Data sheet
Cisco public

The bridge to possible

IOS XR Data Sheet

Contents

New demands require new software 3

IOS XR’s lockstep evolution 4

Simple 5

Modern 6

Trustworthy 6

A simplified architecture for the future 7

Support for Linux workflows 8

Golden ISO (GISO) 9

Zero-Touch Provisioning (ZTP) 10

Hardware root of trust 15

Cisco secure boot versus UEFI secure boot 15

Cisco Trust Anchor Module (TAm) versus industry-standard TPM 16

Cisco environmental sustainability 16

Cisco Capital 17

Document history 18

New demands require new software

Traditional Network Operating Systems (NOSs) are designed to run in a native hardware environment. They have been designed so one size fits all with a rigidity that complicates network device management and requires more time for updates and changes. At Cisco, we believe that NOS platforms need to be simple, modern, and trustworthy. A NOS should be able to support unique configurations, improve operational flexibility, and enhance security. Without a flexible NOS, service provider engineers struggle to efficiently manage and operate a fast, reliable, and flexible network that can cope with unprecedented traffic growth and provide the services and performance that service provider customers demand.

As more devices are connected and more content is consumed, Internet traffic has seen a compounded annual growth rate of 30 percent over the last five years. It's anticipated that by 2022:

- IP traffic will increase to 396 exabytes per month.
- 1/3 of the Internet traffic will be in metropolitan service areas.
- More than 28 billion devices and connections will be online.
- Peak busy hour Internet traffic is growing faster than average Internet traffic because of increases in streaming video and online gaming.[1]

In anticipation of this growth, Cisco designed the IOS® XR NOS to ease network operations. As a modern operating system, IOS XR is designed to help engineers by:

- Providing a single, easy-to-maintain NOS paradigm across the network: edge, aggregation, and core.
- Reducing Operating Expenses (OpEx) with simplified delivery and deployment based on the features you need.
- Using Linux-style workflows and support for standard Linux libraries to simplify provisioning and management of the device.
- Improving operational efficiencies with management API integration to provide near-real-time, actionable telemetry data.
- Allowing for automation to drive smoother implementations and remote configuration updates.
- Validating trust within the network so service providers can work to operate a secure environment.

[1] Cisco Visual Networking Index, 2017-2022

IOS XR's lockstep evolution

The IOS XR Network Operating System has consistently evolved to meet these technological transitions. IOS XR brings with it the best-in-class routing protocols and features, and the continued focus on intent-based transport technologies, such as segment routing and Ethernet Virtual Private Networking (EVPN), has made IOS XR the leading choice for web-scale and large-scale Service Providers (SPs) across network segments.

It is important to recognize the transition toward automated operational workflows in SP networks that are driving vendor network operating systems across the industry to implement open, extensible software architectures. This transition improves operational scalability with model-driven APIs at every layer of the network stack. Development of IOS XR was deliberately focused on complementing this transition and resulted in a number of significant changes:

- In Release 6.x, IOS XR moved from a 32-bit QNX operating system to a 64-bit Linux operating system.
- Embedded support for YANG models with programmatic provisioning and high-scale, real-time telemetry data.
- Highly performance service-layer APIs at lower layers of the stack for custom protocols and controllers, greatly elevated the capabilities of IOS XR to integrate with desired workflows and tools.
- Introduction of Zero-Touch Provisioning (ZTP) capabilities in IOS XR to address Day 0 deployment requirements.
- IOS XR opened up its underlying Linux environment for direct access to end users, allowing for integration of scripting languages (bash, Python, Golang, C++), configuration management tools (Ansible, Puppet, Chef, etc.), and support for containers (LXC and Docker).

With the announcement of the Cisco®8000 series platforms enabling highly scalable, 400G-optimized routers, IOS XR continues to expand its portfolio of supported platforms to include VNF solutions using XRd and XRv9000 virtual routers, NCS 540/560, NCS 5000/5500/5700, ASR 9000, and qualified third-party hardware. IOS XR offers the most comprehensive portfolio of platforms and solutions in the industry which enables consistent operations and feature capability across all segments of the service provider network.

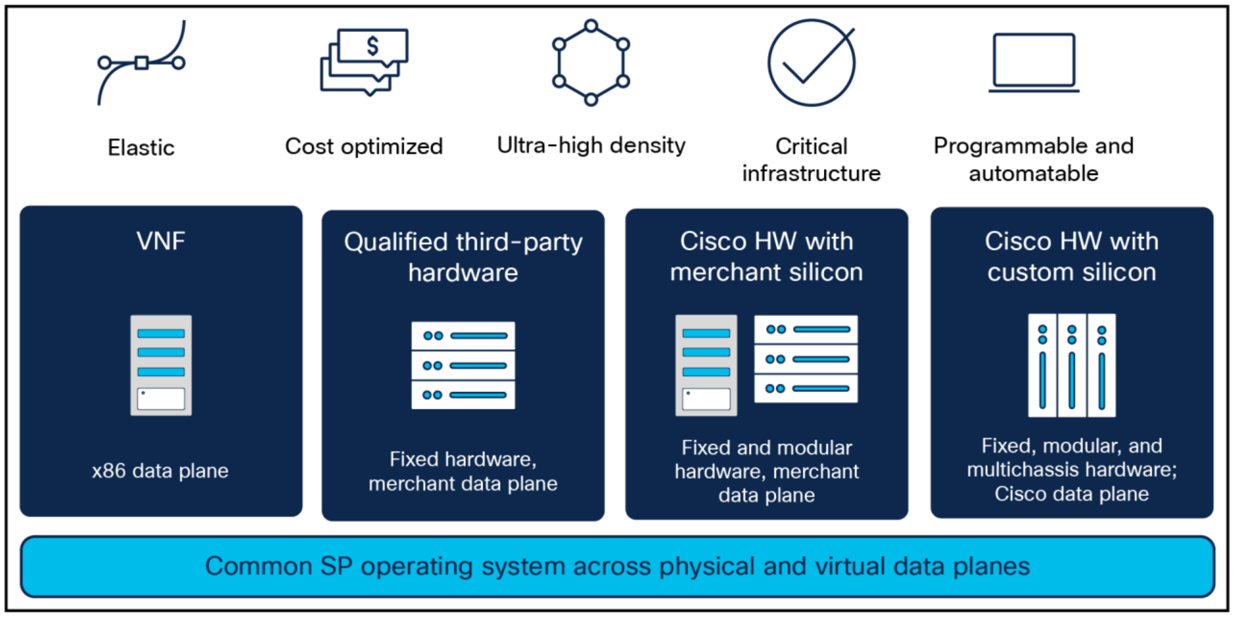
Elastic
Cost optimized
Ultra-high density
Critical infrastructure
Programmable and automatable
VNF
x86 data plane
Qualified third-party hardware
Fixed hardware, merchant data plane
Cisco HW with merchant silicon
Fixed and modular hardware, merchant data plane
Cisco HW with custom silicon
Fixed, modular, and multichassis hardware; Cisco data plane
Common SP operating system across physical and virtual data planes

