



REST

Representational State Transfer



APIs are everywhere



- We use APIs all the time

REST

- Client–server
 - By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.
- Stateless
 - Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server.
 - Session state is therefore kept entirely on the client.
- Cacheable
 - Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests

• Source: <https://restfulapi.net/>



CRUD

- Four actions performed by an application:
 - Create: Allows the client to create some new instances of variables and data structures at the server and initialize their values as kept at the server
 - Read: Allows the client to retrieve (read) the current value of variables that exist at the server, storing a copy of the variables, structures, and values at the client
 - Update: Allows the client to change (update) the value of variables that exist at the server
 - Delete: Allows the client to delete from the server different instances of data variables



CRUD and HTTP Verbs

REST (HTTP) Verb	CRUD Term	Action
POST	CREATE (C)RUD	Create new data structures and variables
GET	READ (C)RUD	Read (Retrieve) variable names, structures and values
PATCH, PUT	UPDATE (C)RUD	Update or replace values of some variable
DELETE	DELETE (C)RUD	Delete some variables and data structures



DNA Center

- DevNet DNA Center Always on Lab
 - <https://sandboxdnac.cisco.com>
 - User: devnetuser
 - Password: Cisco123!
- API documentation:
 - <https://developer.cisco.com/docs/dna-center/api/1-3-3-x/?version=1.2.x>
- DNA Center hello world documentation
 - <https://developer.cisco.com/docs/dna-center/#!/hello-world/device-inventory-example>



DNA Center Inventory



Inventory



Filter

Device Name	IP Address ▲	Reachability Status	Uptime	Last Updated	Resync Interval	Last Sync Status
cat_9k_1.abc.inc	10.10.22.66	✔ Reachable	117 days 20 hrs 17 mins	27 minutes ago	00:25:00	In Progress
cat_9k_2.abc.inc	10.10.22.70	✔ Reachable	117 days 20 hrs 38 mins	8 minutes ago	00:25:00	Managed
cs3850.abc.inc	10.10.22.73	✔ Reachable	117 days 20 hrs 54 mins	22 minutes ago	00:25:00	Managed
asr1001-x.abc.inc	10.10.22.253	✔ Reachable	28 days 10 hrs 04 mins	7 minutes ago	00:25:00	Managed

Show 10 entries

Showing 1 - 4 of 4



DNA Center API

Intent API ^{1.3.3.x}

[Antman-swagger-v1.annotated.json](#)

Cisco DNA Center Platform v. 1.3.3.x

Filter by tag

Authentication Access Token Request

POST `/dna/system/api/v1/auth/token` Authentication API

Sites Create sites, assign devices to them and get site health

Topology Get topology details and overall network health

Devices Manage network devices

GET `/dna/intent/api/v1/network-device/module/{id}` Get Module Info by Id

GET `/dna/intent/api/v1/network-device/{id}/vlan` Get Device Interface VLANs

GET `/dna/intent/api/v1/interface/network-device/{deviceId}/{startIndex}/{recordsToReturn}` Get Device Interfaces by specified range

GET `/dna/intent/api/v1/network-device` Get Device list

POST `/dna/intent/api/v1/network-device` Add Device



REST (Uniform Resource Identifiers) URI

- URI
 - <https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device?type=Cisco ASR 1001-X Router>
 - Protocol = https
 - Server / Host URL = sandboxdnac.cisco.com
 - Resource = /dna/intent/api/v1/network-device
 - Parameters = ?type=Cisco ASR 1001-X Router



Postman

The Collaboration Platform for API Development

Download the free Postman app to get started.

Download the App



10 million
Developers

500,000
Companies

250 million
APIs



You need a token

- Authorization is required
 - URL: <https://sandboxdnac.cisco.com/dna/system/api/v1/auth/token>
 - Username: devnetuser
 - Password: Cisco123!
 - Use POST
- Copy the Token you get back

The screenshot displays a REST client interface for a POST request to the URL `https://sandboxdnac.cisco.com/dna/system/api/v1/auth/token`. The request is configured with Basic Authentication, where the username is `devnetuser` and the password is `Cisco123!`. The response status is `200 OK` with a response time of `758ms` and a size of `590 B`. The response body, shown in JSON format, contains a long alphanumeric string labeled `"Token"`. Red arrows highlight the URL, the POST method, the authentication fields, and the resulting token.

```
POST https://sandboxdnac.cisco.com/dna/system/api/v1/auth/token
```

Authorization: Basic devnetuser:Cisco123!

