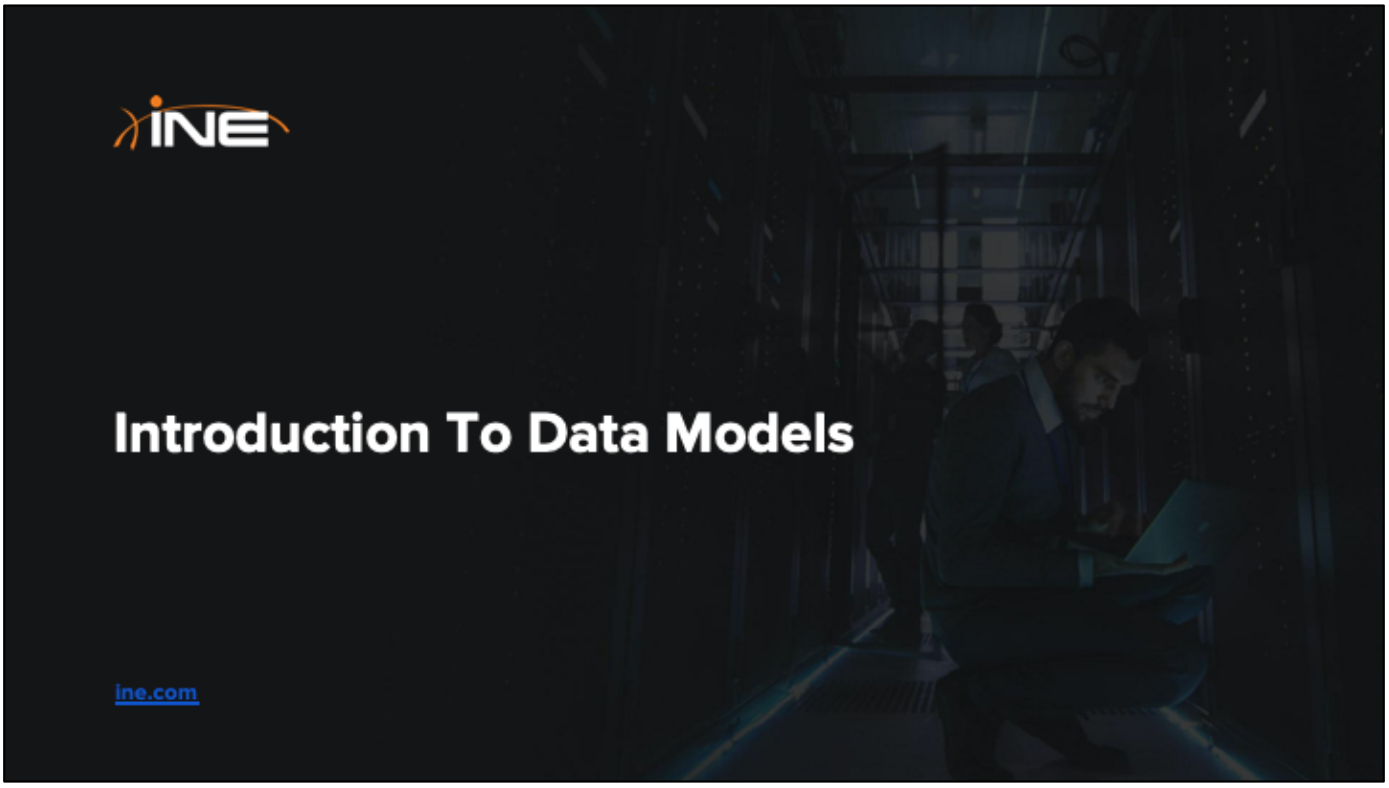
Course Objectives

- Introduce you to the concepts of Data Modeling as implemented by YANG
- Summarize YANG terminology & concepts
- Demonstrate how to view YANG modules
- Introduce NETCONF, why it was needed and what it does
- Introduce RESTCONF, why it was needed and what it does
- Demonstrate both NETCONF & RESTCONF

Course Prerequisites

- Rudimentary Understanding Of SSH & SSL
- Familiarity With The Components Of Common Networking Devices
- Basic Understanding Of HTTP



Topic Overview

+ Why Automating Via The CLI Doesn't Work Well
+ An Introduction To Data Models

The Problem Identified

- Popularity of network automation is increasing
- Historically SNMP was used for monitoring and troubleshooting
- SNMP not practical for network configuration
- Current approaches towards automation hindered due to:
  - Device-specific CLI scripts
  - Implementation on rigid, closed tools
  - No standardization about data representation other than SNMP MIBs
- Vendor-neutral protocols and data models were required

The benefits of network automation are becoming more apparent and the concept is gaining popularity

Network monitoring, troubleshooting and configuration are all components of Automation

Historically, SNMP has been utilized for network monitoring and troubleshooting, but it falls short for configuration of devices.

No standardization of how data should be represented, stored or retrieved other than SNMP MIBs (which lack support for many configuration tasks)

Automation requires vendor-neutral protocols that define data models, and how to retrieve and push data from network devices

Automation Via The CLI?

