ASR9Kモデル駆動型テレメトリホワイトペーパー

内容

概要

対象者

このホワイトペーパーの目的は、モデル駆動型テレメトリ(MDT)機能の概要と、この機能がアグリゲーションサービスルータ(ASR)9000(ASR9K)でどのように実装されているかについて、お客様が迅速に理解できるようにすることです。これには、設計ガイドラインと設定の詳細も含まれます。また、導入に関する考慮事項も含まれており、ASR9Kを使用してこの機能を導入する際に役立ちます。全体として、このホワイトペーパーは、この機能を扱う人のためのクイックリファレンスガイドになります。

テレメトリは一般的な機能として導入されていますが、ASR9Kの実装に重点が置かれています。つまり、他のシスコプラットフォームでサポートされているすべての機能がASR9Kプラットフォームでサポートされているわけではなく、一部の機能の実装はASR9Kに固有のものもあります。

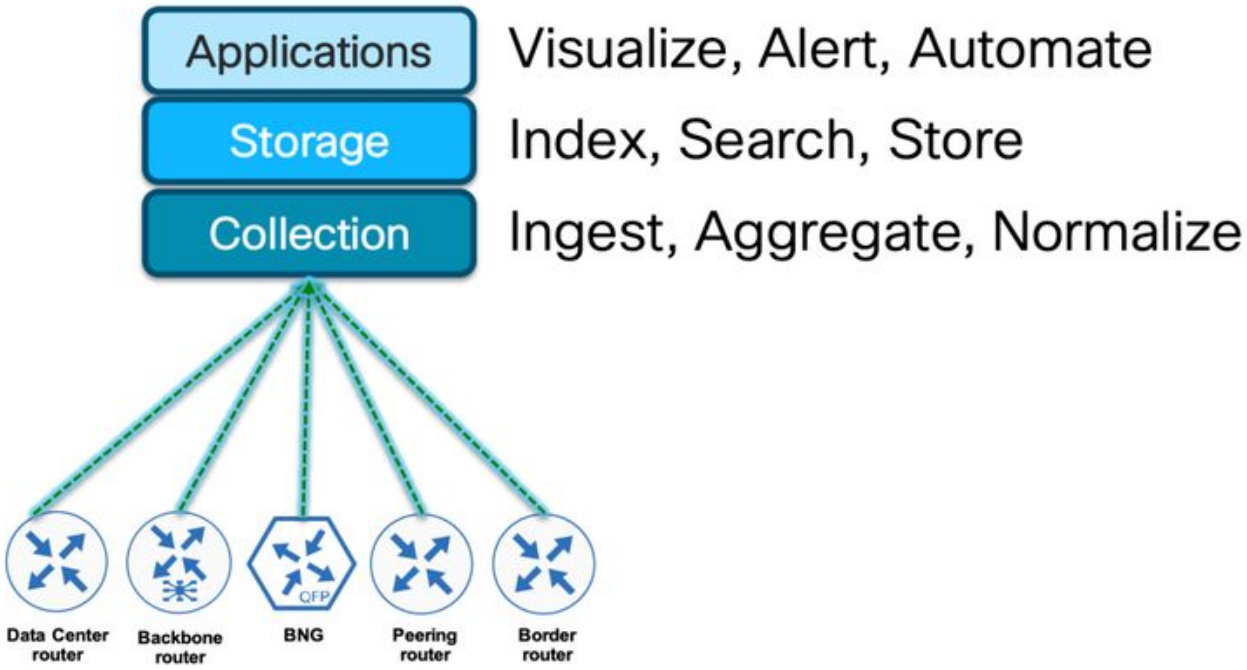
テレメトリの概要

まず簡単に言うと、テレメトリは有用な運用データの収集プ**ロセスです**。Wikipediaによると、テレメトリは自動化された通信プロセスで、測定データやその他のデータをリモートまたはアクセスできない場所で収集し、モニタリングのために受信装置に送信します。テレメトリ語そのものは、ギリシャ語の語源から派生しています。tele = remote、metron = measure。

ネットワーク管理については、ネットワークオペレータはSimple Network Management Protocol（SNMP；簡易ネットワーク管理プロトコル）に長い間依存してきました。SNMPはネットワーク監視に広く採用されていますが、SNMPによる設定機能が常に存在していたにもかかわらず、設定に使用されることはありませんでした。オペレータは日常的な設定タスクを処理する自動化スクリプトを作成していますが、そのようなタスクではスクリプトの作成が困難で、管理が困難です。

そのため、オペレータはデータモデル駆動管理へと移行しました。ネットワーク設定は、netconfなどのプロトコルによってプッシュされるYANGデータモデルに基づいています。設定をプッシュするだけでは、設定されたサービスが実行されているとは限りません。設定と同時にサービスの動作データを監視できるメカニズムが必要です。これは、Telemetryがデバイスから情報をプッシュするために使用するoperデータモデルが役立つ場所です。したがって、設定はYANGデータモデル駆動であるため、同じオブジェクトのセマンティックを持つために、テレメトリの場合と同様に、サービスの検証である必要があります。そのため、この用語は**モデル駆動型テレメトリ**またはストリーミングテレメトリと呼ばれます。

モデル駆動型テレメトリ(MDT)は、**cXR**（32ビットIOS XR）のリリース**6.1.1**以降に導入され、重要なデータをほぼリアルタイムで収集および測定できるため、現代のネットワークの運用上の問題のほとんどにすばやく対応できます。



高レベルテレメトリアーキテクチャ

MDTは、ネットワークデバイスでサポートされる構造化データモデルを活用し、これらのデータモデルで定義された重要なデータを提供します。**テレメトリ**は、共通のネットワーク管理システム、プロセス、およびアプリケーションを使用して、お客様がマルチベンダーネットワークを管理するのに役立ちます。**ネットワークから収集されるデータは標準に基づいており、ベンダーの実装全体で統一されているからです。**

中央集中型の管理ステーション（通常はSNMP NMS）からのデータ取得（プル）を待つのではなく、MDTを使用して、ネットワークデバイスは、パケット転送情報、エラー統計、システムステータス、CPUおよびメモリリソースなどのネットワークの重要な機能に関するパフォーマンスデータをプロアクティブに送信(**プッシュ**)します。

テレメトリが必要な理由

分析およびトラブルシューティングの目的でデータを収集することは、ネットワークの健全性を監視する上で常に重要な側面です。SNMP、CLI、Syslogなど、ネットワークからデータを収集するためのメカニズムがいくつかあります。これらの方法は非常に長い間ネットワークに役立ちましたが、自動化の需要がある現代のネットワークには適していませんが、規模に応じたサービスは基本的です。ネットワークの健全性情報、トラフィック統計情報、および重要なインフラストラクチャ情報は、NMS内のリモートステーションに送信され、そこで運用パフォーマンスの向上とトラブルシューティング時間の短縮に使用されます。クライアントがすべてのネットワークノードをポーリングするsnmpなどのプルモデルは効率的ではありません。ポーリングするクライアントの数が多くなると、ネットワークノードの処理負荷が増加します。一方、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する機能を備えています。テレメトリにより、プッシュモデルが可能になり、ほぼリアルタイムでモニタリングデータにアクセスできます。