Status: 200 OK Time: 758ms Size: 590 B

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI1ZTNkNDI5ZTdjdDQ3ZTAwNGM2NmMGMiLCJhdXRoU291cmNlIjoiaW50ZXJuYmwiLCJ0ZW5hbnR0Yw1lIjoieV5UMCI6InJvbGVzIjpbIjVkyZQ0NGQ1MTQ4NWw1MDA0YzBmYjIxMiJdLCJ0ZW5hbnRJCi6IjVkyZQ0NGQ1MTQ4NWw1MDA0YzBmYjIwYiIsImV4cCI6MTU4MzZlMzYwYmYwIiwiaWF0IjoiMj02MDYwIiwiaXNjaW50eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9"
}
```

How to get information

- URL: <https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device>
- Headers
 - Content-Type: application/json
 - X-Auth-Token: <paste in your token>
- Use GET

Network Devices Comments (0) Examples (0)


GET ▼ <https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device> Send Save ▼

Params Authorization **Headers (9)** Body Pre-request Script Tests Settings Cookies Code

▼ Headers (2)

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Content-Type	application/json				
<input checked="" type="checkbox"/>	X-Auth-Token	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWiiOiI1ZTNk...				
	Key	Value	Description			

▶ Temporary Headers (7) ⓘ



Result:

- Get a list of devices

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

- Method:** GET
- URL:** `https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device`
- Headers (2):**

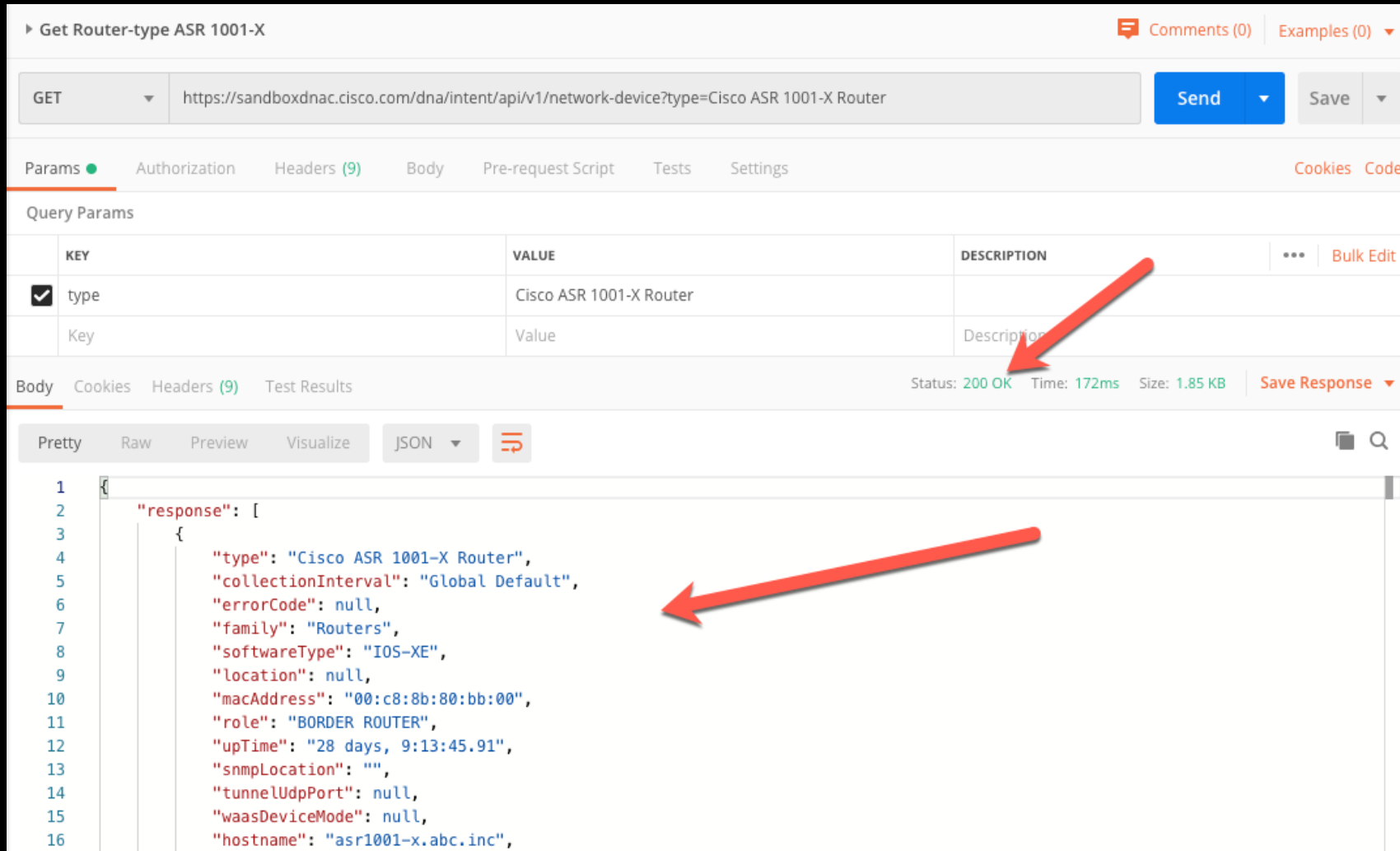
KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> X-Auth-Token	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWwiOiI1ZTNkNDI5ZTdj...	
Key	Value	Description
- Temporary Headers (8):** (Collapsed)
- Status:** 200 OK
- Time:** 361ms
- Size:** 5.45 KB
- Response Body (JSON):**

