

Cisco CMX REST API Getting Started Guide

- Overview of Cisco CMX Mobility Services, on page 1
- Support for REST APIs, on page 1
- Interacting with the REST API, on page 2
- Related Documentation, on page 13

Overview of Cisco CMX Mobility Services

Using CMX Mobility Services, developer may leverage device location via Wi-Fi and device MAC address. Customer experiences may be improved by providing coupons, promotions, and other push notifications to devices. Additionally, various client location based service solutions may be developed providing many useful applications for use by the end device user as well as the venue operator.

This solution is an exciting suite of mobile software solutions that detect, connect, and engage with mobile devices operating in a Wi-Fi field. The individual mobile software offerings work together to create a total solution that you can configure to the benefit of your clients and their end users in a very wide variety of real world situations.

The CMX Mobility Services Restful API allows you to use them across languages, platforms, and frameworks. Using the APIs, you can develop application solutions that will use real-time intelligence gathered from your Wi-Fi network to enable people and their devices to interact more effectively through real-time contextual information such as location, temperature, availability of a user, or mobile device asset.

This guide provides information about the CMX Mobility Services Restful API. For more information about the Cisco CMX Mobility Application, see http://www.cisco.com/c/en/us/td/docs/wireless/mse/8-0/CMX-Connect-and-Engage-Mobile-SDK/guide/Cisco-CMX-ConnectEngage-Mobile-SDK-Config-Guide.html.



Note

To view information related to all CMX Mobility Services features and guides, visit the Cisco DevNet Site.

Support for REST APIs

The Cisco Connect Mobile Experiences (Cisco CMX) provides simplified Representational State Transfer (REST) APIs for use in various ways. The advantages of RESTful APIs are:

• Based on the stateless REST architecture.

- The information is delivered over the well-understood HTTP/HTTPS protocol.
- Simplified design and easy to understand Resource URIs. These are usually self-explanatory.
- Standardized libraries allow content negotiation capabilities by automatically sending the response in the user requested format.
- Gives us a chance to simplify our data model, deprecate the older non scalable APIs, remove hierarchy
 dependency and do other enhancements.

The REST API uses the following HTTP methods:

- GET—The HTTP method that retrieves a representation of a resource.
- **PUT**—The HTTP method that stores an entity at a URI. This method helps to create a new entity or update existing entities.
- **POST**—The HTTP method that requests the resource at the URI to perform some action with the provided entity.
- DELETE—The HTTP method that requests to remove a resource.

The Cisco CMX provides updated documents specific to the REST API. The documentation provide the name, methods, and structure of the REST API query. For more information about REST API documentation, see http://mse-ip-address/apidocs/.

The REST API documentation is divided into five logical categories based on the specific functionality. The APIs are categorized as:

- Configuration API—APIs to configure the Cisco CMX programmatically.
- Location API—APIs to retrieve client location data programmatically from the Cisco CMX.
- Analytics API—APIs to retrieve Analytics data programmatically from the Cisco CMX.
- Connect API—APIs to find user session information.
- Presence API—APIs to find visitor presence data.

Interacting with the REST API

To send queries to the Cisco MSE, you require a username and password. These credentials are required to create a base64 encoded string. After successful authentication, request are sent using the HTTP method (GET, PUT, POST, DELETE).

The following is the workflow for REST API:

- 1. Authentication is sent from the client side, before initiating the request using the Authorization header.
- 2. Combine the username and password credentials to form a sting " username:password".
- **3.** Use Base64 to encode the resulting string literal.
- 4. Add an authorization method, a space and the staring "Basic" before the encoded string.

