IOS XR7 データシート

目次

Cisco IOS XR のロックステップの進化	3
将来を見据えた設計	4
IOS XR7 の概要	5
高信頼性	7
IOS XR RPM	17
アクセス展開用の ZTP	19
強力な新しい IOS XR インストール	27
IOS XR 7 のパッケージ化とインストール	27
IOS XR7 のパッケージと SMU	27
IOS XR7 による基本インストールコマンド	29
リモートリポジトリの設定	29
IOS XR7 インストールの show コマンド	30
ワンタッチインストール	30
ゼロタッチプロビジョニング（ZTP）API	34
サービスレイヤ API	35
オープン フォワーディング アブストラクション（OFA）API	35
ハードウェアの信頼ルート	36
TAm の証明書とキー	39
セキュアブート	41
署名済み RPM	43
実行時の信頼性（IMA）	43
信頼性レポートと可視化	44
シスコの環境維持への取り組み	45
Cisco Capital	45

IOS XR7：シンプル、最新、高信頼性

これからの課題とは

世界中のサービスプロバイダー（SP）が、さし迫った課題への準備を進めています。その課題とは、今後 10 年間のネットワーク加入者とコンテンツ主導トラフィック急増への対処です。このような課題の原因の一端は、モバイル アプリケーション プラットフォームの普及と複数のデジタル コンテンツ プロバイダーの台頭による、既存のネットワークを介したビデオ/アプリケーションコンテンツの消費の増加にあります。

インターネットトラフィックの過去 5 年間の複合年間成長率は、 30% 以上に達しました。2022 年までに、次のことが予想されます。

- IP トラフィックが 1 ヵ月あたり 396 エクサバイトに増加する
- インターネットトラフィックの 1/3 が大都市サービスエリアで発生する
- 280 億台を超えるデバイス/エンドポイントがオンラインになる
- 全デバイスの 48% がビデオ対応になる[1]

来たるべき 5G 対応デバイスのリリースにより、速度が向上し、遅延が減少することで、帯域幅とネットワークパフォーマンスに対する要求は今後も増大の一途をたどるでしょう。接続されたデバイス、オンライン ゲーム プラットフォーム、ビデオ会議プラットフォームから生成される新しいタイプのトラフィックが出現するため、ネットワークに展開されるデバイスの数が大幅に増加します。この成長に対応するネットワークプロバイダーは、ネットワークのあらゆる領域において、シンプルでありながらセキュアで、しかも一貫性のある高度に自動化された運用に重点を置く必要があります。

Cisco IOS XR のロックステップの進化

Cisco® Internetwork Operating System（IOS）XR ソフトウェアは、これらの技術的移行に対応するために、綿密な予測に基づいて継続的に進化してきました。IOS XR には、クラス最高のルーティングプロトコルと機能が組み込まれる一方で、セグメントルーティングやイーサネットバーチャルプライベートネットワーク（EVPN）などのインテントベースのトランスポートテクノロジーにも引き続き重点が置かれています。これにより IOS XR は、ネットワークセグメントを横断する Web スケールのサービスプロバイダーや大規模サービスプロバイダーにとって最有力の選択肢となっています。

また、SP ネットワークが拡張性に優れた自動運用ワークフローに段階的に移行していることを認識することも同様に重要です。この移行により、業界全体のベンダー ネットワーク オペレーティング システムにおいて、ネットワークスタックのすべてのレイヤでモデル駆動型 API を使用した拡張可能なオープン ソフトウェア アーキテクチャの実装が促進されています。IOS XR はリリース 6.x で、業界の要件を満たすために役立つ運用機能に大きな重点を置きました。具体的には、32 ビット QNX オペレーティングシステムから 64 ビット Linux オペレーティングシステムに移行することにより、この方向への第一歩を踏み出しています。このような背景のもとに、プログラム制御プロビジョニングと大規模リアルタイムテレメトリデータのための YANG モデルがサポートされたことに加えて、カスタムプロトコルおよびコントローラのスタックの下位レイヤにハイパフォーマ

[1] Cisco Visual Networking Index（2018 年）

ンスのサービスレイヤ API が導入されたことで、必要なワークフローとツールを統合するための IOS XR の機能が大幅に向上しました。

IOS XR では、Day 0 展開要件に対処するためにゼロタッチプロビジョニング（ZTP）機能が導入されていますが、導入からの 4 年間で ZTP ワークフローを大きく取り込んだことが IOS XR のもう一つの特色となっています。ZTP により、オペレータは、ルータコンソールに触れることなく、また手動操作なしで、IOS XR を実行するルータをネットワークにプロビジョニングできます。これにより、「トラックオフロードからサービスアクティベーション」までの時間が短縮されます。また、デバイスを手動で設定するために特定領域のエキスパート（SME）を現場に派遣するコストも不要になります。初期ロールアウト時に ZTP を使用して節約された時間とコストは最終的な収益に直接影響し、より多くの収益を生み出すために役立つネットワークの最適化と設計へのより大きな投資を可能にします。今後、IOS XR がアクセスネットワークで広く使用されるようになれば、ZTP が、運用コスト（OpEx）を削減するとともに 5G および関連テクノロジーの要求を満たすためにネットワークをスケールアウトするうえで役立つ重要な構成要素であることが明らかになると思われます。

進化は、そこで終わってはいません。IOS XR がエンドユーザへの直接アクセスの基礎となる Linux 環境を開放したため、多数のスクリプト言語（Bash、Python、Golang、C++）と設定管理ツール（Ansible、Puppet、Chef など）が使用可能になりました。また、コンテナ（LXC および Docker）のサポートにより、オペレーターは、任意の Linux ツールをパッケージ化したり、カスタムエージェントおよびプロトコルを実装することで、それらを即座に利用できるようになりました。この設計により、SP は、IOS XR ネットワークスタックの優れた機能をネイティブ Linux 環境と組み合わせて活用し、設置されたネットワークデバイスの管理に関する運用上の複雑さをさらに軽減できます。

将来を見据えた設計

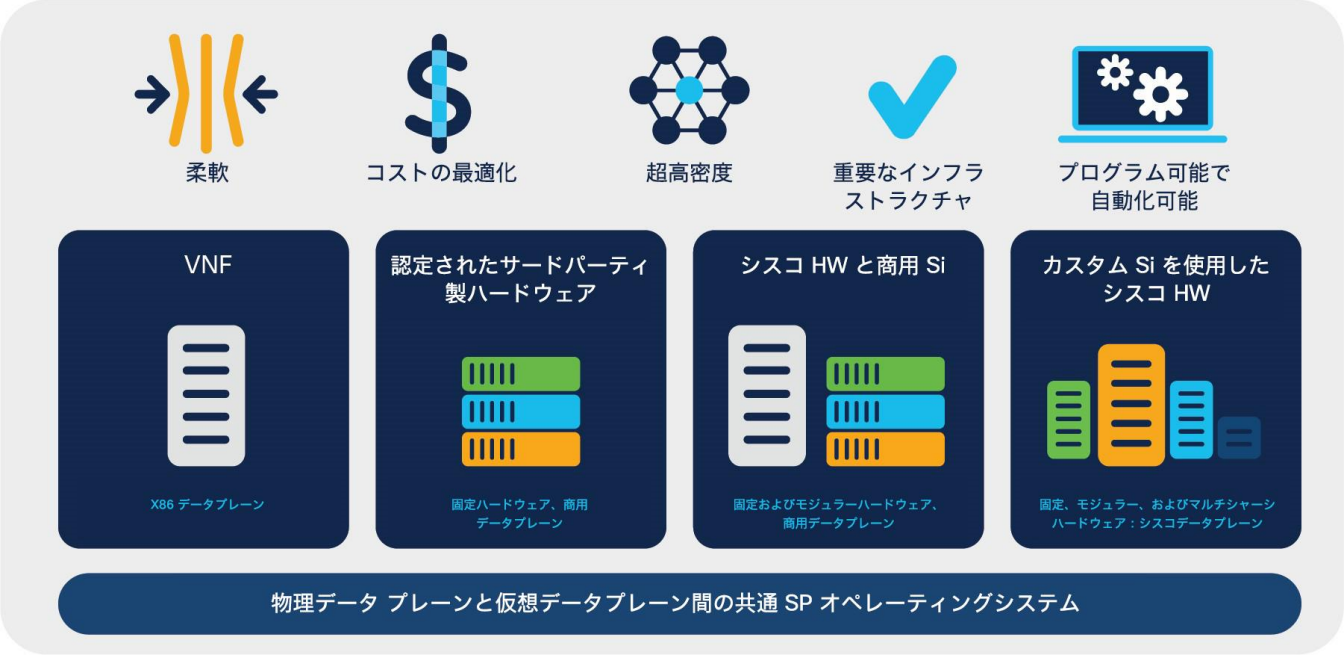
では、これからどこに向かうべきでしょうか。その第一歩は、IOS XR のリリース 6.x で導入された運用上のあらゆる機能強化を活用することです。ハイパフォーマンスの拡張可能な API による大規模な自動化については、SP から引き続き高い需要があります。さらなる進化を促進するには、大規模なストリーミングテレメトリ、Linux スタイルのワークフロー、および業界標準ツールとの統合への追加投資を継続する必要があります。

ネットワークの容量と規模が拡大しつづける場合、対処すべき明確な課題が存在します。それは、運用のさらなる合理化と大幅な簡素化です。Day 0 ZTP や Day 1 プロビジョニング/モニタリングから Day N サービス操作/管理までの展開のあらゆる段階で、自動化をより容易に実現する必要があります。

デバイスの整合性を損なう新しい攻撃ベクトルが毎年出現し、ネットワークデバイスのセキュリティに対する懸念が高まりつづけています。特にアクセスの展開において、SP ネットワークの境界が拡大するにつれて、ネットワークデバイスやプロビジョニングサービスを乗っ取り、ネットワークに侵入して、送受信されているデータを盗み取る中間者（MITM）攻撃が大きな広がりを見せています。セキュアなアウトオブバンド管理ネットワークを使用しないで実行される自動化されたデバイスのプロビジョニングにより、ネットワークの攻撃対象領域が

拡大します。これは、信頼できるネットワークオペレーションセンター（NOC）の外部で行われるケースがほとんどです。自動プロビジョニング用のセキュアなワークフローを実装し、設置されているすべてのアーティファクトを検証し、実行中のプロセスをモニタして攻撃を排除できる「信頼できる」ネットワークデバイスの作成に重点を置く必要があります。

拡張性が高く、400G に最適化されたルータを可能にする Cisco 8000 シリーズプラットフォームの発表により、IOS XR でサポートされるプラットフォームのポートフォリオは拡大しつづけています。XRv9000 仮想ルータ、NCS 5000、NCS 5500 シリーズ、ASR 9000 シリーズ、および認定サードパーティハードウェア上のディスアグリゲーション製品を使用して、仮想ネットワーク機能（VNF）ソリューションで実行することにより、IOS XR7 は、業界で最も包括的なプラットフォームとソリューションのポートフォリオを提供し、サービスプロバイダーネットワークのすべてのセグメントで一貫性のある運用と機能を実現します。



IOS XR7 の概要

IOS XR リリース 7 のアーキテクチャは、これらの要件に正面から対処できるように進化しています。IOS XR7 アーキテクチャでは、次の 3 つの主要な原則が重視されています。

シンプル

さまざまなワークフローのより容易なプロビジョニング、自動化、および統合を実現します。

1. **よりシンプルでリーンなアーキテクチャ：**IOS XR7 上の新しいハードウェアプラットフォームでは、管理プレーンとシステムコンテナが存在しないため、既製のツールとスクリプトソリューションを使用してシステムをこれまでになく容易に管理できます。

2. **よりシンプルな運用：**Linux スタイルのワークフローと統合により、拡張性に優れた設定管理ツール（Ansible、Puppet、Chef）の使用と、標準 Linux アプリケーションのバイナリまたはコンテナ（LXC、Docker）としてのオンボックスでのサポートが実現されます。

3. **よりシンプルでセキュアな Day 0 ロールアウト：**RFC 8572 に基づく強力でセキュアなゼロタッチ機能により、IOS XR ルータとブートストラップサーバの間の YANG モデルトランザクションに基づいたテンプレート駆動型 ZTP スクリプトによるセキュアなデバイスオンボーディングを実現できます。

4. **よりシンプルなソフトウェアの配信と展開：**「ゴールデン インストール ソフトウェア オブジェクト」（GISO）と呼ばれるアーティファクトにより、カスタムスクリプト、アプリケーション、パッケージ、およびファイルが展開可能な ISO アーティファクトに結合されます。個々の IOS XR コンポーネントおよびパッチは、ルータ パフォーマンス モジュール（RPM）を介して配信されます。

5. **IOS XR インストールの強力な新しい設計：**DNF（Dandified YUM）をベースに構築され、モジュラーシャーシシステムと YANG モデル API をサポートするように拡張された IOS XR インストールにより、オペレーターは、インストールプロセスのリアルタイムテレメトリ通知をサポートしながら、IOS XR RPM、ネイティブ Linux RPM、および GISO インストールのライフサイクルを管理できます。

最新

スタックのすべてのレイヤにモデル駆動型 API があります。トランスポートテクノロジーに関して業界をリードするサポートを提供します。

1. **YANG モデルの管理レイヤ API：**デバイスのプロビジョニングと管理が自動化されます。これらのモデルには、ネイティブ IOS XR YANG モデルと OpenConfig モデルが含まれます。

2. **ストリーミングテレメトリ機能：**ケイデンスベースのモニタリングまたはイベント駆動型のモニタリングにより、管理レイヤの YANG モデルパスから派生したデータを gRPC、TCP、または UDP を介してトランスポートとしてモニタできます。

3. **サービスレイヤ（SL）およびオープン フォワーディング アブストラクション（OFA）API：**サービスレイヤ API は、IOS XR の共通ネットワーク インフラストラクチャ レイヤ（RIB、ラベルスイッチデータベース、BFD、L2 など）にあるハイパフォーマンス API です。この API により、コントローラとカスタムプロトコルは、たとえば IOS XR RIB でルートを操作し、オンザフライでラベルスイッチドパスを作成できます。OFA API は、ハードウェア プラットフォームの ASIC SDK 上にあるモデル駆動型の API です。この API により、ネットワークスタックの最下位レイヤへの、モデル化されたまたは直接のアブストラクション（P4 ランタイムなど）を介したハイパフォーマンスアクセスが実現されます。

4. **セグメントルーティングと EVPN：**セグメントルーティングと EVPN は、シンプルさ、規模、およびプログラムによる拡張性を重視する IOS XR ベースのトランスポート展開の中核です。IOS XR7 では、SR Flex-Algo、SRv6 などのサポートの追加により、これらのテクノロジーが引き続き重視されています。EVPN は、レイヤ 2 VPN サービスとレイヤ 3 VPN サービスを含むあらゆるサービスタイプに関して統一されたコントロールプレーンプロトコル（BGP）を提供することで、さらに高度なシンプルさを実現します。

5. **ゼロタッチ API：**Day 0 自動化を実現するゼロタッチプロビジョニング用の包括的な API。これには、Bash および Python ライブラリをオンボックスで使用するコマンドライン インターフェイス（CLI）自動化 API が含まれます。また、IOS XR7 により、オペレータは、ZTP Python ライブラリのオンボックス NETCONF クライアントを使用し、YANG モデル XML 入力を介してシステムを起動することもできます。これはさらに、準備が整えばネットワークチームが CLI から YANG モデルにシステムを移行することを可能にします。

高信頼性

信頼できるセキュアなワークフローの実現には、高信頼性ネットワークデバイス上で実行される高信頼性ネットワーク オペレーティング システム（NOS）が不可欠です。IOS XR7 では、次を可能にすることで、これを実現します。

1. **ハードウェアから始まる信頼性：**改ざんできないセキュアなトラストアンカーモジュール（TAm）には、ハードウェアコンポーネントの既知の適正な値と、シスコをルートとするキーおよび証明書が格納されます。これらは、BIOS ブート時にハードウェアのコンポーネントを検証するために使用されます。

