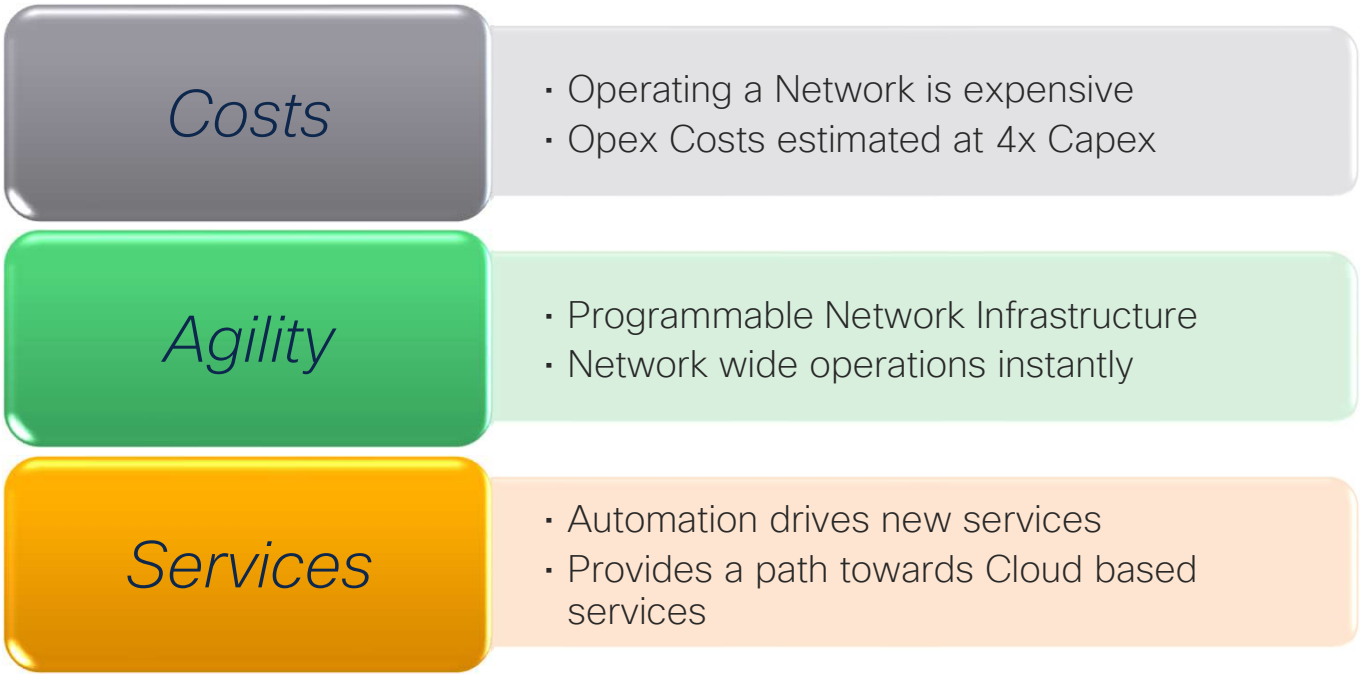
Achieving Improved Operational Simplicity with IOS XR

Amrit Hanspal, Product Manager

Mass scale Infrastructure Group (MIG)
March 2021

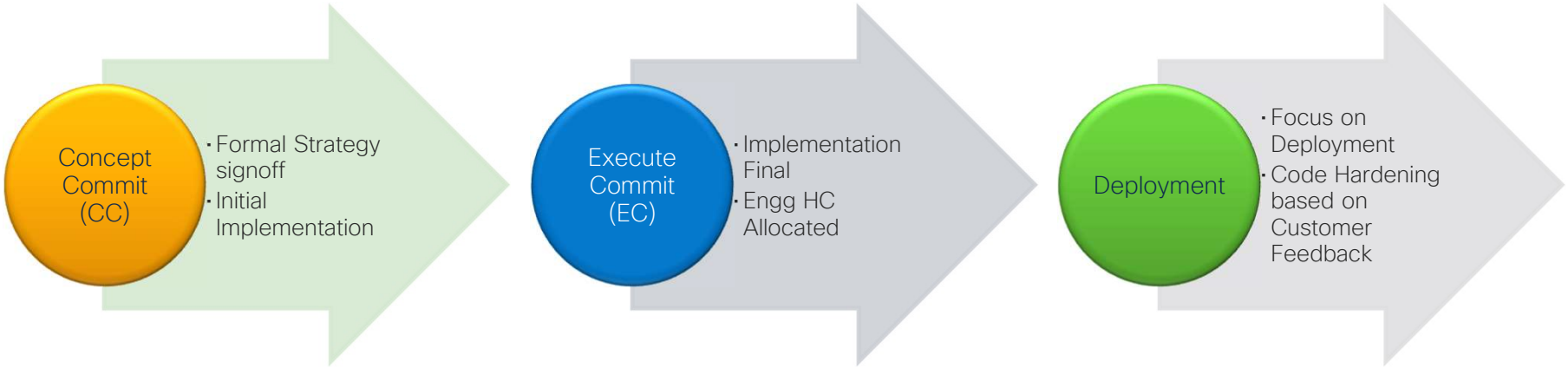
Why Operational Simplification?



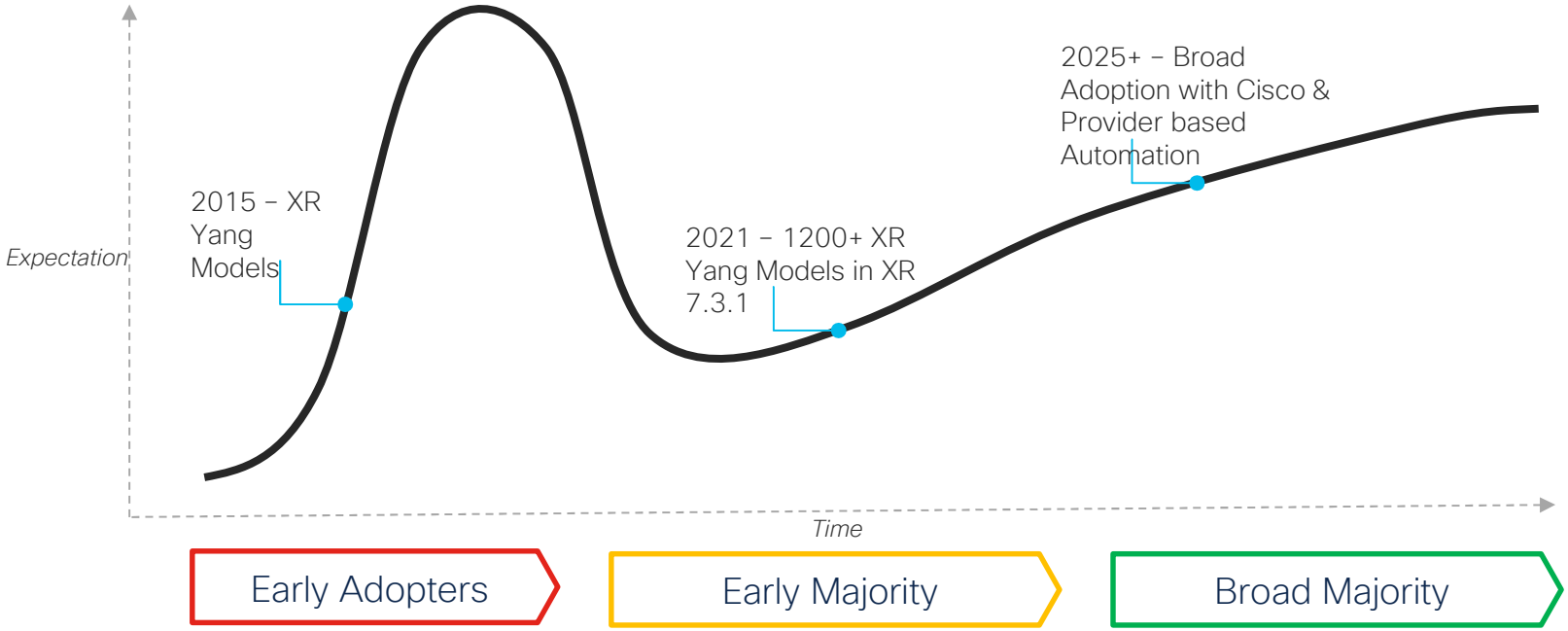
Layout

- Overview
- IOS XR Programmable Infrastructure
  - Manage
  - Monitor
  - Control
- Automation