For example, if the user agent uses 'MyUsername' as username and 'MyPassword as password then the header is formed as follows:

- Authorization: Basic QxhZGluOnNlc2FtIG9wZW4=
- Authentication is sent to the root URI to reach (For example: https://<mseip>/api/contextaware)

A Python 2.7.x sample to create Base64 encoded authentication string (see the below image):

```
import urllib2

/
mse_user = 'username'  ##MSE Username
mse_pass = ;password'  ##MSE Password

password_manager = urllib2.HTTPPasswordMgrWithDefaultrealm()
password_manager.add_password(
    None, 'https://ip-address-of-mse', mse_user, mse_pass ##IP Address or resolvable
hostname
)
auth_handler = urllib2.HTTPBasicAuthHandler(password_manager)
opener = urllib2.build_opener(auth_handler)
urllib2.install_opener(opener)
```

There are three ways to interact with the REST API:

- Try it method
- REST Client Plugin (Supported in browsers Google Chrome or Mozilla Firefox)
- Using Programming Language

Using Try It Method

Cisco CMX is packaged with a REST API documentation, which can be accessed through a browser. Cisco CMX REST API documentation page is hosted on the server. For most of the REST APIs, there is a **Try It**

<i>http://yourcmxIP/apidocs</i> . The Cisco CMX Home page for apidocs is displayed.						
Click View Details corresponding to any of the available service names displayed.						
The individual methods are grouped under broad API categories.						
Enter the Username and Password to access the APIs.						
Figuro 1: Ci	co CMX C	redentials				

An administrator can configure the username and password for accessing the REST APIs.

Step 4 Select an API method to view the list of methods under this category.

Figure 2: API Methods

ocation API					
oggle All Endpoints	Toggle All Methods				
Active Clients AP				List Methods	Expand Methods
GET This API re	turns active clients count /api/location/v2/clients	s/count			
GET This API re	turns all clients /api/location/v2/clients				
Tags Information	API			List Methods	Expand Methods
GET This API re	turns active tags count /api/location/v1/tags/cou	int			
GET This API re	turns all tags /api/iocation/v1/tags				
GET This API re	turns tag by macaddress /api/location/v1/tags/s	macaddress			
Parameter	Value	Туре	Location	Description	
Farannever		String	pathReplace	Tag Macaddress	

- **Step 5** Click the individual methods to expand them, or click the **Expand Methods** button on the right to expand and view all the methods under the selected method.
 - **Note** Each API method is provided with a URL. You can use this URL to access the corresponding API within your code. Alternatively, you can use it on third-party REST clients such as POSTMAN or Advanced REST Client.
- **Step 6** Click **Try It** corresponding to the selected API.

Figure 3: Try It Feature

GET			
This AF	PI returns all notification subscriptions		652
Try it!)		354652

The API details are displayed. You can view the call details, response codes, response headers, and response body.

Step 7 Verify the output.

Figure 4: Try It Output

Notification subscription API	List Methods	Expand Methods
Get all notification subscriptions /api/config/v1/notifications		
his API returns all notification subscriptions		
Try 81 Clear results		
al		
IIEUCONF/IBUCAK/opi/config/v1/notifications		
esponse Code		
200		
esponse Headers		
6		
"x-total-execution-time": "0", "access-control-allow-origin": "",		
"access-control-allow-methods": "GET, POST, DELETE, PUT",		
"access-control-allow-headers": "Content-Type". "cmx+token": "R9CmosEGqlirqG/g6Huqbkeew+4ZvSdtWi/vIDg1e2k=",		
content-type", "application/json",		
"content-length": "6566"		
esponse Body Select body		
"receivers" [-
1		
"uri": "http://notifications2.sand.emsp.xyz:80/mse10/nortechtest", "messageFormat": "JSON",		
"gos"; "AT_MOST_ONCE"		
)		
1		
1		100
]. "enabled": true.		
"enableMacScrambling": false,		
"notificationType": "Association"		
1.		
"name": "Magen test", "userid": "admin",		
"rules": [
1		
"conditions": []		
"subscribers" [
"receivers": [

The following parameters are displayed in the output:

- Call—Displays the structure of the query. To test the syntax, copy and paste the query into any web browser.
- Response Code—Provides information related to query. This indicates whether the query was successful or not.
- Response Headers—Displays the requested information.
- Response Body—Displays the requested information. This parameter helps to understand the PUT form.

Tip

• The Response Body is used to understand the PUT form.

• The Call details

Using Third Party REST Clients

To test the structure of API calls and queries and to Cisco CMX, use the REST Client plugins for the latest versions of the Web browsers, Google Chrome and Mozilla Firefox. For example, Advanced REST Client, Postman, and REST Client. The REST Client used is a personal preference.

