



Programación de Redes – Becas Digitaliza - 2019

PUE – ITC – Formación de Instructores

Sesión 7 – APIs, JSON y XML

Iván Lago - Técnico Cisco Networking Academy ASC/ITC
PUE - ITC/ASC/CA

- APIC-EM
- APIs
- REST APIs
- Data Format & Parsing
 - XML
 - JSON

APIC-EM



The Cisco Application Policy Infrastructure Controller Enterprise Module (APIC-EM) is an SDN controller. It is a software device that runs on servers and concentrates the control plane of the physical network infrastructure.

Cisco APIC-EM SDN controller communicates with the Physical Topology using standard Southbound API protocols such as SNMP, SSH and Telnet rather than a protocol like OpenFlow.

Easy abstraction of the network using a standard **REST API interface**. Therefore, it removes the need for the network staff to configure and manage every single networking device (routers, switches, access points, wireless controllers, etc.) one by one.

APIC-EM: log in

Virtualized APIC-EM Controllers are available in several DevNet Sandboxes:

Always On, NetAcad instances

- For NetAcad users only
- <https://DevNetSBX-NetAcad-APICEM-3.cisco.com>
 - **User:** *devnetuser* **PW:** Xj3BDqbU

Always on, public instance

- For to all DevNet users
- <https://SandBoxAPICEM.cisco.com>
 - **User:** *devnetuser* **PW:** *Cisco123!*



APIC-EM: home page (I)

Expand Navigation Bar

Services

Applications


API documentation










The screenshot shows the APIC-EM home page. The left sidebar contains a navigation menu with icons for Home, Discovery, Device Inventory, Host Inventory, Topology, IWAN, Path Trace, Network Plug and Play, and EasyQoS. The main content area displays a 'DEVICE INVENTORY' donut chart showing 13 devices: 1 Managed (green), 11 In-Progress (yellow), and 1 Collection Failure (red). Other sections include 'DISCOVERY - UNREACHABLE DEVICES' (4 Devices), 'BRANCH SITES' (with a link to the IWAN app), 'NETWORK PLUG AND PLAY PROJECTS' (4 projects: 1 Provisioned, 3 Pre-Provisioned, 0 In-Progress, 0 Failed), 'EASYQOS SCOPES', and 'PATH TRACE' (All Path traces are successful).

Status	Count
Managed	1
In-Progress	11
Collection Failure	1
Total	13

Status	Count
Provisioned	1
Pre-Provisioned	3
In-Progress	0
Failed	0
Total	4

APIC-EM: API (I)

 APIC - Enterprise Module / Swagger



Available APIs
[File](#)
[Flow Analysis](#)
[Grouping](#)
[IP Geolocation](#)
[IP Pool Manager](#)
[Identity-Manager](#)
Inventory
[Network Discovery](#)
[Network Plug and Play](#)
[PKI Broker Service](#)
[Policy Administration](#)
[Role Based Access Control](#)
[Scheduler](#)
[Task](#)
[Topology](#)
[Visibility](#)

Inventory

APIC-EM Service API based on the Swagger™ 1.2 specification

[Terms of service](#)
[Cisco DevNet](#)

device-credential : Device Credential API	Show/Hide	List Operations	Expand Operations	Raw
discovery : Discovery API	Show/Hide	List Operations	Expand Operations	Raw
host : host API	Show/Hide	List Operations	Expand Operations	Raw
GET /host	Retrieve hosts			
GET /host/count	Gives total number of hosts			
GET /host/{id}	Retrieves host based on id			
interface : Interface API	Show/Hide	List Operations	Expand Operations	Raw
license : license API	Show/Hide	List Operations	Expand Operations	Raw
location : Location API	Show/Hide	List Operations	Expand Operations	Raw
network-device : network-device API	Show/Hide	List Operations	Expand Operations	Raw
network-device-config : Network Device Configuration API	Show/Hide	List Operations	Expand Operations	Raw
segment : Segment API. Currently, wireless type is supported	Show/Hide	List Operations	Expand Operations	Raw
tag : Tag API	Show/Hide	List Operations	Expand Operations	Raw
vlan : Vlan API	Show/Hide	List Operations	Expand Operations	Raw

[BASE URL: <https://devnetsbx-netacad-apicem-3.cisco.com/api/v1/api-docs/inventory-manager> , API VERSION: 1.0]

APIC-EM: API (II)

host : host API

[Show/Hide](#)[List Operations](#)[Expand Operations](#)[Raw](#)

GET

[/host](#)[Retrieve hosts](#)

Implementation Notes

Get Hosts

Response Class

Model | [Model Schema](#)

HostListResult {

version (string, optional),

response (array[HostDTO], optional)