ストリーミングテレメトリは、対象のデータをルータから選択し、それを標準形式でリモート管理ステーションに送信して監視するメカニズムを提供します。このメカニズムにより、リアルタイムデータに基づくネットワークの微調整が可能になります。これは、ネットワークのシームレスな運用に不可欠です。テレメトリを通じて利用できるデータの細分性と頻度が高くなることで、パフォーマンスモニタリングが向上し、トラブルシューティングが向上します。

ネットワーク内の帯域利用率、リンク利用率、リスク評価、およびスケーラビリティのサービス効率の向上に役立ちます。ストリーミングテレメトリを使用すると、ネットワークオペレータが自由に使用できる、よりリアルタイムに近いデータが得られるため、意思決定の向上に役立ちます。

SNMPから離れる必要性

SNMPは過去30年間にわたって使用されており、その動作は最新のネットワークの監視ニーズに合わせて変更されていません。実際の問題は、SNMPの実行速度です。

SNMPによって生じる3つの主な課題は、実際にはSNMPの基本的な運用上の動作の一部です。そのため、SNMPには改善の余地がほとんどなく、テレメトリは3つすべてに本質的に対応します。

- **実行速度とリアルタイム監視の必要性**

SNMPはPULL Model - GetBulk / GetNext操作を使用します。この操作は、テーブルを1つのカラムから別のカラムに移動して順番に動作します。また、1つのパケットに収まらない大きなテーブルの場合は、複数の要求が必要です。 これは、SNMPの速度を低下させる最大のボトルネックであり、送信されるデータは数分で特定の時間要因によって古くなることがよくあります。この遅延は、現代のネットワークモニタリング要件には受け入れられません。

MDT(Model Driven Telemetry)はPUSHモデルを使用し、データの送信先と送信間隔を把握しているため、上記の制限を本質的に排除しています。1回のルックアップでデータを収集し、組み込みの内部テンプレートを使用して内部操作を超高速に実行できるため、より多くのデータを短時間で配信できます。

- **その他のオーバーヘッドと最適化オプションの欠如**

SNMPによってプルされるデータは、実際には内部データ構造として保存され、ノードによって内部で変換される必要があります。これは、ネットワークノードが内部データ構造をSNMP形式にマッピングする舞台裏での追加作業です。内部最適化が実行されますが、まだ十分ではありません。

一方、テレメトリは内部データ構造を直接取り出し、このデータを送信する前に最小限の処理を実行するため、最も更新されたデータを最小限の時間と労力で提供できます。

- **ワークロードの線形的性質**

同じ正確なデータを同じ時刻にポーリングする場合でも、追加のポーリングステーションを使用するたびに、ノード上の負荷が増加します。 複数のポーリングステーションから同じMIBに並列アクセスすると、応答が遅くなり、CPU使用率が高くなる可能性があります。これは、特に複数のステーションが同じMIBテーブルの異なる部分にアクセスする大規模なテーブルの場合に顕著です。

一方、テレメトリでは、複数の宛先で同じデータが必要な場合は、データを1回プルし、パケットを複製する必要があります。 プッシュモデルは、速度とスケールのSNMPプルに勝ります。

MDTでは、データ収集のアプローチが根本的に変わり、その基本原則が次の表に示されています。また、SNMPテクノロジーのキーポイントと比較されています。

Simple Network Management Protocol（SNMP）	モデル駆動型テレメトリ(MDT)
非リアルタイム情報	リアルタイム情報
拡張性が低い	高い拡張性

プルモデル	プッシュモデル
非自動化	自動化に対応/データモデル主導

ストリーミングテレメトリの利点

ストリームされたリアルタイムのテレメトリデータは、次の点で役立ちます。

キャパシティ計画/トラフィック最適化：ネットワークの帯域幅使用率とパケットドロップを頻繁に監視する場合、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が容易になります。高速再ルーティングなどの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え、SNMPポーリング間隔メカニズムよりも高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、より高速なトラフィックに対する迅速な応答時間の提供に役立ちます。

可視性の向上：ネットワークで問題が発生した後に発生する障害状況を迅速に検出し、回避できます。

モデル駆動型テレメトリの技術仕様

次のセクションでは、IOS XRモデル駆動型テレメトリ(MDT)の技術的機能と主要コンポーネントについて説明します。

テレメトリ機能

テレメトリフレームワークは、相互にリンクされた3つの機能ブロックで構成されています。

最初のブロックは**データ表現**に関するもので、これは分析や測定を参照する情報がボード上でどのように構成されているかです。

2番目のブロックは**符号化**に関するものです。上記の測定データは、サンプル間隔ごとにテレメトリによって変換され、ネットワーク全体でシリアル化できます。もちろん、相手側のコントローラは、デバイスから送信された元のデータと同一のコピーを保持するために、データをデコードできる必要があります。

最後のブロックは**転送**についてです。これは、デバイス間でデータを転送するために使用されるプロトコルスタックです。

次の表に、モデル駆動遠隔測定の主な構成要素の構造をまとめます。

機能	**コンポーネント**
データ表現	YANGデータモデル
符号化	GPB/GPB自己記述型
トランスポート	TCP/gRPC

テーブル 3 テレメトリの構成要素

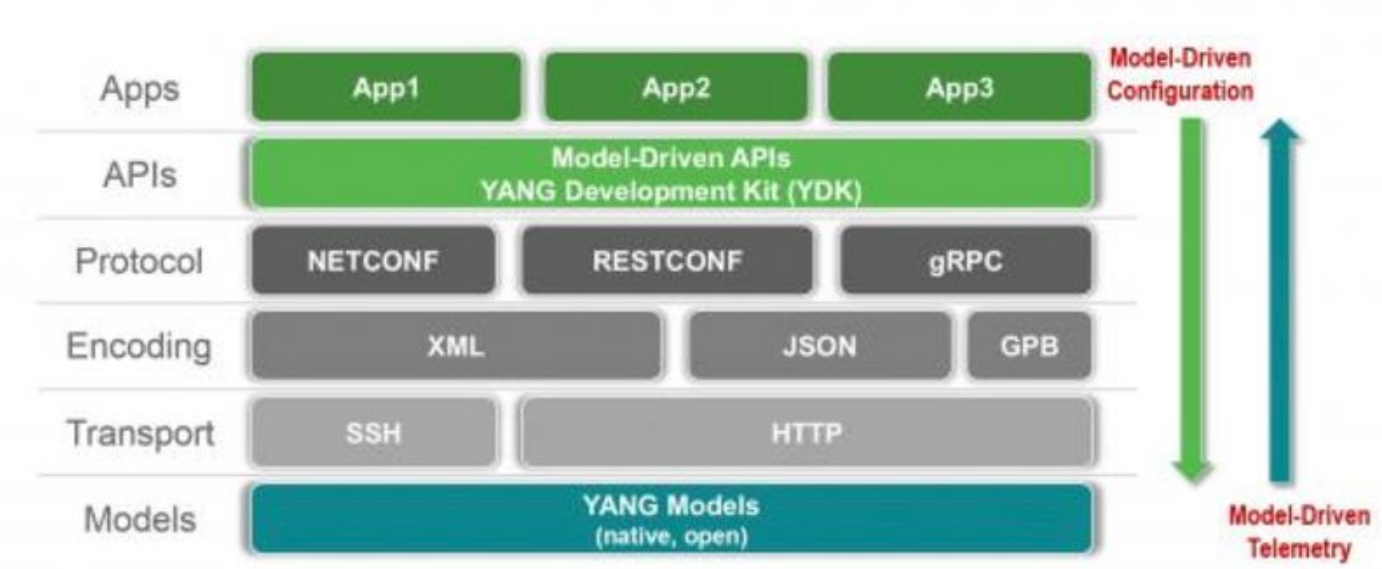
テレメトリコンポーネント

テレメトリと基盤となる設定要素の仕組みを理解する前に、最適な設定を評価するためにテレメトリのさまざまなコンポーネントを理解することが重要です。テレメトリはIOS XRのプログラマ