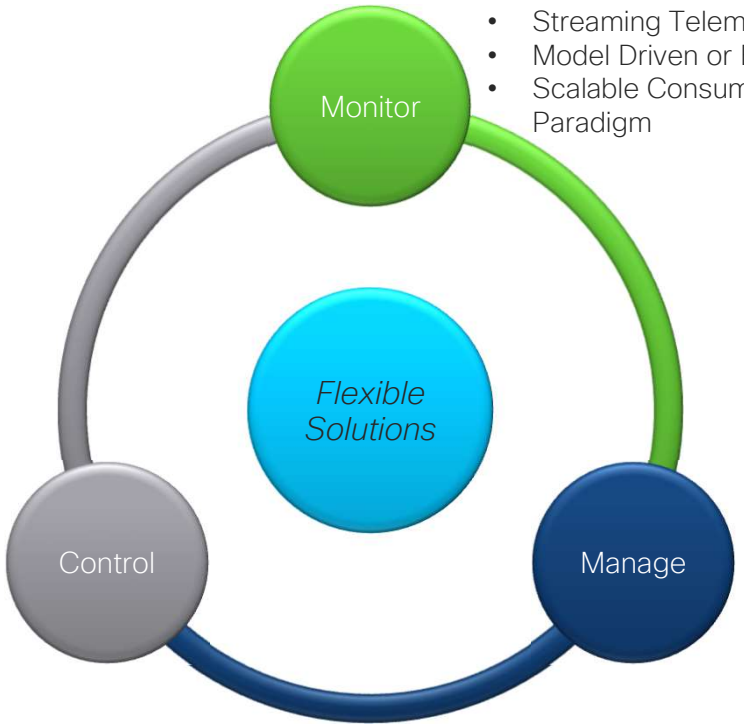
Understanding Roadmap Slides



XR Programmability – As of 2021



IOS XR Programmability – Key Components



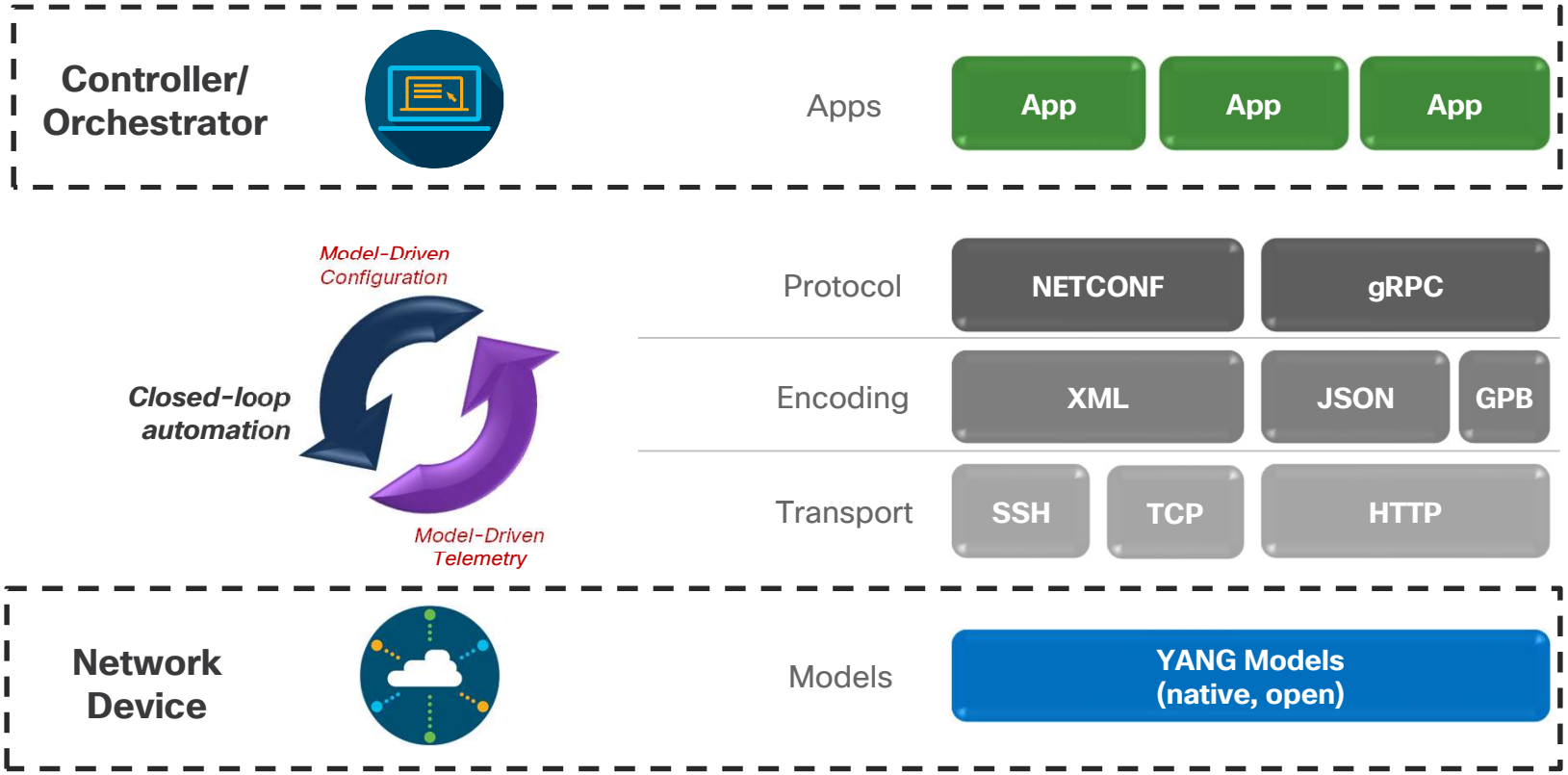
- Streaming Telemetry
- Model Driven or Event Driven
- Scalable Consumption driven Paradigm

- Route/Forwarding level Control functions
- RIB/FIB Level (SL-API) & Protocol Level (PCEP, BGP FS)