- Historically, scripts were developed that pushed CLI commands in a crude attempt at automation
- The CLI is human-friendly
  - Output is separated by commas, whitespace, etc
  - Output is sorted into columns and rows for readability
  - Misspellings occur, but can be overlooked
- The CLI is not suitable for automation

Problems With The CLI For Automation

- CLI is not standardized across different vendors (or even within the same vendor)
- Dependencies exist which vary across platforms
  - Some platforms require VLAN creation prior to interface application...others do not
- CLI provides limited (if any) error reporting
- CLI output is formatted text for human readability...NOT structured data suitable for input to software programs
- Command output can vary from platform to platform

Unpredictable error warnings:

- Sometimes a misconfigured command will produce a warning. Other times, it will not
- Some commands, if conflicting with other commands will not be accepted and generate error output
- Other times the command can be input either with no effect, or disaster

If Not The CLI...Then What?

- A standard Data Model was required to represent things such as:
  - What a device can do
  - What information a device can provide
  - The exact syntax and structure of representing the above-mentioned items

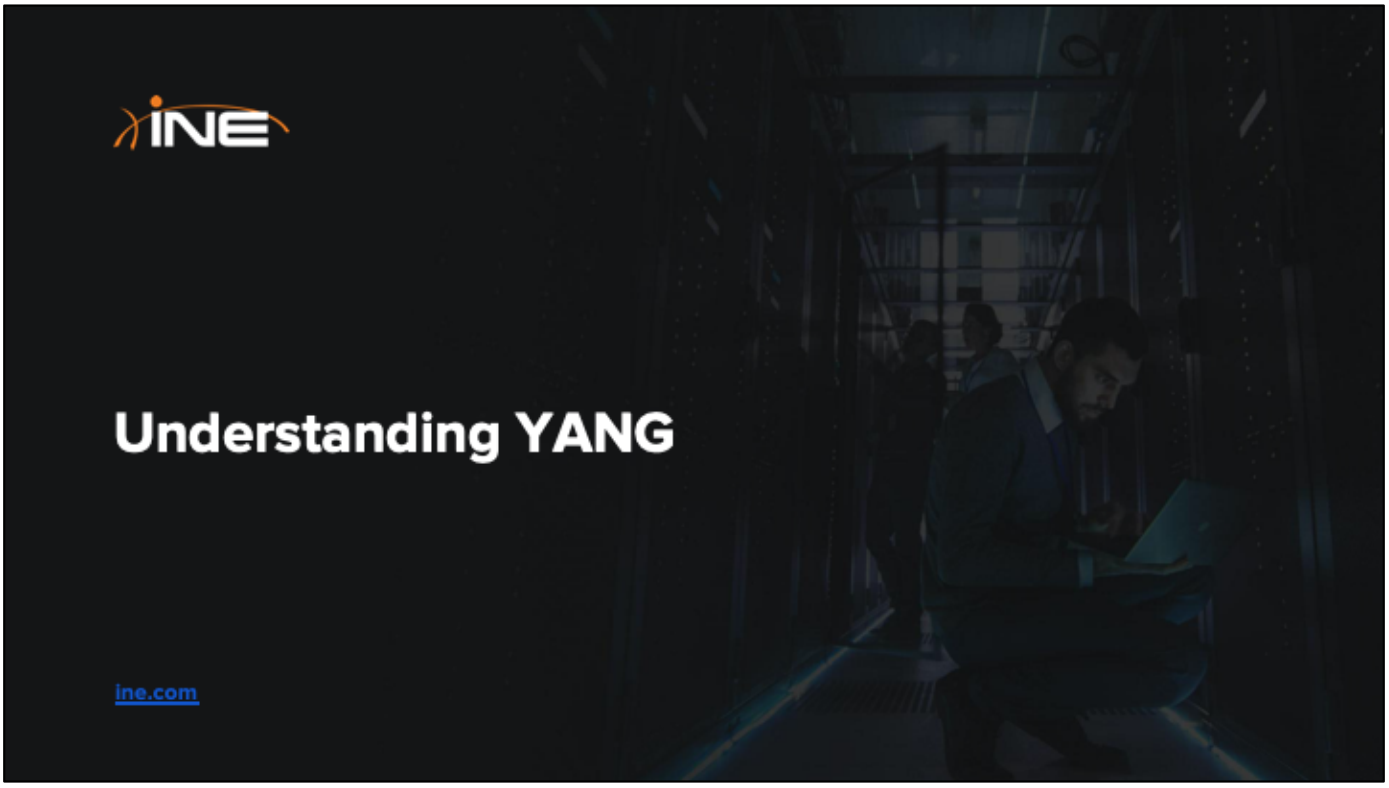
What Are Data Models?

- An intuitive, standard way to describe something (data)
- Describes all of the characteristics/attributes of data
  - What type of data is it?
    - Interface
    - Route Table
  - What characteristics does the data have?
    - A descriptive name?
    - Numerical values that are descriptive of the data (interface numbers, subnet masks, etc)
  - How is the data to be represented
    - Booleans?
    - Strings?
    - Integers?