ビリティスタックに依存し、新しいインフラストラクチャフレームワークがネットワーク自動化に不可欠な機能を提供します。

YANGは最近データモデリングの標準となり、Ciscoのプログラマビリティスタックはこれを使用して、ネットワーク上で可能な限り迅速に符号化および伝送できる構造化データセットを形成します。YANGの柔軟性は、自動化プロセスの設定ツールとしても使用できるという大きな利点を提供します。これらのデータモデルは、特定のエンコーディングフォーマットおよびトランスポートプロトコルと組み合わされて、MDTをネットワーク分析の完全なソリューションにします。

モデル駆動型テレメトリのセットアップでは、収集と分析に必要なデータストリーミングを可能にするために、YANGデータモデルが重要なコンポーネントになります。



IOS XRプログラマビリティスタック

ヤン

Yangは、「ネットワーク管理プロトコルの設定データ、状態データ、および通知をモデル化するために使用されるデータモデリング言語」として定義されています。 YANGは、典型的なプログラミング言語アーキテクチャとは異なる性質を持つため、多様なツールと対話するように実装できます。

YANGモデリングデータ構造は、モジュールとサブモジュールの概念に基づいて構築されます。モジュールとサブモジュールは、ツリー状の形式でデータの階層を定義し、設定アクションや通知処理など、複数の操作に使用できます。

使用できるYANGモデルには複数のソースがあり、そのうち次の3つはプライマリと見なされます（YANGモデルのYANGモデルはプライマリと見なされます）。

- ネイティブモデル/シスコ固有
- OpenConfig
- IETF

シスコ固有のモデル：これらは**ネイティブモデル**とも呼ばれ、シスコを含むさまざまなデバイスベンダーによって公開されています。 例：Cisco-IOS-XR-ptp-oper.yang

OpenConfigモデル： OpenConfigは、ネットワークオペレータの非公式なワーキンググループで

す。OpenConfigは、ミッションクリティカルな機能を設定するために、すべてのベンダーがサポートする必要がある共通のYANGモデルを定義します。 例：openconfig-interfaces.yang

IETFモデル: IETFは、インターフェイスの基本設定やQOSを記述し、その他の一般的なデータタイプ（Ipv4、IPv6など）を定義する一般的なYANGモジュールもいくつか定義しています。 例：ietf-syslog-types.yang

シスコでは、利用可能なOpenconfgモデルをサポートしています。 ベンダーは、マルチベンダー環境をサポートするために、標準化されたデータのモデリング方法に統合しています。

Yangモデルには、**次の3つ**のタイプがあります。

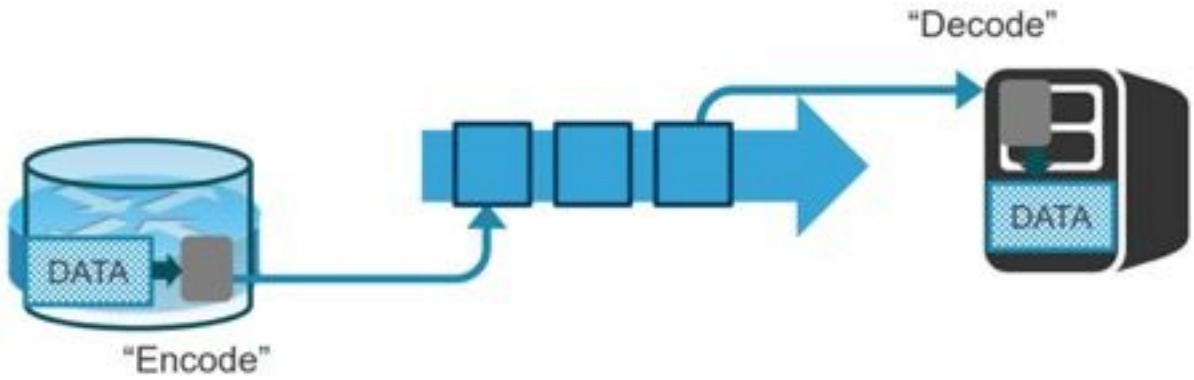
1. Operational
2. コンフィギュレーション
3. アクション

テレメトリは**Operational Yang**モデルのみを対象とし、このモデルは***-oper-*.yang**として識別できます。

YANGはRFC 7950(https://tools.ietf.org/html/rfc7950)で定義されています。

符号化

符号化（または「シリアライゼーション」）は、データ（オブジェクト、状態）をネットワーク経由で送信可能な形式に変換します。受信側がデータをデコード（「逆シリアル化」）すると、元のデータのセマンティック上同一のコピーが作成されます。

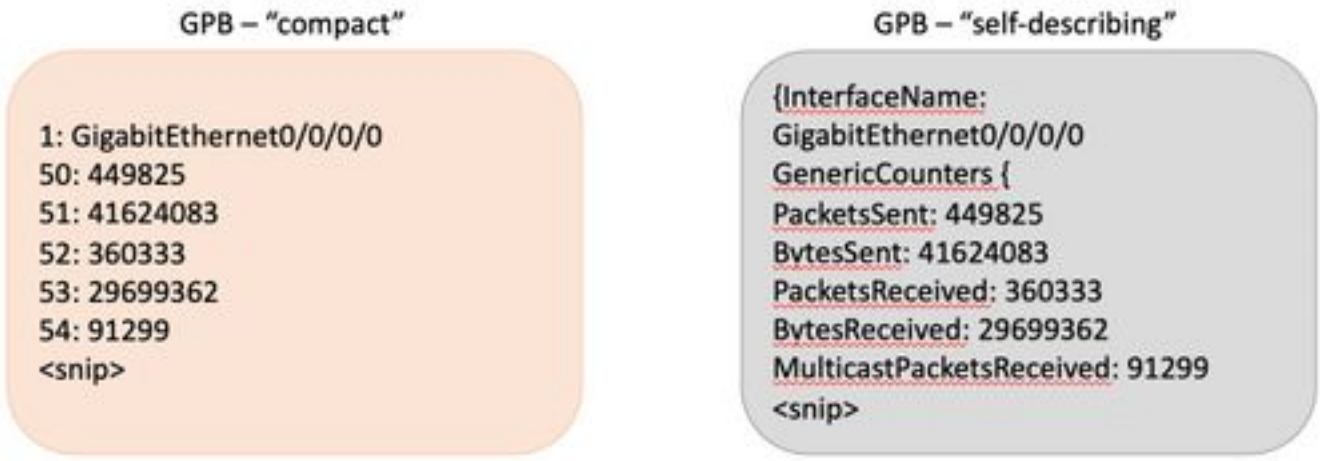


テレメトリの初期の開発段階では、XMLはタグベースの構造を持つため、最初は最初の選択肢のエンコーディングフォーマットと見なされていました。しかし、XMLの問題は非コンパクトなエンコーディング構造でした。GPB(Google Protocol Buffers)は、エンコード操作の効率と速度を向上させるため、最終的にシスコによって採用されました。

