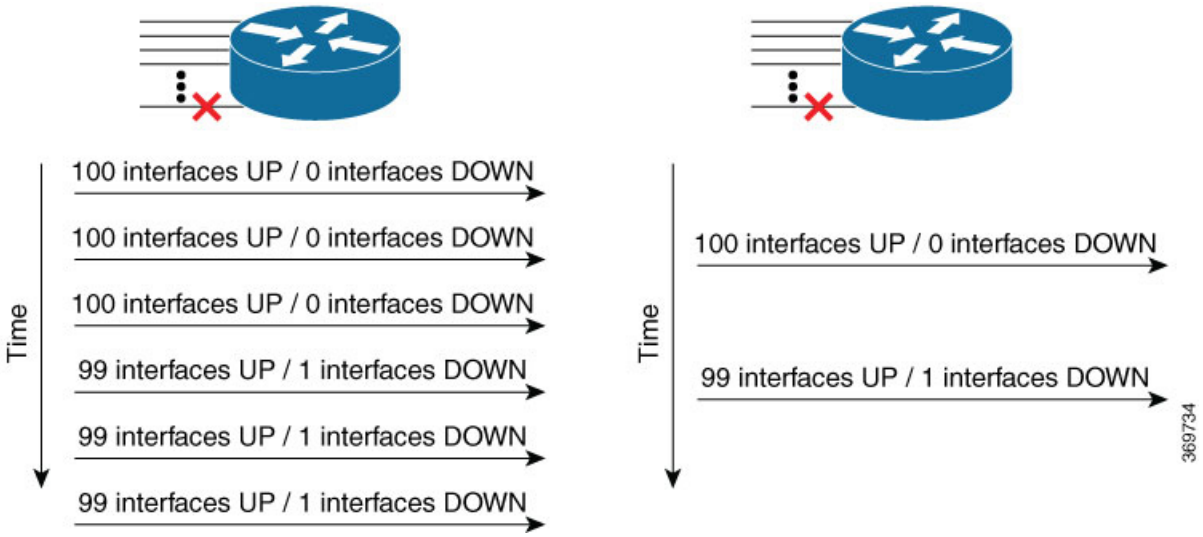
- **分散デバイスのモニタリングと制御**：モニタリング機能は、ストレージおよび分析の機能から分離されています。この分離により、デバイスの依存関係を減らせるほか、パイプラインを使用してデータを柔軟に変換できるようになります。これらのパイプラインは、テレメトリ データを消費および変換し、結果として生成されたコンテンツを（通常は既成の）ダウンストリーム コンシューマに転送するユーティリティです。サポートされているダウンストリーム コンシューマは、Apache Kafka、Influxdata、Prometheus、および Grafana などです。

したがって、テレメトリのストリーミングにより、モニタリングプロセスは、大規模なデータセットを高速で抽出および分析することを可能にし、よりよい意思決定を可能にするビッグデータ計画へと転換されます。

ルータから宛先にテレメトリデータをストリーミングするためのメカニズムの確認

テレメトリデータは、パターン駆動型またはイベント駆動型のいずれかのメカニズムを使用してストリーミングできます。

図 *1:* パターン駆動型テレメトリとイベント駆動型テレメトリ

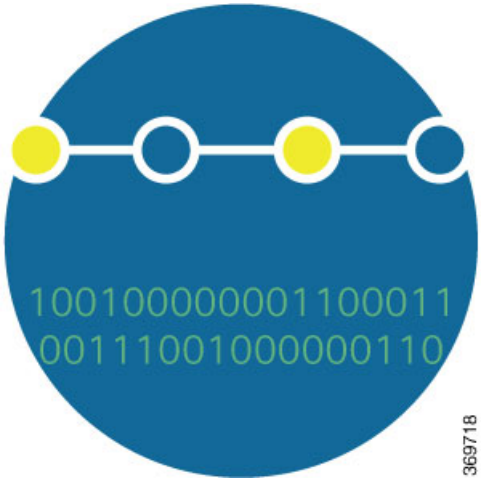


パターン駆動型テレメトリ

パターン駆動型のテレメトリは、データ（運用統計および状態遷移）を設定されたパターンで継続的にストリーミングします。継続的にストリーミングされるデータの頻度が高いほど、ネットワーク内の新たなパターンを厳密に特定するのに役立ちます。

次の図は、設定された時間間隔での継続的なデータ ストリームを示しています。

図 *2:* パターン駆動型テレメトリ

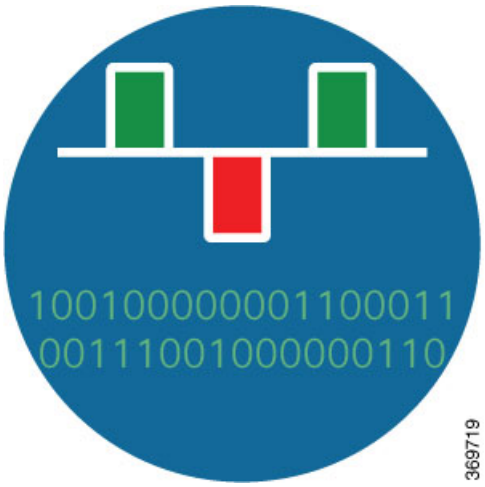


イベント駆動型テレメトリ

イベント駆動型テレメトリは、受信側で収集されたデータを最適化し、状態遷移が発生した場合にのみデータをストリーミングするため、収集されたデータは受信側で最適化されます。たとえば、EDT は、インターフェイスの状態遷移や IP ルートの更新などに関するデータをストリーミングします。

次の図は、状態が変化した後のデータ ストリームを示しています。

図 *3:* イベント駆動型テレメトリ



テレメトリデータのストリーミングを可能にする要素について

次の要素は、ネットワーク内のテレメトリを有効にするための構成要素です。

センサー パス

センサー パスは、コンテナを含む YANG データ モデル内の YANG パスまたはデータ定義のサブセットを記述します。YANGモデルでは、コンテナ階層内の任意のレベルで終了するようにセンサー パスを指定できます。

YANG モジュールは、ルータのデータを介してデータ モデルを定義し、そのデータに対する階層的な組織と制約を定義します。

YANG は 4 つのノード タイプを定義します。各ノードには名前があります。ノード タイプに応じて、ノードは値を定義するか、一連の子ノードを含めます。データ モデリングの場合、次のノード タイプがあります。

- リーフ ノード：特定のタイプの単一の値が含まれています。
- リーフリスト ノード：一連のリーフ ノードが含まれています。
- リスト ノード：一連のリーフリスト エントリが含まれています。リーフリスト エントリのそれぞれは 1 つ以上のキー リーフによって一意に識別されます。
- コンテナ ノード：子ノードのみを含む関連ノードのグループが含まれます。子ノードは4つのノード タイプのいずれかです。

データ モデルの詳細については、『*Programmability Configuration Guide for Cisco Series Routers*』を参照してください。

次の表に、センサー パスの例をいくつか示します。

表 *1*：センサー パス

機能	センサー パス
CPU	`Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization`
メモリ	`Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary`