For the following examples advanced REST Client plugin for Chrome is used.



Note For more information about using the using the REST Clients, see plugin and application instructions for the REST client of your preference.

Example: Notification Definition (PUT)

Figure 5: Notification Definition (PUT)

GET POST PUT PATCH DELETE HEAD OPTIONS Other Rev Form Headers workorization: Basic YWRIzeRH605duYmVUDBRUx Rev Form Files (#) Payload Encode payload Decode payload: Decode payload: Tooming "unersense", "Lise", "Inter "Joon", "Incation-update.deviceType == (liest" IB, "subscribers", []*receivers", []*receivers", []*receivers", []*receivers", []*receivers", []*receivers", []*receivers", []*receivers", []*receivers, []*rec	http://000300	02206.202/ap	/config/v1/notif	cations/	
	GET OPC	ST .PUT	PATCH	ELETE HEAD OPTIONS Other	
Raw Form Files (0) Payload Encode payload Decode payload Encode payload Incode payload man="semanest"_summ", "semant", "semant", "semant", "semant and the semant", "semant and the semant", "semant and the semant and the semather and the semathe semant and the semathe semant	Raw	Form	Headers		
Encode payload Decode payload Teame*"somenumesus" superful "admin" "rules" ("conditions" ("condition" "locationupdate.deviceType == client"(IB, "subscribers" () "receivers" un" http://323/82288,340.4000", "messageFormat" () SON", "gost "AT_MOST_ONCE" (IB, "enabled" true, "notificationType" "LocationUpdate") pplication/json	thorization: 8	lasic YWRtaW46	0.du/m/UbWUx		
Encode payload Decode payload "name" "somenamesus" "userfd" "admin" "vules" ["conditions" [l'condition" "locationupdate.deviceType == client"[]B, "subscribers" [] "receivers" "un" "http://320/82288(380-8000", "messageFormat"; "JSON", "gos" "AT_MOST_ONCE"[]E, "enabled" true, "notificationType" "LocationUpdate"[] "pplication/json " Set "Content-Type" header to overwrite this value.					
Encode payload Decode payload "name" "somenamesus" "userfd" "admin" "vules" ["conditions" [l'condition" "locationupdate.deviceType == client"[]B, "subscribers" [] "receivers" "un" "http://320/82288(380-8000", "messageFormat"; "JSON", "gos" "AT_MOST_ONCE"[]E, "enabled" true, "notificationType" "LocationUpdate"[] "pplication/json " Set "Content-Type" header to overwrite this value.					
Excede payload Decode payload name"'somenamesus", 'userd" "admin", "roles" ("conditions" ("conditions" "locationspdate.deviceType == client"((B, 'subscribers') ("receivers") un" http://J321/J32283(38) 5000", 'messageFormat"; 'JSON', 'goo', 'AT_MOST_ONCE"((B, 'enabled' true, 'notificationType'' LocationUpdate") pplication/json					
name**Somenamesu***Usend***admin************************************	Raw	Form	Files (0)	Payload	
un" http://328/288/288.288.36000","messageFormat";350N","gos";"AT_MOST_ONCE"[[E],"enabled".true,"notificationType";"LocationUpdate"[] pplication/json					
	un"."http://35	Result, "Alleria	1" admin", "rules 8000", "message	"B'conditions" B'condition" "locationupdate.deviceType == client" 88, "subscribers" B'receivers": 'ormat"; "JSON", "gos", "AT_MOST_ONCE" 88, "enabled" :true, "notificationType"; "LocationUpdate" B	
	pplication/jso	0	* Set	"Content-Type" header to overwrite this value.	
Clear Se					

Example: Notification Definition (GET)

Figure 6: Notification Definition (GET)

Rew	JSON	Response	
Copy to clipb	oard Save as	: file	
user	-O: (
) 	and a second sec		
	} led: true ficationTyp	"LocationUpdate"	JEASE