テレメトリストリーミングのエンコーディングオプションとして、GPBには次の2種類があります。

1. コンパクトGPB
2. 自己記述型GPB

2つのGPBテレメトリ形式の主な違いは、データのテレメトリストリーム内のキーをどのように表し、エンコードするかです。



JSONは、非常に理解しやすく、ほぼすべてのアプリケーションがデコードできる、人間に優しいエンコーディングスキーマです。

導入の観点からは、エンコーディングスキーマの長所と短所はほとんどありません。さまざまなエンコーディングスキーマの比較については、「テレメトリ設計ガイドライン」セクションを参照してください。

トランスポート

テレメトリは、次の3つの転送プロトコルの選択肢を提供します。

- TCP
- gRPC
- UDP

テレメトリでは、ノードとコレクタ間のセッションを開始するために、次の2つの異なる開始モードも定義されています。

- ダイヤルアウト
- ダイヤルイン

2つのモードの違いは、トランスポートセッションの確立方法だけです。

ダイヤルアウトセッション中、デバイスは事前設定されたサーバポートにsynパケットを送信して接続を開始します。接続が確立されると、データストリームはすぐにデバイスからプッシュされます。

ダイヤルインセッションの場合、ルータはサーバ接続を待っているtcpポートを受動的にリッスンします。

ただし、セッションが確立されると、ルータはサーバ自体によってポーリングされません。これは、デバイスが引き続きデータ転送操作を行うためです。MDTでは、実際にデータポーリングの概念は存在しません。

TCPは信頼性が高く、オプションとして設定が非常に簡単であるため、デフォルトでは、テレメトリのトランスポート方法として事前定義されています。

gRPCは、あらゆる環境で動作するように設計された最新のオープンソースフレームワークです

。HTTP/2上に構築され、拡張された豊富な機能セットを提供します。

ケイデンスベースのテレメトリとイベントベースのテレメトリ

サブスクライブされたデータセットのデータは、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がケイデンスベースのテレメトリとイベントベースのテレメトリのどちらに設定されているかによって異なります。

イベントベースのテレメトリの設定はケイデンスベースのテレメトリと似ており、差別化要因はサンプル間隔のみです。サンプル間隔の値を0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ます。一方、間隔を0以外の任意の値に設定すると、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ます。

変更に関連するイベントには、イベント駆動型テレメトリを使用することをお勧めします。

テレメトリ設計ガイドライン

説明したように、テレメトリスタックには多くのコンポーネントがあります。ここでは、XRデバイスにテレメトリを実装する際に考慮すべきガイドラインをいくつか示します。

エンコーディングスキーマの選択方法

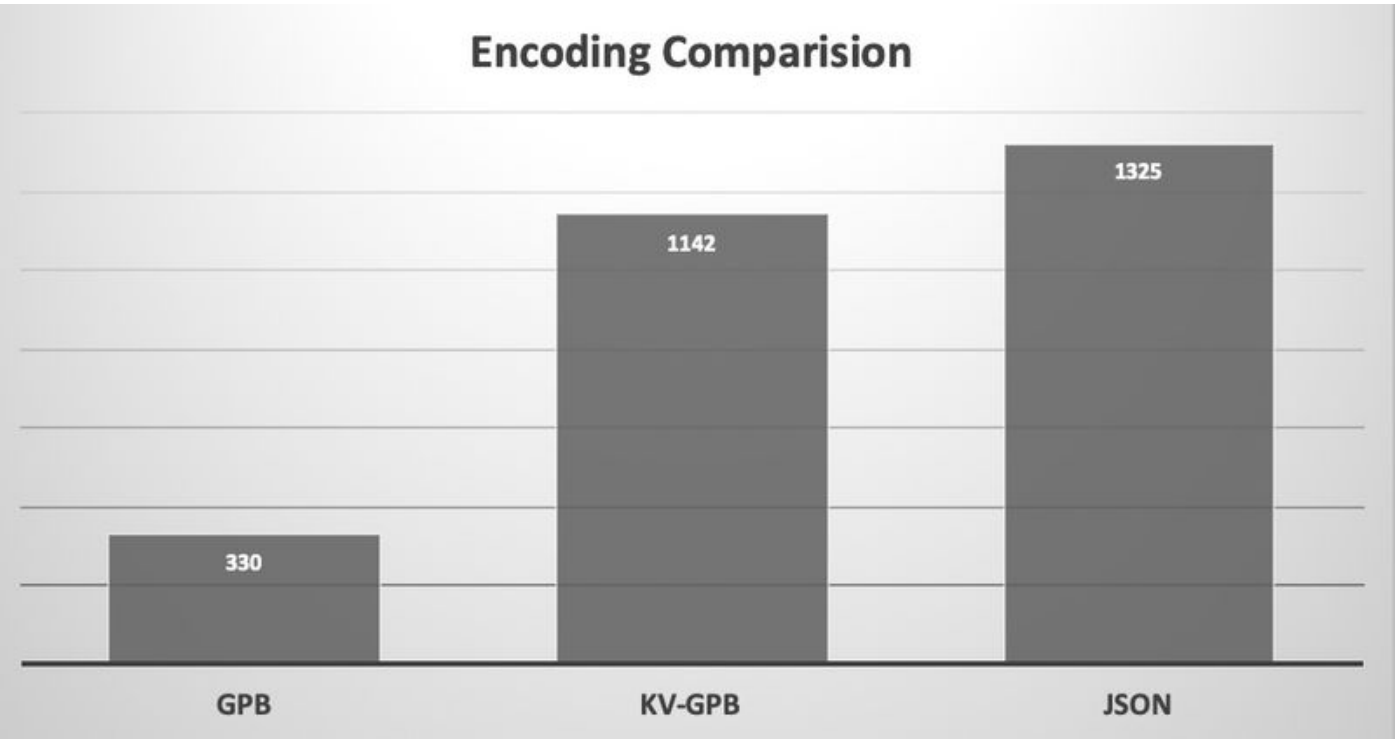
前述したように、符号化またはシリアル化は、データ（オブジェクト、状態）をネットワーク経由で送信可能な形式に変換します。受信側がデータをデコードまたは逆シリアル化すると、元のデータと意味的に同一のコピーが作成されます。

ワイヤの効率と使いやすさは、さまざまなエンコーディングオプションによって異なります。

符号化	簡単な説明	ネットワークの効率性	その他の考慮事項
GPB（コンパクト）	Everythingバイナリ（文字列である値を除く）2倍の速度、運用上の複雑さ（ただし、SNMPとの比較は不可）	高	モデルごとのProtoフル
GPB - KV（キーと値のペア）	文字列キーとバイナリ値（文字列である値を除く）3倍の大きさ、ネイティブモデル：キー名のヒューリスティックが必要	中～低	デコーディング用の単.protoファイル
JSON	Everythingストリング：キーと値	低い	フレンドリー。人間がる、アプリケーションいやすく解析が容易

