# **61. REST APIs**

## **API Review**

* An **API (Application Programming Interface)** is a software interface that allows two applications to communicate with each other.
* APIs are essential not just for network automation but for all kinds of applications.
* In **SDN Architecture**, APIs are used to communicate:
  + Between apps and the SDN controller (via the **Northbound Interface - NBI**).
  + Between the SDN controller and network devices (via the **Southbound Interface - SBI**).
* The **NBI** typically uses **REST APIs**.
* **NETCONF** and **RESTCONF** are popular Southbound APIs.

## **CRUD Operations and HTTP Verbs**

* **CRUD (Create, Read, Update, Delete)** refers to the operations performed using REST APIs.

### **Create**

* Used to create new variables and set their initial values.
  + Example: Create a variable **ip\_address** and set its value to **"10.1.1.1"**.

### **Read**

* Used to read the value of a variable.
  + Example: Read the value of variable **ip\_address** ("10.1.1.1").

### **Update**

* Used to change or update the value of a variable.
  + Example: Change the value of **ip\_address** from **"10.1.1.1"** to **"10.2.3.4"**.

### **Delete**

* Used to delete variables.  
  + Example: Delete variable **ip\_address**.
* **HTTP uses verbs (methods) that map to these CRUD operations**.
* **REST APIs typically use HTTP**.

## **HTTP Request**

* When an **HTTP client** sends a request to an **HTTP server**, the HTTP header includes:  
  + An **HTTP Verb** (e.g., **GET**).
  + A **URI (Uniform Resource Identifier)** indicating the resource it is trying to access.
* The HTTP request can include additional **headers** that pass extra information to the server.
* Example header:  
  + **Accept: application/json** (Indicates that the client expects JSON data in response).
* Standard HTTP headers list: [MDN HTTP Headers](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers).
* Another reference: [Wikipedia List of HTTP Header Fields](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields).

💡 **REST APIs do NOT have to use HTTP for communication, although HTTP is the most common choice.**

## **HTTP Response**

* The **server’s response** includes a **status code** indicating success or failure.
* The **first digit** indicates the class of response:
  + **1xx**: Informational – Request received, processing continues.
  + **2xx**: Successful – Request successfully processed.
  + **3xx**: Redirection – Further action needed.
  + **4xx**: Client Error – Invalid request.
  + **5xx**: Server Error – Server failed to fulfill the request.

### **Examples of HTTP Response Codes**

#### **1xx Informational**

* **102 Processing**: Server received the request and is processing it.

#### **2xx Successful**

* **200 OK**: Request succeeded.
* **201 Created**: Request succeeded, and a new resource was created.

#### **3xx Redirection**

* **301 Moved Permanently**: The requested resource has been moved permanently.

#### **4xx Client Error**

* **403 Unauthorized**: Client must authenticate to get a response.
* **404 Not Found**: The requested resource was not found.

#### **5xx Server Error**

* **500 Internal Server Error**: Server encountered an unexpected condition.

## **REST APIs**

* **REST (Representational State Transfer)** defines a set of architectural constraints for APIs.
* **REST APIs** are also known as **REST-based APIs** or **RESTful APIs**.
  + **REST is not a specific API**, but a set of guidelines for how APIs should be structured.

### **Six Constraints of RESTful Architecture**

1. **Stateless**
2. **Layered System**
3. **Uniform Interface**
4. **Client-Server**
5. **Cacheable or Non-Cacheable**
6. **Code-on-Demand (Optional)**

* **Networking protocols** are required for applications to communicate over a network.
* **For REST APIs, HTTP(S) is the most common choice.**

## **REST: Client-Server**

* REST APIs use a **client-server** architecture.
* The **client** uses API calls (**HTTP requests**) to access **server** resources.
* The separation between client and server allows independent development and evolution.
  + Changes in either the client or server should **not break the interface**.

## **REST: Stateless**

* REST API exchanges are **stateless**.
* Each API exchange is **independent** of previous exchanges.
  + The **server does not store information** about past requests.
* If **authentication** is required, the client must authenticate for **each request**.
* **TCP** is an example of a **stateful** protocol.
* **UDP** is an example of a **stateless** protocol.

**Although REST APIs use HTTP (which uses TCP at Layer 4), HTTP and REST APIs themselves are stateless.**

## **REST: Cacheable or Non-Cacheable**

* REST APIs **must support caching**.
* **Caching** refers to storing data for future use.
  + Example: A web browser caching page elements to improve performance.
* **Not all resources must be cacheable**, but those that are must be **declared as cacheable**.

## **REST API Calls Using Cisco DevNet**

* **Cisco DevNet** is Cisco’s developer program that helps professionals learn about automation and integrations with Cisco products.
* DevNet offers **free resources** like:
  + Courses
  + Tutorials
  + Labs
  + Sandboxes
  + Documentation
* Cisco also has a **DevNet certification track** for those interested in automation.

### **Using Cisco DNA Center Sandbox for REST API Calls**

* **Cisco DNA Center** is one of Cisco’s **SDN Controllers**.
* **Postman** is a platform for building and using APIs.

### **To Get Started:**

1. Create an account on [Cisco DevNet](http://developer.cisco.com/) (Used NetAcademy login).
2. Create an account on [Postman](http://postman.com/) and download the **Postman Desktop App**: [Postman Download](https://www.postman.com/downloads) (Used Gmail login).

This structured format makes the information clearer and easier to read!