

Deploying APIs to a Ruckus Smartzone Controller

This document is to help in the use and deployment of API call to a Ruckus Smartzone 3.4 Controller. The below link is a reference to the locations we will be using as well as a host of other API calls you will ever need. In the location, the first part is a version folder location. This is specific to versions of Smartzone. Here is an example:

Smartzone version 3.2 - /v3_0

Smartzone version 3.4 - /v4_0

Smartzone API Reference 3.4:

<http://docs.ruckuswireless.com/vscg-enterprise/vsz-e-public-api-reference-guide-3-4.html>

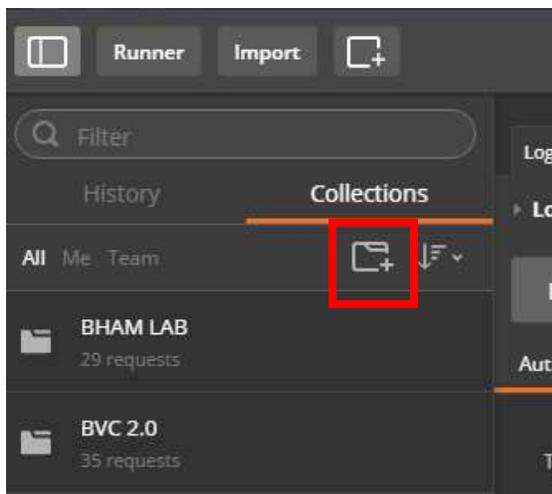
To deploy API commands to a Smartzone controller, there are several ways to accomplish this. For this demonstration, we will be using Postman by Google.

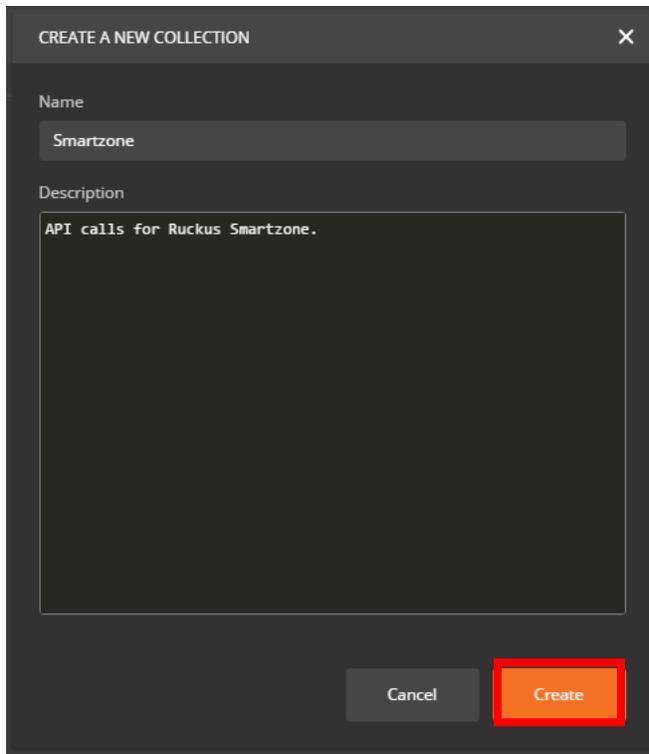
First, you will need to download and install Postman:

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

Once you have Postman installed, create a new Collection. This is where we will keep our API Calls.

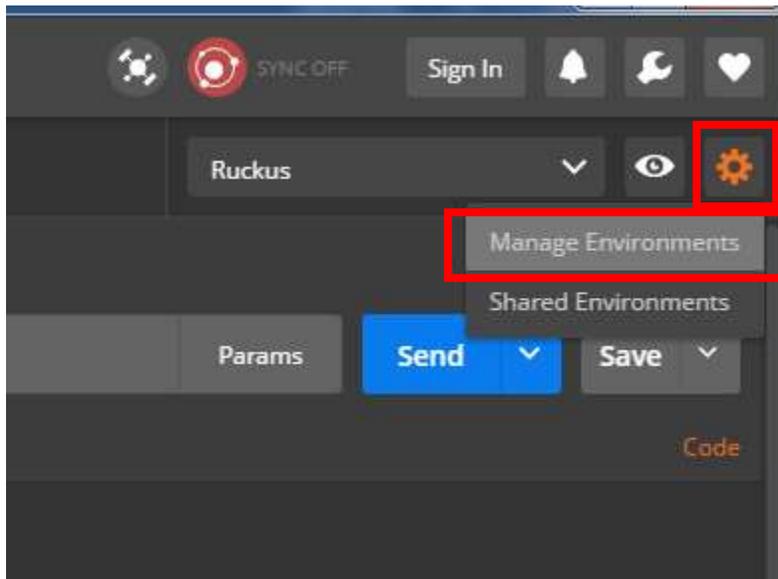
Create a Collection:





Create an Environment:

Select the Gear and Manage Environments:



Select Add:

The screenshot displays a dark-themed user interface for managing environments. At the top, a header bar contains the text "MANAGE ENVIRONMENTS" and a close icon (X). Below the header, there are two tabs: "Manage Environments" (which is active and underlined) and "Environment Templates". A descriptive text line reads "Environments help you customize requests according to variables." followed by a "Learn More" link. A list item for "Ruckus" is visible, with a "Share" button and icons for copy, download, and delete. At the bottom of the interface, there are three buttons: "Globals", "Import", and "Add". The "Add" button is highlighted with a red rectangular border.

Give the Environment a Name and Select Add:

The screenshot shows a dark-themed interface for managing environments. At the top, there is a header 'MANAGE ENVIRONMENTS' with a close button (X) on the right. Below the header, there are two tabs: 'Manage Environments' (which is active and underlined) and 'Environment Templates'. The main content area is titled 'Add Environment'. It features a text input field containing the name 'Smartzone', which is highlighted with a red box. Below the input field is a table with two columns: 'Key' and 'Value'. The first row has 'Key' and 'Value' as headers, with a 'Bulk Edit' link on the right. The second row has 'New key' and 'value' as placeholders. At the bottom right of the form, there are two buttons: 'Cancel' and 'Add'. The 'Add' button is highlighted with a red box.