}

HostDTO {

hostName (string, optional): Name of the host,

source (string): Source from which the host gets collected. Available option:200 for inventory collection and 300 for trap based data collection,

id (string): Id of the host,

vlanId (string, optional): Vlan Id of the host,

subType (string, optional) = ['UNKNOWN' or 'IP_PHONE' or 'TELEPRESENCE' or 'VIDEO_SURVEILLANCE_IP_CAMERA' or 'VIDEO_ENDPOINT'],

lastUpdated (string): Time when the host info last got updated,

avgUpdateFrequency (string): Frequency in which host info gets updated,

connectedAPMacAddress (string, optional): Mac address of the AP to which wireless host gets connected,

connectedAPName (string, optional): Name of the AP to which wireless host gets connected,

connectedInterfaceId (string, optional): Id of the interface to which host gets connected,

connectedInterfaceName (string, optional): Name of the interface to which host gets connected,

connectedNetworkDeviceId (string): Id of the network device to which host gets connected,

connectedNetworkDeviceIpAddress (string): Ip address of the network device to which host gets connected,

hostIp (string): Ip address of the host,

hostMac (string): Mac address of the host,

hostType (string): Type of the host. Available options are: Wired, Wireless,

pointOfAttachment (string, optional): Id of the Host's Point of attachment network device (wlc). Based on mobility,

pointOfPresence (string, optional): Id of the Host's Point of presence network device (wlc). Based on mobility,

attributeInfo (object, optional)

}

Response Content Type: application/json

APIC-EM: API (III)

Parameters

Parameter	Value	Description	Parameter Type	Data Type
limit	<input type="text"/>	limit	query	string
offset	<input type="text"/>	offset	query	string
sortBy	<input type="text"/>	sortBy	query	string
order	<input type="text"/>	order	query	string
hostName	<input type="text"/>	hostName	query	List
hostMac	<input type="text"/>	hostMac	query	List
hostType	<input type="text"/>	hostType	query	List
connectedInterfaceName	<input type="text"/>	connectedInterfaceName	query	List
hostIp	<input type="text"/>	hostIp	query	List
connectedNetworkDeviceIpAddress	<input type="text"/>	connectedNetworkDeviceIpAddress	query	List
subType	<input type="text"/>	Available values: 'UNKNOWN' or 'IP_PHONE' or 'TELEPRESENCE' or 'VIDEO_SURVEILLANCE_IP_CAMERA' or 'VIDEO_ENDPOINT'. Only exact match filtering supported on this field	query	List
filterOperation	<input type="text"/>	startswith/contains/endswith	query	string

Error Status Codes

HTTP Status Code	Reason
200	This Request is OK
403	This user is Forbidden Access to this Resource
401	Not Authorized Yet, Credentials to be supplied
404	No Resource Found

Try it out!

API

Application Programming Interface (API)

- An API allows one piece of software talk to another.
- An API is analogous to a power outlet.
- Without a power outlet, what would you have to do to power your laptop?
 - Open the wall
 - Unsheathe wires
 - Splice wires together
 - Understand all the wires in the wall
- An API defines how a programmer can write a piece of software to extend an existing application's features or even build entirely new applications.



API Example (I)

- Restaurant Recommendation App
 - Returns a list of relevant restaurants in the area
 - Integrates a third-party API to provide map functionality
 - The map API enforces a specification of an interface



API Example (II)

- Edit a sheet
 - Whoever
 - Wherever
 - Whatever



Some known APIs



Google
Sheets



Google Maps

API Example: documentation



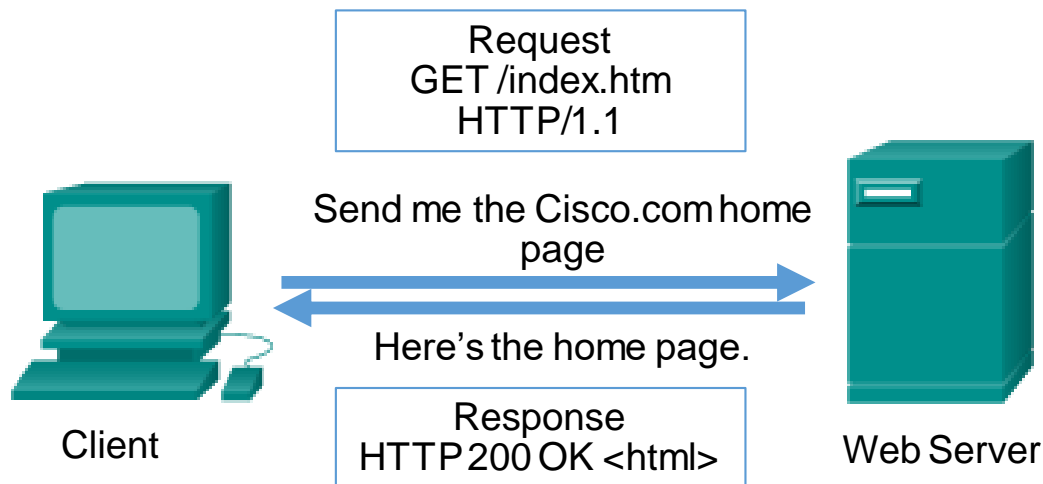
MapQuest Developer:

- <https://developer.mapquest.com/documentation/>

REST API

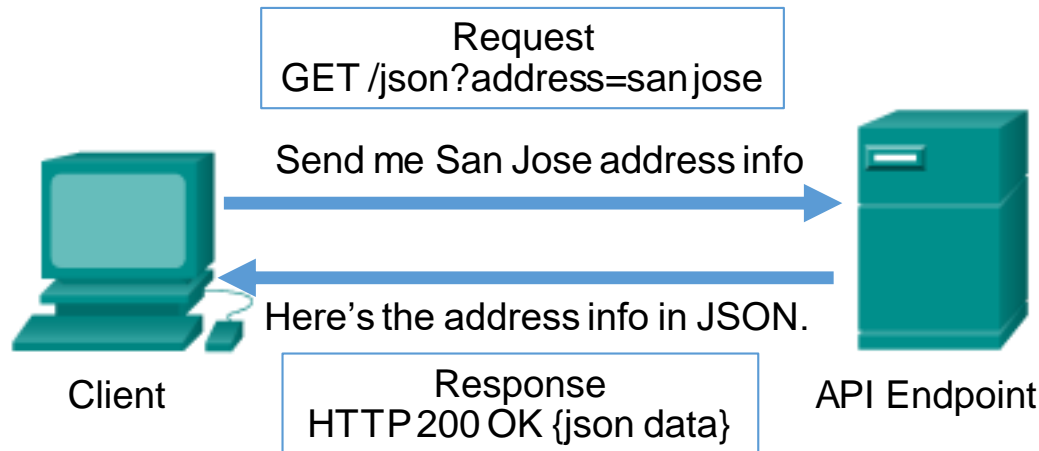
Web Services Interface using HTTP

- Web browsers use Hypertext Transfer Protocol (HTTP) to request (GET) a web page.
- If successfully requested (HTTP status code 200), web servers respond to GET requests with a Hypertext Markup Language (HTML) coded web page.



RESTful API using HTTP

- Representation State Transfer (REST) APIs use HTTP to interface with RESTful services.
- The HTTP request asks for JavaScript Object Notation (JSON) formatted data.
- If successfully formatted according to the API documentation, the server will respond with JSON data.



Anatomy of a RESTful Request

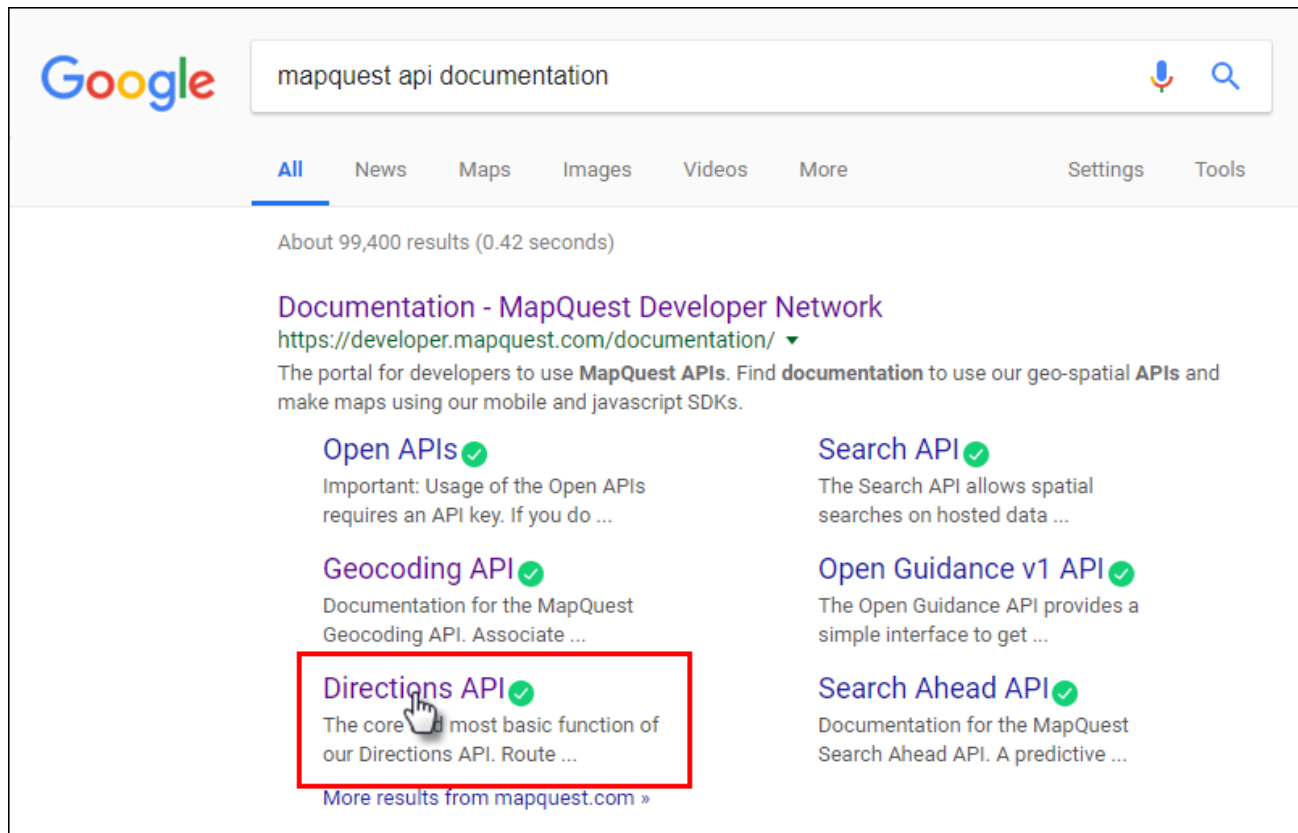
<https://www.mapquestapi.com/directions/v2/route?outFormat=json&key=KEY&...>

