



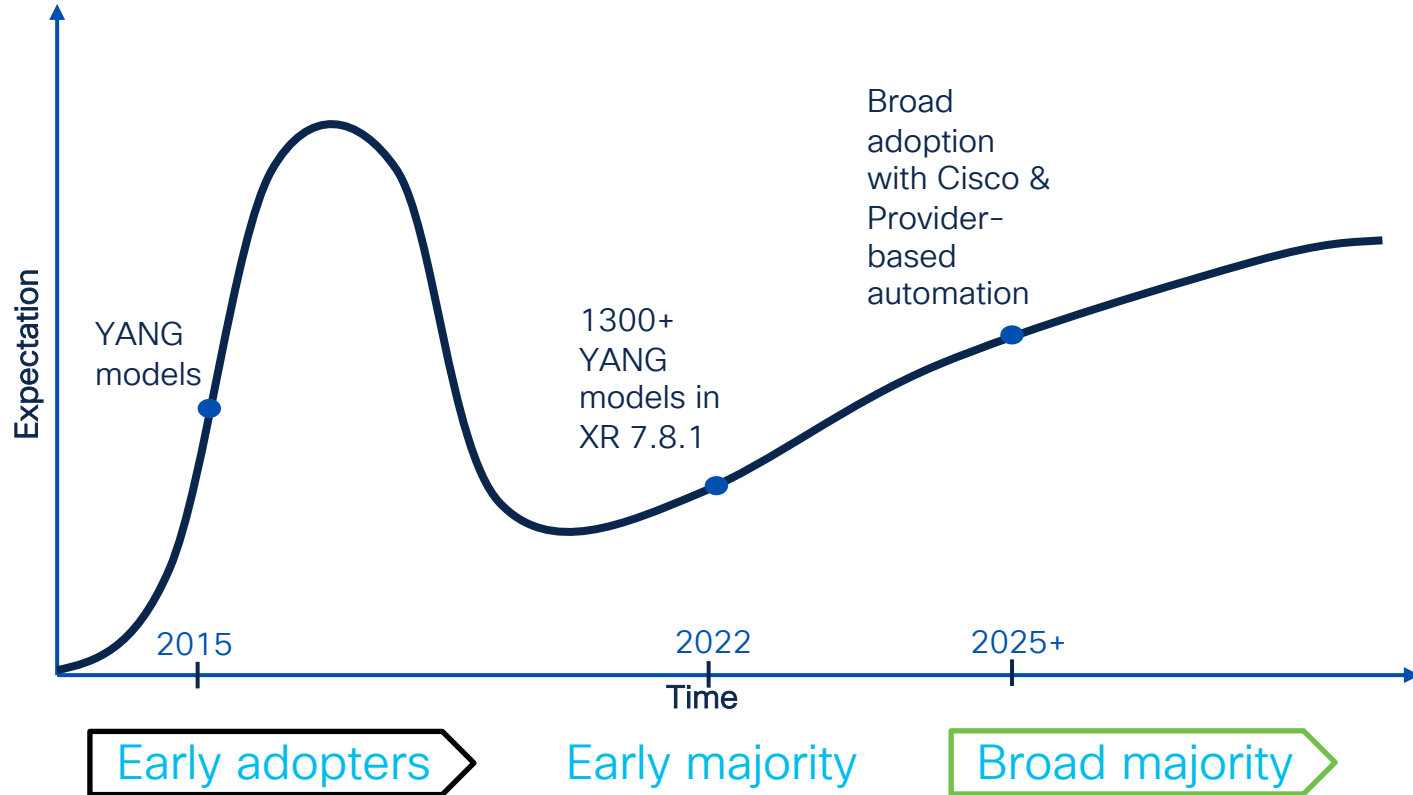
The bridge to possible

# Practical Approach to