Inside the Environment is where we will define variables for you API calls. The Key will be the variable name and the Value will be the value for the Key.

You will need to add two header values to every API you create. Below is an explanation and the values.

Common Request Header

The following parameters are required in the HTTP headers of all API requests (except for the logon API).

Parameter	Value
Content-Type	"application/json;charset=UTF-8"
Cookie	"JSESSIONID={JSESSIONID}"

JSESSIONID is returned as the following parameter in the response header of the logon API.

Parameter	Value
Set-cookie	"JSESSIONID={JSESSIONID}; Path=/wsg; Secure"

Create a Login Session API Call:

Select POST and add the destination to the controller and the location for this call. I have used a variable for the destination. Here is a break down:

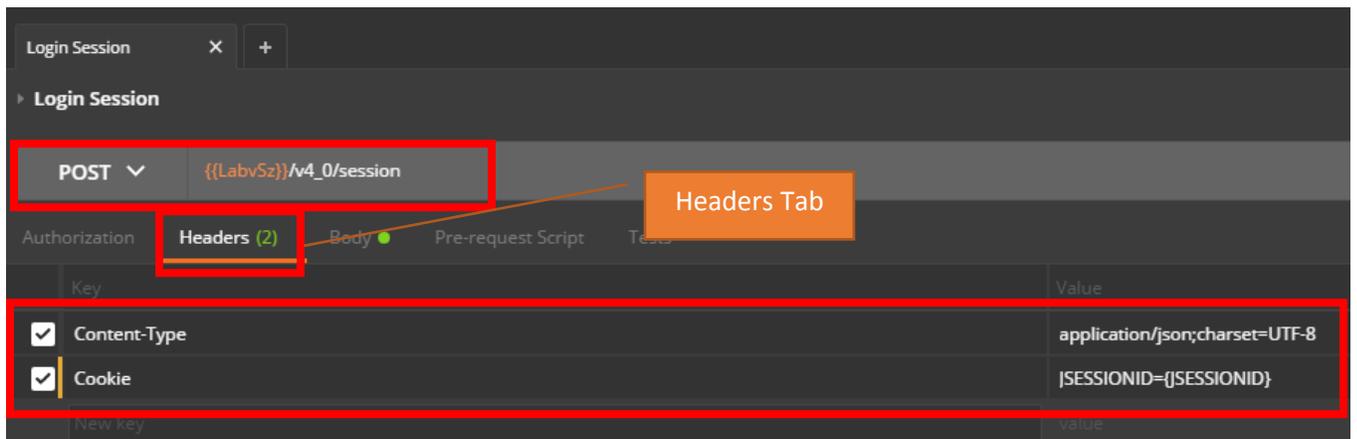
{{LabvSz}} - <https://{host}:7443/api/public> (Where {host} is the IP Address or Hostname of your controller). If you use the {{LabvSz}}, this is a variable created in your environment.

/v4_0/session is the location.

Under the Headers Tab, add the two fields for all your API Calls:

Content-Type - application/json;charset=UTF-8

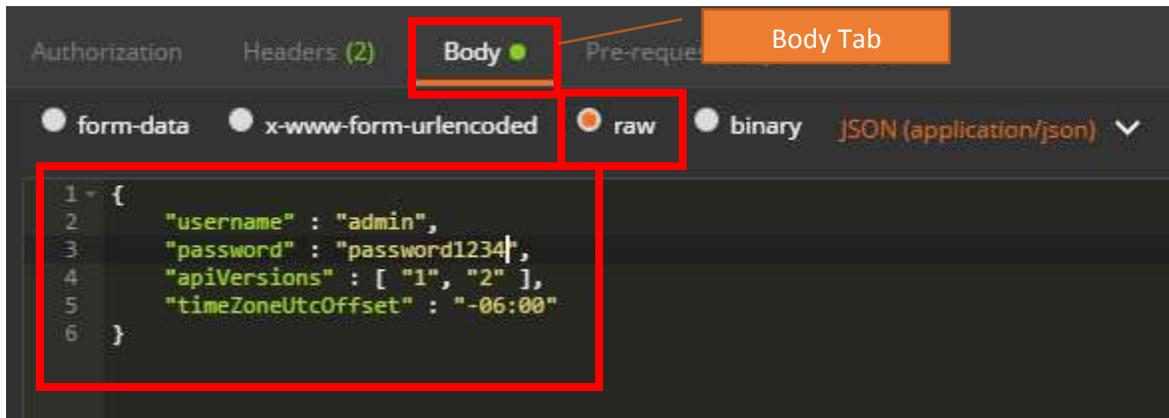
Cookie - JSESSIONID={JSESSIONID}



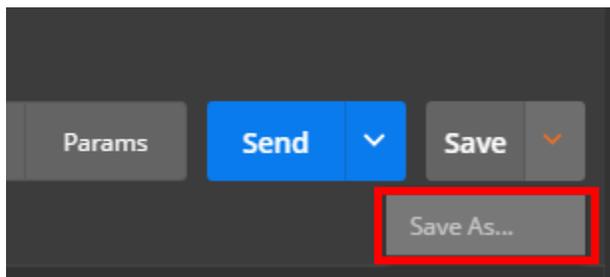
The screenshot shows an API client interface for a "Login Session" call. The method is set to POST and the URL is {{LabvSz}}/v4_0/session. The Headers tab is active, showing two headers: Content-Type (application/json;charset=UTF-8) and Cookie (JSESSIONID={JSESSIONID}).