Figure 1.
IOS XR portfolio of supported platforms

Introducing IOS XR: Simple, Modern, Trustworthy

With Release 7 of IOS XR, the architecture evolves to address these requirements head on. The IOS XR architecture focuses on three primary tenets:

Simple

Easier to provision, automate, and integrate with a wide variety of workflows.

1. **Simpler and leaner architecture:** For newer hardware platforms on IOS XR, there is no admin plane and no system containers, making it easier than ever before to manage the system through off-the-shelf tools and scripting solutions.

2. **Simpler operations:** With Linux-style workflows and integrations, enabling the use of scalable configuration-management tools (Ansible, Puppet, Chef), and support for standard Linux applications on box as binaries or containers (LXC, Docker).

3. **Simpler, secure Day 0 rollouts:** With powerful and secure zero-touch capabilities based on RFC 8572 to help enable secure device onboarding through template-driven ZTP scripts based on YANG-modeled transactions between IOS XR routers and bootstrap servers.

4. **Simpler software delivery and deployment:** Through artifacts called Golden ISOs (GISOs) to combine custom scripts, applications, packages, and files into a deployable ISO artifact. Individual IOS XR components and patches are delivered through RPMs.

5. **Powerful new design of IOS XR Install:** Built on top of DNF (Dandified YUM) and extended to support modular chassis systems and YANG-modeled APIs, IOS XR install allows operators to manage the lifecycle of IOS XR RPMs, native Linux RPMs, and Golden ISO (GISO) installations while supporting real-time telemetry notifications of the install process.

Modern

Model-driven APIs at every layer of the stack. Industry-leading support for transport technologies.

1. **YANG-modeled management layer APIs:** To automate device provisioning and management. These models include native IOS XR YANG models and OpenConfig models.

2. **Streaming telemetry capabilities:** For cadence-based or event-driven monitoring of data derived from YANG-modeled paths in the manageability layer over gRPC, TCP, or UDP as transport.

3. **Service Layer (SL) and Open Forwarding Abstraction (OFA) APIs:** Service-layer APIs are highly performant APIs at the common network infrastructure layer of IOS XR (RIB, Label Switch Database, BFD, L2, etc.) to enable controllers and custom protocols to manipulate routes in IOS XR RIB, create label-switched paths on the fly, etc. The OFA API is a model-driven API on top of the ASIC SDK of the hardware platform to enable high-performance access to the lowest layer of the network stack either directly or through modeled abstractions such as P4 runtime.

4. **Segment routing and EVPN:** Segment routing and EVPN have been the mainstays of IOS XR-based transport deployments with a focus on simplicity, scale, and programmatic extensibility. IOS XR continues focus on these technologies with further support for SR Flex-Algo, SRV6, and more. EVPN drives the next level of simplicity by offering a unified control plane protocol (BGP) for all service types, including Layer2 VPN and Layer3 VPN services.

5. **Zero-touch APIs:** Exhaustive APIs for zero-touch provisioning to enable Day 0 automation. This includes CLI automation APIs through bash and Python libraries on box. With IOS XR, operators will also be able to use the on-box netconf client in the ZTP Python library to bring up the system through YANG-modeled XML inputs, further enabling network teams to transition their systems from CLI to YANG models when ready.

Trustworthy

Secure, trusted workflows are only possible with a trustworthy NOS running on a trustworthy network device. With IOS XR, this is achieved by ensuring:

1. **Trust begins in the hardware:** A secure, nontamperable Trust Anchor Module (TAm) houses known good values of the hardware components, along with keys and certificates rooted to Cisco. These are utilized to verify components of the hardware during BIOS boot.

2. **Secure boot:** Trust is reinforced by validating parts of the network OS (IOS XR) through a secure boot process. This is enabled by validating the signatures of the bootloader and kernel modules against the keys in TAm, before starting the OS.

3. **Trust at runtime:** Trust is maintained at runtime by enforcing IMA (Integrity Measurement Architecture) appraisal checks against all runtime processes and comparing those executables to an integrity list maintained in the TAm. All discrepancies are logged for action by administrators.

4. **Signed RPMs:** Trust is enforced for all IOS XR RPMs and third-party application RPMs by validating signatures on these RPMs before installation, based on the keys in TAm.

5. **Ability to visualize and report on trust:** Remote attestation capabilities are available to visualize and report on all the trust verification operations performed by the system during the lifecycle of the device.