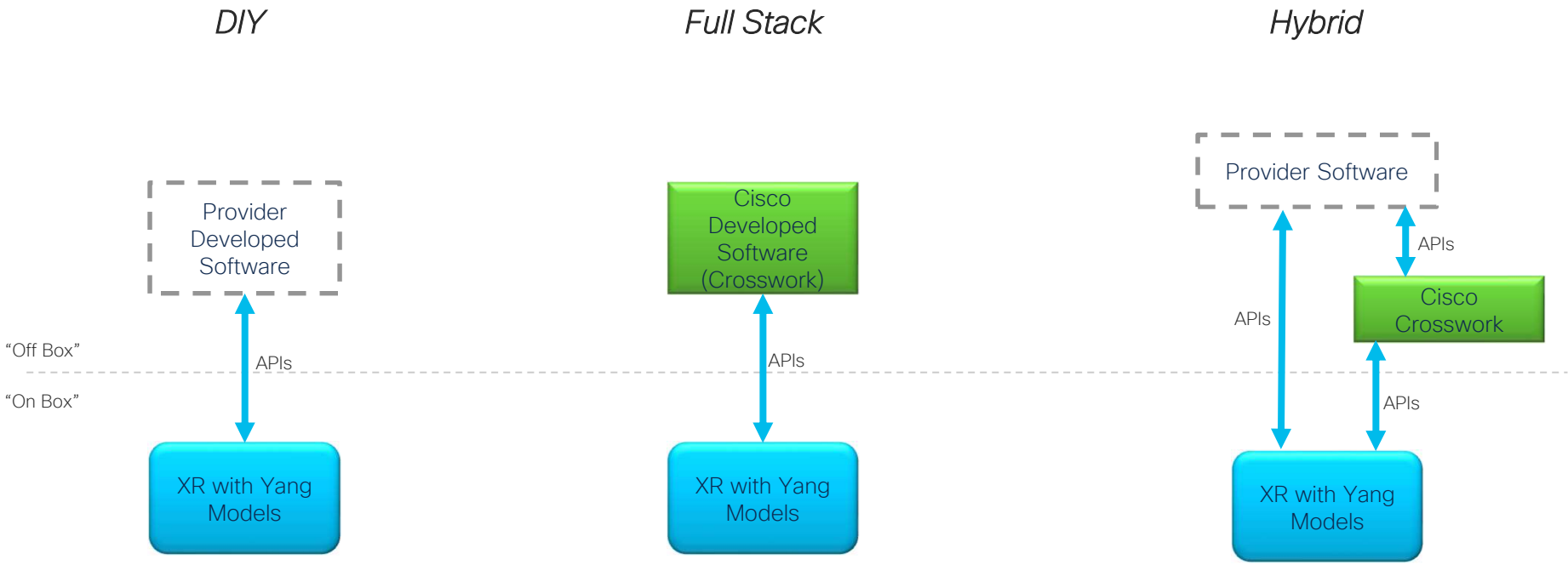
- Device & Network level Configuration
- Scope
  - Day 0 (ZTP)
  - Day 1 (Operate)
  - Day 2 (Service)

Flexible Solutions = Cisco Internal (CrossWork, NSO) or External (HomeGrown or 3rd Party)

IOS XR Programmability – A Primer



Customer Deployment Styles



XR Programmability Infrastructure

Models – “Manage”

IOS XR Models – Styles

Native Models

OpenConfig Models

Comes integrated in IOS XR today (~1100 Native and ~100 OC models – XR 7.3.1)

Note: There is No “one standard” – In Programmability it does NOT matter – APIs matter

https://github.com/YangModels/yang/tree/master/vendor/cisco/xr

IOS XR – Configuration Models

XR Native

Unified

- XR or platform specific
- Full coverage of device functionality
- Single abstraction for YANG and CLI
- Full parity and deterministic coverage
- Same help/doc strings
- Expected to be current

Non-Unified

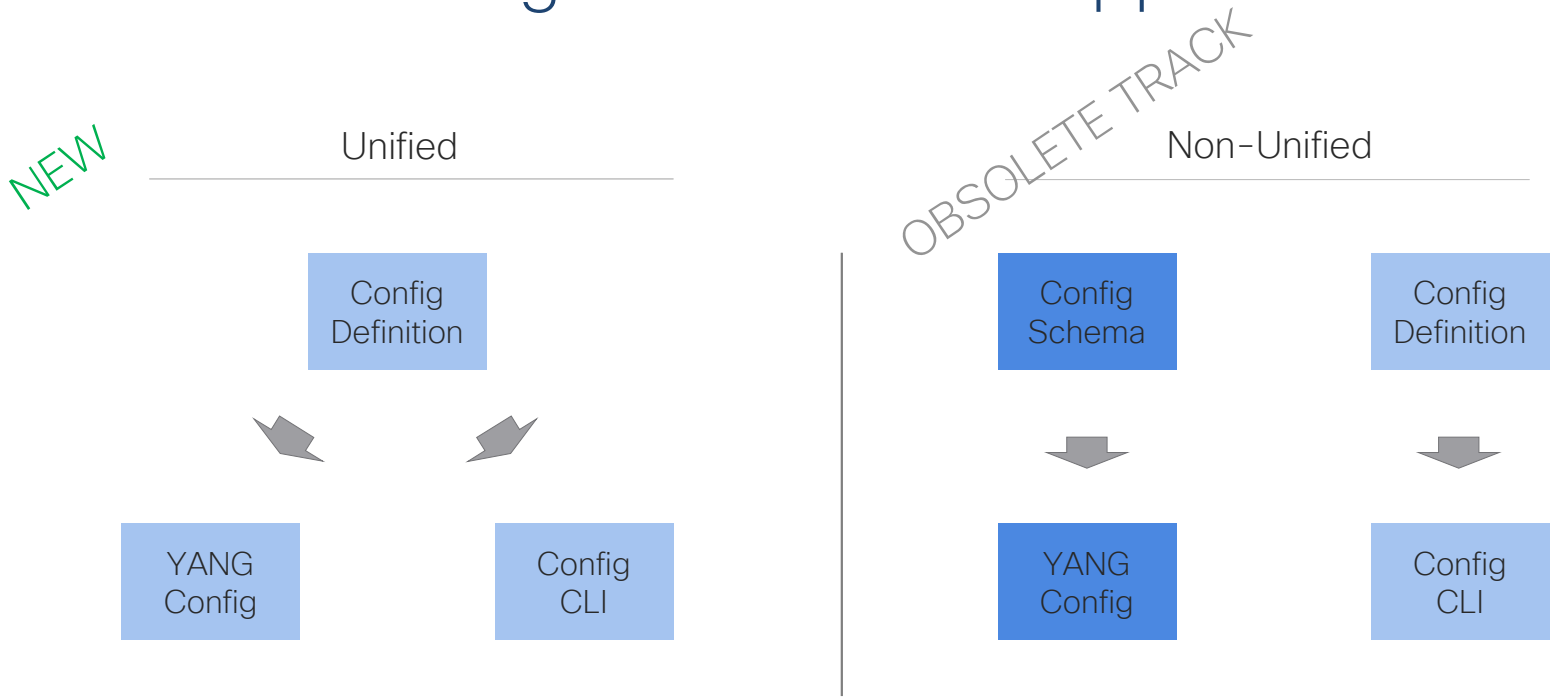
- XR or platform specific
- Full coverage of device functionality
- Different abstractions for YANG and CLI
- Independent testing of parity and coverage
- Expected to be obsoleted

Open

OpenConfig

- Vendor neutral
- Partial coverage of device functionality
- Different abstraction from native model and CLI

IOS XR – Configuration Model Approaches



BGP Neighbor Group Configuration – Schema Native Model (Deprecated) 1/2

```
router bgp 65001
 neighbor-group IBGP
  remote-as 65001
  update-source Loopback0
  address-family ipv4 unicast
  !
 !
!
```

```
<bgp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-bgp-cfg">
  <instance>
    <instance-name>default</instance-name>
    <instance-as>
      <as>0</as>
      <four-byte-as>
        <as>65001</as>
        <bgp-running/>
        <default-vrf>
          <bgp-entity>
            <neighbor-groups>
              <neighbor-group>
                <neighbor-group-name>IBGP</neighbor-group-name>
                <create/>
                <remote-as>
                  <as-xx>0</as-xx>
                  <as-yy>65001</as-yy>
                </remote-as>
                <!-- continued -->
```