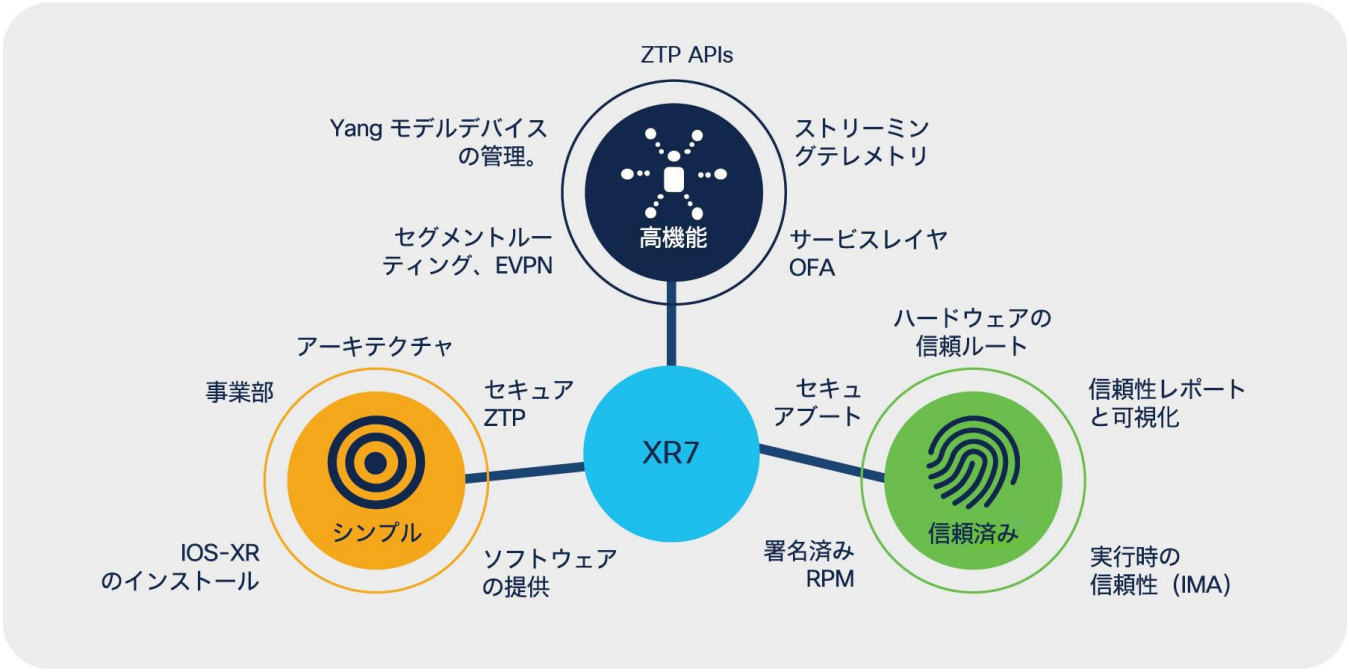
2. **セキュアブート：**セキュアブートプロセスを通じてネットワーク OS（IOS XR7）の一部を検証することにより、信頼性が強化されます。これは、OS を起動する前に、ブートローダーとカーネルモジュールの署名を TAm のキーに対して検証することで実現されます。

3. **実行時の信頼：**すべての実行時プロセス（実行時プロセスから、IOS XR またはサードパーティアプリケーションのいずれかによって起動された後続のプロセスまで）に対して IMA（整合性測定アーキテクチャ）評価チェックを実施し、すべての不一致をログに記録することにより、実行時に信頼性が維持されます。

4. **署名付き RPM：**TAm のキーに基づいて、すべての IOS XR RPM およびサードパーティアプリケーション RPM の署名をインストール前に確認することにより、それらの RPM の信頼性が強化されます。

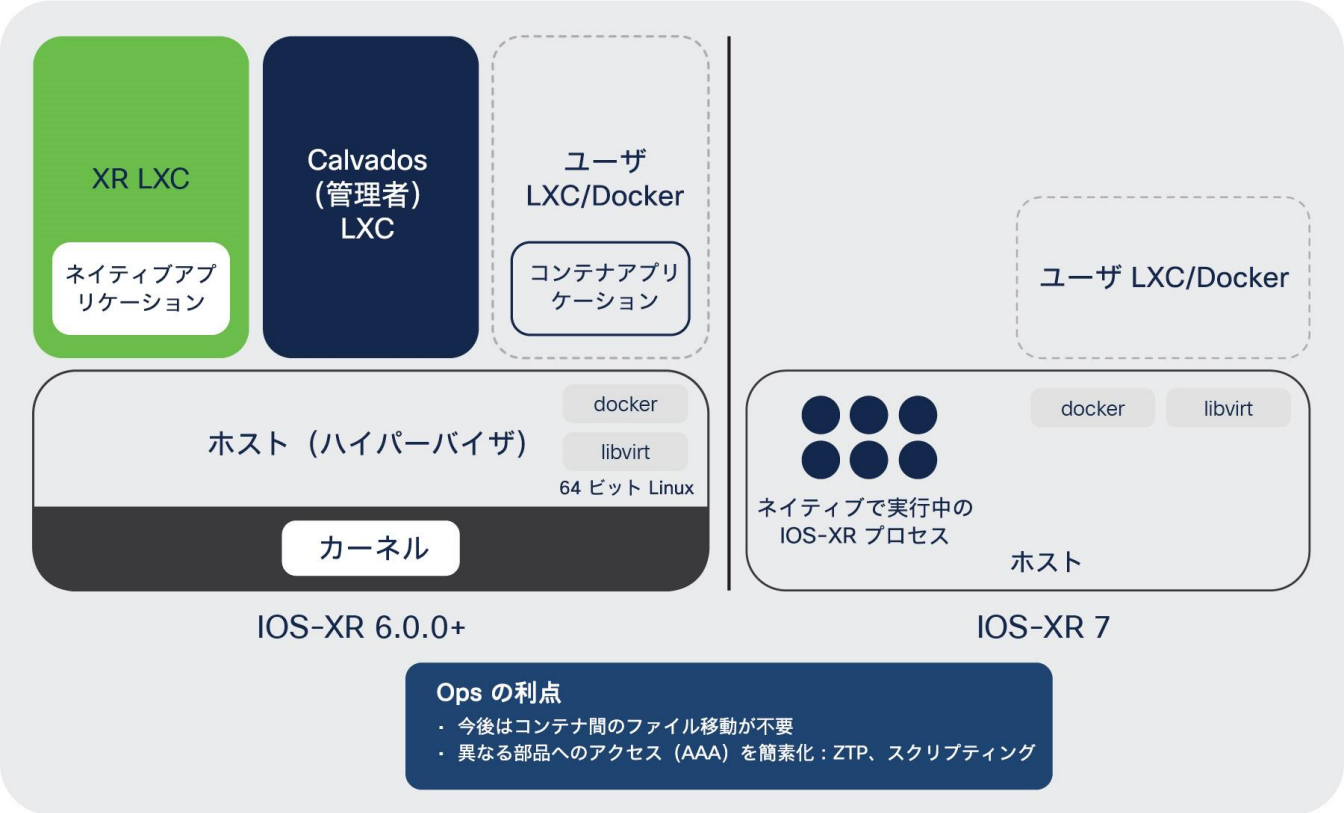
5. **信頼性に関する可視化およびレポート：**デバイスのライフサイクル中にシステムによって実行されるすべての信頼性検証操作を、リモート構成証明機能を使用して可視化およびレポートできます。

これらの概念についてさらに詳しく説明することで、将来の課題に対処しようとするサービスプロバイダーのニーズを満たすために IOS XR が進化しつづけることを明らかにします。



64 ビット IOS XR アーキテクチャについて

64 ビットの IOS XR アーキテクチャは IOS XR リリース 6.X で導入されました。これは、IOS XR が 32 ビットの QNX ベース オペレーティング システムから 64 ビットの Linux ベース オペレーティング システムに移行したためです。



このアーキテクチャは、次の要素で構成されます。

- **ホスト（ハイパーバイザ）レイヤ：**これは基礎となる 64 ビット WRL7 Linux ディストリビューション レイヤで、カーネル（バージョン 3.14）と、libvirt および Docker デーモンを生成するルートファイルシステムで構成されます。システムコンテナ（XR コントロールプレーン LXC と Calvados 管理プレーン LXC）は、デフォルトで libvirt デーモンを使用して起動されます。libvirt デーモンは、ネットワークオペレーターがホストレイヤで独自のアプリケーションを起動するためにも利用できます。Docker デーモンは、サードパーティ アプリケーションにのみ使用されます。
- **XR コントロールプレーン LXC：**XR コントロールプレーン LXC には、IOS XR コントロール プレーン プロセス（XR のすべての機能、API、プロトコルなど）が収容されます。さらに、オペレータは、WRL7 用にコンパイルされたアプリケーションを XR コントロールプレーン LXC 内でネイティブに実行することを選択できます。
- **管理プレーン（Calvados）LXC：**管理プレーン（または Calvados）LXC は、デフォルトのシステムコンテナであり、ブート後にホストレイヤで起動される最初のコンテナです。管理プレーンコンテナは、稼働状態になると、XR コントロール プレーン コンテナを起動します。**管理プレーンコンテナは、XR コントロール プレーン コンテナのライフサイクルに関与します。**
- **オプションのサードパーティ LXC/Docker：**ネットワークオペレータは、IOS XR コントロールプレーン LXC シェルの virsh および Docker クライアントを使用して、ホストレイヤで独自のカスタム LXC または Docker コンテナを起動できます。これにより任意の Linux ディストリビューションを使用して、カスタム Linux アプリケーションをコンパイルし、コンテナにパッケージ化して、ルータに展開できます。

既存プラットフォームの互換性の維持

上記のアーキテクチャによって、管理プレーンとコントロールプレーンの分離に加えて In Service Software Upgrade（ISSU）とマルチテナント機能のユースケースにより、64 ビット IOS XR の独自機能が実現します。この数年、SP は、ネットワークのさまざまな部分に組み込まれた冗長性と、コントローラ、セグメントルーティングなどの機能を介したリアルタイムのパス選択およびフェールオーバー機能のサポートによって、より復元力のあるネットワーク設計に移行しています。その結果、スタンドアロンルータの機能としてのマルチテナント機能と ISSU の必要性は、時間の経過とともに減少しました。

このような背景のもと、NCS5500、NCS1000、ASR9000 などの IOS XR によってサポートされる既存のハードウェア プラットフォームは、それらにおいて IOS XR7 の大半の新機能が有効になっている場合でも、既存の展開操作との互換性を維持するために上記のアーキテクチャを引き続きサポートします。

運用をさらに簡素化し、新しい機能を導入するために、新しいハードウェア プラットフォームの Cisco 8000 と NCS540 が、モディファイされたよりシンプルなアーキテクチャとともに導入されます。このアーキテクチャについて以降で説明します。

IOS XR7：よりシンプル、リーンでモダン

これからのための、よりシンプルなアーキテクチャの進化

IOS XR7 は、以下に示すように、Linux カーネルバージョン 4.8.28 を使用した WindRiver Linux 9（WRL9）Linux ディストリビューションをベースに構築された最新のネットワーク OS です。

```
RP/0/RP0/CPU0:ios#
RP/0/RP0/CPU0:ios#
RP/0/RP0/CPU0:ios#bash
Sun Jul 28 16:56:13.577 UTC
[ios:~]$
[ios:~]$uname -a
Linux ios 4.8.28-WR9.0.0.16_cgl #1 SMP Sun Jun 23 19:01:53 UTC 2019 x86_64 GNU/Linux
[ios:~]$
```

シンプルさと使いやすさが IOS XR7 の背後にある基本原則の 1 つです。このシンプルさは、さまざまな形で現れます。

IOS XR 6.X と比較した場合の IOS XR7 の主要な変更点の 1 つは、新しいハードウェアプラットフォームである Cisco 8000 および NCS540 における**管理プレーンの完全な廃止**です。これは、IOS XR 6.X でサポートされていた、分離された XR コントロールプレーンと管理プレーンコンテナのセットアップ（前の項で説明）が不要になったことを意味します。その結果、XR7 ソフトウェアアーキテクチャは次のようになっています。

- IOS XR コントロール プレーン プロセスは、ホスト上でネイティブに実行されます。
- システム全体（ホストレイヤ）に対して単一の Linux シェルが存在します。
- libvirt デーモンと Docker デーモンは、それぞれの virsh および Docker クライアントとともに、すべて同じ環境（ホストシェル）で実行されます。
- 管理プレーンが排除されたことで、すべてのシステムレベルおよび環境レベルの設定とアクション機能が XR に移行されます。

IOS XR7 を搭載した Cisco 8000 および NCS540 プラットフォームで廃止された管理モード：

たとえば Cisco 8000 プラットフォームでは、「 admin 」コマンドを発行すると、次の「廃止」警告が表示されます。

```
RP/0/RP0/CPU0:ios#
RP/0/RP0/CPU0:ios#
7 月 28 日(日) 07:41:32.553 UTC
WARNING: Admin mode has been deprecated. Please consult the Command Line Interface Guide.
RP/0/RP0/CPU0:ios#
```

管理モードの show コマンド：

「show environment power」や「show environment fan」などの一般的な管理モード CLI show コマンドは、XR CLI に移行されました。

管理モードの exec コマンド：

通常、システムレベルで意味を持つ管理モードの exec コマンドも、XR CLI に移行され、自動化がさらに容易になりました。たとえば NCS5500 または ASR9000 ボックスで IOS XR 6.X のボックスをリロードするコマンドは、管理 CLI シェルからの「hw-module location all reload」です。Cisco 8000 プラットフォームの IOS XR7 では、このコマンドは「reload location all」になります。

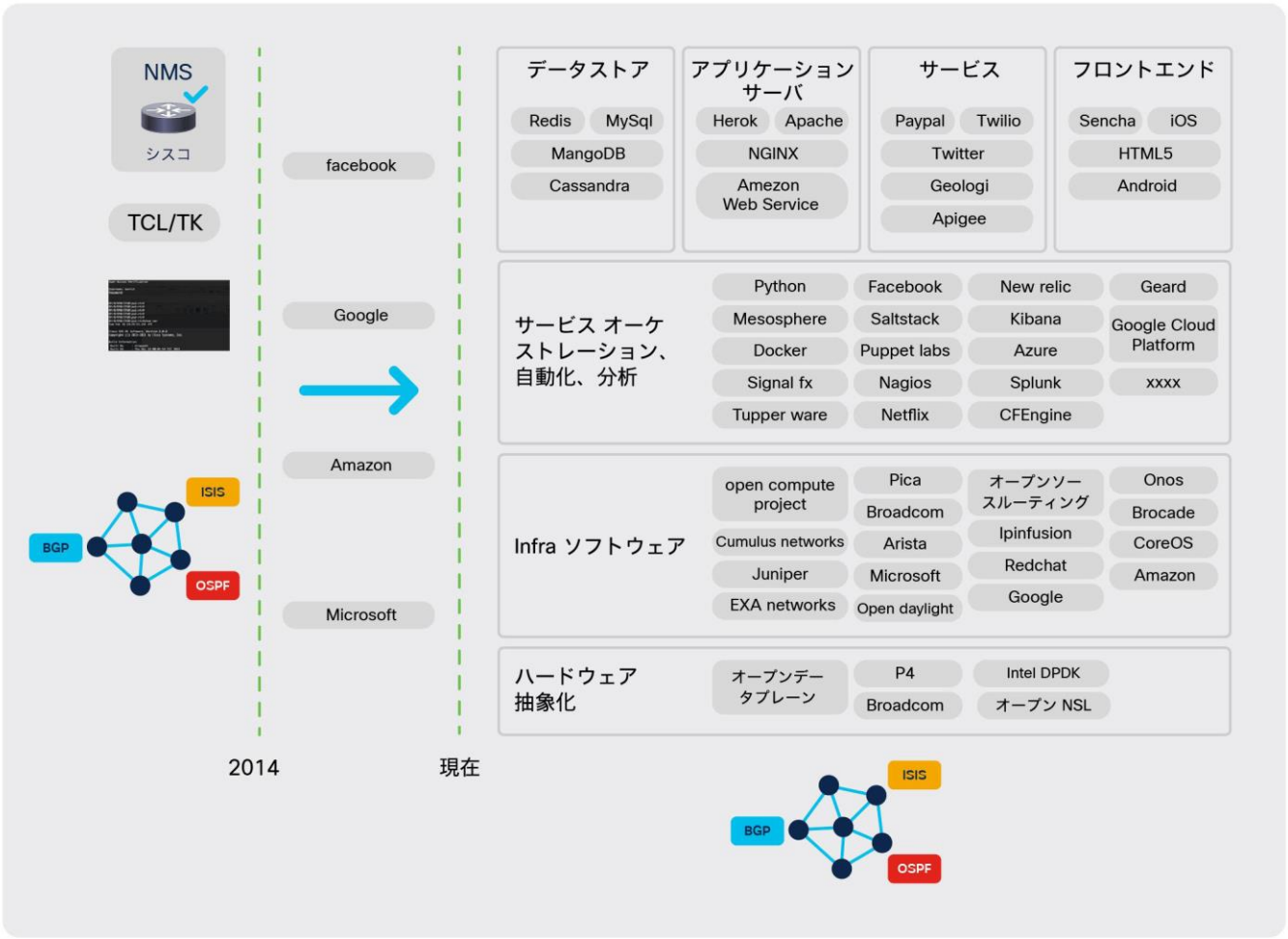
よりシンプルな運用：Linux ワークフローのサポート

過去 10 年間で、SP のネットワーク運用は、サーバの世界で生まれた Linux コミュニティ自動化ワークフローインストール手法を使用するように進化しました。この進化は Web スケールのサービス プロバイダー ネットワークで始まりました。そこでは、手動での操作をほとんど必要とせず、コミュニティ主導の自動化ツールまたは自社開発の自動化ツールを使用してプロビジョニング、管理、および操作するネットワークの作成に重点が置かれていました。