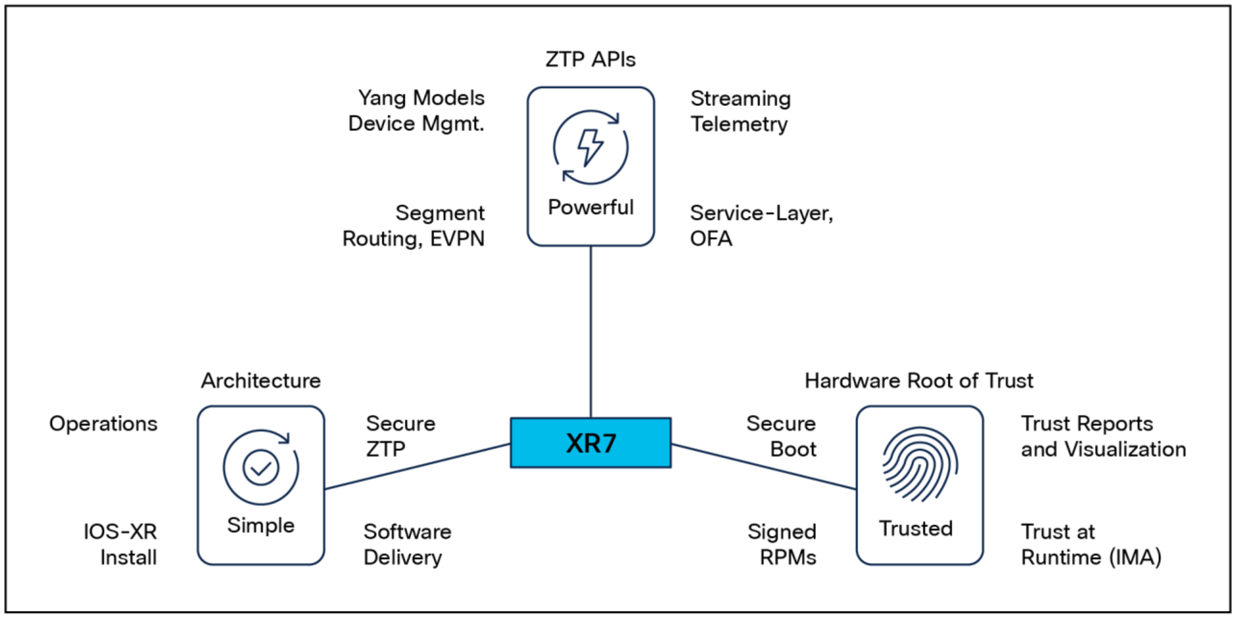

Figure 2.
Trustworthy NOS running on a trustworthy network device

As we elaborate on these concepts further, it should become apparent that IOS XR continues to evolve to meet the needs of service providers as they look to address the challenges of the future.

IOS XR: Simple

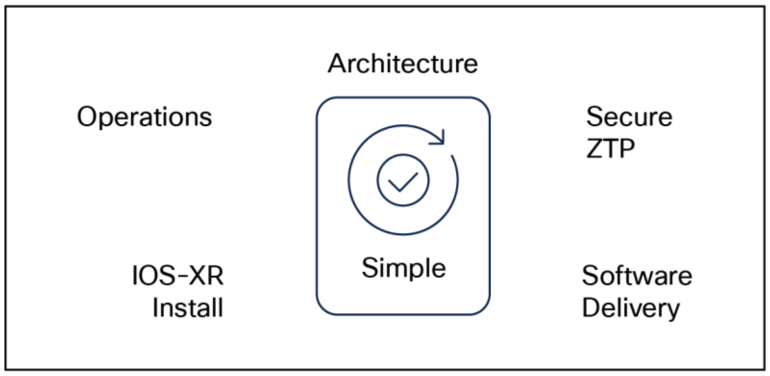

Figure 3.
IOS XR: Simple

A simplified architecture for the future

Simplicity and ease of use are some of the core principles behind IOS XR. These principles manifest themselves in various forms, one of which is that IOS XR is built on top of the WindRiver Linux 9 (WRL9) for distribution using Linux kernel version 4.8.28.

One of the major changes in IOS XR compared to IOS XR 6.X is the complete removal of the admin plane on the newer hardware platforms. This implies that the isolated XR control plane and admin plane container setup that IOS XR 6.X supported is no longer required. Consequently, the XR software architecture is shown below:

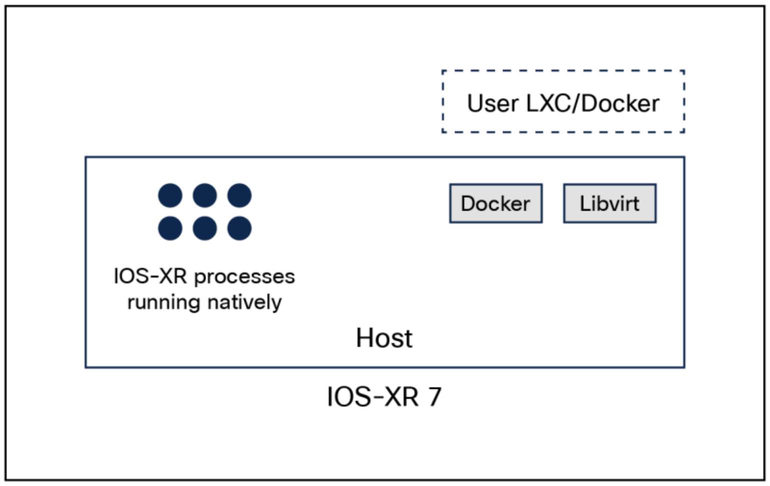

Figure 4.
IOS XR software architecture

- The IOS XR control plane processes run natively on the host.
- There is a single Linux shell for the whole system (the host layer).
- The Libvirt and Docker Daemons along with the respective virsh and docker clients are all running in the same environment (host shell).
- With the removal of the admin plane, all the system- and environment-level configurations and action capabilities now move to XR.

Support for Linux workflows

Over the last decade, SP network operations have evolved to use Linux community automation workflow installation techniques that originated within the server world. This evolution started in the web-scale service provider networks, where the companies focused on creating networks that are provisioned, managed, and manipulated using community-driven or home-grown automation tools, with almost no manual intervention.

These workflows have percolated into Day 0 rollouts as well. These Linux automations help convert a potentially weeks-long manual process of device onboarding into a process that could be completed in a matter of hours based on the level of automation used at Day 0. These workflows fall under the category of ZTP (Zero-Touch Provisioning), which is discussed at great length in a subsequent section.