インターフェイス	`Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters`
	`Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/data-rate`
	`openconfig-interfaces:interfaces/interface`

光出力レベル	`Cisco-IOS-XR-dwdm-ui-oper:dwdm/ports/port/info/optics-info`
ノード サマリー	`Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary`

機能	センサー パス
転送情報ベース（FIB）	Cisco-IOS-XR-fib-common-oper:fib-statistics/nodes/node/drops Cisco-IOS-XR-fib-common-oper:fib/nodes/node/protocols/protocol/vrfs/vrf/summary
MPLS トラフィック エンジニアリング（MPLS-TE）	Cisco-IOS-XR-mpls-te-oper:mpls-te/tunnels/summary Cisco-IOS-XR-ip-rsvp-oper:rsvp/interface-briefs/interface-brief Cisco-IOS-XR-mpls-te-oper:mpls-te/fast-reroute/protections/protection Cisco-IOS-XR-mpls-te-oper:mpls-te/signalling-counters/signalling-summary Cisco-IOS-XR-mpls-te-oper:mpls-te/p2p-p2mp-tunnel/tunnel-heads/tunnel-head
MPLS ラベル配布プロトコル（MPLS-LDP）	Cisco-IOS-XR-mpls-ldp-oper:mpls-ldp/nodes/node/bindings-summary-all Cisco-IOS-XR-mpls-ldp-oper:mpls-ldp/global/active/default-vrf/summary Cisco-IOS-XR-mpls-ldp-oper:mpls-ldp/nodes/node/default-vrf/neighbors/neighbor
ルーティング	Cisco-IOS-XR-clns-isis-oper:isis/instances/instance/statistics-global Cisco-IOS-XR-clns-isis-oper:isis/instances/instance/neighbors/neighbor Cisco-IOS-XR-ip-rib-ipv4-oper:rib/rib-table-ids/rib-table-id/summary-protos/summary-proto Cisco-IOS-XR-clns-isis-oper:isis/instances/instance/levels/level/adjacencies/adjacency Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/process-info Cisco-IOS-XR-ip-rib-ipv6-oper:ipv6-rib/rib-table-ids/rib-table-id/summary-protos/summary-proto

（注） 特定のパスを使用して、目的に合わない可能性のあるデータがストリーミングされないようにします。たとえば、MPLS TEの概要のみに関する情報をストリーミングする場合は、`sensor-path Cisco-IOS-XR-mpls-te-oper:mpls-te` センサーパスではなく、`sensor-path Cisco-IOS-XR-mpls-te-oper:mpls-te/autotunnel/mesh/summary` を使用します。

ルータなどのMDT対応デバイスは、センサーパスをモデル内の最も近いコンテナパスに関連付けます。ルータは、単一のテレメトリ メッセージ内でコンテナ パスをエンコードしてストリーミングします。受信者は、このコンテナ パス以下のすべてのコンテナおよびリーフ ノードに関するデータを受信します。ルータは、1つ以上のセンサーパスのテレメトリ データを、設定された頻度で（パターン駆動型テレメトリ （8ページ））またはイベントの発生時に（イベント駆動型テレメトリ （9ページ））サブスクライブされたセッションを通じて1つ以上のコレクタにストリーミングします。

サブスクリプション

サブスクリプションは、1つ以上のセンサー パスと宛先をバインドします。

コレクタは、このサブスクリプションを使用して、ルータ上のデータの状態に関する最新情報を受信します。サブスクリプションは、1つ以上のセンサーパスで構成できます。サブスクラ

イブしたパスのデータについてストリーミングが開始し、セッションがコレクタによって終了されるか、またはテレメトリのサブスクリプション設定が削除されてサブスクリプションがキャンセルされるまで続きます。

エンコーダ

ルータからストリーミングされるデータは、次のいずれかの形式を使用してエンコードできます。

- **GPB エンコーディング**：GPB エンコーディングを設定するには、コンパイルされた `.proto` ファイル形式のメタデータが必要です。`.proto` ファイルはデータのストリーミングに使用する GPB メッセージ形式を記述します。`.proto` ファイルは、Github の [Cisco Network Telemetry Proto] で入手できます。
  - **コンパクト GPB エンコーディング**：データは、圧縮された形式で、かつ自己記述形式ではない形式でストリーミングされます。ストリーミングされたデータを復号化するには、コレクタは各センサーパスに対応する `.proto` ファイルを使用する必要があります。
  - **自己記述 GPB エンコーディング**：各センサー パスについてストリーミングされるデータは、自己記述形式の ASCII テキスト形式です。センサー パスのデータの復号化には、コレクタにより、単一の `.proto` ファイル `telemetry.proto` が使用されます。自己記述 GPB エンコーディングは、メッセージサイズが大きい場合でも、単一の `.proto` ファイルでどのセンサーパスのデータも復号化できるため、管理が容易です。
- **JSON エンコーディング**：データはキーの文字列でストリーミングされ、その値は人間に読める形式です。

トランスポート

テレメトリ プッシュ モデルでは、ルータはトランスポート プロトコルを使用してテレメトリ データをストリーミングします。生成されたデータは、エンコーダを使用して目的の形式にカプセル化されます。

モデル駆動型テレメトリ（MDT）データは、次のサポートされている次のトランスポート プロトコルを通じてストリーミングされます。

- Google プロトコル RPC（gRPC）：ダイヤルイン モードとダイヤルアウト モードの両方に使用されます。
- Transmission Control Protocol（TCP）：ダイヤルアウト モードにのみ使用されます。
- User Datagram Protocol（UDP）：ダイヤルアウト モードにのみ使用されます。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。これは、ビジーネットワークには最適ではありません。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、プロトコルはデータを再送信しません。

TLS 認証

gRPC プロトコルは、データを暗号化するための Transport Layer Security（TLS）をサポートしています。モデル駆動型テレメトリは、デフォルトでTLSを使用してダイヤルアウトします。

TLS が有効になっている場合、サーバはコレクタに対して認証するための証明書を送信します。コレクタは、どの認証局が署名したかを確認することで証明書を検証し、セッションを暗号化するためのセッション キーを生成します。

TLS 証明書は `/misc/config/grpc/dialout/` パスにコピーする必要があります。`protocol grpc` コマンドのみが設定されている場合、TLSはデフォルトで有効になり、ホスト名はデフォルトで宛先の IP アドレスになります。また、証明書では、Common Name（CN）を `protocol grpc tls-hostname <>` と設定する必要があります。

次の出力は、gRPC がダイヤルアウト セッションを確立するために使用する証明書を示します。

```
Router#run
[node:]$ls -l /misc/config/grpc/dialout/
total 4
-rw-r--r-- 1 root root 4017 dialout.pem
```

TLS オプションをバイパスするには、**protocol grpc no-tls** コマンドを使用します。

テレメトリセッションを確立する方法について確認する

テレメトリ セッションは、ダイヤルアウト モードまたはダイヤルイン モードのいずれかを使用して開始できます。テレメトリセッションを確立するためのモードは異なりますが、どちらのモードも同じデータ モデルを使用し、同じデータをストリーミングします。