これらのワークフローは、Day0 ロールアウトにも広がりました。数週間を要する場合もあるデバイスオンボーディングの手動プロセスを、Day 0 の自動化レベルに基づいて、わずか数時間で完了するようになりました。これらのワークフローは、ZTP（ゼロタッチプロビジョニング）のカテゴリに分類されます。これについては、後の項で詳しく説明します。

SP が使用するツールについては、Telnet の expect 形式の CLI ベーススクリプトからより確定的なモデル駆動型の ssh ベースの設定管理ツールへの大幅な変更が見られました。これらのツールは、拡張性と信頼性を向上させ、多くの場合、マルチベンダー互換性を持ちます。

効果的でありつづけるために常に進化してきた IOS XR は、アプリケーション ホスティング機能およびパケット/ IO 機能を提供します。これらの機能により、Linux のアプリケーションとツールをネイティブバイナリまたはコンテナ（LXC/Docker）としてオンボックスで実行し、それらのアプリケーションが固定シャーシプラットフォームまたはモジュラ シャーシ プラットフォームで動作中にトラフィックを送受信できるようになります。さらに、Linux 環境が機能しているために、ZTP、gRPC ベース API（サービスレイヤ、Yang モデル API）、Telemetry などの機能が有効になりました。



大規模な展開の自動化

今日の Web 展開におけるネットワークデバイスとネットワーク管理者/オペレータの数の比率を調べると、その比率はすでに 3000：1 を経て増加の一途をたどっています。これは、展開が進化するにつれて、Linux 形式の自動化機能がさらに普及する一方であることを意味します。今後数年で、従来のサービスプロバイダーの多くも、Web スケールのサービスプロバイダーや大規模サービスプロバイダーが活用している手法を採用すると見られています。その結果、ネットワーク オペレーティング システムは、これらの要件を満たすように進化する必要があります。

過去 5 ー 6 年間にわたり、IOS XR は、これらの要件を満たすために絶えず進化してきました。IOS XR 6.X では、アプリケーション ホスティング機能およびパケット/IO 機能が導入されたことにより、Linux のアプリケーションとツールをネイティブバイナリまたはコンテナ（LXC/Docker）としてオンボックスで実行し、それらのアプリケーションが固定シャーシプラットフォームまたはモジュラ シャーシ プラットフォームで動作中にトラフィックを送受信できるようになりました。

さらに、Linux 環境が機能しているために、ZTP、gRPC ベース API（サービスレイヤ、Yang モデル API）、Telemetry などの機能が有効になりました。

IOS XR7 では、これがさらに進化します。重要なコンポーネントについて以降で説明します。

Linux パケット/IO のアーキテクチャ コンポーネント

Linux カーネルのインターフェイス：ルートプロセッサ（RP）とラインカード（LC）で IOS XR スタックによって検出されたすべてのインターフェイスが、IOS XR7 のアクティブ RP のカーネルにプログラムされます。これらのインターフェイスにより、Linux アプリケーションは支障なく動作し、通常の Linux システムと同様に機能します。これは、既製の Linux アプリケーションを IOS XR 上で（または IOS XR に対して）実行できるようにするために不可欠です。これによりオペレータは、既存のツールを使用して IOS XR で展開を自動化できます。

Linux カーネルのルート：IOS XR7 の初期リリースでは、アプリケーションによるトラフィックの送受信を可能にするために必要な基本ルートのみが、カーネルにプログラムされます。近日中に、拡張機能が導入される予定です。

カーネルの一部である Linux ネットワークスタックは、通常の Linux アプリケーションがパケットの送受信に使用します。Linux ネットワークスタックが XR ネットワークスタックを調べて使用することを可能にするために、基本ルートを Linux カーネルにプログラムします。

1. **デフォルトルート：**デフォルトルートは、カーネルからの不明なサブネット宛のトラフィックを、ルーティングのために IOS XR ルーティング情報ベース（RIB）/転送情報ベース（FIB）に送信します。

2. **ローカルルート/接続されたルート：**これらは、管理ポートや運用/データポートなどのインターフェイスで設定されたサブネットに関連付けられたルートです。

Linux カーネルの VRF：IOS XR で（CLI または YANG API を介して）設定された VRF は、カーネルに自動的に同期されます。カーネルでは、これらの VRF はネットワーク名前空間（netns）として示されます。この機能により、Linux アプリケーション/プロセスをアウトオブバンド管理 VRF などの特定の VRF に分離して、その VRF のインターフェイスでのみソケットを開いたり、トラフィックを送受信したりすることができます。

Linux カーネルに同期されると、すべての VRF は、VRF と同じ名前でありながら先頭に「 vrf- 」という文字が付いた netns としてプログラムされます。IOS XR のデフォルト VRF の名前は「 default 」です。そのため、Linux カーネルでは、これは「vrf-default」としてプログラムされます。同様に、「 blue 」というカスタム VRF の場合、作成される対応する netns は「 vrf-blue 」になります。

Linux ソケットを開く機能：必要なインターフェイスとルートがカーネルにすでに示されているため、アプリケーションがソケット（TCP/ユーザデータグラムポート（UDP））を開くことを妨げるものはありません。IOS XR インフラストラクチャは、これらのソケットエントリのプログラミングを処理し、ラインカードにも情報を配信します。その結果、ラインカードインターフェイスで受信されたパケットをアクティブな RP のカーネルにパントし、ソケットを開くアプリケーションにパケットを送信できます。

トラフィック送受信機能：ソケットを開く機能とカーネル内のルートを活用する機能の組み合わせにより、Linux アプリケーションはトラフィックを制約なく送受信できます。

新機能：Linux 管理インターフェイスのサポート

IOS XR7 の最新機能の一つは、IOS XR のインターフェイスを完全に「Linux 管理」する機能です。デフォルトでは、IOS XR のすべてのインターフェイスが IOS XR 設定（CLI/YANG-API）を介して管理され、インターフェイスの属性（IP アドレス、マルチテナントユニット（MTU）、状態）は XR のインターフェイスの対応する設定と状態から継承されます。

一方、「Linux 管理」インターフェイスは、その名前が示すように、Linux によって管理されます。そのようなインターフェイスの場合、その属性と状態はすべて、一般的な Linux コマンド（iproute2 ユーティリティ、ifconfig など）を使用して管理でき、これらの Linux ベースの入力は、IOS XR 反映インターフェイスの信頼できる情報源になります。

言い換えると、1 つまたは複数のインターフェイスを Linux 管理に設定でき、Linux サーバで使用される標準の自動化ツールを使用して IOS XR のインターフェイスを管理できます。これは、システムで IOS XR ネットワークスタック機能を有効にしつつ、そのシステムを運用上 Linux ボックスのように動作させるための大きな一歩です。

トラブルシューティング ツール：パケットキャプチャ

IOS XR7 のパケット/IO インフラストラクチャによってもたらされる最も重要な機能強化の一つは、tcpdump および pcap ライブラリを使用して IOS XR コントロール プレーン トラフィックをキャプチャする機能です。

これは、オペレータおよび自動化ツールが tcpdump とそのオプションを容易に活用してシステム上のコントロールプレーンの問題をキャプチャおよびデバッグするために役立つ大きな前進です。使用するユーティリティは、XR シェルに「 xr_tcpdump 」としてパッケージ化されていて、標準の tcpdump および pcap ライブラリから得られるため、tcpdump と同じオプションがサポートされます。オペレータは、このユーティリティを使用して、IOS XR カーネルのインターフェイスでパケットをキャプチャできるようになりました。これは、一般的な Linux ボックスとまったく同じ方法です。

注： データプレーントラフィックは、Linux シェルの xr_tcpdump の使用ではサポートされません。

これは、IOS XR の豊富な機能を標準 Linux スタックにもたらし、オペレータが Linux ワークフローと自動化ツールを使用して IOS XR ノードの管理を開始するとともに、tcpdump などの標準 Linux 手法（上記を参照）を使用してトラブルシューティングすることを可能にするため、IOS XR の進化の重要な一歩となります。

柔軟なソフトウェア配信（ゴールデン ISO、RPM）

ゴールデン ISO（GISO）

IOS XR7 を使用すると、ネットワークオペレータの運用上のニーズに合わせて、ソフトウェアをさまざまな形式で配信できます。その発想は、サーバ群で使用されている Linux ディストリビューションの基本バイナリアーティファクト（ISO）を取得して開き、すべてのサーバでベースとして使用されるパッケージおよび設定ファイルを追加して、その組み合わせから新しい ISO を作成するというものでした。この組み合わされた ISO は、サーバマシンに使用される「ゴールデン」設定を表すため、「ゴールデン ISO 」と呼ばれます。ゴールデン ISO の作成後、設定管理ツールが引き継いで、新しくブートされるデバイスでロールベースの環境をセットアップするまでは、サーバは PXE ブートされ、そのすべてが基本「ゴールデン」状態になります。

IOS XR7 には、ゴールデン ISO ビルドプロセスに導入されるいくつかの新機能があります。GISO プロセスの進化を以下に示します。

IOS XR7 では、GISO ビルドプロセスにより、次のアーティファクトを基本 ISO に追加できます。

1. **IOS XR 機能 RPM：**BGP、OSPF、ISIS、Telnet などの、IOS XR のさまざまなオプションコンポーネントの RPM です。
2. **基本設定：**有効な IOS XR 設定ファイルをゴールデン ISO に追加できます。
3. **ztp.ini ファイル：**IOS XR7 では、IOS XR ZTP ワークフロー用に新しいタイプのファイルが導入されます。ztp.ini ファイルは、特定のデフォルト設定を ZTP プロセス用に設定できるようにすることを目的としています。これによりユーザは、サポートされているノブを介して ZTP ステートマシンに影響を与えることができます。

4. **サードパーティ アプリケーション RPM：**サードパーティの Linux アプリケーションは、IOS XR のゴールデン ISO ビルドプロセスでサポートされます。このために Linux アプリケーションを、基盤となる WindRiver Linux 9（WRL9）ディストリビューション用にコンパイルして RPM にパッケージ化し、「その RPM に署名して」、IOS XR インストールプロセスがブート後に RPM をインストールできるようにする必要があります。

5. **セキュア ZTP アーティファクト：**セキュア ZTP アーティファクトについては、この後の項で詳しく説明します。セキュア ZTP が機能するための基本要件は、ブート時にボックスに所有者（顧客/ネットワークオペレータ）の証明書をインストールすることです。TLS クライアント証明書（オプション）を、セキュア ZTP で必要なワークフローに基づいてパッケージ化することもできます。セキュア ZTP 強制フラグは、ZTP プロセスのステートマシンを変更するためにユーザが指定できる設定です。これにより、セキュア ZTP を強制するか、下位互換性のあるクラシック ZTP ワークフローを有効にできます。

IOS XR のゴールデン ISO のビルドツールはオープンソースであり、次の場所にあります：
https://github.com/IOSXR/gisobuild

IOS XR7 ゴールデン ISO (GISO) イメージの展開：

GISO イメージがオフボックスで作成されると、次に示すようなさまざまな手法を使用して、サポートされているプラットフォームに展開できます。

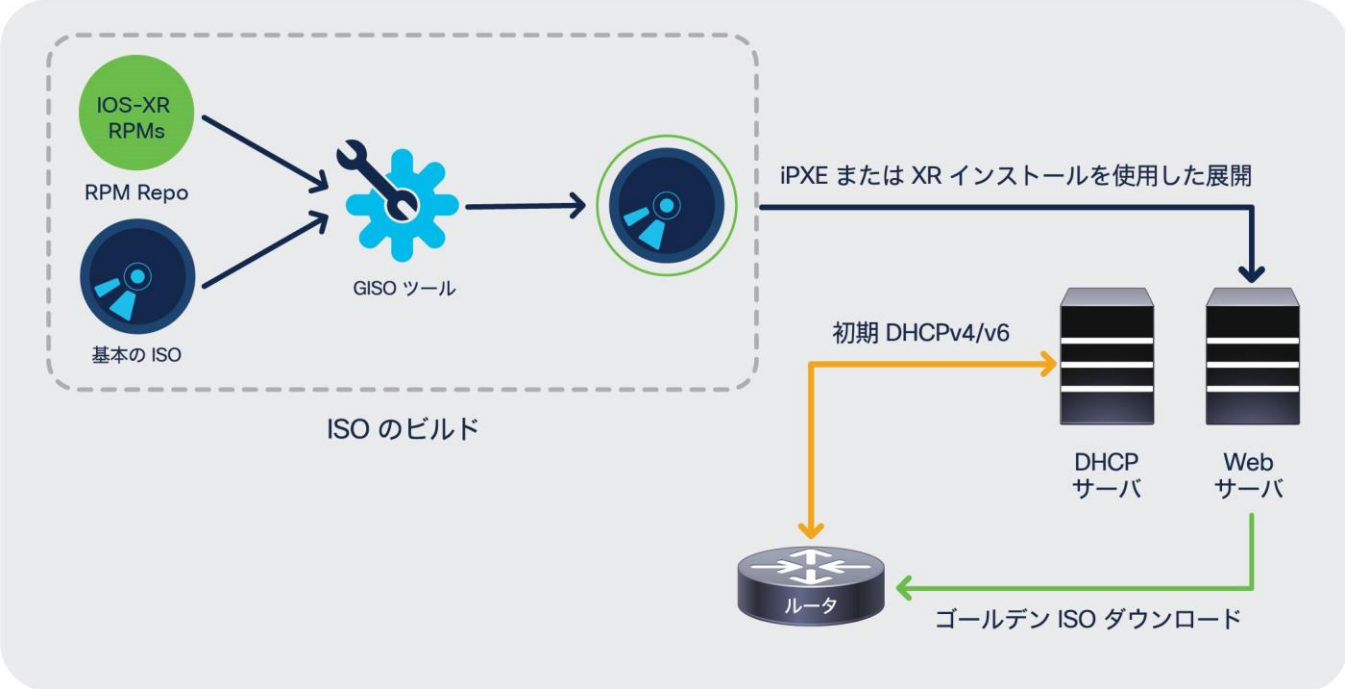
- **iPXE を使用した GISO の展開：**IOS XR と互換性のあるシスコのすべてのベアボーンプラットフォーム（インストールされたイメージなし）は、デフォルトのブートローダーとして iPXE をサポートします。iPXE により、オペレータは、アウトオブバンド管理ネットワークを介して、DHCP サーバを利用し、iPXE を実行するプラットフォームによってダウンロードできる GISO の場所をポイントできます。iPXE について詳しくは、次の場所を参照してください：https://xrdocs.io/device-lifecycle/tutorials/2016-07-27-ipxe-deep-dive/
- **USB ブートを使用した GISO の展開：**サービスプロバイダーによって特にアクセスネットワークで非常に頻繁に使用されるもう一つの手法は、書き込まれたイメージを含む USB を使用して現場のデバイスをブートストラップすることです。必要なアーティファクトを含む GISO イメージは、他の IOS XR イメージと同様に USB に書き込むことができます。完了したら、ブートローダーメニューから USB ブートを選択して GISO をインストールするために、ボックスの電源を入れて再起動する必要があります。このワークフローを次に示します。
- **IOS XR の install replace を使用した GISO のインストール：**IOS XR 6.X の一部として IOS XR インストールに導入された重要な拡張機能の一つは、XR で install replace を発行し、イメージの場所をポイントするだけで、システム上の既存のソフトウェアを別のイメージで置換できるようにすることでした。このプロセスでは、既存のソフトウェアを新しいイメージに含まれる基本 ISO + RPM + サポートされるファイルに置き換えるだけで、新しいソフトウェアがアクティブになります。XR の「install replace」コマンドが、オペレータによってコンパイルされた GISO をポイントしている場合、既存のイメージが GISO のコンテンツによって置き換えられます。これについては、後で説明します。