BGP Neighbor Group Configuration – Schema Native Model (Deprecated) 2/2

```
router bgp 65001
 neighbor-group IBGP
  remote-as 65001
  update-source Loopback0
  address-family ipv4 unicast
  !
 !
!
```

```
                <!-- continued -->
                <update-source-interface>Loopback0</update-source-interface>
                <neighbor-group-afs>
                  <neighbor-group-af>
                    <af-name>ipv4-unicast</af-name>
                    <activate/>
                  </neighbor-group-af>
                </neighbor-group-afs>
              </neighbor-group>
            </neighbor-groups>
          </bgp-entity>
        </default-vrf>
      </four-byte-as>
    </instance-as>
  </instance>
</bgp>
```

BGP Neighbor Group Configuration – Unified Native Model

```
router bgp 65001
 neighbor-group IBGP
  remote-as 65001
  update-source Loopback0
  address-family ipv4 unicast
  !
 !
!
```

```
<router xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-router-bgp-cfg">
  <bgp>
    <as>
      <as-number>65001</as-number>
      <neighbor-groups>
        <neighbor-group>
          <neighbor-group-name>IBGP</neighbor-group-name>
          <remote-as>65001</remote-as>
          <update-source>Loopback0</update-source>
          <address-families>
            <address-family>
              <af-name>ipv4-unicast</af-name>
            </address-family>
          </address-families>
        </neighbor-group>
      </neighbor-groups>
    </as>
  </bgp>
</router>
```

Unified Configuration Models

7.0.1	7.1.1	7.2.1 (Deploy)	7.3.1 (Deploy)	7.4.1/7.5.1 (EC)
Interfaces Bundles ARP LACP VRF Static routing RIB MPLS (LDP, LSD, L3VPN) Telemetry NETCONF gRPC SNMP	BGP ISIS OSPF (v2/v3) MPLS (TE) RSVP	QoS ACL (IPv4, IPv6, Ethernet, prefix list, object group) Multicast (AMT, IGMP, MLD, MSDP, PIM)	Around 40 new models under testing	Over 200 models under development

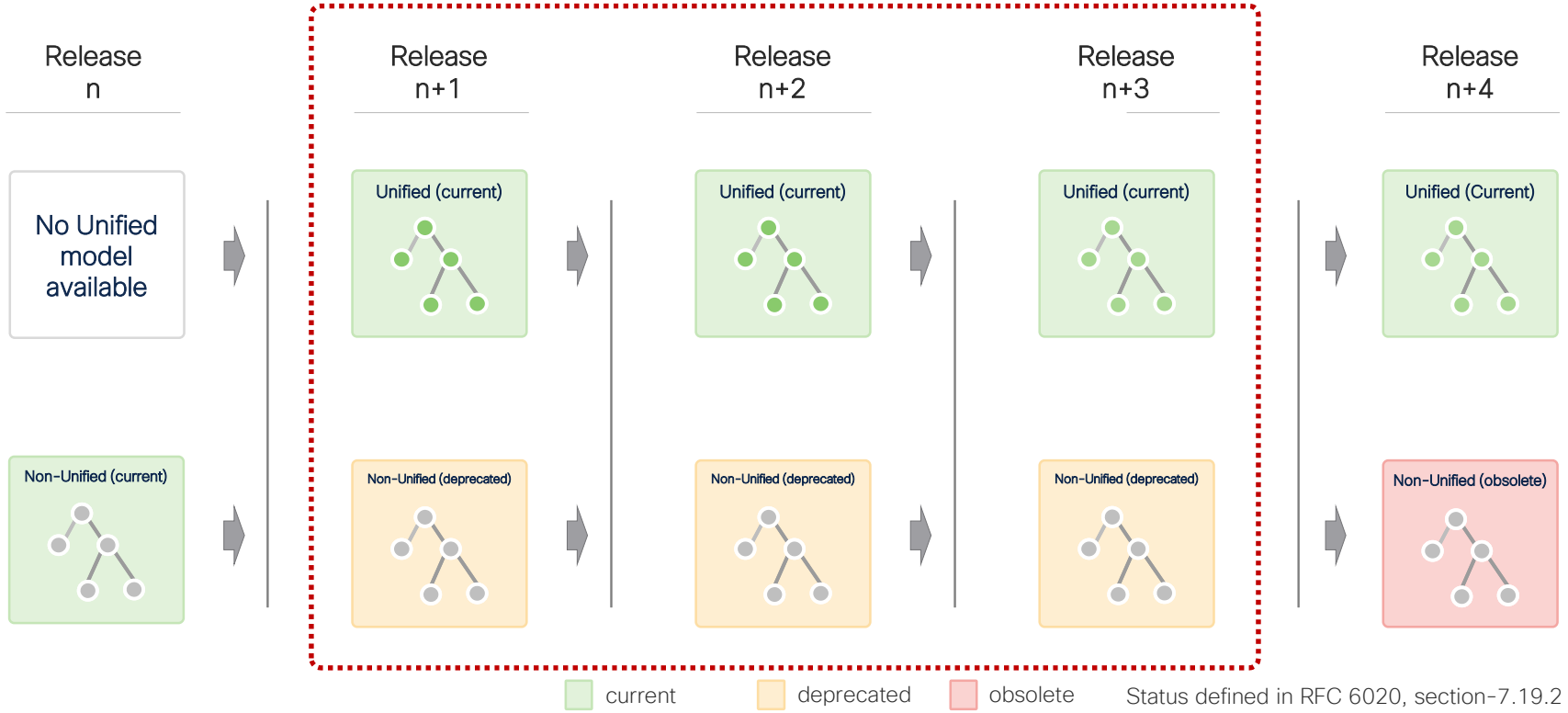
IETF RFC 6020 – Understanding Yang Model Status

- RFC6020 – YANG – A Data Modeling Language
  - Describes 3 states for Yang Models: Current, Obsolete, Deprecate

- Reproduced from RFC 6020, Section 7.19.2
  - "current" means that the definition is current and valid.
  - "deprecated" indicates an obsolete definition, but it permits new/ continued implementation in order to foster interoperability with older/existing implementations.
  - "obsolete" means the definition is obsolete and SHOULD NOT be implemented and/or can be removed from implementations.

https://tools.ietf.org/html/rfc6020#section-7.19.2

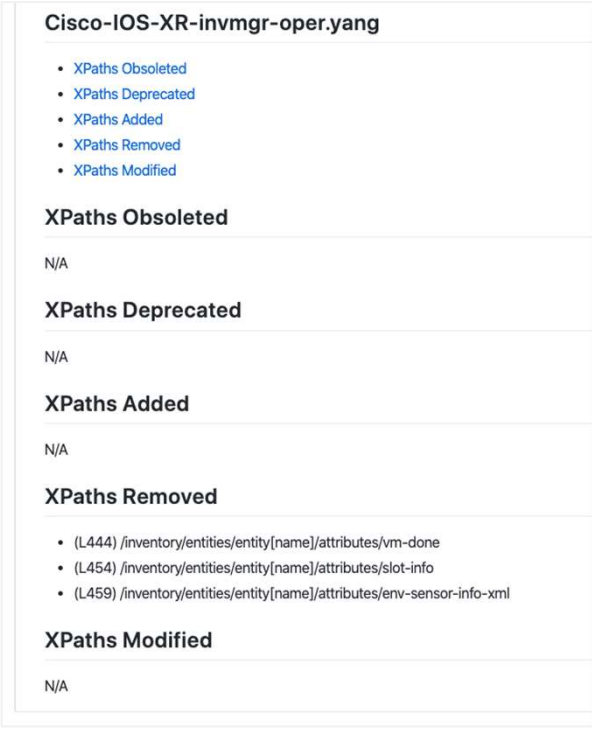
Status Evolution For Non-Unified Models



XR 7.0.2 / 7.1.1

IOS XR Yang Model Documentation

- Backwards incompatible changes are documented on GitHub
  - https://github.com/YangModels/yang/tree/master/vendor/cisco/xr/731/BIC
  - Definitions based on RFC6020, Section 10
- Format
  - HTML
  - JSON (available)
- Full list of Models available in per XR release
  - https://github.com/YangModels/yang/blob/master/vendor/cisco/xr/731/Available-Content.md