ダイヤルアウト モード

ダイヤルアウトモードでは、ルータが受信者にダイヤルアウトし、サブスクリプションベースのテレメトリ セッションを確立します。ルータが接続を開始するため、着信トラフィックのポートを管理する必要はありません。このデフォルトの動作モードでは、セッションを確立するために使用するプロトコルにより、単純性（TCP）またはセキュリティ（gRPC）を柔軟に選択できます。単純なプロトコルで必要なのは、コレクタ上のソケットにアクセスできることだけです。セキュアなプロトコルでは、さらにセッションを認証および暗号化するためのセキュリティ機能が利用できます。そのため、コレクタを保護し、ルータとの通信方法を非常に高度に設定できます。ルータと宛先の間の接続が失われた場合、ルータは宛先との接続を再確立し、データのプッシュを再開します。ただし、再接続中に送信されたデータは失われます。

ダイヤルアウト モードをさらに確認し、ダイヤルアウト セッションを作成するには、ルータからコレクタへのモデル駆動型テレメトリセッションの確立 （15 ページ）を参照してください。

ダイヤルイン モード

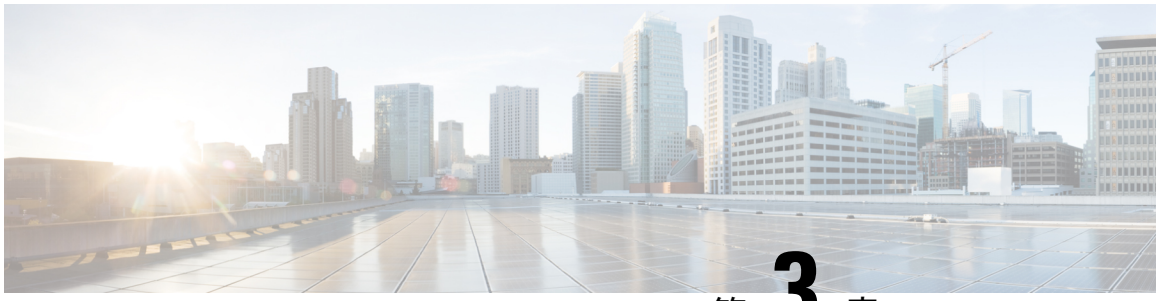
ダイヤルインモードでは、コレクタがルータにダイヤルインし、サブスクリプションで指定された1つ以上のセンサーパスを動的にサブスクライブします。ルータはコレクタからの接続のために開かれています。このモードは、ルータとの単一の通信チャネルを確立するのに便利です。コレクタがセッションを確立するため、設定で接続先を作成する必要はありません。さらに、セッションを確立するために使用されるプロトコル（gRPC）により、セッションを認証および暗号化するための高度なセキュリティ機能が提供されます。ルータとコレクタの間の接続が失われると、セッションはキャンセルされます。データのストリーミングを再開するには、コレクタがルータに再接続する必要があります。ダイヤルインセッションをサポートするのは gRPC のみです。

ダイヤルイン モードをさらに知り、ダイヤルイン セッションを作成するには、コレクタからルータへのモデル駆動型テレメトリセッションの確立 （25 ページ）を参照してください。

ネットワークに適したテレメトリ セッションを特定する

ネットワーク内のトランスポートプロトコルおよびエンコーディング形式は、どちらのモードがニーズに適しているか判断するのに役立ちます。エンコーディングの効率性は、ネットワーク上でデータが占有するスペース、メモリ使用率、およびルータからストリーミングする計画のデータ量によって決まります。

- 単一のルータとコレクタでのシンプルなセットアップを使用してテレメトリ データをストリーミングする計画の場合は、TCP ダイヤルアウト モードを使用します。これは簡単に設定でき、プロトコルに関する幅広い知識は必要ありません。着信接続のポートを管理する必要がなくなります。
- セットアップで多数のデバイスへのスケールアウトを行う場合、またはデータの暗号化が必要な場合は、gRPC ダイヤルアウト モードを使用します。このモードでは、着信接続のポートを管理する必要がなくなります。
- ネットワークで gRPC をすでに使用していて、固定の宛先にデータをストリーミングするのではなくセッションを動的にする場合は、gRPC ダイヤルイン モードを使用します。このモードは、ネットワークの設定および運用データの要求を一元的な方法で行う場合に便利です。



第 3 章

ルータからコレクタへのモデル駆動型テレメトリセッションの確立

テレメトリのストリーミングは、ネットワーク ヘルスをモニタするための新しいパラダイムです。関心のある設定データおよび運用データを Cisco IOS XR ルータから効率的にストリーミングするメカニズムを提供します。このストリーミングされるデータは、モニタリングおよびトラブルシューティングのため、構造化された形式でリモート管理ステーションに送信されます。

テレメトリ データを使用して、データ レイクを作成します。このデータを分析することにより、ネットワークのプロアクティブなモニタリング、CPUおよびメモリの使用率のモニタリング、パターンの特定、予測的な方法でのネットワークのトラブルシューティングを行い、自動化を使用した復元力のあるネットワークを作成するための戦略を考案します。

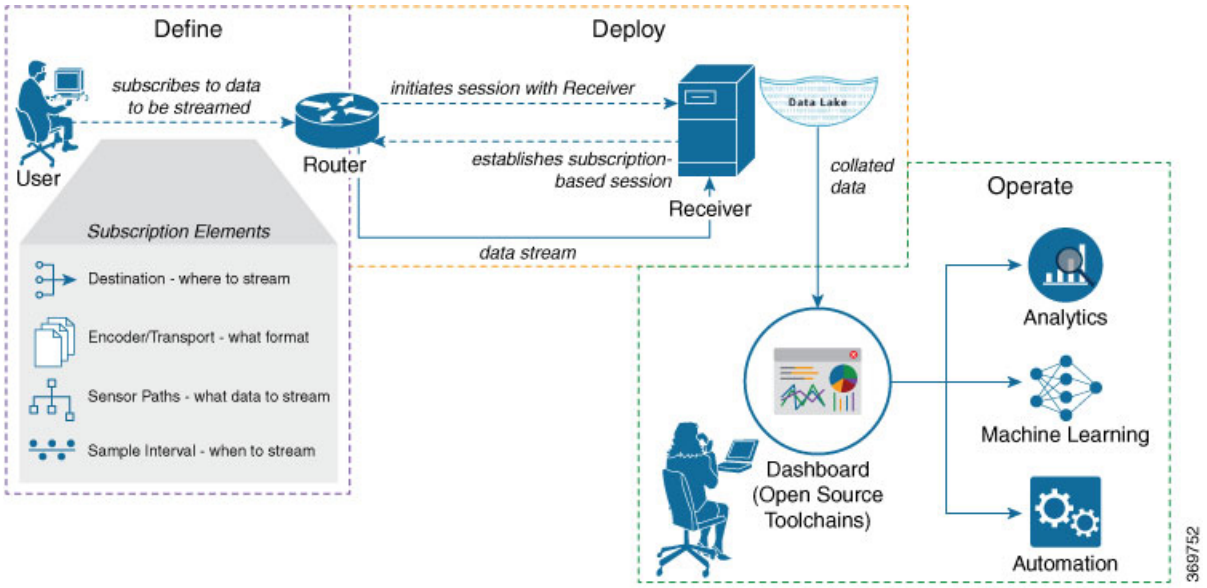
テレメトリは、関心のあるデータをセンサー パスの形でサブスクライブするサブスクリプション モデルで機能します。センサー パスは、OpenConfig データ モデルまたはネイティブのシスコ データ モデルを記述します。テレメトリのための OpenConfig データ モデルおよびネイティブ データ モデルには、バージョン管理のためのホスティング サービスを提供するソフトウェア開発プラットフォームである Github からアクセスできます。ルータと受信者の間にテレメトリ セッションを確立することによって、どのユーザがサブスクリプションを開始するかを選択します。セッションは、ダイヤルアウト モードまたはダイヤルイン モードのいずれかを使用して確立されます。これについては、「テレメトリを使用したネットワーク モニタリング戦略の拡張」の記事を参照してください。