```
1 {
2   "response": [
3     {
4       "type": "Cisco ASR 1001-X Router",
5       "collectionInterval": "Global Default",
6       "errorCode": null,
7       "family": "Routers",
8       "softwareType": "IOS-XE",
9     }
10  ]
11 }
```



Status Code 200: OK

- Life is good



Get Router-type ASR 1001-X

GET <https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device?type=Cisco ASR 1001-X Router> Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> type	Cisco ASR 1001-X Router	
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 172ms Size: 1.85 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "response": [
3     {
4       "type": "Cisco ASR 1001-X Router",
5       "collectionInterval": "Global Default",
6       "errorCode": null,
7       "family": "Routers",
8       "softwareType": "IOS-XE",
9       "location": null,
10      "macAddress": "00:c8:8b:80:bb:00",
11      "role": "BORDER ROUTER",
12      "upTime": "28 days, 9:13:45.91",
13      "snmpLocation": "",
14      "tunnelUdpPort": null,
15      "waasDeviceMode": null,
16      "hostname": "asr1001-x.abc.inc",
```



401 Error Code: Unauthorized

- Token not used

The screenshot displays a REST client interface for a GET request to `https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device`. The response status is `401 Unauthorized` with a time of `292ms` and a size of `289 B`. The response body is shown in JSON format as `{ "message": "Unauthorized" }`. Two red arrows highlight the `DESCRIPTION` column in the Query Params table and the `"message": "Unauthorized"` value in the JSON body.

Unauthorized

Comments (0) Examples (0)

GET `https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device` Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (7) Test Results Status: 401 Unauthorized Time: 292ms Size: 289 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Unauthorized"
3 }
```

400 Error Code: Bad Request

- network-device123 is not found

Code 400: Bad Request

GET `https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device123?type=Cisco ASR 1001-X Router`

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> type	Cisco ASR 1001-X Router	
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 400 Bad Request Time: 88ms Size: 498 B Save Response

```
1 {
2   "response": {
3     "errorCode": "Bad request",
4     "message": "Invalid input request",
5     "detail": "123 is not a valid UUID of device"
6   },
7   "version": "1.0"
8 }
```




MENU



Oops, you've found a broken bridge

404 - Page not found.

Code 404 Error: Not found

- Resource not found

The screenshot displays a REST client interface for a GET request. The URL is `https://sandboxdnac.cisco.com/dna/intent/api/v1/networkdevice?softwaretype=IOS-XE`. The response status is `404 Not Found` with a time of `81ms` and a size of `314 B`. The response body is a JSON object: `{ "error" : "__technicalName, networkdevice is not found." }`. Red arrows highlight the URL, the `softwaretype` parameter, the `404 Not Found` status, and the error message in the response body.

Code 404: Not Found

GET `https://sandboxdnac.cisco.com/dna/intent/api/v1/networkdevice?softwaretype=IOS-XE` Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> softwaretype	IOS-XE			
Key	Value	Description		

Body Cookies Headers (7) Test Results Status: 404 Not Found Time: 81ms Size: 314 B Save Response

Pretty Raw Preview Visualize Text