IOS XR RPM

IOS XR7 では、すべての IOS XR コンポーネントが基本的に RPM です。これらの RPM は次のいずれかです。

- **オプションの RPM**：デバイスをリロードせずにプロセスを再起動するだけで、システムから削除したりアップグレードしたりすることができます。BGP や ISIS などの一連のコンポーネントは、オプションの RPM です。展開とユースケースに基づき、システム内のより多くのコンポーネントが、時間の経過とともにオプションの RPM になる予定です。
- **必須の RPM**：個別にアップグレードできますが、デバイスのリロードが必要になる場合があります。これらの RPM はシステムから削除できません。

これらの RPM では、IOS XR のリリースごとのセマンティックバージョンが使用されます。これにより、どちらも 1 つ以上のアップグレードされた RPM として提供される新機能とバグ修正の提供が大幅に簡素化されます。また、この結果として、新しいハードウェア プラットフォーム上で IOS XR7 を使用する場合、ソフトウェア メンテナンス アップグレード（SMU）は不要になります。その代わりに、バグ修正は単にシステム上の既存の RPM のアップグレードバージョンになります。



セキュア ZTP による、よりシンプルでセキュアな Day 0 ロールアウト

IOS XR7 では、RFC 8572（https://tools.ietf.org/html/rfc8572）に示されている要件をサポートするために、XR のゼロタッチプロビジョニング（ZTP）が拡張されます。これは「セキュア ZTP」と呼ばれます。

前述のように、ゼロタッチプロビジョニングは、今日使用されている SME およびステージング施設に関連する OpEx を削減するためにルータの Day 0 プロビジョニングを自動化しようとしているほとんどのサービスプロバイダーにとって重要なパズルのピースです。SP、特に要件の展開とセキュリティの問題が非常に深刻なアクセスネットワークにとっての ZTP の重要性について以下で詳しく説明します。

アクセス展開の ZTP 要件について

XR7 では、IOS XR はすべて、NCS540 などのアクセスプラットフォームで利用可能になるように設定されています。IOS XR をすでにサポートしている NCS5501 や NCS5502 などの既存の 1RU および 2RU プラットフォームは、アクセス展開で一般的に見られるユースケースにも適用できます。

ほとんどのアクセス展開では、ネットワークオペレータは、購入したルータを受け取り、通常、最初の「トラック」ロールアウトの一環として、それらを事前ステージング施設に渡します。事前ステージング施設では、デバイスが、通常、SME の助けを借りて手動で設定され、それらにブートストラップ設定が適用されます。これらの事前設定済みのボックスは、その後、インストールサイトに送られます。このようなステージング済みのデバイスは、多くの場合、サードパーティのインストーラを使用して簡単にセットアップし、電源を入れることができます。これが 2 番目の「トラック」ロールアウトに該当します。これは、すべてのネットワークオペレータの正確なワークフローとは言えないかもしれませんが、それらの大部分で使用されているワークフローは、ほぼこのようなものです。

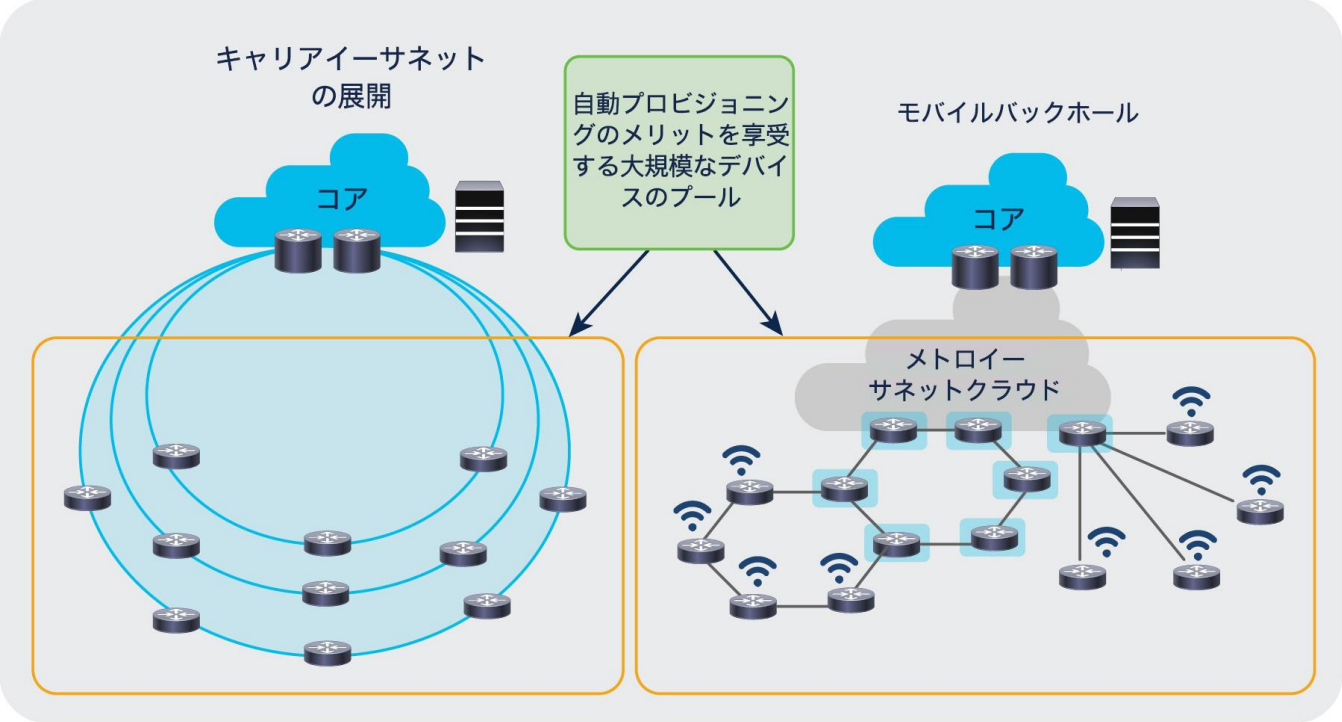
事前ステージング施設での有能な SME の確保、複数のトラックロールアウト、インストール後のロールアウト、およびブートストラップ設定にエラーがあった場合の修正により、ほとんどの大規模サービスプロバイダーで OpEx（運用コスト）が継続的な増加しています。これらの展開におけるデバイスの数が、消費者の需要と新しい 5G アーキテクチャに対応するために増えつづけていることから、OpEx コストを最大限に削減できる手法を見つけることが不可欠になっています。

これらの展開に関連する懸念事項の一部は、次のとおりです。

- OpEx を抑制するために、SME への依存（通常は、事前ステージング施設で、またはインストール後の修正を実施するために現場で）を削減できるか。
- 事前ステージング施設自体を排除できるか（これができれば理想的）。
- これらのデバイスの多くは、これらのデバイスをホストするセキュアな NOC がない現場に送られる。それらは、道路の脇、電柱の上、または同じ場所に配置されるセキュアではない NOC に展開される場合もある。信頼性とセキュリティは、アクセスネットワークにデバイスを展開する際に考慮する必要がある 2 つの重要なパラメータである。

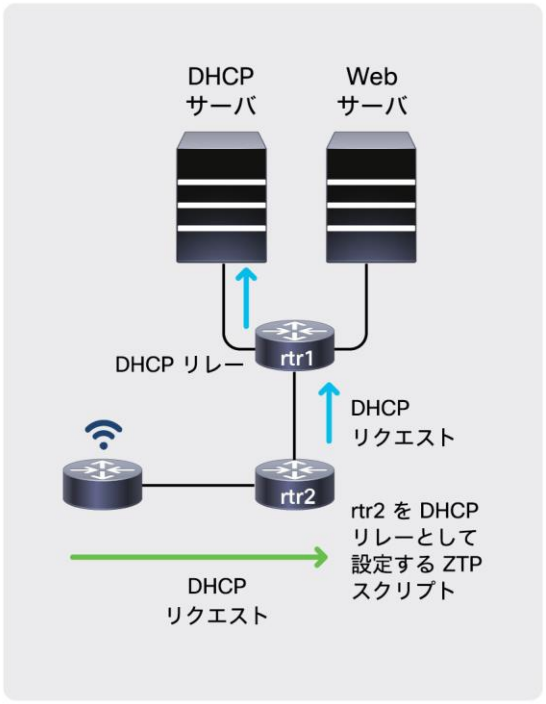
アクセス展開用の ZTP

アクセスネットワーク内でデバイスのゼロタッチオンボーディングが発生する規模は、次の図を見ると明らかです。

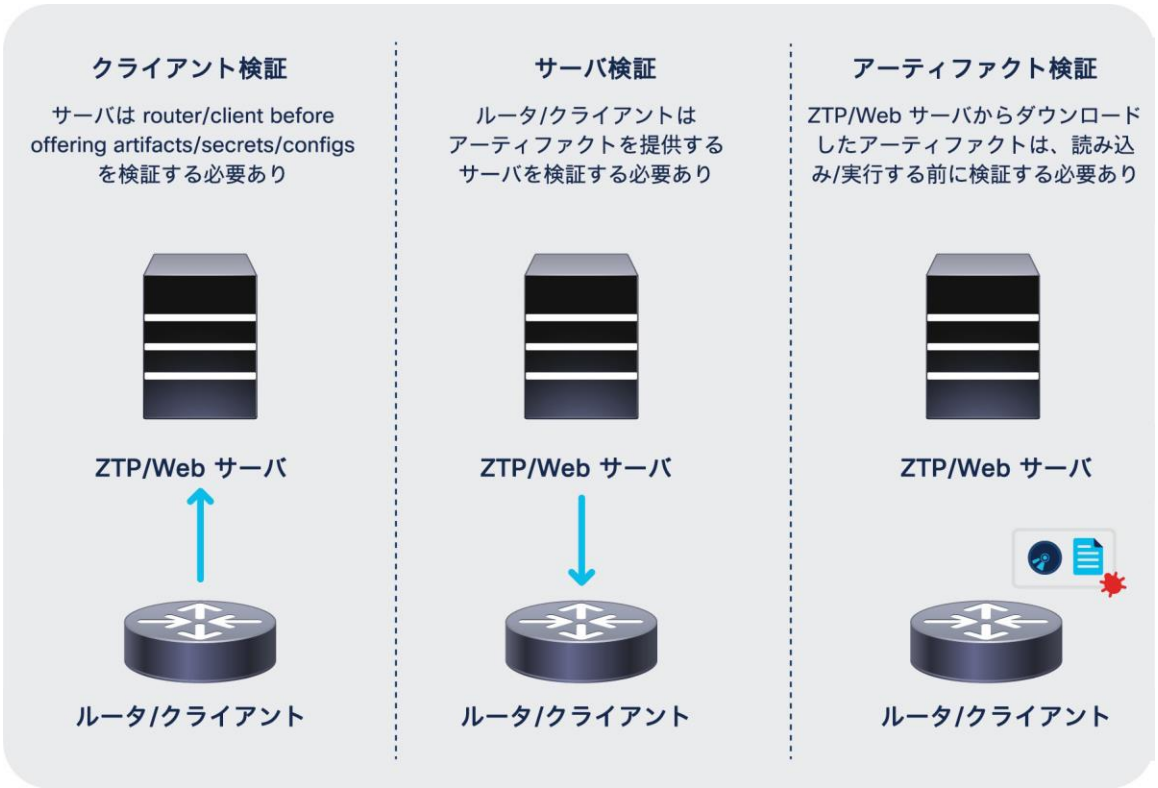


このため、IOS XR7 ZTP は、次の要件に対処できるように設計されています。

1. **ツリーベースのビルドアウト：**トポロジに依存しないようにするには、ツリーベースのビルドアウトが必要です。併用されるアウトオブバンド管理ネットワークは存在しません。そのため、ネットワーク内のすべてのデバイスが DHCP サーバへの L2 接続を持つわけではありません。そこで、すでにプロビジョニングされたデバイスがツリー内の次のデバイスの DHCP リレーとして機能する必要があります。これが、各デバイスで実行される ZTP スクリプトで使用する必要があるロジックです。つまり、デバイスは、ダウンストリームに接続されたデバイスの DHCP リレーに転換されます。



2. セキュリティは非常に重要です。アクセスデバイスは、通常、セキュアではない場所に移動するため、オンボーディングプロセス中に信頼性を確立する必要があります。大まかに言うと、これは次の要件に変換されます。



3. **VLAN 検出による L2 到達可能性：**多くの場合、DHCP サーバや Web サーバなどの初期到達可能性サービスは、モバイルバックホールトポロジなどのレイヤ 2 メトロイーサネットクラウドを介して

のみ到達可能であることがあります。アウトオブバンド管理ネットワークがないため、これらのネットワークの ZTP にはデータ（実稼働）ポートを使用する必要があります。これらのことから、レイヤ 2 クラウド全体で使用中の VLAN を検出し、正しい VLAN で DHCP 要求を適切にタグ付けする（これにより、検出後に ZTP で処理が進められる）VLAN 検出手法が IOS XR7 ZTP によって実装されます。

RFC 8572 のサポート：セキュア ZTP

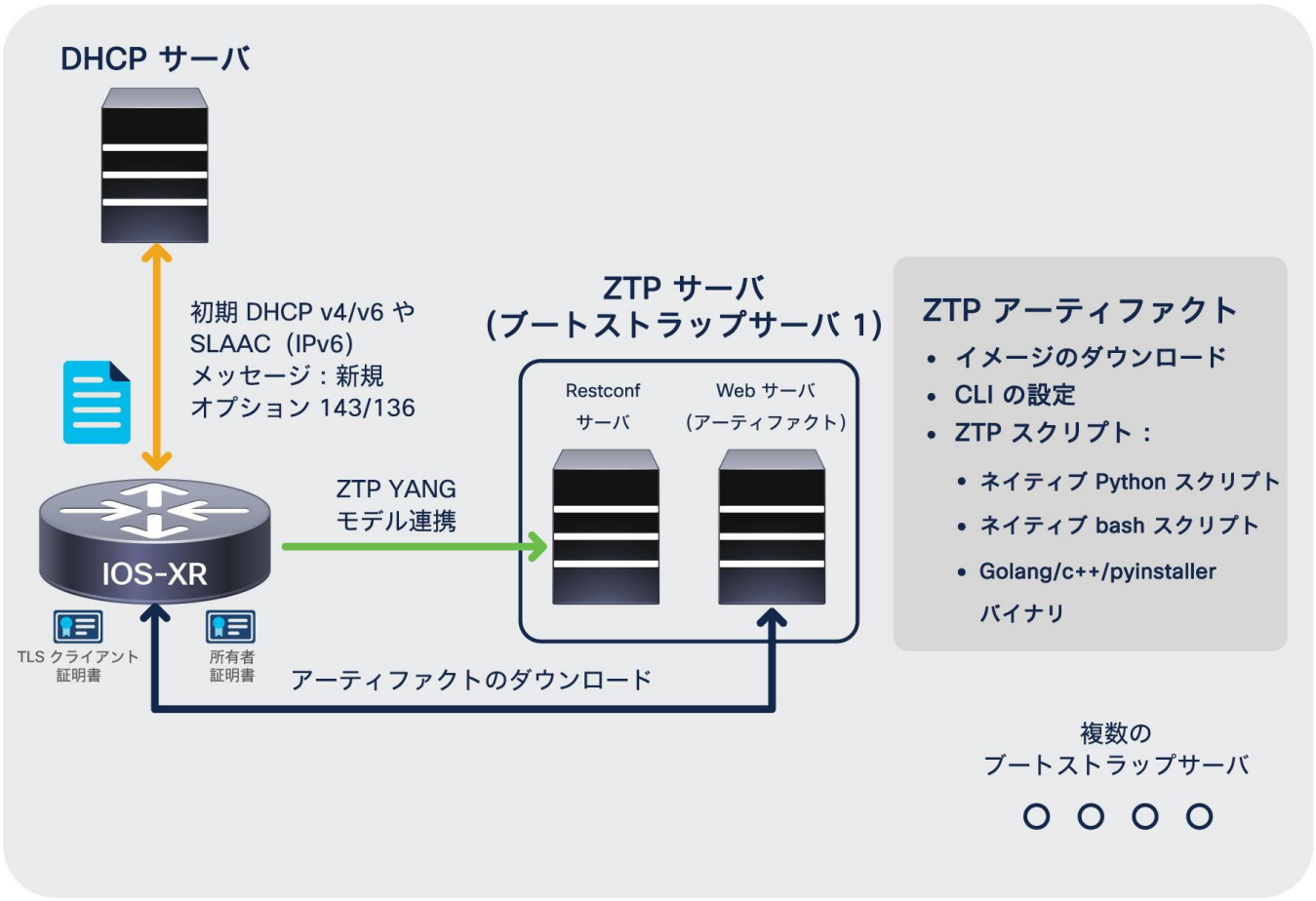
上記の目標を考慮して、IOS XR7 では、RFC-8572 の概念が実装され、IOS XR でセキュア ZTP 機能が実現されます。このサポートは次の 3 つで構成されます。