（注） モデル駆動型テレメトリを使用したリアルタイムでのネットワーク管理がもたらす力については、こちらのビデオをご覧ください。

この記事では、ルータが受信者にダイヤルアウトしてテレメトリ セッションを確立するダイヤルアウト モードについて説明します。このモードでは、宛先とセンサーパスが設定され、1 つ以上のサブスクリプションにまとめられます。ルータは、サブスクリプション内の各宛先とのセッション確立を継続的に試行し、データを受信者にストリーミングします。サブスクリプションのダイヤルアウト モードは永続的です。セッションが終了すると、ルータは一定の間隔で受信者との新しいセッションを継続的に再確立しようとします。

次の図は、ダイヤルアウト モードの概要を示しています。

図 4: ダイヤルアウト モード



この記事では、CPU使用率のモニタリングを示す使用例を使用して、ネットワークの可視性の向上およびネットワークを安定させるための情報を得たうえでの意思決定にテレメトリ データのストリーミングがどのように役立つかを説明します。

- ネットワーク インフラストラクチャを計画するため、テレメトリ データを使用して CPU 使用率をモニタする （16 ページ）

ネットワークインフラストラクチャを計画するため、テレメトリ データを使用して CPU 使用率をモニタする

この使用例では、ダイヤルアウト モードで、テレメトリ データを使用して CPU 使用率をプロアクティブにモニタする方法を示します。CPU使用率をモニタすることにより、ネットワーク内のストレージ機能を効率化することができます。この使用例では、オープンソースの収集スタックにある、テレメトリ データの保存および分析に使用するツールについて説明します。

（注） データ モデル、オープンソースのコレクタ、エンコーディングを利用し、それらをモニタリング ツールに統合するためにモデル駆動型テレメトリを設定する方法については、こちらのビデオをご覧ください。

テレメトリには、次のワークフローがあります。

- **定義**：ルータから受信者にデータをストリーミングするためのサブスクリプションを定義します。サブスクリプションを定義するには、宛先グループとセンサーグループを作成します。
- **展開**：ルータは、サブスクリプションベースのテレメトリ セッションを確立し、受信者にデータをストリーミングします。ルータでサブスクリプションの展開を確認します。
- **運用**：オープンソース ツールを使用してテレメトリ データを消費および分析し、分析に基づいて必要なアクションを実行します。

始める前に

ルータと受信者の間に L3 接続があることを確認します。

ルータから受信者にデータをストリーミングするためのサブスクリプションを定義する

サブスクリプションを作成し、ルータから宛先にストリーミングする対象データを定義します。

手順

ステップ 1 ルータからテレメトリ データを収集する宛先を 1 つ以上作成します。宛先に関する詳細を格納する宛先グループを定義します。宛先グループには、宛先アドレス（ipv4 または ipv6）、ポート、トランスポート、およびエンコーディング形式を含めます。

例：

データ モデルを使用して宛先グループを作成する

この例では、ネイティブ データモデル `Cisco-IOS-XR-um-telemetry-model-driven-cfg.yang` を使用しています。

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get-config>
    <source>
      <candidate/>
    </source>
    <filter>
      <telemetry-model-driven
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-telemetry-model-driven-cfg">
        <destination-groups>
          <destination-group>
            <destination-id>CPU-Health</destination-id>
            <ipv4-destinations>
              <ipv4-destination>
                <ipv4-address>172.0.0.0</ipv4-address>
                <destination-port>57500</destination-port>
                <encoding>self-describing-gpb</encoding>
                <protocol>
                  <protocol>tcp</protocol>
```

```
              </protocol>
            </ipv4-destination>
          </ipv4-destinations>
        </destination-group>
      </destination-groups>
    </telemetry-model-driven>
  </filter>
</get-config>
</rpc>
```

CLI を使用して宛先グループを作成する

```
##Configuration with tls-hostname##
Router(config)#telemetry model-driven
Router(config-model-driven)#destination-group CPU-Health
Router(config-model-driven-dest)#address family ipv4 172.0.0.0 port 57500
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
```

ここで、

- `CPU-Health` は、宛先グループの名前です
- `172.0.0.0` は、データがストリーミングされる宛先の IP アドレスです
- `57500` は、宛先のポート番号です
- `self-describing-gpb` は、データがエンコードされ、宛先にストリーミングされる形式です
- `tcp` は、データが宛先に転送されるプロトコルです

ステップ 2 センサーパスを使用して、ルータからストリーミングするデータのサブセットを指定します。センサー パスは、YANG データモデルの階層内のパスを表します。センサー パスを含むセンサーグループを作成します。

例：

データ モデルを使用して CPU 使用率のセンサーグループを作成する

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <telemetry-model-driven
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-telemetry-model-driven-cfg">
        <sensor-groups>
          <sensor-group>
            <sensor-group-identifier>Monitor-CPU</sensor-group-identifier>
            <sensor-paths>
              <sensor-path>

<telemetry-sensor-path>Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization</telemetry-sensor-path>

              </sensor-path>
            </sensor-paths>
          </sensor-group>
        </sensor-groups>
```

```
      </telemetry-model-driven>
    </config>
  </edit-config>
</rpc>
```

CLI を使用して CPU 使用率のセンサーグループを作成する

```
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group Monitor-CPU
Router(config-model-driven-snsr-grp)# sensor-path
Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization
Router(config-model-driven-snsr-grp)# commit
```

ここで、

- `Monitor-CPU` は、センサーグループの名前です
- `Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization` は、データのストリーミング元となるセンサー パスです。

ステップ 3 ルータからストリーミングされるテレメトリ データをサブスクライブします。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バインドし、ストリーミング方式を設定します。ストリーミング方式は、パターン駆動型テレメトリまたはイベント駆動型テレメトリにすることができます。

例：

（注） イベント駆動型テレメトリの設定は、サンプル間隔が異なる点を除き、パターン駆動型テレメトリに似ています。サンプル間隔の値を `0`（ゼロ）に設定すると、イベント駆動型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、パターン駆動型テレメトリのサブスクリプションが設定されます。