Key	Value
<input checked="" type="checkbox"/> Content-Type	application/json;charset=UTF-8
<input checked="" type="checkbox"/> Cookie	JSESSIONID={JSESSIONID}

Under the Body Tab, you need to add the username and password to login to the controller. You will use the raw format and it must be formatted exactly as below. The apiVersion can be left to the default and the timeZoneUtcOffset is the time zone of the controller.



When you are done, select the down arrow next to Save to do Save As.



SAVE REQUEST ✕

Request Name

Request description (Optional)

Descriptions support Markdown

Save to existing collection / folder
 ▼

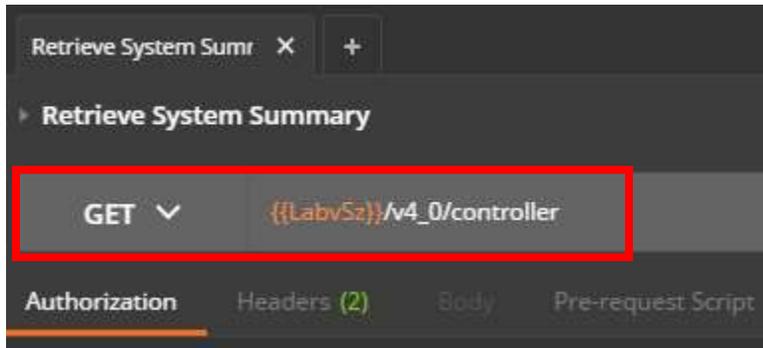
Or create new collection

Save the API to the Collection Name you created earlier.

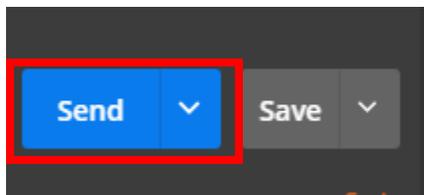
Retrieve the System Summary API:

Now we are going to create an API to Retrieve the System Summary.

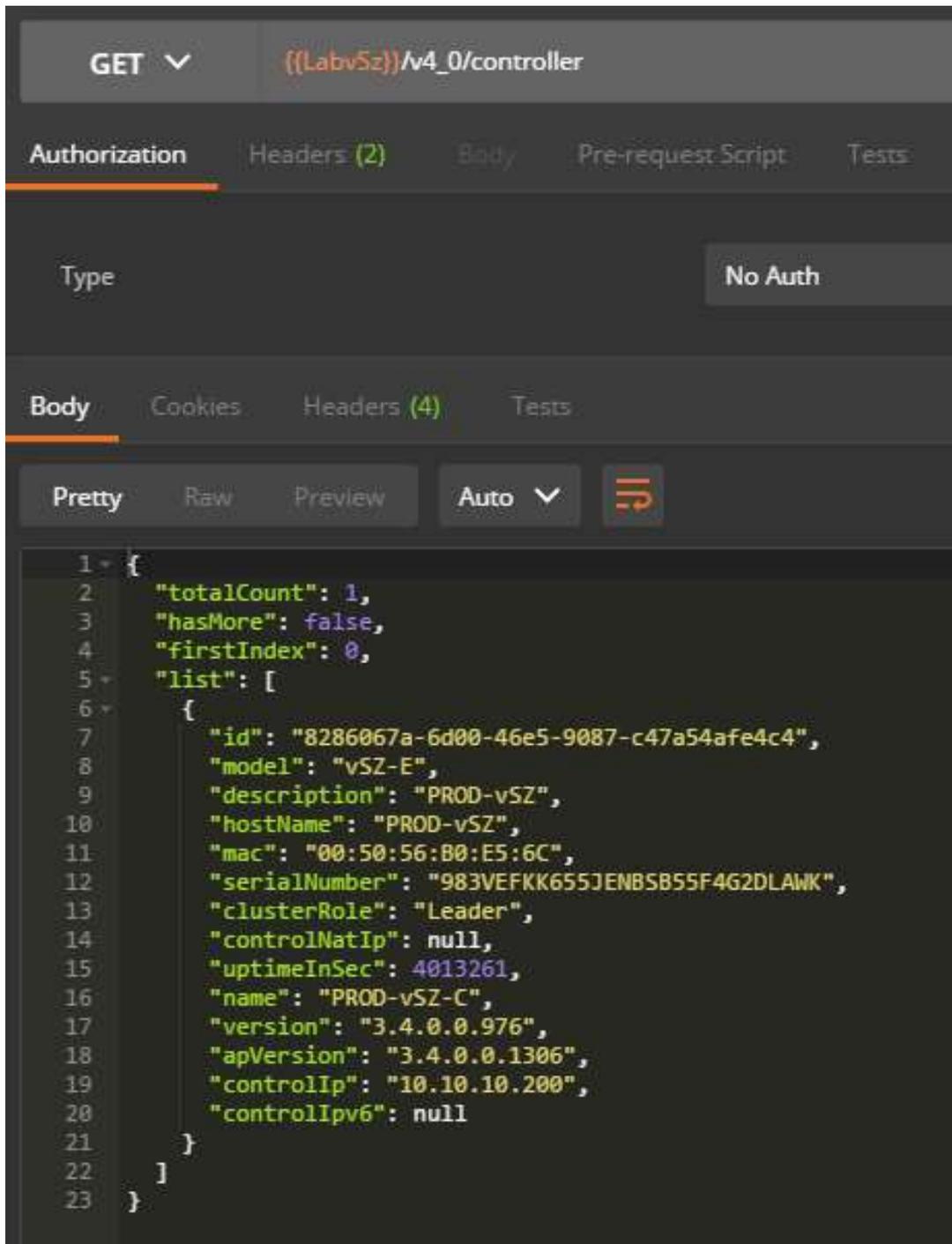
For this call Use GET and add the destination URL plus the location (/v4_0/controller). Remember to add the header information as above and Save AS when you are done.