The tools utilized by SPs have seen drastic changes: moving away from Telnet, expect-style, cli-based scripts to more deterministic, model-driven, SSH-based configuration management tools. These tools provide improved scalability and reliability and are often multivendor compatible.

IOS XR has consistently evolved to remain effective and offers application-hosting and Packet/IO capabilities, which enable Linux applications and tools to run on box as native binaries or containers (LXC/Docker) and allow these applications to send and receive traffic while running on fixed or modular chassis platforms. Additionally, capabilities such as ZTP, gRPC-based APIs (service-layer, YANG-modeled APIs), and telemetry were enabled because of the functioning Linux environment.

Golden ISO (GISO)

With IOS XR, the software can be delivered in a variety of different formats to fit the operational needs of a network operator. The idea was to take a base binary artifact (an ISO) for the Linux distribution being used on the server fleet, open it up, add packages and configuration files that will be used as a base across all the servers, and create a new ISO out of the combination. This combined ISO is called the "Golden ISO" since it represents the "Golden" configuration to be used for the server machines. Following the creation of the Golden ISO, servers would be PXE booted to get all of them to the base "Golden" state before configuration management tools would take over to set up role-based environments on the newly booted devices.

With IOS XR there will be several new capabilities that will be introduced to the Golden ISO build process. The evolution of the GISO process is shown below:

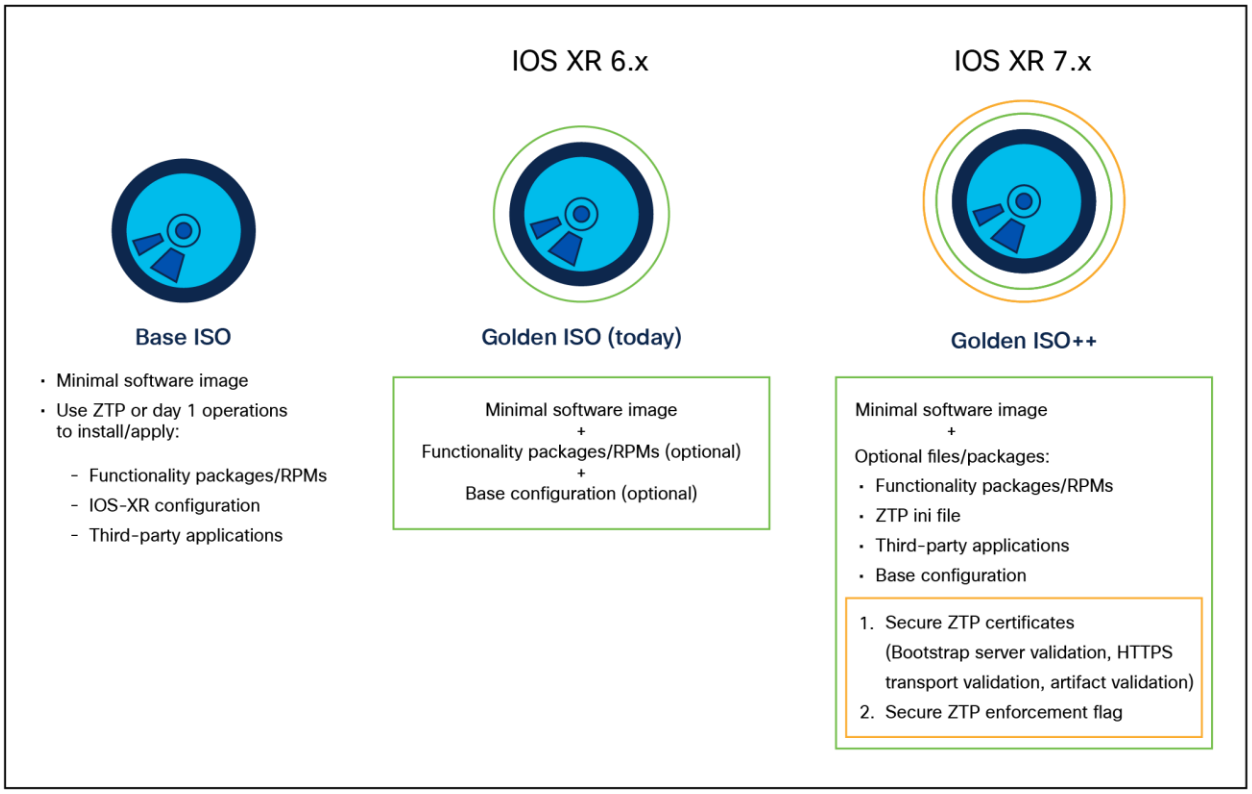
IOS XR 6.x
IOS XR 7.x
Base ISO
Golden ISO (today)
Golden ISO++
Minimal software image
Use ZTP or day 1 operations to install/apply:
Functionality packages/RPMs
IOS-XR configuration
Third-party applications
Minimal software image + Functionality packages/RPMs (optional) + Base configuration (optional)
Minimal software image +
Optional files/packages:
Functionality packages/RPMs
ZTP ini file
Third-party applications
Base configuration
1. Secure ZTP certificates (Bootstrap server validation, HTTPS transport validation, artifact validation)
2. Secure ZTP enforcement flag

Figure 5.
Evolution of the Golden ISO build process

With IOS XR, the GISO build process will enable the following artifacts to be added to the base ISO:

1. **IOS XR functionality RPMs:** RPMs of different optional components in IOS XR, like BGP, OSPF, ISIS, Telnet, etc.

2. **Base configuration:** A valid IOS XR configuration file can be added to the Golden ISO.

3. **ztp.ini file:** With IOS XR, a new type of file will be introduced for IOS XR ZTP workflows. The ztp.ini file is meant to allow certain default settings to be set for the ZTP process, enabling a user to influence the ZTP state machine through supported knobs.

4. **Third-party application RPMs:** Third-party Linux applications will be supported in the Golden ISO build process for IOS XR. For this purpose, Linux applications must be compiled for the underlying WindRiver Linux 9 (WRL9) distribution and packaged into an RPM, and the RPM MUST BE SIGNED to allow the IOS XR install process to install the RPM post boot.