データ モデルを使用してサブスクリプションを作成する

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <telemetry-model-driven
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-telemetry-model-driven-cfg">
        <subscriptions>
          <subscription>
            <subscription-identifier>CPU-Utilization</subscription-identifier>
            <sensor-profiles>
              <sensor-profile>
                <sensorgroupid>Monitor-CPU</sensorgroupid>
                <sample-interval>30000</sample-interval>
              </sensor-profile>
            </sensor-profiles>
            <destination-profiles>
              <destination-profile>
                <destination-id>CPU-Health</destination-id>
              </destination-profile>
            </destination-profiles>
           <source-interface>Interface1</source-interface>
          </subscription>
        </subscriptions>
```

```
      </telemetry-model-driven>
    </config>
  </edit-config>
</rpc>
```

CLI を使用してサブスクリプションを作成する

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription CPU-Utilization
Router(config-model-driven-subs)#sensor-group-id Monitor-CPU sample-interval 30000
Router(config-model-driven-subs)#destination-id CPU-Health
Router(config-model-driven-subs)#source-interface Interface1
Router(config-model-driven-subs)#commit
```

ここで、

- CPU-Utilization はサブスクリプションの名前です
- Monitor-CPU は、センサーグループの名前です
- CPU-Health は、宛先グループの名前です
- Interface1は、テレメトリ セッションの確立に使用される送信元インターフェイスです。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、宛先グループで指定されたものと同じ VRF にあることが必要です。
- 30000 は、ミリ秒単位のサンプル間隔です。サンプル間隔は、2 つのデータ ストリーム間の時間間隔です。この例では、サンプル間隔は 3 万ミリ秒（30 秒）です。

サブスクリプションの展開を確認する

ルータは、受信者にダイヤルアウトし、サブスクリプション内の各宛先とのセッションを確立します。セッションが確立されると、ルータはデータを受信者にストリーミングしてデータ レイクを作成します。

サブスクリプションの展開は、ルータ上で確認できます。

手順

ステップ 1 ルータで、モデル駆動型テレメトリの設定を表示します。

例：

```
Router#show running-config telemetry model-driven
telemetry model-driven
destination-group CPU-Health
address-family ipv4 172.0.0.0 port 57500
encoding self-describing-gpb
protocol tcp
!
sensor-group Monitor-CPU
sensor-path
Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization
```

```
!
subscription CPU-Utilization
sensor-group-id Monitor-CPU sample-interval 30000
destination-id CPU-Health
!
```

ステップ 2 サブスクリプションの状態を確認します。`Active` 状態は、ルータがサブスクリプションに基づいて受信者にデータをストリーミングする準備が整っていることを示します。

例：

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

Subscription:  CPU-Utilization                      State: ACTIVE
-------------
  Sensor groups:
  Id                 Interval(ms)        State
  Monitor-CPU        30000               Resolved

  Destination Groups:
  Id                 Encoding            Transport   State   Port    IP
  CPU-Health         self-describing-gpb tcp         Active  57500   172.0.0.0
```

ルータは、サブスクリプションベースのテレメトリ セッションを使用して受信者にデータをストリーミングし、受信側にデータ レイクを作成します。

ネットワークの詳細な分析のためにテレメトリ データを操作する

データ レイクからのテレメトリ データの消費と分析を開始するには、オープンソースの収集スタックを使用できます。この使用例では、収集スタックの次のツールを使用します。

- Pipeline は、データを収集するために使用される軽量ツールです。Network Telemetry Pipeline は、Github からダウンロードできます。`pipeline.conf` ファイルを使用して、コレクタがルータと通信する方法と処理されたデータを送信する場所を定義します。
- Telegraph（プラグイン駆動型サーバ エージェント）および InfluxDB（時系列データベース（TSDB））は、可視化ツールによって取得されるテレメトリ データを保存します。InfluxDB は、Github からダウンロードできます。`metrics.json` ファイルを使用して、TSDB に含めるデータを定義します。
- Grafana は、ルータからストリーミングされたデータのグラフおよびカウンタを表示する可視化ツールです。

つまり、Pipeline は TCP および gRPC テレメトリ ストリームを受け入れてデータを変換し、そのデータを InfluxDB データベースにプッシュします。Grafana は、InfluxDB データベースからのデータを使用してダッシュボードおよびグラフを作成します。Pipeline と InfluxDB は、同じサーバ上でも異なるサーバ上でも実行できます。

ルータは 約 350 のカウンタのデータを 5 秒ごとにストリーミングし、Telegraf は Pipeline からの情報を 1 秒間隔で要求しているとします。CPU 使用率は、次を使用して 3 つのステージで分析されます。

- 最初の値を取得する単一のルータ
- 値の差異を特定し、パターンを把握する 2 台のルータ
- 証拠に基づく結論に達するための 5 台のルータ

これにより、インフラストラクチャ（この場合はCPU）の展開について、情報を得たうえでビジネス上の意思決定を行うことができます。

手順

ステップ 1 Pipeline を開始し、ルータのクレデンシャルを入力します。

（注） 宛先グループで指定する IP アドレスおよびポートは、Pipeline がリスニングしている IP アドレスおよびとポートと一致している必要があります。

例：

```
$ bin/pipeline -config pipeline.conf

Startup pipeline
Load config from [pipeline.conf], logging in [pipeline.log]

CRYPT Client [grpc_in_mymdtrouter], [http://172.0.0.0:5432]
 Enter username: <username>
 Enter password: <password>
Wait for ^C to shutdown
```

ステップ 2 CPU 使用率に関するメトリックを読み取るため、Telegraph 設定ファイルで次の値を追加します。

例：