```
1 {
2   "error" : "__technicalName, networkdevice is not found."
3 }
```

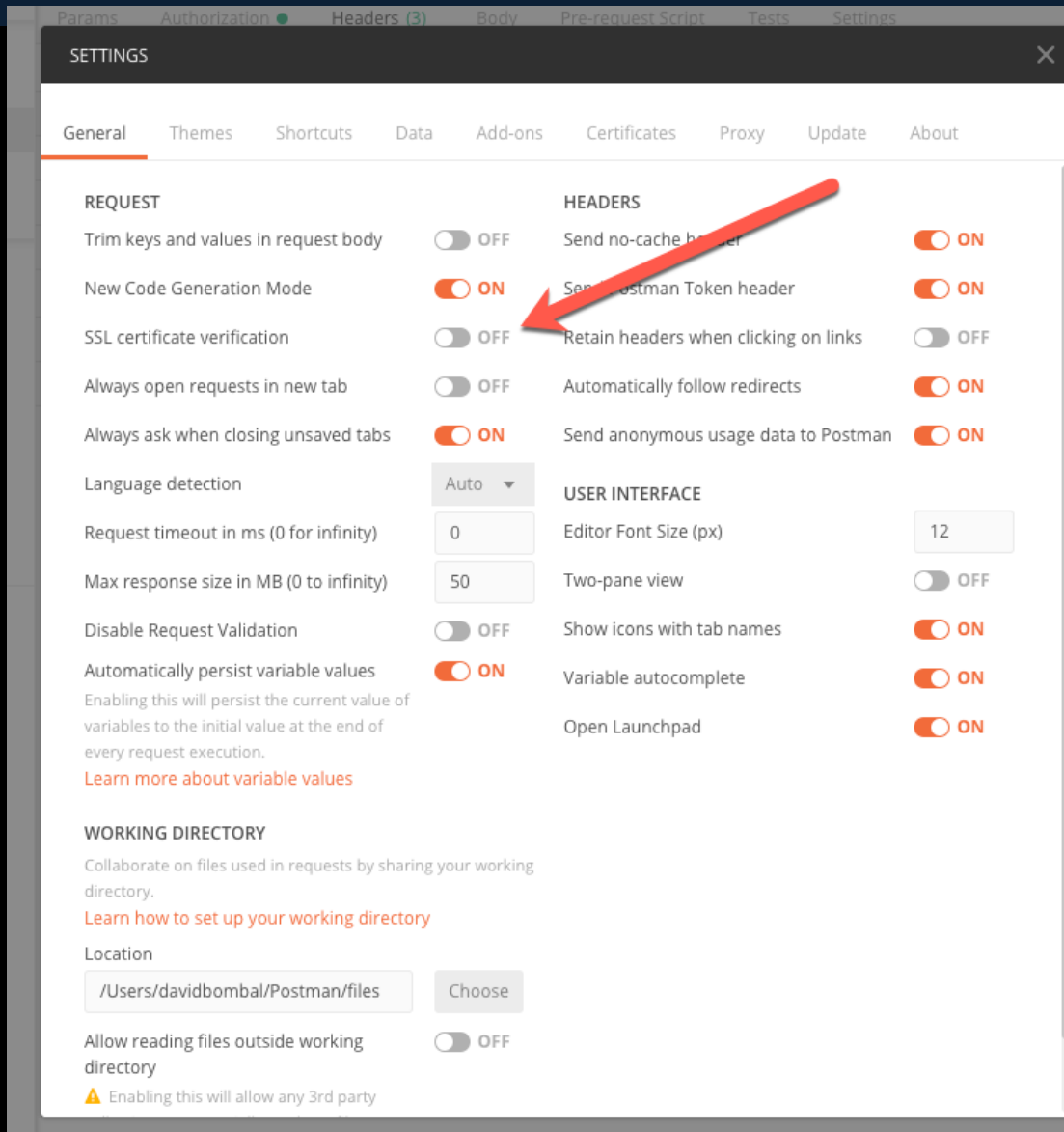


IOS XE

- Use the DevNet IOS XE always on sandbox
 - DNS name: ios-xe-mgmt.cisco.com
 - Protocol: HTTPS
 - Port: 9443
 - Username: developer
 - Password: C1sco12345
- Application:
 - Use Postman
 - Free download from <https://www.postman.com>



Turn of certificate verification



The screenshot shows the Postman Settings window with the 'Certificates' tab selected. The 'SSL certificate verification' toggle is turned off. A red arrow points to this toggle. Other settings are visible in the 'REQUEST' and 'HEADERS' sections.

Section	Setting	Value
REQUEST	Trim keys and values in request body	OFF
	New Code Generation Mode	ON
	SSL certificate verification	OFF
	Always open requests in new tab	OFF
	Always ask when closing unsaved tabs	ON
	Language detection	Auto
	Request timeout in ms (0 for infinity)	0
	Max response size in MB (0 to infinity)	50
	Disable Request Validation	OFF
	Automatically persist variable values	ON
HEADERS	Send no-cache header	ON
	Send Postman Token header	ON
	Retain headers when clicking on links	OFF
	Automatically follow redirects	ON
USER INTERFACE	Send anonymous usage data to Postman	ON
	Editor Font Size (px)	12
	Two-pane view	OFF
	Show icons with tab names	ON
WORKING DIRECTORY	Variable autocomplete	ON
	Open Launchpad	ON
	Location	/Users/davidbombal/Postman/files
WORKING DIRECTORY	Allow reading files outside working directory	OFF
	Warning	Enabling this will allow any 3rd party



Get interfaces Part 1

- Use the DevNet IOS XE always on sandbox
 - URL: <https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces>
 - Method: GET
 - Username: developer
 - Password: C1sco12345

GET ▼ <https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces>

Params **Authorization** ● Headers (9) Body Pre-request Script Tests Settings

TYPE

Basic Auth ▼

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

[Preview Request](#)

! Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we've automatically masked the password field. [Learn more about variables](#)

Username

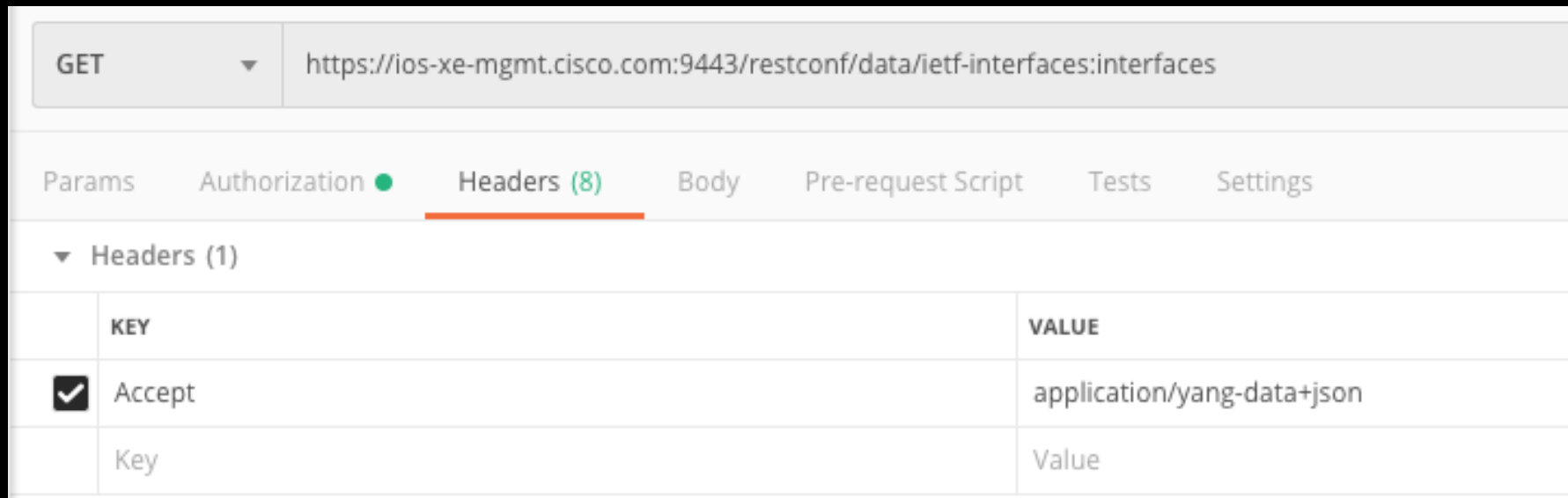
Password

Show Password



Get interfaces Part 2

- Add a header
 - Key 1:
 - Accept
 - application/yang-data+json



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

- Method: GET
- URL: `https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces`
- Active tab: Headers (8)
- Header list: Headers (1)
- Header table:

	KEY	VALUE
<input checked="" type="checkbox"/>	Accept	application/yang-data+json
	Key	Value

Get interfaces Part 3