5. **Secure ZTP artifacts:** We describe the secure ZTP artifacts in greater detail in a subsequent section. The basic requirement for secure ZTP to function is to have an owner (customer/network operator) certificate on the box when it boots. A TLS client certificate (optional) can also be packaged based on the workflow desired with secure ZTP. The secure ZTP enforcement flags are settings that the user may specify to modify the state machine of the ZTP process to either enforce secure ZTP or enable backward-compatible classic ZTP workflows.

The build tool for Golden ISOs for IOS XR is open source and can be found here: https://github.com/IOSXR/gisobuild.

Zero-Touch Provisioning (ZTP)

With IOS XR, Zero-Touch Provisioning (ZTP) in XR will be enhanced to support the requirements described in RFC 8572 (https://tools.ietf.org/html/rfc8572); what we refer to as “secure ZTP.” Zero-touch provisioning, as described earlier, is an important piece of the puzzle for most service providers looking to automate Day 0 provisioning of routers to help reduce OpEx associated with SMEs and staging facilities used today.

In most access deployments, network operators receive their purchased routers and usually ferry them to prestaging facilities as part of the first “truck” rollout. At the prestaging facility, the devices are typically configured manually with the help of SMEs to place bootstrap configurations on them. These preconfigured boxes are then shipped out to the installation site, often using third-party installers to simply set up and power on the devices that were preconfigured. This constitutes the second “truck” rollout. This may not be the exact workflow for all network operators but is fairly representative of the workflows in use across a majority of them.

The use of competent SMEs at prestaging facilities, the multiple truck rollouts, the post installation rollouts, and corrections in case of errors in the bootstrap configurations contribute to the consistently rising OpEx that most large-scale service providers have to incur. As the number of devices in these deployments continues to grow to meet consumer demands and newer 5G architectures, it is imperative to find techniques to reduce the OpEx as much as possible.

IOS XR ZTP is designed to address the following requirements:

1) **Tree-based buildouts:** To be topology agnostic, tree-based buildouts are necessary. There are no out-of-band management networks to work with, so every device in the network does not have L2 connectivity to a DHCP server. This implies that an already-provisioned device must act as a DHCP relay for the next device in the tree.

2) **Security is critical:** Since access devices typically go into insecure locations, it is necessary to establish trust during the onboarding process. At a higher level, this translates into the following requirements:

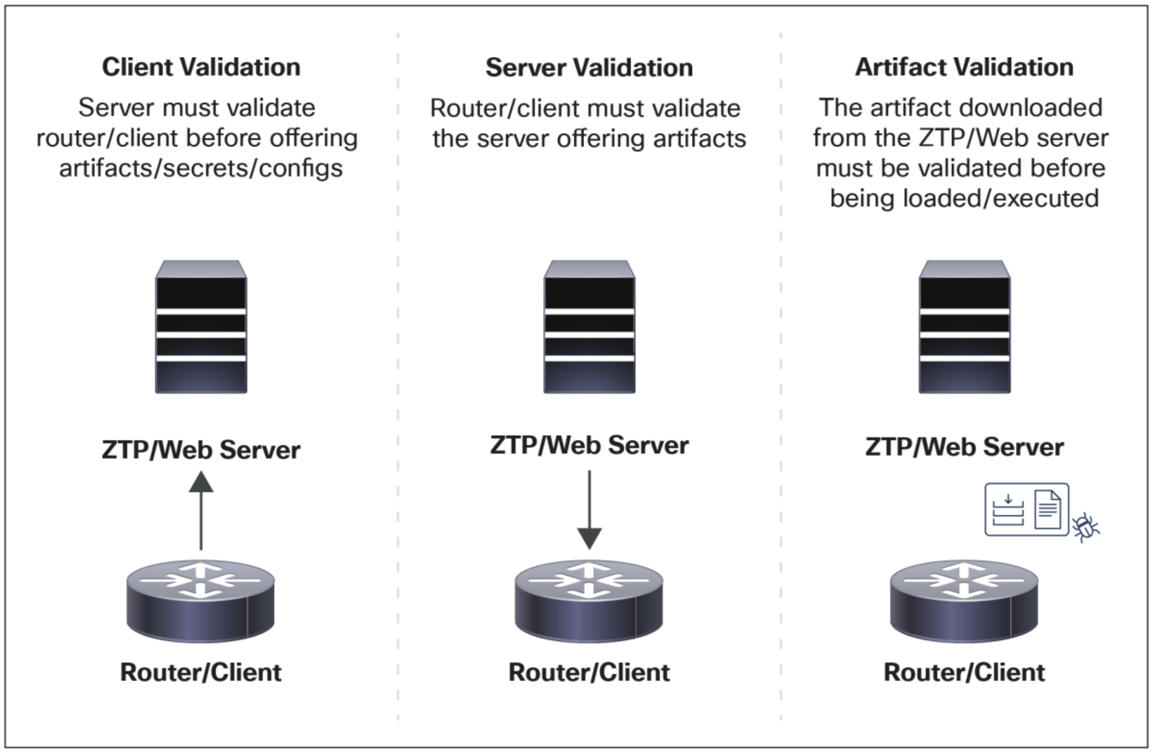
Client Validation
Server must validate router/client before offering artifacts/secrets/configs
Server Validation
Router/client must validate the server offering artifacts
Artifact Validation
The artifact downloaded from the ZTP/Web server must be validated before being loaded/executed
ZTP/Web Server
ZTP/Web Server
ZTP/Web Server
Router/Client
Router/Client
Router/Client

Figure 6.
IOS XR ZTP and establishing trust

3) **L2 reachability through VLAN discovery:** Often, the initial reachability services such as DHCP servers and web servers may only be reachable across a Layer 2 metro Ethernet cloud. Data (production) ports need to be utilized for ZTP in these networks due to absence of out-of-band management networks. For these reasons, VLAN discovery techniques will be implemented by IOS XR ZTP to discover the VLAN in use across the Layer 2 cloud and properly tag the DHCP request with the correct VLAN, post discovery, to proceed with ZTP.