Cisco-IOS-XR-invmgr-oper.yang

- XPaths Obsoleted
- XPaths Deprecated
- XPaths Added
- XPaths Removed
- XPaths Modified

XPaths Obsoleted

N/A

XPaths Deprecated

N/A

XPaths Added

N/A

XPaths Removed

- (L444) /inventory/entities/entity[name]/attributes/vm-done
- (L454) /inventory/entities/entity[name]/attributes/slot-info
- (L459) /inventory/entities/entity[name]/attributes/env-sensor-info-xml

XPaths Modified

N/A

EC XR 7.4.1

Equivalency Tool (CLI to Yang)

- XR Feature that highlights Equivalent Yang Paths for Show & Config commands

- CLI snapshot (Early view)

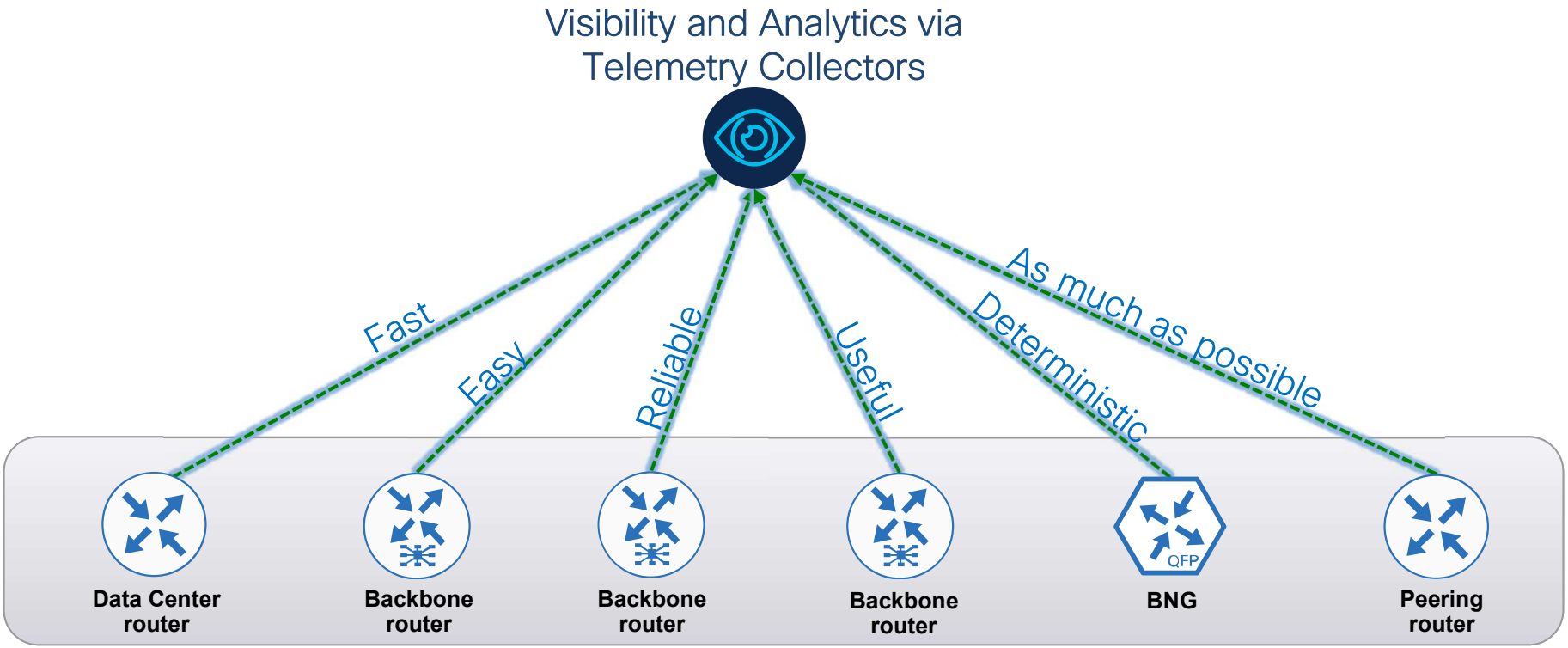
```
RP/0/RP0/CPU0:ios#model-describe operational line show platform location 0/RP0/CPU0
Wed Jan 20 02:38:55.419 UTC
Operation               : Get
Yang Path               : Cisco-IOS-XR-platform-oper:platform/racks/rack/slots/slot/instances/instance/state
XML Schema Hierarchy    : Platform.RackTable.Rack.SlotTable.Slot.InstanceTable.Instance.State

RP/0/RP0/CPU0:ios#model-describe operational line show inventory
Wed Jan 20 02:39:03.528 UTC
Operation               : Datalist
Yang Path               : Cisco-IOS-XR-invmgr-oper:inventory/racks/rack/attributes/inv-basic-bag
XML Schema Hierarchy    : None
```

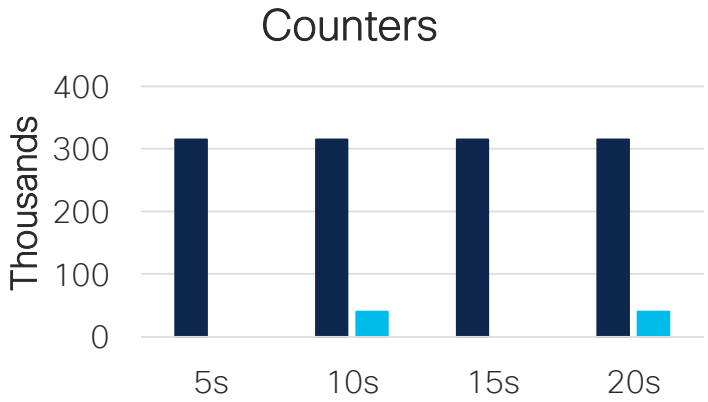
XR Programmability Infrastructure

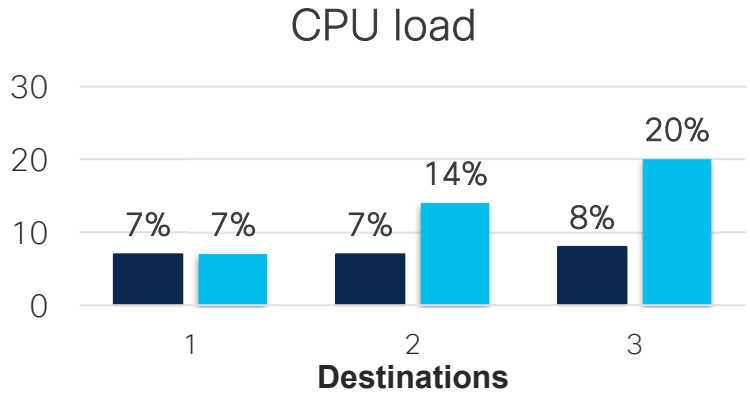
Telemetry – “Monitor”

Model Driven Telemetry – "Consumption Driven"

