

Programmability Webinar Series with DevNet

Session 4: Adding API Skills to Your Networking Toolbox

Patrick Rockholz, Systems Engineer

Hostess: Kara Sullivan

Jointly presented by DevNet & NetAcad

22 January, 2019

© 2018 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Welcome to the 4th session of the Programmability with Cisco DevNet webinar series

- Use the Q and A panel to ask questions.
- Use the Chat panel to communicate with attendees and panelists.
- A link to a recording of the session will be sent to all registered attendees.
- Please take the feedback survey at the end of the webinar.

The Webinar Series

Date Topic

- Oct'18 Networking with Programmability is Easy
- Oct'18 A Network Engineer in the Programmable Age
- Nov'18 Software Defined Networking and Controllers
- ➔ Jan'19 Adding API Skills to Your Networking Toolbox
- Feb'19 The New Toolbox of a Networking Engineer
- Mar'19 Program Networking Devices using their APIs
- Apr'19 Before, During, and After a Security Attack
- May'19 Play with Linux & Python on Networking Devices
- Jun'19 Automate your Network with a Bot



All Series Details can be Found @ <http://bit.ly/devnet2>

The Webinar Series – Raffle & Certificates

Raffle

- ✓ We will be raffling off a total of 15 Amazon gift cards in the amount of \$25 US dollars at the end of this series.*
- ✓ 10 Amazon gift cards in the amount of \$25 US dollars raffled off to everyone who participates in all of the live sessions
- ✓ 5 Amazon gift cards in the amount of \$25 US dollars raffled off to everyone who participates in all of the sessions by either attending the live sessions or viewing/downloading the recording (can be a combination of the two in this raffle).

* Please note that this is a raffle and not everyone who qualifies will receive a gift card. There will be a total of 15 winners.



Certificate of Participation

- ✓ There will be an opportunity to sign up for a Certificate of Participation at the end of this series.
- ✓ To qualify, you must have participated in all sessions of the series.
- ✓ You can do this by attending the live sessions, viewing the recordings, or a combination of the two.
- ✓ Certificates will not be given out for individual sessions, but for the series as a whole.





Session 4

Adding API Skills to Your Networking Toolbox

Patrick Rockholz
Systems Engineer
22 January, 2019

 @patrickrockholz





Stone Age

Spanning Tree
VLANs



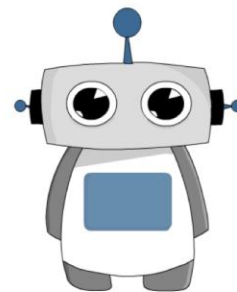
Bronze Age

Routing Protocols
WAN Design
IP-magedon



The Renaissance

SDN
OpenFlow
Controllers
Overlays
MP-BGP
VXLAN
Micro-Segmentation
White Box



Programmable Age

Cloud
Python
REST / APIs
NETCONF / YANG
“Fabrics”
Network Function
Virtualization (NFV)
Containers
DevOps
NetDevOps!

The Four Ages of Networking.....

Digital Organizations Embody Digitization

- A organization uses digital technology as a *competitive advantage* for all internal and external operations.

- ▶ Established Brands are rapidly transforming to a Digital Enterprise to catch up...



- ▶ Disruptors or New Brands have beat established brands at becoming a Digital Enterprise...



... The Network!

Common Challenges



Difficult to Secure

Ever increasing number of users and endpoint types

Increase in complexity to increase scale



Difficult to Integrate and Manage

Multiple steps, user credentials, complex interactions

Multiple touch-points



Slower Issue Resolution

Separate user policies for wired and wireless networks

Unable to find users when troubleshooting

Traditional Networks Cannot Keep Up!

Network as a Platform Considerations

Where to Start?



**FASTER
INNOVATION**
Insights &
Experiences



**REDUCED
COST &
COMPLEXITY**
Automation
& Assurance



LOWER RISK
Security &
Compliance



The Network Intuitive = Intent-based Networking

Digital Business



Mobile



Security



IoT

Business Goals



Insights

Network

Translation

Capture business intent, translate to policies, and check integrity

Activation

Orchestrate policies & configure systems

Assurance

Continuous verification, insights & visibility, and corrective actions

Powered By Intent. Informed by Context.

Cisco DNA Center

Central network management system

Complete network management system

- Single pane of glass for all devices
- End-to-end health information in real time
- Granular visibility
- Simplified workflows

Automation for provisioning

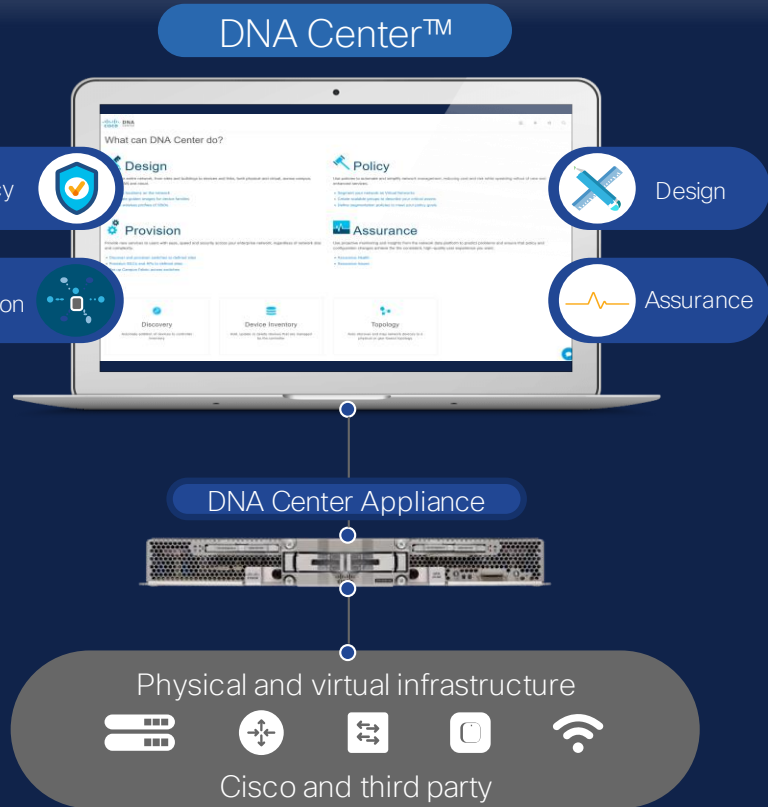
- Zero-touch deployment
- Device lifecycle management
- Policy enforcement

Analytics for assurance

- Verify intent of network settings
- Proactively resolve issues
- Reduce time spent troubleshooting

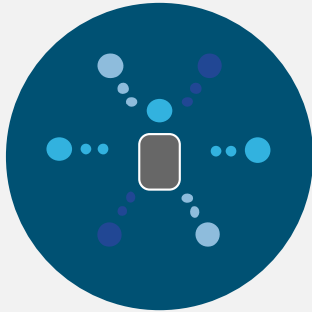
Platform for extensibility

- Integrate APIs with third-party solutions
- Integrate and customize ServiceNow
- Evolve operational tools and processes



Cisco DNA Center Platform

Open Platform



- API Catalog
- 3rd party SDK
- Process Adapters

Partner Ecosystem



- Partner Integrated Solutions

Developer Enablement



- Developer's DevNet Portal

Agenda

- Introduction to APIs & Data Formats
- REST APIs
- APIs -> Postman -> Code!
- Summary and Close