```
Body Cookies Headers (7) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "ietf-interfaces:interfaces": {
3     "interface": [
4       {
5         "name": "GigabitEthernet1",
6         "description": "MANAGEMENT INTERFACE - DON'T TOUCH ME",
7         "type": "iana-if-type:ethernetCsmacd",
8         "enabled": true,
9         "ietf-ip:ipv4": {
10          "address": [
11            {
12              "ip": "10.10.20.48",
13              "netmask": "255.255.255.0"
14            }
15          ]
16        },
17        "ietf-ip:ipv6": {}
18      },
19      {
20        "name": "GigabitEthernet2",
21        "description": "Network Interface",
22        "type": "iana-if-type:ethernetCsmacd",
23        "enabled": false,
24        "ietf-ip:ipv4": {},
25        "ietf-ip:ipv6": {}
26      },
27      {
28        "name": "GigabitEthernet3",
29        "description": "Network Interface",
30        "type": "iana-if-type:ethernetCsmacd",
31        "enabled": false,
32        "ietf-ip:ipv4": {},
33        "ietf-ip:ipv6": {}
34      }
35    ]
36  }
37 }
```



Add an interface Part 1

- Use the DevNet IOS XE always on sandbox
 - URL: <https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces>
 - Method: POST
 - Username: developer
 - Password: C1sco12345

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

- Method:** POST
- URL:** `https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces`
- Authorization:** Basic Auth
- Username:** developer
- Password:** C1sco12345
- Show Password:**

A warning message is displayed: "Heads up! These parameters hold sensitive data. To keep this data secure while working in... Learn more about variables".

There is also a "Preview Request" button.



Add Interface Part 2

- Add a header
 - Key 1:
 - Accept
 - application/yang-data+json
 - Key 2:
 - Content-Type
 - Application/yang-data+json

► Create an interface

POST ▼ <https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces>

Params Authorization ● **Headers (10)** Body ● Pre-request Script Tests Settings

▼ Headers (2)

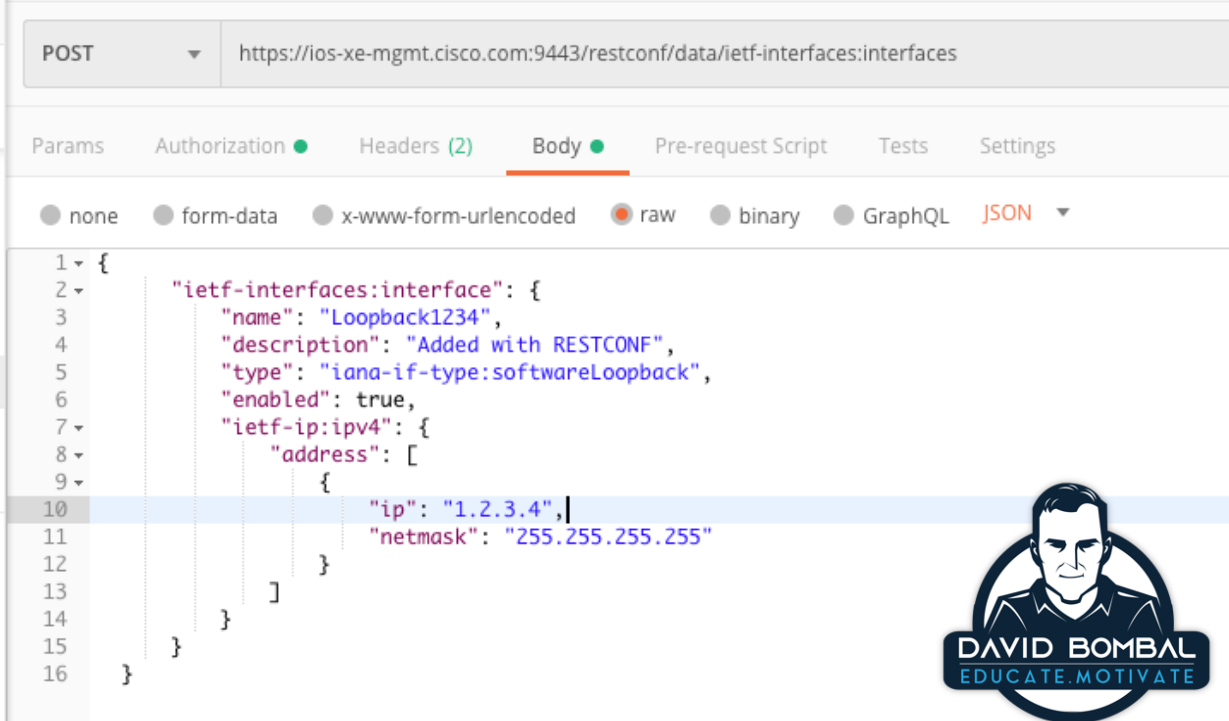
	KEY	VALUE
<input checked="" type="checkbox"/>	Accept	application/yang-data+json
<input checked="" type="checkbox"/>	Content-Type	application/yang-data+json



Add Interface Part 3

- Body:


```
{  
  "ietf-interfaces:interface": {  
    "name": "Loopback1234",  
    "description": "Added with RESTCONF",  
    "type": "iana-if-type:softwareLoopback",  
    "enabled": true,  
    "ietf-ip:ipv4": {  
      "address": [  
        {  
          "ip": "1.2.3.4",  
          "netmask": "255.255.255.255"  
        }  
      ]  
    }  
  }  
}
```



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

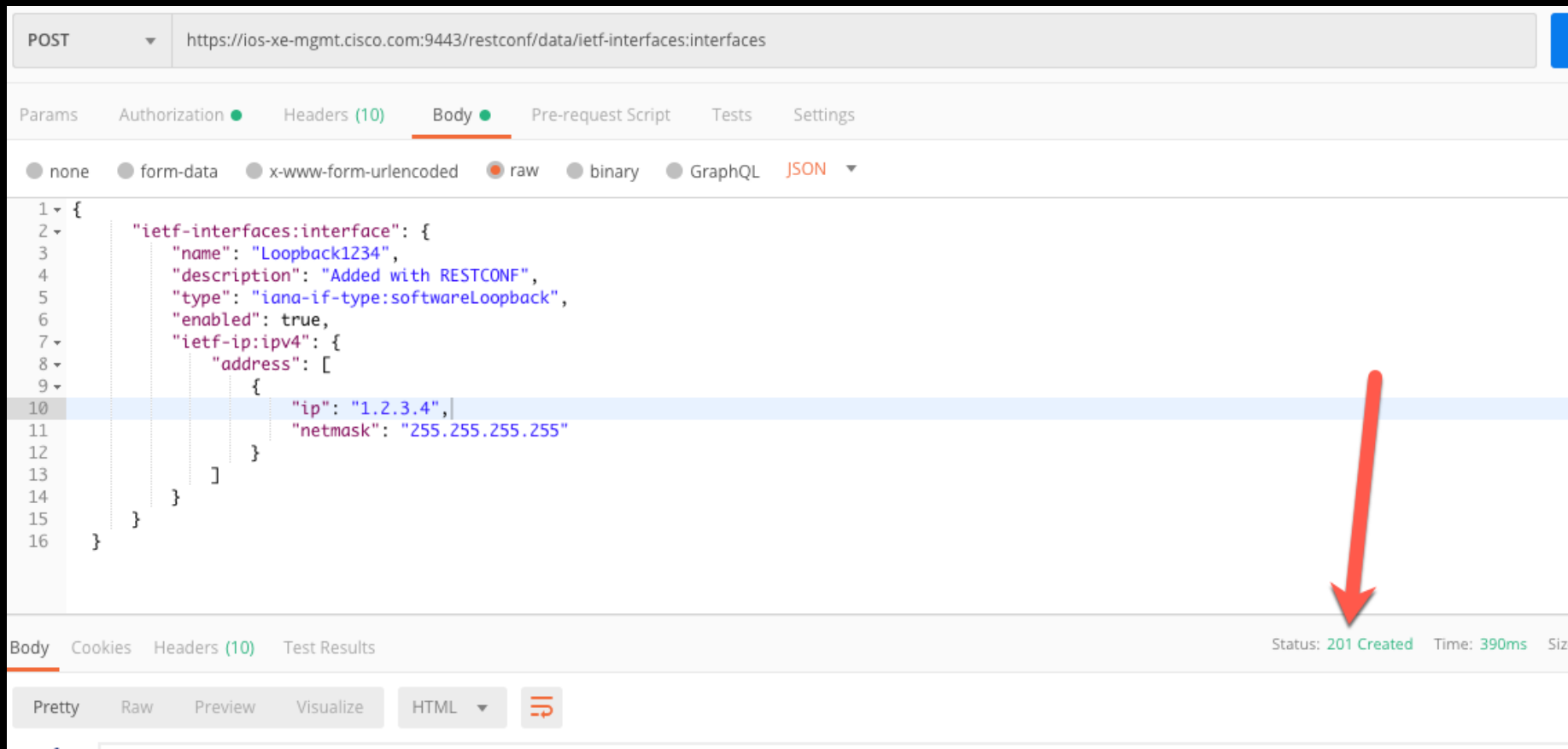
- Method: POST
- URL: `https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces`
- Body tab selected, showing raw JSON data.
- JSON body content (lines 1-16):