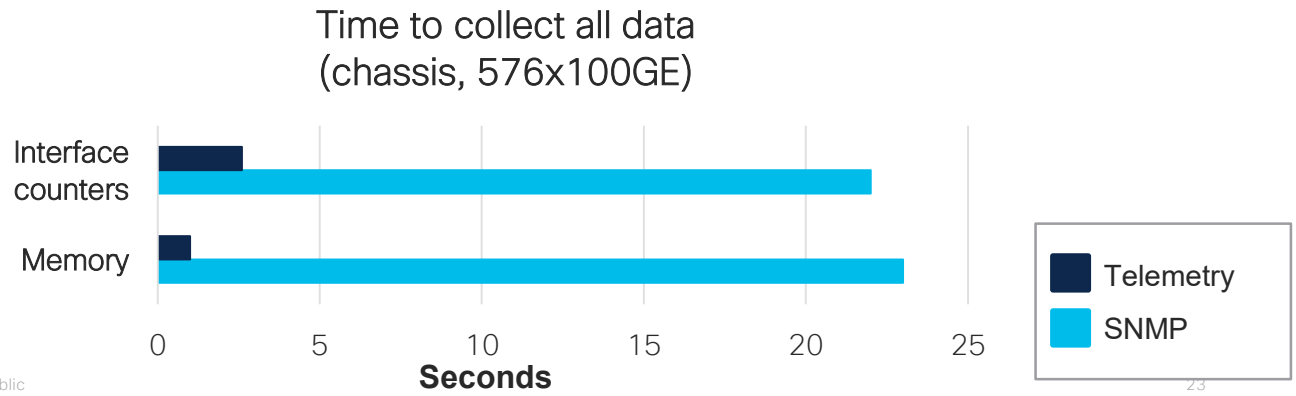


Telemetry vis-à-vis SNMP – “No Contest”





- ✓ More counter data
- ✓ Reduction in CPU load
- ✓ Faster collection



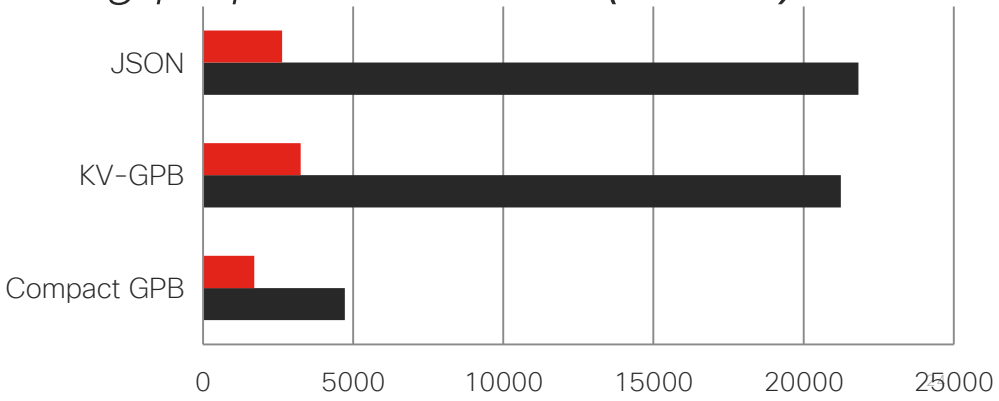
gRPC compression

Deploy

XR 7.1.2 / 7.2.1

- Support for compression has been added to XR gRPC implementation
- No configuration required for gNMI clients
  - Clients use CallOption “UseCompressor”
- New configuration under *protocol grpc per destination* ***(dialout)***

```
telemetry model-driven
 destination-group notls
  address-family ipv4 192.168.122.1 port 9902
   encoding self-describing-gpb
   protocol grpc no-tls gzip
  !
 !
!
```



XR 7.2.1

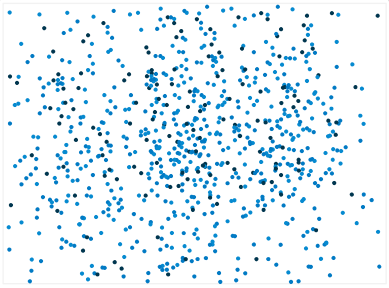
Leaf-level filtering

- Current subscriptions are internally mapped to the corresponding container (gather path)
- New feature to allow subscription at individual leaf level
  - Multiple leaves can be specified in a single subscription
  - Optimized to avoid duplicate internal collections

Deploy XR 7.3.1

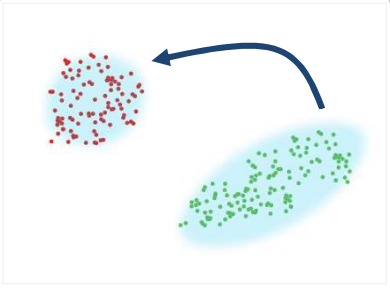
AI Driven Telemetry (ADT)

Collect



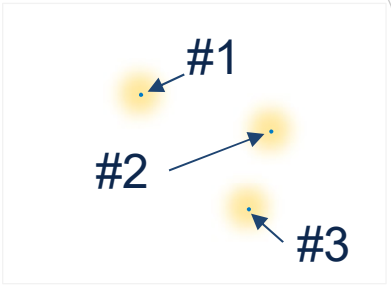
Holistic view:
Collect all counters all the time.
Currently: MDT data, Netflow/CRFT (future)

Detect



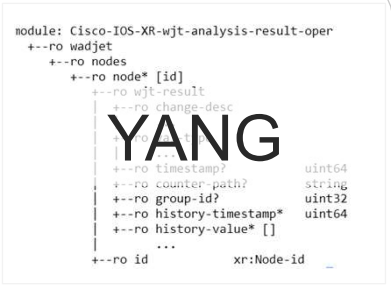
Macroscopic view:
Catch interesting state changes.
Dim.-Redux, Cluster.
Online, unsupervised.

Select



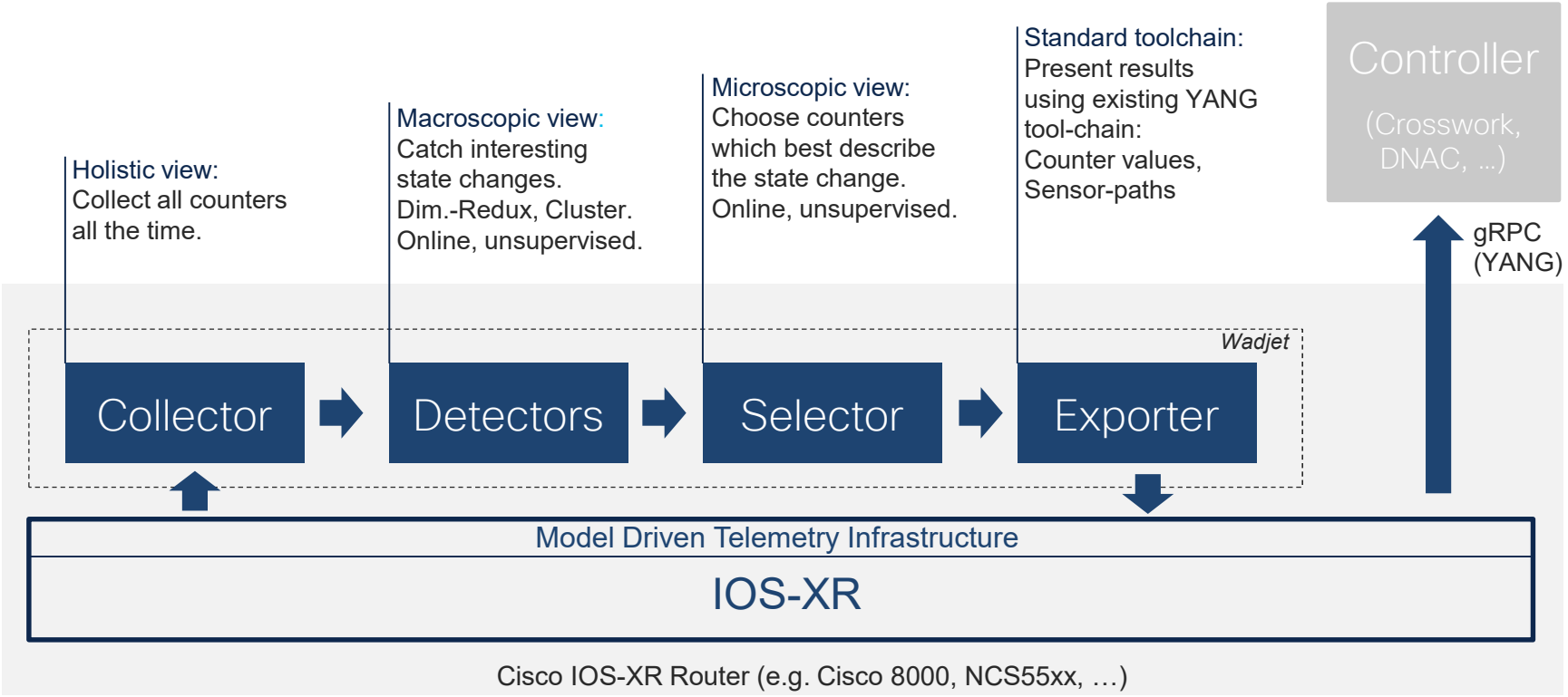
Microscopic view:
Choose counters which best describe the state change.
Online, unsupervised.