1. **セキュア ZTP YANG モデルのサポート：**これによりルータと RESTCONF ベース ZTP サーバの間の相互作用が実装されて、アーティファクト（設定、スクリプト）がダウンロードされ、ルータが自動的にプロビジョニングされます。

2. **セキュアゴールデン ISO のサポート：**これにより、所有者証明書（ネットワークオペレータに関連付けられた）をパッケージ化し、ダウンロードされたアーティファクトの署名を検証して、デバイスが ZTP サーバからの応答に基づいて参加しようとしているネットワークドメインを検証できるようになります。セキュア GISO は、オプションの TLS クライアント証明書の追加にも役立ちます。これにより ZTP サーバへの TLS 接続を確立して、後続のやり取りを保護および暗号化することが容易になります。

3. **改ざんされないセキュア固有識別子（SUDI）を中継するオンボックス インフラストラクチャ：**これは、証明書とシリアル番号の組み合わせを ZTP サーバに中継し、所有者の証明書とキーを保存するためのルータ上のセキュアな場所を提供するオンボックス インフラストラクチャです。

IOS XR7 のセキュア ZTP の展開に必要なネットワーク インフラストラクチャを以下に示します。ZTP サーバ（ブートストラップサーバとも呼ばれる）は、RFC 8572 のガイドラインに従って RESTCONF サーバを実装して、デバイスと通信し、デバイスを検証し、署名されたアーティファクトで応答して、デバイスをプロビジョニングします。



このアーキテクチャとワークフローの基本コンポーネントは次のとおりです。

1. **トラストアンカーモジュール（TAm）のデバイス SUDI：**ルータをシスコのデバイスとして一意に識別するために、IOS XR7 でサポートされるすべてのプラットフォームは、ハードウェアに改ざんされないトラストアンカーモジュール（TAm）が搭載されて出荷されます。デバイスの信頼性とセキュリティに関する後続の項で、TAm の機能と内部に格納されるさまざまなキーおよび証明書について詳しく説明します。ZTP で特に重要になるのは、Cisco-CA にルーティングされるセキュア固有デバイス識別子（SUDI）証明書です。この証明書は、シリアル番号とともに一意の公開キーを含み、デ

バイスと外部サービスのやり取りにおいてデバイスの暗号化されたセキュアな ID として使用されます。

2. **セキュアゴールデン ISO イメージ：**セキュア GISO イメージは、以前の項で説明した通常の GISO 手法を使用して作成されます。ネットワークオペレータは、証明書（「所有者証明書」と呼ばれ、選択した CA にルーティングされる）を GISO イメージにパッケージ化します。さらに、オプションの TLS クライアント証明書を ZTP サーバへの TLS 接続用のイメージにパッケージ化することもできます。

3. **SZTP DHCP オプション：**ルータによってダウンロードされるアーティファクトをホストする ZTP サーバを検出するには、ルータが、ZTP サーバへのリダイレクト URL を何らかのソースから学習する必要があります。このソースとしては、DHCP、DNS、または USB ベースのものがあります。通常、DHCP サーバは、DHCPv4 オプション 143 または DHCPv6 オプション 136（どちらのオプションも RFC 8572 で SZTP DHCP オプションとして定義されている）を使用して、「信頼できない」リダイレクト URL をブートストラップ/ZTP サーバに提供するために使用されます。

4. **ブートストラップサーバ/ZTP サーバ：**ZTP サーバは、SZTP YANG モデルを使用してルータとやり取りし、そのルータに設定/スクリプトアーティファクトを配信する RESTCONF サーバで構成されます。

   注：IOS XR7 エコシステムの一部として、**Cisco Crosswork Network Automation プラットフォーム**（https://developer.cisco.com/docs/crosswork/）は、ZTP サーバ機能をシスコからのサポートされたサービスとして提供します。これにより、ダッシュボード、スクリプト、テンプレート管理などによる Day 0 自動化が実現されます。

5. **SZTP YANG モデル：**RFC 8572 では、ルータと ZTP サーバの間でのモデル駆動型のやり取りに関する YANG モデルが定義されています（RFC 8572：https://tools.ietf.org/html/rfc8572 を参照）。特に注目すべきは、ハードウェア（HW）モデル、OS 名、バージョン、ナンス（オプション）などのルータ固有情報を含む、ルータから ZTP サーバへの最初の要求です。ZTP サーバからの応答では、

オンボーディング情報がエンコードされます。この情報は、デバイスによってフェッチされ、それ自体が再イメージ化またはプロビジョニングされます。

セキュア ZTP：信頼モデルとは

セキュア ZTP の前提は、完全に信頼できるデバイスをオペレータのネットワークにオンボードするために 3 つのタイプの検証を行う必要があるということです。

1. **ルータ/クライアント検証：**ZTP サーバは、最初の YANG 要求でデバイスによって提供される SUDI 証明書を検証する必要があります。

2. **サーバ検証：**この目的のために、セキュア GISO イメージの一部としてボックスにロードされる所有者証明書は、GISO のインストール時または ZTP サーバとのやり取りにおいて、デバイスに配信された「所有者バウチャー」またはドメイン証明書に対して検証されます。これは、ルータが正しいネットワーク「ドメイン」に参加していることを検証するために役立ちます。

3. **アーティファクト検証：**ZTP サーバからの YANG 応答に含まれるオンボーディング情報で指定されたアーティファクトは、次の 4 種類のアーティファクトを識別できます。

    a. **ブートイメージ：**ブートイメージの URL は、オンボーディング情報で指定されます。この URL からダウンロードされるブート イメージ アーティファクトは、UEFI 仕様 2.3.1 で規定されているセキュアブートの原則を使用してデバイスで安全にブートされる個別に署名されたアーティファクトである必要があります。シスコのハードウェアプラットフォームでは、さらに一歩進んで、ハードウェアベースのセキュアブートが有効になっていて、通常の UEFI セキュアブートを損なう可能性のあるルートキット攻撃からの保護が実現されます。

    b. **設定前スクリプト：**これは、ZTP サーバからの YANG 応答に含まれるオンボーディング情報で提供される base64 エンコードスクリプト（Bash、Python など）です。このスクリプトは、通常、設定を適用する前にシステムでハウスキーピングタスク（設定で使用される、必要な IOS XR パッケージのインストールなど）を実行するために使用されます。

    c. **設定：**これは、標準 XR CLI 設定または XML の有効な XR YANG モデル設定であり、オンボーディング情報で指定されたアクションに基づいてマージまたは置換される場合があります。

    d. **設定後スクリプト：**設定後スクリプトは設定前スクリプトに似ていますが、SSH キーの作成、Day 1 モニタリングおよびプロビジョニング ソリューションへの接続と登録、オンボックス アプリケーションの起動といった、ルータの最終セットアップを処理するために使用されます。

IOS XR7 のセキュア ZTP 機能の内容を見ると、IOS XR7 デバイスが、Web スケールのサービスプロバイダーや大規模サービスプロバイダー向けのセキュアで信頼できる Day 0 ロールアウトを大規模に実現するために最適であることは明らかです。特に、アクセスネットワークでは、トポロジに関係なく、運用コストと、NCS540、Cisco 8000 などの IOS XR7 ルータを展開する時間を大幅に節約できるため、これらのワークフローから大きな恩恵を受けることができます。

強力な新しい IOS XR インストール

最新のネットワーク オペレーティング システムには、ソフトウェアを管理するための最新のソリューションが必要です。

IOS XR 7 のパッケージ化とインストール

設計原則：

IOS XR7 では、IOS XR インストールの設計にまったく新しいアプローチが採用されています。その背後にある基本原則は次のとおりです。

1. **Linux の基準に準拠：**「標準および Linux に似たルックアンドフィール」のソフトウェアインストール/アップグレードというユーザエクスペリエンスを提供し、可能なかぎり Linux エコシステムの標準ソリューションを使用してキャリアクラスの要件を満たすために必要な追加機能を統合するとともに、必要に応じて XR ユースケース用に機能を拡張します。
2. **モジュラアーキテクチャの実装：**固定シングルノードシステムから、モジュラシステム、さらには大規模クラスタにいたるまで、さまざまなフォームファクタおよびユースケースに適応できるアーキテクチャを実装します。
3. **自動化可能なインターフェイスの提供：**自動化で使用する準備ができているプログラマティック ノースバウンド インターフェイスを提供します。

さらに、IOS XR7 インストールでは、システムへのすべてのパッケージ（XR RPM または Linux RPM）のインストールが処理されます。これによりユーザは、XR パッケージとカスタムアプリケーションのライフサイクルを統一された方法で管理できます。

アーキテクチャの概要：

IOS XR7 を実行する新しいプラットフォーム（Cisco 8000、NCS540）では、各ノード（ルートプロセッサ（RP）、ラインカード（LC））で動作するソフトウェアが、RPM ベースのディストリビューションである WindRiver Linux 9（WRL9）ディストリビューションを使用します。

このため、IOS XR7 のインストールでは、**「DNF」と呼ばれる Fedora コミュニティに由来する RPM パッケージ管理の最新アーキテクチャが使用されます。**DNF（Dandified yum）**は、次世代バージョンの** yum です。これは、yum との CLI 互換性を大まかに維持していて、拡張機能とプラグイン用の厳密な API を定義します。

パッケージマネージャとしての DNF は単一の Linux ノードでパッケージを処理できるため、IOS XR7 は、DNF の機能を拡張して、モジュラシステム（複数の RP および LC で構成される）のインストールおよびパッケージ管理要求を処理します。

IOS XR7 のパッケージと SMU

IOS XR7 インストールのユーザインターフェイスについて説明する前に、インストールプロセスでサポートされるパッケージのタイプについて説明します。

1. **IOS XR7 パッケージ RPM：**IOS XR7 のすべてのコンポーネントは基本的に RPM です。ただし、これらの RPM は次の 2 つのタイプに分類できます。

a. **オプションパッケージ RPM：**多数の IOS XR パッケージ（BGP、マルチキャスト、ISIS、Telnet などの機能とプロトコル）が、メタデータ フィールドに明確な依存関係ツリーを持つ RPM として公開されます。IOS XR7 のソフトウェア配信について説明した以前の項で説明したように、IOS XR のパッケージのリストはオプションパッケージになりました（つまり、**システムから削除可能**）。これによりオペレータは、設定で使用しない場合にこれらのコンポーネントを削除できます。IOS XR7 の今後の進化において、さらに多くのコンポーネントが個別の RPM に分離され、オペレータが必要とする量のソフトウェアだけで（不要なものなしに）XR のよりスリムなバリアントを実行できるようになる予定です。**オプションパッケージ RPM は、システムをリロードすることなくインストールまたはアップグレードできます。必要なのはプロセスの再起動だけです。**

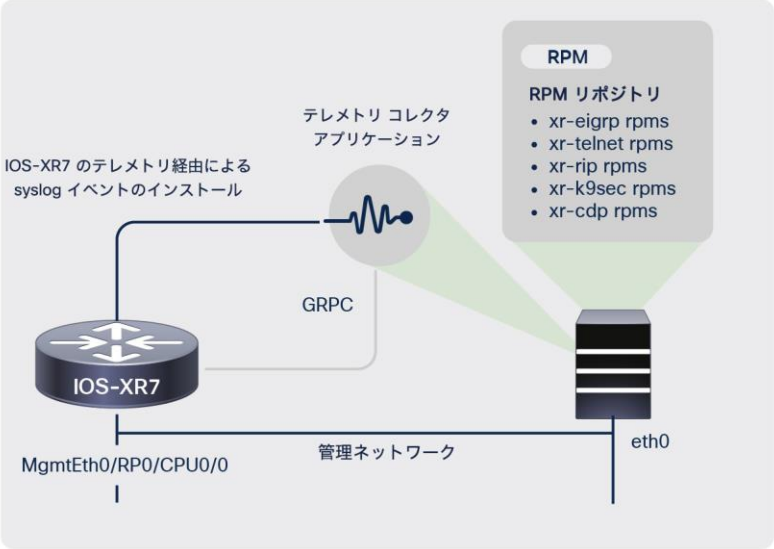
b. **必須パッケージ RPM：**これらは、RPM のままですが、多数の上位オプションコンポーネントとインフラストラクチャコンポーネントが共通のエンティティとして使用するコンポーネントです。そのため、これらの RPM はシステムから削除できませんが、アップグレードしたりパッチを適用することはできます。

必須パッケージ RPM では、アップグレード後にリロードが必要になる場合があります。デフォルトでは、これらは IOS XR7 の基本 ISO イメージの一部です。

2. **ブート可能 IOS XR7 イメージ：**インストールプロセスでは、ブート可能 IOS XR7 イメージをパッケージ（RPM）およびファイル（証明書、設定、構成）のソースとして受け入れることができます。これは、https://software.cisco.com で公開されたイメージか、カスタムゴールデン ISO（パッケージ、特殊ファイル、証明書/キーなどの配信フォーマットとしてのゴールデン ISO について説明している以前の項を参照）です。インストールプロセスでは、提供されたイメージを使用して、システムが ISO で起動されたかのようにイメージ内のソフトウェアがアクティブ化されます。

IOS XR7 インストールのユーザインターフェイスとワークフロー

IOS XR7 インストールでの作業に役立つ一般的な環境を次に示します。



Linux サーバは、このトポロジで次の 2 つの目的を果たします。

- ルータ上の基本イメージに含まれない一連のオプション RPM をホストする RPM リポジトリを提供します。サーバ上の RPM のリストが図に示されています。
- イベント駆動型 Syslog テレメトリデータをセットアップおよび受信して、ルータでのインストール操作をモニタする Python テレメトリコレクタ。

IOS XR7 による基本インストールコマンド

インストールコマンド	機能
install source [repo name| repo url] package-name	これは、システムにパッケージをインストールして適用するためのワンタッチコマンドです。新しいソフトウェアは、次回の再起動時に**のみ**起動されるように設定されます。
Install rollback [id] [noprompt] [commit]	これは、既知のスナップショット ID にロールバックするためのワンタッチコマンドです。
install commit	これはトランザクションの終了をマークするコマンドで、オペレータが更新されたソフトウェアに満足していることと、新しいソフトウェア（トランザクションの開始時に取得されたスナップショットではなく）を起動するようにブートローダーのデフォルトメニューエントリが更新されてことを示すために使用されます。

リモートリポジトリの設定

IOS XR7 では、設定の一部としてリモートリポジトリをセットアップできます。これは、RPM リポジトリからパッケージをインストールする一つの方法です。この方法論は、一般的な Linux システムのセットアップ方法と似ています。つまり、パッケージ管理コマンドを実行する前に DNF または yum リポジトリを設定します。

注：　ルータ自体にローカル DNF リポジトリをセットアップするか、ルータのローカルディレクトリにある一連の RPM （繰り返しコピーされる）を単にポイントして、必要なコンポーネントをインストールすることもできます。