The diagram illustrates the components of the URL `https://www.mapquestapi.com/directions/v2/route?outFormat=json&key=KEY&...` using blue curly brackets underneath. The components are: **API Server** (the domain and path `https://www.mapquestapi.com/directions/v2/route`), **Resources** (the path segment `/route`), **Format** (the query parameter `?outFormat=json`), and **Parameters** (the query parameter `&key=KEY&...`).

- **API Server:** The URL for the server that answers REST requests
- **Resources:** Specifies the API that is being requested.
- **Format:** Usually JSON or XML
- **Parameters:** Specifies what data is being requested

API Documentation

To successfully construct an API request, you must follow the API documentation. You can quickly find API documentation using an Internet search.



API Documentation

- The API documentation will specify...
 - The request **format** (JSON, XML, or text)
 - The request **parameters**
 - The response fields

MapQuest Developer:

- <https://developer.mapquest.com/documentation/>

Directions API

GET Route

Resource URL

`http://www.mapquestapi.com/directions/v2/route`

Resource Information

Response Formats	JSON, XML
Authentication	Yes (Requires Key)
Rate Limited	Yes

Request Parameters

Request Parameter	Description	Required?
key String	The API Key, which is needed to make requests to MapQuest services.	Yes

What is so great about REST*?



Cisco APIC-EM REST APIs

- Hosts
- Devices
- Users
- + more

How does this work?

- Easy to use:
- In mobile apps
 - In console apps
 - In web apps



Representational state transfer (REST) is a standardized way of allowing systems on the Internet to interact with one another. It uses the standard HTTP methods and stateless transport to POST or PUT data and requests to an API and to GET responses from the API. REST is simple, standard, and allows any device that can access the web to interact with web software resources by accessing a URI.

REST APIs (I)

- Use HTTP protocol methods and transport
- API **endpoints** exist as server processes that are accessed through URIs
- Webpages present data and functionality in human-machine interaction driven by a user.
- APIs present data and functionality in machine-machine interactions driven by software.

Directory of Public APIs:

<https://www.programmableweb.com/apis/directory>

REST APIs (II)

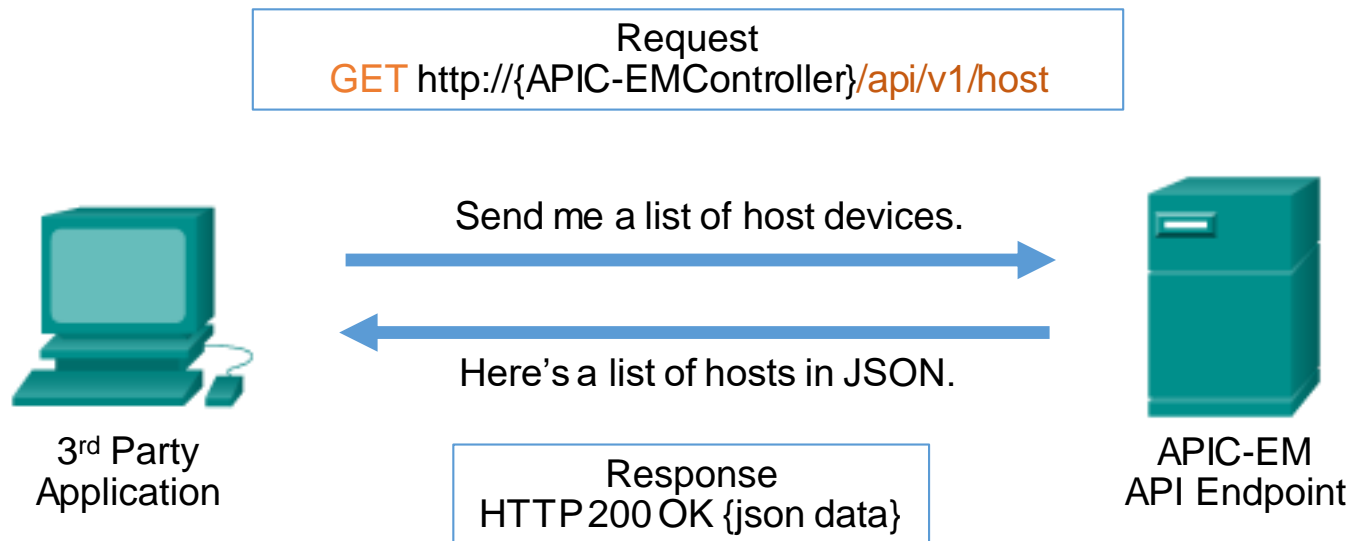
APIs enable vendors to provide data services to other vendors across the web.