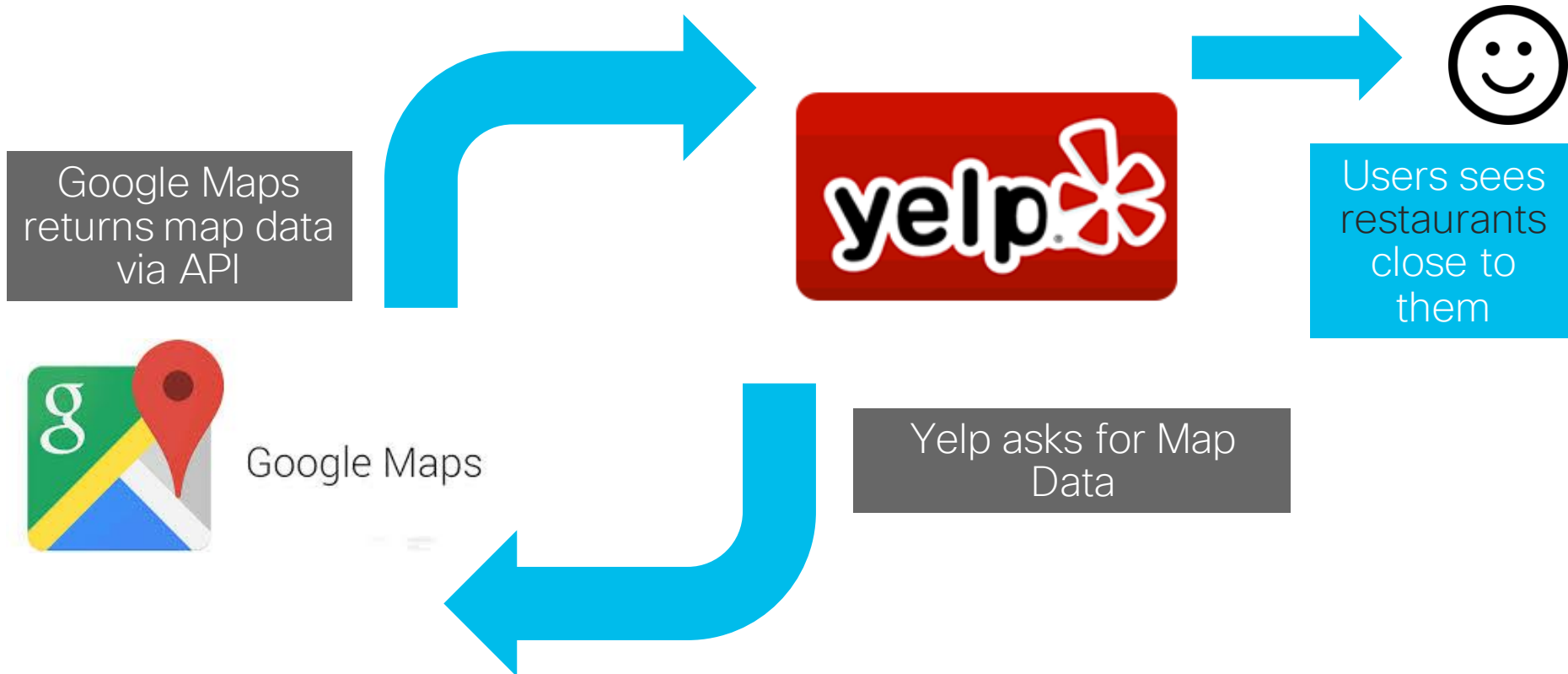
What's an API?

“...a set of clearly defined methods of communication between various software components” – Wikipedia

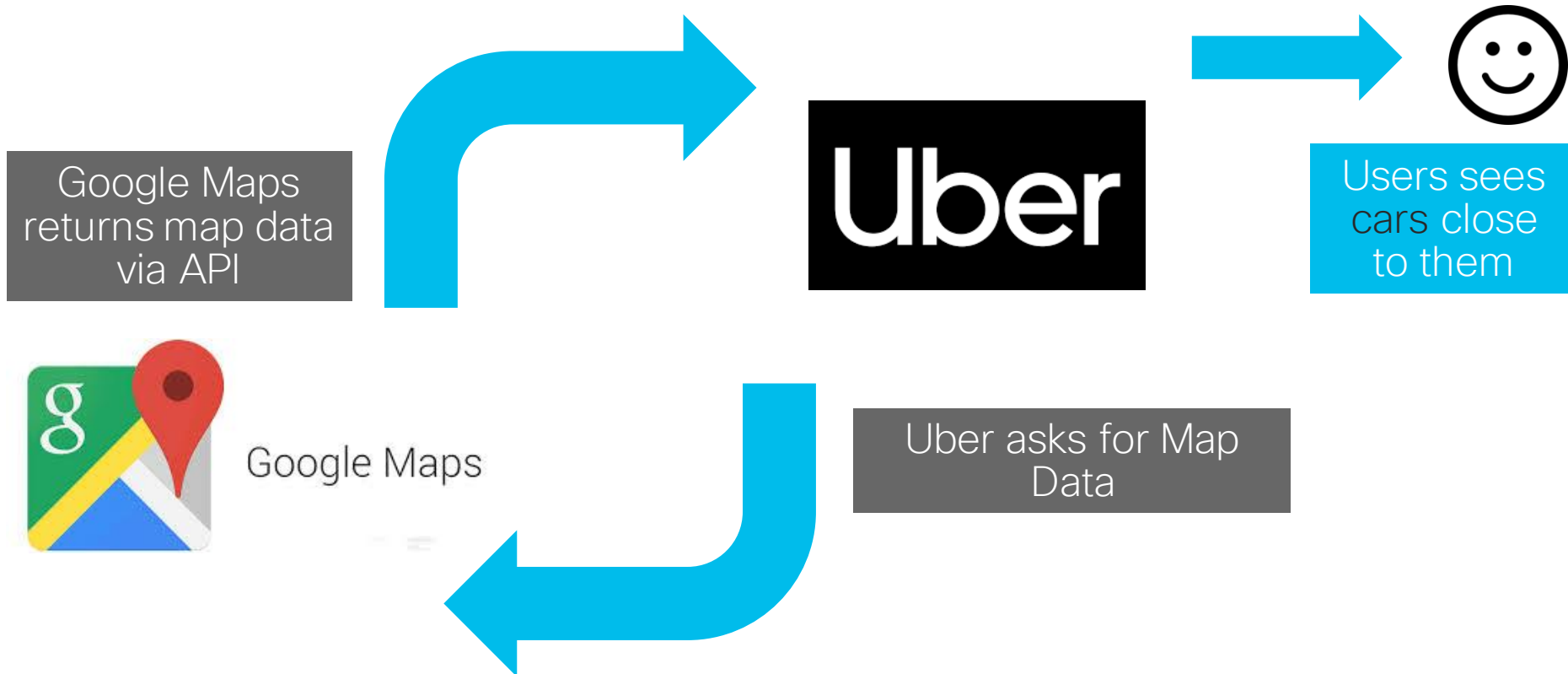
“It's a way for two pieces of software to talk to each other”

Application Programming Interface (API)

APIs help developers create apps that benefit the end user.