IOS XR7 インストールの show コマンド

show install active：この show コマンドは、システム上で現在アクティブな IOS XR パッケージをダンプします。これらは、すでにインストールされているか、プラットフォームの起動時に基本イメージに含まれていた RPM です。

show install cached：このコマンドの出力は、RPM データベースに追加された「すべての」RPM パッケージのリストです。これには、IOS XR パッケージと Linux（WRL9）RPM が含まれます。

show install available：このコマンドは、まず、システムにインストールされている一連の IOS XR パッケージを確認し、次に、設定済みのすべてのリポジトリ（以前に設定された repo1 など）に要求を送信します。現在の RPM データベースにないパッケージがリポジトリにある場合は、それらの追加パッケージが出力にダンプされます。

ワンタッチインストール

前述のとおり、install exec コマンドの一つは「install source」です。このコマンドは、システム上の各ノードで DNF を呼び出して、依存関係管理を処理し、必要なすべてのパッケージをプルして、必要に応じてプロセスを再起動するかボックスをリロードすることにより、必要なソフトウェアをアクティブ化します。

テレメトリによるインストールの進捗状況の確認

上記のインストールプロセス全体を通して、さまざまなインストールイベントが発生すると、構造化された Syslog イベントがテレメトリコレクタに送信されます。メンテナンス時間帯または自動化中にインストール操作をモニタするために非常に役立ちます。いくつかのサンプルデータを次に示します。


```
{
   "encoding_path": "Cisco-IOS XR-infra-syslog-oper:syslog/messages/message",
   "subscription_id_str": "1",
   "collection_start_time": "1564462273170",
   "msg_timestamp": "1564462273182",
   "collection_end_time": "1564462273182",
   "node_id_str": "ios",
   "data_json": [
      {
         "keys": [
            {
               "message-id": 189
            }
         ],
         "timestamp": "1564462273178",
         "content": {
            "category": "INFRA",
            "card-type": "RP",
            "time-stamp": "1564462273000",
```

```
            "group": "INSTALL",
            "time-of-day": "Jul 30 04:51:13.164 UTC: ",
            "time-zone": "UTC",
            "text": "instorch[194]: %INFRA-INSTALL-6-ACTION_BEGIN : Compound install
operation 1.1 started \n",
            "process-name": "instorch",
            "node-name": "0/RP0/CPU0",
            "message-name": "ACTION_BEGIN",
            "severity": "message-severity-informational"
         }
      }
   ],
   "collection_id": "3"
}
            "time-zone": "UTC",
            "text": "instorch[194]: %INFRA-INSTALL-6-ACTION_COMPLETE : Apply by process
restart 1.1 complete \n",
            "process-name": "instorch",
            "node-name": "0/RP0/CPU0",
############################################################ SNIP OUTPUT
#####################################################################          "group":
"INSTALL",
            "time-of-day": "Jul 30 04:53:19.408 UTC: ",
            "time-zone": "UTC",
            "text": "instorch[194]: %INFRA-INSTALL-6-ACTION_COMPLETE : Compound install
operation 1.1 complete \n",
            "process-name": "instorch",
            "node-name": "0/RP0/CPU0",
            "message-name": "ACTION_COMPLETE",
            "severity": "message-severity-informational"
         }
      }
   ],
   "collection_id": "9"
}
```

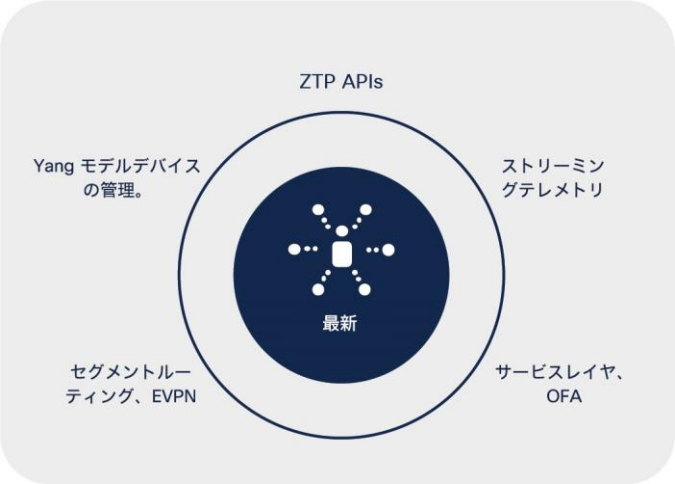

プロセス名「instorch」（インストール オーケストレータ）は、インストールプロセスによって Syslog に送信されるメッセージに関連付けられています。これは、インストール操作に関連するリアルタイムイベントを取得し、エラーをモニタするための非常に便利な手法です。

ロールバックとスナップショット：

IOS XR7 がテーブルにもたらす最も便利な機能の一つは、インストールプロセスのさまざまなコミット段階を表す以前のスナップショットにロールバックできる安全性です。

- IOS XR7 でインストールプロセスが開始されるたびに、ルートファイルシステムのバックアップ スナップショットが作成され、そのスナップショットがディスクに保存されます。さらに、ブートローダーが更新されるため、スナップショットがデフォルトでブートされます。
- インストールプロセスがいずれかの段階で失敗した場合、システムは、作成されたばかりのスナップショットに自動的に戻ります。
- 「install commit」を発行すると、ブートローダーが更新されて、新しいルートファイルシステムがデフォルトのブートエントリとしてポイントされます。古いルートファイルシステムは、ID 付きのスナップショットとして保存されます。
- 必要に応じて、「install rollback」コマンドを使用して、以前のスナップショットにロールバックできます。

IOS XR7：最新、強力、拡張可能



IOS XR では、スタック内でのプログラムおよびモデル駆動型の相互作用ポイントの構築が一貫して重視されていて、ネットワークデバイスの展開と管理を容易にする自動化可能なインターフェイスが提供されます。この設計により、外部ツール、アプリケーション、およびコントローラとの強力な統合によって IOS XR の機能を容易に拡張できます。

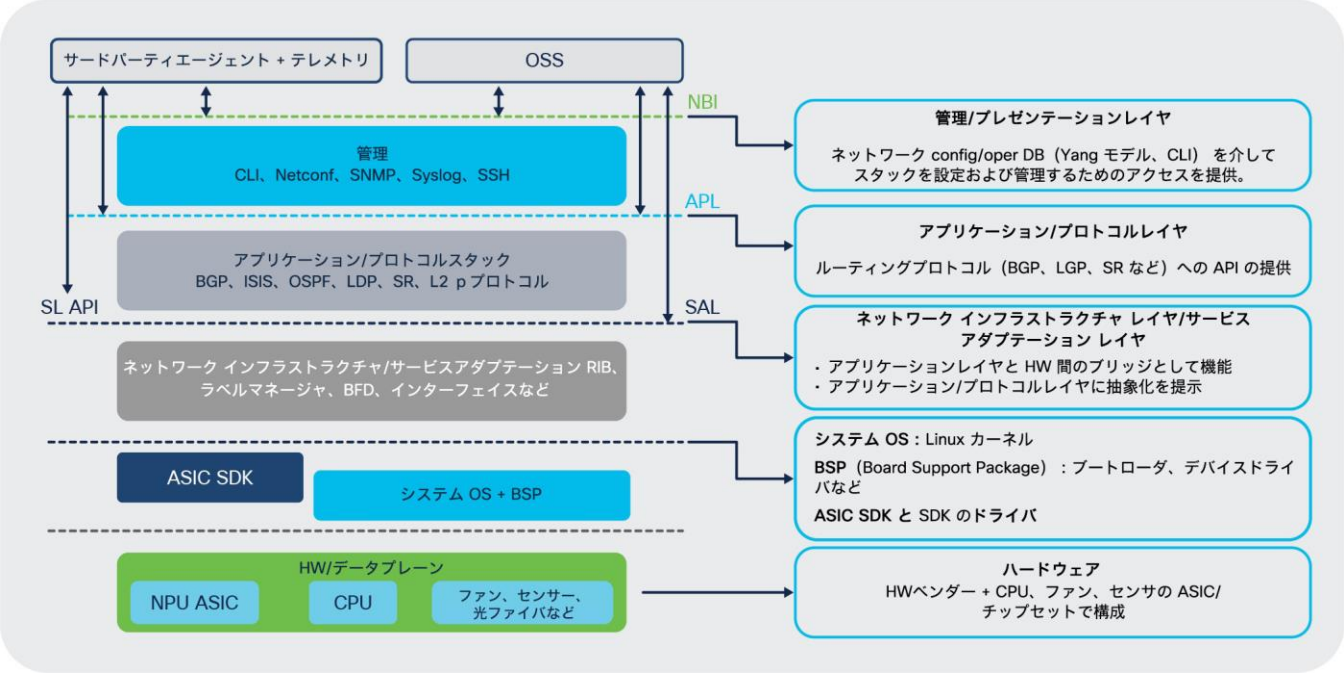
ただし、これらの機能の説明を API に関する話題だけで完結させることは困難です。セグメントルーティングや EVPN などのテクノロジーが進化したことで、ネットワークオペレータは、それらを活用して、従来のトランスポートネットワークを、プロトコル中心の展開（MPLS、RSVP-TE、レイヤ 2 およびレイヤ 3 VPN 展開）からこれらのテクノロジーに基づいた、よりシンプルでリーンなソリューション（多くの場合、外部コントローラおよびプランニングツールを併用）に移行させました。

そのため、進化するネットワーク運用の文脈でこれらの機能を理解することが重要です。通常、ネットワーク運用は次の 3 つのカテゴリに分類できます。

- **Day 0**：デバイスの最初のロールアウト、ZTP による自動展開、Day 1 管理およびモニタソリューションへの登録。ここでは、ゼロタッチプロビジョニング API およびインフラストラクチャが重要な役割を果たします。
- **Day 1**：ネットワークでのロールベースのデバイス設定、一元管理ソリューションおよびコントローラの立ち上げ、拡張可能な設定管理ツールと自動化を使用したアプリケーションとポリシーの大規模展開。デバイスプロビジョニングのためのモデル駆動型 Yang ベース API、デバイスモニタリングのためのモデル駆動型テレメトリ機能、サービスとポリシーをネットワークに展開するためのセグメントルーティングおよび EVPN ソリューションは、Day 1 運用で利用される重要な要素です。
- **Day N**：リアルタイムのネットワークおよびアプリケーション情報を使用してネットワーク内の状態を操作し、ネットワークの劣化の問題を修正し、アラームを発生させてメンテナンス時間をスケジュールし、ネットワークのテレメトリデータおよびパターンを分析してネットワーク設計を進化させることで将来の課題に対応します。サービスレイヤおよびオープン フォワーディング アブストラクション（OFA）API などのハイパフォーマンスの下位レイヤ API、テレメトリを介して実装されるフィードバックループ、および YANG に基づくデバイスプロビジョニング API を活用してネットワークやアプリケーションのイベントに応答するカスタムコントローラ/アプリケーションは、Day N ソリューションの革新を促進する一般的なコンポーネントです。

上記のような API およびテクノロジーとさまざまな運用パラダイムとの対応をふまえて、これらの機能の所在を理解するために、IOS XR7 のネットワークスタックについて詳しく説明します。

IOS XR7 スタックのさまざまなレイヤと、識別される対応機能を次に示します。



- **管理性レイヤ：**このレイヤには、ネットワーク アプリケーションレイヤの機能やプロトコルに対する**コマンドライン インターフェイス（CLI）および YANG モデル API** があります。これらの YANG モデルは、デバイスプロビジョニングおよびストリーミングテレメトリに使用されます。**ZTP API は、このレイヤの CLI および YANG API の上に構築されます。**このレイヤでは、「SysDB」と呼ばれる IOS XR 内の内部「データベース」が活用されます。
- **ネットワーク アプリケーション/プロトコルレイヤ：**このレイヤには、BGP、ISIS、OSPF などのプロトコルと L2VPN、L3VPN などの機能が含まれます。このレイヤは SYSDB とやり取りして、プロトコルや機能の動作および設定状態を保存します。**セグメントルーティングと EVPN の上位レイヤコンポーネントは、このレイヤに分類されます。**
- **ネットワーク インフラストラクチャ/サービス適応レイヤ：**通常、RIB、ラベルスイッチデータベース、BFD、ネットワーク インターフェイス ハンドラ、コントローラ/エージェント用 API などのコンポーネントで構成されます。**このレイヤの上でエンドユーザ向けに提供される API は「サービスレイヤ API」と呼ばれます。すべての XR コントロール プレーン プロトコル（BGP、ISIS、OSPF など）、セグメントルーティング、および EVPN のインフラストラクチャ コンポーネントは、このレイヤにマッピングされます。**
- **ハードウェア アブストラクション レイヤ（OFA）：**このレイヤは、ASIC SDK（ASIC ベンダーが提供）とやり取りし、RIB の状態や LSD（ラベルスイッチデータベース）の状態に基づいてデータプレーンのプログラミングを処理します。**オープン フォワーディング アブストラクション（OFA）API は、外部アプリケーション向けにこのレイヤの機能を抽象化します。**
- **プラットフォーム統合レイヤ：**通常、ファン、LED、センサーなどのデバイス用のカーネルやユーザランドドライバで構成されます。これらのデバイス固有ドライバとソフトウェアスタックの上位レイヤの統合は、このレイヤで行われます。ソフトウェアスタックがセンサーから情報（温度など）を抽出し、コンポーネントの状態（ファン速度など）に影響を与えることを可能にしているのは、このレイヤです。

管理性レイヤ：YANG モデル、デバイスプロビジョニング、テレメトリ

管理性レイヤには「SysDB」と呼ばれる XR の内部設定および運用状態データベースが格納されます。これは、Netconf や gRPC などのプロトコルを介した外部ツールによる自動化のために用意されている YANG モデルの基盤です。さらに IOS XR は、デバイスのプロビジョニングと管理のために、ネイティブ（XR 固有）YANG モデルと OpenConfig YANG モデルをサポートしています。これらの YANG モデルは、ここで GitHub 上の XR のすべてのリリース向けに公開されています。

IOS XR テレメトリにより、トランスポートとして TCP、UDP、および gRPC を介し、エンコード形式として GPB または JSON を介した、デバイスからの構造化された運用状態データのケイデンスベースおよびイベントベースのストリーミングが実現されます。XR のリリースごとの IOS XR テレメトリ用の gRPC クライアントコレクタのプロトファイルは、ここにあります。

さらに、IOS XR7 は、gNOI および gNMI のサポートによって、デバイス管理のための OpenConfig 手法もサポートしています。

ゼロタッチプロビジョニング（ZTP）API

ゼロタッチプロビジョニング API は、Bash ライブラリおよび Python ライブラリとしてオンボックスで使用できます。これらの API を活用して、すべての CLI 操作（設定、show コマンド、exec コマンド）および YANG

ベースの RPC 呼び出し（config、oper、action）を実行して、システムをプロビジョニングできます。これらの API は、Day 0 運用の一部として、または Day 1 以降のオンボックススクリプトによるデバイスの自動化のために、ZTP スクリプトで活用できます。ZTP ライブラリの詳細については、こちらをご参照ください。

サービスレイヤ API

ほとんどの自動化ユースケースには、CLI、YANG モデル、およびストリーミングテレメトリ機能を提供する管理性レイヤが適しています。ただし、この数年、Web スケールのサービス プロバイダー ネットワークおよび大規模サービス プロバイダー ネットワークでは、従来のプロトコルまたは機能のステートマシンを抽出し、それらの動作をネットワーク上の特定のアプリケーションセットの要件に結び付けるオフボックスコントローラまたはオンボックスエージェントへの依存度が高まっています。

これらのエージェント/コントローラには、ネットワークスタックの最下層（RIB、ラベルスイッチデータベース（LSD）、BFD イベント、インターフェイスイベントなど）へのハイパフォーマンスアクセスが必要であり、この機能を提供する API は「サービスレイヤ API」と呼ばれます。これらの API には gRPC を介してアクセスできます。また、これらは、GPB インターフェイス記述言語（IDL）を使用して記述されるモデルによるモデル駆動型 API です。これらのモデルとサンプルクライアントは、ここの GitHub にあります。