Example Data Model

```
module example-sports {

  namespace "http://example.com/example-sports";
  prefix sports;

  import ietf-yang-types { prefix yang; }

  typedef season {
    type string;
    description
      "The name of a sports season, including the type and the year, e.g,
       'Champions League 2014/2015'.";
  }

  container sports {
    config true;

    list person {
      key name;
      leaf name { type string; }
      leaf birthday { type yang:date-and-time; mandatory true; }
    }

    list team {
      key name;
      leaf name { type string; }
      list player {
        key "name season";
        unique number;
        leaf name { type leafref { path "/sports/person/name"; } }
        leaf season { type season; }
        leaf number { type uint16; mandatory true; }
        leaf scores { type uint16; default 0; }
      }
    }
  }
}
```

- This image is courtesy of https://en.wikipedia.org/wiki/YANG



Topic Overview

+ Introduction To YANG
+ YANG Terminology
+ Examples Of YANG Modules
+ Where Do YANG Modules Come From?

Introduction To YANG

- YANG = Yet Another Next Generation
- IETF standard defined in RFC 6020 and RFC 7950
- A language for building and defining data models
- Can be used to build data models for ANY kind of data
  - How would one describe what a "person" is so that a computer program could query data about a person?
    - Build a model based on YANG principles
  - How would one describe what a "car" is so that a computer program could query data about a car?
    - Build a model based on YANG principles
  - How would one describe what a "router" is so that a computer program could query data about a router?
    - Build a model based on YANG principles

One of the earliest data modeling languages was known as Abstract Syntax Notation One (ASN.1). It was from this that the "Structure of Management Information" (SMI) was derived as another data modeling language which was used with SNMP. The structure of the SNMP MIB (Management Information Base) is based on SMI data modeling.

SMI was then upgraded to SMIv2 in 1999

Just prior to SMIv2 becoming standardized another effort was underway to upgrade it…YET AGAIN…and that effort was called SMIng (SMI Next Generation).

SMIng did not succeed in the IETF. However, later on the NETCONF (NETwork CONFiguration) Protocol was developed as a means of securely connecting with network devices for the purposes of managing and configuring them. "Yet Another Generation" of a Data Modeling Language was needed to go with NETCONF, so the principles behind SMIng were borrowed to create YANG.

YANG Modules

- YANG models that describe things are called “Modules”
- There isn’t just one YANG module, but several YANG modules for different kinds of data, and new ones developed all the time by the IETF, private companies, etc.
  - YANG modules to describe interfaces
  - YANG modules to describe Access-Lists
  - YANG modules to describe Routing tables
- Network devices typically support dozens or even hundreds of YANG modules to describe various components of themselves

Although data models, built from YANG principles, can describe literally anything, most existing models are related to computer networking.

You can explore hundreds of YANG models available at https://github.com/YangModels/yang

YANG Terminology: Containers

- YANG models utilize a tree-like structure, similar in format to XML
- YANG modules start with top-level container objects
  - An interior data node that exists in at most one instance in the data tree. A container has no value, but rather a set of child nodes. – RFC 6020
  - Generic constructs that need to be further described/defined)
    - "interface" is an example of one container object

After first bullet: Open and display the file, "XML-Configuration-Example" within "Instructor Resources folder.

YANG Terminology: Leafs

- Containers contain one or more descriptive objects technically called “Leafs”
  - Name of interface
  - Interface description
  - Type of interface
  - Operational state of interface
- Each leaf must have an associated “type” for how to describe the leaf
  - String
  - Integer
  - Boolean
  - These (and more) are examples of “types”

YANG Terminology: RO vs RW

- YANG makes a clear distinction between configurable items, and state-related information
  - RW (read-write) refers to configurable items such as
    - Description
    - IP address
    - Is interface enabled or disabled?
  - RO (read-only) refers to data that references the state of an object
    - Operational status of an interface
    - SNMP if-index value of an interface