To execute this call, select the send button.



Below is a Sample Output:



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

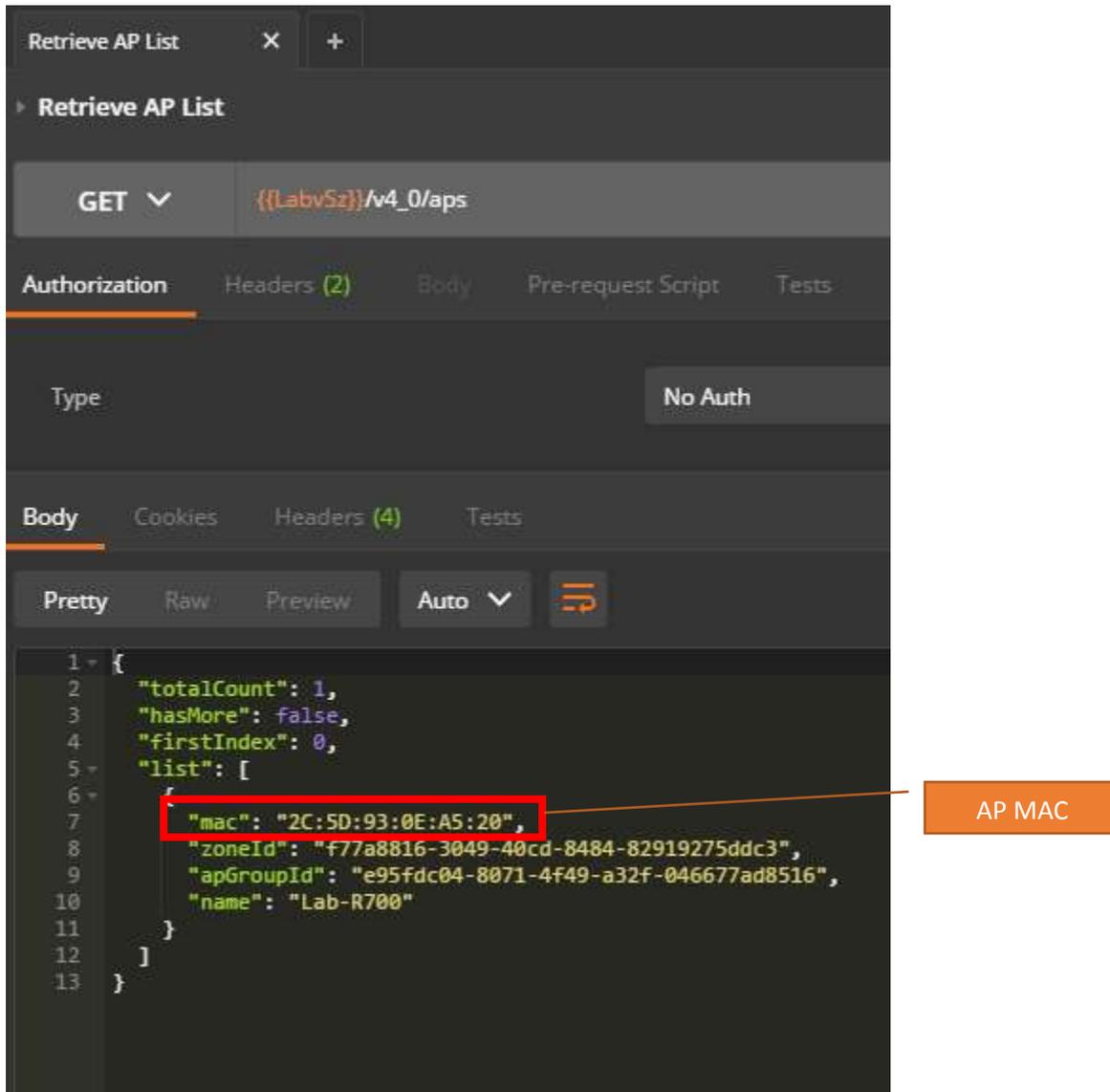
- Method:** GET
- URL:** {{LabvSz}}/v4_0/controller
- Authorization:** No Auth
- Body:** Pretty (selected), Raw, Preview, Auto (selected), and a refresh icon.
- Response Body (JSON):**

```
1 {
2   "totalCount": 1,
3   "hasMore": false,
4   "firstIndex": 0,
5   "list": [
6     {
7       "id": "8286067a-6d00-46e5-9087-c47a54afe4c4",
8       "model": "vSZ-E",
9       "description": "PROD-vSZ",
10      "hostName": "PROD-vSZ",
11      "mac": "00:50:56:80:E5:6C",
12      "serialNumber": "983VEFKK655JENBSB55F4G2DLAWK",
13      "clusterRole": "Leader",
14      "controlNatIp": null,
15      "uptimeInSec": 4013261,
16      "name": "PROD-vSZ-C",
17      "version": "3.4.0.0.976",
18      "apVersion": "3.4.0.0.1306",
19      "controlIp": "10.10.10.200",
20      "controlIpv6": null
21    }
22  ]
23 }
```

Retrieve AP List API:

Now we are going to create an API to retrieve a list of APs from the controller.

For this call Use GET and add the destination URL plus the location (/v4_0/aps). Remember to add the header information as above and Save AS when you are done. Below is the command and sample output.