- Google provides APIs to other vendors. Amazon can access users' search history on Google and identify products that a user may have searched for. These searches can then inform Amazon as to which product offerings it should push to a specific user based on their interests. All of this is done in software. Amazon has code that accesses the Google API, analyzes the search history for thousands, or millions, of users, and identifies offerings that can be pushed to individual users, perhaps on Facebook, based on their Facebook's API.

These API services allow a business to monetize its data by offering it to other businesses in a way that it can easily be utilized. Rather than a company buying data from another company and storing it, the data can be accessed directly from the owner's storage resources.

REST APIs (III)

While individual users request webpages with browsers, and web servers respond by returning text, graphics, and media based on the pages requested, APIs enable machines to talk to machines using software in much the same way. Many types of data are returned, but most often it is in the form of text-based data formats like XML or JavaScript object notation (JSON).



How does this work? (I)

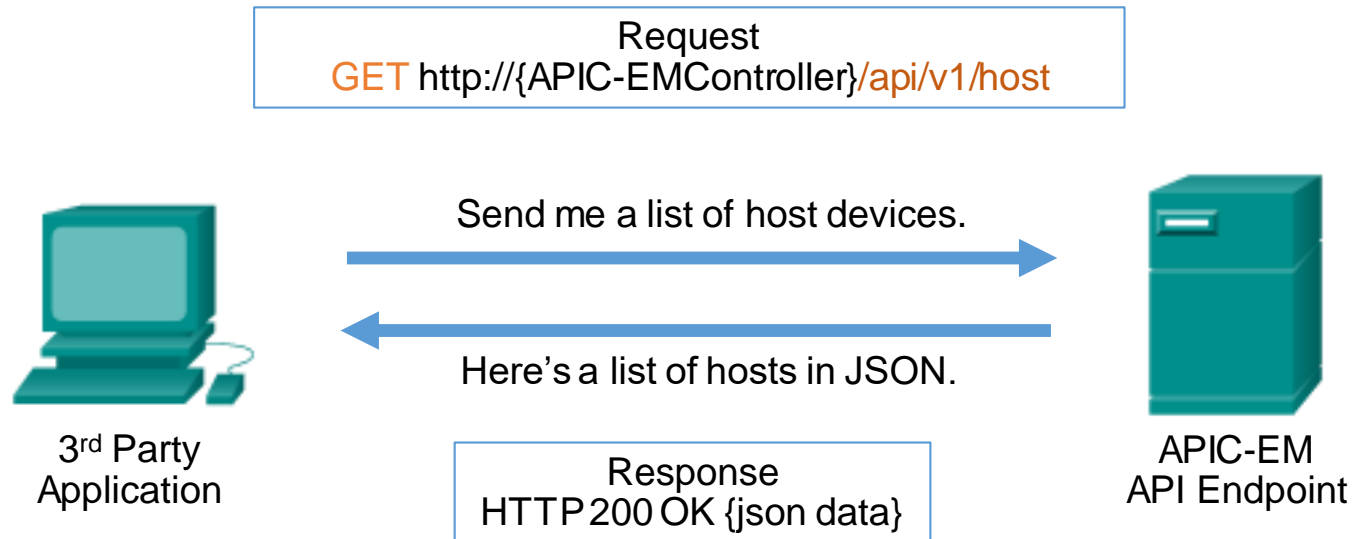
An application or program is used to issue requests to a server that is accepting HTTP requests. This API has different specialized functionalities depending on the endpoint, or URI, that is used.

For example, an endpoint may return the list of host devices in the network. The endpoints can be identified by paths that are appended to the URL. Each endpoint may have different requirements for the format of the REST request.

These requirements are available in the API documentation that is provided to developers. Many APIs are freely available for experimentation and limited to use for free. Many require simple registration and the use of a API key that identifies the user account that is accessing the API endpoint. This is used for accounting and security purposes.

How does this work? (II)

In the example, a generic APIC-EM URL is shown with a path to an endpoint called `/host`. The host endpoint provides an inventory of hosts on the network that is attached to the APIC-EM. If the GET request is correctly formatted and authenticated, the API will return detailed information about the hosts in the form JSON-formatted data. Note that the response includes a standard HTTP status code that indicates that transaction was successful.



Anatomy of a REST Request

REST requests require the following elements (requirements may differ depending on the API):

Method

- GET (retrieve), POST (create), PUT (update), DELETE (remove)

URL

- Example: `http://{APIC-EMController}/api/v1/host`

Authentication

- Basic HTTP, OAuth, none, Custom

Custom Headers

- HTTP Headers
- Example: `Content-Type: application/json`

Request Body

- JSON or XML containing data needed to complete request

Request anatomy: URL & HTTP methods

REST requests require the HTTP method and the URL for the API endpoint.

- The GET method can be used to retrieve some information, such as a list of IOS versions that are being used in the network infrastructure if the request is addressed to the appropriate API endpoint.
- The POST method is used to create new objects, such as a new authentication ticket or a new QoS policy.
- The PUT method is used to update existing objects, such as updating the QoS policy.
- The DELETE method is used to remove existing objects.

