Introduction to IOS-XR 6.0

Joachim Jerberg Jensen – joajense@cisco.com
System Engineer, Global Service Providers
CCIE SP #42403
Agenda

• Introduction
• Software Architecture Overview
• Flexible Packaging
• Application Hosting
• Configuration, Monitoring and Troubleshooting
• Conclusions
Introduction
… coming to a platform closer to you

Q4 CY15
NCS5508
NCS5001
NCS5002
NCS1002
NCS5011
NCS5502
NCS5501
NCS002 w/macsec
NCS 6000

Q2 CY16
ASR9000(*)

Q4 CY16
NCS1002 w/ macsec

* On ASR9k, 32-bit QNX images and 64-bit linux images will be supported
Guiding Principles for IOS-XR operational enhancements

**Bring Your own Application**
- Provide a platform on which customers can host their apps (3rd party apps, customer apps, cisco apps)

**Automatable interfaces**
- Provide visibility into the device through machine friendly interfaces

**Open architecture \(\rightarrow\) Decrease tool chain variance**
- Fit into customer’s operational workflow
## Pillars of IOS-XR operational enhancements

<table>
<thead>
<tr>
<th>Visibility &amp; Telemetry</th>
<th>Evolved Programmability</th>
<th>Application Hosting</th>
<th>Flexible Platform and Packaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Operational Data, Deep analytical hooks</td>
<td>• Data accessible via published model driven interfaces</td>
<td>• Ability to run 3rd party off the shelf applications built with Linux tool chains</td>
<td>• Packages can be inspected on box using standard tool chain (RPM tools)</td>
</tr>
<tr>
<td>• Policy-based, flexible, Push Model</td>
<td>• Machine friendly</td>
<td>• Run custom applications inside an LXC container on the 64-bit Linux host</td>
<td>• Automated package dependency checkers</td>
</tr>
<tr>
<td></td>
<td>• Enables automation @ scale</td>
<td></td>
<td>• Open Bootloaders (iPXE) and end-to-end auto-provision</td>
</tr>
</tbody>
</table>
Software Architecture
IOS XR 6.0 introduces a new software infrastructure offering

- 64-bit OpenEmbedded Linux support.
  - Processes containerization.
  - Brings in standard Linux toolchain.
  - Third-Party Applications Support.

- NCS 5500, NCS 5000 and NCS1002 will support only 64-bit Linux

- ASR 9000 will still have 32-bit QNX support
Introduction to Containers

- LXC (LinuX Containers) let you run a Linux system within another Linux system.
- A container is a group of processes on a Linux machine.
- Those processes form an isolated environment.
- Inside the container, it looks like a Virtual Machine.
- Outside the container, it looks like normal processes running on the system.
- Containers look like Virtual Machines, but are more efficient.
Why Containers?

- **They are Fast** ➔ Deploy and Boot in less than one Second vs Minutes for Virtual Machines
- **They are Lightweight** ➔ Only a few MB of Disk Space per Container vs several hundred MB for traditional Virtual Machines.
- **They provide Similar Services as VMs** ➔ Each container has:
  - Its own network interfaces
    - Can be bridged, routed... just like with KVM.
  - Its own filesystem
    - e.g.: RedHat host can run Debian container.
  - Isolation (security)
    - Two containers can't harm (or even see) each other.
  - Isolation (resource usage)
    - Soft & Hard quotas for RAM, CPU, I/O.
Linux Containers – Kernel Requirements

• To create a virtual environment, containers use the following kernel features.

1. Namespaces: Partition essential kernel structures to create virtual environments:
   • pid (processes)
   • net (network interfaces, routing...)

2. Control Groups: Limit, account, and isolate resource usage:
   • Exposed through a virtual filesystem

3. Chroot: operation that changes the apparent root directory of the container process.
IOS-XR Container Architecture

Modular Router

- LXC Control Plane
- LXC Admin Plane
- LXC Third Party
- 64-bit Host OS
- Routing Processor
- Fabric
- Line Card

Fixed Router

- Control Plane
- Admin Plane
- Third Party
- 64-bit Host OS
- Routing Processor
- Fabric
- Front Panel
IOS-XR Container Roles: The Host

- Runs Yocto based 64-bit Open Embedded Linux kernel.
  - Built using Windriver 7

- The main functions of the host are:
  - Interact directly with the underlying hardware.
  - Provide kernel services for the containers.
  - Provide libraries, tools, and utilities to help launch, monitor, and maintain containers.