GPB-KVは、エンコーディングスキーマに優れたバランスの中間点を提供します。

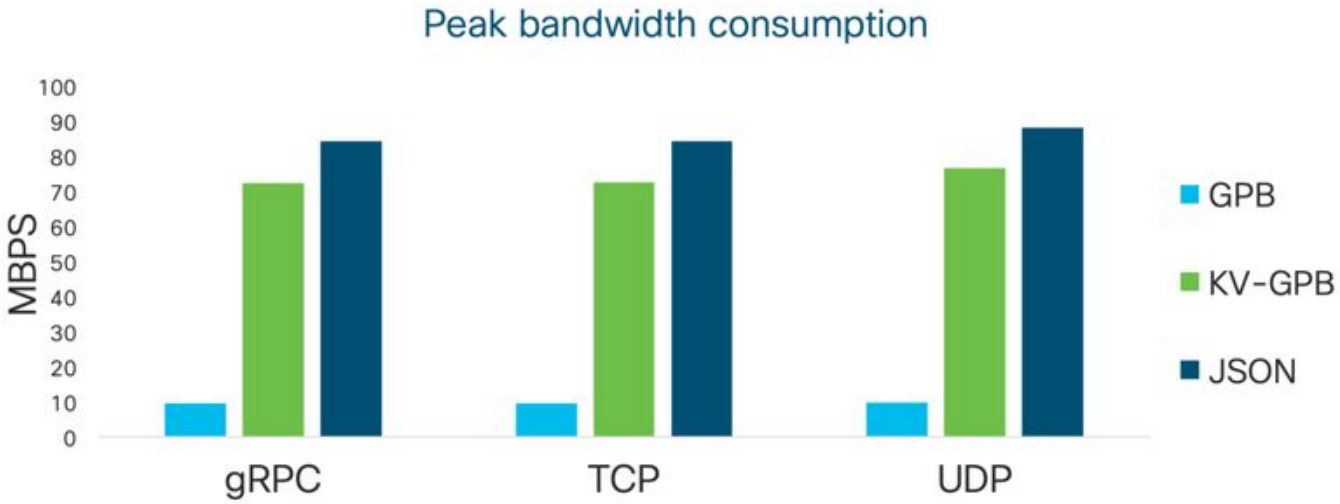
選択したエンコーディングスキーマのメッセージ長に関して、次にワイヤ上の比較を示します。



エンコードの比較 – メッセージ長 (バイト単位)

トランスポートネットワーク設計の考慮事項

エンコーディングオプションが異なると、必要な帯域幅も異なります。テレメトリを検討する際、ネットワークオペレータは、選択したエンコーディングスキーマに従って十分な帯域幅プロビジョニングを行う必要があります。エンコーディングスキーマの比較ごとの帯域幅使用量を下回るという公平な考えを持つためです。



ネットワーク帯域幅の比較

KV-GPBの使用を推奨します。効率性と利便性の中間点として機能します。

テレメトリ設定オプションの評価

モデル駆動型テレメトリを設定する際、オペレータはテレメトリに関連するすべてのコンポーネントを理解している必要があります。前述のように、トランスポート、エンコード、およびストリーミングの方向に使用できるオプションに基づいて、環境に適した組み合わせをさらに選択できます。

4つの主要コンポーネントは次のとおりです

1. トランスポート
2. 符号化
3. セッションの方向
4. YANGデータモデル

トランスポート: 前述したように、ノードはTCP、UDP、またはgRPC over HTTP/2を使用してテレメトリデータを配信できます。

TCPはシンプル化を目的とした推奨される選択肢ですが、gRPCはオプションのTLS機能を提供します。これはセキュリティの観点から追加の利点と考えられる場合があります。

エンコード： ルータは、コンパクトGPBと自己記述型GPBの2種類のGoogleプロトコルバッファでテレメトリデータを配信できます。

Compact GPBは最も効率的なエンコーディングですが、ストリームされる各YANGモデルに対して一意の.protoが必要です。自己記述GPBは効率が低いですが、キーが.proto内で文字列として渡されるため、単一の.protoファイルを使用してすべてのYANGモデルをデコードします。

セッション方向： テレメトリ導入でのセッション開始には、2つのオプションがあります。ルータはコレクタに「ダイヤルアウト」でき、コレクタはルータに「ダイヤルイン」できます。

YANGは、データモデリングの業界標準として業界で受け入れられています。シスコのプログラマビリティスタックは、この規格を使用して、ネットワーク上でできる限り迅速にエンコードおよび伝送できる構造化データセットを作成します。

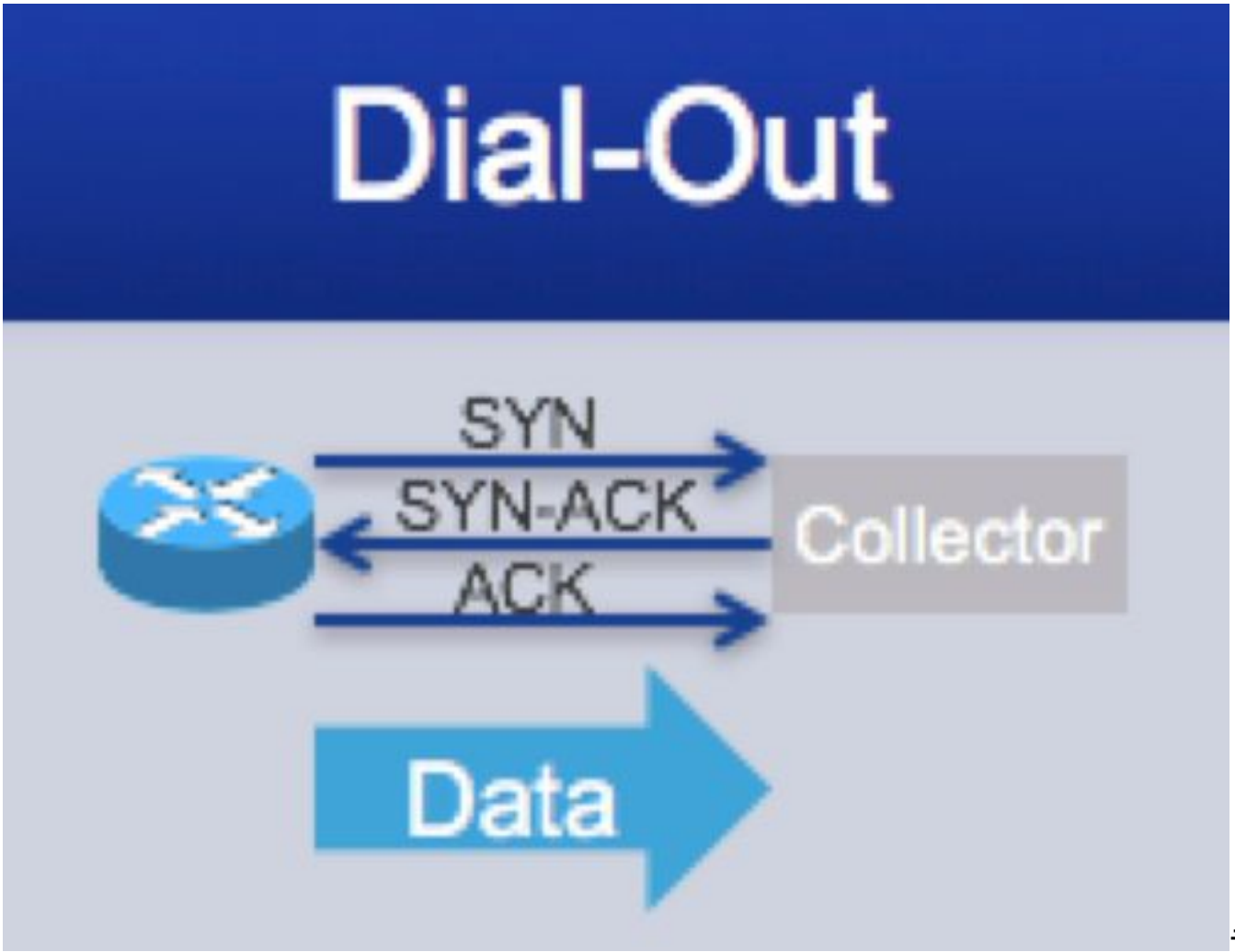
これらのデータモデルと前述の特定のエンコーディングフォーマットおよびトランスポートプロトコルを組み合わせることで、モデル駆動型テレメトリ(MDT)は分析のための完全なソリューションになります。