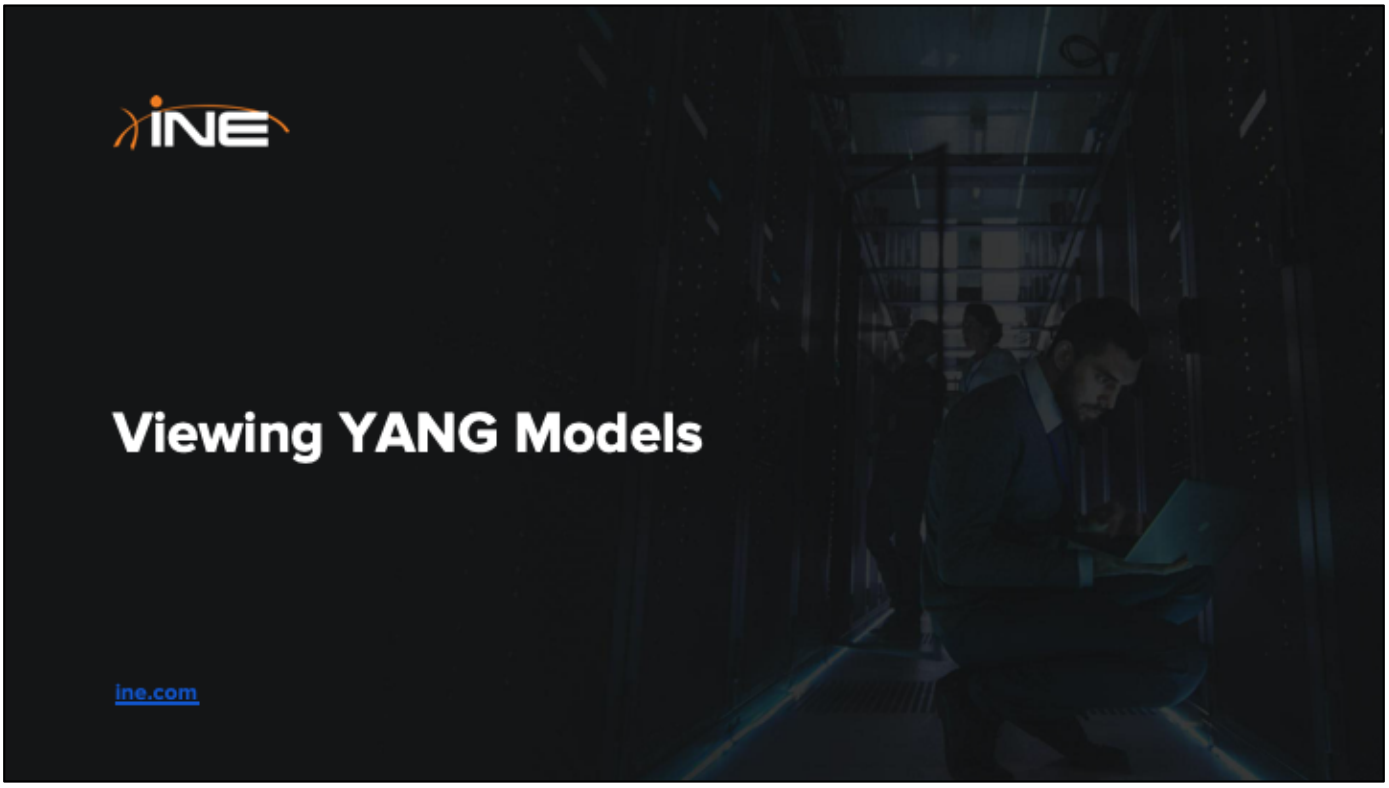
YANG Module Example

```
module ietf-interfaces {
  import ietf-yang-types {
    prefix yang;
  }
  container interfaces {
    list interface {
      key "name";
      leaf name {
        type string;
      }
      leaf enabled {
        type boolean;
        default "true";
    }
}
```

Where Do YANG Models Come From?

- YANG models come from a variety of sources
  - Private companies such as Cisco and Juniper
    - These are called “Native Models”
  - Standards bodies such as the IETF
  - Consortiums of private companies, network operators and private individuals that are working together such as the OpenConfig Organization
- Networking devices typically have built-in YANG module support depending on their software version
- Cisco IOS first incorporated some YANG modules in 16.2 (Denali) release

Models submitted from private companies are also sometimes called, “Vendor Models”.



Topic Overview

+ How To View YANG Modules

Viewing YANG Models

- YANG models are very verbose
- Stored in GitHub
- Tree structure can be viewed using pyang tool
  - "An extensible YANG (RFC 6020/7950) validator. Provides a framework for plugins that can convert YANG modules to other formats."
  - https://pypi.org/project/pyang/

Demonstration-1: Go to Github and view a text file of a YANG module to demonstrate verboseness.

Demonstration-2: Use pyang tool to demonstrate tree structure of the same YANG model that was previously demonstrated.

Topic Overview

+ Overview Of NETCONF
+ NETCONF & SNMP Compared
+ NETCONF Data Encoding
+ NETCONF Protocol Stack
+ NETCONF Terminology
+ Datastores

NETCONF Overview

- NETCONF = Network Configuration protocol
- Defined in RFCs 4741 & 6241
- NETCONF & RESTCONF describe the protocols and methods for transport of network management data
  - Utilizes YANG data models to communicate with network devices
  - **Securely runs over SSH (Port-830)**
  - TLS and SOAP (Simple Object Access Protocol) also supported

NETCONF Overview

- Unlike SNMP, NETCONF can distinguish between configuration data and operational (state) data
- NETCONF utilizes paths (similar to HTTP) to describe resources whereas SNMP utilizes OIDs
- Encodes data using either XML or JSON (most commonly XML)

NETCONF & SNMP Compared

Feature	SNMP	NETCONF
Resources	OIDs	Paths
Data models	Defined in MIBs	YANG core models
Data modeling language	SMI	YANG
Management operations	SNMP	NETCONF
Encoding	BER	XML, JSON
Transport stack	UDP	SSH/TCP

BER = Basic Encoding Rules

Basic Encoding Rules (BER) is the set of rules for encoding ASN.1 defined data into a particular representation for transmitting to another system

NETCONF Data Encoding

- NETCONF encodes data using either XML or JSON
  - Most commonly XML
- Router configs can be displayed in XML format
- XML can be easily translated into JSON
- https://codebeautify.org/xmltojson

NETCONF Protocol Stack

NETCONF Layer	Examples
Content	XML (Primarily using YANG)
Operations	<get>, <validate> <get-config>, <lock>, <unlock>, <delete-config>, **many others**
Messages	<RPC>, <RPC-REPLY>
Protocols	SSHv2, SOAP, TLS

What Can NETCONF Do?

- Collect the status of specific fields
- Change the configuration of specific fields
- Take administrative actions
- Send event notifications
- Backup and restore configurations
- Test configurations before finalizing the transaction

Administrative actions would involve things like collecting statistics and data, shutting down interfaces, reloading a router, etc.