```
[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
  ## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics.
  collect_cpu_time = false
  ## If true, compute and report the sum of all non-idle CPU states.
  report_active = false
```

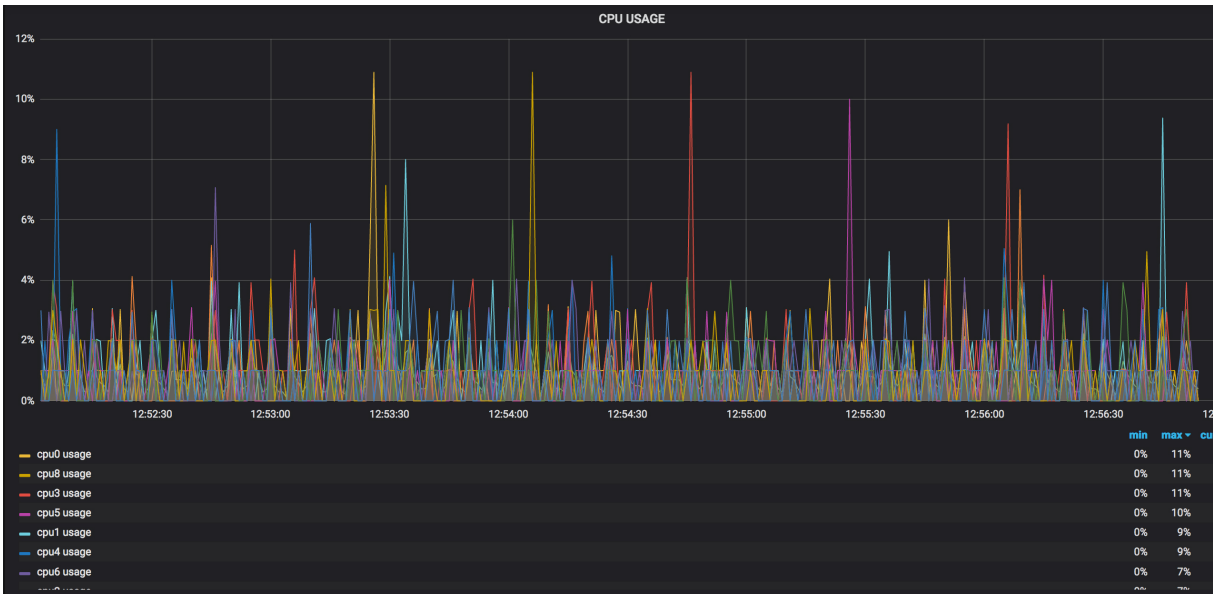
ステップ 3 Grafana を使用してダッシュボードを作成し、CPU 使用率に関するデータを可視化します。

1 台のルータ

ルータは 5 秒ごとにカウンタをプッシュします。

すべての CPU コアに負荷が均等に分散され、約 10% または 11% までのスパイクが発生しています。

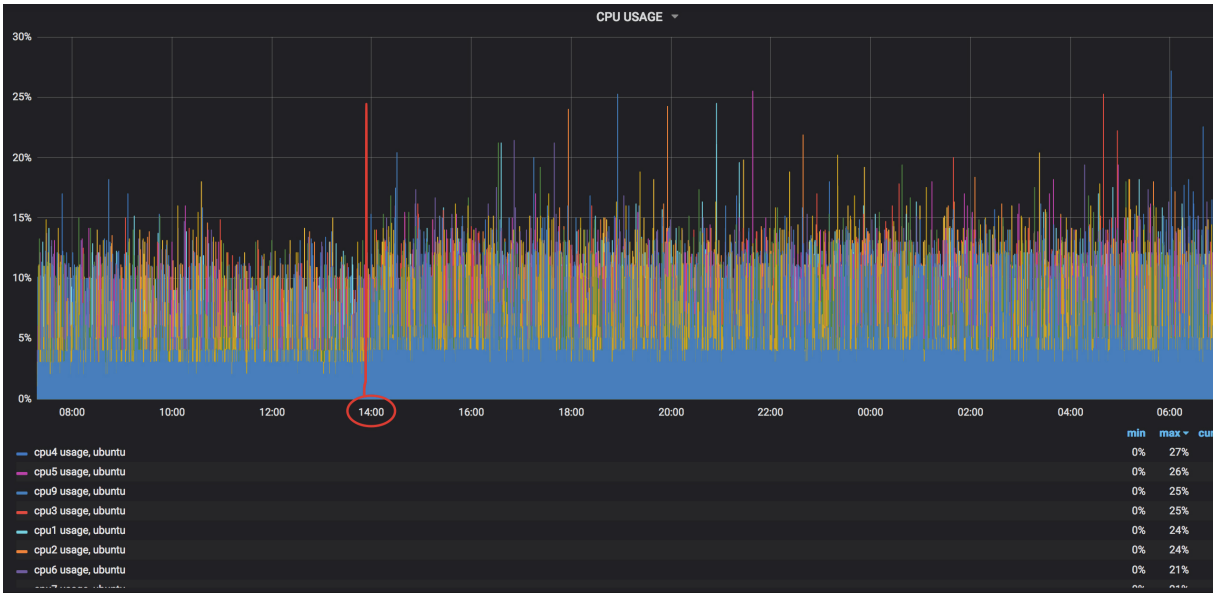
図 *5 :* 単一ルータの場合の *CPU* 使用率グラフ



2 台のルータ

2 台目のルータがタイムラインの `14:00` に追加され、スパイクが約 25% に増加していることを示しています。中間値は 15% です。

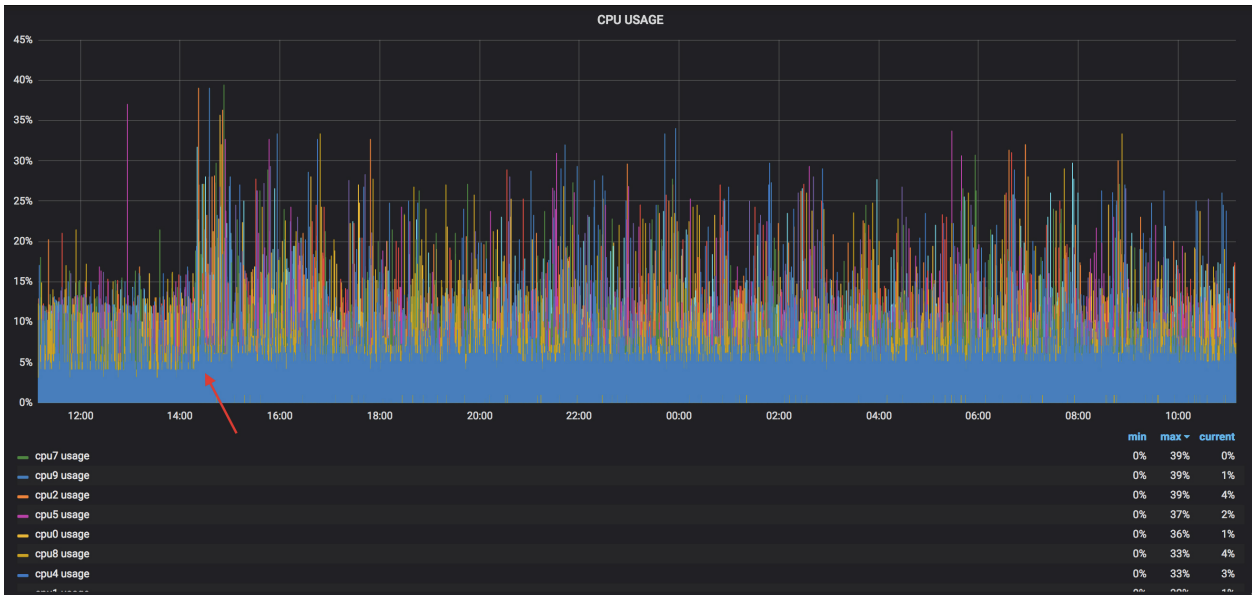
図 *6 : 2* 台のルータを使用した場合の *CPU* 使用率グラフ



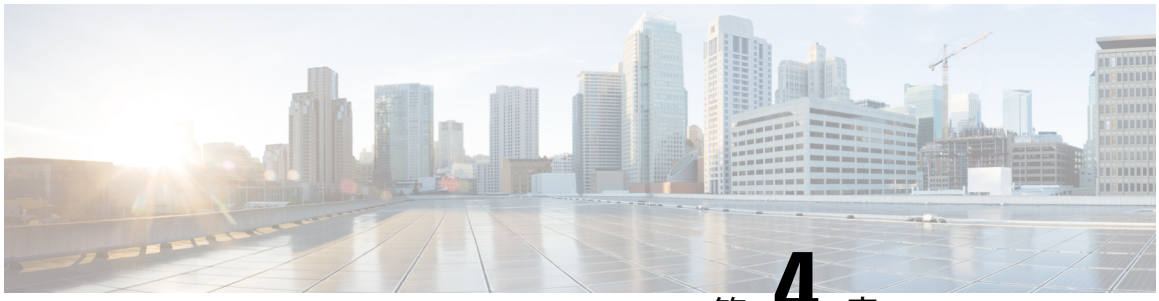
5 台のルータ

5 台のルータが使用され、約 40% をピークとするスパイクが発生しています。中間値は 約 22 ～ 25% の範囲です。

図 *7: 5* 台のルータを使用した場合の *CPU* 使用率グラフ



結論として、テレメトリ データは、プロセスがすべての CPU コアに対しほぼ均等に分散されていることを示しています。コアのサブセットには線形的な増加はありません。この分析は、ストリーミングするカウンタの数に基づいて CPU 使用率を計画するうえで役立ちます。



第 4 章

コレクタからルータへのモデル駆動型テレメトリセッションの確立

テレメトリのストリーミングは、ネットワーク ヘルスをモニタするための新しいパラダイムです。関心のある設定データおよび運用データを Cisco IOS XR ルータから効率的にストリーミングするメカニズムを提供します。このストリーミングされるデータは、モニタリングおよびトラブルシューティングのため、構造化された形式でリモート管理ステーションに送信されます。

テレメトリ データを使用して、データ レイクを作成します。このデータを分析することにより、ネットワークのプロアクティブなモニタリング、CPUおよびメモリの使用率のモニタリング、パターンの特定、予測的な方法でのネットワークのトラブルシューティングを行い、自動化を使用した復元力のあるネットワークを作成するための戦略を考案します。

テレメトリは、関心のあるデータをセンサー パスの形でサブスクライブするサブスクリプション モデルで機能します。センサー パスは、OpenConfig データ モデルまたはネイティブのシスコ データ モデルを記述します。テレメトリのための OpenConfig データ モデルおよびネイティブ データ モデルには、バージョン管理のためのホスティング サービスを提供するソフトウェア開発プラットフォームである Github からアクセスできます。ルータと受信者の間にテレメトリ セッションを確立することによって、どのユーザがサブスクリプションを開始するかを選択します。セッションは、ダイヤルアウト モードまたはダイヤルイン モードのいずれかを使用して確立されます。これについては、「テレメトリを使用したネットワーク モニタリング戦略の拡張」の記事を参照してください。