Cisco CMX REST API Getting Started Guide

notification	ar	🕹 Save	Open
* https://10	XXXXXXXXX		
/api/config/v	Inotifications		90
Query paran	teters Add		
HASH			
	NOST OPUT OPATCH DELETE HEAD OPTIONS Other		
	Form Headers Basic YWR04W65004Y7VUWUx		
		Clear	Send
itatus	200 CK 🧐 Loading time: 64 ms		
Request weaders	User-Agent: Mocille'S 0 (Macintosh: Intel Mac OS X 10_9_5) AppleWeb/01537.36 (XHTML_ like Geoko) Chrome/40.0.2214.81 Balav/537.36 Anterstation: Balav/537.36 Accept-Language: en-US.enge-06.doi:no.ende Accept-Language: en-US.enge-06.doi:no.ende Accept-Language: en-US.enge-06.doi:no.ende Ceaks: user: N_D15201/1271372.com/2012/Nation/12713/22abron/2215/371422036655072% Nov.#72250422632% _pac/AL1150508018.1121773772.com/2012/Nation/122502/Nation/221502012753200007012932% _pac/AL1150508018.1121773772.com/endeLange-33221/NyueFBgbb/000007C104072/N256ap04750/WFg00EBBe/CAM/700007471481V230, cme Loke		
tesponse waders	X-1071A_4XEEUTION-TAME.0 Aureas-Cantrol Allow Onlyn.* Acress-Cantrol Allow Onlyn.* Ancess-Cantrol Allow Hasters: Cantrol Type Bet-Ceakle: cmr.tosan-refAntAfactocktifty/26ft/vizit+7.pyuBanGBm6V/8Ew=;Patter/ Esplice: Tut		

Example: All Clients (GET)

Figure 7: All Clients (GET)

tp://10.0.0.8/	+ +	No er	wironment 🗸
GET	/ http://XEXO3DEAapi/location/V2/clients	Params Send	Save
Body Co	okies Headers (7) Tests	Status: 2	200 OK Time: 61 m
Pretty	Raw Preview JSON Y		Q Save Respons
9	"width": 41.50263,		
10	"height": 9.84252,		
11	"offsetX": 0.		
12	"offsetY": 0.		
13	"unit": "FEET"		
14	},		
15 -	"image": {		
16	"imageName": "domain_0_1454937815642.jpg",		
17	"zoomLevel": 4,		
18	"width": 910.		
19	"height": 853.		
20	"size": 910,		
21	"maxResolution": 4.		
22	"colorDepth": 8		
23	}		
24	},		
25 -	"mapCoordinate": {		
26	"x": 38.39772,		
27	"y": 23.597885,		
28	"z": 0.		
29	"unit": "FEET"		
30	3.		
31	"currentlyTracked": true,		
32	"confidenceFactor": 48,		
33 -	"statistics": {		
34	"currentServerTime": "2016-06-21T13:30:11.343+0200",		
35	"firstLocatedTime": "2016-06-21113:54:55.400+0200",		
36	"lastLocatedTime": "2016-06-21T13:21:51.398+0200",		
37 -	"maxDetectedRssi": {		
38	"apMacAddress": "1c:6a:7a:68:8c:e0",		
39	"band": "IEEE_802_11_8",		
40	"slot": 0,		
40	"rssi": -50,		
42	"antennaIndex": 0.		
43	"lastHeardInSeconds": 1		
40	tustneurumsetonus : 1		

Example: Client History by MAC Address (GET)

Figure 8: Client History by MAC Address (GET)