A device that is able to accomplish “streaming telemetry” could utilize NETCONF to package its event notifications.

Testing of configurations doesn’t necessarily mean testing if a feature will work as expected, but rather is the data you pushed to the device supported? Are you asking the device to do something it doesn’t understand? Do any of your commands conflict with existing commands?

NETCONF Terminology

- NETCONF Agent
  - Router, switch, other NETCONF-capable devices
- NETCONF Manager
  - Client application used by Network Admins
- Datastores
  - Database/table of stored data by the Agent
  - Target of NETCONF commands
  - Manipulated using NETCONF operational commands sent as RPC messages

NETCONF commands (to read data or modify it) are targeted towards certain datastore
----“Running” is an example of a NETCONF datastore and the ONLY one that is mandatory

Some datastores are writeable while others are not

NETCONF datastore are manipulated using the following commands:
<get> Requests the running configuration and state information of the device
<get-config> Requests some or all of the configuration from a datastore
<edit-config> Edits a configuration datastore by using CRUD operations
<copy-config> Copies the configuration to another datastore
<delete-config> Deletes the configuration

NETCONF commands sent using RPC (Remote Procedure Call) messages in XML format

NETCONF Datastores

- Target of NETCONF Operations
- May hold copies of configuration data
- Not all datastores supported by all devices
- “Running Config” is the only required datastore
- Some datastores are RW (read-write), others are RO (read-only)

Topic Overview

+ NETCONF Implementation
+ Enabling NETCONF In IOS-XE
+ Monitoring NETCONF Sessions
+ Verifying NETCONF Datastores

NETCONF Implementation

- NETCONF was not meant to be utilized manually, but implemented by software applications
- Within Cisco IOS versions that support NETCONF, a "Global Session Lock" option is available
- NETCONF can be manually invoked for testing and lab purposes from several tools
  - Postman
  - Python scripts
  - cURL (Linux Command Line Tool)
  - Terminal windows

Because NETCONF sends and receives using XML it is not human-friendly

Session-Locks: Owner of a NETCONF session can lock the configuration parser
---Ensures consistency and prevents conflict from other NETCONF users as well as non-NETCONF clients (such as SNMP, CLI scripts and human users)
---Global lock is revoked once an active session is killed

Software applications abstract the details so we don't have to be overwhelmed with XML output

Cisco IOS-XE NETCONF Configuration

1. Create a self-signed trustpoint if none exists

```
Router(config)#ip http secure-server
% Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 1 seconds)

Router(config)#
*Nov 14 17:20:00.919: %SSH-5-ENABLED: SSH 1.99 has been enabled
*Nov 14 17:20:01.095: %PKI-4-NOAUTOSAVE: Configuration was modified.  Issue "write memory" to save new certificate
```

```
crypto pki trustpoint TP-self-signed-4044123920
 enrollment selfsigned
 subject-name cn=IOS-Self-Signed-Certificate-4044123920
 revocation-check none
 rsakeypair TP-self-signed-4044123920
!
!
crypto pki certificate chain TP-self-signed-4044123920
 certificate self-signed 01
  30820228 30820194 A0030201 02020101 300D0609 2A864886 F70D0101 05050030
  31312F30 2D060355 04031326 494F532D 53656C66 2D536967 6E65642D 43657274
  69666963 6174652D 34303434 31323339 3230301E 170D3139 31313134 31373230
  30315A17 0D323030 31303130 30303030 305A3031 312F302D 06035504 03132649
  4F532D53 656C662D 5369676E 65642D43 65727469 66696361 74652D34 30343431
  32333932 3030819F 300D0609 2A864886 F70D0101 01050003 818D0030 81890281
  81008CD0 7E7B7739 6C5FB0ED 6D124E28 BE3D50BD A391524F 341B3F5E A45CF750
  A6095A7A FC7779A7 68B88531 BA1C202D 39D714B7 13D7FB94 BA768585 5D6E6F25
  EE920E91 4711F0E5 C5D4A64E FAF23C80 75B94FD1 CAC09E8F 3E475F1E 001E2D3E
  685DB220 E7F931E0 14E3B884 3D6C8756 556D91E9 8A9754FF 5038C36D 65651D01
  A4B50203 010001A3 53305130 0F060355 1D130101 FF040530 030101FF 301F0603
  551D2304 18301680 14736EB3 1D0325C5 C788B57D C8907C4D 2F198F81 F3301D06
  03551D0E 04160414 736EB31D 0325C5C7 88B57DC8 907C4D2F 198F81F3 300D0609
  2A864886 F70D0101 05050003 81810088 C4AFBBD9 E1A91CDB 7C0B7D77 22A7D9CA
  10FD203A 984C8EDB 35CC2911 15737E71 E95B2D60 3EA95221 DC39BC5C 5A6F86F4
  586D81BC 4CBD872C 702D09E6 46D3DCD7 C68C9B61 8107ED1F AA47B51F 567C88BD
```

NETCONF transport is SSH, therefore device requires a trustpoint (i.e PKI Certificate Authority)

Because NETCONF does NOT utilize HTTPS you don't HAVE to issue the "ip http secure-server" command, but this command is a quick-and-easy way to create a self-signed Digital Certificate in the device. Many versions of Cisco IOS already have a Certificate pre-configured.