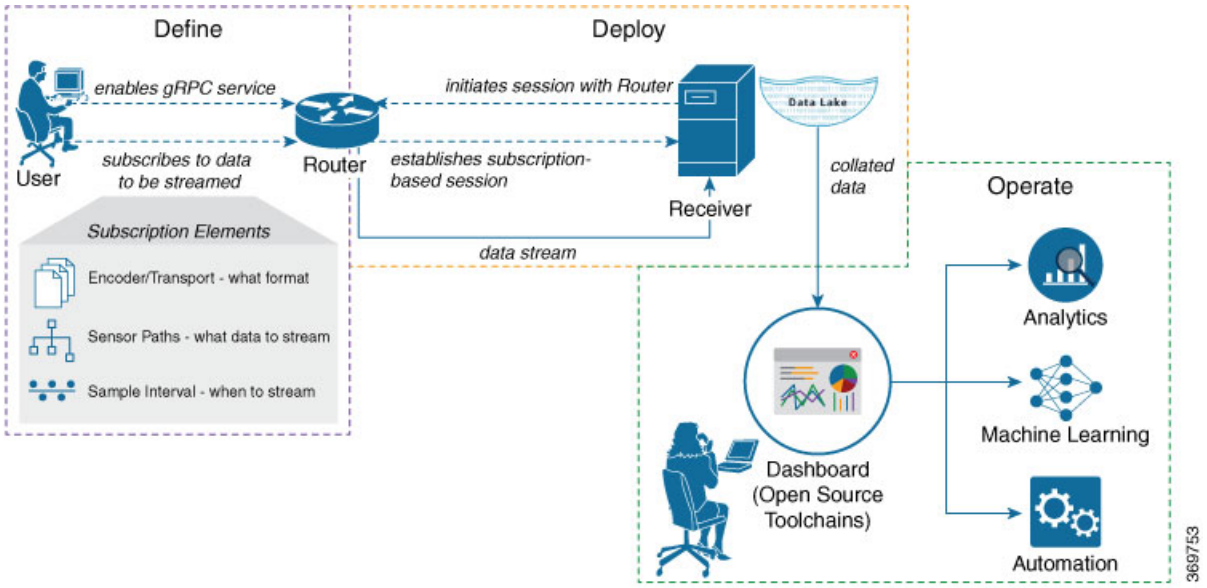
（注） モデル駆動型テレメトリを使用したリアルタイムでのネットワーク管理がもたらす力については、こちらのビデオをご覧ください。

この記事では、受信者がルータにダイヤルインしてテレメトリ セッションを確立するダイヤルイン モードについて説明します。このモードでは、受信者がルータにダイヤルインし、サブスクリプションで指定された 1 つ以上のセンサー パスに動的にサブスクライブします。ルータは、受信者が確立したセッションと同じセッションを介してテレメトリ データをストリーミングします。サブスクリプションのダイヤルイン モードは動的です。この動的サブスクリプショ

ンは、受信者がサブスクリプションをキャンセルしたとき、またはセッションが終了したときに終了します。

次の図は、ダイヤルイン モードの概要を示しています。

図 *8:* ダイヤルイン モード



この記事では、ネットワーク内のさまざまなパラメータの同時モニタリングを示す使用例を使用して、ネットワークの可視性の向上およびネットワークを安定させるための情報を得たうえでの意思決定にテレメトリ データのストリーミングがどのように役立つかを説明します。

- プロアクティブな分析のためのテレメトリ データを使用してネットワーク パラメータをモニタする （26 ページ）

プロアクティブな分析のためのテレメトリデータを使用してネットワーク パラメータをモニタする

この使用例では、ダイヤルイン モードで、テレメトリ データを使用してネットワークに関するさまざまなパラメータをストリーミングする方法について説明します。このデータは、パターンをモニタして問題をプロアクティブにトラブルシューティングする予測分析に使用します。この使用例では、オープンソースの収集スタックにある、テレメトリ データの保存および分析に使用するツールについて説明します。

（注） データ モデル、オープンソースのコレクタ、エンコーディングを利用し、それらをモニタリング ツールに統合するためにモデル駆動型テレメトリを設定する方法については、こちらのビデオをご覧ください。

テレメトリには、次のワークフローがあります。

- **定義**：ルータから受信者にデータをストリーミングするためのサブスクリプションを定義します。サブスクリプションを定義するには、センサーグループを作成します。
- **展開**：受信者は、ルータとのセッションを開始し、サブスクリプションベースのテレメトリ セッションを確立します。ルータは、受信者にデータをストリーミングします。ルータでサブスクリプションの展開を確認します。
- **運用**：オープンソース ツールを使用してテレメトリ データを消費および分析し、分析に基づいて必要なアクションを実行します。

始める前に

次の依存関係を満たしていることを確認します。

- ルータと受信者の間に L3 接続があることを確認します。
- ルータの gRPC サーバが受信者からの着信接続を受け入れるようにします。

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

ポート番号の範囲は 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。

次の例は、ルータで TLS が有効になっている場合の gRPC 設定の出力を示します。