Programmability

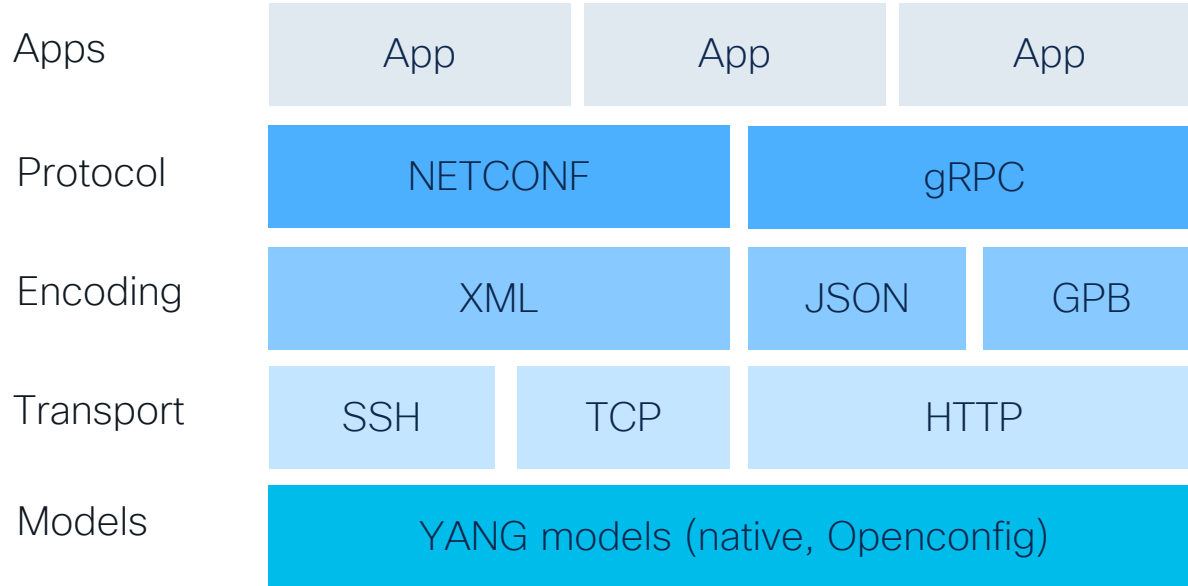
Mike Korshunov, TME @ Provider Connectivity Group