Cisco IOS-XE NETCONF Configuration

2. Configure a username/password statement that provides Privilege-Level-15 access

   Router(config)#**username Admin privilege 15 password INE**

3. Enable NETCONF-YANG

   Router(config)#**netconf-yang**

4. (Optional) Change default NETCONF TCP port

   Router(config)#**netconf-yang ssh port <1-65535>**

The default TCP port for NETCONF is 830

NOTE: IOS-XE configuration guides indicate you must also configure AAA Authentication and Authorization, but in my lab tests, as long as I had a “usename xxxx privilege 15 password yyyy” command, I was able to login via SSH and issue NETCONF-YANG XML commands WITHOUT AAA.

Monitoring NETCONF Sessions

```
RS1-CSRv-1#show netconf-yang session
R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 1

session-id  transport    username            source-host          global-lock

-------------------------------------------------------------------------------
25          netconf-ssh  Admin               199.99.99.241        R
```

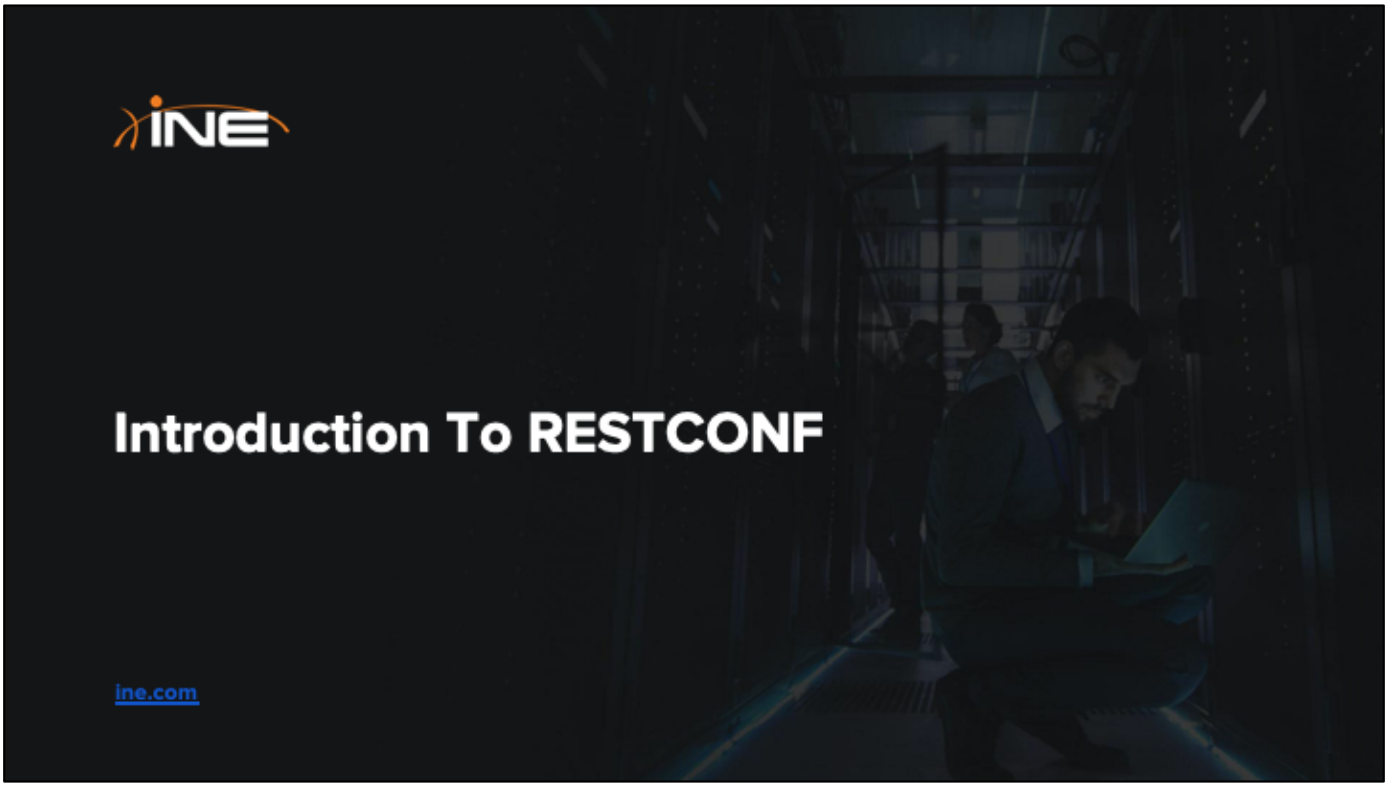
Notice that in this output, not only do we have confirmation of a NETCONF-YANG session, but we can see that whoever is controlling the session has locked the Running-Config datastore with a Global Lock.

Verifying NETCONF Datastores

```
RS1-CSRv-1#show netconf-yang datastores
Datastore Name                : running
Datastore Name                : candidate
```

Topic Overview

+ Demonstrating NETCONF-YANG



Topic Overview

+ An Overview Of REST
+ RESTCONF Introduction
+ NETCONF Or RESTCONF?
+ RESTCONF Commands

REST

- **Re**presentational **S**tate **T**ransfer
- An architectural style for **distributed hypermedia systems developed by Roy Fielding in his 2000 dissertation**
- REST API commands utilize standard HTTP “verbs” (GET, PUT, POST, DELETE)
- REST APIs typically encode data in JSON or XML format

RESTCONF represents a way of doing the same things that NETCONF can do, but using RESTful APIs. So it makes sense to start with an overview of REST.