get $ \checkmark $	http://30(8)0(8/api/loca	ition/v2/clients?macAddress	=28:5a:eb:38:2b:5b		Params Ser	nd 💙 Save 🔅
Authorization •	Headers (1) Body	Pre-request Script	Tests			Generate C
Туре		Basic Auth	~		Clear	Update Request
Username		admin		The authorization header will be generated and added as a custom header		
Password				Save helper data to request		
Body Cookies	Headers (7) Te	Show Password			Sta	tus: 200 OK Time: 46 r
Pretty Raw	Preview JSON	~ 5			Q] Q. Save Respon
26 "x 27 "y 28 "z	pCoordinate": { x": 40.782036, y": 25.887224, z": 0,					
30 }, 31 "cur 32 "con	unit": "FEET" rrentlyTracked": true nfidenceFactor": 16,	2,				
34 "c 35 "f	firstLocatedTime": "	2016-07-05T12:32:24.1 2016-07-05T08:33:59.01 216-07-05T12:29:03.206	1+0200",			
37 • "# 38 39	<pre>maxDetectedRssi": { "apMacAddress": "1c: "band": "IEEE_802_11</pre>	:6a:7a:68:8c:e0",				
41 42	"slot": 1, "rssi": -31, "antennaIndex": 0,					
43	"lastHeardInSeconds"	: 6				

Using Programming Language

To test the structure of API calls, use Python programming language. Python is simple and extensively used for configuration and interaction of resource in the data center environment. Both Python 2.7. and Python 3.4.x are used. The comments in the code specifies the Python version.



Note These examples are simple and formatting and other features are not implemented.



Tip To get the best support for creating applications that access the CMX REST API, join the Cisco Devnet. For more information, see https://developer.cisco.com/site/devnet/home/index.gsp.

Example: Client Location List (GET)

Figure 9: Client Location List (GET)

```
#using python 3.4.x
from http.client import HTTPSConnection
from base64 import b64encode
#Create the https connection
c = HTTPSConnection("173.37.206.202")
#encode as Base64
#decode to ascil (python 3 stores as byte string, we don't want that as we need to pass ascil value for auth)
#userAndPass = b64encode(b"admin:!wnbuTme1").decode("ascil")
usernamepasssword = b64encode(b"admin:!wnbuTme1").decode("ascil")
headers = { 'Authorization' : 'Basic %s' % usernamepasssword}
#connect and ask for resource
c.request('GET', '/api/location/v1/clients/', headers=headers)
#reponse
res = c.getresponse()
data = res.read()
#print the results|
print (data)
```

3: { macAddress: "28:b2:bd:33:71:e2" -mapInfo: { mapHierarchyString: "Richardson_TX_75082>Cisco_Building_5>2250_East_PGBT_First_Floor>Lab_Coverage_Area" floorReffd: "-5970138651793817391" floorReffd: "-5970138651793817391"

```
-floorDimension: {
length: 259
      width: 419
      height: 10
       offsetX: 0
      offsetY: 0
      unit: "FEET"
   3
  -image: {
imageName: *domain_0_1410184557303.png*
       sourceFile: null
       zoomLevel: 0
       width: 0
       height: 0
      size: 0
       maxResolution: 0
      colorDepth: 0
   }
   tagList: [0]
}
mapCoordinate: {
x: 168.54506
   y: 40.6289
   unit: "FEET"
3
currentlyTracked: true
confidenceFactor: 144
statistics: {
    currentServerTime: "2015-01-27T17:41:57.105+0000"
   firstLocatedTime: "2015-01-27717:14:50.903+0000"
   lastLocatedTime: "2015-01-27T17:41:51.654+0000"
  -rssiList: [4]
```

354664

maxDetectedRssi: null	
)	
historyLogReason: null	
geoCoordinate: null	
networkStatus: "ACTIVE"	
changedOn: 1422380511654	
ipAddress: null	
userName: ""	
ssid: ""	
sourceTimestamp: null	
band: "UNKNOWN"	
apMacAddress: ""	
dot11Status: "UNKNOWN"	
manufacturer: "Intel"	
-areaGlobalIdList: [8]	
0: 21	
1: 23	
2: 3	
3: 2	
41 1	
5: 24	
6: 25	
7: 18	
detectingControllers: "173.37.206.31"	
bytesSent: 0	
bytesReceived: 0	
guestUser: false	
84: (
macAddress: "00:ee:bd:al:9f:ab"	
-mapInfo: { mapHierarchyString: "Richardson TX 750	80
floorRefId: "-5970138651793817391"	-
-floorDimension: {	
length: 259	
width: 419	
height: 10	
offsetX: 0	5
offsetY: 0	2
unit: "FEET"	
	0

```
84: {
    macAddress: "00:ee:bd:al:9f:ab"
-mapInfo: {
    mapRierarchyString: "Richardson_TX_75082>Cisco_Building_5>2250_East_PGBT_First_Floor>Lab_Coverage_Area"
    floorRefId: "-5970138651793817391"
    -floorDimension: {
        length: 259
        width: 419
        height: 10
        offsetX: 0
        offsetX: 0
        offsetY: 0
        unit: "FEBT"
    }
    -image: {
        imageName: "domain_0_1410184557303.png"
        sourceFile: null
```