Request anatomy: Authentication, Custom Headers, and Request Body

Some requests require some sort of authentication. Custom HTTP headers are also used to identify the content type for the request body and other parameters. The body of the request may also be required to carry details of the request. APIs differ in their requirements.

The first step in working with an API is to review the documentation in order to understand how to create API calls and process the data that is returned.

HTTP Response

- HTTP Status Codes

- Status codes: <http://www.w3.org/Protocols/HTTP/HTRESP.html>

- Headers

- Body

- JSON
 - XML



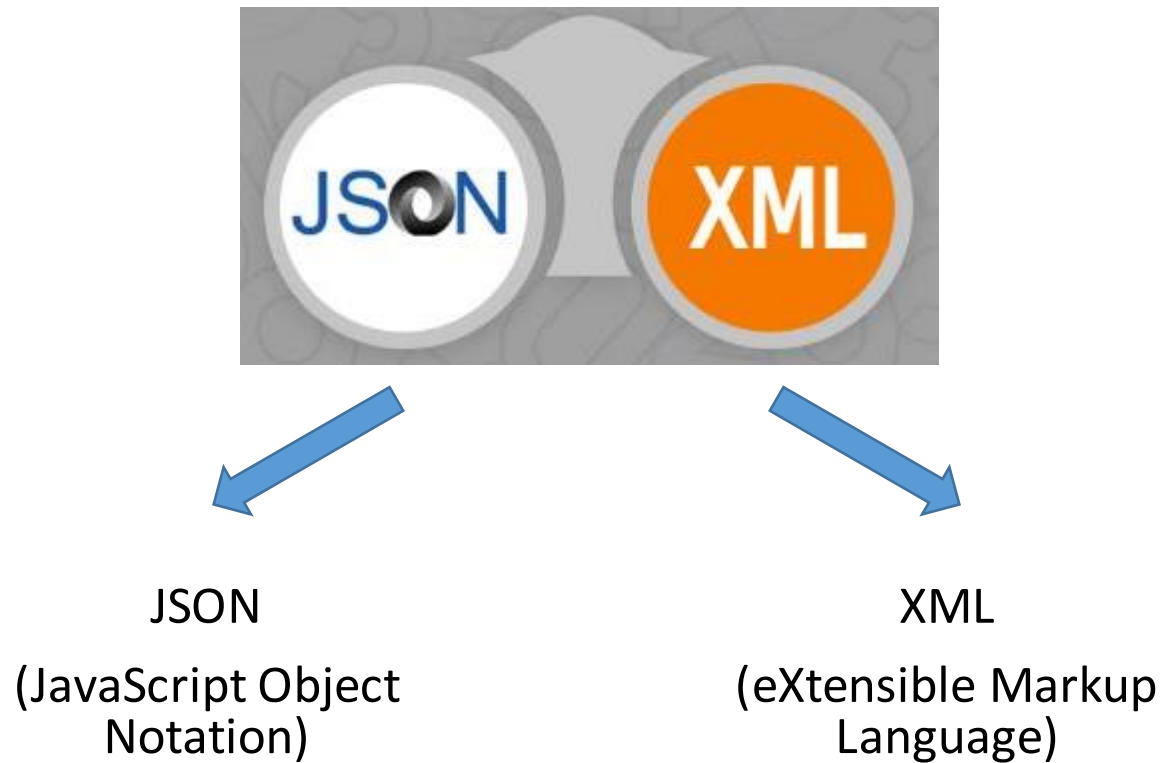
Postman example: shows an example response body. The response status is 201 and JSON-formatted data is contained in the body of the response.

HTTP Response: Codes, Headers and Body

- HTTP status codes are returned for each request:
 - 200 or 201 indicates a successful request.
 - 404 resource not found error can indicate that the URI that was used to make the request contains an error, or that the path to the endpoint has changed.
 - 401 or 403 errors indicate that authentication requirements have not been met.
 - 500 indicates internal error
 - Coders extract the status code from the response and write code to take action based on it.
- Important information may be returned in the header.
- Large amounts of data are usually returned in the body of the response either as JSON or XML.

DATA FORMAT & PARSING

Data Formatting



JSON: what it is?

JavaScript Object Notation (JSON) is a format for storing and exchanging text between a server and a client application.