Each “thing” that a REST API can retrieve, modify, or delete is called a “resource” whether it be an item in a database, a document, or an image. A “resource” is a way of abstracting data so you don’t have to be specific about what KIND of data it is.

-

There are a lot of very good (and highly detailed) explanations about the differences between “PUT” and “POST”. For the purposes of this introduction to REST APIs let’s just say that both can be used to Create or Update a resource. If you actually plan on creating (i.e software development) a REST API from scratch, then you’ll definitely want to dig into the specifics and understand the minutia between them.

HTTP Verbs & CRUD

- HTTP verbs are used to perform actions on REST API resources
- For database developers (familiar with CRUD) the HTTP verbs used by REST map nicely to CRUD:

HTTP Verb Used By REST	CRUD Database Commands
Post	Create
Get	Read
Put	Update
Delete	Delete

If you're already familiar with the concept of HTTP verbs you might notice that one is missing from the chart above…"Patch". PUT and PATCH are very similar;

- PUT: Commonly used to completely replace/override all contents of a resource record (if certain fields are not included they are removed).
- PATCH: used to update a record based on the information provided (or patch it with what you provide), leaving any fields not passed along, intact.

Cisco's "Intent" REST API does not make any use of any "PATCH" commands.

RESTCONF Introduction

- RESTful APIs were introduced at the same time as NETCONF
- There were benefits to merging REST concepts with NETCONF...hence, RESTCONF
- RESTCONF = NETCONF/YANG but with changes
  - Removed SSH as transport
  - Added HTTPS as transport
  - Added JSON as data encoding method (does support XML)
- Defined in RFC 8040

"The goal of RESTCONF is to provide a RESTful API experience while still leveraging the device abstraction capabilities provided by NETCONF." - Cisco ENCOR OCG

NETCONF Or RESTCONF??

- RESTCONF is a functional sub-set of NETCONF, not a replacement for NETCONF
- As HTTPS-based, each RESTCONF message is independent of other messages
  - REST does not keep track of state
  - REST operations do not know what has been done previously, or what may come next
- Some network operations must be done within the context of a single session
  - IOS-XR, changing config and committing it must be done within a single session
  - Not suitable for RESTCONF

RESTCONF Commands

+ RESTCONF utilizes standard HTTP Verbs utilized by typical RESTful APIs

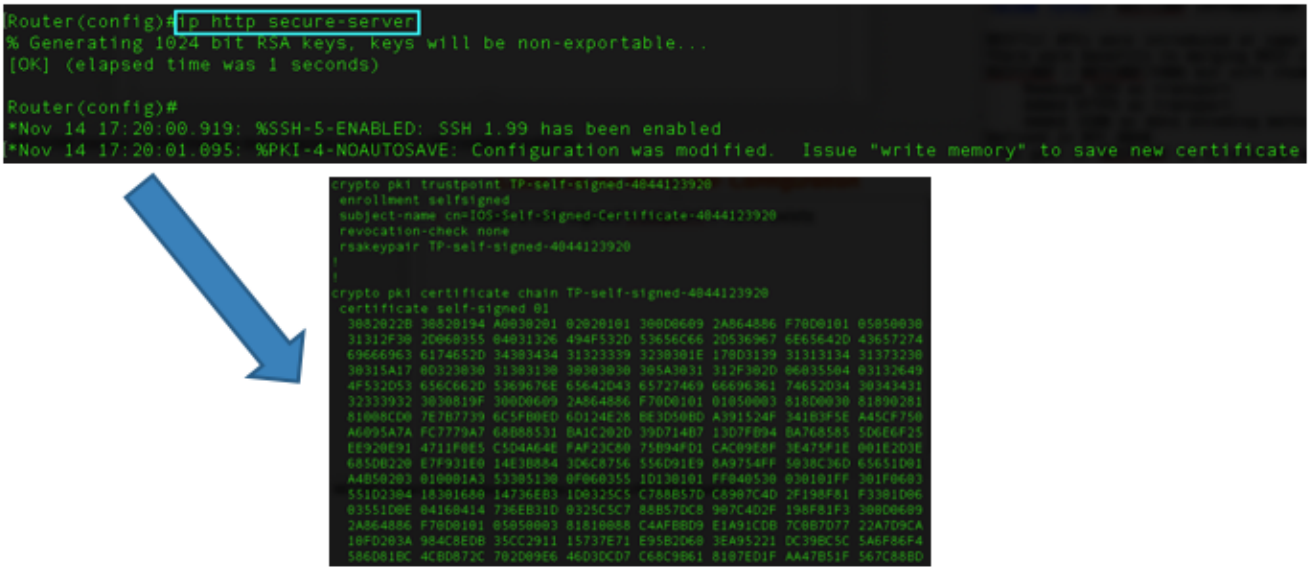
RESTCONF	NETCONF
GET	`<get>`, `<get-config>`
POST	`<edit-config>` (operation="create")
PUT	`<edit-config>` (operation="create/replace")
PATCH	`<edit-config>` (operation="merge")
DELETE	`<edit-config>` (operation="delete")

Topic Overview

+ Enabling RESTCONF On IOS-XE Devices

Cisco IOS-XE RESTCONF Configuration

1. Create a self-signed trustpoint if none exists



RESTCONF transport is SSL/TLS, therefore device requires a trustpoint (i.e PKI Certificate Authority)

Because RESTCONF utilizes HTTPS you have to issue the “ip http secure-server” command. This command is also a quick-and-easy way to create a self-signed Digital Certificate in the device. Many versions of Cisco IOS already have a Certificate pre-configured.