Considering the goals described above, IOS XR implements concepts from RFC-8572 to enable secure ZTP capabilities in IOS XR. This support is threefold:

1) **Support for the secure ZTP YANG model** to implement interactions between the router and a RESTCONF-based ZTP server to download artifacts (config, scripts) to automatically provision the router.

2) **Support for a secure Golden ISO** that can package an owner certificate (associated with the network operator) to verify signatures on downloaded artifacts and validate the network domain that the device is trying to join based on responses from the ZTP server. The secure GISO also supports adding an optional TLS client certificate to help establish a TLS connection to the ZTP server to secure and encrypt subsequent interactions.

3) **An on-box infrastructure to relay a nontamperable Secure Unique Identifier (SUDI)**, such as a combination of a certificate and serial number to the ZTP server, and provide a secure location on the router to store owner certificates and keys.

IOS XR: Modern

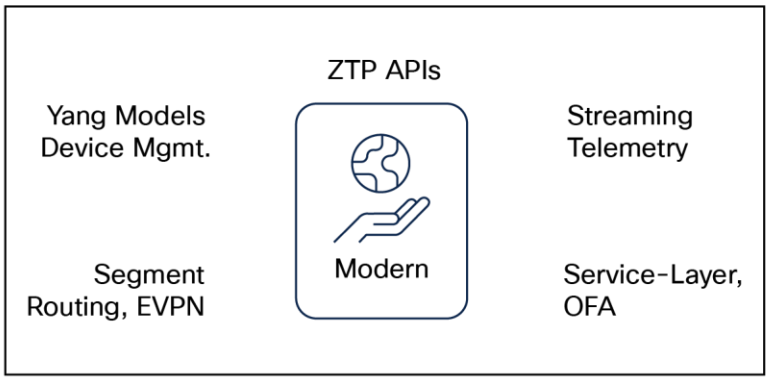

Figure 7.
IOS XR: Modern

IOS XR has consistently focused on building programmatic and model-driven interaction points within its stack to provide automatable interfaces that enable easier deployment and management of network devices. This design allows the capabilities of IOS XR to be easily extended through powerful integrations with external tools, applications, and controllers.

But it is hard to pigeon-hole these capabilities into a conversation about APIs alone. As technologies such as segment routing and EVPN have evolved, network operators have leveraged them to transition their legacy transport networks from protocol-heavy deployments (MPLS, RSVP-TE, Layer 2 and Layer 3 VPN deployments) to simpler, leaner solutions built on these technologies, often in tandem with external controllers and planning tools.

It is therefore important to understand these capabilities in the context of evolving network operations. Typically, we can classify network operations into three categories:

- **Day 0:** The initial rollout of devices, automated deployments through ZTP, registration with Day 1 management, and monitoring solutions. Zero-touch provisioning APIs and infrastructure play a key role here.
- **Day 1:** Role-based device configuration in the network, bring-up of centralized management solutions and controllers, mass deployment of applications and policies using scalable configuration management tools and automation. Model-driven, YANG-based APIs for device provisioning, model-driven telemetry capabilities for device monitoring, segment routing, and EVPN solutions to rollout services and policies in the network are the key elements utilized in Day 1 operations.
- **Day N:** Manipulate state in the network using real-time network and application information, remediate network degradation issues, raise alarms and schedule maintenance windows, analyze network telemetry data and patterns to evolve the network design, and meet future challenges. Custom controllers and applications that respond to network and application events, leveraging high-performance, lower-layer APIs such as service layer and Open Forwarding Abstraction (OFA) APIs, feedback loops implemented through telemetry, and device-provisioning APIs built on YANG are the typical components that drive innovation in Day N solutions.

Having understood how these APIs and technologies align with different operational paradigms described above, let’s break down the IOS XR network stack to understand where these capabilities lie.

Shown below are the different layers of the IOS XR stack with corresponding capabilities identified:

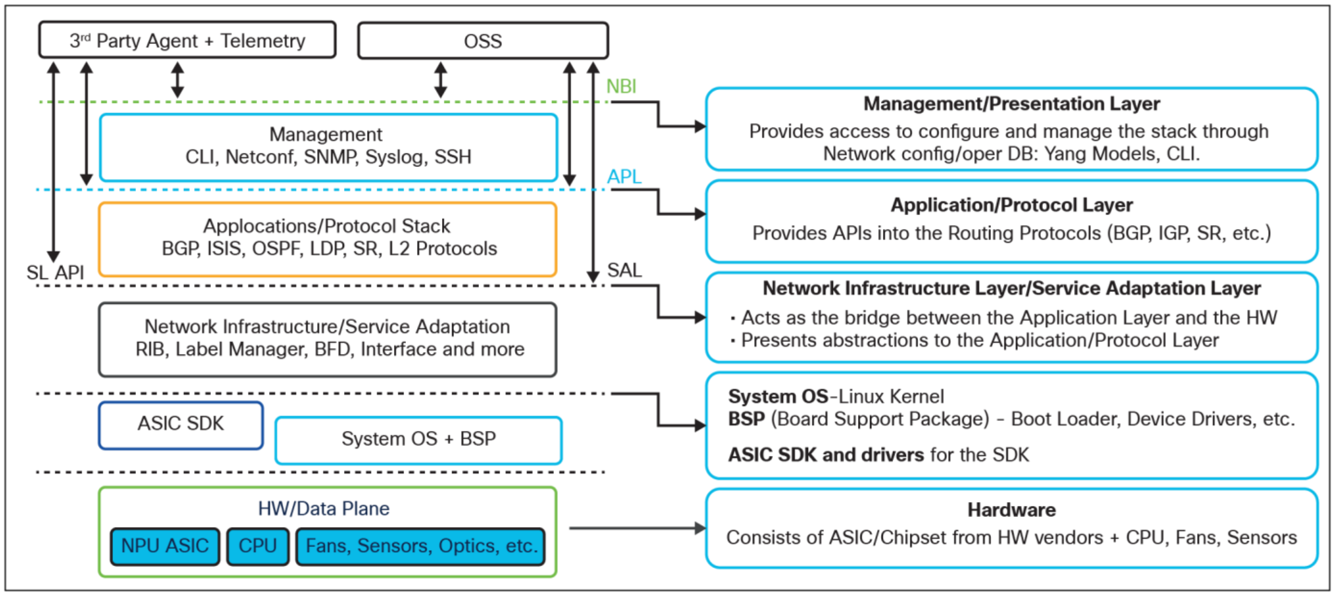
3rd Party Agent + Telemetry
OSS
NBI
APL
SAL
SL API
Management
CLI, Netconf, SNMP, Syslog, SSH
Applocations/Protocol Stack
BGP, ISIS, OSPF, LDP, SR, L2 Protocols
Network Infrastructure/Service Adaptation
RIB, Label Manager, BFD, Interface and more
ASIC SDK
System OS + BSP
HW/Data Plane
NPU ASIC
CPU
Fans, Sensors, Optics, etc.
Management/Presentation Layer
Provides access to configure and manage the stack through Network config/oper DB: Yang Models, CLI.
Application/Protocol Layer
Provides APIs into the Routing Protocols (BGP, IGP, SR, etc.)
Network Infrastructure Layer/Service Adaptation Layer
• Acts as the bridge between the Application Layer and the HW
• Presents abstractions to the Application/Protocol Layer
System OS-Linux Kernel
BSP (Board Support Package) - Boot Loader, Device Drivers, etc.
ASIC SDK and drivers for the SDK
Hardware
Consists of ASIC/Chipset from HW vendors + CPU, Fans, Sensors