```
1 {  
2   "ietf-interfaces:interface": {  
3     "name": "Loopback1234",  
4     "description": "Added with RESTCONF",  
5     "type": "iana-if-type:softwareLoopback",  
6     "enabled": true,  
7     "ietf-ip:ipv4": {  
8       "address": [  
9         {  
10          "ip": "1.2.3.4",  
11          "netmask": "255.255.255.255"  
12        }  
13      ]  
14    }  
15  }  
16 }
```

 DAVID BOMBAL
EDUCATE. MOTIVATE

Add Interface Part 4

- 201 Status means success



The screenshot shows a REST client interface with a POST request to `https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces`. The request body is a JSON object defining a loopback interface named "Loopback1234" with IP address "1.2.3.4" and netmask "255.255.255.255". The response status is "201 Created" with a time of "390ms". A red arrow points from the response status area back to the request body.

```
1 {
2   "ietf-interfaces:interface": {
3     "name": "Loopback1234",
4     "description": "Added with RESTCONF",
5     "type": "iana-if-type:softwareLoopback",
6     "enabled": true,
7     "ietf-ip:ipv4": {
8       "address": [
9         {
10          "ip": "1.2.3.4",
11          "netmask": "255.255.255.255"
12        }
13      ]
14    }
15  }
16 }
```

Status: 201 Created Time: 390ms Size: ...



Add Interface Part 5

- Use GET to check that interface has been created