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

- Request Method: GET
- Request URL: `{{LabvSz}}/v4_0/aps`
- Authorization: No Auth
- Response Body (Pretty):

```
1 {
2   "totalCount": 1,
3   "hasMore": false,
4   "firstIndex": 0,
5   "list": [
6     {
7       "mac": "2C:5D:93:0E:A5:20",
8       "zoneId": "f77a8816-3049-40cd-8484-82919275ddc3",
9       "apGroupId": "e95fdc04-8071-4f49-a32f-046677ad8516",
10      "name": "Lab-R700"
11    }
12  ]
13 }
```

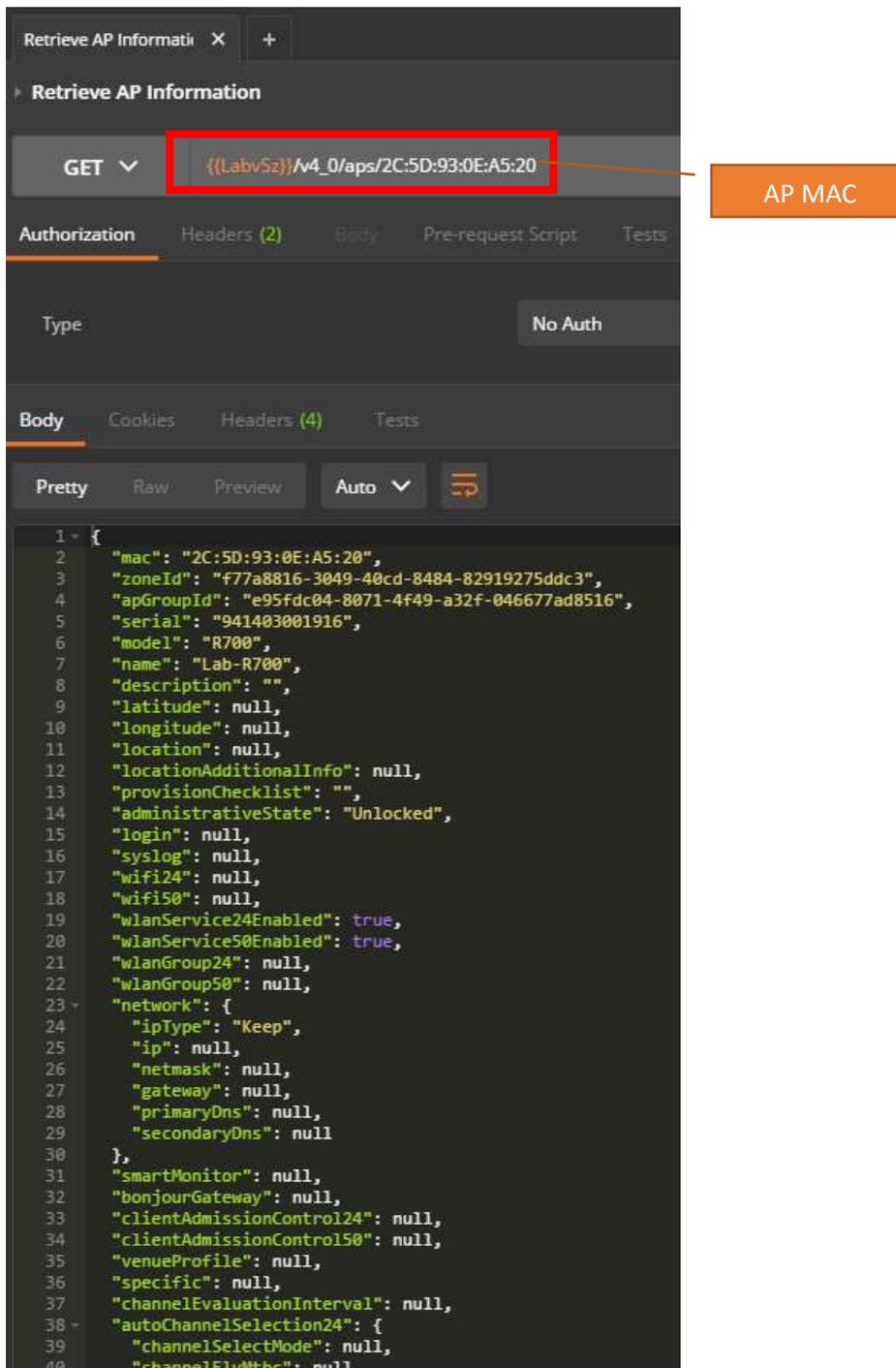
The MAC address `"mac": "2C:5D:93:0E:A5:20"` is highlighted in red, and an orange callout box labeled "AP MAC" points to it.

In this example, we received a list of APs from the controller. I only have one AP. From here we can get the MAC Address of the AP to use in the next API.

Retrieve AP Information API:

Now we are going to create an API to retrieve information about a specific AP from the controller.

For this call Use GET and add the destination URL plus the location (/v4_0/aps/{apMac}). For this example, we need the AP MAC Address from the last example to replace {apMac}. Remember to add the header information as above and Save AS when you are done. Below is the command and sample output.