Figure 8.
IOS XR stack and corresponding capabilities

- **Manageability layer:** This layer exposes the Command Line Interface (CLI) and YANG-modeled APIs for features and protocols in the network application layer. These YANG models are used for device provisioning and streaming telemetry. ZTP APIs are built on top of CLI and YANG APIs in this layer. This layer leverages an internal "database" within IOS XR called SysDB.
- **Network application/protocol layer:** This layer contains protocols (BGP, ISIS, OSPF, etc.) and features such as L2VPN, L3VPN, etc. This layer interacts with SYSDB to store the operational and configuration state of the protocols and features. The higher-layer components of segment routing and EVPN fall into this layer.
- **Network infrastructure/service adaptation layer:** Typically consists of components such as the RIB, Label Switch database, BFD, network interface handler, and APIs for controllers/agents. The API provided on top of this layer for end users is called the service-layer API. The infrastructure components of all XR control plane protocols (BGP, ISIS, OSPF, etc.), segment routing, and EVPN get mapped to this layer.
- **Hardware abstraction layer (OFA):** This layer interacts with the ASIC SDK (provided by the ASIC vendor) and handles programming of the data plane based on RIB state, or LSD (label switch database) state, etc. The Open-Forwarding Abstraction (OFA) API abstracts the capabilities of this layer for external applications.
- **Platform integration layer:** Typically consists of kernel and/or user-side drivers for devices such as fans, LEDs, sensors, etc. The integration of these device-specific drivers with the higher layers of the software stack happens at this layer. It this layer that allows a software stack to extract information (e.g., temperature) from sensors and influence the component state (e.g., fan speed).

IOS XR: Trustworthy

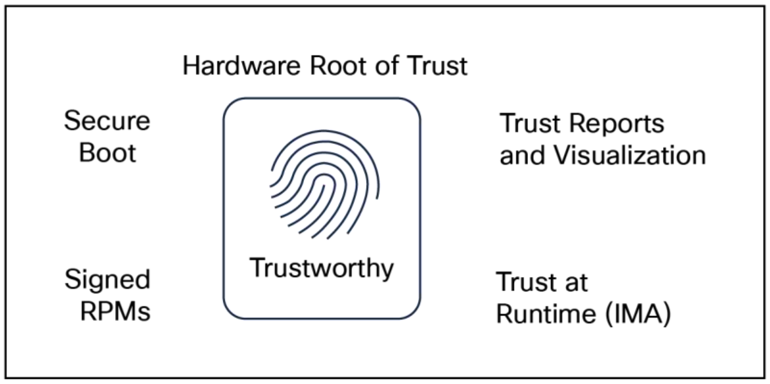

Figure 9.
IOS XR: Trustworthy

In IOS XR, a running theme around any discussion of device or NOS security is the idea of trust. The goal of IOS XR and its compatible hardware is to build a "Trustworthy Platform" where device and OS integrity are as important as the idea of security. In fact, it is important to note that no device can ever be considered secure: this is an ideal state, but not necessarily achievable. A level of trust can be measured, verified, and audited.

It is important to articulate the primary concerns associated with network device security:

- How does an operator ensure that the router is running only the software that was intended?
- How does one know that a router hasn't been physically altered since Cisco built it?

It is possible to address these concerns by immediately building security controls such as enabling only signed software to install and run or having known good values for the hardware components to match and verify during boot. Taking it a step further and ensuring that security controls are backed by tamper-proof hardware that stores certificates and keys and the known good values to use for validation helps provide complete validation. Attestation capabilities allow the verification results to be visualized and reported, which can prove the ability to trust the component and network.

To address these concerns and understand the trust paradigm within IOS XR and its associated platforms, we need to understand the chain of trust for a typical network device.

Hardware root of trust

The concept is simple: if we trace back the boot process, then pushing the root of trust as far back as possible ensures the highest level of trust in the system.