March 29, 2023

# XR Model-Driven Programmability Evolution



# Integration Layers



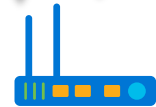
Controller/Orchestrator



Model-driven configuration

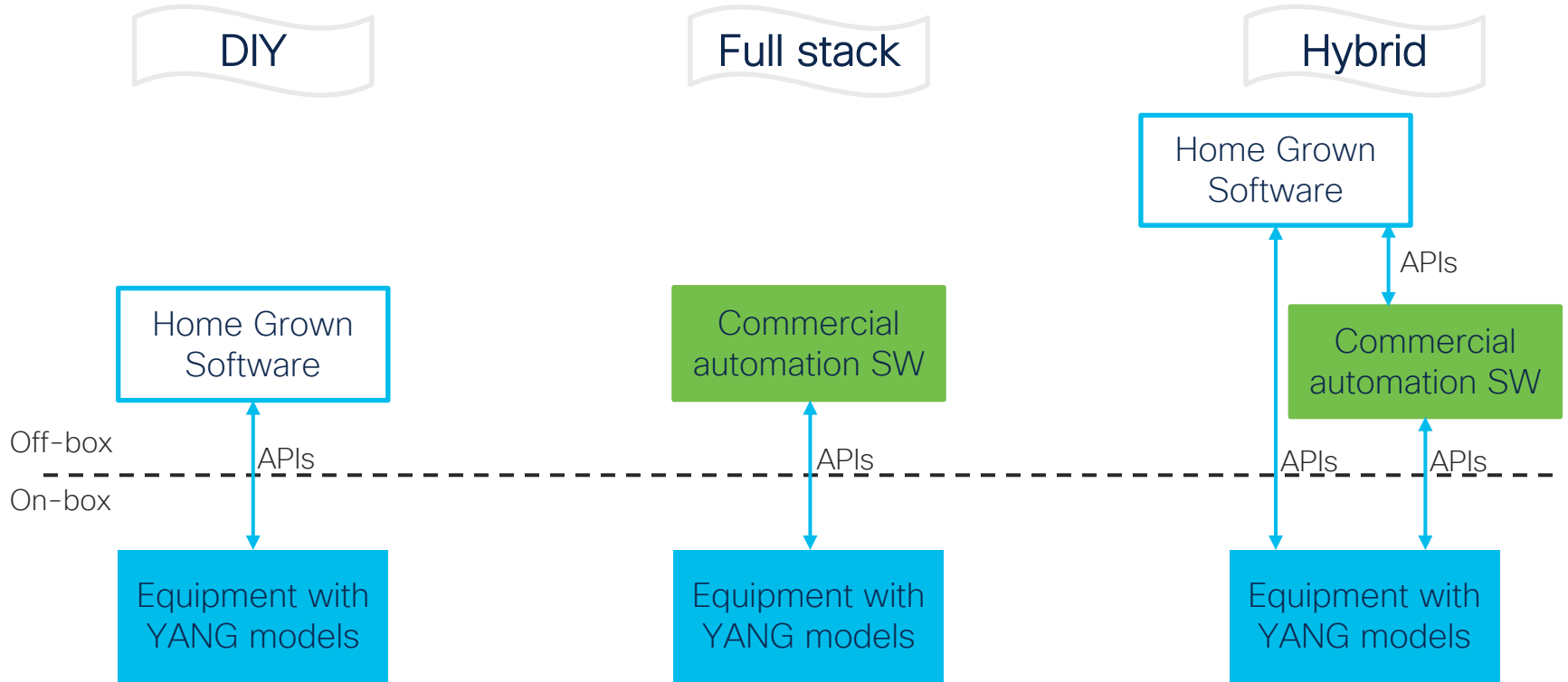
Closed-loop automation

Model-driven telemetry



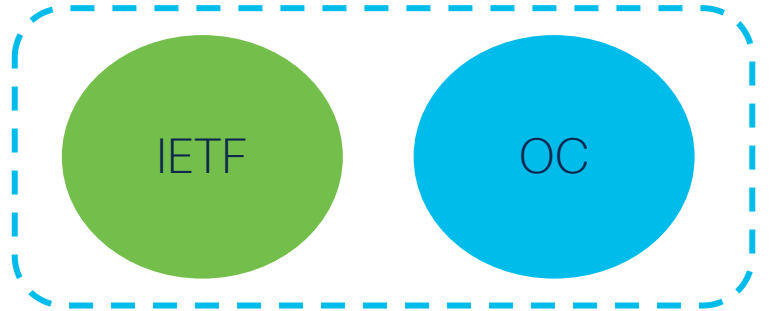
Network device

# Customer Deployment Styles



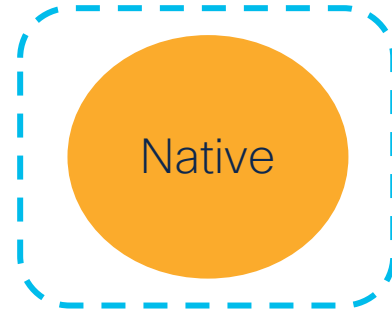
# Data Models and YANG

- Data model explicitly and precisely determines the structure, syntax, semantics of the data which is externally available and visible;
- Ensures completeness and consistency of interactions between systems and clients.



