Programmable Network Overview

Model-driven programmability for network devices
Replace CLI/SNMP with structured, automated config
YANG models define data structure (RFC 6020/7950)
NETCONF — XML over SSH (RFC 6241)
RESTCONF — REST/HTTP(S) over JSON or XML (RFC 8040)
gRPC/gNMI — Google protocol, streaming telemetry
Controller ↔ Device via standardised data models
Enables Infrastructure as Code (IaC) workflows
Cisco IOS-XE, NX-OS, IOS-XR all support NETCONF

NETCONF — Key Facts

Standard	RFC 6241 (NETCONF v1.1)
Transport	SSH — TCP port 830
Encoding	XML only
Session type	Stateful — persistent SSH session
Operations	get, get-config, edit-config, copy-config…
Datastores	running, candidate, startup
Notifications	NETCONF event notifications (RFC 5277)
Capabilities	Exchanged in <hello> at session start
YANG	Data modelling language for NETCONF/RESTCONF
Namespaces	urn:ietf:params:xml:ns:netconf:base:1.0

NETCONF Operations

<get>	Retrieve running config + state data
<get-config>	Retrieve config from specific datastore
<edit-config>	Modify config — merge/replace/delete
<copy-config>	Copy one datastore to another
<delete-config>	Delete a datastore (not running)
<lock>	Lock datastore — prevent concurrent changes
<unlock>	Release datastore lock
<commit>	Apply candidate to running
<close-session>	Gracefully close NETCONF session
<kill-session>	Force-terminate another session

RESTCONF — Key Facts

Standard	RFC 8040
Transport	HTTPS — TCP port 443 (HTTP 80 possible)
Encoding	JSON or XML (Accept/Content-Type headers)
Session type	Stateless — standard REST HTTP
Base path	/restconf (root discovery via /.well-known)
Data path	/restconf/data/<yang-module>:<container>
Ops path	/restconf/operations/<rpc-name>
Auth	HTTP Basic Auth or token-based
Methods	GET POST PUT PATCH DELETE HEAD OPTIONS
Content-Type	application/yang-data+json or +xml

RESTCONF HTTP Methods

GET	Retrieve resource — read-only — idempotent
POST	Create new resource or invoke an operation (RPC)
PUT	Create or replace entire resource
PATCH	Partial update — merge changes into resource
DELETE	Remove resource from configuration
HEAD	Same as GET but no response body
OPTIONS	Retrieve supported methods for resource

RESTCONF HTTP Status Codes

200 OK	Success — response body included
201 Created	Resource successfully created (POST/PUT)
204 No Content	Success — no response body (DELETE/PATCH)
400 Bad Request	Malformed request or invalid data
401 Unauthorized	Authentication failure
404 Not Found	Resource or path does not exist
405 Not Allowed	HTTP method not supported on resource
409 Conflict	Resource already exists (POST duplicate)

NETCONF vs RESTCONF

Feature	NETCONF	RESTCONF
RFC	RFC 6241	RFC 8040
Transport	SSH (port 830)	HTTPS (port 443)
Encoding	XML only	JSON or XML
Session	Stateful	Stateless REST
Datastores	run/candidate/startup	running only
Transactions	candidate + commit	No transactions
Use case	Full config mgmt	REST API / scripts

YANG Data Model — Key Constructs

Purpose	Data modelling language — RFC 6020 / 7950
Describes	Config data, state data, RPCs, notifications
module	Top-level container for YANG model
container	Groups related nodes — no value itself
leaf	Single value node (string, int, bool…)
leaf-list	Multiple values of same type
list	Multiple instances (like a table row)
key	Unique identifier for list entry
choice/case	Mutually exclusive config options
augment	Extend another YANG model
deviation	Vendor-specific model modification
typedef	Define reusable type
grouping/uses	Reusable set of nodes
rpc	Define remote procedure calls
notification	Define async event notifications

Platform Support & Tools

Cisco IOS-XE	netconf-yang / restconf (enable in config)
Cisco NX-OS	feature netconf / feature restconf
Cisco IOS-XR	netconf agent tty (native support)
ncclient	Python library for NETCONF (pip install)
requests	Python HTTP library for RESTCONF
Postman	REST API testing GUI tool
Yang Suite	Cisco YANG model browser and tester
pyang	YANG validator and converter tool

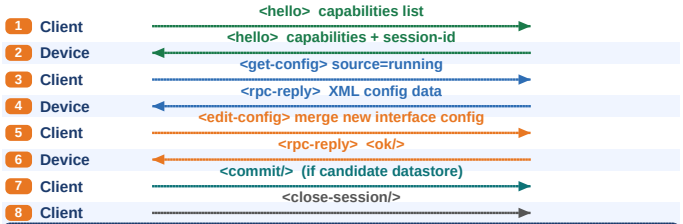
Enabling NETCONF & RESTCONF on Cisco IOS-XE

```
netconf-yang                ! enable NETCONF
restconf                    ! enable RESTCONF
ip http secure-server       ! required for RESTCONF
username admin privilege 15 secret <pass>
aaa new-model  +  local auth   ! authentication
```

NETCONF Architecture & Session Flow

NETCONF Protocol Stack (Manager ↔ Agent)

NETCONF Session Exchange (step by step)



NETCONF Datastores — running / candidate / startup

running: active config — always present.
candidate: staging area — edit then commit to running.
startup: config loaded at boot — persist with copy run start.
Use <lock> before editing to prevent race conditions.
candidate datastore requires :candidate capability.

RESTCONF Architecture & HTTP Operations

RESTCONF Architecture (stateless REST over HTTPS)

RESTCONF HTTP Methods → NETCONF Equivalent

GET	Read config or state	<get-config> / <get>
POST	Create new resource	<edit-config> op=create
PUT	Create or replace	<edit-config> op=replace
PATCH	Partial update / merge	<edit-config> op=merge
DELETE	Remove resource	<edit-config> op=delete

RESTCONF URL Anatomy

RESTCONF Content-Type Headers

Accept: application/yang-data+json ← request JSON response
Accept: application/yang-data+xml ← request XML response
Content-Type: application/yang-data+json ← sending JSON body
Discover root: GET /.well-known/host-meta → finds /restconf

YANG Model Structure & Data Encoding

YANG Module Tree Structure

```
module ietf-interfaces {
  container interfaces {
    list interface {
      key "name";
      leaf name { type string; }
      leaf enabled { type boolean; }
    }  }  }
```

Same YANG Data — XML (NETCONF) vs JSON (RESTCONF)

XML (NETCONF)

```
<interfaces xmlns=
 "ietf-interfaces">
 <interface>
  <name>Gi0/1</name>
  <enabled>true</enabled>
 </interface>
</interfaces>
```

JSON (RESTCONF)

```
{
 "ietf-interfaces:
  interfaces": {
  "interface": [{
   "name": "Gi0/1",
   "enabled": true
  }] } }
```

YANG Node Types

module	Top-level namespace container for a YANG model
container	Groups related nodes — no value, just structure
leaf	Single scalar value (string / int / bool / enum)
leaf-list	List of same-typed leaf values (like an array)
list	Sequence of entries each identified by a key
choice/case	Mutually exclusive configuration alternatives
rpc	Remote Procedure Call — action on the device
notification	Asynchronous event sent from device to client
grouping	Reusable block of nodes (referenced with uses)
augment	Add nodes to existing module (vendor extension)

Python ncclient — NETCONF get-config example

```
from ncclient import manager
m = manager.connect(host='10.0.0.1',
    port=830, username='admin',
    password='Cisco123', hostkey_verify=False)
cfg = m.get_config(source='running')
print(cfg)
```