サービスレイヤ API の詳細については、https://xrdocs.io/cisco-service-layer/ の xrdocs をご参照ください。

オープン フォワーディング アブストラクション（OFA）API

オープン フォワーディング アブストラクション（OFA）API は、基盤となるハードウェア SDK を抽象化するために構築されたモデル駆動型 API であり、ハードウェア プラットフォームに関係なく、プログラムに一貫性のあるインターフェイスを提供することにより、アプリケーションやコントローラがパフォーマンスチャネルを介してハードウェア状態を直接操作することを可能にします。

内部的には、この API は、複数のデータプレーンおよびハードウェアプラットフォームにわたる IOS XR の柔軟性と迅速な有効化を実現します。これにより、IOS XR ポートフォリオがさらに拡張されます。

OFA API を抽象化として利用して、P4 非互換ハードウェア上で P4 ランタイムなどの機能を提供することもできます。これにより、クライアント アプリケーションおよびコントローラが特定の抽象化レイヤ向けに設計された固有のユースケースを実現できます。

OFA API はモデル駆動型 API として設計されているため、トランスポートオプションとして gRPC を介して利用できます。この API は、一般利用向けにリリースされる予定です。

IOS XR7：高信頼性

IOS XR7 では、信頼性の概念が、デバイスまたは NOS セキュリティに関するあらゆる議論のテーマとなっています。IOS XR7 とその互換ハードウェアの目標は、デバイスと OS の整合性がセキュリティの概念と同様に重要視された「高信頼性プラットフォーム」を構築することです。実際には、そもそもセキュアであると見なせるデバイスは存在しないと認識することが重要です。ただし、それは理想的な状態であり、必ずしも達成できるとはかぎりません。信頼性のレベルは、測定、検証、および監査できます。

ネットワークデバイスのセキュリティに関連する主要な懸念事項を明確にすることが重要です。

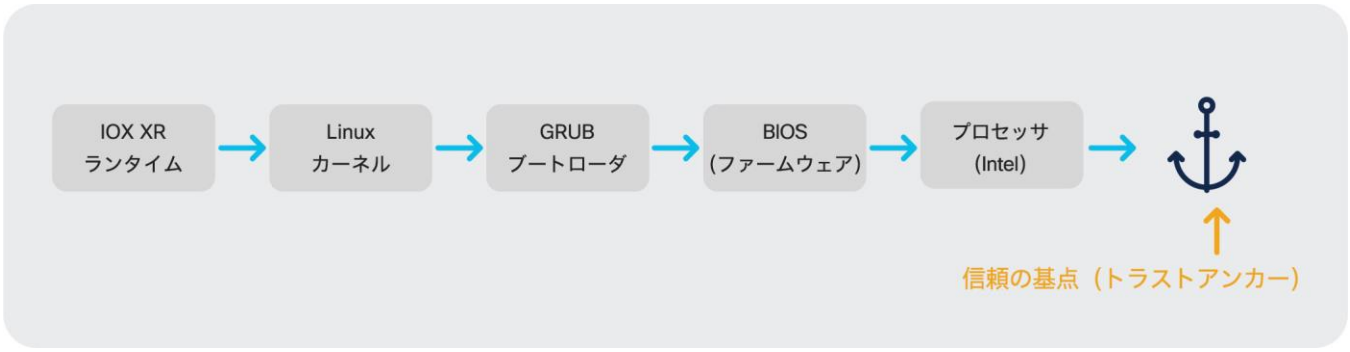
- オペレータは、ルータが意図されたソフトウェアのみを実行していることを、どのようにして確認できますか。
- シスコが構築したルータがその後に物理的に変更されていないことを、どのようにして確認すればよいでしょうか。

署名されたソフトウェアのみのインストール/実行を許可する、ブート中にハードウェアコンポーネントの既知の適正な値を照合して検証する、といったセキュリティ制御をただちに構築することで、これらの懸念事項に対処できます。さらに一歩踏み込んで、証明書およびキーと検証に使用する既知の適正な値を保存する、改ざん防止機能を備えたハードウェアによってセキュリティ制御が支援されるようにすることで、完全な検証の実現が容易になります。構成証明機能を使用すると、検証結果を可視化してレポートできます。これにより、コンポーネントとネットワークが信頼できることを証明できます。

これらの懸念事項に対処するとともに、IOS XR7 およびその関連プラットフォーム内の信頼性パラダイムを理解するには、一般的なネットワークデバイスの信頼性のチェーンを理解する必要があります。

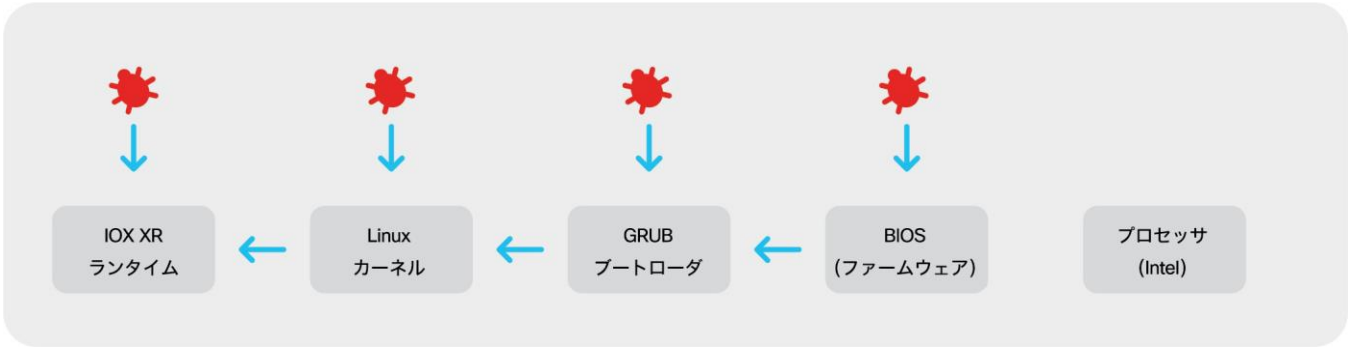
ハードウェアの信頼ルート

この概念はシンプルです。ブートプロセスをトレースバックする場合、信頼ルートを可能なかぎり後方にプッシュすることで、システムにおいて最高レベルの信頼性が確保されます。

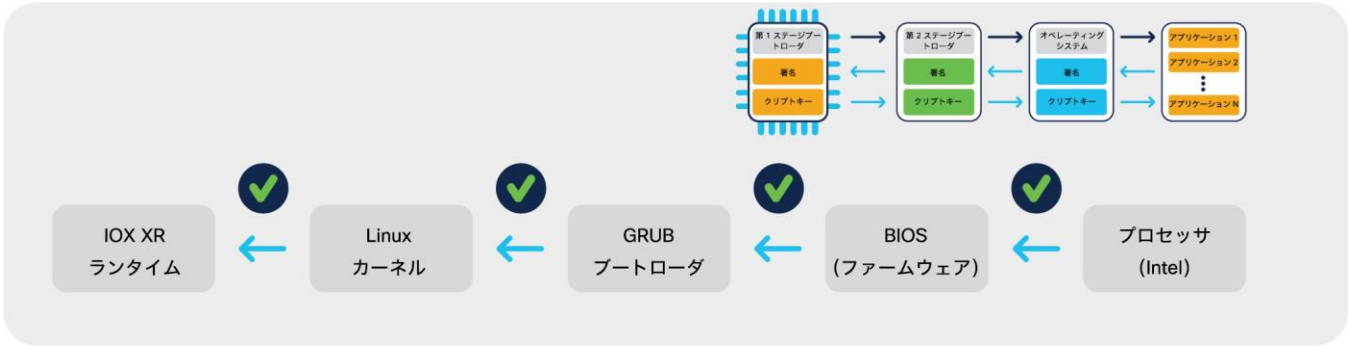


一般に、ランタイム OS、カーネル、およびブートローダー（GRUB）が侵害される既知の攻撃や脆弱性が存在します。

UEFI 仕様 2.3.1 では概念としてのセキュアブートが規定されています。この仕様では、BIOS 段階で「信頼性」が開始され、その後に、ブートローダー、カーネル、OS がチェックされることが想定されています。しかし、この数年は、BIOS を侵害する可能性のある UEFI ルートキットの出現が確認されていて、セキュアブートプロセスが危険にさらされています。さらに、いずれも Intel ハードウェア上の BIOS 拡張機能である Intel ME および AMT に関するセキュリティ問題の範囲が大幅に拡大しつつあります。UEFI ファームウェアエコシステムの問題や、インプラント/ルートキットの配布およびインストールが比較的簡単であることについては、すでに多くの研究が発表されています。

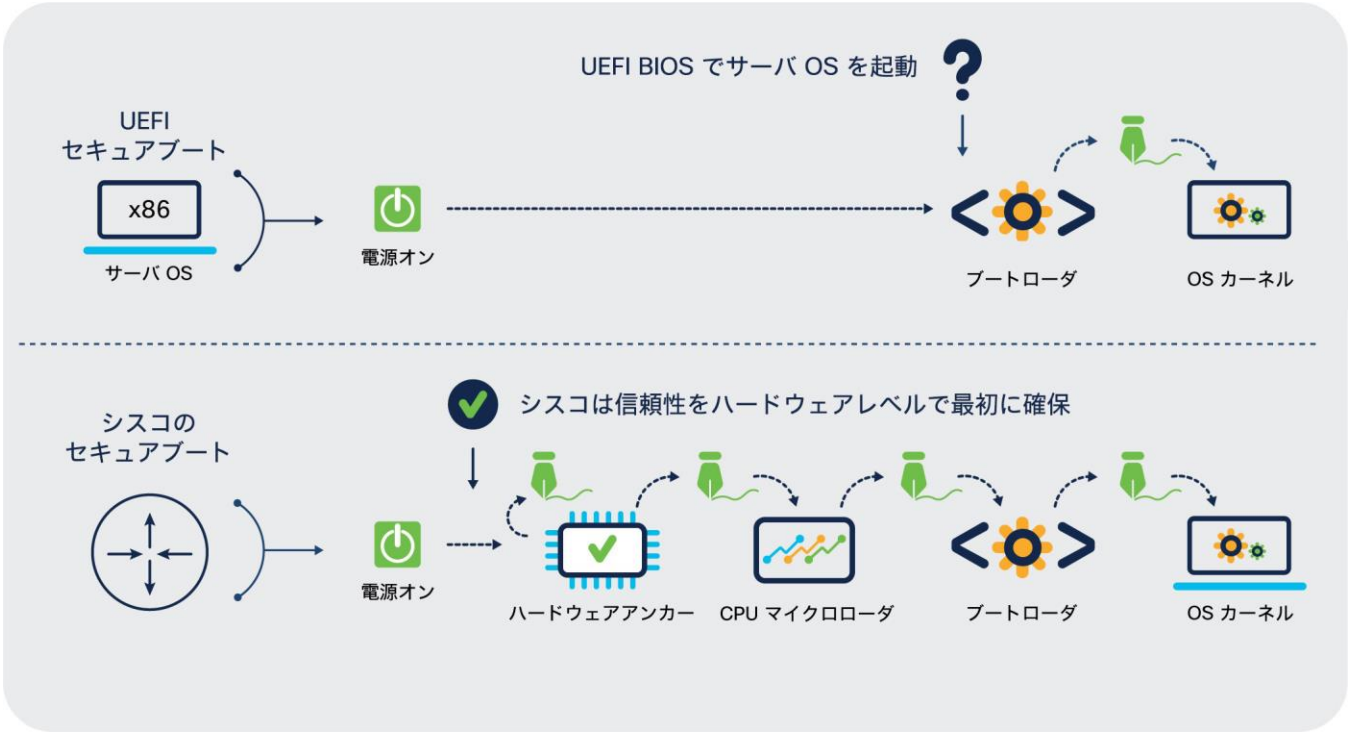


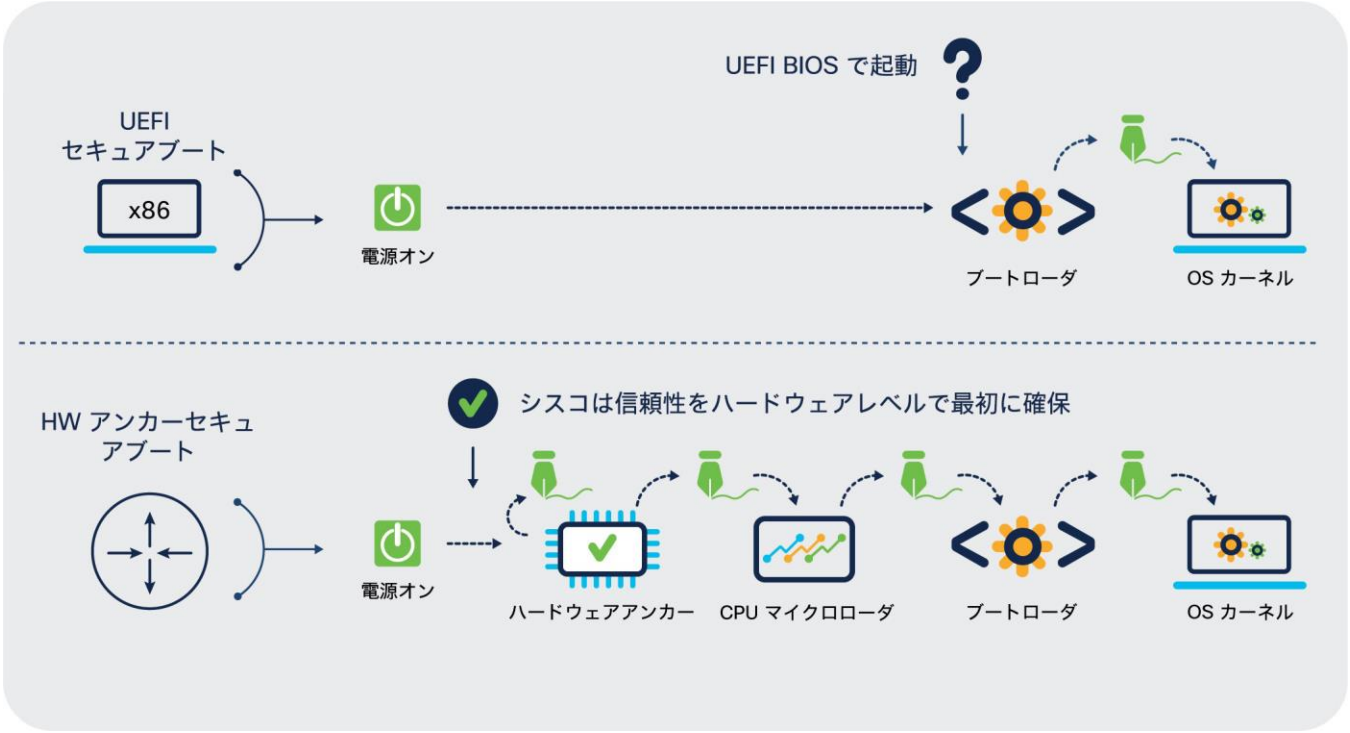
つまり、ブートプロセスを適切に保護するには、BIOS がブートする前であっても、ブートサイクルの非常に早い段階で検証チェックを開始する必要があります。IOS XR7 プラットフォームでは、まさにそれが行われます。シスコのハードウェア プラットフォームでは、「ハードウェア アンカー セキュア ブート」と呼ばれるものが実行されます。



シスコのセキュアブートと UEFI セキュアブート

シスコのセキュアブートは、セキュアブートプロセスをハードウェアに固定することで差別化され、これにより非常に堅牢なセキュリティを提供します。ハッカーがデバイスを物理的に所有している場合でも、ハードウェアの変更は難しく、コストがかかり、隠蔽も容易ではないため、堅牢です。





このプロセスにより、マイクロローダーからブートローダーを介してオペレーティングシステムへの「信頼性のチェーン」が作成され、ソフトウェアの信頼性が確立されます。デジタル署名のチェックに成功しないと、シスコのデバイスはそのソフトウェアを起動させないため、悪意のあるコードは実行されません。