ietf-ospf.yang

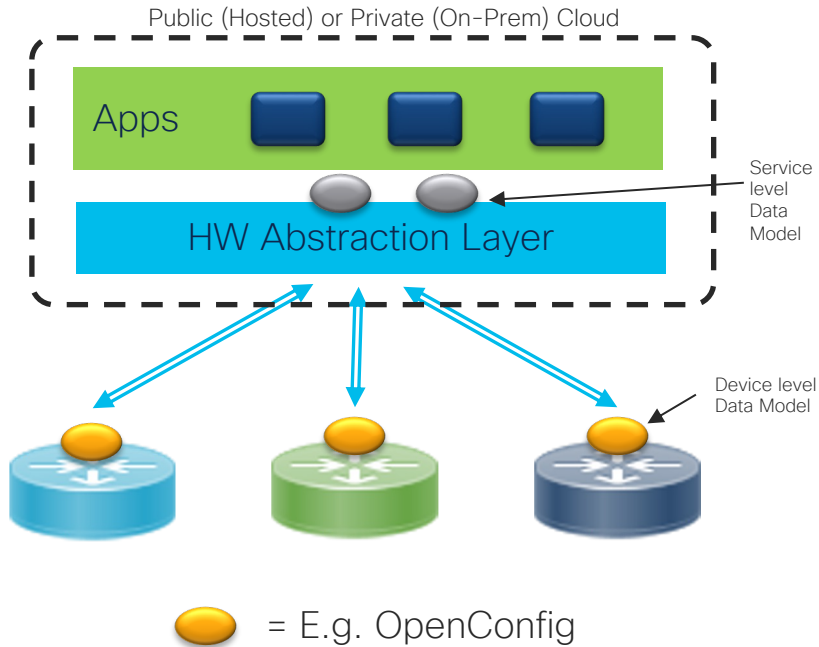
openconfig-interfaces.yang



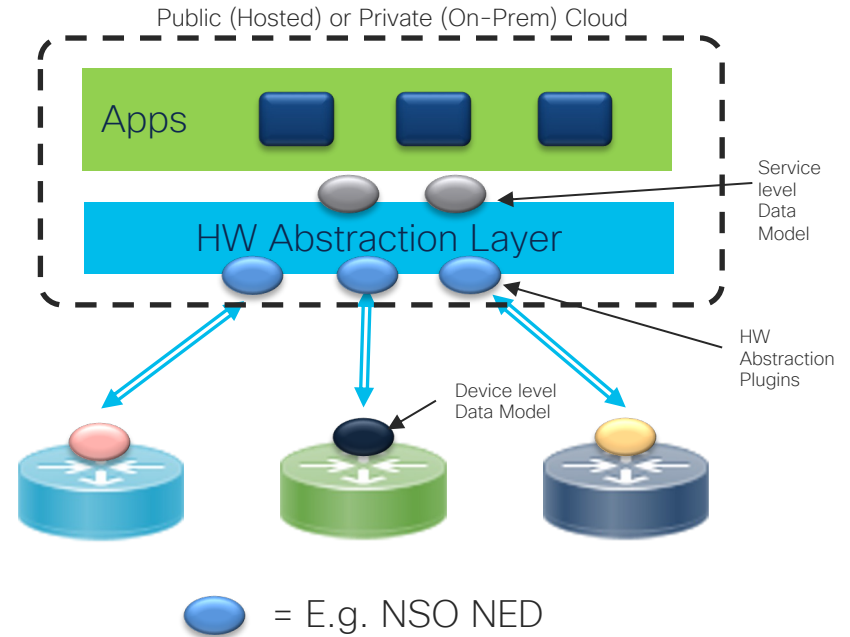
Cisco-IOS-XR-ipv4-ospf-oper.yang

# Vendor Neutrality – Options for Operators

*Device Level Vendor Neutrality*



*Network Level Vendor Neutrality*



# How to find the data you want to stream?

All the information is formatted using YANG models. Don't stream everything, find what you need inside the model and configure the full path for your telemetry. Use [pyang](#) for this, it shows any model in a nice way.

```
$ pyang -f tree Cisco-IOS-XR-infra-statsd-oper.yang --tree-path infra-statistics/interfaces/interface/latest/generic-counters
```

```
module: Cisco-IOS-XR-infra-statsd-oper
  +--ro infra-statistics
    +--ro interfaces
      +--ro interface* [interface-name]
        +--ro latest
          +--ro generic-counters
            +--ro packets-received?      uint64
            +--ro bytes-received?       uint64
            +--ro packets-sent?         uint64
            +--ro bytes-sent?           uint64
            +--ro multicast-packets-received?  uint64
            +--ro broadcast-packets-received?  uint64
```