テレメトリの設定例

IOS-XR

ダイヤルアウト設定の分割

ダイヤルアウトモードでは、ルータはコレクタに対してTCPセッションを開始し、サブスクリプションのセンサーグループによって指定されたデータを送信します。



テレメトリダイヤルアウト設定の観点から見ると、テレメトリ設定は3段階のプロセスです。まず、ストリームする情報を特定し、Sensor Group設定でキャプチャします。次に、情報のストリーミング先となる宛先を特定し、宛先グループの設定でキャプチャします。3つ目は、前の2つの手順で特定した情報を使用して、実際のサブスクリプションを設定することです。

1. センサーグループの定義
2. 通知先グループの定義
3. サブスクリプションの定義

センサーグループの定義

Sensorグループの設定は、ストリーミングされる情報を識別します。次の設定テンプレートは、センサーグループの設定に必要な設定を提供します。

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group <Sensor-Group-Name>
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path <Sensor-Path>
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# commit
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# end
RP/0/RP0/CPU0:XR#
```

次の例は、センサーグループの設定の実際の例を示しています。

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group SensorGroup101
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path Cisco-IOS-XR-infra-statsd-
oper:infra-statistics/interfaces/interface/latest/generic-counters
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# commit
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# end
RP/0/RP0/CPU0:XR#
```

同じSensorGroup定義の一部として複数のSensor Pathを設定できます。

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group SensorGroup101
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path sensor-path Cisco-IOS-XR-infra-
statsd-oper:infra-statistics/interfaces/interface/data-rate
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path Cisco-IOS-XR-infra-statsd-
oper:infra-statistics/interfaces/interface/latest/generic-counters
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# commit
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# end
RP/0/RP0/CPU0:XR#
```

通知先グループの定義

宛先グループの設定では、情報のストリーム先となる宛先を指定します。

これには3つの主要なパラメータがあります

1. セッションの方向
2. 使用するエンコード
3. 使用するトランスポートプロトコル

次に例を示します。

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)# telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)# destination-group DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-dest)#  address family ipv4 10.1.1.1 port 5432
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#   encoding self-describing-gpb
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#   protocol tcp
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)# commit
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# end
RP/0/RP0/CPU0:XR#
```

サブスクリプションの定義

Subscriptionは、設定の最終部分としてSensor-GroupとDestination-Groupの情報をバインドします。サンプル間隔は、サブスクリプションの一部として定義されます。

次に例を示します。

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 30000
RP/0/RP0/CPU0:XR(config-model-driven-subs)#destination-id DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-subs)# commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)# end
RP/0/RP0/CPU0:XR#
```

完全な設定例

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#conf
RP/0/RP0/CPU0:XR(config)#
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group SensorGroup101
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path Cisco-IOS-XR-infra-statsd-
oper:infra-statistics/interfaces/interface/latest/generic-counters
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)#destination-group DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-dest)#address family ipv4 10.1.1.2 port 5432
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#encoding self-describing-gpb
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#protocol tcp
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 30000
RP/0/RP0/CPU0:XR(config-model-driven-subs)#destination-id DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)#end
RP/0/RP0/CPU0:XR#
```

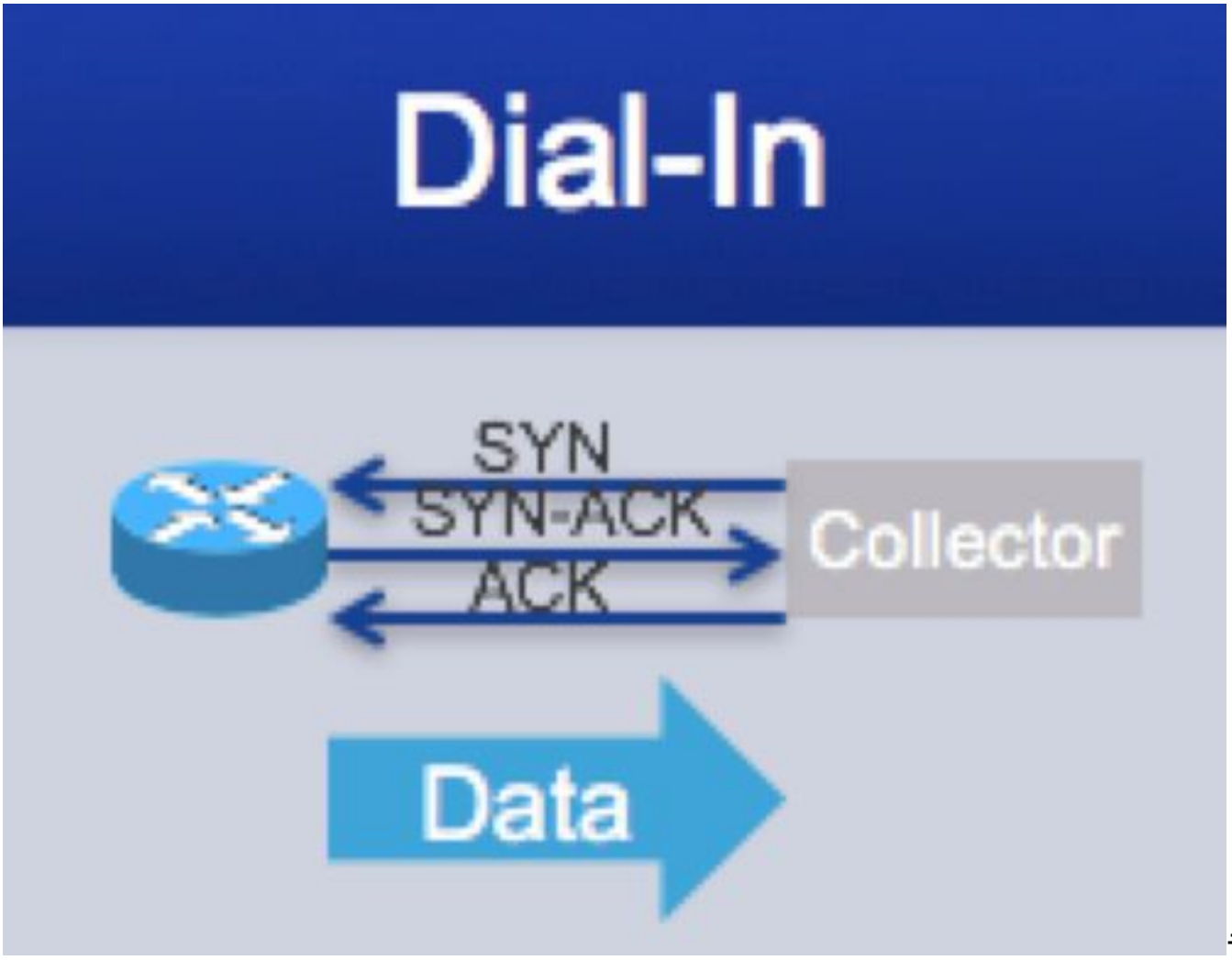
ダイヤルアウトの利点

- 転送オプションの柔軟性が向上します。
- インバウンド管理トラフィック用にポートを開く必要はありません。
- エニーキャストとロードバランシング

ダイヤルイン設定の分割

ダイヤルインモードでは、MDTコレクタ/レシーバ/オーケストレータがルータにダイヤルインし、1つ以上のセンサーパスまたはサブスクリプションに動的に登録します。ルータはサーバとして機能し、クライアントはレシーバとして機能します。

