CISCO Engage | Tech Day

Getting Started with Cisco APIs

Adam Sankey
Solutions Engineer

Date placeholder

Welcome Cisco Network Academy Students!

- Started in 1997
- Over 28,000,000 students
- 12,200 organizations offering Cisco Netacad courses
- Available in 195 countries
- Equips students with technical and professional skills aligned to industry needs and certifications

Agenda

1. What are we doing here?
2. REST APIs
3. HTTP Methods, Headers, and Status Codes
4. API Authentication
5. API Formatting
6. Webhooks
7. API Security
8. AI / LLM
9. Demos

Why Automate?

Seriously though, why?

Scale

- As networks grow, managing them manually becomes increasingly challenging
- Reduce time spent on routine tasks
- Advanced / custom reporting
- Perform repetitive tasks faster and with more accuracy

Consistency

- Reduce the opportunity for human error
- Enforce and verify consistent security and compliance policies

Rapid Deployment

- Standardized applications/environments can be delivered much faster

CLIs, GUIS, and APIs

CLI

- Efficient way to interact with a single device
- Can be difficult to learn and use, especially for beginners
- Great for one-off changes
- Not efficient for interacting with many devices
- Doesn't offer visual feedback

GUI

- Underrated form of automation
- More intuitive and easier to learn
- Provide visual feedback
- Fewer opportunities to make syntax errors
- Difficult to automate mouse clicks

API

- Programmatic interface for applications/software to send and receive data between each other
- Designed for automation
- Allow different software systems to communicate and interact with each other
- Not ideal for one-off changes
- APIs can be complex to use and often require programming knowledge

REST vs SOAP APIs

REST

- REpresentational State Transfer
- REST is an architectural style, not a protocol
- Communicate over HTTP protocol
- Uses standard HTTP methods like GET PUT POST PATCH DELETE
- Data can be formatted in JSON, XML, plain text, etc.
- Used by every Cisco product except one

SOAP

- Simple Object Access Protocol
- SOAP is a protocol
- Primarily uses XML
- Can use HTTP, SMTP, TCP, and more
- Used by Cisco Unified Call Manager

HTTP Methods

HTTP Method	Typical Purpose (CRUD)	Description
GET	Read	Retrieve resource details from the system. Example: Get the running-config from a device
POST	Create	Used to create a new object or resource. Example: Add a new SVI to a router
PUT	Update	Typically used to replace or update an existing resource. Example: Update interface details on a device
PATCH	Update	Used to modify some details about a resource. Example: Similar to PUT
DELETE	Delete	Remove a resource from the system. Example: Delete an SVI from a device

HTTP Headers

Header	Example Values	Purpose
Content-Type	application/json application/xml	Specify the format of data being sent to the server
Accept	application/json application/xml	Request the format of data sent from the server
Authorization	Basic *ENCODEDSTRING* Bearer *TOKEN*	Provide authentication to the server for verification

JSON vs XML

JSON

```
{
  "Cisco-IOS-XE-native:Loopback": {
    "name": 1,
    "ip": {
      "address": {
        "primary": {
          "address": "1.1.1.1",
          "mask": "255.255.255.255"
        }
      }
    }
  }
```

XML

```
<Loopback xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native" ... >
    <name>1</name>
    <ip>
        <address>
            <primary>
                <address>1.1.1.1</address>
                <mask>255.255.255.255</mask>
            </primary>
        </address>
    </ip>
</Loopback>
```

HTTP Status Codes

Status Code	Message	Meaning
200	OK	Request was successfully sent
201	Created	New Resource Created
204	No Content	Successful request, but there is no data to return
301	Permanently Moved	The URL/URI has permanently changed
400	Bad Request	Request was invalid due to malformed syntax
401	Unauthorized	Authentication is missing or invalid
403	Forbidden	Request is understood but not allowed
404	Not Found	Requested resource is not found
500	Internal Server Error	Generic server-side error
503	Service Unavailable	Server is unable to complete request

API Authentication

- None (No Auth): the web API resource is public; anybody can use it
- HTTP Basic Authentication: a username and password are passed to the server in an encoded string (Base64 encoding)
  - Authorization: Basic *ENCODEDSTRING*
- Open Authorization (Oauth): Standard framework for a flow to retrieve an access token from an Identity Provider
- API Keys: Client-Server secret key. The API Key can be sent in a query string, in the header (using Authorization or a custom key) or in a cookie.
  - Meraki – Authorization: Bearer *TOKEN*
  - Catalyst Center – x-auth-token (after basic authentication)
- Authorization may require refreshing of tokens

Meraki API

- Generate an API Key
- Organization > APIs and Webhooks > API Keys and Access > Generate API Key
- Get your Org ID
  - Listed at the bottom of most page
  - OR
  - Make a GET request to https://api.meraki.com/api/v1/organizations
- Get your Network ID
  - Make a GET request to https://api.meraki.com/api/v1/organizations/*ORG_ID*/networks
- Make API Requests using the Organization and Network IDs

Catalyst Center API

- Create an API user account
  - Menu > System > Users and Roles
  - Use a pre-define role such as Super-Admin, Network Admin, or Observer
  - Alternatively, you can create a new custom role
- Request a Token
  - Send POST request to https://*CATC_SERVER*/api/system/v1/auth/token
- Send subsequent API requests with the x-auth-token returned by the previous step

Webhooks

- Used to send real-time data from one application to another whenever a given event occurs
- Generally in the form of a HTTP POST request, which is sent from the source app to the destination URL when a certain event happens
- Webhooks are more efficient than polling for data as they only send data when there are actual changes to report
- Similar to SNMP Trap
- Many Cisco products have built-in webhook integrations
  - ServiceNow
  - Pager Duty
  - Webex Messenger
  - MS Teams

Flask

- Flask is a micro web framework written in Python
- Flask is known for its simplicity and ease-of-use. It provides the bare essentials for web development without the complexity of learning a full-stack framewor
- Flask is designed to make creating RESTful services easy
- HTTP Request Handling: Flask provides request handling using the request object, and allows easy access to cookies, data, form information, and file uploads across different HTTP methods

Webex Bot (Flask App) Flow

1. User sends a message to the bot in the Webex app
2. Webex sends a webhook to URI specified in the bot configuration
3. Bot web server (AWS EC2 instance) receives webhook
4. Python flask app parses data and determines if there is a GIF on my S3 bucket that has a name matching the message contents
5. If a matching GIF is found, the GIF is sent as a message to the Webex API

API/Automation Security Considerations

- Don’t store passwords or keys in your code
  - Store API keys and authentication credentials as environment variables
  - OR
  - Use a dedicated key management system or service, which can store, manage, and maintain API keys securely.
- Use regular key rotation
- Use the Least Privilege Principle
- Don’t run scripts with root/admin rights

AI / LLM Usage

- Code Generation
  - Create python scripts or flask apps
  - LLMs have knowledge of many libraries and APIs
- Code Review
  - Describe what each block of code does
  - Debug code
  - Make suggestions regarding best practices, error handling, efficiency
- Documentation
  - Add comments to code

Recommended Learning

- Cisco DevNet (developer.cisco.com)
  - Guided Labs
  - Sandbox Environments
  - API documentation
- Cisco Automation Certification(s)
- Cisco U – https://u.cisco.com
- Postman
  - Use collections!
- W3schools python course
- Flask
  - Start with a ”hello world!”
  - Dump POST requests to the console
  - Parse and output specific text to console

Thank you