*<output snipped for brevity>*

You can start using pyang as is, without “--tree-path” to see all paths. This one just gives more exact view

You can see what is inside the path (this info will be streamed)

# Cisco YANG Suite – explore models with Ease

The screenshot displays the Cisco YANG Suite web interface. At the top, the breadcrumb navigation reads 'YANG Suite / Exploring YANG / YANG set "IOS XE 16.7.1" / Modules'. The main header is 'Explore YANG Models'. Below this, there are search and filter options: 'Select a YANG set' (IOS XE 16.7.1), 'Select YANG module(s)' (openconfig-interfaces), and a 'Load module(s)' button. There are also search boxes for 'Icon legend', 'Search XPath', and 'Search nodes', and a 'Display' toggle set to 'Schema nodes only'.

The left sidebar shows a tree view of the 'openconfig-interfaces' module, with 'interface' selected. The tree includes nodes for 'name', 'config', 'state', 'hold-time', 'subinterfaces', 'oc-eth:ethernet', 'oc-vlan:routed-vlan', and 'oc-lag:aggregation'.

The main content area is titled 'Node Properties' and displays the following information for the 'interface' node:

- Module:** openconfig-interfaces
- Revision:** 2016-12-22
- Revision Info:** Fixes to Ethernet interfaces model
- Description:** Model for managing network interfaces and subinterfaces. This module also defines convenience types / groupings for other models to create references to interfaces:
  - base-interface-ref (type) - reference to a base interface
  - interface-ref (grouping) - container for reference to a interface + subinterface
  - interface-ref-state (grouping) - container for read-only (opstate) reference to interface + subinterface
- Organization:** OpenConfig working group
- Imports:**
  - "ietf-interfaces"
  - "ietf-yang-types"
  - "openconfig-extensions"
- Namespace:** http://openconfig.net/yang/interfaces
- Prefix:** oc-if
- Namespace Prefixes:**

cisco-if-eth	http://cisco.com/ns/yang/cisco-xe-openconfig-if-ethernet-ext
ianaif	urn:ietf:params:xml:ns:yang:iana-if-type
ietf-if	urn:ietf:params:xml:ns:yang:ietf-interfaces
if	urn:ietf:params:xml:ns:yang:ietf-interfaces



# YANG Models Documentation - Github

- List of models per XR release:  
<https://github.com/YangModels/yang/blob/main/vendor/cisco/xr/761/Available-Content.md>
- Backwards-incompatible changes based on RFC 6020, Section 10 (since 7.0.2):  
<https://github.com/YangModels/yang/tree/main/vendor/cisco/xr/761/BIC>
- Check backwards-incompatibility  

```
$ ./check-models.sh -b 751 # Check incompatibility between 7.6.1 and 7.5.1
```

🔗 Cisco-IOS-XR-ipv4-bgp-oper.yang
<ul style="list-style-type: none"><li>• XPathS Obsoleted</li><li>• XPathS Deprecated</li><li>• XPathS Added</li><li>• XPathS Removed</li><li>• XPathS Modified</li></ul>
<b>XPathS Obsoleted</b>
N/A
<b>XPathS Deprecated</b>
N/A
<b>XPathS Added</b>
N/A
<b>XPathS Removed</b>
N/A
<b>XPathS Modified</b>
<ul style="list-style-type: none"><li>• (L10290) {BGP-NBR-BAG}/af-data[af-name]</li></ul>