```
GET https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces

Pretty Raw Preview Visualize JSON ↕

36     "name": "GigabitEthernet3",
37     "description": "Network Interface",
38     "type": "iana-if-type:ethernetCsmacd",
39     "enabled": false,
40     "ietf-ip:ipv4": {},
41     "ietf-ip:ipv6": {}
42   },
43   {
44     "name": "Loopback1234",
45     "description": "Added with RESTCONF",
46     "type": "iana-if-type:softwareLoopback",
47     "enabled": true,
48     "ietf-ip:ipv4": {
49       "address": [
50         {
51           "ip": "1.2.3.4",
52           "netmask": "255.255.255.255"
53         }
54       ]
55     },
56     "ietf-ip:ipv6": {}
57   }
58 ]
59 }
```



Delete an Interface Part 1

- Use the DevNet IOS XE always on sandbox
 - URL: <https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces/interface=Loopback1234>
 - Replace Interface name with correct name.
 - Method: DELETE
 - Username: developer
 - Password: C1sco12345

DELETE <https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces/interface=Loopback1234...>

Params **Authorization** Headers (10) Body Pre-request Script Tests Settings

TYPE
Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

[Preview Request](#)

! Heads up! These parameters hold sensitive data. To keep this data secure while w
[Learn more about variables](#)

Username developer

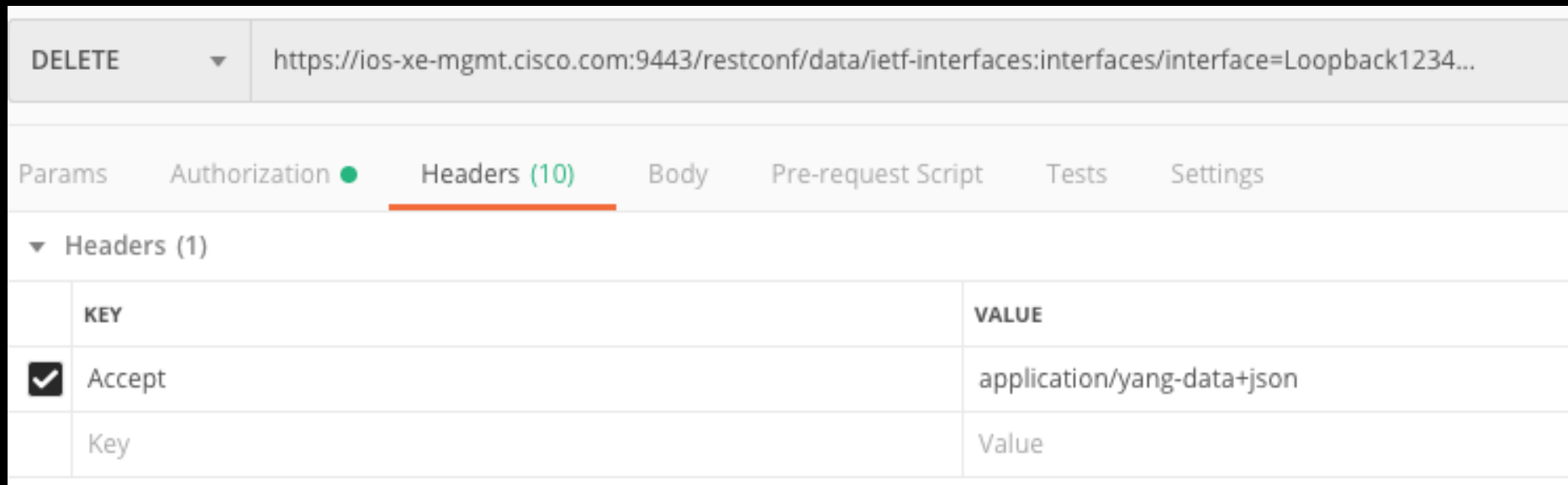
Password C1sco12345

Show Password



Delete and Interface Part 2

- Add a header
 - Key 1:
 - Accept
 - application/yang-data+json



DELETE <https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces/interface=Loopback1234...>

Params Authorization ● Headers (10) Body Pre-request Script Tests Settings

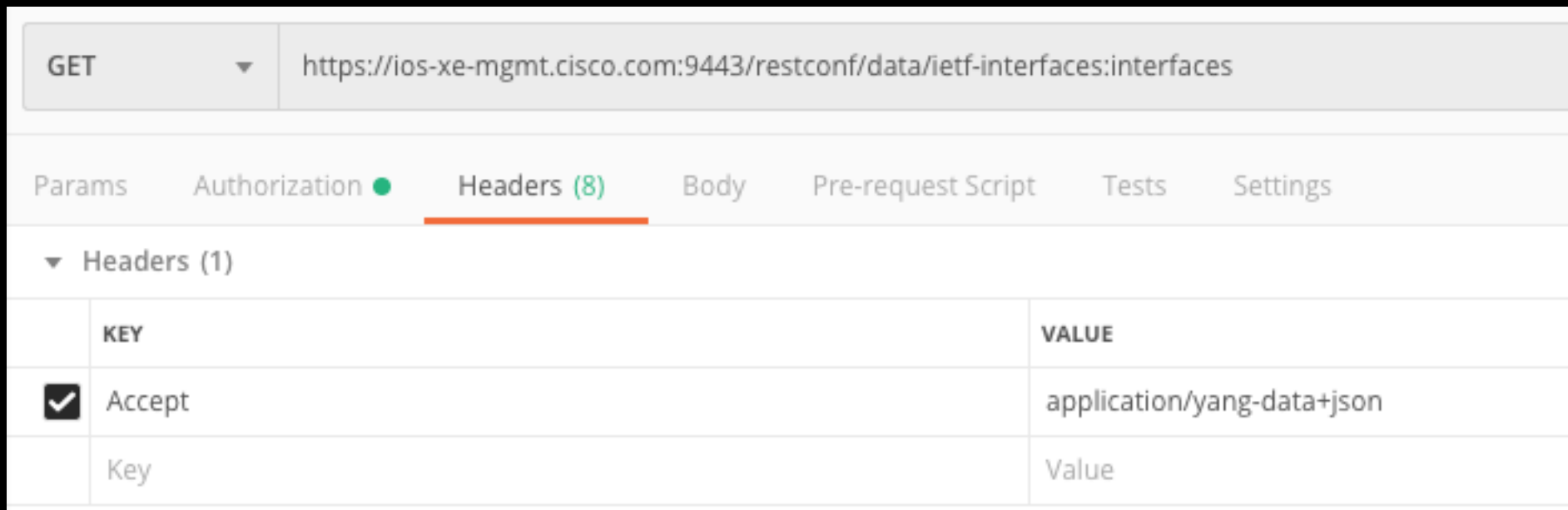
▼ Headers (1)

	KEY	VALUE
<input checked="" type="checkbox"/>	Accept	application/yang-data+json
	Key	Value



Delete and Interface Part 3

- Result:
 - Run the get interface command to check that interface has been removed



The screenshot shows a REST client interface with a GET request to the URL `https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-interfaces:interfaces`. The 'Headers' tab is selected, showing a table with one header: 'Accept' with the value 'application/yang-data+json'. There is also a 'Key' and 'Value' row at the bottom of the table.

KEY	VALUE
<input checked="" type="checkbox"/> Accept	application/yang-data+json
Key	Value

Python: Get Interfaces

```
import requests
import sys

# Allow self signed certs
requests.packages.urllib3.disable_warnings()

# Credentials
USER = 'developer'
PASS = 'C1sco12345'
```



Python: Get Interfaces

```
# URL for GET request
url = "https://ios-xe-mgmt.cisco.com:9443/restconf/data/ietf-
interfaces:interfaces"

# Set yang+json as the data formats
headers = {'Content-Type': 'application/yang-data+json',
           'Accept': 'application/yang-data+json'}

# Run GET
response = requests.get(url, auth=(USER, PASS),
                        headers=headers, verify=False)

print(response.text)
```



Good DevNet labs

- Go here for more:
- <https://developer.cisco.com/learning/tracks/iosxe-programmability/intro-device-level-interfaces/intro-restconf/step/1>
- <https://devnetsandbox.cisco.com/RM/Diagram/Index/27d9747a-db48-4565-8d44-df318f3e37ad?diagramType=Topology>





REST

Representational State Transfer