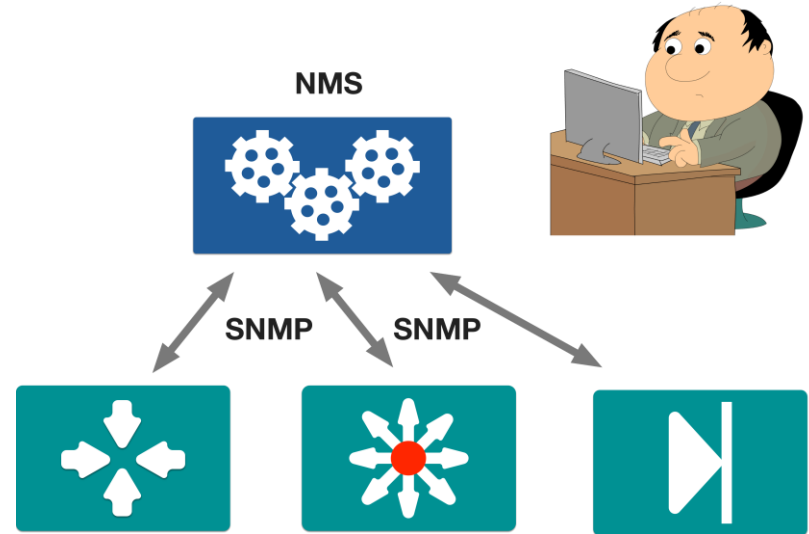


APIs help developers create apps that benefit the end user.



Simple Network Management Protocol (SNMP)

- “*designed as a programmatic interface between management applications and devices*”^{*}
- Widely used for monitoring
- Limited use for configuration
- Network Management Systems primary consumer

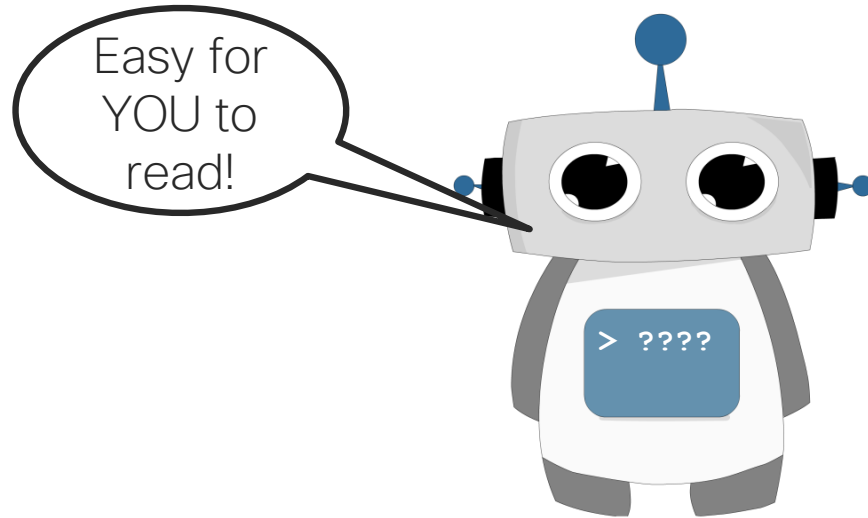


* <https://tools.ietf.org/html/rfc3535>

Importance of a Data Format

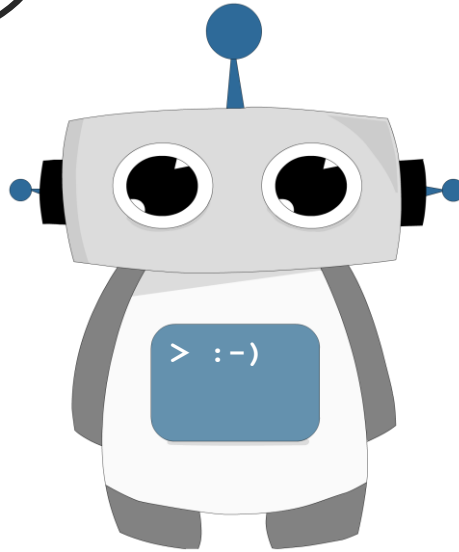
Know Your Audience

Interface	IP-Address	OK?	Method	Status	Protocol
GigabitEthernet1	10.0.2.15	YES	DHCP	up	up
GigabitEthernet2	172.16.0.2	YES	manual	up	up
GigabitEthernet3	172.17.0.1	YES	manual	up	up



Know Your Audience

Easy for
us to
read



```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "GigabitEthernet2",
        "description": "Wide Area Network",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": true,
        "ietf-ip:ipv4": {
          "address": [
            {
              "ip": "172.16.0.2",
              "netmask": "255.255.255.0"
            }
          ]
        }
      },
      {
        "name": "GigabitEthernet3",
        "description": "Local Area Network",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": true,
        "ietf-ip:ipv4": {
          "address": [
            {
              "ip": "172.17.0.1",
              "netmask": "255.255.255.0"
            }
          ]
        }
      }
    ]
  }
}
```

Common Data Formats in Programming

JSON

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet2",
    "description": "Wide Area Network",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "172.16.0.2",
          "netmask": "255.255.255.0"
        }
      ]
    }
  }
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<interface xmlns="ietf-interfaces">
  <name>GigabitEthernet2</name>
  <description>Wide Area Network</description>
  <enabled>true</enabled>
  <ipv4>
    <address>
      <ip>172.16.0.2</ip>
      <netmask>255.255.255.0</netmask>
    </address>
  </ipv4>
</interface>
```

YAML

```
---
ietf-interfaces:interface:
  name: GigabitEthernet2
  description: Wide Area Network
  enabled: true
  ietf-ip:ipv4:
    address:
      - ip: 172.16.0.2
        netmask: 255.255.255.0
```

"Key" : "Value"

- "Key" identifies/labels a set of data
- Left side of the colon
- Inside of "quotes"
- "Value" is the Data
- Right side of colon
- Can be:
 - String
 - Integer
 - Array/List
 - Bool
 - Object

```
{  
  "name": "GigabitEthernet2",  
  "description": "Wide Area  
Network",  
  "enabled": true  
}
```

JSON - JavaScript Object Notation

A **human readable** data structure that **applications use** to store, transfer, and read data.

- A data-interchange text format
- Notated with {} for objects, [] for arrays
- **Key/Value** representation
 - **"key": value**
- Whitespace not significant

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet2",
    "description": "Wide Area Network",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "172.16.0.2",
          "netmask": "255.255.255.0"
        }
      ]
    }
  }
}
```

JSON Object

- Data surrounded by { }
- An object can contain other objects or data entries
- Key/Value set separated by comma
 - No comma at the end!