L

Example: Single Client Location Information (GET)

Figure 10: Single Client Location Information (GET)

```
from http.client import HTTPSConnection
from base64 import b64encode
c = HTTPSConnection("173.37.206.202")
usernamepasssword = b64encode(b"admin:!wnbuTmel").decode("ascii")
headers = { 'Authorization' : 'Basic %s' % usernamepasssword}
#connect and ask for resource
c.request('GET', '/api/location/v1/clients/28:b2:bd:33:71:e2', headers=headers)
res = c.getresponse()
data = res.read()
                                                                                                                                                         354668
#print the results
print (data) |
  macAddress: "28:b2:bd:33:71:e2"
 -mapInfo: {
mapHierarchyString: "Richardson_TX_75082>Cisco_Building_5>2250_East_PGBT_First_Floor>Lab_Coverage_Area"
floorRefId: "-5970138651793817391"
     -floorDimension: {
length: 259
width: 419
           height: 10
           offsetX: 0
           offsetY: 0
           unit: "FEET"
     -image: {
    imageName: "domain_0_1410184557303.png"
    sourceFile: null
           zoomLevel: 0
           width: 0
           height: 0
           size: 0
          maxResolution: 0
colorDepth: 0
       tagList: [0]
 -mapCo
       Coordinate: {
x: 168.33615
       y: 45.074123
       unit: "FEET"
  currentlyTracked: true
confidenceFactor: 112
 -statistics: {
currentServerTime: "2015-01-27117:52:53.982+0000"
       firstLocatedTime: "2015-01-27T171515153.98240000"
lastLocatedTime: "2015-01-27T17114150.903+0000"
     -rssilist: [4]
-0: {
    apMecAddress: "20:3a:07:07:6d:b0"
    band: "IEEE_802_11_B"
               slot: 0
                                                                                                                                                         354669
               rssi: -66
               antennaIndex: 0
lastHeardInSeconds: 61
           }
```

\mathcal{P}

Tip We recommend to join Cisco Devnet to get the best support for creating application that access the Cisco CMX REST APIs. For more information, see https://developer.cisco.com/site/devnet/home/index.gsp.

Example: Location Setup (POST)

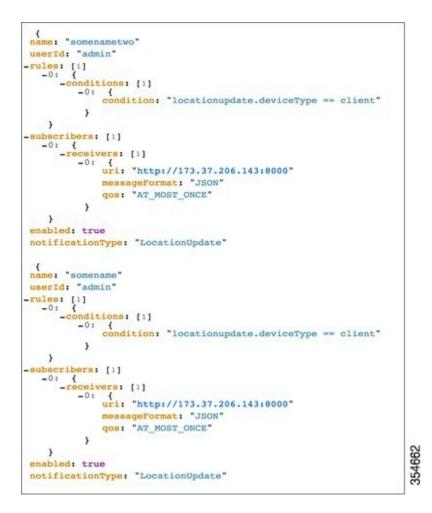
Returns 204. No Content.

Example: Notification Subscription (GET)

Cisco MSE can send a real-time stream of all the activity for all clients to a destination. These notifications are a superset of the location update events. Use the PUT and GET APIs to enable the notification feature.

Figure 11: Notification Subscription (GET)





Example: Notification Subscription (PUT)

Returns 204. No Content.

Related Documentation

The *Cisco Connected Mobile Experiences REST API Guide, Release 10.2*, list all REST APIs for the Cisco CMX solution. For more information, see https://www.cisco.com/c/en/us/td/docs/wireless/mse/10-2/api/b_ cmx_102_api_reference.html.