XR 7.6.1

Monitor with Telemetry

# Is It Enough To State gRPC Support?

## Cisco gRPC call proto

```
service gRPCConfigOper {  
  // Configuration related commands  
  rpc GetConfig(ConfigGetArgs) returns(stream ConfigGetReply) {};  
  rpc MergeConfig(ConfigArgs) returns(ConfigReply) {};  
  rpc DeleteConfig(ConfigArgs) returns(ConfigReply) {};  
  rpc ReplaceConfig(ConfigArgs) returns(ConfigReply) {};  
  rpc CliConfig(CliConfigArgs) returns(CliConfigReply) {};  
  rpc CommitReplace(CommitReplaceArgs)  
    returns (CommitReplaceReply) {};  
  // Do we need implicit or explicit commit  
  rpc CommitConfig(CommitArgs) returns(CommitReply) {};  
  rpc ConfigDiscardChanges(DiscardChangesArgs)  
    returns(DiscardChangesReply) {};  
  // Get only returns oper data  
  rpc GetOper(GetOperArgs) returns(stream GetOperReply) {};  
  // Get Telemetry Data  
  rpc CreateSubs(CreateSubsArgs) returns(stream CreateSubsReply)  
};  
}
```

[https://github.com/cisco/bigmuddy-network-telemetry-  
proto/blob/master/staging/mdt\\_grpc\\_dialin/mdt\\_grpc\\_dialin.proto](https://github.com/cisco/bigmuddy-network-telemetry-<br/>proto/blob/master/staging/mdt_grpc_dialin/mdt_grpc_dialin.proto)

## Juniper gRPC call proto

```
service OpenConfigTelemetry {  
  // Request an inline subscription for data at the specified path.  
  // The device should send telemetry data back on the same  
  // connection as the subscription request.  
  rpc telemetrySubscribe(SubscriptionRequest)  
    returns (stream OpenConfigData) {}  
  // Terminates and removes an existing telemetry subscription  
  rpc cancelTelemetrySubscription(CancelSubscriptionRequest)  
    returns (CancelSubscriptionReply) {}  
  // Get the list of current telemetry subscriptions from the  
  // target. This command returns a list of existing subscriptions  
  // not including those that are established via configuration.  
  rpc getTelemetrySubscriptions(GetSubscriptionsRequest)  
    returns (GetSubscriptionsReply) {}  
  // Get Telemetry Agent Operational States  
  rpc getTelemetryOperationalState(GetOperationalStateRequest)  
    returns (GetOperationalStateReply) {}  
  // Return the set of data encodings supported by the device for telemetry  
  rpc getDataEncodings(DataEncodingRequest)  
    returns (DataEncodingReply) {}  
}
```

[https://github.com/Juniper/itimom/blob/master/t  
elemetry/telemetry.proto](https://github.com/Juniper/itimom/blob/master/t<br/>elemetry/telemetry.proto)

# gNMI RPCs

- **Capabilities** - Initial handshake to exchange capability info (e.g. supported data models)
- **Set** - Modifies data from server (network device)
- **Get** - Retrieves data on server (network device)
- **Subscribe** - Control data subscriptions on server (network device)

<https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md>

# gNMI support across Cisco Products

Protocol	IOS XR	IOS XE	NX OS
gNMI	6.5.1	16.12*	9.3(x)
NETCONF	4.1	16.6*	7.x

\* Feature availability is platform dependent

# Telemetry – Best Practices

- Telemetry requires Collector based architectures
  - Limited processing of data on the router due to limited compute
  - Generic Server compute with Data Lake type approach
- Key factors for scaling Telemetry
  - Cadence – Interval between Sensor path updates
  - Interfaces/Sensor paths – Amount of data to be streamed out of each device
  - Devices to Collector Ratio
    - Compute or Bandwidth should not be constrained
    - Distributed Collectors across the network
- Deployments case:
  - In production: around 30 Devices per Collector with aggregate of 1000 interfaces with 30 second cadence
  - In discussions: Varied Devices/Collector ratio at 1-10 min cadence



The bridge to possible