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet2",
    "description": "Wide Area Network",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "172.16.0.2",
          "netmask": "255.255.255.0"
        }
      ]
    }
  }
}
```


JSON List

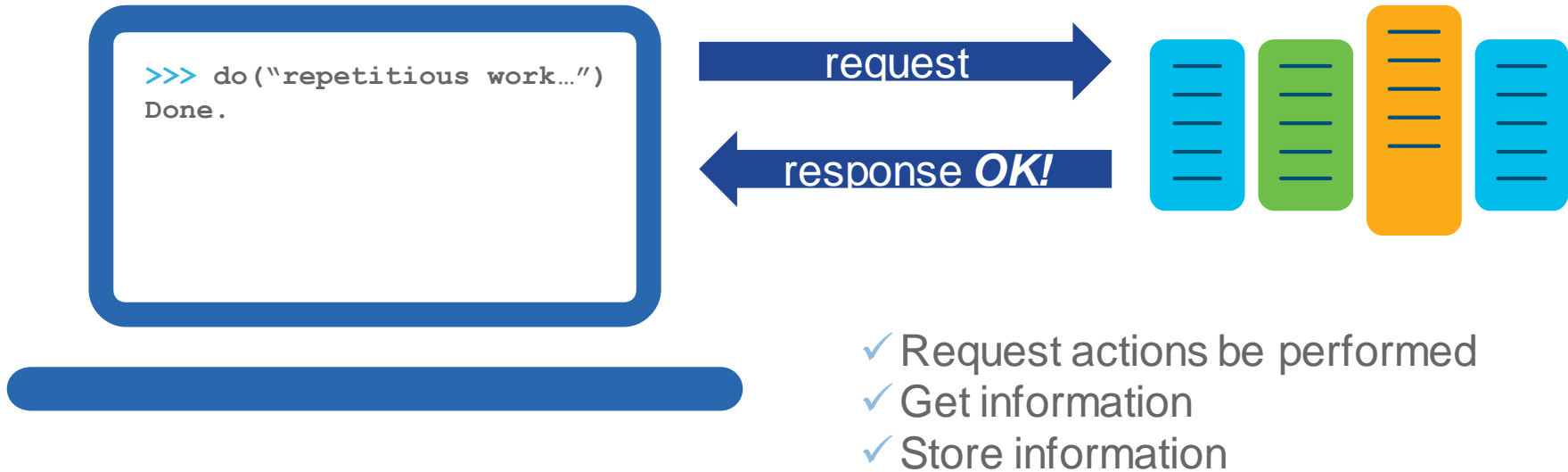
- List of data
 - Can be composed of JSON objects
- Notated with brackets
- Comma Separated

```
{
  "addresses": [
    {
      "ip": "172.16.0.2",
      "netmask": "255.255.255.0"
    },
    {
      "ip": "172.16.0.3",
      "netmask": "255.255.255.0"
    },
    {
      "ip": "172.16.0.4",
      "netmask": "255.255.255.0"
    }
  ]
}
```

Agenda

- Introduction to APIs & Data Formats
- **REST APIs**
- APIs -> Postman -> Code!
- Summary and Close

The Value-Proposition for APIs



The API is the User Interface for the software system

Just Another Use for the HTTP Protocol

- Representational state transfer (REST)
- API framework built on HTTP
- APIs often referred to as web services
- Popular due to performance, scale, simplicity, and reliability

GET

POST

PUT

DELETE

{REST}

The URI: What are you Requesting?

`http://maps.googleapis.com/maps/api/geocode/json?address=sanjose`

Server or Host Resource Parameters

- `http://` or `https://`
 - Define whether secure or open
http
- **Server or Host**
 - Resolves to the IP and port to connect to
- **Resource**
 - The location of the data or object of interest on the server
- **Parameters**
 - Details to scope, filter, or clarify a request. Often optional.

URI = Uniform Resource Identifier

HTTP Methods: What to do?

HTTP Verb	Typical Purpose (CRUD)	Description
POST	Create	Used to create a new object, or resource. <i>Example: Add new book to library</i>
GET	Read	Retrieve resource details from the system. <i>Example: Get list of books from the library</i>
PUT	Update	Typically used to replace or update a resource. Can be used to modify or create. <i>Example: Update the borrower details for a book</i>
PATCH	Update	Used to modify some details about a resource. <i>Example: Change the author of a book</i>
DELETE	Delete	Remove a resource from the system. <i>Example: Delete a book from the library.</i>

Response Status Codes: Did it work?

Status Code	Status Message	Meaning
200	OK	All looks good
201	Created	New resource created
400	Bad Request	Request was invalid
401	Unauthorized	Authentication missing or incorrect
403	Forbidden	Request was understood, but not allowed
404	Not Found	Resource not found
500	Internal Server Error	Something wrong with the server
503	Service Unavailable	Server is unable to complete request

Headers: Details and meta-data

Header	Example Value	Purpose
Content-Type	application/json	Specify the format of the data in the body
Accept	application/json	Specify the requested format for returned data
Authorization	Basic dmFncmFudDp2YWdyYW50	Provide credentials to authorize a request
Date	Tue, 25 Jul 2017 19:26:00 GMT	Date and time of the message

- Used to pass information between client and server
- Included in both REQUEST and RESPONSE
- Some APIs will use custom headers for authentication or other purpose

Data: Sending and Receiving

- Contained in the body
- POST, PUT, PATCH requests typically include data
- GET responses will include data
- Format typically JSON or XML
 - Check “Content-Type” header

```
{  
  'title': 'Hamlet',  
  'author': 'Shakespeare'  
}
```

HTTP Authentication and Security

- **None**: the Web API resource is public, anybody can place call.
- **Basic HTTP**: a username and password are passed to the server in an encoded string.
 - Authorization: Basic ENCODEDSTRING
- **Token**: a secret generally retrieved from the Web API developer portal. Keyword (ie token) is API dependent
 - Authorization: Token aikasf8adf9asd9akasdf0asd
- **OAuth**: Standard framework for a flow to retrieve an access token from an Identity Provider.
(Often used to allow users to authorize access on their behalf)
 - Authorization: Bearer 8a9af9adadf0asdf0adfa0af
- Authorization can be short-lived and require refreshing of tokens

Some REST Examples

The Internet Chuck Norris Database

```
DevNet$ curl https://api.icndb.com/jokes/random
```

```
{
  "type": "success",
  "value": {
    "id": 201,
    "joke": "Chuck Norris was what Willis was talkin' about.",
    "categories": []
  }
}
```

```
DevNet$ curl https://api.icndb.com/jokes/random?limitTo=nerdy
```

```
{
  "type": "success",
  "value": {
    "id": 537,
    "joke": "Each hair in Chuck Norris's beard contributes to make the world's largest DDOS.",
    "categories": [
      "nerdy"
    ]
  }
}
```

- <http://www.icndb.com/api/>
- No authentication needed
- Well constructed API with many options

Network Programmability with RESTCONF

The Request

```
DevNet$ curl -vk \  
  -u root:D_Vay\!\_10\& \  
  -H 'accept: application/yang-data+json' \  
https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet2  
  