Export



Present results using existing YANG tool-chain:
Counter values, Sensor-paths

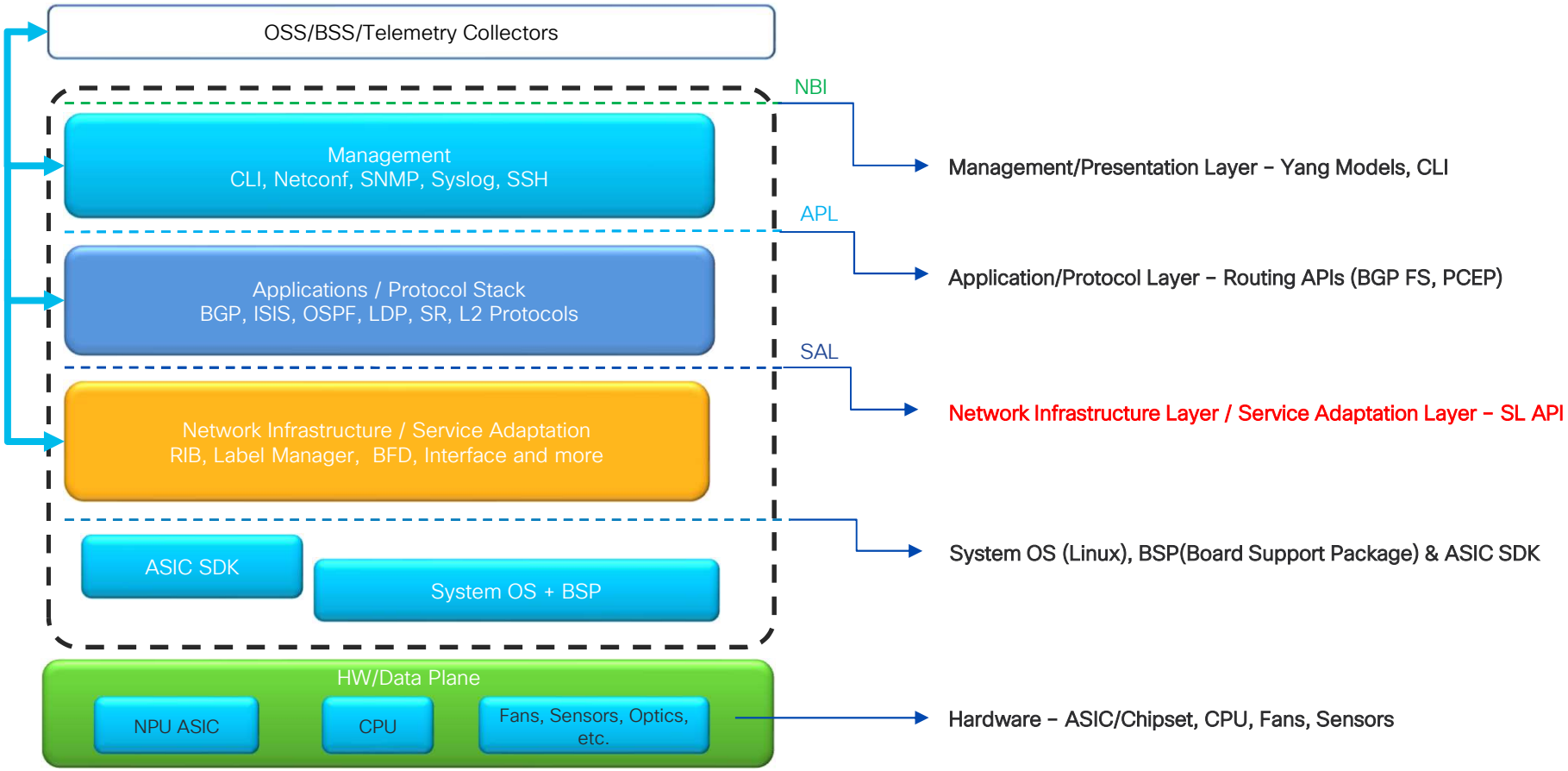
ADT in IOS-XR: Architecture



XR Programmability Infrastructure

Service Level API (SL-API) for “Control”

API Layers in IOS XR -> “Control” with SL-API



Examples of Using Service Level API (SL-API)

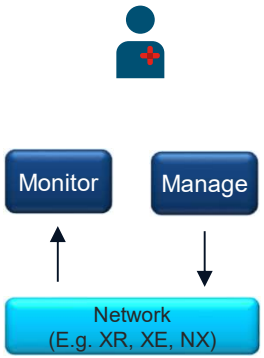
Use Case	Github Code Location
Open/R running on IOS-XR as an IGP	https://github.com/akshshar/openr-xr
Programmable BGP Route Download	https://github.com/Cisco-Service-Layer/openbmp-controller
Egress Traffic Controller Telemetry based route selection	https://github.com/Maikor/nanog71-hackathon
IPv6 neighbor based path failover (Telemetry+SL-API)	https://github.com/akshshar/xrtelemetry-slapi
Interface Events based path failover (SL-API + YDK)	https://github.com/akshshar/ydk-slapi-remediation

Automation Apps

Cisco Software – Automation Gradient

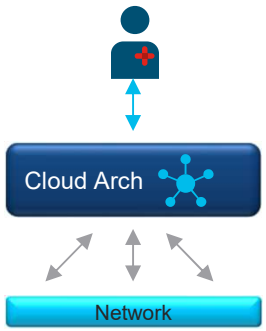
Analogy = SAE Level 1-5 for Self Driving cars

"Level 1" = NMS 2005



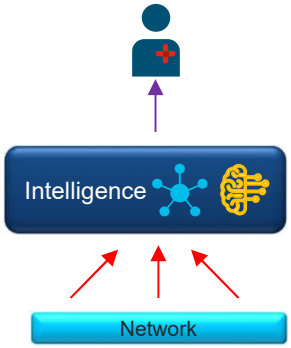
- No APIs
- Distinct Apps
- Fixed GUI

"Level 2" = Evolve Device to Network + APIs



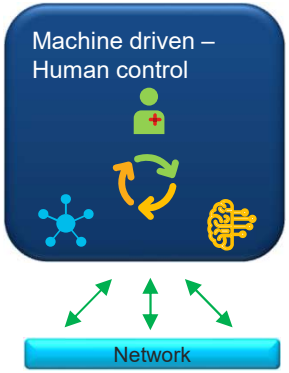
- Open APIs
- Service Models
- Flexible GUI

"Level 3" = Profile with ML/MR Algos



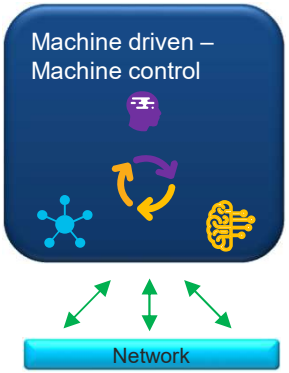
- Use ML algos for Trends
- Use MR expert systems for Troubleshooting