Provide the network infrastructure to allow containers to communicate.
IOS-XR Container Roles: The Control Plane

- The heart of IOS-XR 6.0
- Runs a Yocto based 64-bit OELinux composed of 2 types of packages:
  1. Cisco developed packages for core network functions (BGP, MPLS, etc.)
  2. Yocto packages for standard Linux tools and libraries (bash, python, tcpdump, etc.).
IOS-XR Container Roles: The Admin Plane

- Runs a Yocto based 64-bit Linux.
- Provides services that were originally provided by the admin mode of XR.
- Runs processes responsible to perform system diagnostics, monitor environmental variables, and manage hardware components.
- First container to be booted by the host, and is responsible for the start and maintenance of all the other containers in the system.
IOS-XR Container Roles: Third Party

- Runs any 64-bit Linux distribution.
- Launched from the XR container using virsh and libvird.
- Access Network Interface through the Third Party Network Name Space (TPNNS).
XR Boot Process

• XR 6.0 image will be released in the form of bootable self-extracting ISOs
  • Similar to any Linux distribution.

• NCS 5000 and 5500 possess a BIOS that offers NetBooting using iPXE.
  • iPXE is an open-source network boot firmware that supports:
    1. Booting from HTTP/HTTPS.
    2. Controlling the boot process via scripts.
    3. Performing image validation.

• XR 6.0 comes with an Auto-Provision process
  • Executed at the end of the control-plane boot sequence.
  • Executed inside the Shell.
  • Can execute Scripts or apply Static Configuration.
IOS-XR Boot Process with iPXE

1. DHCP Request
2. GET Image-new.ISO
3. DHCP Request
4. GET script-SN.sh or config-SN.txt
5. GET addon scripts/packages/configuration

DHCP SERVER
- IP address
- Next-server
- IP address
- Next-server
- Filename=http://<http-srv>/AutoProv-SN.sh
- Filename=http://<http-srv>/Config-SN.txt

HTTP SERVER
- Image-new.ISO
- script-SN.sh
- config-SN.txt
- Additional Scripts
- Packages, etc…

iPXE boot

iPXE

XR Install

XR Boot

AutoProvision execution

Apply Configuration

Execute script
host NCS5500-rp0 {
  hardware ethernet e4:c7:22:be:10:ba;
  fixed-address 192.168.0.10;
  If exists user-class and option user-class = "iPXE" {
    # Image request, provide ISO
    filename "http://192.168.0.10/images/ncs5500-mini-x.iso-r6.0.0";
  }
  elsif exists user-class and option user-class = "exr-config" {
    # Auto-provision request, provide script or configuration
    filename "http://192.168.0.10/scripts/ncs5500-rp0.sh";
  }
}
NCS 5000 on board ports

- Management Ethernet 0 and 1 is mapped to IOS-XR Permanently
- Console port uses Console mux feature (By using CTRL+O we can switch between the HOST OS(Linux), Admin LXC and XR LXC
NCK5K BIOS

- NCS5K BIOS can be accessed by either pressing “esc” or “F12” Key when Box is Powered.
- It will list out following options
  1. UEFI: SMART eUSB HS-SD/MMC (Internal Disk)
  2. UEFI: Sony Storage Media 0100 (External USB)
  3. UEFI: Built-in EFI IPXE
  4. UEFI: Built-in EFI Shell
- Boot order can be Changed on the BIOS Menu
Boot With External USB

- Download the uncompressed image file in external USB.
- It contains folder structure with following files
  - EFI/Cisco/ncs5k-mini-x.iso
  - EFI\Cisco\grub.efi
  - EFI\Cisco\bootx64.efi
  - EFI\Cisco\grub.cfg
What will you see

## iPXE in action

iPXE 1.0.0+ (3e573) -- Open Source Network Boot Firmware -- http://ipxe.org

Features: DNS HTTP TFTP VLAN EFI ISO9660 NBI Menu

Trying net0...

net0: c4:72:95:a6:14:e1 using dh8900cc on PCI01:00.1 (open)

  [Link:up, TX:0 TXE:0 RX:0 RXE:0]

Configuring (net0 c4:72:95:a6:14:e1).................. Ok

Talking to DHCP/PXE server to obtain network information

net0: 1.37.1.101/255.255.0.0  gw 1.37.1.0
net0: fe80::c672:95ff:fea6:14e1/64
net1: fe80::c672:95ff:fea6:14e3/64 (inaccessible)