> GET /restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet2 HTTP/1.1  
> Host: 10.10.20.21  
> User-Agent: curl/7.51.0  
> accept: application/yang-data+json  
> authorization: Basic dmFncmFudDp2YWdyYW50  
>
```

- `-u` provides `user:password` for Basic Authentication
- `-H` to set headers
- Lines beginning with “>” indicate Request elements
- Lines beginning with “<” indicate Response elements (next slide)

Network Programmability with RESTCONF

The Response - Headers

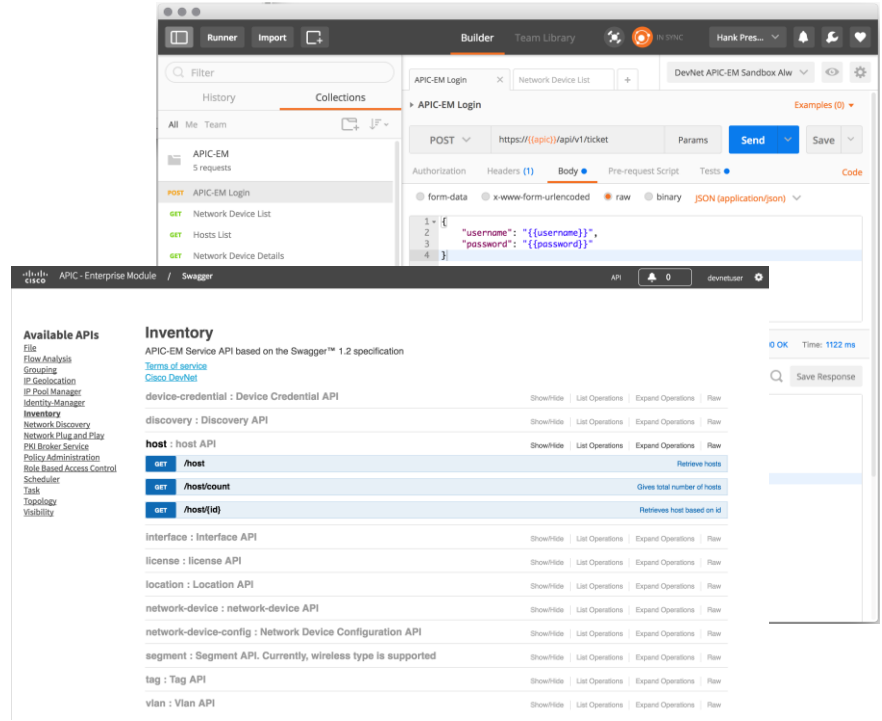
```
< HTTP/1.1 200 OK
< Server: nginx
< Date: Thu, 27 Jul 2017 00:01:52 GMT
< Content-Type: application/yang-data+json
< Transfer-Encoding: chunked
< Connection: close
< Last-Modified: Tue, 25 Jul 2017 19:15:57 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1501-10157-179272
< Pragma: no-cache
<
```

The Response - Data

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet2",
    "description": "Wide Area Network",
    "type": "iana-if-type:ethernetCsmacd",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "172.16.0.2",
          "netmask": "255.255.255.0"
        }
      ]
    },
    "ietf-ip:ipv6": {
    }
  }
}
```

Many Options for Working with REST APIs

- curl
 - Linux command line application
- Postman
 - Chrome browser plugin and application
- Requests
 - Python library for scripting
- OpenAPI/Swagger
 - Dynamic API Documentation
- Browser Developer Tools
 - View traffic and details within browser



The image shows two screenshots related to REST API work. The top screenshot is from Postman, showing a 'Builder' tab for an API endpoint. The endpoint is 'https://(apic)/api/v1/ticket' with a 'POST' method. The body is set to 'raw' and contains a JSON object: `{ "username": "{{username}}", "password": "{{password}}" }`. The bottom screenshot is from Swagger UI, displaying the 'Inventory' API documentation. It lists various endpoints such as 'device-credential', 'discovery', 'host', 'interface', 'license', 'location', 'network-device', 'network-device-config', 'segment', 'tag', and 'vlan', each with its respective HTTP method and a brief description.

Agenda

- Introduction to APIs & Data Formats
- REST APIs
- **APIs -> Postman -> Code!**
- Summary and Close

Postman: Powerful but Simple REST API Client

- Quickly test APIs in GUI
- Save APIs into Collections for reuse
- Manage multiple environments
- Auto generate code from API calls
- Cross platform
- Free-to-use



<https://www.getpostman.com>

The image shows a REST client interface with several components highlighted by red boxes and annotated with blue text and red arrows:

- URI:** The address bar shows `https://api.icndb.com`.
- Set Method:** A dropdown menu is set to `GET`.
- Manage request authorization, headers, and data (body):** The `Headers (1)` tab is active, showing a table with one header: `Accept: application/json`. The `Body` tab is also visible.
- Send Request:** A blue `Send` button is highlighted.
- Easily input Headers with auto-completion:** The header table is highlighted.
- Response Status Code:** The status bar shows `Status: 200 OK`.
- View Response details:** The `Body` tab is active, showing the response body in JSON format.
- Response Body:** The JSON response is highlighted:

```
{
  "type": "success",
  "value": {
    "id": 403,
    "joke": "Chuck Norris crossed the road. No one has ever dared question his motives.",
    "categories": []
  }
}
```

Constructing a POST Request

- Choose method
- Enter URI
- Configure headers and authentication
- Provide data
- Send and verify status

Authorization	Headers (1)	Body	Pre-request Script	Tests				
	<table border="1"><thead><tr><th>Key</th><th>Value</th></tr></thead><tbody><tr><td><input checked="" type="checkbox"/> Content-Type</td><td>application/json</td></tr></tbody></table>	Key	Value	<input checked="" type="checkbox"/> Content-Type	application/json			
Key	Value							
<input checked="" type="checkbox"/> Content-Type	application/json							

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URI:** https://sandboxapicem.cisco.com/api/v1/ticket
- Authorization:** Headers (1)
- Body:** JSON (application/json)
- Request Body:**

```
1- {
2-   "username": "devnetuser",
3-   "password": "Cisco123!"
4- }
```
- Status:** 200 OK, Time: 1121 ms
- Response Body:**

```
1- {
2-   "response": {
3-     "serviceTicket": "ST-6811-rMyZG3IACsdVHeqitjg0-cas",
4-     "idleTimeout": 1800,
5-     "sessionTimeout": 21600
6-   },
7-   "version": "1.0"
8- }
```

Save and Organize API Calls into Collections

The screenshot shows the Postman Builder interface. On the left, the 'Collections' sidebar is visible, with a red box highlighting the 'APIC-EM' collection containing 5 requests. The main area shows a POST request to 'https://sandboxapicem.cisco.com/api/v1...' with a 'Content-Type' header set to 'application/json'. The response is displayed in the 'Body' tab, showing a JSON object with a 'response' field containing a 'serviceTicket' and an 'idleTimeout'.

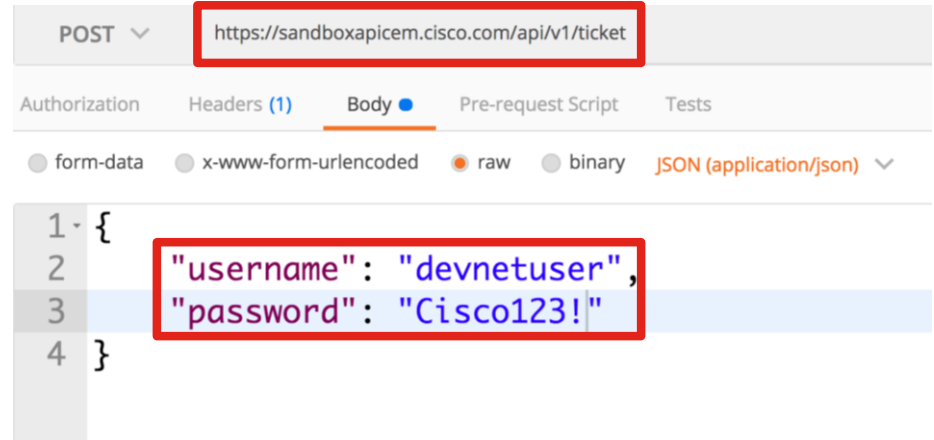
Key	Value	Description	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Content-Type	application/json		
<input type="checkbox"/>	New key	Value		Description

```
1 {
2   "response": {
3     "serviceTicket": "ST-6811
4       -rMyZG3IACSDvHeqitjg0-cas",
5     "idleTimeout": 1800,
6     "sessionTimeout": 21600
7   },
8   "version": "1.0"
}
```

The 'SAVE REQUEST' dialog box is shown on the right. It contains the following text: 'Requests in Postman are saved in collections (a group of requests). Learn more about creating collections'. Below this, there is a 'Request Name' field with the value 'APIC-EM Login' and a 'Request description (Optional)' field with the value 'Login to APIC-EM and return ticket'. There is also a 'Save to existing collection / folder' dropdown menu with 'APIC-EM' selected, and a 'Collection Name' field. At the bottom, there are 'Cancel' and 'Save' buttons.