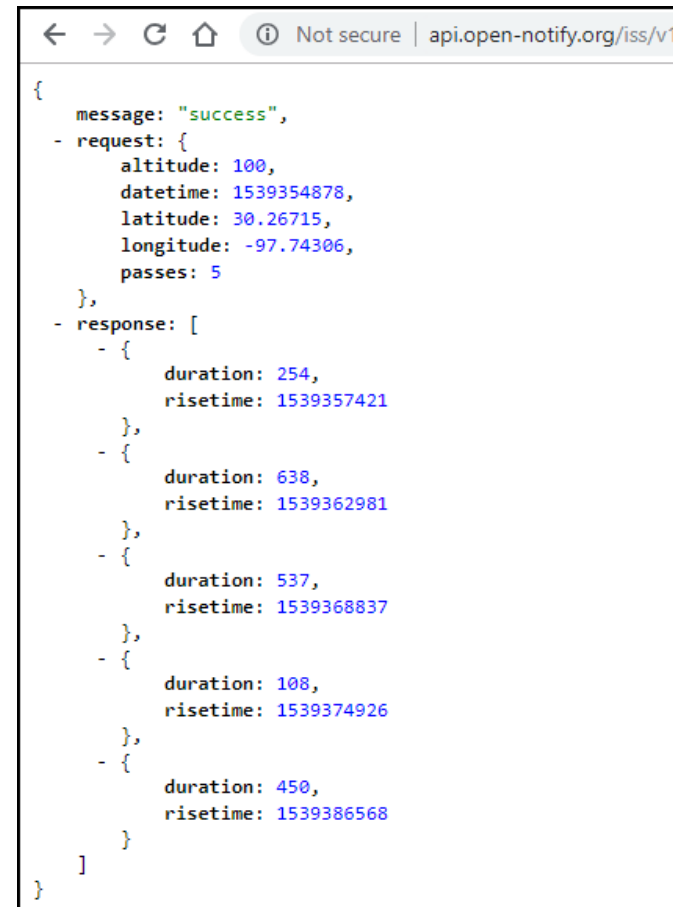
- Easy to read and write.
- Popular format that web services and APIs use to provide public data because it is easy to parse and can be used with most modern programming languages (including Python).
- Objects are indicated by curly braces and resemble Python dictionaries.
- JSON arrays are held in square brackets and resemble Python lists.

We will refer to the JSON structures using the familiar Python terms. However, JSON data is usually converted to Python data structures before being used by Python apps.

JSON Response Data (I)

<http://api.open-notify.org/iss/v1/?lat=30.26715&lon=-97.74306>

- Enter the above URL in your browser to get a JSON response.
- JSON data looks a lot like a Python dictionary (JSON vs data without processing)
- The third item in the dictionary is the **response** element.
- Inside the list are five dictionaries.

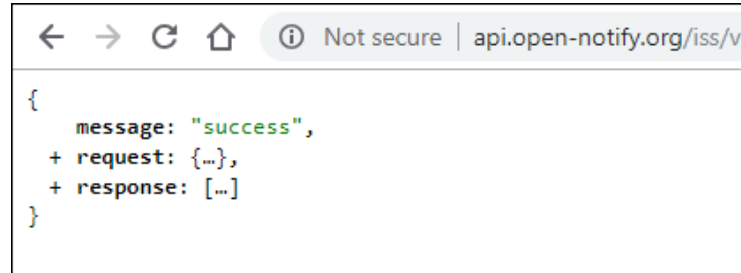


The screenshot shows a web browser window with the address bar displaying the URL `api.open-notify.org/iss/v1`. The page content shows a JSON response. The browser's address bar indicates "Not secure". The JSON data is as follows:

```
{
  message: "success",
  - request: {
    altitude: 100,
    datetime: 1539354878,
    latitude: 30.26715,
    longitude: -97.74306,
    passes: 5
  },
  - response: [
    - {
      duration: 254,
      risetime: 1539357421
    },
    - {
      duration: 638,
      risetime: 1539362981
    },
    - {
      duration: 537,
      risetime: 1539368837
    },
    - {
      duration: 108,
      risetime: 1539374926
    },
    - {
      duration: 450,
      risetime: 1539386568
    }
  ]
}
```

JSON Response Data (II)

<http://api.open-notify.org/iss/v1/?lat=30.26715&lon=-97.74306>

A screenshot of a web browser window. The address bar shows the URL 'http://api.open-notify.org/iss/v1/?lat=30.26715&lon=-97.74306'. The page content displays a JSON object: { message: "success", + request: {...}, + response: [...] }. The JSON is formatted with syntax highlighting: 'success' is in quotes, and the keys are in a light blue font. The browser's status bar at the bottom indicates 'Not secure' and the full URL.

```
{
  message: "success",
  + request: {...},
  + response: [...]
}
```

Collapse components of the JSON to better view its structure. In the example, JSON returns three root elements: **message**, **request**, and **response**

- The **message** root returns the HTTP status code.
- The **request** root shows default parameters and the parameters entered in the request.
- The **response** root typically returns an array with relevant responses.