Cisco IOS-XE RESTCONF Configuration

2. Configure a username/password statement that provides Privilege-Level-15 access

   Router(config)#**username Admin privilege 15 password INE**

3. Enable RESTCONF

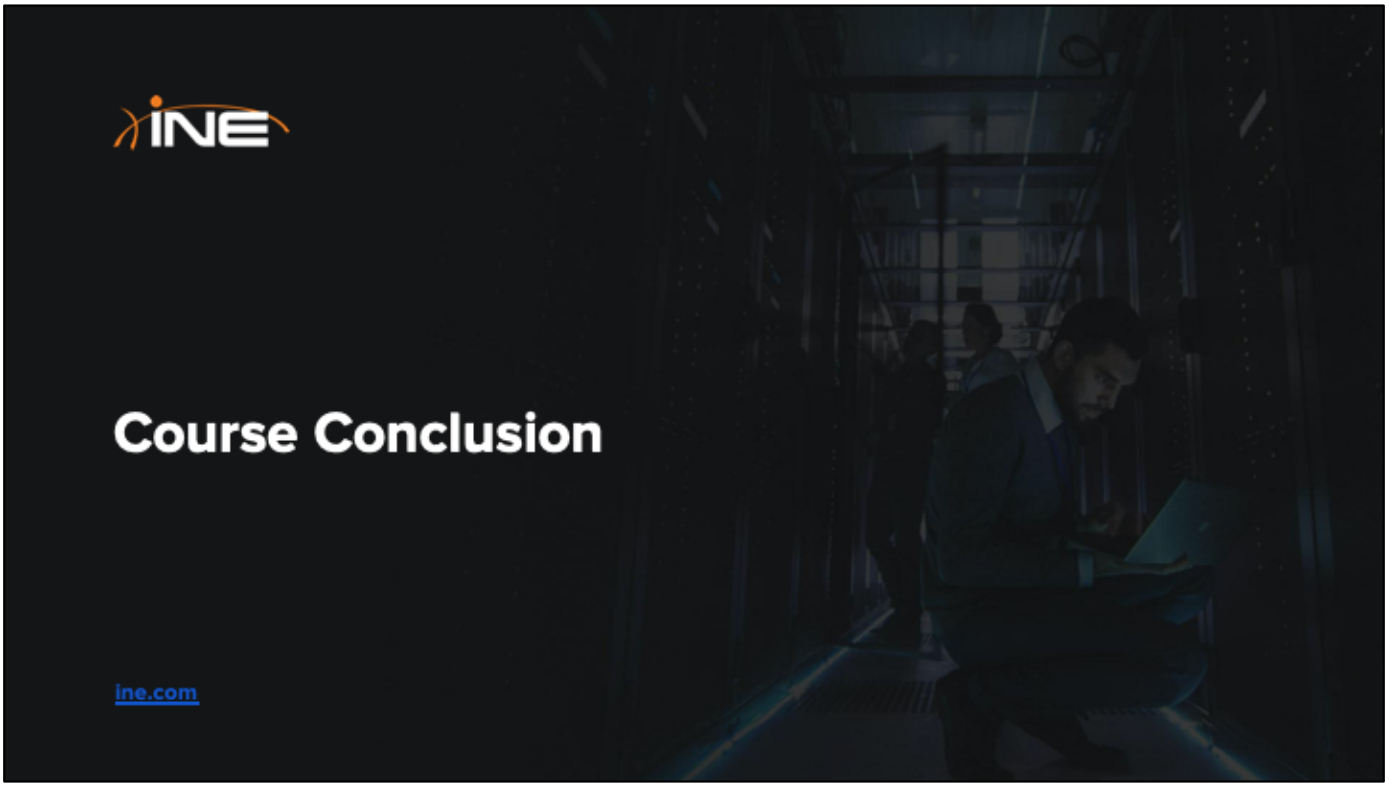
   Router(config)#**restconf**

The default TCP port for NETCONF is 830

NOTE: IOS-XE configuration guides indicate you must also configure AAA Authentication and Authorization, but in my lab tests, as long as I had a “usename xxxx privilege 15 password yyyy” command, I was able to login via SSH and issue NETCONF-YANG XML commands WITHOUT AAA.

Topic Overview

+ Simple RESTCONF Demonstration



Course Conclusion

- Introduced you to the concepts of Data Modeling as implemented by YANG
- Summarized YANG terminology & concepts
- Demonstrated how to view YANG modules
- Introduced NETCONF, why it was needed and what it does
- Introduced RESTCONF, why it was needed and what it does
- Demonstrated both NETCONF & RESTCONF