Variables Make Requests Reusable and Flexible

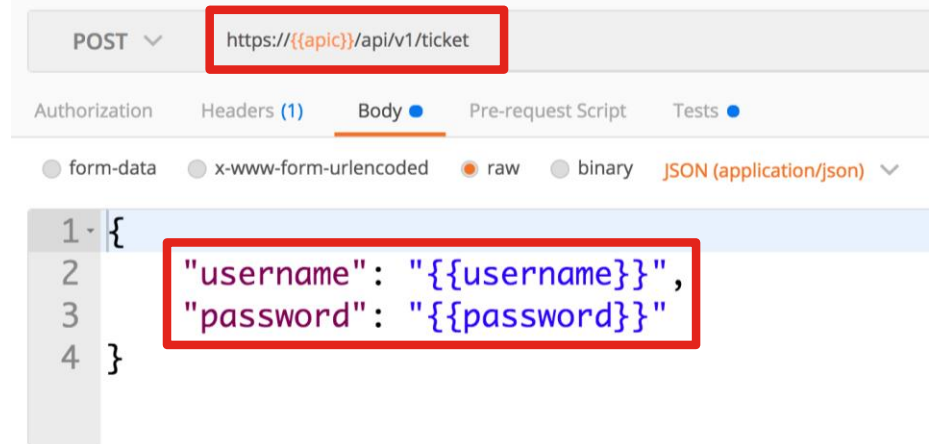
- Never good to hardcode details
- What if you want to connect to different host?
- What if credentials change?



The screenshot displays a REST client interface for a POST request. The URL bar shows `https://sandboxapicem.cisco.com/api/v1/ticket`, which is highlighted with a red box. Below the URL bar, the 'Body' tab is selected, and the request body is shown as a JSON object: `{ "username": "devnetuser", "password": "Cisco123!" }`. The JSON body is also highlighted with a red box. The interface includes tabs for 'Authorization', 'Headers (1)', 'Body', 'Pre-request Script', and 'Tests'. At the bottom, there are radio buttons for 'form-data', 'x-www-form-urlencoded', 'raw' (selected), and 'binary', along with a dropdown menu for 'JSON (application/json)'.

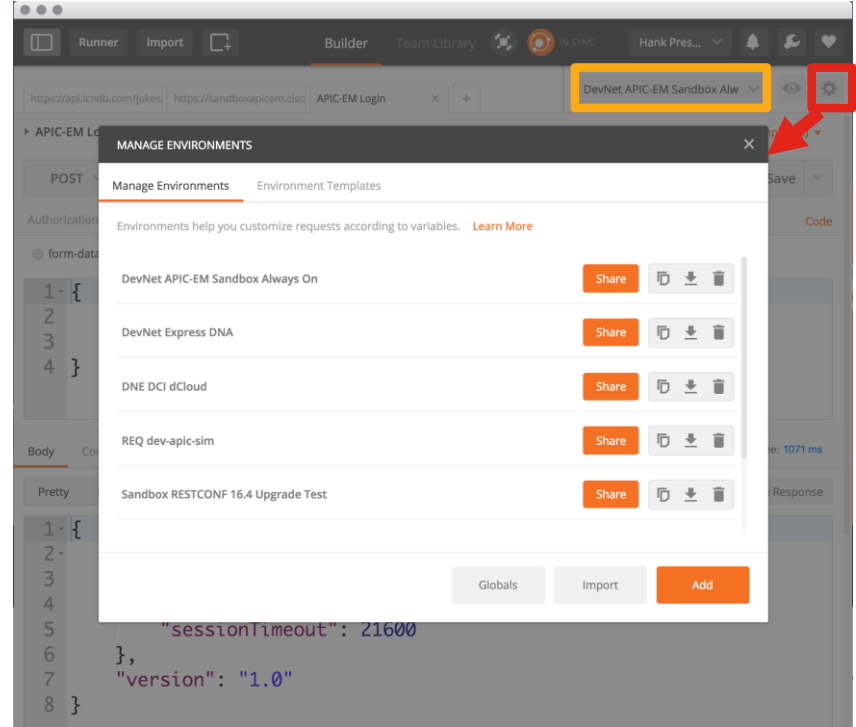
Variables Make Requests Reusable and Flexible

- Variables References
 - {{apic}}
 - {{username}}
 - {{password}}



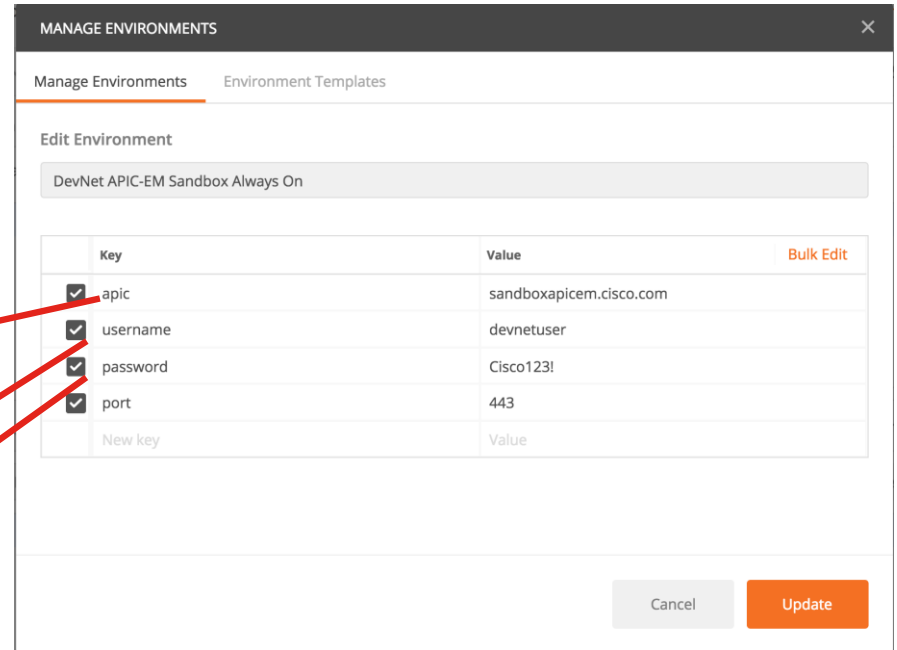
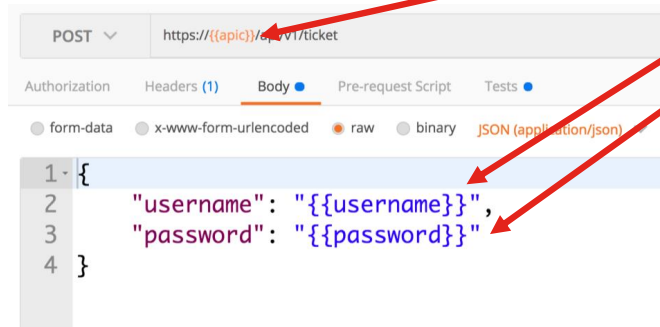
Managing Environments