Next server: 1.37.1.235

Filename: http://1.37.1.235/nkhade/skywarp-mini-x.iso

http://1.37.1.235/nkhade/skywarp-mini-x.iso... 58%
Flexible Packaging
Networking Software Delivery Direction

What
- Server-like Workflows
- Modular, Disaggregated
- Integration with Stack
- Reduced Delivery Cycles

How
- Linux Operations
- Business Driven Packaging
- Programmatic Validation
- Targeted Validation
RPM: XR New Package Format

- RPM Package Manager is the new Package format starting with IOS-XR 6.0.
- Packages are placed in a reachable repository and accessed via FTP/SFTP/SCP/TFTP or HTTP or pre-staged on the box.
- Third Party packages are installed with RPM or YUM inside the Shell.
- IOS-XR packages are installed with “install update/upgrade”.
- Install commands are a wrapper around YUM to provide multi-arch support.
- Both YUM and install commands provide dependency verification/resolution.
Anatomy of RPM Packages

- **RPM**
  - Archive
  - MetaData
  - Scriptlet

- CPIO binary Archive
- Describe package contents
- Install structure
- Dependencies
- Pre and Post Install Instructions

Database of installed packages

/var/lib/rpm
XR Packages Naming Convention

Package: `<name>-<version>-<release>.<architecture>.rpm`

- ncs5500-mpls-1.0.0.0-r600.x86_64.rpm

SMU: `<name>-<version>-<release>.<defect>.<architecture>.rpm`

- ncs5500-mpls-1.0.0.1-r600.CSCab12345.x86_64.rpm
Packaging Repository

www.cisco.com

Platform
Release
XR Software
XR SMUs
Svc Pak

Mini ISO + Pkgs
Mini ISO + k9 pkg + Pkgs
Full K9 ISO
Full ISO

<REPO_ROOT>
6.0.0.17L
ncs-5500-mini-x.iso-6.0.0.17L.iso
ncs-5500-mpls-te-rsvp-1.1.0.0-r60017L.x86_64.rpm
ncs-5500-bgp-1.0.0.0-r60017L.x86_64.rpm
ncs-5500-eigrp-1.0.0.0-r60017L.x86_64.rpm
ncs-5500-k9sec-1.0.0.0-r60017L.x86_64.rpm
ncs-5500-mgbl-2.0.0.0-r60017L.x86_64.rpm
ncs-5500-mpls-1.1.0.0-r60017L.x86_64.rpm
ncs-5500-m2m-1.0.0.0-r60017L.x86_64.rpm
ncs-5500-mpls-1.1.0.1-r60017L.CSCab12345.x86_64.rpm

Local Repository

www.cisco.com
## Updating XR Packages

<table>
<thead>
<tr>
<th>Command Line</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>install update source &lt;repository&gt;</td>
<td>No package specified, update latest SMUs of all installed packages</td>
</tr>
<tr>
<td>install update source &lt;repository&gt; ncs5500-mpls</td>
<td>Package name specified, will install that package, update all latest SMUs of that package(s) (along with its dependencies).</td>
</tr>
<tr>
<td>install update source &lt;repository&gt; ncs5500-mpls-1.0.0.1-r622.CSCab12345.x86_64.rpm</td>
<td>SMU installation: the SMU will be downloaded and installed (along with its dependent SMUs).</td>
</tr>
<tr>
<td>install update source &lt;repository&gt; ncs5500-mpls-1.0.2.0-r622.x86_64.rpm</td>
<td>Asynchronous package upgrade, that package will be installed (along with its dependent SMUs). Available in future release.</td>
</tr>
</tbody>
</table>
# Upgrading XR Packages

<table>
<thead>
<tr>
<th>Command Line</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>install upgrade source &lt;repository&gt; version 6.1.1</code></td>
<td>Upgrade the base image to the specified version. All installed packages will be upgraded to same release as the base package.</td>
</tr>
<tr>
<td><code>install upgrade source &lt;repository&gt; version 6.1.1 ncs5000-mpls-1.0.2.0-r623.x86_64.rpm</code></td>
<td>Perform install upgrade and install update for a specific package(s) in one operation.</td>
</tr>
</tbody>
</table>
**PIE Install**

- No dependency management
- Offline process required to copy packages
- Require multiple operations
  - install add
  - install activate
  - install commit
- CSM for package content

**RPM Install**

- Dependency management
- Online process over secure transport
- Single operation
  - install update or install upgrade
- On-box / Off-box package inspection using rpm tool
  - Description
  - Dependencies
  - Content
Installing and Updating Third Party Packages

• Third Party Packages are traditional Linux tools available from the Shell
  • Communication: lighttpd, openssh, wget, curl, etc.
  • Programming: python, ruby, perl, etc.
  • Utilities: sed, gawk, tar, gzip, vi, etc.