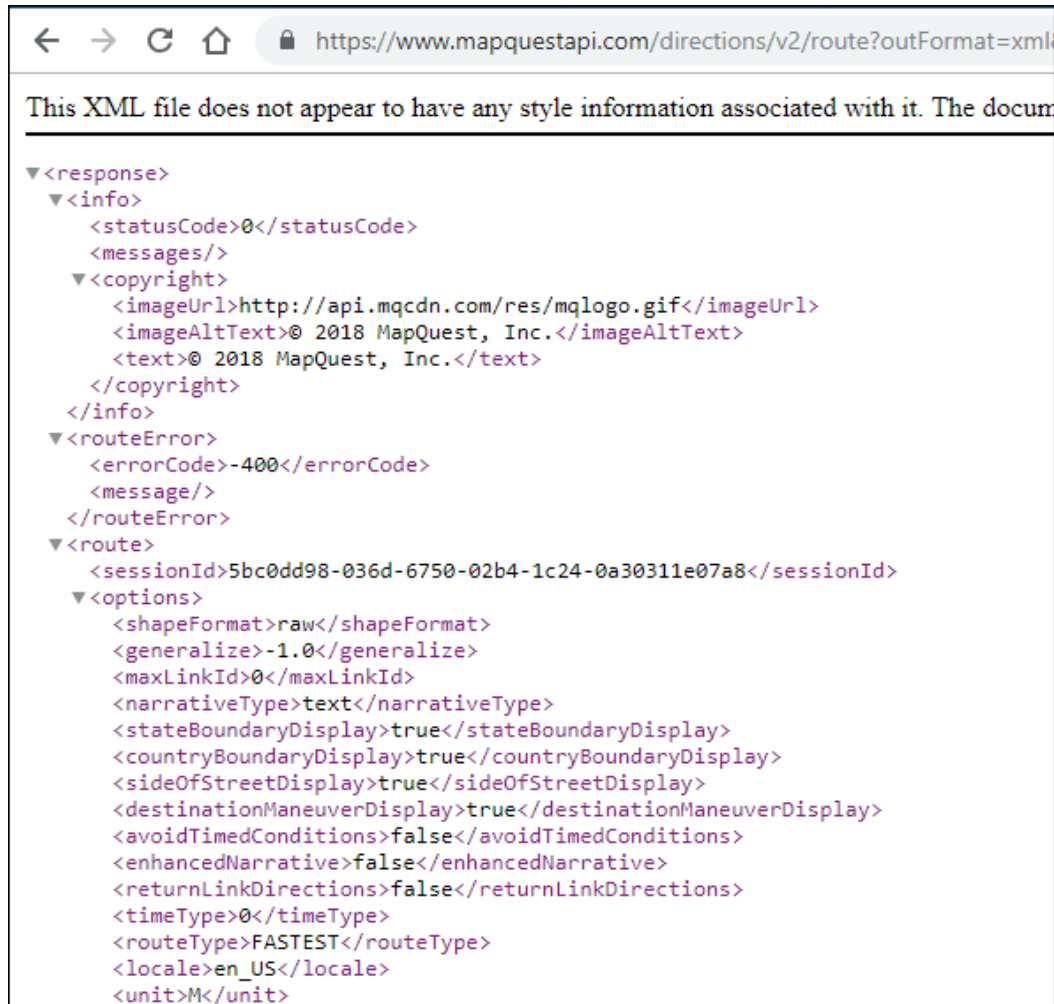
Pycharm example

```
import requests
import time

url = 'http://api.open-notify.org/iss/v1/?lat=30.26715&lon=-97.74306'
json_data = requests.get(url).json()
print(json_data)
epoch = json_data['response'][0]['risetime']
next_pass = time.strftime("%a, %d %b %Y %H:%M:%S %Z", time.localtime(epoch))
print("The next ISS pass will be: " + (next_pass))
```

XML Response Data (I)

<https://www.mapquestapi.com/>



```
<?xml version="1.0" encoding="UTF-8" />
<response>
  <info>
    <statusCode>0</statusCode>
    <messages/>
    <copyright>
      <imageUrl>http://api.mqcdn.com/res/mqlogo.gif</imageUrl>
      <imageAltText>© 2018 MapQuest, Inc.</imageAltText>
      <text>© 2018 MapQuest, Inc.</text>
    </copyright>
  </info>
  <routeError>
    <errorCode>-400</errorCode>
    <message/>
  </routeError>
  <route>
    <sessionId>5bc0dd98-036d-6750-02b4-1c24-0a30311e07a8</sessionId>
    <options>
      <shapeFormat>raw</shapeFormat>
      <generalize>-1.0</generalize>
      <maxLinkId>0</maxLinkId>
      <narrativeType>text</narrativeType>
      <stateBoundaryDisplay>true</stateBoundaryDisplay>
      <countryBoundaryDisplay>true</countryBoundaryDisplay>
      <sideOfStreetDisplay>true</sideOfStreetDisplay>
      <destinationManeuverDisplay>true</destinationManeuverDisplay>
      <avoidTimedConditions>false</avoidTimedConditions>
      <enhancedNarrative>false</enhancedNarrative>
      <returnLinkDirections>false</returnLinkDirections>
      <timeType>0</timeType>
      <routeType>FASTEST</routeType>
      <locale>en_US</locale>
      <unit>M</unit>
    </options>
  </route>
</response>
```

XML Response Data (II)

- Example:
 - https://www.mapquestapi.com/directions/v2/route?outFormat=xml&key=your_key&from=Moscow&to=Beijing
- Extensible Markup Language (XML) extends the functionality of HTML allowing web programmers to construct custom tags.
- To get XML data instead of JSON from the MapQuest API, add the **outFormat=xml** to the URL structure.
- You can see in the XML response that the embedded dictionaries have the same basic structure as JSON.

Gracias por vuestra atención



pue

IMPULSANDO EL CONOCIMIENTO
TIC CUALIFICADO

Iván Lago - Técnico Cisco Networking Academy ASC/ITC
PUE - ITC/ASC/CA
Área de Proyectos de Educación