- Create any number of environments needed
- Change between environments with drop down list



Managing Environments

- Add as many variables as needed
- Reference anywhere with `{{variable name}}` syntax



Setting Environment Variables Dynamically

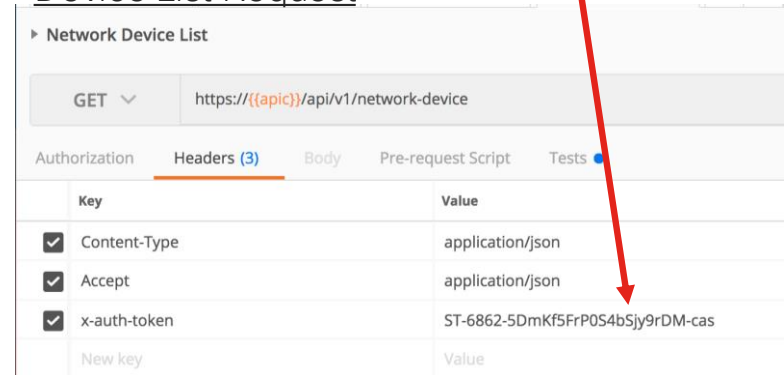
- What about when info from one request is needed in another?
- Manually copying/pasting slow and error prone
- Manually updating environment variables is slow and awkward

Login Response



```
1 {
2   "response": {
3     "serviceTicket": "ST-6862-5DmKf5FrP0S4bSjy9rDM-cas",
4     "idleTimeout": 1800,
5     "sessionTimeout": 21600
6   },
7   "version": "1.0"
8 }
```

Device List Request



Network Device List

GET `https://(apic)/api/v1/network-device`

Key	Value
<input checked="" type="checkbox"/> Content-Type	application/json
<input checked="" type="checkbox"/> Accept	application/json
<input checked="" type="checkbox"/> x-auth-token	ST-6862-5DmKf5FrP0S4bSjy9rDM-cas
New key	Value

“Tests” Enable Dynamic Environment Variables

- Each API Request offers both pre and post actions
- Pre -> Pre-request Script
- Post -> Tests
- Written in JavaScript

The screenshot shows the Postman interface for an API request. At the top, a modal window titled "DevNet APIC-EM Sandbox Always On" displays environment variables: apic (sandboxapicem.cisco.com), username (devnetuser), password (Cisco123!), port (443), and token (ST-6875-uGQzfhTzdLlg0bsfyBVZ-cas). The token value is highlighted with a red box. Below this, the main interface shows a POST request to "https://(apic)/api/v1/ticket". The "Tests" tab is selected and highlighted with a red box. The test script contains the following JavaScript code:

```
1 var jsonData = JSON.parse(responseBody);
2 postman.setEnvironmentVariable("token",
3                               jsonData.response.serviceTicket);
4
```

Red arrows point from the "token" variable in the test script to the "token" environment variable in the modal window, and from the "jsonData.response.serviceTicket" property in the test script to the "serviceTicket" property in the response body below.

The response body is shown as a JSON object:

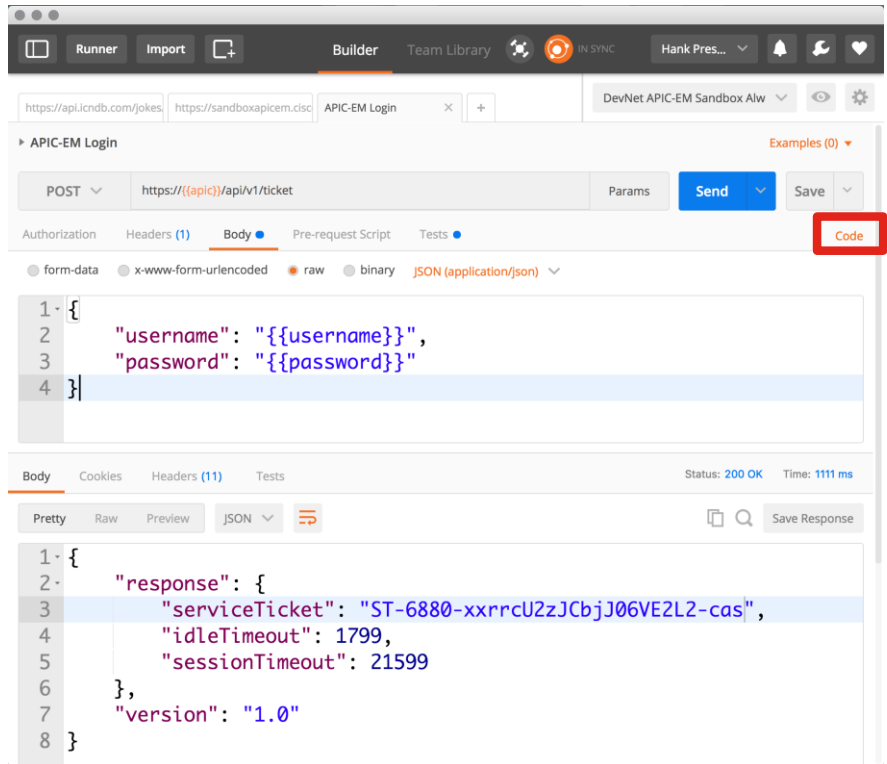
```
{
  "response": {
    "serviceTicket": "ST-6862-5DmKf5FrP0S4bSjy9rDM-cas",
    "idleTimeout": 1800,
    "sessionTimeout": 21600
  },
  "version": "1.0"
}
```

The text "Response Body" is written to the right of the JSON object.

Postman to Code!

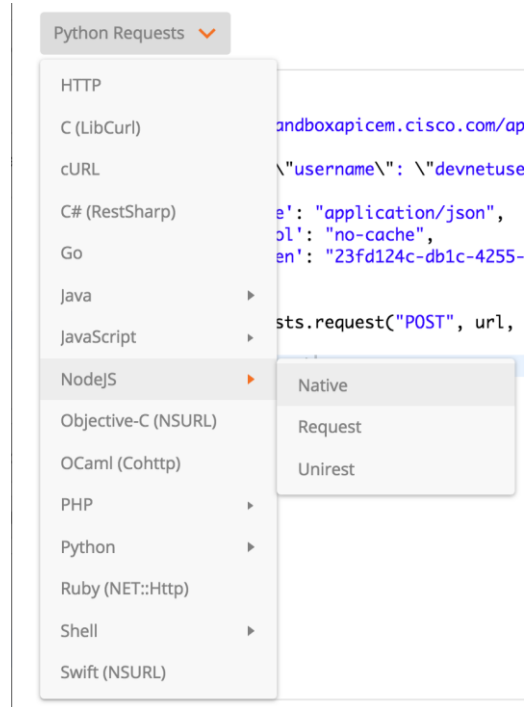
You'll eventually want to write some code...