単一のセッションだけが形成され、ルータは同じセッションを通じてテレメトリデータをストリームします。このダイナミック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するか、セッションが終了すると終了します。



テレメトリダイヤルイン

コレクタはルータに「ダイヤルイン」するため、設定で各MDT宛先を指定する必要はありません。ルータでgRPCサービスを有効にし、クライアントを接続して、必要なテレメトリサブスクリプションを動的に有効にするだけです。

設定の観点から見ると、テレメトリ設定は上記と同様の3ステップのプロセスです。まず、gRPCを有効にする必要があります。次に、情報をストリーミングする宛先を特定し、セッショングループ設定でキャプチャします。3つ目は、前の2つの手順で特定した情報を使用して、実際のサブスクリプションを設定することです。

1. gRPCを有効にする
2. センサーグループの定義
3. サブスクリプションの定義

クリックして展開

ダイヤルインモードはgRPCでのみサポートされています

ダイヤルインモードはgRPCでのみサポートされています

gRPCを有効にする

まず、ルータでgRPCサーバを有効にして、コレクタからの着信接続を受け入れる必要があります。

クリックして展開

- <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。
<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。

```
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)# grpc
RP/0/RP0/CPU0:XR(config-grpc)#port 57890
RP/0/RP0/CPU0:XR(config-grpc)#commit
RP/0/RP0/CPU0:XR(config-grpc)#end
RP/0/RP0/CPU0:XR#
```

センサーグループの定義

Sensorグループの設定は、ストリーミングされる情報を識別します。次の設定テンプレートは、センサーグループの設定に必要な設定を提供します。

次の例は、センサーグループの設定の実際の例を示す、ルータCLIからの実際の例です

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group SensorGroup101
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path openconfig-
interfaces:interfaces/interface
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# commit
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# end
RP/0/RP0/CPU0:XR#
```

サブスクリプションの定義

サブスクリプションは、設定の最後の部分としてSensor-GroupとgRPCをバインドします。サンプル間隔は、サブスクリプションの一部として定義されます。

次の設定テンプレートは、サブスクリプションの設定に必要な設定を提供します。

次の例は、Subscriptionを作成し、Sensor-GroupとDestination-Groupをバインドし、さらにサンプルレートを定義しているルータCLIの実際の例を示しています。

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 30000
RP/0/RP0/CPU0:XR(config-model-driven-subs)# commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)# end
RP/0/RP0/CPU0:XR#
```

完全な設定テンプレートと例

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)# grpc
```

```
RP/0/RP0/CPU0:XR(config-grpc)#port 57890
RP/0/RP0/CPU0:XR(config-grpc)telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 30000
RP/0/RP0/CPU0:XR(config-model-driven-subs)# commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)# end
RP/0/RP0/CPU0:XR#
```

ダイヤルインの利点

- 設定とストリーミング用の単一チャネル
- ルータ/デバイスのリスニングポート
- 一時的な接続
- 現在はgRPC/gNMIのみ使用可能

イベント駆動型テレメトリ

イベント駆動型テレメトリでは、イベントが発生した場合にのみ、サブスクライブされたデータセットのデータがストリーミングされます。

イベント駆動型テレメトリの設定

イベントベースのテレメトリの設定はケイデンスベースのテレメトリと似ていますが、イベントドリブンテレメトリの設定の違いはサンプル間隔の設定だけです。サンプル間隔の値を0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ます。

完全な設定テンプレートとDIAL-OUTの例

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#conf
RP/0/RP0/CPU0:XR(config)#
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group <Sensor-Group-Name>
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path <Sensor-Path>
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)#destination-group <Destination-Group-Name>
RP/0/RP0/CPU0:XR(config-model-driven-dest)#address family ipv4 <Destination-IP>  port
<Destination-Port>
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#encoding <Encoding-Type>
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#protocol <Transport-Protocol>
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#subscription <Subscription-Name>
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id <Sensor-Group-Name> sample-interval
<0>
RP/0/RP0/CPU0:XR(config-model-driven-subs)#destination-id <Destination-Group-Name>
RP/0/RP0/CPU0:XR(config-model-driven-subs)#commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)#end
RP/0/RP0/CPU0:XR#
```

次の例は、ルータのCLIからの実際の例を示しています。

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#conf
RP/0/RP0/CPU0:XR(config)#
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
```

```
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group SensorGroup101
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path Cisco-IOS-XR-infra-statsd-
oper:infra-statistics/interfaces/interface/latest/generic-counters
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)#destination-group DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-dest)#address family ipv4 10.1.1.2 port 5432
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#encoding self-describing-gpb
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#protocol tcp
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 0
RP/0/RP0/CPU0:XR(config-model-driven-subs)#destination-id DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)#end
RP/0/RP0/CPU0:XR#
```

完全な設定テンプレートとダイヤルインの例

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)# grpc
RP/0/RP0/CPU0:XR(config-grpc)#port <port-number>
RP/0/RP0/CPU0:XR(config-grpc)telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#subscription <Subscription-Name>
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id <Sensor-Group-Name> sample-interval
<0>
RP/0/RP0/CPU0:XR(config-model-driven-subs)#commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)#end
RP/0/RP0/CPU0:XR#
```

次の例は、ルータのCLIからの実際の例を示しています。

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)# grpc
RP/0/RP0/CPU0:XR(config-grpc)#port 57890
RP/0/RP0/CPU0:XR(config-grpc)telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 0
RP/0/RP0/CPU0:XR(config-model-driven-subs)# commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)# end
RP/0/RP0/CPU0:XR#
```

SHOWコマンドによるテレメトリの検証

ルータの観点からは、各センサーグループ、宛先グループ、およびサブスクリプションに対して設定されたパラメータを確認できます

```
// ALL CONFIGURED SUBSCRIPTIONS
RP/0/RP0/CPU0:XR#show telemetry model-driven subscription

Subscription:  Subscription101          State: ACTIVE
-------------
  Sensor groups:
  Id                Interval(ms)        State
  SensorGroup101    30000               Resolved
```

```
  Destination Groups:
  Id               Encoding            Transport   State   Port    IP
  DestGroup101     self-describing-gpb tcp         Active  5432    172.16.128.3
```