• Additional packages provided by vendors (No Cisco Support)
  • Chef
  • Puppet

• Installed using yum or rpm

  yum-config-manager --add-repo=http://192.168.0.254/XR/6.0.0
  yum install chef -y
Application Hosting
Third Party Network Name Space

• Provide visibility of fabric attached interfaces outside of XR CLI.
• Available to processes in the XR containers or Third Party containers.
• Requires that the interface is Up with a valid IP address.
• Routing handled by XR.

[xr-vm_node0_RP0_CPU0:~]$ip netns exec tpnns bash
Third Party Container Application Workflow

- Create the Container archive on a Linux Server.
- Copy the archive file to /misc/app_host.
- Unarchive in a rootfs directory.
- Create XML file specifying LXC parameters.
- Run virsh command.

```
virsh -c lxc+tcp://10.11.12.15:16509 create <XML File>
```
Configuration, Monitoring and Troubleshooting
Streaming Telemetry: Introduction

Where Data Is Created

sensing & measurement

Where Data Is Useful

storage & analysis

SNMP

syslog

CLI

Scale Issues

Subject to Change

Unstructured
Streaming Telemetry
Design Vision

Performance
• Get as much data off the box as quickly as possible

Coverage
• Grant full access to all operational data on the box

Automation
• Serialize the data in a flexible, efficient way that fits customers automated tools
Telemetry

- Common modeling language: Goal is YANG (experimental SysDB name space in 6.0)
  - Describes monitoring data structure and attributes
- Push Model
  - Stream data continuously with incremental updates based on subscriptions
- Data delivery:
  - JSON (compressed) inside TCP.
  - Google Protocol Buffer inside UDP.
  - Google Protocol Buffer inside gRPC.
- Observe network state through a time-series data stream
Streaming Telemetry Model

- **Telemetry Configuration**
  - Described in JSON.
  - Define one or multiple collection group(s).
  - Each group contains a rate and a pointer to one or multiple objects in the experimental SysDB path (6.0 only)

- **Telemetry Policy**
  - Define the encoder, transport and the receiver(s) for each policy.

- **Telemetry Agent**
  - XR process that runs automatically and looks for registered policies to act on.
Streaming Telemetry Example

```json
{
    "Name": "GenericCounters",
    "Metadata": {
        "Version": 25,
        "Description": "This is a sample policy to demonstrate the syntax",
        "Comment": "This is the first draft",
        "Identifier": "<data that may be sent by the encoder to the mgmt station>"
    },
    "CollectionGroups": {
        "FirstGroup": {
            "Period": 30,
        }
    }
}
```
Network Configuration Protocol (NETCONF)

- NETCONF provides mechanisms to install, manipulate, and delete device configuration.
- It uses an XML based data encoding for the configuration data as well as the protocol messages.
- NETCONF protocol operations are realized as remote procedure calls (RPCs) over SSH.

RPC calls are divided in 4 groups of methods: Retrieve, Configure, Copy, and Delete. Additional methods can be supported.
Programmatic Interface

Secure Transport → Connection Oriented

SSHd → XR Programmatic Interface

SSH Proxy Server → NETCONF Agent

YANG Framework

RPC-based communication model

Negotiate Capabilities

Configuration Manager

Backend Apps

SysDB

SysDB Client Library

XR YANG

Web Server

RESTCONF Agent

Oriented RPC-based communication model

Negotiate Capabilities

Secure Transport
Conclusions
Key Takeaway

• No Change in traditional CLI Operation
• In 6.0, available on NCS-5500, NCS-5000 and NCS1000 Platform
• In 6.1.1, available on ASR9k (traditional 32-bit QNX offering will still be supported)
• Operational enhancement for Automated Operations
• Operating System change for Open Source tooling
IOS XR 6.0 Operational Enhancements

- Visibility & Telemetry
- Evolved programmability
- Application Hosting
- Flexible Platform & Packaging
We’re ready. Are you?