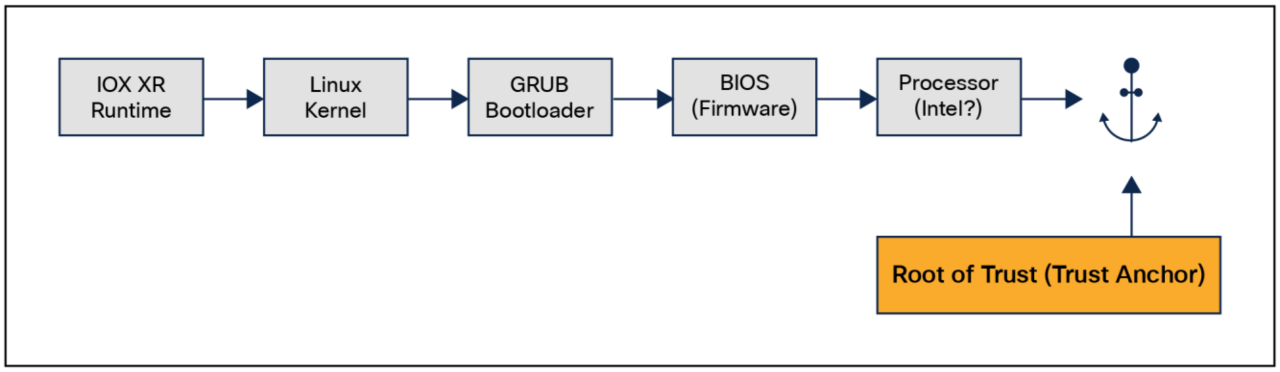

Figure 10.
Hardware root of trust

There are well-known attacks and vulnerabilities that compromise the runtime OS, kernel, and bootloader (GRUB). Secure boot as a concept is laid out in the UEFI specification 2.3.1 and expects trust to begin at the BIOS stage, subsequently checking the bootloader, kernel, and then the OS. But we've seen UEFI rootkits in the wild that can compromise the BIOS, thereby jeopardizing the secure boot process. Further, the scope of security issues for Intel ME and AMT, both BIOS extensions on Intel hardware, is significantly increasing. Research has already been presented about problems in the UEFI firmware ecosystem and how easy it is to deliver and install implant/rootkit.

This implies that to properly secure the boot process, validation checks have to begin much earlier in the boot cycle, even before the BIOS boots. With IOS XR platforms, we do exactly that. Cisco hardware platforms execute what we call a hardware-anchored secure boot.

Cisco secure boot versus UEFI secure boot

Cisco secure boot is different by anchoring the secure boot process in hardware, thus providing the most robust security. This is an improved security posture because hardware modification is difficult, expensive, and not easy to conceal, even if the hacker has physical possession of the device.

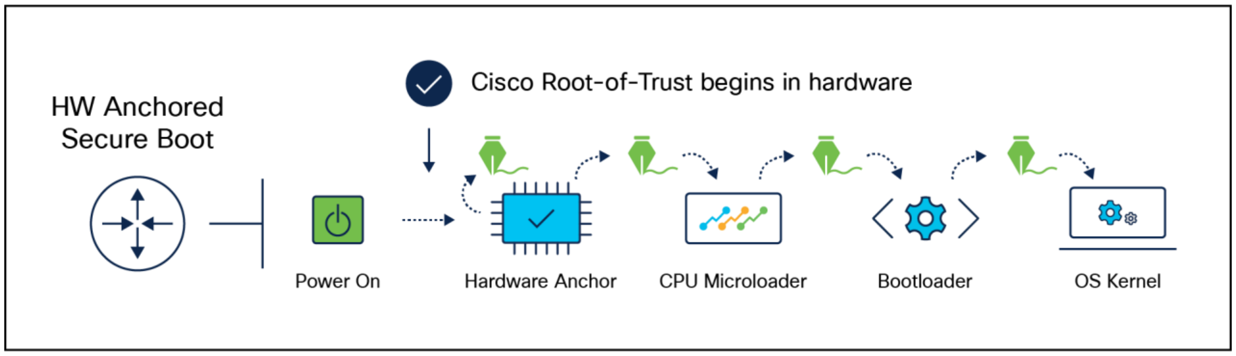

Figure 11.
Cisco secure boot

This process creates a “chain of trust” from microloader through bootloader to the operating system, establishing the authenticity of the software. If any digital signature check fails, the Cisco device will not let that software boot, so that malicious code will not run.

Cisco Trust Anchor Module (TAm) versus industry-standard TPM

When the Cisco hardware-anchored secure boot has authenticated the software as genuine Cisco software, the operating system then queries the TAm to verify that the hardware is authentic. It does this by cryptographically checking the TAm for a Secure Unique Device Identifier (SUDI) that only Cisco provided.

The SUDI is permanently programmed into the TAm and logged by Cisco during Cisco’s closed, secured, and audited manufacturing processes. This programming provides strong supply chain security, which is particularly important for embedded systems like routers and switches.

In contrast, the industry standard TPM has similar functions to those of the TAm, but it is not permanently programmed during manufacture with a unique device identifier. Building an SUDI in this instance is left to the end user and requires user intervention and development, which does provide flexibility, but increases the risk of not identifying unauthorized supply chain modifications.

Cisco environmental sustainability

Information about Cisco’s environmental sustainability policies and initiatives for our products, solutions, operations, and extended operations or supply chain is provided in the “Environment Sustainability” section of Cisco’s Corporate Social Responsibility (CSR) Report.

Reference links to information about key environmental sustainability topics (mentioned in the “Environment Sustainability” section of the CSR Report) are provided in the following table:

Sustainability topic	Reference
Information on product material content laws and regulations	Materials
Information on electronic waste laws and regulations, including products, batteries, and packaging	WEEE compliance

Cisco makes the packaging data available for informational purposes only. It may not reflect the most current legal developments, and Cisco does not represent, warrant, or guarantee that it is complete, accurate, or up to date. This information is subject to change without notice.

Cisco Capital

Flexible payment solutions to help you achieve your objectives

Cisco Capital makes it easier to get the right technology to achieve your objectives, enable business transformation and help you stay competitive. We can help you reduce the total cost of ownership, conserve capital, and accelerate growth. In more than 100 countries, our flexible payment solutions can help you acquire hardware, software, services and complementary third-party equipment in easy, predictable payments. Learn more.

Document history

New or revised topic	Described in	Date
Updated IOS XR naming for consistency	All sections	April 2024

Printed in USA C78-743014-02 05/24