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

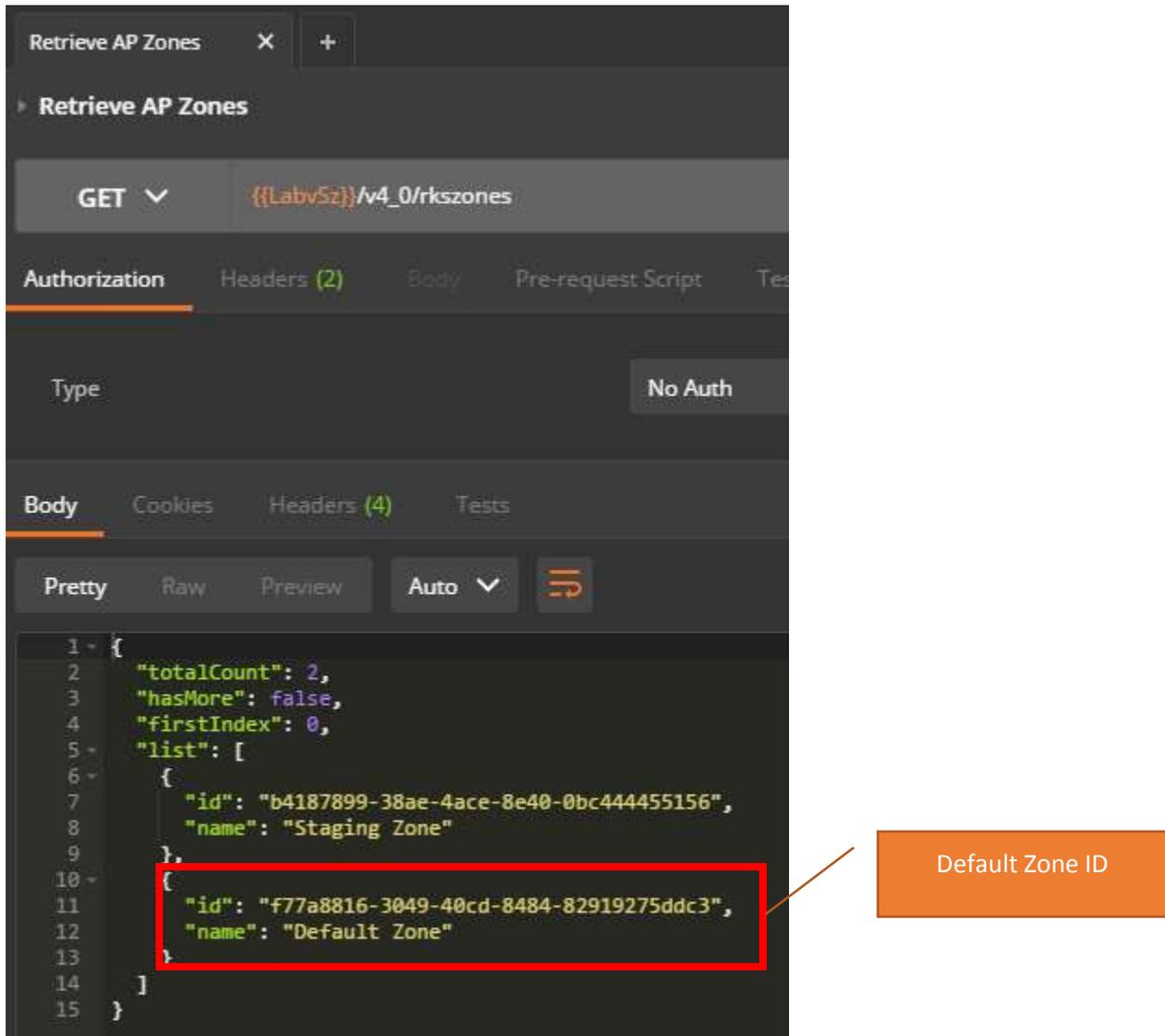
- Method:** GET
- URL:** `{{.LabvSz}}/v4_0/aps/2C:5D:93:0E:A5:20` (highlighted with a red box and labeled "AP MAC")
- Authorization:** No Auth
- Body:** Pretty view of the JSON response.

```
1 {
2   "mac": "2C:5D:93:0E:A5:20",
3   "zoneId": "f77a8816-3049-40cd-8484-82919275ddc3",
4   "apGroupId": "e95fdc04-8071-4f49-a32f-046677ad8516",
5   "serial": "941403001916",
6   "model": "R700",
7   "name": "Lab-R700",
8   "description": "",
9   "latitude": null,
10  "longitude": null,
11  "location": null,
12  "locationAdditionalInfo": null,
13  "provisionChecklist": "",
14  "administrativeState": "Unlocked",
15  "login": null,
16  "syslog": null,
17  "wifi24": null,
18  "wifi50": null,
19  "wlanService24Enabled": true,
20  "wlanService50Enabled": true,
21  "wlanGroup24": null,
22  "wlanGroup50": null,
23  "network": {
24    "ipType": "Keep",
25    "ip": null,
26    "netmask": null,
27    "gateway": null,
28    "primaryDns": null,
29    "secondaryDns": null
30  },
31  "smartMonitor": null,
32  "bonjourGateway": null,
33  "clientAdmissionControl24": null,
34  "clientAdmissionControl50": null,
35  "venueProfile": null,
36  "specific": null,
37  "channelEvaluationInterval": null,
38  "autoChannelSelection24": {
39    "channelSelectMode": null,
40    "channelFlvMtrc": null
```

Retrieve AP Zones API:

Now we are going to create an API to retrieve a list of AP Zones from the controller.

For this call Use GET and add the destination URL plus the location (/v4_0/rkszones). Remember to add the header information as above and Save AS when you are done. Below is the command and sample output.



The screenshot shows a REST client interface for the "Retrieve AP Zones" API. The request method is GET and the URL is `{{LabvSz}}/v4_0/rkszones`. The response is displayed in the "Body" tab, formatted as JSON. The response structure is as follows:

```
1 {
2   "totalCount": 2,
3   "hasMore": false,
4   "firstIndex": 0,
5   "list": [
6     {
7       "id": "b4187899-38ae-4ace-8e40-0bc444455156",
8       "name": "Staging Zone"
9     },
10    {
11      "id": "f77a8816-3049-40cd-8484-82919275ddc3",
12      "name": "Default Zone"
13    }
14  ]
15 }
```

An orange callout box labeled "Default Zone ID" points to the `"id": "f77a8816-3049-40cd-8484-82919275ddc3"` field in the "Default Zone" object.

We will use the "Default Zone" ID to perform the next API.

Create WLAN Group API:

Now we are going to create an API to create a WLAN Group on the controller.

For this call Use POST and add the destination URL plus the location (/v4_0/rkszones/{zoneid}/wlangroups). Remember to add the header information as above. Here we will also have to add information to the Body Tab. The body tab has the name of the WLAN Group we want to create and the Description. This is input in the raw format. Refer to the API Guide for more details. Be sure to Save AS when you are done. Below is the command and sample output.