```
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-aes128-sha
Operational disable : none
```

ルータから受信者にデータをストリーミングするためのサブスクリプションを定義する

サブスクリプションを作成し、ルータから宛先にストリーミングする対象データを定義します。

手順

ステップ 1 センサーパスを使用して、ルータからストリーミングするデータのサブセットを指定します。センサー パスは、YANG データモデルの階層内のパスを表します。この例では、ネイティブ データモデル `Cisco-IOS-XR-um-telemetry-model-driven-cfg.yang` を使用しています。センサーパスを含むセンサーグループを作成します。

例：

```
sensor-group health
  sensor-path Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization
  sensor-path Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary
  sensor-path Cisco-IOS-XR-shellutil-oper:system-time/uptime
 !
 sensor-group interfaces
  sensor-path
Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters

  sensor-path Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-summary
 !
 sensor-group optics
  sensor-path
Cisco-IOS-XR-controller-optics-oper:optics-oper/optics-ports/optics-port/optics-info
 !
 sensor-group routing
  sensor-path
Cisco-IOS-XR-clns-isis-oper:isis/instances/instance/levels/level/adjacencies/adjacency
  sensor-path Cisco-IOS-XR-clns-isis-oper:isis/instances/instance/statistics-global
  sensor-path
Cisco-IOS-XR-ip-rib-ipv4-oper:rib/vrfs/vrf/afs/af/safs/saf/ip-rib-route-table-names/ip-rib-route-table-name/protocol/isis/as/information

  sensor-path
Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/process-info

 !
 sensor-group mpls-te
 sensor-path Cisco-IOS-XR-mpls-te-oper:mpls-te/tunnels/summary
 sensor-path Cisco-IOS-XR-ip-rsvp-oper:rsvp/interface-briefs/interface-brief
 sensor-path Cisco-IOS-XR-ip-rsvp-oper:rsvp/counters/interface-messages/interface-message

 !
```

ステップ 2 ルータからストリーミングされるテレメトリ データをサブスクライブします。サブスクリプションは、センサーグループをバインドし、ストリーミング方式を設定します。ストリーミング方式は、パターン駆動型テレメトリまたはイベント駆動型テレメトリにすることができます。センサーパスを異なるサブスクリプションに分割すると、ルータの効率が向上し、運用データを大規模に取得できるようになります。

例：

（注） イベント駆動型テレメトリの設定は、サンプル間隔が異なる点を除き、パターン駆動型テレメトリに似ています。サンプル間隔の値を 0（ゼロ）に設定すると、イベント駆動型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、パターン駆動型テレメトリのサブスクリプションが設定されます。

```
subscription health
  sensor-group-id health strict-timer
  sensor-group-id health sample-interval 30000
 !
 subscription interfaces
  sensor-group-id interfaces strict-timer
  sensor-group-id interfaces sample-interval 30000
 !
 subscription optics
  sensor-group-id optics strict-timer
  sensor-group-id optics sample-interval 30000
 !
 subscription routing
  sensor-group-id routing strict-timer
  sensor-group-id routing sample-interval 30000
 !
 subscription mpls-te
  sensor-group-id mpls-te strict-timer
  sensor-group-id mpls-te sample-interval 30000
 !
```

サブスクリプションの展開を確認する

受信者は、受信者にダイヤルインし、サブスクリプションに基づいて動的なセッションを確立します。セッションが確立されると、ルータはデータを受信者にストリーミングしてデータ レイクを作成します。

サブスクリプションの展開は、ルータ上で確認できます。

手順

サブスクリプションの状態を確認します。`Active`状態は、ルータがサブスクリプションに基づいて受信者にデータをストリーミングする準備が整っていることを示します。

例：

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

ルータは、サブスクリプションベースのテレメトリ セッションを使用して受信者にデータをストリーミングし、受信側にデータ レイクを作成します。

ネットワークの詳細な分析のためにテレメトリ データを操作する

データ レイクからのテレメトリ データの消費と分析を開始するには、オープンソースの収集スタックを使用できます。この使用例では、収集スタックの次のツールを使用します。

- Pipelineは、データを収集するために使用される軽量ツールです。Network Telemetry Pipelineは、Github からダウンロードできます。`pipeline.conf` ファイルを使用して、コレクタがルータと通信する方法と処理されたデータを送信する場所を定義します。
- Telegraph または InfluxDB は、可視化ツールによって取得されるテレメトリ データを保存する時系列データベース（TSDB）です。InfluxDBは、Github からダウンロードできます。`metrics.json` ファイルを使用して、TSDB に含めるデータを定義します。
- Grafana は、ルータからストリーミングされたデータのグラフおよびカウンタを表示する可視化ツールです。

つまり、Pipeline は TCP および gRPC テレメトリ ストリームを受け入れてデータを変換し、そのデータを InfluxDB データベースにプッシュします。Grafana は、InfluxDB データベースからのデータを使用してダッシュボードおよびグラフを作成します。Pipeline と InfluxDB は、同じサーバ上でも異なるサーバ上でも実行できます。

ルータを次のパラメータでモニタすることを検討してください。

- メモリおよび CPU 使用率
- インターフェイス カウンタおよびインターフェイス サマリー
- 光コントローラからのトランスミッタおよびレシーバの電力レベル
- ISIS ルートカウントおよび ISIS インターフェイス
- BGP ネイバー、パス カウント、およびプレフィックス カウント
- MPLS-TE トンネルの概要
- 各インターフェイスでの RSVP 制御メッセージと帯域幅の割り当て

手順

ステップ 1 Pipeline を開始し、ルータのクレデンシャルを入力します。

例：

```
$ bin/pipeline -config pipeline.conf

Startup pipeline
Load config from [pipeline.conf], logging in [pipeline.log]

CRYPT Client [grpc_in_mymdtrouter], [http://172.0.0.0:5432]
 Enter username: <username>
 Enter password: <password>
Wait for ^C to shutdown
```

ストリーミングされたテレメトリ データは、InfluxDB に保存されます。

ステップ 2 Grafanaを使用して、ダッシュボードを作成し、ストリーミングされたデータを可視化します。

図 *9*：テレメトリ データを使用したネットワーク ヘルスの視覚的分析

図 *10*：テレメトリ データを使用したシステム モニタリングの視覚的分析

結論として、テレメトリデータは、ネットワークのさまざまなパラメータを同時にモニタできることを示しています。このデータは、ネットワークのパフォーマンスに影響を与えることな

く、ほぼリアルタイムでストリーミングされます。このデータを使用すると、ネットワークの可視性が向上します。