```
// DETAILS ON A PARTICULAR SUBSCRIPTION
RP/0/RP0/CPU0:XR#show telemetry model-driven subscription Subscription101

Subscription:  Subscription101
-------------
  State:       ACTIVE
  Sensor groups:
  Id: SensorGroup101
    Sample Interval:      30000 ms
    Sensor Path:          Cisco-IOS-XR-infra-statsd-oper:infra-
statistics/interfaces/interface/latest/generic-counters
    Sensor Path State:    Resolved

  Destination Groups:
  Group Id: DestGroup101
    Destination IP:       172.16.128.3
    Destination Port:     5432
    Encoding:             self-describing-gpb
    Transport:            tcp
    State:                Active
    Total bytes sent:     4893
    Total packets sent:   1
    Last Sent time:       2019-11-01 10:04:11.2378949664 +0000

    Collection Groups:
  ------------------
    Id: 1
    Sample Interval:      30000 ms
    Encoding:             self-describing-gpb
    Num of collection:    5
    Collection time:      Min:     6 ms Max:    29 ms
    Total time:           Min:     6 ms Avg:    12 ms Max:    29 ms
    Total Deferred:       0
    Total Send Errors:    0
    Total Send Drops:     0
    Total Other Errors:   0
    Last Collection Start:2019-11-01 10:06:11.2499000664 +0000
    Last Collection End:  2019-11-01 10:06:11.2499006664 +0000
    Sensor Path:          Cisco-IOS-XR-infra-statsd-oper:infra-
statistics/interfaces/interface/latest/generic-counters

RP/0/RP0/CPU0:XR#
```

```
// ALL CONFIGURED DESTINATIONS
RP/0/RP0/CPU0:XR#show telemetry model-driven destination
  Group Id        IP              Port    Encoding            Transport   State
  ------------------------------------------------------------------------------
  DestGroup101    172.16.128.3    5432    self-describing-gpb tcp         Active
RP/0/RP0/CPU0:XR#
```

```
// PARTICULAR DESTINATION
RP/0/RP0/CPU0:XR#show telemetry model-driven destination DestGroup101
  Destination Group:  DestGroup101
  -----------------
    Destination IP:       172.16.128.3
    Destination Port:     5432
    State:                Active
    Encoding:             self-describing-gpb
```

```
    Transport:            tcp
    Total bytes sent:     83181
    Total packets sent:   17
    Last Sent time:       2019-11-01 10:12:11.2859133664 +0000

    Collection Groups:
    ------------------
      Id: 1
      Sample Interval:      30000 ms
    Encoding:             self-describing-gpb
      Num of collection:    17
      Collection time:      Min:     5 ms Max:    29 ms
      Total time:           Min:     6 ms Max:    29 ms Avg:    10 ms
      Total Deferred:       0
      Total Send Errors:    0
      Total Send Drops:     0
      Total Other Errors:   0
      Last Collection Start:2019-11-01 10:12:11.2859128664 +0000
      Last Collection End:  2019-11-01 10:12:11.2859134664 +0000
      Sensor Path:          Cisco-IOS-XR-infra-statsd-oper:infra-
statistics/interfaces/interface/latest/generic-counters

RP/0/RP0/CPU0:XR#

// ALL CONFIGURED SENSOR GROUPS
RP/0/RP0/CPU0:XR#show telemetry model-driven sensor-group
  Sensor Group Id:SensorGroup101
    Sensor Path:        Cisco-IOS-XR-infra-statsd-oper:infra-
statistics/interfaces/interface/latest/generic-counters
    Sensor Path State:  Resolved

// PARTICULAR SENSOR GROUPS
RP/0/RP0/CPU0:XR#show telemetry model-driven sensor-group SensorGroup101
  Sensor Group Id:SensorGroup101
    Sensor Path:        Cisco-IOS-XR-infra-statsd-oper:infra-
statistics/interfaces/interface/latest/generic-counters
    Sensor Path State:  Resolved

RP/0/RP0/CPU0:XR#
```

テレメトリ収集スタック

ルータの設定のほかに、テレメトリベースのソリューションには、コレクタ、データベース、モニタリング/分析ソフトウェアなどの複数のコンポーネントが必要でした。これらのコンポーネントは、個別に構成することも、単一の包括的な製品の一部として構成することもできます。

- 着信パケットを取得して転送し、さらに保存するには、デコーダをインストールする必要があります。
- ストリームされた情報を保存するには、時系列データベース(TSDB)が必要です。
- 内部データベースから取得したデータを視覚化するグラフィカルツールも必要です。

コレクションスタックを詳細に説明することは範囲を超えています。Cisco Crossworks Health Insightsを使用すると、デバイスにテレメトリ設定が自動的にプロビジョニングされ、テーブル/スキーマが時系列データベース(TSDB)に作成されるゼロタッチテレメトリが可能になります。データの収集とクレンジングに関する運用およびネットワーク管理のオーバーヘッドを合理化し、オペレータがビジネス目標に集中できるようにします。SNMP、CLI、およびモデル駆動型テレメトリを介してネットワークデバイスデータを収集する共通コレクタを使用することで、データの重複を回避し、デバイスおよびネットワークの負荷を軽減できます。

ネットワークにおけるテレメトリの導入に関する考慮事項

ネットワークにおけるテレメトリの導入を分析する際に考慮すべき事項を次に示します。

スケーリング

テレメトリは大量のデータをストリーミングできるため、スケーラビリティの側面を慎重に検討することをお勧めします。

必要なデータのみをストリーム配信する

各Yangモデルには複数のリーフノードがあります。必要な情報と不要な情報を具体的に指定することをお勧めします。Yangモデルを調べ、テレメトリのユースケースに必要なデータパスを特定することをお勧めします。

ストリーミングデータの量を検討する

ストリーミングされるテレメトリデータの総量は、次の点を考慮する必要があります。

1. ネットワーク内の帯域幅割り当て
2. QoS
3. アプリケーション/ソフトウェアのパフォーマンスの向上 コレクタソフトウェア時系列データベース分析/視覚化ソフトウェア
4. 符号化の効率（「テレメトリ設計ガイドライン」の項で説明）は、ストリーミングされるデータの量に直接影響します。可能な限り、コンパクトGPBを推奨します。
5. 収集間隔は、以下を含む複数の側面に直接影響します。 ネットワークの帯域幅使用率データベースのストレージ要件データをストリーミングするデバイスのパフォーマンス

アプリケーション要件に基づいて収集の頻度を評価することをお勧めします。

全体として、送信元または宛先で不要なデータをフィルタリングすることが可能であると考えられます。不要なデータをフィルタリングするオプションがあります。フィルタリングは2つのレベルで実行できます。

1. 送信元：データをストリーミングするデバイス。
2. 宛先では、コレクタがデータを収集して正規化します。（コレクタでのフィルタリングはこのドキュメントの範囲外です）。

次の例は、ワイルドカードを適用して、センサーパス内のHundered Gigインターフェイスに対してのみデータをフィルタ処理する方法を示しています。

```
sensor-path Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface[interface-name='HundredGigE*']/latest/generic-counters
```

参考資料

https://blogs.cisco.com/sp/the-limits-of-snmp

https://blogs.cisco.com/sp/why-you-should-care-about-model-driven-telemetry

https://www.cisco.com/c/en/us/td/docs/iosxr/asr9000/telemetry/b-telemetry-cg-asr9000-61x.html

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。