The screenshot shows an API client interface for a request titled "Create WLAN Group". The request method is "POST" and the URL is `{{LabvSz}}/v4_0/rkszones/{{zoneid}}/wlangroups`. The "Body" tab is selected, and the "raw" format is chosen. The body content is a JSON object:

```
1 {
2   "name" : "JSON-WLANgroup",
3   "description" : "Group created from JSON API"
4 }
5
```

 Annotations include: "Zone ID from previous example" pointing to the `{{zoneid}}` placeholder in the URL; "Body Tab" pointing to the "Body" tab; and "Input must be formatted exactly as above" pointing to the JSON body content.

When you execute this command with the Send Button, you will see the following:

The screenshot shows the response body in the "Body" tab, formatted as "Pretty". The response is a JSON object:

```
1 {
2   "id": "59256821-399a-11e7-a039-000000007255"
3 }
```

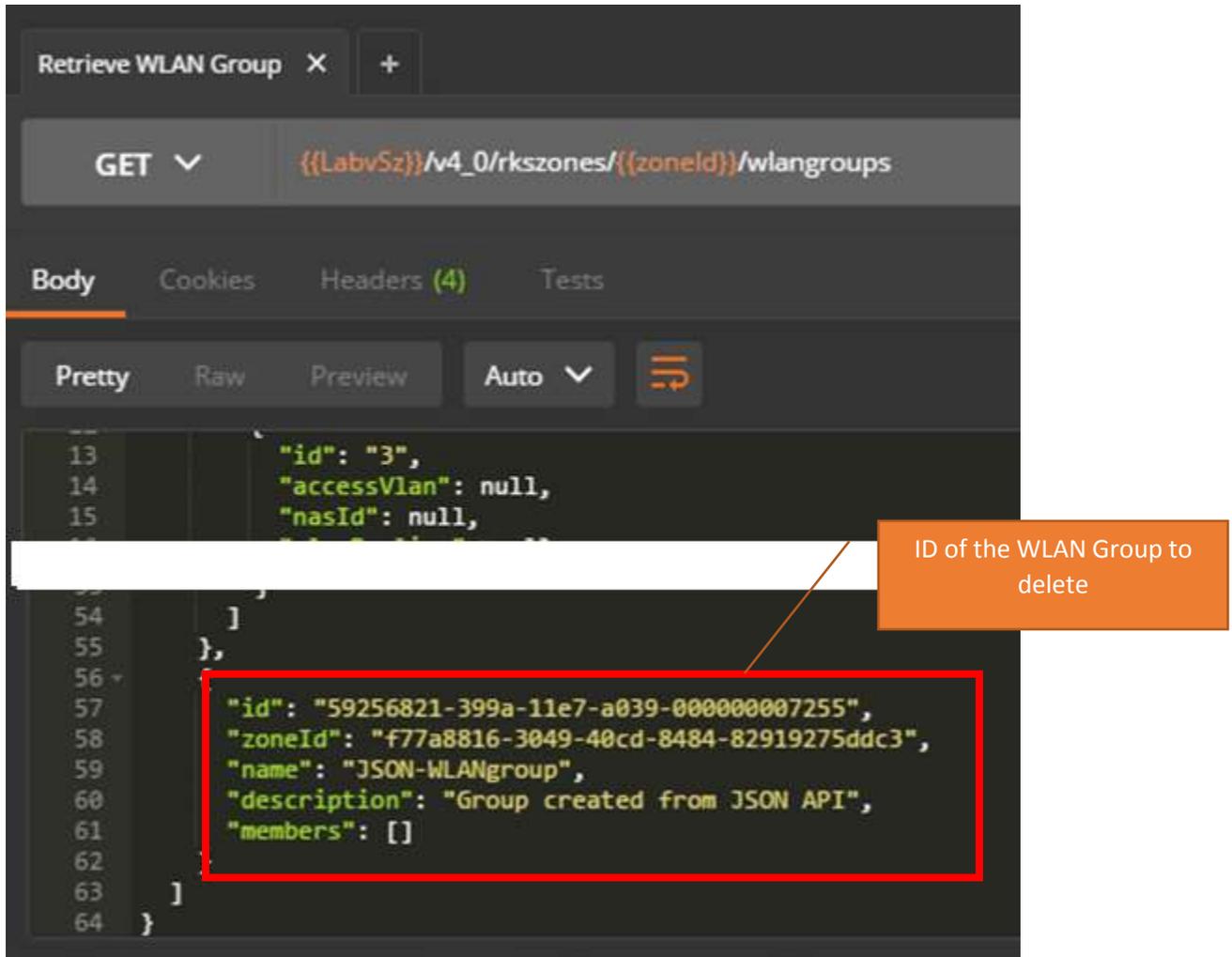
 An annotation "ID for new WLAN Group" points to the value of the "id" field.

Delete WLAN Group API:

Now we are going to create an API to delete a WLAN Group on the controller.

For this call Use DELETE and add the destination URL plus the location (/v4_0/rkszones/{zoneId}/wlangroups/{id}). We will use the zone ID from the previous example and the ID of the WLAN Group we want to delete. This can be found from a retrieve WLAN List API. Remember to add the header information as above and Save AS when you are done.