シスコのトラストアンカーモジュール（TAm）と業界標準の TPM

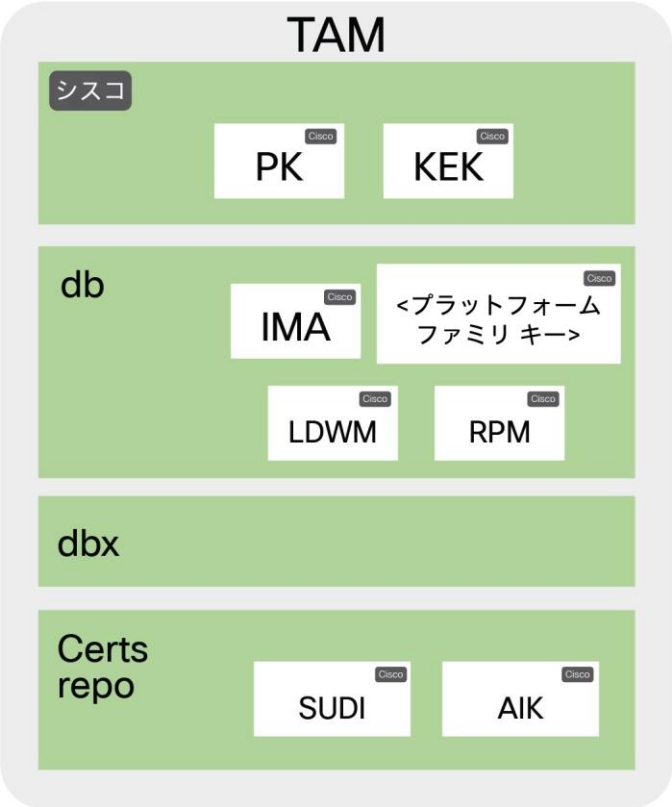
TAm を備えたシスコのデバイスで、シスコのハードウェア アンカー セキュア ブートにより、ソフトウェアがシスコの正規ソフトウェアとして認証されると、オペレーティングシステムは、TAm に照会してハードウェアが本物であることを確認します。これは、シスコからのみ提供されるセキュア固有デバイス識別子（SUDI）に関して TAm を暗号的にチェックすることによって行われます。

SUDI は、TAm に恒久的にプログラムされていて、閉鎖され、セキュリティ保護され、監査されたシスコの製造プロセスにおいてシスコによって記録されます。このプログラミングは強力なサプライ チェーン セキュリティを提供します。これは、ルータやスイッチなどの組み込みシステムにとって特に重要です。

これに対し、業界標準の TPM は、TAm と同様の機能を備えていますが、一般に、製造中に固有デバイス識別子によって恒久的にプログラムされることはなく、エンドユーザ任せになっています。このプロセスは、ユーザによる介入と開発が必要であり、柔軟性はありますが、サプライチェーンが変更されるリスクも大きくなっています。

TAm の証明書とキー

次の図は TAm の基本的なセットアップを示しています。TAm のキーと証明書の機能を理解し、ブートプロセスの各コンポーネント（BIOS から、ブートローダー、カーネル、OS、RPM、アプリケーションプロセスまで）がどのようにして信頼性を確保するのかを確認することは有益です。



これらのキー（PK、KEK、DB/DBx）は、UEFI セキュアブート仕様で定義されています。

シスコの TAm も同じ仕様に従っています。

特に、次のキーは、シスコの TAm 実装に固有のものです。

- **<platform-family> キー：**これは、単一のファミリー（たとえば Cisco 8000）に属するすべてのタイプのプラットフォームに共通の公開キー証明書です。**このキーは、ブートローダー（GRUB）、initrd、カーネル、およびカーネルモジュールに署名するために使用されます。**
- **RPM キー：**このキーは、RPM に署名するために使用されます。
- **IMA 公開キー証明書：**実行時にプロセスを起動するバイナリのハッシュに署名するために使用されます。
- **LDWM キー：**このキーは、システム ブート ファームウェア（BIOS）およびフィールド プログラマブル ゲート アレイ（FPGA）イメージに署名するために使用されます。LDWM ドラフトに従い、これらのキーのフットプリントは RSA キーより小さくなっています。
- **SUDI（セキュア固有デバイス識別子）：**SUDI は、非対称キーペアおよび関連する証明書を指します。この証明書には、デバイス UDI（PID + シリアル番号）が含まれています。

- **構成証明キー（AIK）**：RSA キーペアであり、TAm 自体に保存されているデータに署名するために使用されます。TAm の外部のデータに署名するためには使用できないため、「TAm 見積もり」（整合性測定の信頼性に関する整合性の高いレポート）のなりすましは非常に困難です。さらに、AIK は、PCR 状態（ブートシーケンスの各検証プロセスの後に TAm で生成される状態）を構成証明するためにも使用されます。TAm 見積もりと PCR 見積もりの取得は「構成証明」と呼ばれます。

注： UEFI 仕様では強制されていませんが、シスコの TAm 設計では、TAm のすべてのキーは、シスコのパブリックルート証明書にトラックバックできる信頼チェーンの一部であることが強制されています。

TAm のすべてのキーは、製造プロセスにおいてシスコによってインストールされます。

セキュアブート

前述のように、シスコのプラットフォームはハードウェア アンカー セキュア ブートを実行します。ブートプロセスとそれを超えるさまざまな段階を検証するために使用される一連のキーをホストする「TAm」と呼ばれる改ざん防止機能を備えたモジュールを使用することで、ハードウェアの信頼ルートが有効になります。基本的なセキュア ブート フローを以下に示します。

BIOS の起動と検証：

1. TAm に含まれるシスコの公開キー（PK、KEK、DB）は、初期ブートプロセス中に署名を検証するために使用されます。電源投入時に、TAm のマイクロローダーは、まず、TAm の LDWM キーを使用して BIOS のデジタル署名を検証します。
2. 検証が完了すると、BIOS は、TAm のデータベースにあるハードウェアの既知の適正な値（KGV）に対してハードウェアの検証を実行します。これらの既知の適正な値は、製造者によってプログラムされています。障害がある場合は、障害がログに記録されます。

ブートローダーの起動と検証：

3. 次に、BIOS は TAm DB の <platform-family> キーを使用してブートローダーのデジタル署名を検証します。

カーネル、initrd、grub-config の検証：

4. ブートローダーが検証されると、ブートローダーが BIOS によって起動されます。その後、ブートローダーは、BIOS の助けを借りて、カーネル、initrd、および grub-config を検証します。
5. 各検証操作がログに記録されます。その後、initrd が展開され、ルートファイルシステムが作成されます。

カーネルモジュールの検証：

6. カーネルが起動し、必要なキー（PK、KEK、IMA、RPM）がカーネルキーリングにロードされます。
7. その後、カーネルがカーネルモジュールを検証し、結果がログに記録されます。

OS プロセスブート：

8. 実行時に署名を検証するために使用される IMA は、init プロセスを検証するために適切な IMA ポリシーで起動されます。

XR RPM のインストール

9. ここで、IOS XR インストールプロセスが実行され、イメージに含まれる IOS XR RPM がインストールされます。

10. IOS XR インストールプロセスでは、TAm からカーネルキーリングにロードされた RPM キーを使用して、インストール前にすべての RPM の署名が検証されます。RPM 検証の結果はログに記録されます。

XR プロセスの起動

11. 最後に、XR プロセスが起動されます。各プロセスは、起動前にハッシュの署名を検証する IMA ポリシーチェックの対象となります。

署名済み RPM

IOS XR7 の各 XR RPM は、TAm の RPM キーを使用して署名されます。これにより、IOS XR7 インストール（既存および新しいプラットフォーム）を使用して、署名済み RPM のみをシステムにインストールできる状態が確保されます。これは、IOS XR7 の基礎となる Linux ディストリビューション用にコンパイルされたサードパーティアプリケーションをインストールするには署名する必要があることも意味します。この目的のために、オペレータは、カスタム証明書をシステムにロードし、カスタム証明書を使用してアプリケーションに署名できます。これにより、これらのアプリケーションの管理ライフサイクルが非常に柔軟なものになります。

実行時の信頼性（IMA）

整合性測定アーキテクチャ（IMA）は、ロードの準備をしている実行ファイルが元の形式から変更されていないことを保証するためのプロセスです。システムが改ざんされたコードにさらされていない状態を確保できることは重要です。

IOS XR のすべてのバイナリおよびプロセスは、デフォルトで IMA 用にセットアップされますが、ネットワークオペレータは、サードパーティプロセスを実行する前に、サードパーティ アプリケーション バイナリに署名し、署名済みハッシュをシステムにロードする必要があります。

IMA プロセスには次の 2 つの部分があります。

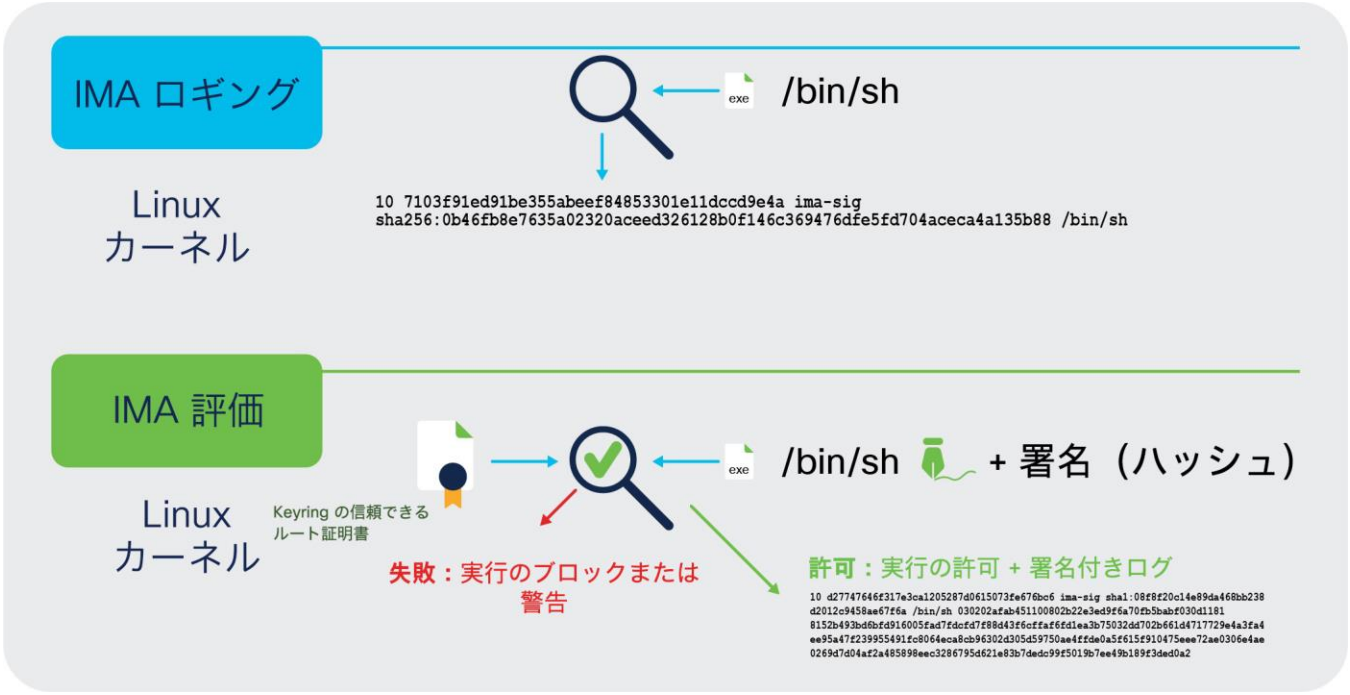
- **IMA 測定/ロギング**：IMA は、実行時測定リストを保持します。シスコのプラットフォームでは、それがハードウェア信頼プラットフォームモジュール（TAm）に固定されています。IMA は、このリスト全体の集約整合性値も保持し、それを TAm にプッシュします。集約整合性値を TAm に保存する利点は、測定リストがソフトウェア攻撃によって侵害されず、検出されないことです。

  そのため、信頼できるブートシステムでは、IMA を使用してシステムの実行時整合性を構成証明できます。

- **IMA 評価**：IMA 評価拡張機能により、ローカル整合性検証と、拡張属性として保存された「適正」値に対する IMA 測定の実施が追加されます。検証プロセスでは、ファイルデータの整合性をチェックされ、デジタル署名も検証されるため、信頼性が提供されます。

  IMA の評価では、通常、評価が失敗するとアプリケーションがブロックされます。これが強制モードです。

注： 初期リリースの IOS XR7 では、アプリケーションはブロックされず、検証後に失敗が単にログに記録されます。これは、オペレータが、カスタムプロセスおよびアプリケーションをパッケージ化し、それを IOS XR システムで実行することに慣れてから、強制モードに進むことを想定しているためです。

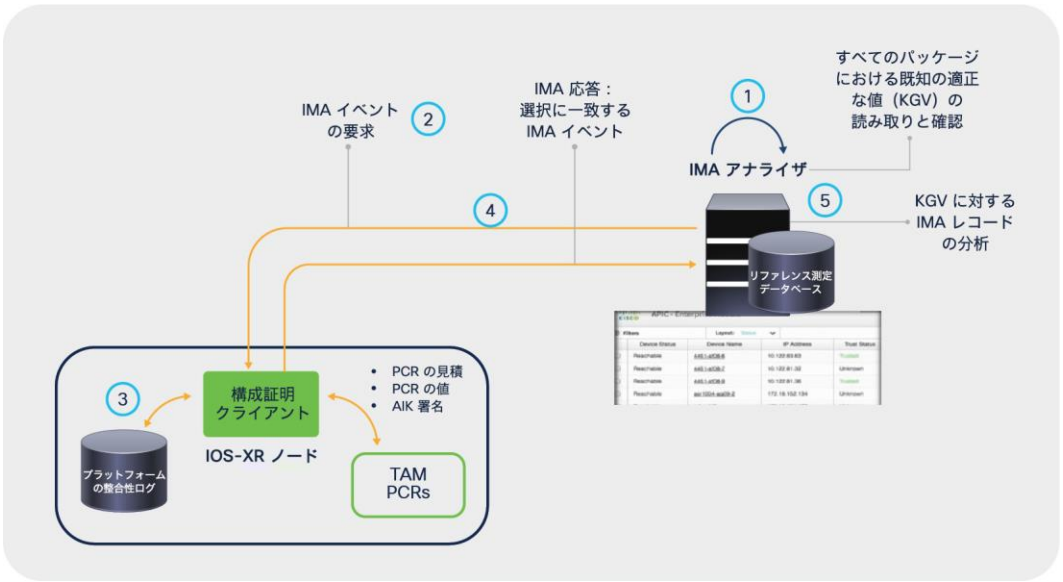


信頼性レポートと可視化

システムは、ハードウェア整合性、ブートアップ、ソフトウェアの実行、ソフトウェアのインストール/アップグレード、IPC 整合性といった、そのライフサイクルのさまざまな時点で多くの暗号化測定を行います。デバイスの各ノードで行われる測定の可視性は、 システムの信頼性を確保するために不可欠です。

リモート構成認証は、デバイス内の各ノードで行われた測定値について TAm に照会し、照会後に可視化するプロセスです。

ネットワークデバイスの整合性構成証明プロセスの大まかなフローを次に示します。



シスコの環境維持への取り組み

シスコの企業の社会的責任（CSR）レポートの「環境の持続性」セクションでは、製品、ソリューション、運用・拡張運用、サプライチェーンに対する、シスコの環境持続性ポリシーとイニシアチブを掲載しています。

次の表に、環境の持続可能性に関する主要なトピック（CSR レポートの「環境の持続性」セクションに記載）への参照リンクを示します。

持続可能性に関するトピック	参考資料
製品の材料に関する法律および規制に関する情報	材料
製品、バッテリ、パッケージを含む電子廃棄物法規制に関する情報	WEEE 適合性

シスコでは、パッケージデータを情報共有目的でのみ提供しています。これらの情報は最新の法規制を反映していない可能性があります。シスコは、情報が完全、正確、または最新のものであることを表明、保証、または確約しません。これらの情報は予告なしに変更されることがあります。

Cisco Capital

目標の達成を支援する柔軟な支払い方法

Cisco Capital は、お客様が目標の達成、ビジネス変革の実現、競争力の維持に合ったテクノロジーを導入できるよう支援します。総所有コスト（TCO）の削減、資金の節約、成長促進を支援します。100 カ国以上で利用できる Cisco Capital の柔軟なソリューションにより、ハードウェア、ソフトウェア、サービス、補完的なサードパーティ製機器を、予測可能な方法で容易に支払うことができます。詳細はこちら

お問い合せ先

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

C78-743014-00JA 20.04