- Postman great for testing and validating APIs
- But it's about atomic actions
- Business Logic, stringing APIs together, etc all need code
- Jumpstart with auto-generated code by Postman



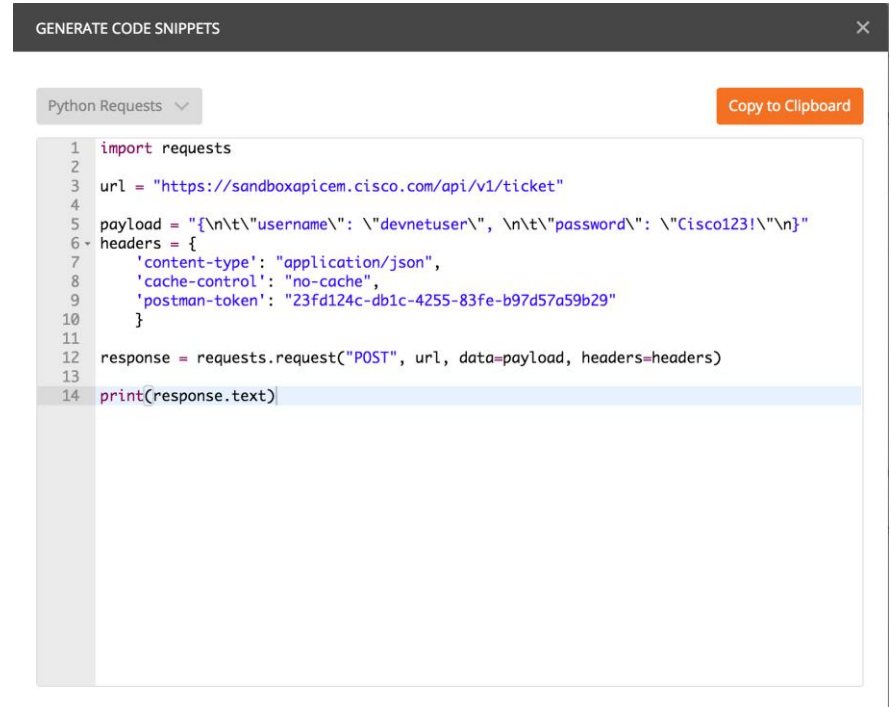
You'll eventually want to write some code...

- Many, many options for languages available



Full API Request to Code!

- Headers, payload data, and URI all included
- Environment variables are translated
- Great starting point, but expect to edit and update



The screenshot shows a web interface for generating code snippets. At the top, there is a dark header with the text "GENERATE CODE SNIPPETS" and a close button. Below the header, there is a dropdown menu set to "Python Requests" and a "Copy to Clipboard" button. The main area contains a code editor with the following Python code:

```
1 import requests
2
3 url = "https://sandboxapicem.cisco.com/api/v1/ticket"
4
5 payload = "{\n\t\"username\": \"devnetuser\", \n\t\"password\": \"Cisco123!\n}"
6 headers = {
7     'content-type': "application/json",
8     'cache-control': "no-cache",
9     'postman-token': "23fd124c-db1c-4255-83fe-b97d57a59b29"
10 }
11
12 response = requests.request("POST", url, data=payload, headers=headers)
13
14 print(response.text)
```

Demo

The screenshot shows a REST client application interface. At the top, there's a navigation bar with 'New', 'Import', 'Runner', and 'My Workspace' options. Below this is a search bar and a 'Collections' tab. The left sidebar shows a tree view of collections, including '01-DNAC-Sandbox' with 72 requests, and sub-collections like '1.Ticket' and '2.Network Device'. The main area displays a GET request to 'https://{{dnac}}:{{port}}/api/v1/network-device/'. The response is shown in JSON format, indicating a successful GET (200 OK) with a response body containing a list of network device details.

GET `https://{{dnac}}:{{port}}/api/v1/network-device/` Send Save

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (9) Test Results Status: 200 OK Time: 241 ms Size: 5.46 KB Save Download

Pretty Raw Preview JSON ≡

```
1 {
2   "response": [
3     {
4       "family": "Routers",
5       "errorCode": null,
6       "type": "Cisco ASR 1001-X Router",
7       "locationName": null,
8       "location": null,
9       "role": "BORDER ROUTER",
10      "lastUpdateTime": "1546744110500",
11      "macAddress": "00:c8:8b:80:bb:00",
12      "hostname": "asr1001-x.microsoft.com",
13      "serialNumber": "FXS193201SE",
14      "softwareVersion": "16.3.2",
15      "upTime": "50 days, 8:41:24.10",
16      "lastUpdated": "2019-01-06 03:08:30",
17      "tagCount": "0",
18      "inventoryStatusDetail": "<status><general code=\\"SUCCESS\\"/></status>",
19      "errorDescription": null,
20      "softwareType": "IOS-XE",
21      "collectionInterval": "Global Default",
22      "roleSource": "AUTO",
23      "bootDateTime": "2018-10-14 16:59:30".
```



Agenda

- Introduction to APIs & Data Formats
- REST APIs
- APIs -> Postman -> Code!
- **Summary and Close**

What have we learned?

- What is an API
- A brief look at working with REST APIs
- Using Postman as a way to work with APIs

Resources and Starting Points

- DNA-Center Sandbox
<https://sandboxdnac.cisco.com> - username=devnetuser, password=Cisco123!
- APIC-EM Sandbox (deprecated, but still useful to test API calls)
<https://sandboxapicem.cisco.com> - username=devnetuser, password=Cisco123!
- Cisco DevNet Postman Collections -
https://github.com/CiscoDevNet/netprog_basics/tree/master/postman_config
- Other APIs to try
 - The Internet Chuck Norris Database - <http://www.icndb.com/api/>
 - Deck of Cards API - <https://deckofcardsapi.com>
 - Free Public API repository - <https://github.com/toddmotto/public-apis/blob/master/README.md>

CiscoLive

SOLDGT-1000 – Cisco DNA Center Platform

DEVNET-1897 – Coding 1001 - Intro to APIs and REST

DEVNET-2877 – Exploring Cisco DNA-C as a Platform

Questions?



Want to learn more about the DevNet community?

The image shows a browser window displaying the Cisco DevNet website. A white box highlights the URL `http://bit.ly/devnetcommunity` in the address bar, with a red arrow and the number '1' pointing to it. In the top right corner of the website, there are buttons for 'SIGN UP FREE' and 'LOG IN', with a red arrow and the number '2' pointing to the 'LOG IN' button. A modal window titled 'DEVNET' is open in the center, listing several login options: 'Login with Github', 'Login with Google', 'Login with Facebook', 'Login with a Cisco ID', 'Login with Webex Teams', and 'Login with Cisco NetAcad'. The 'Login with Cisco NetAcad' option is highlighted with a red dashed border, and a red arrow with the number '3' points to it. The background of the website shows a banner for 'Cisco Networking Academy' with the text 'Explore DevNet tools and resources to support yourself for the future of automation, programming, and the Internet of Things. Gain exposure to software development and industry transformation.'

Make sure to login with your Cisco NetAcad Login at step 3!

Next DevNet Webinar: 13 February 2019

Date Topic

- Oct'18 Networking with Programmability is Easy
- Oct'18 A Network Engineer in the Programmable Age
- Nov'18 Software Defined Networking and Controllers
- Jan'19 Adding API Skills to Your Networking Toolbox
- ➔ Feb'19 The New Toolbox of a Networking Engineer
- Mar'19 Program Networking Devices using their APIs
- Apr'19 Before, During, and After a Security Attack
- May'19 Play with Linux & Python on Networking Devices
- Jun'19 Automate your Network with a Bot



All Series Details can be Found @ <http://bit.ly/devnet2>