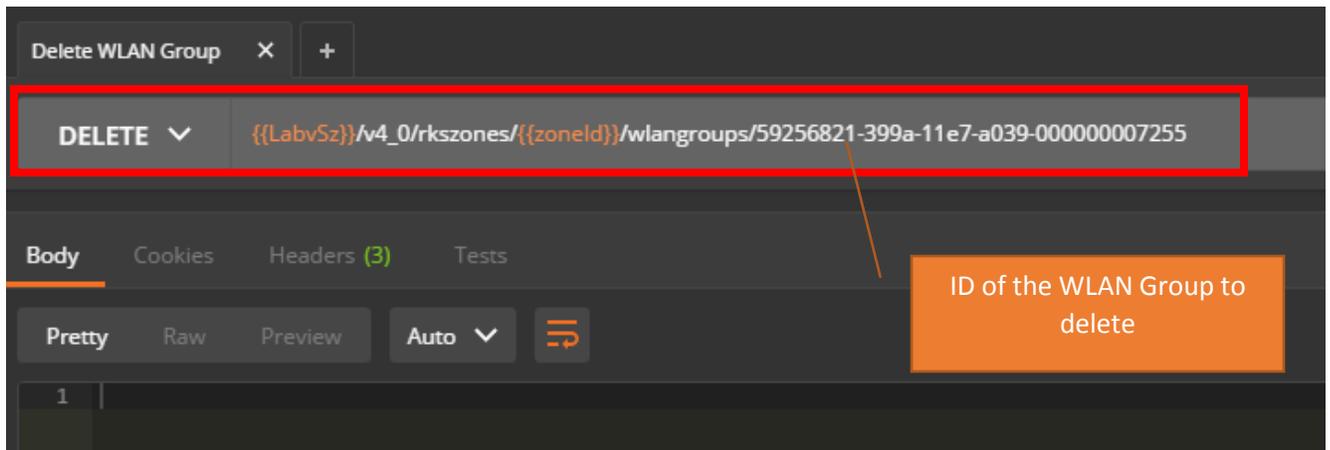
Below is the output from the Retrieve WLAN Group List: Total Count is 2 for my example.



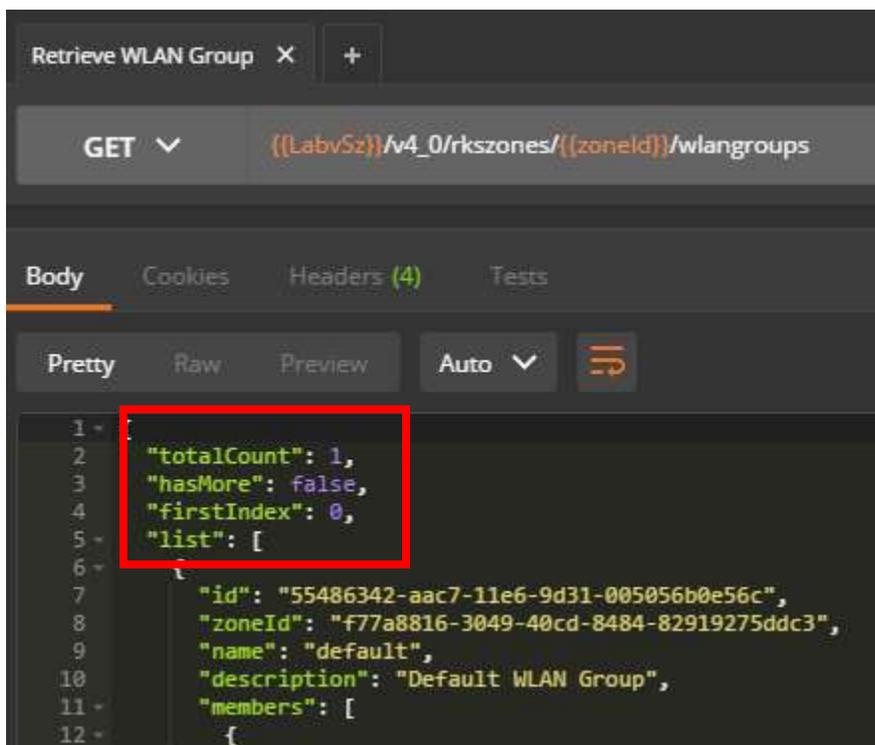
```
Retrieve WLAN Group X +
GET {{LabvSz}}/v4_0/rkszones/{{zoneId}}/wlangroups
Body Cookies Headers (4) Tests
Pretty Raw Preview Auto
13   "id": "3",
14   "accessVlan": null,
15   "nasId": null,
16   "description": "Group created from JSON API",
17   "members": []
18 }
19
20
21
22
23
24 ]
25 },
26
27 "id": "59256821-399a-11e7-a039-000000007255",
28 "zoneId": "f77a8816-3049-40cd-8484-82919275ddc3",
29 "name": "JSON-WLANgroup",
30 "description": "Group created from JSON API",
31 "members": []
32
33 ]
34 }
35 }
```

ID of the WLAN Group to delete

Below is the command and sample output.

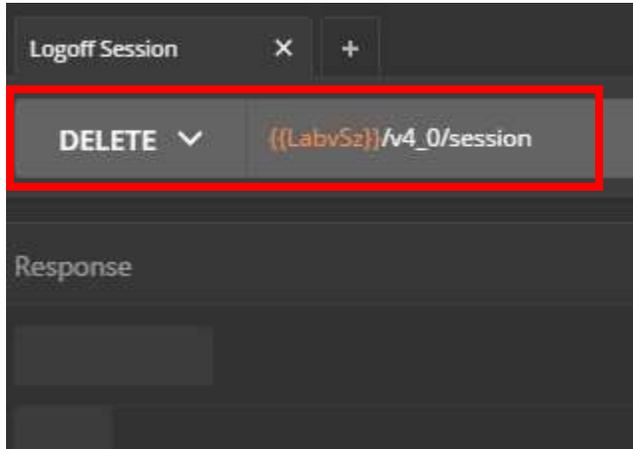


Now we run the WLAN Group Retrieve API again and the group has been deleted. Total Count is now one.



Create a Logout API:

When you are finished, we need to logout of the controller. For this call Use DELETE and add the destination URL plus the location (/v4_0/session). Remember to add the header information as above and Save AS when you are done.



This is only the beginning of the APIs you can create. Please refer to the API Guide for reference.