"Level 4" = Closed Loop (With Intervention)



- Use ML + MR Algos with ability to control Network
- Human Control Intact

"Level 5" = Closed Loop (No Intervention)



- Final End State – Decommission Elements when unable to handle
- Human intervention for 'out of sample' scenarios

Cisco's Service Provider Automation Offerings

Day -1: Planning

All Planning Functions prior to Network

Product(s) Available:
- Crosswork Qualification Engine (CQE)

Day 0: Bring Up

Bring Up Functions Including ZTP & Software Image Management (SWIM)

Product(s) Available:
- XR ZTP
- Crosswork ZTP

Day 1: Operate

Network Services & Transport Operations

Product(s) Available:
- XR Yang Models & Telemetry

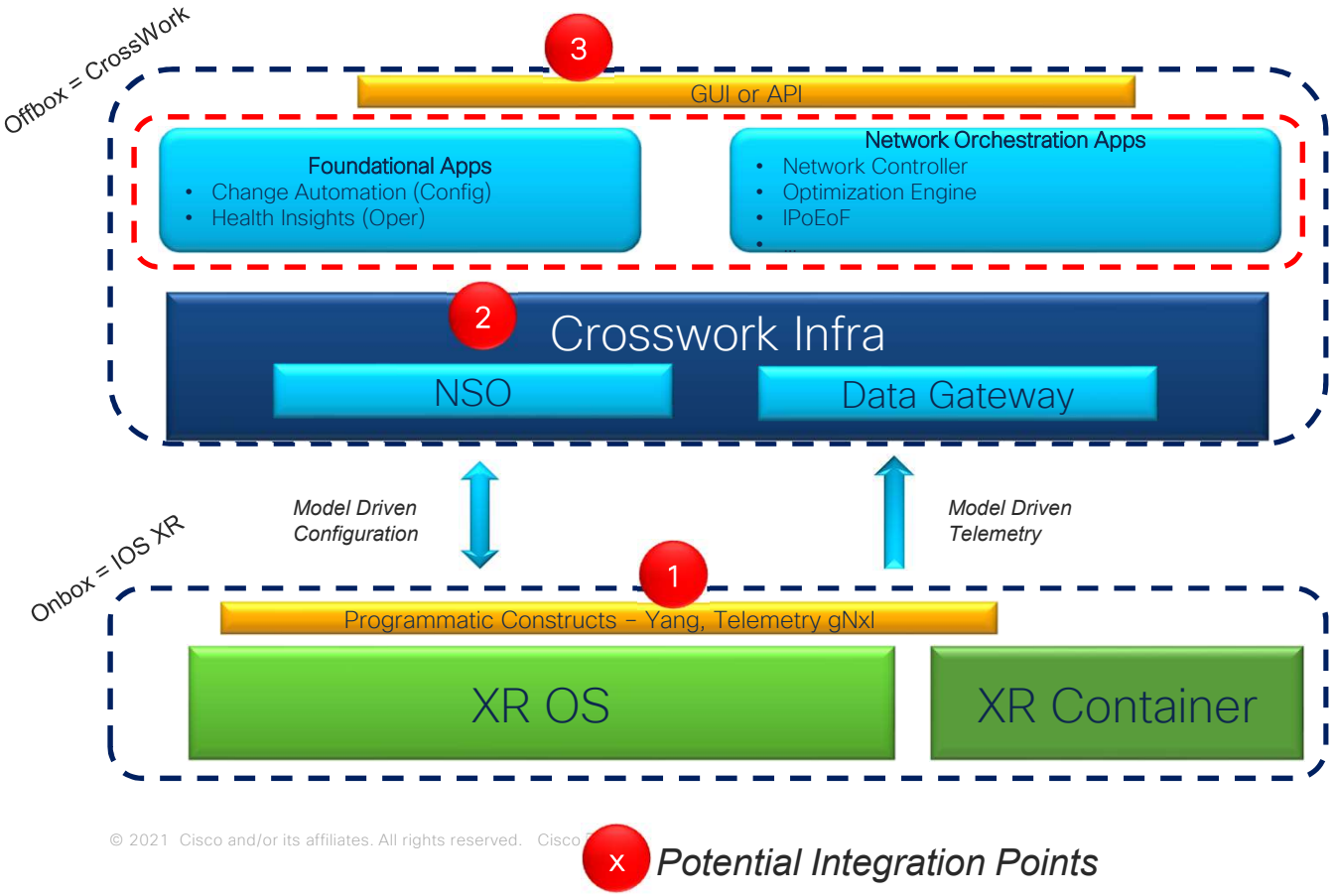
- Change Automation
- Health Insights
- Network Controller

Day 2: Assurance

Protocol level Troubleshooting

Product(s) Available:
- Day 1 Offerings

Cisco SP – Full Stack Software Offerings



- Rich Application Ecosystem
- Consume (or extend) via GUI or API
- App Ecosystem leverage APIs internally
- Built for Cloud scale with Microservices architecture

- 'Onbox' Software within Router – the Base Layer
- Fully Open Yang Models – Extend for Provider Software
- Container provides pre-processing capabilities for Crosswork

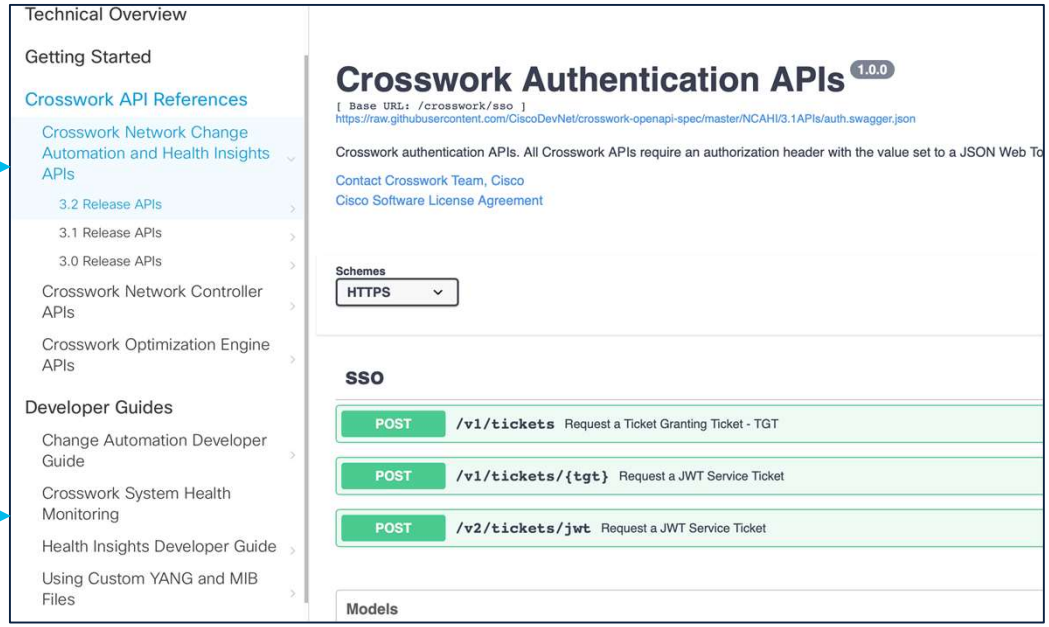
x *Potential Integration Points*

Crosswork APIs

https://developer.cisco.com/crosswork/

API Docs

Developer Guides



Detailed API with Swagger based Documentation

Takeaways

Takeaways

- Operational Simplification in XR delivered via Data Models
- XR Programmability Infrastructure Hardening
- Wide range of deployment styles envisioned – DIY, Full Stack & Hybrid


CISCO